aws

ユーザーガイド

AWS Elemental MediaStore

AWS Elemental MediaStore: ユーザーガイド

Table of Contents

サポート終了通知: 2025 年 11 月 13 日、 AWS は AWS Elemental MediaStore のサポートを終了します。2025 年 11 月 13 日以降、MediaStore コンソールまたは MediaStore リソースにアクセスできなくなります。詳細については、こちらのブログ記事をご覧ください。

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。

AWS Elemental MediaStore とは

AWS Elemental MediaStore は、ライブ配信に必要な高パフォーマンスと即時の整合性を実現する、動画配信およびストレージサービスです。MediaStore では、動画アセットをコンテナのオブジェクトとして管理し、信頼できるクラウドベースのメディアワークフローを構築できます。

サービスを使用するには、オブジェクトをエンコーダーやデータフィードなどのソースから MediaStore で作成したコンテナにアップロードします。

MediaStore は、断片化された動画ファイルを保存するために、厳密な整合性、低レイテンシーの読み取りと書き込み、および同時リクエストの大量処理が要求される場合に最適です。動画のライブストリーミングを配信しない場合は、代わりに Amazon Simple Storage Service (Amazon S3) を使用することをお勧めします。

トピック

AWS Elemental MediaStore の概念と用語

ARN

Amazon リソースネーム。

[Body] (本文)

オブジェクトにアップロードするデータ。

(バイト) 範囲

対象のオブジェクトデータのサブセット。詳細については、HTTP 仕様の「range (範囲)」を参照してください。

コンテナ

オブジェクトを保持する名前空間。コンテナのエンドポイントを使用してオブジェクトの書き込みと取得、アクセスポリシーのアタッチを行うことができます。

Endpoint

MediaStore サービスのエントリポイントであり、HTTPS ルート URL として提供されます。

ETag

エンティティタグ。オブジェクトデータのハッシュです。

フォルダ

コンテナの区分。フォルダはオブジェクトおよび他のフォルダを保持できます。

項目

オブジェクトとフォルダを指す用語。

オブジェクト

アセット。Amazon S3 オブジェクトに似ています。オブジェクトは、MediaStore に保存される基本エンティティです。すべてのファイルタイプがサポートされます。

配信サービス

MediaStore は、メディアコンテンツ配信の起点であるため、配信サービスと見なされます。

パス

オブジェクトまたはフォルダの一意の識別子。コンテナ内の位置を示します。

パーツ

オブジェクトのデータ (チャンク) のサブセット。

ポリシー

IAM ポリシー。

リソース

操作可能な AWS のエンティティ。各 AWS リソースには、一意の識別子として機能する Amazon リソースネーム (ARN) が割り当てられています。MediaStore では、これはリソースとその ARN 形式です。

- コンテナ: `aws:mediastore:region:account-id:container/:containerName`

関連サービス

- Amazon CloudFront は、データやビデオを視聴者に安全に配信するグローバルコンテンツ配信ネットワーク (CDN) サービスです。CloudFront を使用すると、最大限のパフォーマンスでコンテ

ンツを配信できます。詳細については、「Amazon CloudFront デベロッパーガイド」を参照してください。

- AWS CloudFormation は、 AWS リソースのモデル化とセットアップに役立つサービスです。必要なすべてのリソース (MediaStore コンテナなど) AWS を記述するテンプレートを作成し、 AWS CloudFormation がそれらのリソースのプロビジョニングと設定を行います。 AWS リソースを個別に作成して設定し、何が何に依存しているかを把握する必要はありません。 はこれらをすべて AWS CloudFormation 処理します。詳細については、「AWS CloudFormation ユーザーガイド」を参照してください。
- AWS CloudTrail は、AWS マネジメントコンソールによる呼び出しやその他のサービスなど AWS CLI、アカウントの CloudTrail API に対する呼び出しをモニタリングできるサービスです。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 CloudWatch は、 AWS クラウドリソースとユーザーが実行するアプリケーションのモニタリングサービスです AWS。CloudWatch Events を使用して、MediaStore のコンテナやオブジェクトのステータスの変化を追跡します。詳細については、「Amazon CloudWatch のドキュメント」を参照してください。
- AWS Identity and Access Management (IAM) は、ユーザーの AWS リソースへのアクセスを安全に制御するのに役立つウェブサービスです。IAM を使用して、どのユーザーが AWS リソースを使用できるかを制御し (認証)、さらに、どのリソースをユーザーがどのように使用できるかを制御します (認可)。詳細については、「AWS Elemental MediaStore のセットアップ」を参照してください。
- Amazon Simple Storage Service (Amazon S3) はオブジェクトストレージであり、任意の場所の任意の量のデータを保存および取得するように構築されています。詳細については、Amazon S3 のドキュメント を参照してください。

AWS Elemental MediaStore へのアクセス

次のいずれかの方法で MediaStore にアクセスできます。

- AWS マネジメントコンソール - このガイドの手順では、AWS マネジメントコンソールを使用して MediaStore のタスクを実行する方法について説明しています。コンソールを使用して MediaStore にアクセスするには、次のようにします。

```
https://<region>.console.aws.amazon.com/mediastore/home
```

- AWS Command Line Interface - 詳細については、「AWS Command Line Interface ユーザーガイド」を参照してください。CLI エンドポイントを使用して MediaStore にアクセスするには、次のようにします。

```
aws mediastore
```

- MediaStore API – SDK が提供されていないプログラミング言語を使用している場合、API アクションの情報と API リクエストの作成方法については、「AWS Elemental MediaStore API リファレンス」を参照してください。REST API エンドポイントを使用して MediaStore にアクセスするには、次のようにします。

```
https://mediastore.<region>.amazonaws.com
```

- AWS SDK – AWS によって SDK が提供されているプログラミング言語を使用している場合は、SDK を使用して MediaStore にアクセスできます。SDK では、認証を簡素化し、開発環境と容易に統合して、MediaStore のコマンドに簡単にアクセスできます。詳細については、「Amazon ウェブ サービスのツール」を参照してください。
- AWS Tools for Windows PowerShell - 詳細については、「AWS Tools for Windows PowerShell ユーザーガイド」を参照してください。

AWS Elemental MediaStore の料金

他の AWS 製品と同様に、MediaStore を使用するための契約や最低契約金はありません。コンテンツをサービスに取り込んだときに 1 GB あたりの取り込み料金が発生し、コンテンツをサービスに保存したときに 1 GB あたりの月額料金が発生します。料金の詳細については、「AWS Elemental MediaStore の料金」を参照してください。

AWS Elemental MediaStore のリージョンとエンドポイント

アプリケーションのデータレイテンシーを減らすため、MediaStore ではリージョンのエンドポイントからリクエストを実行できます。

```
https://mediastore.<region>.amazonaws.com
```

MediaStore を使用できる AWS リージョンの完全なリストを表示するには、「AWS 全般のリファレンス」の「AWS Elemental MediaStore エンドポイントとクォータ」を参照してください。

AWS Elemental MediaStore のセットアップ

このセクションでは、AWS Elemental MediaStore にアクセスするユーザーの設定に必要なステップを詳しく説明します。MediaStore 向けの Identity and Access Management に関する背景と追加情報については、「AWS Elemental MediaStore でのアイデンティティ管理とアクセス管理」を参照してください。

AWS Elemental MediaStore の使用を開始するには、以下のステップを完了します。

トピック

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、通話呼び出し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するように求められます。

   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、 のセキュリティを確保し 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 Management Consoleとして にサインインします。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の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「権限設定を作成する」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「グループの結合」を参照してください。

AWS Elemental MediaStore の開始方法

この「開始方法」チュートリアルでは、AWS Elemental MediaStore を使用してコンテナを作成し、オブジェクトをアップロードする方法について説明します。

トピック

ステップ 1: AWS Elemental MediaStore にアクセスする

AWS アカウントをセットアップして ユーザーおよびロールを作成したら、AWS Elemental MediaStore のコンソールにサインインします。

AWS Elemental MediaStore にアクセスするには

- にサインイン AWS Management Console し、https://console.aws.amazon.com/mediastore/ で MediaStore コンソールを開きます。

> **Note**
>
> ログインには、このアカウントで作成した IAM 認証情報のいずれかを使用できます。IAM 認証情報の作成については、「AWS Elemental MediaStore のセットアップ」を参照してください。

ステップ 2: コンテナを作成する

AWS Elemental MediaStore のコンテナを使用してフォルダとオブジェクトを保存します。コンテナを使用して関連するオブジェクトをグループ化できます。ファイルシステムでディレクトリを使用してファイルをグループ化するのと同じ方法です。コンテナの作成時に料金は発生しません。オブジェクトをコンテナにアプロードしたときにのみ料金が発生します。

コンテナを作成するには

1. [Containers (コンテナ)] ページで、[Create container (コンテナの作成)] を選択します。
2. [Container name] (コンテナ名) に、コンテナの名前を入力します。詳細については、「コンテナ名に関するルール」を参照してください。
3. [コンテナの作成] を選択します。AWS Elemental MediaStore によって、コンテナのリストに新しいコンテナが追加されます。コンテナの最初のステータスである [Creating (作成中)] が [Active (アクティブ)] に変わります。

ステップ 3: オブジェクトをアップロードする

オブジェクト (最大 25 MB/オブジェクト) をコンテナ、またはコンテナ内のフォルダにアップロードします。オブジェクトをフォルダにアップロードするには、フォルダへのパスを指定します。フォルダが既に存在する場合、AWS Elemental MediaStore はそのフォルダにオブジェクトを保存します。フォルダが存在しない場合は、フォルダが自動的に作成されて、そのフォルダにオブジェクトが保存されます。

> **Note**
>
> オブジェクトのファイル名には、文字、数字、ピリオド (.)、アンダースコア (_)、チルダ (~)、およびハイフン (-) のみを使用できます。

オブジェクトをアップロードするには

1. [Containers (コンテナ)] ページで、先ほど作成したコンテナの名前を選択します。コンテナの詳細ページが表示されます。
2. [Upload object (オブジェクトのアップロード)] を選択します。
3. [Target path (ターゲットのパス)] に、フォルダへのパスを入力します 例えば、`premium/canada`。パスのフォルダがまだ存在しない場合は、AWS Elemental MediaStore によって自動的に作成されます。
4. [オブジェクト] で、[参照] を選択します。
5. 適切なフォルダに移動し、アップロードする 1 つのオブジェクトを選択します。
6. [Open (開く)]、[Upload (アップロード)] の順に選択します。

ステップ 4: オブジェクトにアクセスする

オブジェクトを指定先のエンドポイントにダウンロードできます。

1. [Containers (コンテナ)] ページで、ダウンロードするオブジェクトが含まれているコンテナの名前を選択します。
2. ダウンロードするオブジェクトがサブフォルダ内にある場合は、オブジェクトが表示されるまで繰り返しフォルダ名を選択します。
3. オブジェクトの名前を選択します。
4. オブジェクトの詳細ページで、[Download (ダウンロード)] を選択します。

AWS Elemental MediaStore のコンテナ

MediaStore のコンテナを使用してフォルダとオブジェクトを保存します。コンテナを使用して関連するオブジェクトをグループ化できます。ファイルシステムでディレクトリを使用してファイルをグループ化するのと同じ方法です。コンテナの作成時に料金は発生しません。オブジェクトをコンテナにアプロードしたときにのみ料金が発生します。料金の詳細については、「AWS Elemental MediaStore 料金」を参照してください。

トピック

コンテナ名に関するルール

コンテナの名前を選択するときは、以下の点に留意してください。

- 名前は現在の AWS リージョンの現在のアカウント内で一意である必要があります。
- 名前には大文字、小文字、数字、およびアンダースコア (_) を使用できます。
- 名前は 1～255 文字の長さである必要があります。
- 名前では、大文字と小文字が区別されます。たとえば、`myContainer` コンテナと `mycontainer` フォルダは一意の名前であるため、一緒に使用できます。
- コンテナを作成した後に名前を変更することはできません。

コンテナの作成

AWS アカウントごとに最大 100 個のコンテナを作成できます。コンテナ内で 10 レベルを超えて入れ子にしない限り、無制限の数のフォルダを作成できます。さらに、各コンテナに必要な数のオブジェクトをアップロードできます。

> **Tip**
>
> AWS CloudFormation テンプレートを使用してコンテナを自動的に作成することもできます。 AWS CloudFormation テンプレートは 5 つの API アクションのデータを管理し、コンテナの作成、アクセスのログ記録の設定を追加、デフォルトのコンテナポリシーの更新、Cross-Origin Resource Sharing (CORS) ポリシーの追加、およびライフサイクルポリシーオブジェクトを追加します。詳細については、「AWS CloudFormation ユーザーガイド」を参照してください。

コンテナを作成するには (コンソール)

1. https://console.aws.amazon.com/mediastore/ で MediaStore コンソールを開きます。
2. [Containers (コンテナ)] ページで、[Create container (コンテナの作成)] を選択します。
3. [Container name (コンテナ名)] に、コンテナの名前を入力します。詳細については、「コンテナ名に関するルール」を参照してください。
4. [コンテナの作成] を選択します。AWS Elemental MediaStore によって、コンテナのリストに新しいコンテナが追加されます。コンテナの最初のステータスである [Creating (作成中)] が [Active (アクティブ)] に変わります。

コンテナを作成するには (AWS CLI)

- で AWS CLI、 `create-container` コマンドを使用します。

```
aws mediastore create-container --container-name ExampleContainer --region us-
west-2
```

戻り値の例を以下に示します。

```
{
    "Container": {
        "AccessLoggingEnabled": false,
        "CreationTime": 1563557265.0,
        "Name": "ExampleContainer",
        "Status": "CREATING",
        "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleContainer"
    }
```

```
}
```

コンテナの詳細の表示

コンテナの詳細には、コンテナポリシー、エンドポイント、ARN、作成日時などが含まれます。

コンテナの詳細を表示するには (コンソール)

1. https://console.aws.amazon.com/mediastore/ で MediaStore コンソールを開きます。
2. [Containers (コンテナ)] ページで、コンテナの名前を選択します。

   コンテナの詳細ページが表示されます。このページは 2 つのセクションに分かれています。

   - [オブジェクト] セクションには、コンテナ内のオブジェクトとフォルダが一覧表示されます。
   - [Container policy (コンテナポリシー)] セクションには、このコンテナに関連付けられているリソースベースのポリシーが表示されます。リソースポリシーの詳細については、「コンテナポリシー」を参照してください。

コンテナの詳細を表示するには (AWS CLI)

- で AWS CLI、 `describe-container` コマンドを使用します。

```
aws mediastore describe-container --container-name ExampleContainer --region us-
west-2
```

戻り値の例を以下に示します。

```
{
    "Container": {
        "CreationTime": 1563558086.0,
        "AccessLoggingEnabled": false,
        "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleContainer",
        "Status": "ACTIVE",
        "Name": "ExampleContainer",
        "Endpoint": "https://aaabbbcccdddee.data.mediastore.us-
west-2.amazonaws.com"
    }
```

```
}
```

コンテナのリストの表示

アカウントに関連付けられているすべてのコンテナのリストを表示できます。

コンテナのリストを表示するには (コンソール)

- https://console.aws.amazon.com/mediastore/ で MediaStore コンソールを開きます。

  [Containers (コンテナ)] ページに、アカウントに関連付けられているすべてのコンテナが一覧表示されます。

コンテナのリストを表示するには (AWS CLI)

- で AWS CLI、`list-containers` コマンドを使用します。

```
aws mediastore list-containers --region us-west-2
```

戻り値の例を以下に示します。

```
{
    "Containers": [
        {
            "CreationTime": 1505317931.0,
            "Endpoint": "https://aaabbbcccdddee.data.mediastore.us-
west-2.amazonaws.com",
            "Status": "ACTIVE",
            "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleLiveDemo",
            "AccessLoggingEnabled": false,
            "Name": "ExampleLiveDemo"
        },
        {
            "CreationTime": 1506528818.0,
            "Endpoint": "https://fffggghhhiiijj.data.mediastore.us-
west-2.amazonaws.com",
            "Status": "ACTIVE",
            "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleContainer",
```

```
            "AccessLoggingEnabled": false,
            "Name": "ExampleContainer"
        }
    ]
}
```

コンテナの削除

コンテナにオブジェクトが含まれていない場合に限り、コンテナを削除できます。

コンテナを削除するには (コンソール)

1. https://console.aws.amazon.com/mediastore/ で MediaStore コンソールを開きます。
2. [Containers (コンテナ)] ページで、コンテナ名の左にあるオプションを選択します。
3. [削除] を選択します。

コンテナを削除するには (AWS CLI)

- で AWS CLI、 `delete-container` コマンドを使用します。

  ```
  aws mediastore delete-container --container-name=ExampleLiveDemo --region us-west-2
  ```

  このコマンドの戻り値はありません。

AWS Elemental MediaStore のポリシー

以下のポリシーを 1 つ以上、AWS Elemental MediaStore コンテナに適用できます。

- コンテナポリシー - コンテナ内のすべてのフォルダとオブジェクトへのアクセス権を設定します。MediaStore ではデフォルトポリシーが設定され、ユーザーはコンテナですべてのMediaStore オペレーションを実行できます。このポリシーは、すべてのオペレーションを HTTPS で実行する必要があることを指定します。コンテナを作成すると、コンテナポリシーを編集できます。
- クロスオリジンリソース共有 (CORS) ポリシー - 特定のドメインのクライアントウェブアプリケーションが別のドメインのリソースと通信することを許可します。MediaStore ではデフォルトの CORS ポリシーは設定されません。
- メトリクスポリシー - MediaStore がメトリクスを Amazon CloudWatch に送信することを許可します。MediaStore ではデフォルトのメトリクスポリシーは設定されません。
- オブジェクトのライフサイクルポリシー - MediaStore コンテナにオブジェクトを保持する期間を制御します。MediaStore ではデフォルトのオブジェクトライフサイクルポリシーは設定されません。

AWS Elemental MediaStore のコンテナポリシー

コンテナごとにリソースベースのポリシーが割り当てられます。このポリシーで、コンテナ内のすべてのフォルダとオブジェクトへのアクセス権限を管理します。すべての新しいコンテナに自動的に割り当てられるデフォルトポリシーでは、コンテナに対するすべての AWS Elemental MediaStore オペレーションへのアクセスが許可されます。このアクセスの条件として、オペレーションでは HTTPS を使用する必要があります。コンテナを作成したら、そのコンテナにアタッチされているポリシーを編集することができます。

また、オブジェクトのライフサイクルポリシーを指定して、コンテナ内のオブジェクトの有効期限を管理することもできます。指定したオブジェクトが有効期限に達すると、コンテナからオブジェクトが自動的に削除されます。

トピック

- コンテナポリシーを表示する
- コンテナポリシーを編集する
- コンテナポリシーの例

コンテナポリシーを表示する

コンソールまたは を使用して AWS CLI 、コンテナのリソースベースのポリシーを表示できます。

コンテナポリシーを表示するには (コンソール)

1. https://console.aws.amazon.com/mediastore/ で MediaStore コンソールを開きます。
2. [Containers (コンテナ)] ページで、コンテナの名前を選択します。

   コンテナの詳細ページが表示されます。ポリシーは [Container policy (コンテナポリシー)] セクションに表示されます。

コンテナポリシーを表示するには (AWS CLI)

- で AWS CLI、 `get-container-policy` コマンドを使用します。

```
aws mediastore get-container-policy --container-name ExampleLiveDemo --region us-
west-2
```

戻り値の例を以下に示します。

```
{
  "Policy": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Sid": "PublicReadOverHttps",
        "Effect": "Allow",
        "Principal": {
          "AWS": "arn:aws:iam::111122223333:root",
        },
        "Action": [
          "mediastore:GetObject",
          "mediastore:DescribeObject",
        ],
        "Resource": "arn:aws:mediastore:us-
west-2:111122223333:container/ExampleLiveDemo/*",
        "Condition": {
          "Bool": {
            "aws:SecureTransport": "true"
          }
```

```
        }
      }
    ]
  }
}
```

コンテナポリシーを編集する

デフォルトのコンテナポリシーへのアクセス許可を編集するか、新しいポリシーを作成してデフォルトのポリシーと置き換えることができます。新しいポリシーが有効になるまで、最大で 5 分かかります。

コンテナポリシーを編集するには (コンソール)

1. https://console.aws.amazon.com/mediastore/ で MediaStore コンソールを開きます。
2. [Containers (コンテナ)] ページで、コンテナの名前を選択します。
3. [ポリシーの編集] を選択します。異なるアクセス許可を設定する方法の例については、「the section called “コンテナポリシーの例”」を参照してください。
4. 適切な変更を行い、[Save] (保存) を選択します。

コンテナポリシーを編集するには (AWS CLI)

1. コンテナポリシーを定義するファイルを作成します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadOverHttps",
      "Effect": "Allow",
      "Action": ["mediastore:GetObject", "mediastore:DescribeObject"],
      "Principal": "*",
      "Resource": "arn:aws:mediastore:us-
west-2:111122223333:container/ExampleLiveDemo/*",
      "Condition": {
        "Bool": {
            "aws:SecureTransport": "true"
        }
      }
```

```
    }
  ]
}
```

2. で AWS CLI、 `put-container-policy` コマンドを使用します。

```
aws mediastore put-container-policy --container-name ExampleLiveDemo --
policy file://ExampleContainerPolicy.json --region us-west-2
```

このコマンドの戻り値はありません。

コンテナポリシーの例

さまざまなユーザーグループ向けに作成されたコンテナポリシーの例を以下に示します。

トピック

- コンテナポリシーの例: デフォルト
- コンテナポリシーの例: HTTPS 経由のパブリック読み取りアクセス
- コンテナポリシーの例: HTTP または HTTPS 経由のパブリック読み取りアクセス
- コンテナポリシーの例: クロスアカウントの読み取りアクセス (HTTP 対応)
- コンテナポリシーの例: HTTPS 経由のクロスアカウント読み取りアクセス
- コンテナポリシーの例: ロールへのクロスアカウント読み取りアクセス
- コンテナポリシーの例: ロールへのクロスアカウントフルアクセス
- コンテナポリシーの例: 特定の IP アドレスに制限されたアクセス

コンテナポリシーの例: デフォルト

コンテナを作成すると、AWS Elemental MediaStore によって自動的に以下のリソースベースのポリシーが割り当てられます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "MediaStoreFullAccess",
      "Action": [ "mediastore:*" ],
```

```
      "Principal":{
          "AWS" : "arn:aws:iam::<aws_account_number>:root"},
      "Effect": "Allow",
      "Resource": "arn:aws:mediastore:<region>:<owner acct number>:container/<container
 name>/*",
      "Condition": {
        "Bool": { "aws:SecureTransport": "true" }
      }
    }
  ]
}
```

ポリシーはサービスに組み込まれているため、作成する必要はありません。ただし、デフォルトポリシーのアクセス許可が、コンテナに使用するアクセス許可と一致していない場合は、コンテナのポリシーを編集できます。

すべての新しいコンテナに割り当てられるデフォルトポリシーでは、コンテナに対するすべての MediaStore オペレーションへのアクセスが許可されます。このアクセスの条件として、オペレーションでは HTTPS を使用する必要があります。

コンテナポリシーの例: HTTPS 経由のパブリック読み取りアクセス

このポリシー例では、ユーザーが HTTPS リクエストを行ってオブジェクトを取得することができます。これにより、認証済みユーザーや匿名ユーザー (ログインしていないユーザー) など、すべてのユーザーに安全な SSL/TLS 接続を経由した読み取りアクセスが許可されます。ステートメント名は PublicReadOverHttps です。任意のオブジェクト (リソースパスの末尾に * を使用して指定) に対する GetObject オペレーションと DescribeObject オペレーションへのアクセスが許可されます。このアクセスの条件として、オペレーションでは HTTPS を使用する必要があります。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadOverHttps",
      "Effect": "Allow",
      "Action": ["mediastore:GetObject", "mediastore:DescribeObject"],
      "Principal": "*",
      "Resource": "arn:aws:mediastore:<region>:<owner acct number>:container/<container
 name>/*",
      "Condition": {
        "Bool": {
```

```
            "aws:SecureTransport": "true"
          }
        }
      }
    ]
}
```

コンテナポリシーの例: HTTP または HTTPS 経由のパブリック読み取りアクセス

このポリシー例では、任意のオブジェクト (リソースパスの末尾に「*」を使用して指定) に対する GetObject オペレーションと DescribeObject オペレーションへのアクセスが許可されます。これにより、すべての認証済みユーザーや匿名ユーザー (ログインしていないユーザー) など、すべてのユーザーに読み取りアクセスが許可されます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadOverHttpOrHttps",
      "Effect": "Allow",
      "Action": ["mediastore:GetObject", "mediastore:DescribeObject"],
      "Principal": "*",
      "Resource": "arn:aws:mediastore:<region>:<owner acct number>:container/<container
 name>/*",
      "Condition": {
        "Bool": { "aws:SecureTransport": ["true", "false"] }
      }
    }
  ]
}
```

コンテナポリシーの例: クロスアカウントの読み取りアクセス (HTTP 対応)

このポリシー例では、ユーザーが HTTP リクエストを行ってオブジェクトを取得することができます。このアクセスは、クロスアカウントアクセス権限を持つ認証済みユーザーに許可されます。オブジェクトは、SSL/TLS 証明書を持つサーバーでホストされている必要はありません。

```
{
  "Version" : "2012-10-17",
  "Statement" : [ {
    "Sid" : "CrossAccountReadOverHttpOrHttps",
```

```
    "Effect" : "Allow",
    "Principal" : {
      "AWS" : "arn:aws:iam::<other acct number>:root"
    },
    "Action" : [ "mediastore:GetObject", "mediastore:DescribeObject" ],
    "Resource" : "arn:aws:mediastore:<region>:<owner acct number>:container/<container
 name>/*",
    "Condition" : {
      "Bool" : {
        "aws:SecureTransport" : [ "true", "false" ]
      }
    }
  } ]
}
```

コンテナポリシーの例: HTTPS 経由のクロスアカウント読み取りアクセス

このポリシー例では、指定した <other acct number> のルートユーザーによって所有されている任意のオブジェクト (リソースパスの末尾に * で指定) に対する GetObject オペレーションと DescribeObject オペレーションへのアクセスが許可されます。このアクセスの条件として、オペレーションでは HTTPS を使用する必要があります。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CrossAccountReadOverHttps",
      "Effect": "Allow",
      "Action": ["mediastore:GetObject", "mediastore:DescribeObject"],
      "Principal":{
        "AWS": "arn:aws:iam::<other acct number>:root"},
      "Resource": "arn:aws:mediastore:<region>:<owner acct number>:container/<container
 name>/*",
      "Condition": {
        "Bool": {
            "aws:SecureTransport": "true"
        }
      }
    }
  ]
}
```

コンテナポリシーの例: ロールへのクロスアカウント読み取りアクセス

このポリシー例では、<owner acct number> によって所有されている任意のオブジェクト (リソースパスの末尾に * で指定) に対する GetObject オペレーションと DescribeObject オペレーションへのアクセスを許可します。このアクセスは、他のアカウント <other acct number> が <role name> に指定されたロールを引き受けた場合、そのアカウントのすべてのユーザーに許可されます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CrossAccountRoleRead",
      "Effect": "Allow",
      "Action": ["mediastore:GetObject", "mediastore:DescribeObject"],
      "Principal":{
        "AWS": "arn:aws:iam::<other acct number>:role/<role name>"},
      "Resource": "arn:aws:mediastore:<region>:<owner acct number>:container/<container
 name>/*",
    }
  ]
}
```

コンテナポリシーの例: ロールへのクロスアカウントフルアクセス

このポリシー例では、ユーザーが HTTP 経由でログインしていることを条件として、ユーザーにアカウント内のすべてのオブジェクトを更新するクロスアカウントアクセスが許可されます。また、クロスアカウントアクセスでは、指定されたロールを引き受けたアカウントへ HTTP または HTTPS 経由でのオブジェクトの削除、ダウンロード、紹介が許可されます。

- 最初のステートメントは CrossAccountRolePostOverHttps です。これにより、任意のオブジェクトに対する PutObject オペレーションへのアクセスが許可されます。また、このアクセスは、指定したアカウントが <role name> に指定されたロールを引き受けた場合、そのアカウントのすべてのユーザーに許可されます。このアクセスの条件として、オペレーションでは HTTPS を使用する必要があります (この条件は、PutObject へのアクセスを提供する場合に必ず含める必要があります)。

  つまり、クロスアカウントアクセスを持つすべてのプリンシパルが PutObject にアクセスできますが、アクセス方法は HTTPS 経由に限られます。
- 2 番目のステートメントは CrossAccountFullAccessExceptPost です。任意のオブジェクトに対する、PutObject 以外のすべてのオペレーションへのアクセスが許可されます。このアクセ

スは、指定したアカウントが <role name> に指定されたロールを引き受けた場合、そのアカウントのすべてのユーザーに許可されます。このアクセスでは、オペレーションで HTTPS を使用することが条件として要求されません。

つまり、クロスアカウントアクセス権限を持つすべてのアカウントが DeleteObject、GetObject など (ただし、PutObject を除く) にアクセスできます。また、アクセスには HTTP または HTTPS を使用できます。

2 番目のステートメントから PutObject を除外しない場合、ステートメントを有効になりません (PutObject を含める場合は、HTTPS を条件として明示的に設定する必要があるためです)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CrossAccountRolePostOverHttps",
      "Effect": "Allow",
      "Action": "mediastore:PutObject",
      "Principal":{
        "AWS": "arn:aws:iam::<other acct number>:role/<role name>"},
      "Resource": "arn:aws:mediastore:<region>:<owner acct number>:container/<container
 name>/*",
      "Condition": {
        "Bool": {
            "aws:SecureTransport": "true"
        }
      }
    },
    {
      "Sid": "CrossAccountFullAccessExceptPost",
      "Effect": "Allow",
      "NotAction": "mediastore:PutObject",
      "Principal":{
        "AWS": "arn:aws:iam::<other acct number>:role/<role name>"},
      "Resource": "arn:aws:mediastore:<region>:<owner acct number>:container/<container
 name>/*"
    }
  ]
}
```

コンテナポリシーの例: 特定の IP アドレスに制限されたアクセス

このポリシーの例では、指定されたコンテナ内のオブジェクトに対するすべての AWS Elemental MediaStore オペレーションへのアクセスが許可されます。ただし、リクエストは条件で指定された IP アドレス範囲からのリクエストである必要があります。

このステートメントの条件では、198.51.100 を特定します。* の範囲のインターネットプロトコルバージョン 4 (IPv4) のIP アドレスが許可されています。ただし、 198.51.100.188 を除きます。

Condition ブロックでは、IpAddress および NotIpAddress 条件と、aws:SourceIp 条件キーが使用されています。これは AWS 全体を対象とする条件キーです。aws:sourceIpIPv4 値は標準の CIDR 表記を使用します。詳細については、「IAM ユーザーガイド」の「IP アドレス条件演算子」を参照してください。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AccessBySpecificIPAddress",
            "Effect": "Allow",
            "Action": [
                "mediastore:GetObject",
                "mediastore:DescribeObject"
            ],
            "Principal": "*",
            "Resource": "arn:aws:mediastore:<region>:<owner acct number>:container/
<container name>/*",
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": [
                        "198.51.100.0/24"
                    ]
                },
                "NotIpAddress": {
                    "aws:SourceIp": "198.51.100.188/32"
                }
            }
        }
    ]
}
```

AWS Elemental MediaStore のクロスオリジンリソース共有 (CORS) ポリシー

Cross-Origin Resource Sharing (CORS) は、特定のドメインにロードされたクライアントウェブアプリケーションが異なるドメイン内のリソースと通信する方法を定義します。AWS Elemental MediaStore での CORS のサポートにより、MediaStore を使用して機能豊富なクライアント側ウェブアプリケーションを構築し、MediaStore リソースに対するクロスオリジンアクセスを選択的に許可することができるようになりました。

> **Note**
>
> CORS ポリシーのあるコンテナからコンテンツを配信するために Amazon CloudFront を使用している場合、必ず AWS Elemental MediaStore の配信を設定 (CORS をセットアップするためにキャッシュ動作を編集するステップを含む) するようにします。

このセクションでは、CORS の概要を示します。サブトピックでは、AWS Elemental MediaStore コンソールを使用するか、MediaStore REST API と AWS SDK を使用してプログラムで CORS を有効にする方法について説明します。

トピック

CORS ユースケースのシナリオ

CORS のユースケースの例を以下に示します。

- シナリオ 1: LiveVideo という AWS Elemental MediaStore コンテナのライブストリーミング動画を配信するとします。ユーザーは、`http://livevideo.mediastore.ap-`

southeast-2.amazonaws.com などの特定のオリジンから、動画マニフェストエンドポイント www.example.com をロードします。認証されていない GET リクエストと PUT リクエストを介してこのコンテナから発信される動画にアクセスするには、JavaScript ビデオプレーヤーを使用します。通常、ブラウザは JavaScript をブロックしてこれらのリクエストを許可しませんが、コンテナに対して CORS ポリシーを設定することで、www.example.com からのリクエストを明示的に有効化できます。

- シナリオ 2: シナリオ 1 と同じライブストリームを MediaStore コンテナでホストするとします。ただし、任意のオリジンからのリクエストを許可します。ワイルドカード (*) のオリジンを許可し、任意のオリジンのリクエストから動画にアクセスできるように、CORS ポリシーを設定できます。

コンテナへの CORS ポリシーの追加

このセクションでは、クロスオリジンリソース共有 (CORS) 設定を AWS Elemental MediaStore コンテナに追加する方法について説明します。CORS は、特定のドメインにロードされたクライアントウェブアプリケーションが別のドメイン内のリソースと通信できるようにします。

クロスオリジンリクエストを許可するようにコンテナを設定するには、CORS ポリシーをコンテナに追加します。CORS ポリシーでは、コンテナへのアクセスを許可するオリジン、各オリジンでサポートされるオペレーション (HTTP メソッド)、その他のオペレーション固有の情報を識別するルールを定義します。

CORS ポリシーをコンテナに追加すると、コンテナポリシーが (コンテナへのアクセス権限を管理するために) 継続的に適用されます。

CORS ポリシーを更新するには (コンソール)

1. https://console.aws.amazon.com/mediastore/ で MediaStore コンソールを開きます。
2. [Containers (コンテナ)] ページで、CORS ポリシーを作成する対象のコンテナの名前を選択します。

   コンテナの詳細ページが表示されます。
3. [Container CORS policy (コンテナの CORS ポリシー)] セクションで、[Create CORS policy (CORS ポリシーの作成)] を選択します。
4. ポリシーを JSON 形式で挿入し、[Save (保存)] を選択します。

CORS ポリシーを追加するには (AWS CLI)

1. CORS ポリシーを定義するファイルを作成します。

```
[
  {
    "AllowedHeaders": [
      "*"
    ],
    "AllowedMethods": [
      "GET",
      "HEAD"
    ],
    "AllowedOrigins": [
      "*"
    ],
    "MaxAgeSeconds": 3000
  }
]
```

2. で AWS CLI、 `put-cors-policy` コマンドを使用します。

```
aws mediastore put-cors-policy --container-name ExampleContainer --cors-policy
 file://corsPolicy.json --region us-west-2
```

このコマンドの戻り値はありません。

CORS ポリシーの表示

Cross-Origin Resource Sharing (CORS) は、特定のドメインにロードされたクライアントウェブアプリケーションが異なるドメイン内のリソースと通信する方法を定義します。

CORS ポリシーを表示するには (コンソール)

1. https://console.aws.amazon.com/mediastore/ で MediaStore コンソールを開きます。
2. [Containers (コンテナ)] ページで、CORS ポリシーを表示する対象のコンテナの名前を選択します。

   コンテナの詳細ページの [Container CORS policy (コンテナの CORS ポリシー)] セクションに CORS ポリシーが表示されます。

CORS ポリシーを表示するには (AWS CLI)

- で AWS CLI、 `get-cors-policy` コマンドを使用します。

```
aws mediastore get-cors-policy --container-name ExampleContainer --region us-west-2
```

戻り値の例を以下に示します。

```
{
    "CorsPolicy": [
        {
            "AllowedMethods": [
                "GET",
                "HEAD"
            ],
            "MaxAgeSeconds": 3000,
            "AllowedOrigins": [
                "*"
            ],
            "AllowedHeaders": [
                "*"
            ]
        }
    ]
}
```

CORS ポリシーの編集

Cross-Origin Resource Sharing (CORS) は、特定のドメインにロードされたクライアントウェブアプリケーションが異なるドメイン内のリソースと通信する方法を定義します。

CORS ポリシーを編集するには (コンソール)

1. https://console.aws.amazon.com/mediastore/ で MediaStore コンソールを開きます。
2. [Containers (コンテナ)] ページで、CORS ポリシーを編集する対象のコンテナの名前を選択します。

   コンテナの詳細ページが表示されます。
3. [Container CORS policy (コンテナの CORS ポリシー)] セクションで、[Edit CORS policy (CORS ポリシーの編集)] を選択します。

4. ポリシーを変更し、[Save (保存)] を選択します。

CORS ポリシーを編集するには (AWS CLI)

1. 更新された CORS ポリシーを定義するファイルを作成します。

```
[
  {
    "AllowedHeaders": [
      "*"
    ],
    "AllowedMethods": [
      "GET",
      "HEAD"
    ],
    "AllowedOrigins": [
      "https://www.example.com"
    ],
    "MaxAgeSeconds": 3000
  }
]
```

2. で AWS CLI、 `put-cors-policy` コマンドを使用します。

```
aws mediastore put-cors-policy --container-name ExampleContainer --cors-policy
 file://corsPolicy2.json --region us-west-2
```

このコマンドの戻り値はありません。

CORS ポリシーの削除

Cross-Origin Resource Sharing (CORS) は、特定のドメインにロードされたクライアントウェブアプリケーションが異なるドメイン内のリソースと通信する方法を定義します。コンテナから CORS ポリシーを削除すると、クロスオリジンリクエストのアクセス許可が削除されます。

CORS ポリシーを削除するには (コンソール)

1. https://console.aws.amazon.com/mediastore/ で MediaStore コンソールを開きます。
2. [Containers (コンテナ)] ページで、CORS ポリシーを削除する対象のコンテナの名前を選択します。

コンテナの詳細ページが表示されます。

3. [Container CORS policy (コンテナの CORS ポリシー)] セクションで、[Delete CORS policy (CORS ポリシーの削除)] を選択します。
4. [Continue] を選択して確認し、[Save] を選択します。

CORS ポリシーを削除するには (AWS CLI)

- で AWS CLI、 `delete-cors-policy` コマンドを使用します。

```
aws mediastore delete-cors-policy --container-name ExampleContainer --region us-west-2
```

このコマンドの戻り値はありません。

CORS の問題のトラブルシューティング

CORS ポリシーが存在するコンテナにアクセスするときに予期しない動作が発生した場合は、以下のステップに従って問題のトラブルシューティングを行います。

1. CORS ポリシーがコンテナにアタッチされていることを確認します。

   手順については、the section called “CORS ポリシーの表示” を参照してください。
2. 任意のツール (ブラウザの開発者コンソールなど) を使用して、完全なリクエストとレスポンスをキャプチャします。コンテナにアタッチされている CORS ポリシーに、リクエストのデータと一致する少なくとも 1 つの CORS ルールが含まれていることを確認します。

   a. リクエストに `Origin` ヘッダーがあることを確認します。

      このヘッダーがないリクエストは、AWS Elemental MediaStore でクロスオリジンリクエストとして扱われず、CORS レスポンスヘッダーはレスポンスで返送されません。

   b. リクエストの `Origin` ヘッダーが、特定の `AllowedOrigins` の `CORSRule` 要素の少なくとも 1 つと一致していることを確認します。

      `Origin` リクエストヘッダーのスキーム、ホスト、およびポートの値は、`AllowedOrigins` の `CORSRule` と一致する必要があります。たとえば、オリジン `CORSRule` を許可するように `http://www.example.com` を設定すると、リクエストの

https://www.example.com オリジンと http://www.example.com:80 オリジンのいずれも、設定で許可されているオリジンと一致しません。

c. リクエストのメソッド (またはプリフライトリクエストの場合は Access-Control-Request-Method に指定されたメソッド) が、同じ AllowedMethods の CORSRule 要素の 1 つであることを確認します。

d. プリフライトリクエストの場合、それに Access-Control-Request-Headers ヘッダーが含まれているときに、CORSRule に AllowedHeaders ヘッダーのすべての値に対する Access-Control-Request-Headers エントリが含まれていること。

CORS ポリシーの例

クロスオリジンリソース共有 (CORS) ポリシーの例を以下に示します。

トピック

CORS ポリシーの例: 任意のドメインへの読み取りアクセス

次のポリシーでは、AWS Elemental MediaStore コンテナからコンテンツを取得することをすべてのドメインのウェブページに許可します。リクエストには、送信元ドメインからのすべての HTTP ヘッダーが含まれます。このサービスでは、送信元ドメインからの HTTP GET リクエストと HTTP HEAD リクエストにのみ応答します。結果は、新しい結果セットが配信されるまで 3,000 秒間キャッシュされます。

```
[
  {
    "AllowedHeaders": [
      "*"
    ],
    "AllowedMethods": [
      "GET",
      "HEAD"
    ],
    "AllowedOrigins": [
      "*"
    ],
    "MaxAgeSeconds": 3000
```

```
  }
]
```

CORS ポリシーの例: 特定のドメインへの読み取りアクセス

次のポリシーでは、AWS Elemental MediaStore コンテナからコンテンツを取得することを `https://www.example.com` のウェブページに許可します。リクエストには、`https://www.example.com` からのすべての HTTP ヘッダーが含まれます。このサービスでは、`https://www.example.com` からの HTTP GET リクエストと HTTP HEAD リクエストにのみ応答します。結果は、新しい結果セットが配信されるまで 3,000 秒間キャッシュされます。

```
[
  {
    "AllowedHeaders": [
      "*"
    ],
    "AllowedMethods": [
      "GET",
      "HEAD"
    ],
    "AllowedOrigins": [
      "https://www.example.com"
    ],
    "MaxAgeSeconds": 3000
  }
]
```

AWS Elemental MediaStore のオブジェクトのライフサイクルポリシー

オブジェクトのコンテナ内保存期間を管理する、オブジェクトのライフサイクルポリシーをコンテナごとに作成することができます。オブジェクトが指定された有効期限に達すると、AWS Elemental MediaStore によりオブジェクトが削除されます。不要になったオブジェクトを削除して、ストレージコストを削減することもできます。

オブジェクトが特定の経過時間に達した後、MediaStore がオブジェクトを低頻度アクセス (IA) ストレージクラスに移動するように指定することもできます。IA ストレージクラスに保存されているオブジェクトは、標準のストレージクラスに保存されているオブジェクトとはストレージおよび取得の速度が異なります。詳細については、「MediaStore 料金」を参照してください。

オブジェクトのライフサイクルポリシーには、サブフォルダごとにオブジェクトの保持期間を決定するというルールが含まれています (オブジェクトのライフサイクルポリシーをオブジェクトに個別に割り当てることはできません)。1 つのコンテナに対してオブジェクトのライフサイクルポリシーを 1 つ割り当てることができますが、オブジェクトのライフサイクルポリシーごとに最大 10 個のルールを追加できます。詳細については、「オブジェクトのライフサイクルポリシーのコンポーネント」を参照してください。

トピック

オブジェクトのライフサイクルポリシーのコンポーネント

オブジェクトのライフサイクルポリシーでは、AWS Elemental MediaStore コンテナにオブジェクトを保持する期間を管理します。各オブジェクトのライフサイクルポリシーには 1 つ以上のルールがあり、オブジェクトの保持期間を決定します。ルールは、1 つのフォルダ、複数フォルダ、コンテナ全体に適用されます。

1 つのコンテナに対して 1 つのオブジェクトのライフサイクルポリシーをアタッチすることができ、各オブジェクトのライフサイクルポリシーには最大 10 個のルールを含めることができます。オブジェクトのライフサイクルポリシーを個々のオブジェクトに割り当てることはできません。

オブジェクトのライフサイクルポリシーのルール

次の 3 種類のルールを作成できます。

- 一時データ
- DELETE Object
- ライフサイクル移行

一時データ

一時的なデータルールで、数秒以内に有効期限が切れるようにオブジェクトを設定します。このタイプのルールは、ポリシーが有効になった後にコンテナに追加されたオブジェクトにのみ適用されます。MediaStore がコンテナに新しいポリシーを適用するまでに、最大 20 分かかります。

一時データのルールの例は次のようになります。

```
        {
            "definition": {
                "path": [ {"wildcard": "Football/index*.m3u8"} ],
                "seconds_since_create": [
                    {"numeric": [">", 120]}
                ]
            },
            "action": "EXPIRE"
        },
```

一時データのルールには以下の 3 つの部分があります。

- path: 常に wildcard に設定されます。このパートを使用して、削除するオブジェクトを定義します。1 つ以上のワイルドカードを使用できます。これはアスタリスク (*) で表されます。各ワイルドカードは、0 個以上の文字の任意の組み合わせを表します。たとえば、"path": [{"wildcard": "Football/index*.m3u8"}], は、Football フォルダのパターン index*.m3u8 (例: index.m3u8、index1.m3us8、および index123456.m3u8) に一致するすべてのファイルに適用されます。1 つのルールに最大 10 個の パスを含めることができます。
- seconds_since_create: 常に numeric に設定されます。1～300秒の値を指定できます。演算子は、より大きい (>) または以上 (>=) に設定することもできます。
- action: 常に EXPIRE に設定されます。

一時データルールの場合 (オブジェクトは数秒以内に期限切れになる)、オブジェクトの期限切れとオブジェクトの削除の間に遅延はありません。

> **Note**
>
> 一時データルールの対象となるオブジェクトは、list-items レスポンスには含まれていません。また、一時的なデータルールのために期限切れになるオブジェクトは、有効期限が切れたときに CloudWatch イベントを発行しません。

DELETE Object

オブジェクト削除ルールは、オブジェクトが数日以内に期限切れになるように設定します。このタイプのルールは、ポリシーが作成される前にオブジェクトがコンテナに追加された場合でも、コンテナ内のすべてのオブジェクトに適用されます。MediaStore がコンテナに新しいポリシーを適用するまでに最大 20 分かかりますが、オブジェクトがコンテナからクリアされるまでには最大 24 時間かかる場合があります。

オブジェクトの削除に関する 2 つのルールの例は以下のようになります。

```
        {
            "definition": {
                "path": [ { "prefix": "FolderName/" }  ],
                "days_since_create": [
                    {"numeric": [">" , 5]}
                ]
            },
            "action": "EXPIRE"
        },
        {
            "definition": {
                "path": [ { "wildcard": "Football/*.ts" }  ],
                "days_since_create": [
                    {"numeric": [">" , 5]}
                ]
            },
            "action": "EXPIRE"
        }
```

オブジェクトの削除ルールには以下の 3 つの部分があります。

- path: prefix または wildcard に設定します。同じルールで prefix と wildcard を混在させることはできません。両方を使用する場合は、上記の例に示すように、prefix に 1 つのルールを作成し、wildcard に別のルールを作成する必要があります。
  - prefix - 特定のフォルダ内のすべてのオブジェクトを削除する場合は、パスを prefix に設定します。パラメータが空 ("path": [{ "prefix": "" }],) の場合は、現在のコンテナ内に保存されているすべてのオブジェクトがターゲットになります。1 つのルールに最大 10 個の prefix パスを含めることができます。
  - wildcard - ファイル名やファイルタイプに基づいて特定のオブジェクトを削除する場合は、パスを wildcard に設定します。1 つ以上のワイルドカードを使用できます。これはアスタリス

ク (*) で表されます。各ワイルドカードは、0 個以上の文字の任意の組み合わせを表します。たとえば、"path": [{"wildcard": "Football/*.ts"}], は、Football フォルダ内で *.ts のパターンに一致するすべてのファイル (filename.ts、filename1.ts、filename123456.ts など) に適用されます。1 つのルールに最大 10 個の wildcard パスを含めることができます。

- days_since_create: 常に numeric に設定されます。1 ~ 36,500の値を指定できます。演算子は、より大きい (>) または以上 (>=) に設定することもできます。
- action: 常に EXPIRE に設定されます。

オブジェクト削除ルールの場合 (オブジェクトは数日以内に期限切れになる)、オブジェクトの期限切れとオブジェクトの削除の間にはわずかな遅延が生じます。ただし、オブジェクトの有効期限が切れるとすぐに請求が変更されます。たとえば、ライフサイクルルールで 10 days_since_create を指定した場合、オブジェクトがまだ削除されていない場合でも、オブジェクトの作成から 10 日が経過すると、アカウントでそのオブジェクトには課金されなくなります。

ライフサイクル移行

ライフサイクル移行ルールは、オブジェクトが一定の期間 (日単位) に達した後に低頻度アクセス (IA) ストレージクラスに移動されるよう設定します。IA ストレージクラスに保存されているオブジェクトは、標準のストレージクラスに保存されているオブジェクトとはストレージおよび取得の速度が異なります。詳細については、「MediaStore 料金」を参照してください。

オブジェクトを IA ストレージクラスに移動すると、標準のストレージクラスに戻すことはできません。

ライフサイクル移行ルールは、ポリシーの作成前にコンテナに追加されていたオブジェクトも含め、コンテナ内のすべてのオブジェクトに適用されます。MediaStore がコンテナに新しいポリシーを適用するまでに最大 20 分かかりますが、オブジェクトがコンテナからクリアされるまでには最大 24 時間かかる場合があります。

ライフサイクル移行ルールの例は次のようになります。

```
{
    "definition": {
        "path": [
            {"prefix": "AwardsShow/"}
        ],
        "days_since_create": [
            {"numeric": [">=" , 30]}
        ]
    },
```

```
            "action": "ARCHIVE"
        }
```

ライフサイクル移行ルールには、以下の 3 つの部分があります。

- path: prefix または wildcard に設定します。同じルールで prefix と wildcard を混在させることはできません。両方を使用する場合は、prefix に 1 つのルールを作成し、wildcard には別のルールを作成する必要があります。
  - prefix - 特定のフォルダ内のすべてのオブジェクトを IA ストレージクラスに移行する場合は、パスを prefix に設定します。パラメータが空 ("path": [{ "prefix": "" }],) の場合は、現在のコンテナ内に保存されているすべてのオブジェクトがターゲットになります。1 つのルールに最大 10 個の prefix パスを含めることができます。
  - wildcard - ファイル名やファイルタイプに基づいて IA ストレージクラスに特定のオブジェクトを移行する場合は、パスを wildcard に設定します。1 つ以上のワイルドカードを使用できます。これはアスタリスク (*) で表されます。各ワイルドカードは、0 個以上の文字の任意の組み合わせを表します。たとえば、"path": [{"wildcard": "Football/*.ts"}], は、Football フォルダ内で *.ts のパターンに一致するすべてのファイル (filename.ts、filename1.ts、filename123456.ts など) に適用されます。1 つのルールに最大 10 個の wildcard パスを含めることができます。
- days_since_create: 常に "numeric": [">=" , 30] に設定されます。
- action: 常に ARCHIVE に設定されます。

例

「LiveEvents 」という名前のコンテナに、4 つのサブフォルダ Football、Baseball、Basketball、AwardsShow があるとします。LiveEvents フォルダに割り当てられたオブジェクトのライフサイクルポリシーは、次のようになります。

```
{
    "rules": [
         {
            "definition": {
                "path": [
                    {"prefix": "Football/"},
                    {"prefix": "Baseball/"}
                ],
                "days_since_create": [
                    {"numeric": [">" , 28]}
```

```
                ]
            },
            "action": "EXPIRE"
        },
        {
            "definition": {
                "path": [ { "prefix": "AwardsShow/" }  ],
                "days_since_create": [
                    {"numeric": [">=" , 15]}
                ]
            },
            "action": "EXPIRE"
        },
        {
            "definition": {
                "path": [ { "prefix": "" }  ],
                "days_since_create": [
                    {"numeric": [">" , 40]}
                ]
            },
            "action": "EXPIRE"
        },
        {
            "definition": {
                "path": [ { "wildcard": "Football/*.ts" }  ],
                "days_since_create": [
                    {"numeric": [">" , 20]}
                ]
            },
            "action": "EXPIRE"
        },
        {
            "definition": {
                "path": [
                    {"wildcard": "Football/index*.m3u8"}
                ],
                "seconds_since_create": [
                    {"numeric": [">" , 15]}
                ]
            },
            "action": "EXPIRE"
        },
        {
                "definition": {
```

```
                    "path": [
                        {"prefix": "Program/"}
                    ],
                    "days_since_create": [
                        {"numeric": [">=" , 30]}
                    ]
                },
                "action": "ARCHIVE"
            }
    ]
}
```

前述のポリシーでは、以下を指定します。

- 最初のルールでは、LiveEvents/Football フォルダと LiveEvents/Baseball フォルダに 28 日以上保存されているオブジェクトを削除するように AWS Elemental MediaStore に指示します。
- 2 つ目のルールでは、LiveEvents/AwardsShow フォルダに 15 日以上保存されているオブジェクトが削除されるようにサービスに指示します。
- 3 つ目のルールでは、LiveEvents コンテナ内のどこかに 40 日以上保存されているオブジェクトが削除されるように指定します。このルールは、 LiveEvents コンテナに直接保存されているオブジェクト、コンテナの 4 つのサブフォルダのいずれかに保存されているオブジェクトに適用されます。
- 4 番目のルールでは、Football フォルダ内でパターン *.ts に一致するオブジェクトを 20 日経過したら削除するようにサービスに指示します。
- 5 番目のルールでは、Football フォルダ内でパターン index*.m3u8 に一致するオブジェクトを 15 秒経過したら削除するようにサービスに指示します。MediaStore は、コンテナに配置されてから 16 秒後にこれらのファイルを削除します。
- 6 番目のルールは、30 日経過後に Program フォルダ内のオブジェクトを IA ストレージクラスに移動するようにサービスに指示します。

オブジェクトライフサイクルポリシーの例の詳細については、「オブジェクトのライフサイクルポリシーの例」を参照してください。

コンテナにオブジェクトのライフサイクルポリシーを追加する

オブジェクトのライフサイクルポリシーでは、ユーザーがコンテナにオブジェクトを保存する期間を指定することができます。有効期限を設定し、有効期限が切れたら、AWS Elemental MediaStore に

よってオブジェクトが削除されます。サービスがコンテナに新しいポリシーを適用するまでに、最大 20 分かかります。

ライフサイクルポリシーを作成する方法については、「オブジェクトのライフサイクルポリシーのコンポーネント」を参照してください。

> **Note**
>
> オブジェクト削除ルールの場合 (オブジェクトは数日以内に期限切れになる)、オブジェクトの期限切れとオブジェクトの削除の間にはわずかな遅延が生じます。ただし、オブジェクトの有効期限が切れるとすぐに請求が変更されます。たとえば、ライフサイクルルールで 10 `days_since_create` を指定した場合、オブジェクトがまだ削除されていない場合でも、オブジェクトの作成から 10 日が経過すると、アカウントでそのオブジェクトには課金されなくなります。

オブジェクトのライフサイクルポリシーを追加するには (コンソール)

1. https://console.aws.amazon.com/mediastore/ で MediaStore コンソールを開きます。
2. [Containers (コンテナ)] ページで、オブジェクトのライフサイクルポリシーを作成する対象のコンテナの名前を選択します。

   コンテナの詳細ページが表示されます。
3. [オブジェクトのライフサイクルポリシー] セクションで、[オブジェクトのライフサイクルポリシーの作成] を選択します。
4. ポリシーを JSON 形式で挿入し、[Save (保存)] を選択します。

オブジェクトのライフサイクルポリシーを追加するには (AWS CLI)

1. オブジェクトのライフサイクルポリシーを定義するファイルを作成します。

```
{
    "rules": [
        {
           "definition": {
               "path": [
                   {"prefix": "Football/"},
                   {"prefix": "Baseball/"}
               ],
```

```
                "days_since_create": [
                    {"numeric": [">" , 28]}
                ]
            },
            "action": "EXPIRE"
        },
        {
            "definition": {
                "path": [
                    {"wildcard": "AwardsShow/index*.m3u8"}
                ],
                "seconds_since_create": [
                    {"numeric": [">" , 8]}
                ]
            },
            "action": "EXPIRE"
        }
    ]
}
```

2. で AWS CLI、 `put-lifecycle-policy` コマンドを使用します。

```
aws mediastore put-lifecycle-policy --container-name LiveEvents --lifecycle-
policy file://LiveEventsLifecyclePolicy.json --region us-west-2
```

このコマンドの戻り値はありません。指定されたポリシーがコンテナにアタッチされます。

オブジェクトのライフサイクルポリシーを表示する

オブジェクトのライフサイクルポリシーでは、オブジェクトをコンテナに保存する期間を指定します。

オブジェクトのライフサイクルポリシーを表示するには (コンソール)

1. https://console.aws.amazon.com/mediastore/ で MediaStore コンソールを開きます。
2. [Containers (コンテナ)] ページで、オブジェクトのライフサイクルポリシーを表示する対象のコンテナの名前を選択します。

   [オブジェクトのライフサイクルポリシー] セクションのオブジェクトのライフサイクルポリシーによって、コンテナの詳細ページが表示されます。

オブジェクトのライフサイクルポリシーを表示するには (AWS CLI)

- で AWS CLI、 `get-lifecycle-policy` コマンドを使用します。

```
aws mediastore get-lifecycle-policy --container-name LiveEvents --region us-west-2
```

戻り値の例を以下に示します。

```
{
    "LifecyclePolicy": "{
        "rules": [
            {
                "definition": {
                    "path": [
                        {"prefix": "Football/"},
                        {"prefix": "Baseball/"}
                    ],
                    "days_since_create": [
                        {"numeric": [">" , 28]}
                    ]
                },
             "action": "EXPIRE"
            }
        ]
    }"
}
```

オブジェクトのライフサイクルポリシーを編集する

既存のオブジェクトのライフサイクルポリシーを編集することはできません。ただし、代わりのポリシーをアップロードして、既存のポリシーを変更することができます。サービスが、更新されたポリシーをコンテナに適用するまでに、最大 20 分かかります。

オブジェクトのライフサイクルポリシーを編集するには (コンソール)

1. https://console.aws.amazon.com/mediastore/ で MediaStore コンソールを開きます。
2. [Containers (コンテナ)] ページで、オブジェクトのライフサイクルポリシーを編集する対象のコンテナの名前を選択します。

   コンテナの詳細ページが表示されます。

3. [オブジェクトのライフサイクルポリシー] セクションで、[オブジェクトのライフサイクルポリシーの編集] を選択します。
4. ポリシーを変更し、[Save (保存)] を選択します。

オブジェクトのライフサイクルポリシーを編集するには (AWS CLI)

1. 更新するオブジェクトのライフサイクルポリシーを定義するファイルを作成します。

```
{
    "rules": [
         {
            "definition": {
                "path": [
                    {"prefix": "Football/"},
                    {"prefix": "Baseball/"}
                    {"prefix": "Basketball/"}
                ],
                "days_since_create": [
                    {"numeric": [">" , 28]}
                ]
            },
            "action": "EXPIRE"
        }
    ]
}
```

2. で AWS CLI、 `put-lifecycle-policy` コマンドを使用します。

```
aws mediastore put-lifecycle-policy --container-name LiveEvents --lifecycle-
policy file://LiveEvents2LifecyclePolicy --region us-west-2
```

このコマンドの戻り値はありません。このサービスでは、指定されたポリシーを以前のポリシーと置き換えてコンテナにアタッチします。

オブジェクトのライフサイクルポリシーを削除する

オブジェクトライフサイクルポリシーを削除すると、サービスがコンテナに変更を適用するまで最大 20 分かかります。

オブジェクトのライフサイクルポリシーを削除するには (コンソール)

1. https://console.aws.amazon.com/mediastore/ で MediaStore コンソールを開きます。
2. [Containers (コンテナ)] ページで、オブジェクトのライフサイクルポリシーを削除する対象のコンテナの名前を選択します。

   コンテナの詳細ページが表示されます。
3. [オブジェクトのライフサイクルポリシー] セクションで、[ライフサイクルポリシーの削除] を選択します。
4. [Continue] を選択して確認し、[Save] を選択します。

オブジェクトのライフサイクルポリシーを削除するには (AWS CLI)

- で AWS CLI、 `delete-lifecycle-policy` コマンドを使用します。

```
aws mediastore delete-lifecycle-policy --container-name LiveEvents --region us-
west-2
```

  このコマンドの戻り値はありません。

オブジェクトのライフサイクルポリシーの例

次の例は、オブジェクトのライフサイクルポリシーを示しています。

トピック

- オブジェクトライフサイクルポリシーの例: 数秒以内に期限切れにする
- オブジェクトライフサイクルポリシーの例:数日以内に期限切れにする
- オブジェクトライフサイクルポリシーの例: 低頻度アクセスのストレージクラスへの移行
- オブジェクトライフサイクルポリシーの例: 複数のルール
- オブジェクトライフサイクルポリシーの例: 空のコンテナ

オブジェクトライフサイクルポリシーの例: 数秒以内に期限切れにする

次のポリシーでは、MediaStore が次の条件すべてに一致するオブジェクトを削除するように指定します。

- オブジェクトは、ポリシーが有効になった後にコンテナに追加されます。
- オブジェクトは、Football フォルダに保存されます。
- オブジェクトのファイル拡張子は m3u8 です。
- このオブジェクトはコンテナに 20 秒以上保持されています。

```
{
    "rules": [
        {
            "definition": {
                "path": [
                    {"wildcard": "Football/*.m3u8"}
                ],
                "seconds_since_create": [
                    {"numeric": [ ">", 20 ]}
                ]
            },
            "action": "EXPIRE"
        }
    ]
}
```

オブジェクトライフサイクルポリシーの例:数日以内に期限切れにする

次のポリシーでは、MediaStore が次の条件すべてに一致するオブジェクトを削除するように指定します。

- オブジェクトは、Program フォルダに保存されます
- オブジェクトのファイル拡張子は ts です
- オブジェクトがコンテナに 5 日以上保持されています

```
{
    "rules": [
        {
            "definition": {
                "path": [
                    {"wildcard": "Program/*.ts"}
                ],
                "days_since_create": [
```

```
                    {"numeric": [ ">", 5 ]}
                ]
            },
            "action": "EXPIRE"
        }
    ]
}
```

オブジェクトライフサイクルポリシーの例: 低頻度アクセスのストレージクラスへの移行

次のポリシーでは、30 日経過したときに、MediaStore がオブジェクトを低頻度アクセス (IA) ストレージクラスに移動するように指定します。IA ストレージクラスに保存されているオブジェクトは、標準のストレージクラスに保存されているオブジェクトとはストレージおよび取得の速度が異なります。

days_since_create フィールドは "numeric": [">=" ,30] に設定する必要があります。

```
{
    "rules": [
        {
            "definition": {
                "path": [
                    {"prefix": "Football/"},
                    {"prefix": "Baseball/"}
                ],
                "days_since_create": [
                    {"numeric": [">=" , 30]}
                ]
            },
            "action": "ARCHIVE"
        }
    ]
}
```

オブジェクトライフサイクルポリシーの例: 複数のルール

次のポリシーは、MediaStore が次のことを行うことを指定します。

- AwardsShow フォルダに保存されているオブジェクトを、30 日後に低頻度アクセス (IA) ストレージクラスに移動します

- ファイル拡張子が m3u8 で、Football フォルダに保存されて 20 秒経過したオブジェクトを削除します
- April フォルダに保存されて 10 日間経過したオブジェクトを削除します
- ファイル拡張子が ts で、Program フォルダに保存されて 5 日間経過したオブジェクトを削除します

```
{
    "rules": [
        {
            "definition": {
                "path": [
                    {"prefix": "AwardsShow/"}
                ],
                "days_since_create": [
                    {"numeric": [ ">=" , 30 ]}
                ]
            },
            "action": "ARCHIVE"
        },
        {
            "definition": {
                "path": [
                    {"wildcard": "Football/*.m3u8"}
                ],
                "seconds_since_create": [
                    {"numeric": [ ">", 20 ]}
                ]
            },
            "action": "EXPIRE"
        },
        {
            "definition": {
                "path": [
                    {"prefix": "April"}
                ],
                "days_since_create": [
                    {"numeric": [ ">", 10 ]}
                ]
            },
            "action": "EXPIRE"
        },
```

```
        {
            "definition": {
                "path": [
                    {"wildcard": "Program/*.ts"}
                ],
                "days_since_create": [
                    {"numeric": [ ">", 5 ]}
                ]
            },
            "action": "EXPIRE"
        }
    ]
}
```

オブジェクトライフサイクルポリシーの例: 空のコンテナ

次のオブジェクトライフサイクルポリシーは、コンテナに追加されてから 1 日後に、MediaStore がコンテナ内のすべてのオブジェクト (フォルダやサブフォルダを含む) を削除するように指定します。このポリシーが適用される前にコンテナにオブジェクトが保持されている場合、MediaStore はポリシーが有効になってから 1 日後にオブジェクトを削除します。サービスがコンテナに新しいポリシーを適用するまでに、最大 20 分かかります。

```
{
    "rules": [
        {
            "definition": {
                "path": [
                    {"wildcard": "*"}
                ],
                "days_since_create": [
                    {"numeric": [ ">=", 1 ]}
                ]
            },
            "action": "EXPIRE"
        }
    ]
}
```

AWS Elemental MediaStore のメトリクスポリシー

コンテナごとに、メトリクスポリシーを追加して、AWS Elemental MediaStore がメトリクスを Amazon CloudWatch に送信できるようにすることができます。新しいポリシーが有効になるまで、最大 20 分かかります。各 MediaStore メトリクスの説明については、「MediaStore メトリクス」を参照してください。

メトリクスポリシーには、次のものが含まれます。

- コンテナレベルでメトリクスを有効または無効にする設定。
- オブジェクトレベルでメトリクスを有効にする 0 から 5 までのルールの任意の場所。ポリシーにルールが含まれている場合は、各ルールに次の両方を含める必要があります。
  - グループに含めるオブジェクトを定義するオブジェクトグループ。定義にはパスまたはファイル名を使用できますが、900 文字を超えることはできません。有効な文字は、a ~ z、A ~ Z、0 ~ 9、_ (アンダースコア)、= (等しい)、: (コロン)、. (ピリオド)、- (ハイフン)、~ (チルダ)、/ (スラッシュ)、* (アスタリスク) です。ワイルドカード (*) も使用できます。
  - オブジェクトグループを参照できるオブジェクトグループ名。名前は 30 文字を超えることはできません。有効な文字は、a ~ z、A ~ Z、0 ~ 9、_ (下線) です。

オブジェクトが複数のルールに一致する場合、CloudWatch は一致するルールごとにデータポイントを表示します。たとえば、オブジェクトが `rule1` と `rule2` いう名前の 2 つのルールに一致する場合、CloudWatch はこれらのルールの 2 つのデータポイントを表示します。最初のルールには、`ObjectGroupName=rule1` のディメンションがあり、2 番目のルールは `ObjectGroupName=rule2` のディメンションがあります。

トピック

- メトリクスポリシーの追加
- メトリクスポリシーの表示
- メトリクスポリシーの編集
- メトリクスポリシーの例

メトリクスポリシーの追加

メトリクスポリシーには、AWS Elemental MediaStore が Amazon CloudWatch に送信するメトリクスを指定するルールが含まれています。メトリクスポリシーの例については、「メトリクスポリシーの例」を参照してください。

メトリクスポリシーを追加するには (コンソール)

1. https://console.aws.amazon.com/mediastore/ で MediaStore コンソールを開きます。
2. [Containers (コンテナ)] ページで、メトリクスポリシーを追加する対象のコンテナの名前を選択します。

   コンテナの詳細ページが表示されます。
3. [メトリクスポリシー] セクションで、[Create metric policy (メトリクスポリシーの作成)] を選択します。
4. ポリシーを JSON 形式で挿入し、[Save (保存)] を選択します。

メトリクスポリシーの表示

コンソールまたは を使用して AWS CLI 、コンテナのメトリクスポリシーを表示できます。

メトリクスポリシーを表示するには (コンソール)

1. https://console.aws.amazon.com/mediastore/ で MediaStore コンソールを開きます。
2. [Containers (コンテナ)] ページで、コンテナの名前を選択します。

   コンテナの詳細ページが表示されます。ポリシーが [メトリクスポリシー] セクションに表示されます。

メトリクスポリシーの編集

メトリクスポリシーには、AWS Elemental MediaStore が Amazon CloudWatch に送信するメトリクスを指定するルールが含まれています。既存のメトリクスポリシーを編集する場合、新しいポリシーが有効になるまでに最大 20 分かかります。メトリクスポリシーの例については、「メトリクスポリシーの例」を参照してください。

メトリクスポリシーを編集するには (コンソール)

1. https://console.aws.amazon.com/mediastore/ で MediaStore コンソールを開きます。
2. [Containers (コンテナ)] ページで、コンテナの名前を選択します。
3. [メトリクスポリシー] セクションで、[メトリクスポリシーの編集] を選択します。
4. 適切な変更を行い、[Save] (保存) を選択します。

メトリクスポリシーの例

さまざまなユースケース向けに作成されたメトリクスポリシーの例を以下に示します。

トピック

メトリクスポリシーの例: コンテナレベルのメトリクス

このサンプルポリシーは、AWS Elemental MediaStore がメトリクスをコンテナレベルで Amazon CloudWatch に送信する必要があることを示します。たとえば、これには、コンテナに対して行われた Put リクエストの数をカウントする RequestCount メトリクスが含まれます。または、これを DISABLED に設定することもできます。

このポリシーにはルールがないため、MediaStore はパスレベルでメトリクスを送信しません。たとえば、このコンテナ内の特定のフォルダに対して行われた Put リクエストの数を確認することはできません。

```
{
   "ContainerLevelMetrics": "ENABLED"
}
```

メトリクスポリシーの例: パスレベルのメトリクス

このサンプルポリシーは、AWS Elemental MediaStore がメトリクスをコンテナレベルで Amazon CloudWatch に送信しないことを示します。また、MediaStore は、baseball/saturday および football/saturday の 2 つの特定のフォルダ内のオブジェクトのメトリクスを送信する必要があります。MediaStore のリクエストのメトリクスは次のとおりです。

- baseball/saturday フォルダへのリクエストには CloudWatch の ディメンション ObjectGroupName=baseballGroup が含まれます。
- football/saturday フォルダへのリクエストにはディメンション ObjectGroupName=footballGroup が含まれます。

```
{
   "ContainerLevelMetrics": "DISABLED",
   "MetricPolicyRules": [
     {
       "ObjectGroup": "baseball/saturday",
       "ObjectGroupName": "baseballGroup"
     },
     {
       "ObjectGroup": "football/saturday",
       "ObjectGroupName": "footballGroup"
     }
   ]
}
```

メトリクスポリシーの例: コンテナレベルおよびパスレベルのメトリクス

このサンプルポリシーは、AWS Elemental MediaStore がメトリクスをコンテナレベルで Amazon CloudWatch に送信する必要があることを示します。また、MediaStore は、`baseball/saturday` および `football/saturday` の 2 つの特定のフォルダ内のオブジェクトのメトリクスを送信する必要があります。MediaStore のリクエストのメトリクスは次のとおりです。

- `baseball/saturday` フォルダへのリクエストには CloudWatch の ディメンション `ObjectGroupName=baseballGroup` が含まれます。
- `football/saturday` フォルダへのリクエストには CloudWatch の ディメンション `ObjectGroupName=footballGroup` が含まれます。

```
{
   "ContainerLevelMetrics": "ENABLED",
   "MetricPolicyRules": [
     {
       "ObjectGroup": "baseball/saturday",
       "ObjectGroupName": "baseballGroup"
     },
     {
       "ObjectGroup": "football/saturday",
       "ObjectGroupName": "footballGroup"
     }
   ]
}
```

メトリクスポリシーの例: ワイルドカードを使用したパスレベルのメトリクス

このサンプルポリシーは、AWS Elemental MediaStore がメトリクスをコンテナレベルで Amazon CloudWatch に送信する必要があることを示します。また、MediaStore は、ファイル名に基づいてオブジェクトのメトリクスも送信する必要があります。ワイルドカードは、オブジェクトがコンテナ内のどこにでも保存され、.m3u8 拡張子で終わる限り、任意のファイル名を持つことができることを示します。

```
{
   "ContainerLevelMetrics": "ENABLED",
   "MetricPolicyRules": [
     {
       "ObjectGroup": "*.m3u8",
       "ObjectGroupName": "index"
     }
   ]
 }
```

メトリクスポリシーの例: ルールが重複するパスレベルのメトリクス

このサンプルポリシーは、AWS Elemental MediaStore がメトリクスをコンテナレベルで Amazon CloudWatch に送信する必要があることを示します。さらに、MediaStore は、sports/football/saturday と sports/football の 2 つのフォルダのメトリクスを送信する必要があります。

sports/football/saturday フォルダへの MediaStore のリクエストのメトリクスには、CloudWatch のディメンション ObjectGroupName=footballGroup1 が含まれます。sports/football フォルダに保存されているオブジェクトは両方のルールに一致するため、CloudWatch では、これらのオブジェクトの 2 つのデータポイントが表示されます。1 つはディメンション ObjectGroupName=footballGroup1 のデータポイントで、もう 1 つはディメンション ObjectGroupName=footballGroup2 のデータポイントです。

```
{
   "ContainerLevelMetrics": "ENABLED",
   "MetricPolicyRules": [
     {
       "ObjectGroup": "sports/football/saturday",
       "ObjectGroupName": "footballGroup1"
     },
     {
       "ObjectGroup": "sports/football",
```

```
        "ObjectGroupName": "footballGroup2"
      }
    ]
}
```

AWS Elemental MediaStore のフォルダ

フォルダはコンテナ内の区分です。フォルダを使用してコンテナを分割します。ファイルシステムでサブフォルダを作成してフォルダを分割するのと同じです。最大 10 レベルのフォルダを作成できます (コンテナ自体は含みません)。

フォルダは省略可能です。オブジェクトは、フォルダを使わずに直接コンテナにアップロードできます。ただし、フォルダを使用するとオブジェクトを整理しやすくなります。

オブジェクトをフォルダにアップロードするには、フォルダへのパスを指定します。フォルダが既に存在する場合、AWS Elemental MediaStore はそのフォルダにオブジェクトを保存します。フォルダが存在しない場合は、フォルダが自動的に作成されて、そのフォルダにオブジェクトが保存されます。

例えば、`movies` というコンテナがあり、`mlaw.ts` というファイルをアップロードするとします。アップロードパスは `premium/canada` です。AWS Elemental MediaStore は、オブジェクトを premium フォルダの下の canada サブフォルダに保存します。どちらのフォルダも存在しない場合は、`premium` フォルダと `canada` サブフォルダの両方が自動的に作成され、オブジェクトが `canada` サブフォルダに保存されます。パスを指定せずにコンテナ `movies` のみを指定すると、オブジェクトはコンテナに直接保存されます。

フォルダから最後のオブジェクトを削除すると、AWS Elemental MediaStore によってフォルダが自動的に削除されます。そのフォルダの上位にあるすべての空のフォルダも削除されます。たとえば、premium というフォルダ内にはファイルが存在せず、1 つのサブフォルダ `canada` のみが存在するとします。`canada` サブフォルダには、`mlaw.ts` というファイルが 1 つ含まれています。ここで `mlaw.ts` ファイルを削除すると、`premium` ファルダと `canada` フォルダの両方が自動的に削除されます。この自動削除はフォルダにのみ適用されます。空のコンテナは削除されません。

トピック

フォルダ名に関するルール

フォルダの名前を選択する際は、次の点に注意してください。

- 名前に使用できる文字は、大文字 (A ~ Z)、小文字 (a ~ z)、数字 (0 ~ 9)、ピリオド (.)、ハイフン (-)、チルダ (~)、アンダースコア (_)、等号 (=)、およびコロン (:) です。
- 名前は 1 文字以上である必要があります。空のフォルダ名 (`folder1//folder3/` など) は使用できません。
- 名前では、大文字と小文字が区別されます。たとえば、`myFolder` フォルダと `myfolder` フォルダは一意の名前であるため、同じコンテナまたはフォルダ内で使用できます。
- 名前は親コンテナまたは親フォルダ内でのみ一意である必要があります。たとえば、`myfolder` というフォルダを 2 つの異なるコンテナ (`movies/myfolder` と `sports/myfolder`) に作成できます。
- 名前は親コンテナと同じ名前にすることができます。
- フォルダを作成した後に名前を変更することはできません。

フォルダの作成

オブジェクトをアップロードするときにフォルダを作成できます。オブジェクトをフォルダにアップロードするには、フォルダへのパスを指定します。フォルダが既に存在する場合、AWS Elemental MediaStore はそのフォルダにオブジェクトを保存します。フォルダが存在しない場合は、フォルダが自動的に作成されて、そのフォルダにオブジェクトが保存されます。

詳細については、「the section called “オブジェクトのアップロード”」を参照してください。

フォルダの削除

フォルダが空の場合にのみフォルダを削除できます。オブジェクトが含まれているフォルダは削除できません。

フォルダから最後のオブジェクトを削除すると、AWS Elemental MediaStore によってフォルダが自動的に削除されます。そのフォルダの上位にあるすべての空のフォルダも削除されます。たとえば、`premium` というフォルダ内にはファイルが存在せず、1 つのサブフォルダ `canada` のみが存在するとします。`canada` サブフォルダには、`mlaw.ts` というファイルが 1 つ含まれています。ここで `mlaw.ts` ファイルを削除すると、`premium` ファルダと `canada` フォルダの両方が自動的に削除されます。この自動削除はフォルダにのみ適用されます。空のコンテナは削除されません。

詳細については、「オブジェクトの削除」を参照してください。

AWS Elemental MediaStore のオブジェクト

AWS Elemental MediaStore のアセットは、オブジェクトと呼ばれます。オブジェクトは、コンテナまたはコンテナ内のフォルダにアップロードできます。

MediaStore では、オブジェクトのアップロード、ダウンロード、削除を行うことができます。

- アップロード - オブジェクトをコンテナまたはフォルダに追加します。これは、オブジェクトの作成とは異なります。オブジェクトを MediaStore にアップロードするには、事前にローカルでオブジェクトを作成する必要があります。
- ダウンロード - オブジェクトを MediaStore から別の場所にコピーします。オブジェクトは MediaStore からは削除されません。
- 削除 - オブジェクトを MediaStore から完全に削除します。オブジェクトを個別に削除するか、オブジェクトライフサイクルポリシーを追加して、指定された期間が経過するとコンテナ内のオブジェクトが自動的に削除されるように設定することができます。

MediaStore は、すべてのファイルタイプをサポートしています。

トピック

オブジェクトのアップロード

オブジェクトをコンテナ、またはコンテナ内のフォルダにアップロードできます。オブジェクトをフォルダにアップロードするには、フォルダへのパスを指定します。フォルダが既に存在する場合、AWS Elemental MediaStore はそのフォルダにオブジェクトを保存します。フォルダが存在しない場合は、フォルダが自動的に作成されて、そのフォルダにオブジェクトが保存されます。フォルダの詳細については、「AWS Elemental MediaStore のフォルダ」を参照してください。

MediaStore コンソールまたは AWS CLI を使用して、オブジェクトをアップロードできます。

MediaStore では、オブジェクトのチャンク転送をサポートしており、アップロード中のオブジェクトをダウンロード可能にすることでレイテンシーが低減されます。この機能を使用するには、オブジェクトのアップロードの可用性を `streaming` に設定します。API を使用してオブジェクトをアップロードする場合に、このヘッダーの値を設定できます。リクエストでこのヘッダーを指定しない場合は、MediaStore によって、オブジェクトのアップロードの可用性にデフォルト値 `standard` が割り当てられます。

オブジェクトのサイズは、標準アップロードの可用性に対しては 25 MB を、ストリーミングアップロードの可用性に対しては 10 MB を超えることはできません。

> **Note**
>
> オブジェクトのファイル名には、文字、数字、ピリオド (.)、アンダースコア (_)、チルダ (~)、ハイフン (-)、等号 (=)、およびコロン (:) のみを使用できます。

オブジェクトをアップロードするには (コンソール)

1. https://console.aws.amazon.com/mediastore/ で MediaStore コンソールを開きます。
2. [Containers (コンテナ)] ページで、コンテナの名前を選択します。コンテナの詳細パネルが表示されます。
3. [Upload object (オブジェクトのアップロード)] を選択します。
4. [Target path (ターゲットのパス)] に、フォルダへのパスを入力します 例えば、`premium/canada`。指定したパスにフォルダがまだ存在していない場合は、自動的に作成されます。
5. [オブジェクト] セクションで、[参照] を選択します。
6. 適切なフォルダに移動し、アップロードする 1 つのオブジェクトを選択します。
7. [Open (開く)]、[Upload (アップロード)] の順に選択します。

> **Note**
>
> 選択したフォルダ内に同じ名前のファイルが既に存在する場合、元のファイルはアップロードしたファイルで置き換えられます。

オブジェクトをアップロードするには (AWS CLI)

- で AWS CLI、 `put-object` コマンドを使用します。また、`content-type`、`cache-control` (呼び出し側によるオブジェクトのキャッシュ動作のコントロールを許可)、`path` (オブジェクトをコンテナ内のフォルダに配置) のパラメータを含めることもできます。

> **Note**
>
> オブジェクトをアップロードした後、`content-type`、`cache-control`、または `path` を編集することはできません。

```
aws mediastore-data put-object --endpoint https://
aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com --body README.md --path /
folder_name/README.md --cache-control "max-age=6, public" --content-type binary/
octet-stream --region us-west-2
```

戻り値の例を以下に示します。

```
{
    "ContentSHA256":
 "74b5fdb517f423ed750ef214c44adfe2be36e37d861eafe9c842cbe1bf387a9d",
    "StorageClass": "TEMPORAL",
    "ETag": "af3e4731af032167a106015d1f2fe934e68b32ed1aa297a9e325f5c64979277b"
}
```

オブジェクトのリストの表示

AWS Elemental MediaStore コンソールでは、最上位のコンテナまたはフォルダに保存されている項目 (オブジェクトとフォルダ) を表示できます。現在のコンテナまたはフォルダのサブフォルダに保存されている項目は表示されません。を使用して、コンテナ内にあるフォルダまたはサブフォルダの数に関係なく、コンテナ内のオブジェクトとフォルダのリスト AWS CLI を表示できます。

特定のコンテナに含まれているオブジェクトのリストを表示するには (コンソール)

1. https://console.aws.amazon.com/mediastore/ で MediaStore コンソールを開きます。
2. [Containers (コンテナ)] ページで、表示するフォルダが含まれているコンテナの名前を選択します。

3. リストからフォルダの名前を選択します。

   詳細ページに、フォルダに保存されているすべてのフォルダとオブジェクトが表示されます。

特定のフォルダに含まれているオブジェクトのリストを表示するには (コンソール)

1. https://console.aws.amazon.com/mediastore/ で MediaStore コンソールを開きます。
2. [Containers (コンテナ)] ページで、表示するフォルダが含まれているコンテナの名前を選択します。

   詳細ページに、コンテナに保存されているすべてのフォルダとオブジェクトが表示されます。

特定のコンテナに含まれているオブジェクトとフォルダのリストを表示するには (AWS CLI)

- で AWS CLI、`list-items` コマンドを使用します。

```
aws mediastore-data list-items --endpoint https://
aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com --region us-west-2
```

戻り値の例を以下に示します。

```
{
    "Items": [
        {
            "ContentType": "image/jpeg",
            "LastModified": 1563571859.379,
            "Name": "filename.jpg",
            "Type": "OBJECT",
            "ETag":
 "543ab21abcd1a234ab123456a1a2b12345ab12abc12a1234abc1a2bc12345a12",
            "ContentLength": 3784
        },
        {
            "Type": "FOLDER",
            "Name": "ExampleLiveDemo"
        }
    ]
}
```

> **Note**
>
> seconds_since_create ルールの対象なるオブジェクトは、list-items レスポンスには含まれていません。

特定のフォルダに含まれているオブジェクトとフォルダのリストを表示するには (AWS CLI)

- で AWS CLI、 list-items コマンドを使用し、リクエストの最後に指定されたフォルダ名を指定します。

```
aws mediastore-data list-items --endpoint https://
aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com --path /folder_name --
region us-west-2
```

戻り値の例を以下に示します。

```
{
    "Items": [
        {
            "Type": "FOLDER",
            "Name": "folder_1"
        },
        {
            "LastModified": 1563571940.861,
            "ContentLength": 2307346,
            "Name": "file1234.jpg",
            "ETag":
 "111a1a22222a1a1a222abc333a444444b55ab1111ab2222222222ab333333a2b",
            "ContentType": "image/jpeg",
            "Type": "OBJECT"
        }
    ]
}
```

> **Note**
>
> `seconds_since_create` ルールの対象なるオブジェクトは、`list-items` レスポンスには含まれていません。

オブジェクトの詳細の表示

オブジェクトをアップロードすると、AWS Elemental MediaStore は変更日、コンテンツの長さ、ETag (エンティティタグ)、コンテンツタイプなどの詳細を保存します。オブジェクトのメタデータの使用方法については、「MediaStore による HTTP キャッシュの操作」を参照してください。

オブジェクトの詳細を表示するには (コンソール)

1. https://console.aws.amazon.com/mediastore/ で MediaStore コンソールを開きます。
2. [Containers (コンテナ)] ページで、表示するオブジェクトが含まれているコンテナの名前を選択します。
3. 表示するオブジェクトがフォルダ内にある場合は、オブジェクトが表示されるまで繰り返しフォルダ名を選択します。
4. オブジェクトの名前を選択します。

   詳細ページにオブジェクトに関する情報が表示されます。

オブジェクトの詳細を表示するには (AWS CLI)

- で AWS CLI、 `describe-object` コマンドを使用します。

```
aws mediastore-data describe-object --endpoint https://
aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com --path /folder_name/
file1234.jpg --region us-west-2
```

戻り値の例を以下に示します。

```
{
    "ContentType": "image/jpeg",
    "LastModified": "Fri, 19 Jul 2019 21:32:20 GMT",
```

```
    "ContentLength": "2307346",
    "ETag": "2aa333bbcc8d8d22d777e999c88d4aa9eeeeee4dd89ff7f555555555555da6d3"
}
```

オブジェクトのダウンロード

オブジェクトをダウンロードするには、コンソールを使用できます。を使用して、オブジェクトまたはオブジェクトの一部のみ AWS CLI をダウンロードすることができます。

オブジェクトをダウンロードするには (コンソール)

1. https://console.aws.amazon.com/mediastore/ で MediaStore コンソールを開きます。
2. [Containers (コンテナ)] ページで、ダウンロードするオブジェクトが含まれているコンテナの名前を選択します。
3. ダウンロードするオブジェクトがフォルダ内にある場合は、オブジェクトが表示されるまで繰り返しフォルダ名を選択します。
4. オブジェクトの名前を選択します。
5. [オブジェクト] の詳細ページで、[Download (ダウンロード)] を選択します。

オブジェクトをダウンロードするには (AWS CLI)

- で AWS CLI、 `get-object` コマンドを使用します。

```
aws mediastore-data get-object --endpoint https://
aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com --path=/folder_name/
README.md README.md --region us-west-2
```

戻り値の例を以下に示します。

```
{
    "ContentLength": "2307346",
    "ContentType": "image/jpeg",
    "LastModified": "Fri, 19 Jul 2019 21:32:20 GMT",
    "ETag": "2aa333bbcc8d8d22d777e999c88d4aa9eeeeee4dd89ff7f555555555555da6d3",
    "StatusCode": 200
}
```

オブジェクトの一部をダウンロードするには (AWS CLI)

- で AWS CLI、 `get-object` コマンドを使用して範囲を指定します。

```
aws mediastore-data get-object --endpoint https://
aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com --path /folder_name/
README.md --range="bytes=0-100" README2.md --region us-west-2
```

戻り値の例を以下に示します。

```
{
    "StatusCode": 206,
    "ContentRange": "bytes 0-100/2307346",
    "ContentLength": "101",
    "LastModified": "Fri, 19 Jul 2019 21:32:20 GMT",
    "ContentType": "image/jpeg",
    "ETag": "2aa333bbcc8d8d22d777e999c88d4aa9eeeeee4dd89ff7f555555555555da6d3"
}
```

オブジェクトの削除

AWS Elemental MediaStore には、コンテナからオブジェクトを削除するためのさまざまなオプションがあります。

- 個々のオブジェクトを削除します。料金が適用されます。
- コンテナを空にして、コンテナ内のすべてのオブジェクトを一度に削除します。このプロセスでは API コールが使用されるため、通常の API 料金が適用されます。
- オブジェクトライフサイクルポリシーを追加して特定の期間に達したときにオブジェクトを削除します。料金が適用されます。

オブジェクトの削除

コンソールまたは AWS CLIを使用してオブジェクトを個別に削除することができます。または、オブジェクトライフサイクルポリシーを追加して、コンテナ内の特定の期間に達した後にオブジェクトを自動的に削除したり、コンテナを空にしてそのコンテナ 内のすべてのオブジェクトを削除したりできます。

> **Note**
>
> フォルダ内の唯一のオブジェクトを削除すると、AWS Elemental MediaStore は自動的にそのフォルダを削除し、さらにそのフォルダの上位にある空のフォルダも削除します。たとえば、`premium` というフォルダ内にはファイルが存在せず、1 つのサブフォルダ `canada` のみが存在するとします。`canada` サブフォルダには、`mlaw.ts` というファイルが 1 つ含まれています。ここで `mlaw.ts` ファイルを削除すると、`premium` ファルダと `canada` フォルダの両方が自動的に削除されます。

オブジェクトを削除するには (コンソール)

1. https://console.aws.amazon.com/mediastore/ で MediaStore コンソールを開きます。
2. [Containers (コンテナ)] ページで、削除したいオブジェクトが含まれているコンテナの名前を選択します。
3. 削除するオブジェクトがフォルダ内にある場合は、オブジェクトが表示されるまで繰り返しフォルダ名を選択します。
4. オブジェクト名の左にあるオプションを選択します。
5. [削除] を選択します。

オブジェクトを削除するには (AWS CLI)

- で AWS CLI、 `delete-object` コマンドを使用します。

  例:

  ```
  aws mediastore-data --region us-west-2 delete-object --endpoint=https://
  aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com --path=/folder_name/
  README.md
  ```

  このコマンドの戻り値はありません。

コンテナを空にする

コンテナを空にすると、コンテナ内に保存されているすべてのオブジェクトを削除できます。または、オブジェクトライフサイクルポリシーを追加して、コンテナ内の特定の期間に達した後にオブジェクトを自動的に削除したり、オブジェクトを個別に削除したりできます。

コンテナを空にするには (コンソール)

1. https://console.aws.amazon.com/mediastore/ で MediaStore コンソールを開きます。
2. [コンテナ] ページで、空にするコンテナのオプションを選択します。
3. [Empty container (コンテナを空にする)] を選択します。確認メッセージが表示されます。
4. テキストフィールドにコンテナ名を入力し、[空にする] を選択して、コンテナを空にすることを確認します。

AWS Elemental MediaStore 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 、セキュリティを最も重視する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からメリットを得られます。

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説明しています。

- クラウドのセキュリティ – AWS は、 で 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責任を担います AWS クラウド。 は、お客様が安全に使用できるサービス AWS も提供します。サードパーティーの監査者は、AWS コンプライアンスプログラムコンプライアンスプログラムの一環として、当社のセキュリティの有効性を定期的にテストおよび検証。AWS Elemental MediaStore に適用されるコンプライアンスプログラムの詳細については、「コンプライアンスプログラムAWS による対象範囲内のサービスコンプライアンスプログラム」を参照してください。
- クラウドのセキュリティ – 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、ユーザーは、データの機密性、会社の要件、適用される法律や規制など、その他の要因についても責任を負います。

このドキュメントは、MediaStore 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。以下のトピックでは、セキュリティおよびコンプライアンスの目的に合わせて MediaStore を設定する方法について説明します。また、MediaStore 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

AWS Elemental MediaStore でのデータ保護

責任 AWS 共有モデル、AWS Elemental MediaStore でのデータ保護に適用されます。このモデルで説明されているように、 AWS はすべての を実行するグローバルインフラストラクチャを保護する責任があります AWS クラウド。ユーザーは、このインフラストラクチャでホストされるコンテンツに対する管理を維持する責任があります。また、使用する「 AWS のサービス 」のセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、データプライバシーに関するよくある質問を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティブログに投稿された AWS 責任共有モデルおよび GDPR のブログ記事を参照してください。

データ保護の目的で、認証情報を保護し AWS アカウント 、 AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- を使用して API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。CloudTrail 証跡を使用して AWS アクティビティをキャプチャする方法については、「 AWS CloudTrail ユーザーガイド」のCloudTrail 証跡の使用」を参照してください。
- AWS 暗号化ソリューションと、その中のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
- Amazon Macie などの高度な管理されたセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API AWS を介して にアクセスするときに FIPS 140-3 検証済み暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-3」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報を、タグ、または [名前] フィールドなどの自由形式のテキストフィールドに含めないことを強くお勧めします。これは、コンソール、API、または SDK を使用して MediaStore AWS CLIまたは他の AWS のサービス を操作する場合も同様です。AWS SDKs タグ、または名前に使用される自由記述のテキストフィールドに入力したデータは、請求または診断ログに使用される場合があります。外部サーバーに URL を提供する場合、そのサーバーへのリクエストを検証できるように、認証情報を URL に含めないことを強くお勧めします。

データ暗号化

MediaStore は、業界標準の AES-256 アルゴリズムを使用して保管中のコンテナとオブジェクトを暗号化します。MediaStore を使用して、以下の方法でデータを保護することをお勧めします。

- コンテナポリシーを作成して、コンテナ内のすべてのフォルダとオブジェクトへのアクセス権を制御します。詳細については、「the section called “コンテナポリシー”」を参照してください。
- クロスオリジンリソース共有 (CORS) ポリシーを作成して、MediaStore リソースに対してクロスオリジンアクセスを選択的に許可します。CORS を使用すると、特定のドメインにロードされたクライアントウェブアプリケーションが、異なるドメイン内のリソースと通信できるようになります。詳細については、「the section called “CORS ポリシー”」を参照してください。

AWS Elemental MediaStore でのアイデンティティ管理とアクセス管理

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御 AWS のサービス するのに役立つ です。IAM 管理者は、誰を認証 (サインイン) し、誰に MediaStore リソースの使用を許可する (アクセス許可を持たせる) かを制御します。IAM は、追加料金なしで使用できる AWS のサービス です。

トピック

- 対象者
- アイデンティティを使用した認証
- ポリシーを使用したアクセスの管理
- AWS Elemental MediaStore が IAM と連動する方法
- AWS Elemental MediaStore でのアイデンティティベースのポリシーの例
- AWS Elemental MediaStore のアイデンティティとアクセスに関するトラブルシューティング

対象者

AWS Identity and Access Management (IAM) の使用方法は、MediaStore で行う作業によって異なります。

サービスユーザー – ジョブを実行するために MediaStore サービスを使用する場合は、管理者から必要なアクセス許可と認証情報が与えられます。作業を実行するために使用する MediaStore 機能が増えるにつれて、追加のアクセス許可が必要になる場合があります。アクセスの管理方法を理解すると、管理者に適切なアクセス許可をリクエストするのに役に立ちます。MediaStore の機能にアクセスできない場合は、「AWS Elemental MediaStore のアイデンティティとアクセスに関するトラブルシューティング」を参照してください。

サービス管理者 - 社内の MediaStore リソースを担当している場合は、通常、MediaStore へのフルアクセスがあります。サービスのユーザーがどの MediaStore 機能やリソースにアクセスするかを決めるのは管理者の仕事です。その後、IAM 管理者にリクエストを送信して、サービスユーザーの権限を変更する必要があります。このページの情報を点検して、IAM の基本概念を理解してください。会社で MediaStore と IAM を併用する方法の詳細については、「AWS Elemental MediaStore が IAM と連動する方法」を参照してください。

IAM 管理者 - IAM 管理者である場合は、MediaStore へのアクセスを管理するポリシーの作成方法の詳細について理解しておくことをお勧めします。IAM で使用できる MediaStore のアイデンティティベースポリシーの例を確認するには、「AWS Elemental MediaStore でのアイデンティティベースのポリシーの例」を参照してください。

アイデンティティを使用した認証

認証は、ID 認証情報 AWS を使用して にサインインする方法です。として、IAM ユーザーとして AWS アカウントのルートユーザー、または IAM ロールを引き受けることによって、認証 (にサインイン AWS) される必要があります。

ID ソースを介して提供された認証情報を使用して、フェデレーティッド ID AWS として にサインインできます。 AWS IAM Identity Center （ IAM Identity Center) ユーザー、会社のシングルサインオン認証、Google または Facebook 認証情報は、フェデレーション ID の例です。フェデレーティッド ID としてサインインする場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。フェデレーション AWS を使用して にアクセスすると、間接的にロールを引き受けることになります。

ユーザーのタイプに応じて、 AWS Management Console または AWS アクセスポータルにサインインできます。へのサインインの詳細については AWS、「 AWS サインイン ユーザーガイド」の「 へのサインイン AWS アカウント方法」を参照してください。

AWS プログラムで にアクセスする場合、 はソフトウェア開発キット (SDK) とコマンドラインインターフェイス (CLI) AWS を提供し、認証情報を使用してリクエストを暗号化して署名します。AWS ツールを使用しない場合は、自分でリクエストに署名する必要があります。リクエストに自分

で署名する推奨方法の使用については、「IAM ユーザーガイド」の「API リクエストに対するAWS Signature Version 4」を参照してください。

使用する認証方法を問わず、追加セキュリティ情報の提供をリクエストされる場合もあります。たとえば、 では、アカウントのセキュリティを高めるために多要素認証 (MFA) を使用する AWS ことをお勧めします。詳細については、「AWS IAM Identity Center ユーザーガイド」の「多要素認証」および「IAM ユーザーガイド」の「IAM のAWS 多要素認証」を参照してください。

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、アカウント内のすべての およびリソースへの AWS のサービス 完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。この ID は AWS アカウント ルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサインインすることでアクセスできます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザーの認証情報は保護し、ルートユーザーでしか実行できないタスクを実行するときに使用します。ルートユーザーとしてサインインする必要があるタスクの完全なリストについては、「IAM ユーザーガイド」の「ルートユーザー認証情報が必要なタスク」を参照してください。

フェデレーティッドアイデンティティ

ベストプラクティスとして、管理者アクセスを必要とするユーザーを含む人間のユーザーに、ID プロバイダーとのフェデレーションを使用して一時的な認証情報 AWS のサービス を使用して にアクセスすることを要求します。

フェデレーティッド ID は、エンタープライズユーザーディレクトリ、ウェブ ID プロバイダー、、AWS Directory Serviceアイデンティティセンターディレクトリ、または ID ソースを介して提供された認証情報 AWS のサービス を使用して にアクセスするユーザーです。フェデレーティッド ID がアクセスすると AWS アカウント、ロールを引き受け、ロールは一時的な認証情報を提供します。

アクセスを一元管理する場合は、 AWS IAM Identity Centerを使用することをお勧めします。IAM Identity Center でユーザーとグループを作成するか、独自の ID ソースのユーザーとグループのセットに接続して同期し、すべての AWS アカウント とアプリケーションで使用できます。IAM Identity Center の詳細については、「AWS IAM Identity Center ユーザーガイド」の「What is IAM Identity Center?」(IAM Identity Center とは) を参照してください。

IAM ユーザーとグループ

IAM ユーザーは、単一のユーザーまたはアプリケーションに対して特定のアクセス許可 AWS アカウント を持つ 内の ID です。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を保

有する IAM ユーザーを作成する代わりに、一時的な認証情報を使用することをお勧めします。ただし、IAM ユーザーでの長期的な認証情報が必要な特定のユースケースがある場合は、アクセスキーをローテーションすることをお勧めします。詳細については、「IAM ユーザーガイド」の「長期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションする」を参照してください。

IAM グループは、IAM ユーザーの集団を指定するアイデンティティです。グループとしてサインインすることはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できます。多数のユーザーグループがある場合、グループを使用することで権限の管理が容易になります。例えば、IAMAdmins という名前のグループを設定して、そのグループに IAM リソースを管理する許可を与えることができます。

ユーザーは、ロールとは異なります。ユーザーは 1 人の人または 1 つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユーザーには永続的な長期の認証情報がありますが、ロールでは一時認証情報が提供されます。詳細については、「IAM ユーザーガイド」の「IAM ユーザーに関するユースケース」を参照してください。

IAM ロール

IAM ロールは、特定のアクセス許可 AWS アカウント を持つ 内の ID です。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。で IAM ロールを一時的に引き受けるには AWS Management Console、ユーザーから IAM ロール (コンソール) に切り替えることができます。ロールを引き受けるには、 または AWS API オペレーションを AWS CLI 呼び出すか、カスタム URL を使用します。ロールを使用する方法の詳細については、「IAM ユーザーガイド」の「ロールを引き受けるための各種方法」を参照してください。

IAM ロールと一時的な認証情報は、次の状況で役立ちます:

- フェデレーションユーザーアクセス – フェデレーティッド ID に許可を割り当てるには、ロールを作成してそのロールの許可を定義します。フェデレーティッド ID が認証されると、その ID はロールに関連付けられ、ロールで定義されている許可が付与されます。フェデレーションのロールについては、「IAM ユーザーガイド」の「サードパーティー ID プロバイダー (フェデレーション) 用のロールを作成する」を参照してください。IAM Identity Center を使用する場合は、許可セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity Center は、権限セットを IAM のロールに関連付けます。アクセス許可セットの詳細については、「AWS IAM Identity Center User Guide」の「Permission sets」を参照してください。
- 一時的な IAM ユーザー権限 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる権限を一時的に IAM ロールで引き受けることができます。

- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースにアクセスすることを、別のアカウントの人物 (信頼済みプリンシパル) に許可できます。クロスアカウントアクセス権を付与する主な方法は、ロールを使用することです。ただし、一部の では AWS のサービス、(プロキシとしてロールを使用する代わりに) リソースに直接ポリシーをアタッチできます。クロスアカウントアクセスにおけるロールとリソースベースのポリシーの違いについては、「IAM ユーザーガイド」の「IAM でのクロスアカウントのリソースへのアクセス」を参照してください。
- クロスサービスアクセス – 一部の は他の の機能 AWS のサービス を使用します AWS のサービス。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスでは、呼び出し元プリンシパルの許可、サービスロール、またはサービスリンクロールを使用してこれを行う場合があります。
  - 転送アクセスセッション (FAS) – IAM ユーザーまたはロールを使用してアクションを実行すると AWS、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、 を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストをリクエストする と組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。
  - サービスロール - サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除することができます。詳細については、「IAM ユーザーガイド」の「AWS のサービスに許可を委任するロールを作成する」を参照してください。
  - サービスにリンクされたロール – サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。IAM 管理者は、サービスリンクロールのアクセス許可を表示できますが、編集することはできません。
- Amazon EC2 で実行されているアプリケーション – IAM ロールを使用して、EC2 インスタンスで実行され、 AWS CLI または AWS API リクエストを行うアプリケーションの一時的な認証情報を管理できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。EC2 インスタンスに AWS ロールを割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスにアタッチされたインスタンスプロファイルを作成します。インスタンスプロ

ファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、「IAM ユーザーガイド」の「Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して許可を付与する」を参照してください。

ポリシーを使用したアクセスの管理

でアクセスを制御する AWS には、ポリシーを作成し、ID AWS またはリソースにアタッチします。ポリシーは AWS 、アイデンティティまたはリソースに関連付けられているときにアクセス許可を定義する のオブジェクトです。 は、プリンシパル (ユーザー、ルートユーザー、またはロールセッション) がリクエストを行うときに、これらのポリシー AWS を評価します。ポリシーでの権限により、リクエストが許可されるか拒否されるかが決まります。ほとんどのポリシーは JSON ドキュメント AWS として に保存されます。JSON ポリシードキュメントの構造と内容の詳細については、「IAM ユーザーガイド」の「JSON ポリシー概要」を参照してください。

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

デフォルトでは、ユーザーやロールに権限はありません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き受けることができます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの許可を定義します。例えば、`iam:GetRole` アクションを許可するポリシーがあるとします。そのポリシーを持つユーザーは、 AWS Management Console、、 AWS CLIまたは AWS API からロール情報を取得できます。

アイデンティティベースのポリシー

アイデンティティベースポリシーは、IAM ユーザーグループ、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティベースポリシーの作成方法については、「IAM ユーザーガイド」の「カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する」を参照してください。

アイデンティティベースのポリシーは、さらにインラインポリシーまたはマネージドポリシーに分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれています。管理ポリシーは、 内の複数のユーザー、グループ、ロールにアタッチできるスタンドアロンポリシーです AWS アカウント。管理ポリシーには、 AWS 管理ポリシーとカスタマー管理ポリシー

が含まれます。マネージドポリシーまたはインラインポリシーのいずれかを選択する方法については、「IAM ユーザーガイド」の「管理ポリシーとインラインポリシーのいずれかを選択する」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーでは、IAM の AWS マネージドポリシーを使用できません。

アクセスコントロールリスト (ACL)

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon S3、および Amazon VPC は AWS WAF、ACLs。ACL の詳細については、「Amazon Simple Storage Service デベロッパーガイド」の「アクセスコントロールリスト (ACL) の概要」を参照してください。

その他のポリシータイプ

AWS は、一般的でない追加のポリシータイプをサポートします。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- アクセス許可の境界 - アクセス許可の境界は、アイデンティティベースポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる権限の上限を設定する高度な機能です。エンティティにアクセス許可の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとそのアクセス許可の境界の共通部分になります。`Principal` フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、アクセス許可の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になりま

す。アクセス許可の境界の詳細については、「IAM ユーザーガイド」の「IAM エンティティのアクセス許可の境界」を参照してください。

- サービスコントロールポリシー (SCPs) – SCPsは、 の組織または組織単位 (OU) の最大アクセス許可を指定する JSON ポリシーです AWS Organizations。 AWS Organizations は、ビジネスが所有する複数の をグループ化して一元管理するためのサービス AWS アカウント です。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP は、各 を含むメンバーアカウントのエンティティのアクセス許可を制限します AWS アカウントのルートユーザー。Organizations と SCP の詳細については、「AWS Organizations ユーザーガイド」の「サービスコントロールポリシー (SCP)」を参照してください。
- リソースコントロールポリシー (RCP) – RCP は、所有する各リソースにアタッチされた IAM ポリシーを更新することなく、アカウント内のリソースに利用可能な最大数のアクセス許可を設定するために使用できる JSON ポリシーです。RCP は、メンバーアカウントのリソースのアクセス許可を制限し、組織に属しているかどうかにかかわらず AWS アカウントのルートユーザー、 を含む ID の有効なアクセス許可に影響を与える可能性があります。RCP をサポートする のリストを含む Organizations と RCP の詳細については、AWS Organizations RCPs「リソースコントロールポリシー (RCPs」を参照してください。 AWS のサービス
- セッションポリシー - セッションポリシーは、ロールまたはフェデレーションユーザーの一時的なセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果としてセッションの権限は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポリシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合もあります。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。詳細については、「IAM ユーザーガイド」の「セッションポリシー」を参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解するのがさらに難しくなります。が複数のポリシータイプが関与する場合にリクエストを許可するかどうか AWS を決定する方法については、「IAM ユーザーガイド」の「ポリシー評価ロジック」を参照してください。

AWS Elemental MediaStore が IAM と連動する方法

IAM を使用して MediaStore へのアクセスを管理する前に、MediaStore でどの IAM 機能が使用できるかを理解しておく必要があります。

AWS Elemental MediaStore で使用できる IAM 機能

IAM 機能	MediaStore のサポート
アイデンティティベースポリシー	はい
リソースベースのポリシー	はい
ポリシーアクション	はい
ポリシーリソース	はい
ポリシー条件キー (サービス固有)	はい
ACL	いいえ
ABAC (ポリシー内のタグ)	部分的
一時的な認証情報	はい
プリンシパル権限	はい
サービスロール	はい
サービスリンクロール	いいえ

MediaStore およびその他の AWS のサービスがほとんどの IAM 機能と連携する方法の概要については、「IAM ユーザーガイド」のAWS 「IAM と連携する のサービス」を参照してください。

MediaStore のアイデンティティベースのポリシー

アイデンティティベースのポリシーのサポート: あり

アイデンティティベースポリシーは、IAM ユーザーグループ、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。ID ベースのポリシーの作成方法については、「IAM ユーザーガイド」の「カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する」を参照してください。

IAM アイデンティティベースのポリシーでは、許可または拒否するアクションとリソース、およびアクションを許可または拒否する条件を指定できます。プリンシパルは、それが添付されている

ユーザーまたはロールに適用されるため、アイデンティティベースのポリシーでは指定できません。JSON ポリシーで使用できるすべての要素について学ぶには、「IAM ユーザーガイド」の「IAM JSON ポリシーの要素のリファレンス」を参照してください。

MediaStore のアイデンティティベースのポリシーの例

MediaStore のアイデンティティベースのポリシーの例については、「AWS Elemental MediaStore でのアイデンティティベースのポリシーの例」を参照してください。

MediaStore 内のリソースベースのポリシー

リソースベースのポリシーのサポート: あり

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

クロスアカウントアクセスを有効にするには、アカウント全体、または別のアカウントの IAM エンティティをリソースベースのポリシーのプリンシパルとして指定します。リソースベースのポリシーにクロスアカウントのプリンシパルを追加しても、信頼関係は半分しか確立されない点に注意してください。プリンシパルとリソースが異なる場合 AWS アカウント、信頼されたアカウントの IAM 管理者は、プリンシパルエンティティ (ユーザーまたはロール) にリソースへのアクセス許可も付与する必要があります。IAM 管理者は、アイデンティティベースのポリシーをエンティティにアタッチすることで権限を付与します。ただし、リソースベースのポリシーで、同じアカウントのプリンシパルへのアクセス権が付与されている場合は、アイデンティティベースのポリシーをさらに付与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」の「IAM でのクロスアカウントリソースアクセス」を参照してください。

> **Note**
>
> また、MediaStore は、コンテナに対してアクションを実行できるプリンシパルエンティティ (アカウント、ユーザー、ロール、フェデレーティッドユーザー) を定義するコンテナポリシーをサポートしています。詳細については、「コンテナポリシー」を参照してください。

MediaStore ポリシーアクション

ポリシーアクションのサポート:あり

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

JSON ポリシーの `Action` 要素にはポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。ポリシーアクションの名前は通常、関連付けられた AWS API オペレーションと同じです。一致する API オペレーションのない許可のみのアクションなど、いくつかの例外があります。また、ポリシーに複数のアクションが必要なオペレーションもあります。これらの追加アクションは依存アクションと呼ばれます。

このアクションは関連付けられたオペレーションを実行するためのアクセス許可を付与するポリシーで使用されます。

AWS Elemental MediaStore アクションのリストを確認するには、「サービス認可リファレンス」の「AWS Elemental MediaStore で定義されるアクション」を参照してください。

MediaStore のポリシーアクションは、アクションの前に次のプレフィックスを使用します：

```
mediastore
```

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、アクションをカンマで区切ります。

```
"Action": [
      "mediastore:action1",
      "mediastore:action2"
         ]
```

MediaStore のアイデンティティベースのポリシーの例については、「AWS Elemental MediaStore でのアイデンティティベースのポリシーの例」を参照してください。

MediaStore ポリシーリソース

ポリシーリソースのサポート: あり

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

Resource JSON ポリシー要素はアクションが適用されるオブジェクトを指定します。ステートメントにはResource または NotResource 要素を含める必要があります。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。これは、リソースレベルの許可と呼ばれる特定のリソースタイプをサポートするアクションに対して実行できます。

オペレーションのリスト化など、リソースレベルの権限をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

```
"Resource": "*"
```

MediaStore リソースのタイプとその ARN のリストを確認するには、「サービス認可リファレンス」の「AWS Elemental MediaStore で定義されるリソース」を参照してください。どのアクションで各リソースの ARN を指定できるかについては、「AWS Elemental MediaStore で定義されるアクション」を参照してください。

MediaStore コンテナリソースには次の ARN が含まれています。

```
arn:${Partition}:mediastore:${Region}:${Account}:container/${containerName}
```

ARN の形式の詳細については、「Amazon リソースネーム (ARNs AWS 「サービス名前空間」を参照してください。

たとえば、ステートメントで AwardsShow コンテナを指定するには、次の ARN を使用します。

```
"Resource": "arn:aws:mediastore:us-east-1:111122223333:container/AwardsShow"
```

MediaStore ポリシー条件キー

サービス固有のポリシー条件キーのサポート: あり

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

Condition 要素 (または Condition ブロック) を使用すると、ステートメントが有効な条件を指定できます。Condition 要素はオプションです。イコールや未満などの 条件演算子 を使用して条件式を作成して、ポリシーの条件とリクエスト内の値を一致させることができます。

1 つのステートメントに複数の Condition 要素を指定する場合、または 1 つの Condition 要素に複数のキーを指定する場合、 AWS では AND 論理演算子を使用してそれらを評価します。1 つの条件キーに複数の値を指定すると、 は論理ORオペレーションを使用して条件 AWS を評価します。ステートメントの権限が付与される前にすべての条件が満たされる必要があります。

条件を指定する際にプレースホルダー変数も使用できます。例えば IAM ユーザーに、IAM ユーザー名がタグ付けされている場合のみリソースにアクセスできる権限を付与することができます。詳細については、「IAM ユーザーガイド」の「IAM ポリシーの要素: 変数およびタグ」を参照してください。

AWS は、グローバル条件キーとサービス固有の条件キーをサポートしています。すべての AWS グローバル条件キーを確認するには、「IAM ユーザーガイド」のAWS 「グローバル条件コンテキストキー」を参照してください。

MediaStore の条件キーのリストを確認するには、「サービス認可リファレンス」の「AWS Elemental MediaStore の条件キー」を参照してください。どのアクションおよびリソースと条件キーを使用できるかについては、「AWS Elemental MediaStore で定義されるアクション」を参照してください。

MediaStore のアイデンティティベースのポリシーの例については、「AWS Elemental MediaStore でのアイデンティティベースのポリシーの例」を参照してください。

MediaStore での ACL

ACL のサポート: なし

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

MediaStore を使用した ABAC

ABAC (ポリシー内のタグ) のサポート: 一部

属性ベースのアクセス制御 (ABAC) は、属性に基づいてアクセス許可を定義する認可戦略です。では AWS、これらの属性はタグと呼ばれます。タグは、IAM エンティティ (ユーザーまたはロール) および多くの AWS リソースにアタッチできます。エンティティとリソースのタグ付けは、ABAC の最初

の手順です。その後、プリンシパルのタグがアクセスしようとしているリソースのタグと一致した場合にオペレーションを許可するように ABAC ポリシーをします。

ABAC は、急成長する環境やポリシー管理が煩雑になる状況で役立ちます。

タグに基づいてアクセスを管理するには、`aws:ResourceTag/key-name`、`aws:RequestTag/key-name`、または `aws:TagKeys` の条件キーを使用して、ポリシーの条件要素でタグ情報を提供します。

サービスがすべてのリソースタイプに対して 3 つの条件キーすべてをサポートする場合、そのサービスの値はありです。サービスが一部のリソースタイプに対してのみ 3 つの条件キーのすべてをサポートする場合、値は「部分的」になります。

ABAC の詳細については、「IAM ユーザーガイド」の「ABAC 認可でアクセス許可を定義する」を参照してください。ABAC をセットアップする手順を説明するチュートリアルについては、「IAM ユーザーガイド」の「属性ベースのアクセスコントロール (ABAC) を使用する」を参照してください。

MediaStore での一時認証情報の使用

一時的な認証情報のサポート: あり

一部の AWS のサービス は、一時的な認証情報を使用してサインインすると機能しません。一時的な認証情報 AWS のサービス を使用する場合などの詳細については、IAM ユーザーガイドAWS のサービス の「IAM と連携する 」を参照してください。

ユーザー名とパスワード以外の AWS Management Console 方法で にサインインする場合は、一時的な認証情報を使用します。たとえば、会社のシングルサインオン (SSO) リンク AWS を使用してにアクセスすると、そのプロセスによって一時的な認証情報が自動的に作成されます。また、ユーザーとしてコンソールにサインインしてからロールを切り替える場合も、一時的な認証情報が自動的に作成されます。ロールの切り替えに関する詳細については、「IAM ユーザーガイド」の「ユーザーから IAM ロールに切り替える (コンソール)」を参照してください。

一時的な認証情報は、 AWS CLI または AWS API を使用して手動で作成できます。その後、これらの一時的な認証情報を使用してアクセスすることができます AWS。長期的なアクセスキーを使用する代わりに、一時的な認証情報を動的に生成 AWS することをお勧めします。詳細については、「IAM の一時的セキュリティ認証情報」を参照してください。

MediaStore のクロスサービスプリンシパル許可

転送アクセスセッション (FAS) のサポート: あり

IAM ユーザーまたはロールを使用して でアクションを実行すると AWS、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、 を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストをリクエストする と組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。

MediaStore サービスロール

サービスロールのサポート: あり

サービスロールとは、サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、「IAM ユーザーガイド」の「AWS のサービスに許可を委任するロールを作成する」を参照してください。

> **Warning**
>
> サービスロールの許可を変更すると、MediaStore の機能が破損する可能性があります。MediaStore が指示する場合以外は、サービスロールを編集しないでください。

MediaStore サービスにリンクされたロール

サービスにリンクされたロールのサポート: なし

サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールのアクセス許可を表示できますが、編集することはできません。

サービスにリンクされたロールの作成または管理の詳細については、「IAM と提携するAWS のサービス」を参照してください。表の「サービスリンクロール」列に Yes と記載されたサービスを見つけます。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[はい] リンクを選択します。

AWS Elemental MediaStore でのアイデンティティベースのポリシーの例

デフォルトでは、IAM ユーザーおよびロールには、MediaStore リソースを作成または変更する権利はありません。また、、 AWS Command Line Interface （ AWS CLI ） AWS Management Console、または AWS API を使用してタスクを実行することはできません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き継ぐことができます。

これらサンプルの JSON ポリシードキュメントを使用して、IAM アイデンティティベースのポリシーを作成する方法については、「IAM ユーザーガイド」の「IAM ポリシーを作成する (コンソール)」を参照してください。

MediaStore が定義するアクションとリソースタイプ (リソースタイプごとの ARN の形式を含む) の詳細については、「サービス認可リファレンス」の「AWS Elemental MediaStore のアクション、リソース、および条件キー」を参照してください。

トピック

- ポリシーに関するベストプラクティス
- MediaStore コンソールを使用する
- 自分の権限の表示をユーザーに許可する

ポリシーに関するベストプラクティス

ID ベースのポリシーは、ユーザーのアカウントで誰が MediaStore リソースを作成、アクセス、または削除できるかを決定します。これらのアクションを実行すると、 AWS アカウントに料金が発生する可能性があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS 管理ポリシーを開始し、最小特権のアクセス許可に移行する – ユーザーとワークロードにアクセス許可の付与を開始するには、多くの一般的なユースケースにアクセス許可を付与するAWS 管理ポリシーを使用します。これらは で使用できます AWS アカウント。ユースケースに固有の AWS カスタマー管理ポリシーを定義することで、アクセス許可をさらに減らすことをお勧めします。詳細については、「IAM ユーザーガイド」の「AWS マネージドポリシー」または「ジョブ機能のAWS マネージドポリシー」を参照してください。
- 最小特権を適用する – IAM ポリシーで許可を設定する場合は、タスクの実行に必要な許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権アクセス許可とも呼ばれています。IAM を使用して許可を適用する

方法の詳細については、「IAM ユーザーガイド」の「IAM でのポリシーとアクセス許可」を参照してください。

- IAM ポリシーで条件を使用してアクセスをさらに制限する - ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。例えば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。条件を使用して、サービスアクションが などの特定の を通じて使用されている場合に AWS のサービス、サービスアクションへのアクセスを許可することもできます AWS CloudFormation。詳細については、「IAM ユーザーガイド」の「IAM JSON ポリシー要素:条件」を参照してください。
- IAM Access Analyzer を使用して IAM ポリシーを検証し、安全で機能的な権限を確保する - IAM Access Analyzer は、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、「IAM ユーザーガイド」の「IAM Access Analyzer でポリシーを検証する」を参照してください。
- 多要素認証 (MFA) を要求する – で IAM ユーザーまたはルートユーザーを必要とするシナリオがある場合は AWS アカウント、MFA をオンにしてセキュリティを強化します。API オペレーションが呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細については、「IAM ユーザーガイド」の「MFA を使用した安全な API アクセス」を参照してください。

IAM でのベストプラクティスの詳細については、「IAM ユーザーガイド」の「IAM でのセキュリティのベストプラクティス」を参照してください。

MediaStore コンソールを使用する

AWS Elemental MediaStore コンソールにアクセスするには、最小限のアクセス許可のセットが必要です。これらのアクセス許可により、 の MediaStore リソースの詳細を一覧表示および表示できます AWS アカウント。最小限必要な許可よりも制限が厳しいアイデンティティベースのポリシーを作成すると、そのポリシーを持つエンティティ (ユーザーまたはロール) に対してコンソールが意図したとおりに機能しません。

AWS CLI または AWS API のみを呼び出すユーザーには、最小限のコンソールアクセス許可を付与する必要はありません。代わりに、実行しようとしている API オペレーションに一致するアクションのみへのアクセスが許可されます。

ユーザーとロールが引き続き MediaStore コンソールを使用できるようにするには、MediaStore *ConsoleAccess*または *ReadOnly* AWS 管理ポリシーもエンティティにアタッチします。詳細については、「IAM ユーザーガイド」の「ユーザーへのアクセス許可の追加」を参照してください。

自分の権限の表示をユーザーに許可する

この例では、ユーザーアイデンティティにアタッチされたインラインおよびマネージドポリシーの表示を IAM ユーザーに許可するポリシーの作成方法を示します。このポリシーには、コンソールで、または AWS CLI または AWS API を使用してプログラムでこのアクションを実行するアクセス許可が含まれています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS Elemental MediaStore のアイデンティティとアクセスに関するトラブルシューティング

次の情報は、MediaStore と IAM の使用時に発生する可能性がある一般的な問題の診断と修正に役立ちます。

トピック

MediaStore でアクションを実行する権限がない

アクションを実行する権限がないというエラーが表示された場合は、そのアクションを実行できるようにポリシーを更新する必要があります。

次のエラー例は、mateojackson IAM ユーザーがコンソールを使用して、ある *my-example-widget* リソースに関する詳細情報を表示しようとしたことを想定して、その際に必要なmediastore:*GetWidget* アクセス許可を持っていない場合に発生するものです。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 mediastore:GetWidget on resource: my-example-widget
```

この場合、mediastore:*GetWidget* アクションを使用して *my-example-widget*リソースへのアクセスを許可するように、mateojackson ユーザーのポリシーを更新する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン認証情報を提供した担当者が管理者です。

iam:PassRole を実行する権限がありません

iam:PassRole アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更新して MediaStore にロールを渡すことができるようにする必要があります。

一部の AWS のサービス では、新しいサービスロールまたはサービスにリンクされたロールを作成する代わりに、そのサービスに既存のロールを渡すことができます。そのためには、サービスにロールを渡す権限が必要です。

以下の例のエラーは、marymajor という IAM ユーザーがコンソールを使用して MediaStore でアクションを実行しようする場合に発生します。ただし、このアクションをサービスが実行するには、サービスロールから付与された権限が必要です。メアリーには、ロールをサービスに渡す許可がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

この場合、Mary のポリシーを更新してメアリーに iam:PassRole アクションの実行を許可する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン資格情報を提供した担当者が管理者です。

自分の 以外のユーザーに MediaStore リソース AWS アカウント へのアクセスを許可したい

他のアカウントのユーザーや組織外の人が、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセスコントロールリスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください:

- MediaStore がこれらの機能をサポートしているかどうかを確認するには、「AWS Elemental MediaStore が IAM と連動する方法」を参照してください。
- 所有 AWS アカウント している のリソースへのアクセスを提供する方法については、IAM ユーザーガイドの「所有 AWS アカウント している別の の IAM ユーザーへのアクセスを提供する」を参照してください。
- リソースへのアクセスをサードパーティーに提供する方法については AWS アカウント、IAM ユーザーガイドの「サードパーティー AWS アカウント が所有する へのアクセスを提供する」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、「IAM ユーザーガイド」の「外部で認証されたユーザー (ID フェデレーション) へのアクセスの許可」を参照してください。
- クロスアカウントアクセスにおけるロールとリソースベースのポリシーの使用方法の違いについては、「IAM ユーザーガイド」の「IAM でのクロスアカウントのリソースへのアクセス」を参照してください。

でのログ記録とモニタリング AWS Elemental MediaStore

このセクションでは、セキュリティ上の目的で AWS Elemental MediaStore 内でログ記録およびモニタリングを行うためのオプションについての概要を説明します。MediaStore でのロギングおよびモニタリングの詳細については、「AWS Elemental MediaStore でのモニタリングとタグ付け」を参照してください。

モニタリングは、 および AWS Elemental MediaStore 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、その障害をより簡単に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必要があります。 には、MediaStore リソースをモニタリングし、潜在的なインシデントに対応するための複数のツール AWS が用意されています。

Amazon CloudWatch アラーム

Amazon CloudWatch アラームを使用して、指定した期間にわたって 1 つのメトリクスを確認します。メトリクスが特定のしきい値を超えると、Amazon SNS トピックまたは AWS Auto Scaling ポリシーに通知が送信されます。CloudWatch アラームは、特定の状態にあるという理由ではアクションを呼び出しません。その代わり、状態が変更され、指定期間にわたって維持される必要があります。詳細については、「CloudWatch によるモニタリング」を参照してください。

AWS CloudTrail ログ

CloudTrail は、 のユーザー、ロール、または AWS のサービスによって実行されたアクションの記録を提供します AWS Elemental MediaStore。CloudTrail が収集した情報を使用して、MediaStore に対して行われたリクエスト、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、およびその他の詳細情報を確認できます。詳細については、「CloudTrail による API コールのログ記録」を参照してください。

AWS Trusted Advisor

Trusted Advisor は、数十万人の AWS お客様にサービスを提供することから学んだベストプラクティスを活用しています。 は、AWS 環境 Trusted Advisor を検査し、コスト削減、システムの可用性とパフォーマンスの向上、セキュリティギャップの解消に役立つ機会があれば、レコメンデーションを行います。すべての AWS お客様は、5 つの Trusted Advisor チェックにアクセスできます。ビジネスまたはエンタープライズサポートプランをご利用のお客様は、すべての Trusted Advisor チェックを表示できます。

詳細については、「AWS Trusted Advisor」を参照してください。

AWS Elemental MediaStore 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範囲内」を参照して、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 「 Compliance ProgramsAssurance」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。 は、コンプライアンスに役立つ以下のリソース AWS を提供します。

- セキュリティのコンプライアンスとガバナンス – これらのソリューション実装ガイドでは、アーキテクチャ上の考慮事項について説明し、セキュリティとコンプライアンスの機能をデプロイする手順を示します。
- HIPAA 対応サービスのリファレンス – HIPAA 対応サービスの一覧が提供されています。すべての AWS のサービス が HIPAA の対象となるわけではありません。
- AWS コンプライアンスリソース – このワークブックとガイドのコレクションは、お客様の業界と地域に適用される場合があります。
- AWS カスタマーコンプライアンスガイド – コンプライアンスの観点から責任共有モデルを理解します。このガイドは、複数のフレームワーク (米国国立標準技術研究所 (NIST)、Payment Card Industry Security Standards Council (PCI)、国際標準化機構 (ISO) を含む) のセキュリティコントロールを保護し、そのガイダンスに AWS のサービス マッピングするためのベストプラクティスをまとめたものです。
- 「 デベロッパーガイド」の「ルールによるリソースの評価」 – この AWS Config サービスは、リソース設定が内部プラクティス、業界ガイドライン、および規制にどの程度準拠しているかを評価します。 AWS Config
- AWS Security Hub – これにより AWS のサービス 、 内のセキュリティ状態を包括的に把握できます AWS。Security Hub では、セキュリティコントロールを使用して AWS リソースを評価し、セキュリティ業界標準とベストプラクティスに対するコンプライアンスをチェックします。サポートされているサービスとコントロールの一覧については、Security Hub のコントロールリファレンスを参照してください。
- Amazon GuardDuty – 不審なアクティビティや悪意のあるアクティビティがないか環境をモニタリングすることで AWS アカウント、、ワークロード、コンテナ、データに対する潜在的な脅威 AWS のサービス を検出します。GuardDuty を使用すると、特定のコンプライアンスフレームワー

クで義務付けられている侵入検知要件を満たすことで、PCI DSS などのさまざまなコンプライアンス要件に対応できます。

- AWS Audit Manager – これにより AWS のサービス 、 AWS 使用状況を継続的に監査し、リスクの管理方法と規制や業界標準への準拠を簡素化できます。

AWS Elemental MediaStore での耐障害性

AWS グローバルインフラストラクチャは、 AWS リージョン およびアベイラビリティーゾーンを中心に構築されています。複数の物理的に分離および分離されたアベイラビリティーゾーン 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ています。アベイラビリティーゾーンでは、ゾーン間で中断することなく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が高く、フォールトトレラントで、スケーラブルです。

AWS リージョン および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。

MediaStore には、 AWS グローバルインフラストラクチャに加えて、データの耐障害性とバックアップのニーズをサポートするのに役立つ機能がいくつか用意されています。

AWS Elemental MediaStore で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 AWS Elemental MediaStore は、 AWS グローバルネットワークセキュリティで保護されています。 AWS セキュリティサービスと がインフラストラクチャ AWS を保護する方法については、AWS 「 クラウドセキュリティ」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して AWS 環境を設計するには、「Security Pillar AWS Well-Architected Framework」の「Infrastructure Protection」を参照してください。

AWS 公開された API コールを使用して、ネットワーク経由で MediaStore にアクセスします。クライアントは以下をサポートする必要があります。

- Transport Layer Security (TLS)。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- DHE (楕円ディフィー・ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。これらのモードはJava 7 以降など、ほとんどの最新システムでサポートされています。

また、リクエストにはアクセスキー ID と、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、テンポラリ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。

サービス間の混乱した代理の防止

混乱した代理問題は、アクションを実行するためのアクセス許可を持たないエンティティが、より特権のあるエンティティにアクションの実行を強制できてしまう場合に生じる、セキュリティ上の問題です。では AWS、サービス間のなりすましにより、混乱した代理問題が発生する可能性があります。サービス間でのなりすましは、1 つのサービス (呼び出し元サービス) が、別のサービス (呼び出し対象サービス) を呼び出すときに発生する可能性があります。呼び出し元サービスは、本来ならアクセスすることが許可されるべきではない方法でその許可を使用して、別のお客様のリソースに対する処理を実行するように操作される場合があります。これを防ぐため、 AWS では、アカウントのリソースへのアクセス権が付与されたサービスプリンシパルで、すべてのサービスのデータを保護するために役立つツールを提供しています。

リソースポリシー内で `aws:SourceArn` および `aws:SourceAccount` グローバル条件コンテキストキーを使用して、AWS Elemental MediaStore がそのリソースに対して別のサービスに付与する許可を制限することをお勧めします。クロスサービスアクセスにリソースを 1 つだけ関連付けたい場合は、`aws:SourceArn` を使用します。そのアカウント内のリソースをクロスサービスの使用に関連付けることを許可する場合は、`aws:SourceAccount` を使用します。

混乱した代理問題から保護するための最も効果的な方法は、リソースの完全な ARN を指定して、`aws:SourceArn` グローバル条件コンテキストキーを使用することです。リソースの完全な ARN が不明な場合や、複数のリソースを指定する場合には、グローバルコンテキスト条件キー `aws:SourceArn` で、ARN の未知部分を示すためにワイルドカード文字 (*) を使用します。例えば、`arn:aws:`*`servicename`*`:*:`*`123456789012`*`:*`。

`aws:SourceArn` の値に Amazon S3 バケット ARN などのアカウント ID が含まれていない場合は、両方のグローバル条件コンテキストキーを使用して、アクセス許可を制限する必要があります。

`aws:SourceArn` の値は、お使いのリージョンおよびアカウント内で、MediaStore が CloudWatch のログを発行する対象の設定である必要があります。

以下は、「混乱した代理」問題を防止するために、MediaStore で `aws:SourceArn` および `aws:SourceAccount` グローバル条件コンテキストキーを使用する方法の例です。

```
{
  "Version": "2012-10-17",
```

```
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "servicename.amazonaws.com"
    },
    "Action": "servicename:ActionName",
    "Resource": [
      "arn:aws:servicename:::ResourceName/*"
    ],
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:servicename:*:123456789012:*"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

AWS Elemental MediaStore でのモニタリングとタグ付け

モニタリングは、AWS Elemental MediaStore およびその他の 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。 には、MediaStore をモニタリングし、問題が発生したときに報告し、必要に応じて自動アクションを実行するための以下のモニタリングツール AWS が用意されています。

- AWS CloudTrail は、アカウントによって、または AWS アカウントに代わって行われた API コールおよび関連イベントをキャプチャし、指定した Amazon S3 バケットにログファイルを配信します。が呼び出したユーザーとアカウント AWS、呼び出し元のソース IP アドレス、および呼び出しの発生日時を特定できます。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 CloudWatch は、 AWS リソースと で実行されるアプリケーションを AWS リアルタイムでモニタリングします。メトリクスの収集と追跡、カスタマイズしたダッシュボードの作成、および指定したメトリクスが指定したしきい値に達したときに通知またはアクションを実行するアラームの設定を行うことができます。例えば、CloudWatch で Amazon EC2 インスタンスの CPU 使用率などのメトリクスを追跡し、必要に応じて新しいインスタンスを自動的に起動できます。詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。
- Amazon CloudWatch Events は、 AWS リソースの変更を記述するシステムイベントのストリームを提供します。通常、 AWS サービスは CloudWatch Events にイベント通知を数秒で配信しますが、1 分以上かかる場合があります。CloudWatch Events は、特定のイベントを監視し、これらのイベントが発生したときに他の AWS サービスで自動アクションをトリガーするルールを記述できるため、自動イベント駆動型コンピューティングを有効にします。詳細については、「Amazon CloudWatch Events ユーザーガイド」を参照してください。
- Amazon CloudWatch Logs では、Amazon EC2 インスタンス、CloudTrail、およびその他のソースからのログファイルをモニタリング、保存、およびアクセスできます。CloudWatch Logs は、ログファイル内の情報をモニタリングし、特定のしきい値が満たされたときに通知します。高い耐久性を備えたストレージにログデータをアーカイブすることも可能です。詳細については、「Amazon CloudWatch Logs ユーザーガイド」を参照してください。

また、MediaStore コンテナにメタデータをタグ形式で割り当てることもできます。各タグは、お客様が定義するキーと値で構成されるラベルです。タグを使用することで、リソースの管理、検索、フィルタリングを行うことができます。タグを使用して、 AWS マネジメントコンソールで AWS リソースを整理し、すべてのリソースで使用状況と請求レポートを作成し、インフラストラクチャの自動化アクティビティ中にリソースを AWS フィルタリングできます。

トピック

AWS CloudTrailを使用した AWS Elemental MediaStore API コールのログ記録

AWS Elemental MediaStore は AWS CloudTrail、MediaStore のユーザー、ロール、または のサービスによって実行されたアクションを記録する AWS サービスである と統合されています。CloudTrail は、MediaStore コンソールからの呼び出しや MediaStore API へのコード呼び出しを含め、MediaStore の API コールのサブセットをイベントとしてキャプチャします。証跡を作成する場合は、Amazon S3 バケットへの CloudTrail イベント (MediaStore のイベントを含む) の継続的な配信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 で収集した情報を使用して、MediaStore に対して行われたリクエスト、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。

トピック

CloudTrail 内の AWS Elemental MediaStore 情報

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。AWS Elemental MediaStore 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 CloudTrail イベントとイベント履歴の他の AWS サービス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。

MediaStore 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、ログファイルを CloudTrail で 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、すべての AWS リージョンに証跡が適用されます。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口グに記録し、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するように他の AWS サービスを設定できます。詳細については、以下の各トピック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 口グファイルを受け取るおよび複数のアカウントから CloudTrail 口グファイルを受け取る

AWS Elemental MediaStore は CloudTrail 口グファイルのイベントとして次のオペレーションを記録します。

- CreateContainer
- DeleteContainer
- DeleteContainerPolicy
- DeleteCorsPolicy
- DescribeContainer
- GetContainerPolicy
- GetCorsPolicy
- ListContainers
- PutContainerPolicy
- PutCorsPolicy

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。

- ルートユーザーまたはユーザー認証情報のどちらを使用してリクエストが送信されたか
- リクエストが、ロールとフェデレーティッドユーザーのどちらの一時的なセキュリティ認証情報を使用して送信されたか
- リクエストが別の AWS サービスによって行われたかどうか

詳細については、「CloudTrail userIdentity 要素」を参照してください。

例: AWS Elemental MediaStore 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。イベントはあらゆるソース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、リクエストのパラメータなどの情報が含まれます。CloudTrail ログファイルは、公開 API コール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。

CreateContainerオペレーションを示すCloudTrailログエントリの例は、次のとおりです。

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "ABCDEFGHIJKL123456789",
        "arn": "arn:aws:iam::111122223333:user/testUser",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "testUser",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2018-07-09T12:55:42Z"
            }
        },
        "invokedBy": "signin.amazonaws.com"
    },
    "eventTime": "2018-07-09T12:56:54Z",
    "eventSource": "mediastore.amazonaws.com",
    "eventName": "CreateContainer",
    "awsRegion": "ap-northeast-1",
    "sourceIPAddress": "54.239.119.16",
    "userAgent": "signin.amazonaws.com",
    "requestParameters": {
        "containerName": "TestContainer"
    },
    "responseElements": {
        "container": {
            "status": "CREATING",
            "creationTime": "Jul 9, 2018 12:56:54 PM",
            "name": " TestContainer ",
            "aRN": "arn:aws:mediastore:ap-northeast-1:111122223333:container/
TestContainer"
```

```
        }
    },
    "requestID":
 "MNCTGH4HRQJ27GRMBVDPIVHEP4LO2BN6MUVHBCPSHOAWNSOKSXCO24B2UEOBBND5DONRXTMFK3TOJ4G7AHWMESI",
    "eventID": "7085b140-fb2c-409b-a329-f567912d704c",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

Amazon CloudWatch による AWS Elemental MediaStore のモニタリング

raw データを収集して読み取り可能なメトリクスに処理する CloudWatch を使用して AWS Elemental MediaStore をモニタリングできます。これらの統計は 15 か月間保持されるため、履歴情報にアクセスし、ウェブアプリケーションまたはサービスの動作をより的確に把握できます。また、特定のしきい値を監視するアラームを設定し、これらのしきい値に達したときに通知を送信したりアクションを実行したりできます。詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。

AWS には、MediaStore を監視し、問題が発生したときに報告し、必要に応じて自動アクションを実行するための以下のモニタリングツールが用意されています。

- Amazon CloudWatch Logs を使用すると、AWS Elemental MediaStore などの AWS サービスのログファイルをモニタリング、保存し、このログファイルにアクセスすることができます。CloudWatch Logs を使用すると、ログデータを使用してアプリケーションやシステムをモニタリングできます。例えば、CloudWatch Logs では、アプリケーションログに存在するエラーの数をトラッキングし、エラー率が指定のしきい値を超えたときに管理者に通知を送信することができます。ログデータが CloudWatch Logs によるモニタリングに使用されるので、コードの変更は不要です。たとえば、特定のリテラル用語 (「ValidationException」など) がアプリケーションログに含まれていないかを監視したり、特定の期間中に行われた `PutObject` リクエストの数をカウントしたりすることができます。検索した語句が見つかると、CloudWatch Logs は指定された CloudWatch メトリクスにデータをレポートします。ログデータは、転送時や保管時に暗号化されます。
- Amazon CloudWatch Events は、MediaStore オブジェクトなどの AWS リソースの変更を記述するシステムイベントを配信します。通常、 AWS サービスは CloudWatch Events にイベント通知を数秒で配信しますが、1 分以上かかる場合があります。このようなイベント (`DeleteObject` リクエストなど) に一致するルールを設定して、これらのイベントを 1 つ以上のターゲット関数またはストリームにルーティングできます。CloudWatch Events が発生すると、運用上の変更が認識

されます。また、CloudWatch Events は、これらのオペレーションの変更に応答し、必要に応じて、応答メッセージを環境に送り、機能をアクティブ化し、変更を行い、状態情報を収集することによって、修正アクションを実行します。

CloudWatch Logs

アクセスのログ記録には、コンテナ内でオブジェクトに対して行われたリクエストの詳細が記録されます。アクセスログは、セキュリティやアクセスの監査などの多くのアプリケーションに役立ちます。また、顧客基盤について知り、MediaStore の請求を理解することにも役立ちます。CloudWatch Logs は次のように分類されます。

- ログストリームは、同じソースを共有する一連のログイベントです。
- ロググループは、保持、モニタリング、アクセス制御について同じ設定を共有するログストリームのグループです。コンテナでアクセスのログ記録を有効にすると、`/aws/mediastore/MyContainerName` などの名前のロググループが MediaStore によって作成されます。ロググループを定義して、各グループに入れるストリームを指定できます。1 つのロググループに属することができるログストリームの数にクォータはありません。

デフォルトでは、ログは無制限に保持され、失効しません。ロググループごとに保持ポリシーを調整し、無制限の保持期間を維持するか、1 日間 ~ 10 年間の保持期間を選択することができます。

Amazon CloudWatch のアクセス許可のセットアップ

AWS Identity and Access Management (IAM) を使用して、AWS Elemental MediaStore に Amazon CloudWatch へのアクセスを許可するロールを作成します。アカウントに対して CloudWatch Logs を公開するには、これらのステップを実行する必要があります。CloudWatch は、お使いのアカウントに関するメトリクスを自動的に発行します。

MediaStore に CloudWatch へのアクセスを許可するには

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. IAM コンソールのナビゲーションペインで、[ポリシー] を選択し、[ポリシーの作成] を選択します。
3. [JSON] タブを選択し、以下のポリシーを貼り付けます。

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups",
                "logs:CreateLogGroup"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/mediastore/*"
        }
    ]
}
```

このポリシーにより、MediaStore は AWS アカウント内の任意のリージョンのコンテナのロググループとログストリームを作成できます。

4. [ポリシーの確認] を選択します。
5. [Review policy] (ポリシーの確認) ページで、[Name] (名前) に「**MediaStoreAccessLogsPolicy**」と入力し、[Create policy] (ポリシーの作成) を選択します。
6. 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。
7. [別の AWS アカウント] ロールタイプを選択します。
8. アカウント ID には、 AWS アカウント ID を入力します。
9. [Next: Permissions] (次のステップ: 許可) を選択します。
10. 検索ボックスに「**MediaStoreAccessLogsPolicy**」と入力します。
11. 新しいポリシーの横にあるチェックボックスをオンにし、[Next: Tags (次の手順: タグ)] を選択します。
12. [Next: Review (次の手順: 確認)] を選択し、新しいユーザーをプレビューします。
13. [ロール名] に「**MediaStoreAccessLogs**」と入力し、[ロールの作成] を選択します。
14. 確認メッセージで、作成したロールの名前 (**MediaStoreAccessLogs**) を選択します。

15. ロールの [概要] ページで、[Trust relationship (信頼関係)] タブを選択します。
16. [Edit trust relationship (信頼関係の編集)] を選択します。
17. ポリシードキュメントで、プリンシパルを MediaStore サービスに変更します。プリンシパルは以下のようになります。

```
"Principal": {
   "Service": "mediastore.amazonaws.com"
},
```

ポリシー全体は以下のようになります。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "mediastore.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {}
    }
  ]
}
```

18. [Update Trust Policy] (信頼ポリシーの更新) をクリックします。

コンテナのアクセスログ記録の有効化

デフォルトでは、AWS Elemental MediaStore はアクセスログを収集しません。コンテナでアクセスのログ記録を有効にすると、MediaStore ではそのコンテナに保存されているオブジェクトのアクセスログを Amazon CloudWatch に配信します。アクセスログには、コンテナに保存されているすべてのオブジェクトに対して行われたリクエストの詳細が記録されます。この情報には、リクエストタイプ、リクエストで指定したリソース、リクエストを処理した日時などが含まれます。

> **Important**
>
> MediaStore コンテナでアクセスのログ記録を有効にしても追加料金はかかりません。ただし、サービスが配信するいずれのログファイルの格納に対しても通常の料金がかかります

(ログファイルはいつでも削除できます)。 AWS のログファイルの配信に対してはデータ転送料金はかかりませんが、ログファイルへのアクセスに対しては通常のデータ転送料金がかかります。

アクセスのログ記録を有効にするには (AWS CLI)

- で AWS CLI、 `start-access-logging` コマンドを使用します。

```
aws mediastore start-access-logging --container-name LiveEvents --region us-west-2
```

このコマンドの戻り値はありません。

コンテナのアクセスログ記録の無効化

コンテナでアクセスのログ記録を無効にすると、AWS Elemental MediaStore が Amazon CloudWatch へのアクセスログの送信を停止します。これらのアクセスログは保存されないため、取り出せなくなります。

アクセスのログ記録を無効にするには (AWS CLI)

- で AWS CLI、 `stop-access-logging` コマンドを使用します。

```
aws mediastore stop-access-logging --container-name LiveEvents --region us-west-2
```

このコマンドの戻り値はありません。

AWS Elemental MediaStore でのアクセスのログ記録のトラブルシューティング

AWS Elemental MediaStore のアクセスログが Amazon CloudWatch に表示されない場合は、以下の表に記載されている考えられる原因と解決策を参照してください。

> **Note**
>
> AWS CloudTrail Logs を有効にして、トラブルシューティングプロセスに役立ててください。

症状	考えられる原因	解決策
CloudTrail ログが有効になっていても、CloudTrail イベントがまったく表示されない。	IAM ロールが存在しないか、正しくない名前、アクセス許可、または信頼ポリシーが含まれています。	正しい名前、アクセス許可、信頼ポリシーを使用してロールを作成します。「the section called “CloudWatch のアクセス許可のセットアップ”」を参照してください。
`DescribeContainer` API リクエストを送信したが、`AccessLoggingEnabled` パラメータに `False` の値が含まれていることがレスポンスに示されている。また、`MediaStoreAccessLogs` ロールが `DescribeLogGroup` 、`CreateLogGroup` 、`DescribeLogStream` 、または `CreateLogStream` の呼び出しに成功したことを示す CloudTrail イベントがまったく表示されない。	IAM ロールが存在しないか、正しくない名前、アクセス許可、または信頼ポリシーが含まれています。	正しい名前、アクセス許可、信頼ポリシーを使用してロールを作成します。「the section called “CloudWatch のアクセス許可のセットアップ”」を参照してください。
	アクセスのログ記録がコンテナで有効になっていません。	コンテナのアクセスログを有効にします。「the section called “アクセスログ記録の有効化”」を参照してください。

CloudTrail コンソールに、`MediaStoreAccessLogs` ロールに関連するアクセス拒否エラーを含むイベントが表示される。CloudTrail イベントに、次のような行が含まれている場合がある。
 	
`"eventSource": "logs.amazonaws.com",`
 	
`"errorCode": "AccessDenied",`
 	
`"errorMessage": "User: arn:aws:sts::111122223333:assumed-role/MediaStoreAccessLogs/`
	IAM ロールに AWS Elemental MediaStore に対する正しいアクセス許可がありません。	適切なアクセス許可と信頼ポリシーを持つように IAM ロールを更新します。「the section called “CloudWatch のアクセス許可のセットアップ”」を参照してください。

症状	考えられる原因	解決策
`MediaStoreAccessLogsSession is not authorized to perform: logs:DescribeLogGroups on resource: arn:aws:logs:us-west-2:111122223333:log-group::log-stream:",`		
1 つのコンテナまたは複数のコンテナのログが表示されない。	お客様のアカウントが、1 アカウント、1 リージョンあたりのロググループに対する CloudWatch のクォータを超えている可能性があります。「Amazon CloudWatch Logs ユーザーガイド」のロググループのクォータを参照してください。	CloudWatch コンソールで、アカウントがロググループの CloudWatch のクォータに達しているかどうかを判断します。必要に応じて、クォータの引き上げをリクエストします。
CloudWatch に、予期しているすべてのログではなく、一部のログしか表示されない。	お客様のアカウントが、1 秒、1 アカウント、1 リージョンあたりのトランザクションの CloudWatch のクォータを超えている可能性があります。「Amazon CloudWatch Logs ユーザーガイド」で `PutLogEvents` のクォータを参照してください。	1 秒、1 アカウント、1 リージョンあたりの CloudWatch トランザクションのクォータの引き上げをリクエストします。

アクセスログの形式

アクセスログファイルは、一連の JSON 形式のログレコードで構成されており、各ログレコードは 1 つのリクエストを表します。ログ内のフィールドの順序は変わることがあります。2 つのログレコードで構成されるログの例を次に示します。

```
{
  "Path": "/FootballMatch/West",
  "Requester": "arn:aws:iam::111122223333:user/maria-garcia",
  "AWSAccountId": "111122223333",
  "RequestID":
 "aaaAAA111bbbBBB222cccCCC333dddDDD444eeeEEE555fffFFF666gggGGG777hhhHHH888iiiIII999jjjJJJ",
  "ContainerName": "LiveEvents",
  "TotalTime": 147,
  "BytesReceived": 1572864,
  "BytesSent": 184,
  "ReceivedTime": "2018-12-13T12:22:06.245Z",
  "Operation": "PutObject",
  "ErrorCode": null,
  "Source": "192.0.2.3",
  "HTTPStatus": 200,
  "TurnAroundTime': 7,
  "ExpiresAt": "2018-12-13T12:22:36Z"
}
 {
  "Path": "/FootballMatch/West",
  "Requester": "arn:aws:iam::111122223333:user/maria-garcia",
  "AWSAccountId": "111122223333",
  "RequestID":
 "dddDDD444eeeEEE555fffFFF666gggGGG777hhhHHH888iiiIII999jjjJJJ000cccCCC333bbbBBB222aaaAAA",
  "ContainerName": "LiveEvents",
  "TotalTime": 3,
  "BytesReceived": 641354,
  "BytesSent": 163,
  "ReceivedTime": "2018-12-13T12:22:51.779Z",
  "Operation": "PutObject",
  "ErrorCode": "ValidationException",
  "Source": "198.51.100.15",
  "HTTPStatus": 400,
  "TurnAroundTime": 1,
  "ExpiresAt": null
}
```

次のリストは、ログレコードのフィールドについて説明しています。

AWSAccountId

リクエストの実行に使用された AWS アカウントのアカウント ID。

BytesReceived

MediaStore サーバーが受信するリクエストボディのバイト数。

BytesSent

MediaStore サーバーが送信するレスポンス本文のバイト数。この値は、多くの場合、サーバーレスポンスに含まれている `Content-Length` ヘッダーの値と同じです。

ContainerName

リクエストを受信したコンテナの名前。

ErrorCode

MediaStore のエラーコード (`InternalServerError` など)。エラーが発生しなかった場合は、- の文字が表示されます。ステータスコードが 200 (接続が閉じられているか、サーバーがレスポンスのストリーミングを開始した後にエラーが発生したことを示す) であってもエラーコードが表示される場合があります。

ExpiresAt

オブジェクトの有効期限の日時。この値は、コンテナに適用されるライフサイクルポリシーに含まれている transient data rule によって設定された有効期限に基づいています。この値は `ISO-8601` の日時で、リクエストに対応したホストのシステムクロックに基づいています。ライフサイクルポリシーにオブジェクトに適用される一時的なデータルールが含まれていない場合、またはコンテナに適用されるライフサイクルポリシーがない場合、このフィールドの値は `null` です。このフィールドは、`PutObject`、`GetObject`、`DescribeObject`、および `DeleteObject` の各オペレーションにのみ適用されます。

HTTPStatus

レスポンスの HTTP ステータスの数値。

Operation

`PutObject` や `ListItems` などの、実行されたオペレーション。

パス

オブジェクトが保存されているコンテナ内のパス。オペレーションがパスのパラメータを使用しない場合は、- の文字が表示されます。

ReceivedTime

リクエストを受け取った時刻。この値は ISO-8601 の日時で、リクエストに対応したホストのシステムクロックに基づいています。

リクエスタ

リクエストを行うために使用されたアカウントのユーザーの Amazon リソースネーム (ARN)。認証されていないリクエストの場合、この値は anonymous になります。認証が完了する前にリクエストが失敗した場合は、このフィールドがログに表示されない可能性があります。このようなリクエストでは、ErrorCode で認可の問題を特定できる場合があります。

RequestID

各リクエストを一意に識別するために AWS Elemental MediaStore で生成される文字列。

ソース

呼び出しを行った AWS サービスのリクエスタまたはサービスプリンシパルの表面上のインターネットアドレス。中間プロキシやファイアウォールにより、リクエストを作成したマシンのアドレスが不明確になる場合、値は null に設定されます。

TotalTime

サーバーから見た、リクエストの転送中の時間数 (ミリ秒単位)。これは、サービスがリクエストを受信してから、レスポンスの最終バイトが送信されるまでの時間を計測した値です。この値は、サーバーの観点から計測されます。クライアント側の観点で計測された値は、ネットワークレイテンシーの影響を受けるためです。

TurnAroundTime

MediaStore でリクエストの処理に要した時間数 (ミリ秒単位)。これは、リクエストの最終バイトが受信されてから、レスポンスの先頭バイトが送信されるまでの時間を計測した値です。

ログのフィールドの順序は変わることがあります。

ログ記録ステータスの変更が有効になるまでの期間

コンテナのログ記録ステータスの変更がログファイルの配信に反映されるまでには時間がかかります。たとえば、コンテナ A のログ記録を有効にした場合、その後数時間に行われるリクエストは記

録されることもあれば、されないこともあります。コンテナ B のログ記録を無効にした場合、その後数時間はログが引き続き配信されることもあれば、されないこともあります。いずれの場合も、最終的には新しい設定が有効になるため、追加の操作は不要です。

ベストエフォート型のサーバーログ配信

アクセスログレコードの配信は、ベストエフォートで行われます。コンテナがログ記録用に適切に設定されている場合、そのコンテナへのほとんどのリクエストについてログレコードが配信されます。ほとんどのログレコードは、記録された時間から数時間以内に配信されますが、配信間隔は短くなる場合もあります。

アクセスのログ記録の完全性や適時性は保証されません。リクエストのログレコードが、リクエストが実際に処理されてからかなり後に配信されたり、配信すらされなかったりすることもあり得ます。アクセスログの目的は、コンテナに対するトラフィックの特性を理解することです。ログレコードが失われることはまれですが、すべてのリクエストが完全に報告されるとは限りません。

アクセスのログ記録機能はベストエフォート型であるため、AWS ポータルで利用できる使用状況レポート (AWS Management Console でレポートされる請求およびコスト管理レポート) には、アクセスログに記録されていないアクセスリクエストが含まれる場合があります。

アクセスログの形式のプログラミングに関する考慮事項

新しいフィールドを追加することで、アクセスログの形式を随時拡張する場合があります。アクセスログを解析するコードは、追加のフィールドを理解できなくても処理するよう作成する必要があります。

CloudWatch Events

Amazon CloudWatch Events を使用すると、 AWS サービスを自動化し、アプリケーションの可用性の問題やリソースの変更などのシステムイベントに自動的に対応できます。簡単なルールを記述して、注目するイベントと、イベントがルールに一致した場合に自動的に実行するアクションを指定できます。

> **Important**
>
> 通常、 AWS サービスは CloudWatch Events にイベント通知を数秒で配信しますが、1 分以上かかる場合があります。

ファイルをコンテナにアップロードするか、コンテナから削除すると、CloudWatch サービスで 2 つのイベントが連続して発生します。

1. the section called “オブジェクト状態の変更イベント”
2. the section called “コンテナ状態の変更イベント”

これらのイベントにサブスクライブする方法については、Amazon CloudWatch を参照してください。

自動的にトリガーできるオペレーションには、以下が含まれます。

- AWS Lambda 関数の呼び出し
- Amazon EC2 Run Command の呼び出し
- Amazon Kinesis Data Streams へのイベントの中継
- AWS Step Functions ステートマシンのアクティブ化
- Amazon SNS トピックまたは AWS SMS キューの通知

AWS Elemental MediaStore で CloudWatch Events を使用する例をいくつか次に示します。

- コンテナが作成されるたびに Lambda 関数をアクティブ化する。
- オブジェクトが削除されたときに Amazon SNS トピックに通知する。

詳細については、「Amazon CloudWatch Events ユーザーガイド」を参照してください。

トピック

- AWS Elemental MediaStore オブジェクト状態の変更イベント
- AWS Elemental MediaStore コンテナ状態の変更イベント

AWS Elemental MediaStore オブジェクト状態の変更イベント

このイベントは、オブジェクトの状態が変わると (オブジェクトがアップロードまたは削除されると)、発行されます。

> **Note**
>
> 一時的なデータルールのために期限切れになるオブジェクトは、有効期限が切れたときに CloudWatch イベントを発行しません。

このイベントにサブスクライブする方法については、Amazon CloudWatch を参照してください。

オブジェクトの更新

```
{
  "version": "1",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "MediaStore Object State Change",
  "source": "aws.mediastore",
  "account": "111122223333",
  "time": "2017-02-22T18:43:48Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:mediastore:us-east-1:111122223333:MondayMornings/Episode1/
Introduction.avi"
  ],
  "detail": {
    "ContainerName": "Movies",
    "Operation": "UPDATE",
    "Path":"TVShow/Episode1/Pilot.avi",
    "ObjectSize":123456,
    "URL": "https://a832p1qeaznlp9.files.mediastore-us-west-2.com/Movies/
MondayMornings/Episode1/Introduction.avi"
  }
}
```

オブジェクトの削除

```
{
  "version": "1",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "MediaStore Object State Change",
  "source": "aws.mediastore",
  "account": "111122223333",
  "time": "2017-02-22T18:43:48Z",
  "region": "us-east-1",
```

```
  "resources": [
    "arn:aws:mediastore:us-east-1:111122223333:Movies/MondayMornings/Episode1/
Introduction.avi"
  ],
  "detail": {
    "ContainerName": "Movies",
    "Operation": "REMOVE",
    "Path":"Movies/MondayMornings/Episode1/Introduction.avi",
    "URL": "https://a832p1qeaznlp9.files.mediastore-us-west-2.com/Movies/
MondayMornings/Episode1/Introduction.avi"
  }
}
```

AWS Elemental MediaStore コンテナ状態の変更イベント

このイベントは、コンテナの状態が変わると (コンテナが追加または削除されると)、発行されます。このイベントにサブスクライブする方法については、Amazon CloudWatch を参照してください。

コンテナの作成

```
{
  "version": "1",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "MediaStore Container State Change",
  "source": "aws.mediastore",
  "account": "111122223333",
  "time": "2017-02-22T18:43:48Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:mediastore:us-east-1:111122223333:container/Movies"
  ],
  "detail": {
    "ContainerName": "Movies",
    "Operation": "CREATE"
    "Endpoint": "https://a832p1qeaznlp9.mediastore-us-west-2.amazonaws.com"
  }
}
```

コンテナの削除

```
{
  "version": "1",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
```

```
  "detail-type": "MediaStore Container State Change",
  "source": "aws.mediastore",
  "account": "111122223333",
  "time": "2017-02-22T18:43:48Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:mediastore:us-east-1:111122223333:container/Movies"
  ],
  "detail": {
    "ContainerName": "Movies",
    "Operation": "REMOVE"
  }
}
```

Amazon CloudWatch メトリクスによる AWS Elemental MediaStore のモニタリング

raw データを収集して読み取り可能なメトリクスに処理する CloudWatch を使用して AWS Elemental MediaStore をモニタリングできます。これらの統計は 15 か月間保持されるため、履歴情報にアクセスし、ウェブアプリケーションまたはサービスの動作をより的確に把握できます。また、特定のしきい値を監視するアラームを設定し、これらのしきい値に達したときに通知を送信したりアクションを実行したりできます。詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。

AWS Elemental MediaStore では、`BytesDownloaded` を確認して、メトリクスが特定のしきい値に達したときに、自分自身に E メールを送信することができます。

CloudWatch コンソールを使用してメトリクスを表示するには

メトリクスはまずサービスの名前空間ごとにグループ化され、次に各名前空間内のさまざまなディメンションの組み合わせごとにグループ化されます。

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで [Metrics (メトリクス)] を選択してください。
3. [All metrics (すべてのメトリクス)] で、[AWS/MediaStore] 名前空間を選択します。
4. メトリクスディメンションを選択して、メトリクスを表示します。たとえば、`Request metrics by container` を選択して、コンテナに送信されたさまざまなタイプのリクエストのメトリクスを表示します。

を使用してメトリクスを表示するには AWS CLI

- コマンドプロンプトで、次のコマンドを使用します。

```
aws cloudwatch list-metrics --namespace "AWS/MediaStore"
```

AWS Elemental MediaStore メトリクス

次の表に、AWS Elemental MediaStore が CloudWatch に送信するメトリクスを示します。

> **Note**
>
> メトリクスを表示するには、コンテナにメトリクスポリシーを追加して、MediaStore がメトリクスを Amazon CloudWatch に送信できるようにする必要があります。

メトリクス	説明
RequestCount	MediaStore コンテナに対して行われた HTTP リクエストの総数。オペレーションタイプ (Put、Get、Delete、Describe、List) で分けられます。 単位: カウント 有効なディメンション: • コンテナ名 • オブジェクトグループ名 • リクエストタイプ 有効な統計: Sum
4xxErrorCount	4xx エラーを発生させた、MediaStore に対して行われた HTTP リクエストの数。 単位: カウント 有効なディメンション:

メトリクス	説明
	• コンテナ名 • オブジェクトグループ名 • リクエストタイプ 有効な統計: Sum
5xxErrorCount	5xx エラーを発生させた、MediaStore に対して行われた HTTP リクエストの数。 単位: カウント 有効なディメンション: • コンテナ名 • オブジェクトグループ名 • リクエストタイプ 有効な統計: Sum
BytesUploaded	MediaStore コンテナに対して行われたリクエストに対してアップロードされたバイト数。リクエストには本文が含まれます。 単位: バイト 有効なディメンション: • コンテナ名 • オブジェクトグループ名 有効な統計: Average (リクエストあたりのバイト数)、Sum (期間あたりのバイト数)、Sample Count、Min (P0.0 と同じ)、Max (p100 と同じ)、p0.0 ~ p99.9 のパーセンタイル

メトリクス	説明
BytesDownloaded	MediaStore コンテナに対して行われたリクエストに対してダウンロードされたバイト数。レスポンスには本文が含まれます。 単位: バイト 有効なディメンション: • コンテナ名 • オブジェクトグループ名 有効な統計: Average (リクエストあたりのバイト数)、Sum (期間あたりのバイト数)、Sample Count、Min (P0.0 と同じ)、Max (p100 と同じ)、p0.0 ~ p99.9 のパーセンタイル
TotalTime	サーバーから見た、リクエストの転送中の時間数 (ミリ秒単位)。この値は、MediaStore がリクエストを受信してから、レスポンスの最終バイトを送信するまでの時間を計測した値です。この値は、サーバーの観点から計測されます。クライアント側の観点で計測された値は、ネットワークレイテンシーの影響を受けるためです。 単位: ミリ秒 有効なディメンション: • コンテナ名 • オブジェクトグループ名 • リクエストタイプ 有効な統計情報: Average、Min (P0.0 と同じ)、Max (p100 と同じ)、p0.0 と p100 の間のパーセンタイル

メトリクス	説明
TurnaroundTime	MediaStore でリクエストの処理に要した時間数 (ミリ秒単位)。この値は、MediaStore がリクエストの最終バイトを受信してから、レスポンスの先頭バイトを送信するまでの時間を計測した値です。 単位: ミリ秒 有効なディメンション: • コンテナ名 • オブジェクトグループ名 • リクエストタイプ 有効な統計情報: Average、Min (P0.0 と同じ)、Max (p100 と同じ)、p0.0 と p100 の間のパーセンタイル
ThrottleCount	MediaStore に対して行われた、スロットリングされた HTTP リクエストの数。 単位: カウント 有効なディメンション: • コンテナ名 • オブジェクトグループ名 • リクエストタイプ 有効な統計: Sum

AWS Elemental MediaStore リソースのタグ付け

タグは、ユーザーが割り当てるか、 AWS リソース AWS に割り当てるカスタム属性ラベルです。各タグは 2 つの部分で構成されます。

- タグキー (例： CostCenter、Environment、または Project)。タグキーでは、大文字と小文字が区別されます。

- タグ値として知られるオプションのフィールド (例： `111122223333` または `Production`)。タグ値を省略すると、空の文字列を使用した場合と同じになります。タグキーと同様に、タグ値では大文字と小文字が区別されます。

タグは、以下のことに役立ちます。

- AWS リソースを特定して整理します。多くの AWS のサービスではタグ付けがサポートされるため、さまざまなサービスまでリソースの関連を示すことができリソースに同じタグを割り当てることができます。例えば、 AWS Elemental MediaLive の入力に割り当てるのと同じタグを AWS Elemental MediaStore の####に割り当てることができます。
- AWS のコストの追跡。 AWS Billing and Cost Management ダッシュボードでこれらのタグをアクティブ化します。AWS はタグを使用してコストを分類し、月単位のコスト配分レポートを提供します。詳細については、「AWS Billing ユーザーガイド」の「コスト配分タグの使用」(Use Cost Allocation Tags) を参照してください。

以下のセクションでは、AWS Elemental MediaStore のタグについてさらに詳しく説明します。

AWS Elemental MediaStore でサポートされるリソース

AWS Elemental MediaStore では、以下のリソースがタグ付けをサポートしています。

- ####

タグの追加と管理の詳細については、「タグの管理」を参照してください。

AWS Elemental MediaStore は、 AWS Identity and Access Management (IAM) のタグベースのアクセスコントロール機能をサポートしていません。

タグの命名規則と使用規則

AWS Elemental MediaStore リソースでのタグの使用には、次の基本的な命名規則と使用規則が適用されます。

- 各リソースには、最大 50 個のタグを設定できます。
- タグキーは、リソースごとにそれぞれ一意である必要があります。また、各タグキーに設定できる値は 1 つのみです。
- タグキーの最大長は UTF-8 で 128 Unicode 文字です。

- タグ値の最大長は UTF-8 で 256 Unicode 文字です。
- 使用できる文字は、UTF-8 対応の文字、数字、スペースと、文字 (. : + = @ _ / -) (ハイフン) です。Amazon EC2 リソースでは、任意の文字を使用できます。
- タグのキーと値では、大文字と小文字が区別されます。ベストプラクティスとして、タグを大文字にするための戦略を決定し、その戦略をすべてのリソースタイプにわたって一貫して実装します。たとえば、`Costcenter`、`costcenter`、`CostCenter` のいずれを使用するかを決定し、すべてのタグに同じ規則を使用します。大文字と小文字の扱いについて、同様のタグに整合性のない規則を使用することは避けてください。
- `aws:` プレフィックスはタグに対して禁止されており、 AWS 用に予約されています。このプレフィックスを持つタグのキーや値を編集または削除することはできません。このプレフィックスの付いたタグは、リソースあたりのタグ数のクォータにカウントされません。

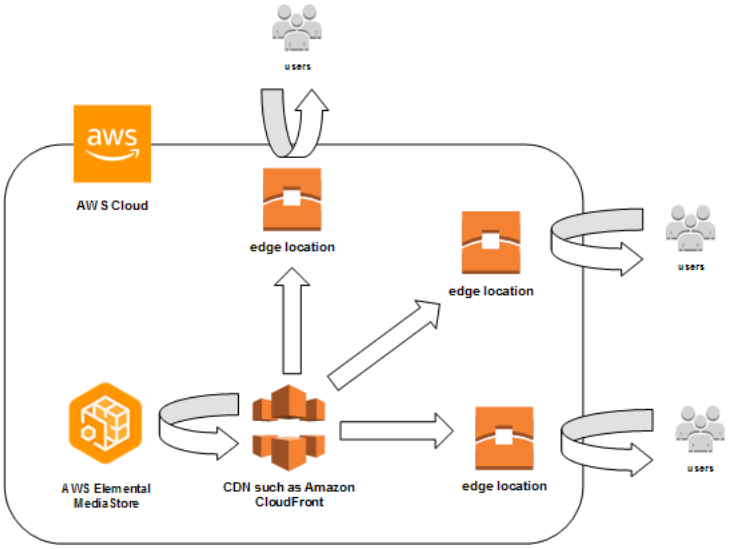
タグの管理

タグは、リソースの `Key` および `Value` プロパティで構成されています。 AWS CLI または MediaStore API を使用して、これらのプロパティの値を追加、編集、または削除できます。タグの使用については、「AWS Elemental MediaStore API リファレンス」の以下のセクションを参照してください。

- CreateContainer
- ListTagsForResource
- リソース
- TagResource
- UntagResource

コンテンツ配信ネットワーク (CDN) の使用

Amazon CloudFront などのコンテンツ配信ネットワーク (CDN) を使用して、AWS Elemental MediaStore に保存したコンテンツを配信することができます。CDN は、グローバルに分散されたサーバーのセットであり、動画などのコンテンツをキャッシュします。ユーザーがコンテンツをリクエストすると、CDN はそのリクエストを最もレイテンシーが低いエッジロケーションにルーティングします。コンテンツがこのエッジロケーションにキャッシュ済みである場合、CDN はコンテンツを直ちに配信します。コンテンツがこのエッジロケーションに現在存在しない場合、CDN は、オリジン (MediaStore コンテナなど) からそのコンテンツを取得してユーザーに配信します。



トピック

- Amazon CloudFront に AWS Elemental MediaStore コンテナへのアクセスを許可する
- AWS Elemental MediaStore による HTTP キャッシュの操作

Amazon CloudFront に AWS Elemental MediaStore コンテナへのアクセスを許可する

Amazon CloudFront を使用して、AWS Elemental MediaStore に保存したコンテンツを配信することができます。これには次のいずれかの方法があります。

- オリジンアクセスコントロール (OAC) の使用 - (推奨) が CloudFront の OAC 機能 AWS リージョン をサポートしている場合は、このオプションを使用します。

- 共有シークレットの使用 - このオプションは、 AWS リージョン が CloudFront の OAC 機能をサポートしていない場合に使用します。

オリジンアクセスコントロール (OAC) の使用

Amazon CloudFront のオリジンアクセスコントロール (OAC) 機能を使用すると、強化されたセキュリティで AWS Elemental MediaStore オリジンを保護できます。MediaStore オリジンの CloudFront のAWS Signature Version 4 (SigV4) を有効にし、CloudFront がリクエストに署名するタイミングおよび署名するかどうかを設定できます。CloudFront の OAC 機能には、コンソール、API、SDK、CLI からアクセスでき、追加使用料金はかかりません。

MediaStore で OAC 機能を使用する方法の詳細については、 Amazon CloudFront 開発者ガイドの「MediaStore オリジンへのアクセスの制限」を参照してください。

共有シークレットの使用

AWS リージョン が Amazon CloudFront の OAC 機能をサポートしていない場合は、CloudFront に読み取りアクセス以上のアクセスを許可するポリシーを AWS Elemental MediaStore コンテナにアタッチできます。

> **Note**
>
> が AWS リージョン サポートしている場合は、OAC 機能を使用することをお勧めします。以下の手順では、MediaStore コンテナへのアクセスを制限するために、共有シークレットを使用して MediaStore と CloudFront を設定する必要があります。セキュリティのベストプラクティスに従い、この手動設定ではシークレットを定期的にローテーションする必要があります。MediaStore オリジンで OAC を使用すると、SigV4 を使用してリクエストに署名し、MediaStore に転送して署名を照合するように CloudFront に指示できます。これにより、シークレットの使用やローテーションの必要がなくなります。これにより、メディアコンテンツが提供される前にリクエストが自動的に検証され、MediaStore と CloudFront を介したメディアコンテンツの配信がより簡単かつ安全になります。

コンテナへのアクセスを CloudFront に許可するには (コンソール)

1. https://console.aws.amazon.com/mediastore/ で MediaStore コンソールを開きます。
2. [Containers (コンテナ)] ページで、コンテナの名前を選択します。

コンテナの詳細ページが表示されます。

3. [Container policy] (コンテナポリシー) セクションで、Amazon CloudFront に読み取りアクセス権限以上を付与するポリシーをアタッチします。

   Example

   次のポリシーの例は HTTPS 経由のパブリック読み取りアクセスのポリシーの例に似ていますが、この例がこれらの要件に一致するのは、このポリシーが HTTPS を介してドメインにリクエストを送信するすべての送信元からの `GetObject` コマンドと `DescribeObject` コマンドを許可するためです。さらに以下のサンプルポリシーでは、リクエストが HTTPS 接続を介して発生し、かつ正しい Referer ヘッダーが含まれている場合にのみ CloudFront が MediaStore オブジェクトにアクセスできるようになるため、ワークフローの安全性が高まります。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CloudFrontRead",
      "Effect": "Allow",
      "Principal": "*",
      "Action": [
        "mediastore:GetObject",
        "mediastore:DescribeObject"
      ],
      "Resource": "arn:aws:mediastore:<region>:<owner acct
 number>:container/<container name>/*",
      "Condition": {
        "StringEquals": {
          "aws:Referer": "<secretValue>"
        },
        "Bool": {
          "aws:SecureTransport": "true"
        }
      }
    }
 ]}
```

4. [Container CORS policy (コンテナの CORS ポリシー)] セクションで、適切なアクセスレベルを許可するポリシーを割り当てます。

> **Note**
>
> CORS ポリシーは、ブラウザベースのプレーヤーにアクセスを許可する場合にのみ必要です。

5. 以下の詳細を書き留めます。

   - コンテナに割り当てられたデータエンドポイント。この情報は [コンテナ] ページの [情報] セクションにあります。CloudFront では、データエンドポイントはオリジンドメイン名と呼ばれます。
   - オブジェクトが保存されているコンテナのフォルダ構造。CloudFront では、これはオリジンパスと呼ばれます。ただし、この設定は省略可能です。オリジンパス詳細については、Amazon CloudFront デベロッパーガイドを参照してください。

6. CloudFront でディストリビューションを作成し、AWS Elemental MediaStore のコンテンツを配信するように設定します。前のステップで収集した情報が必要です。

ポリシーを MediaStore コンテナにアタッチした後、オリジンリクエストに HTTPS 接続のみを使用するように CloudFront を設定し、正しいシークレット値を含むカスタムヘッダーを追加する必要があります。

Referer ヘッダー (コンソール) にシークレット値を指定した HTTPS 接続経由でコンテナにアクセスするように CloudFront を設定するには

1. CloudFront コンソールを で開きます。
2. [オリジン] ページで、MediaStore オリジンを選択します。
3. [編集] を選択します。
4. API プロトコルに [HTTPのみ] を選択します。
5. [カスタムヘッダーの追加] セクションで、[ヘッダーを追加] を選択します。
6. [名前] には [Referer] を選択します。[値] には、*<secretValue>*コンテナポリシーで使用したものと同じ文字列を使用します。
7. [保存] を選択し、変更をデプロイします。

AWS Elemental MediaStore による HTTP キャッシュの操作

AWS Elemental MediaStore は、Amazon CloudFront などのコンテンツ配信ネットワーク (CDN) によって正しく効率的にキャッシュできる方法でオブジェクトを保存します。エンドユーザーまたは CDN が MediaStore からオブジェクトを取得すると、サービスはオブジェクトのキャッシュ動作に影響する HTTP ヘッダーを返します (HTTP 1.1 キャッシュ動作の標準については、RFC2616 セクション 13 を参照してください)。これらのヘッダーは次のとおりです。

- **`ETag`** (カスタマイズ不可) - エンティティタグヘッダーは、MediaStore が送信するレスポンスの一意の識別子です。標準に準拠した CDN およびウェブブラウザは、このタグを、オブジェクトをキャッシュするためのキーとして使用します。MediaStore は、オブジェクトがアップロードされると、ETag を各オブジェクトに自動的に生成します。オブジェクトの詳細を表示して、ETag 値を決定できます。
- **`Last-Modified`** (カスタマイズ不可) — このヘッダーの値は、オブジェクトが変更された日時を示します。MediaStore は、オブジェクトがアップロードされると、この値を自動的に生成します。
- **`Cache-Control`** (カスタマイズ可能) - このヘッダーの値は、変更されたかどうかを CDN が確認するまでオブジェクトをキャッシュする時間を制御します。CLI または API を使用して、MediaStore コンテナにオブジェクトをアップロードするときに、このヘッダーを任意の値に設定できます。有効な値の完全なセットについては、HTTP/1.1 のドキュメントに記載されています。オブジェクトをアップロードするときにこの値を設定しない場合、MediaStore はオブジェクトの取得時にこのヘッダーを返しません。

  Cache-Control ヘッダーの一般的ユースケースは、オブジェクトをキャッシュする期間の指定です。たとえば、エンコーダーによって頻繁に上書きされるビデオマニフェストファイルがあるとします。`max-age` を 10 に設定すると、オブジェクトを 10 秒間キャッシュする必要があることを指定できます。または、上書きされないビデオセグメントが保存されているとします。このオブジェクトの `max-age` を 31536000 に設定して、約 1 年間キャッシュできます。

条件付きリクエスト

MediaStore への条件付きリクエスト

MediaStore は、条件付きリクエスト (RFC7232 に記載されているように `If-Modified-Since` や `If-None-Match` などのリクエストヘッダーを使用) と無条件要求に対して同じように応答します。つまり、MediaStore が有効な `GetObject` リクエストを受信すると、クライアントがすでにオブジェクトを持っている場合でも、サービスは常にオブジェクトを返します。

CDN への条件付きリクエスト

MediaStore に代わってコンテンツを配信する CDN は、RFC7232 セクション 4.1 に記載されているように、`304 Not Modified` を返すことで条件付きリクエストを処理できます。これは、リクエスタが条件付きリクエストに一致するオブジェクトをすでに持っているため、オブジェクトのコンテンツ全体を転送する必要がないことを示します。

CDN (および HTTP/1.1 に準拠したその他のキャッシュ) は、オリジンサーバーによって転送される `ETag` および `Cache-Control` ヘッダーに基づいてこれらの決定を行います。CDN が MediaStore オリジンサーバーに繰り返し取得されたオブジェクトの更新をクエリする頻度を制御するには、MediaStore にアップロードするときにそれらのオブジェクトの `Cache-Control` ヘッダーを設定します。

AWS SDK でのこのサービスの使用

AWS Software Development Kit (SDKsは、多くの一般的なプログラミング言語で使用できます。各 SDK には、デベロッパーが好みの言語でアプリケーションを簡単に構築できるようにする API、コード例、およびドキュメントが提供されています。

SDK ドキュメント	コード例
AWS SDK for C++	AWS SDK for C++ コード例
AWS CLI	AWS CLI コード例
AWS SDK for Go	AWS SDK for Go コード例
AWS SDK for Java	AWS SDK for Java コード例
AWS SDK for JavaScript	AWS SDK for JavaScript コード例
AWS SDK for Kotlin	AWS SDK for Kotlin コード例
AWS SDK for .NET	AWS SDK for .NET コード例
AWS SDK for PHP	AWS SDK for PHP コード例
AWS Tools for PowerShell	Tools for PowerShell のコード例
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) コード例
AWS SDK for Ruby	AWS SDK for Ruby コード例
AWS SDK for Rust	AWS SDK for Rust コード例
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP コード例
AWS SDK for Swift	AWS SDK for Swift コード例

このサービスに固有の例については、「SDK を使用した MediaStore のコード例 AWS SDKs」を参照してください。

可用性の例

必要なものが見つからなかった場合。このページの下側にある [Provide feedback (フィードバックを送信)] リンクから、コードの例をリクエストしてください。

SDK を使用した MediaStore のコード例 AWS SDKs

次のコード例は、 AWS Software Development Kit (SDK) で MediaStore を使用する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

コードの例

- SDK を使用した MediaStore の基本的な例 AWS SDKs
  - SDK を使用した MediaStore のアクション AWS SDKs
    - AWS SDK または CLI CreateContainerで を使用する
    - AWS SDK または CLI DeleteContainerで を使用する
    - AWS SDK DeleteObjectで を使用する
    - AWS SDK または CLI DescribeContainerで を使用する
    - AWS SDK または CLI GetObjectで を使用する
    - AWS SDK または CLI ListContainersで を使用する
    - AWS SDK または CLI PutObjectで を使用する

SDK を使用した MediaStore の基本的な例 AWS SDKs

次のコード例は、 SDKs AWS Elemental MediaStore で AWS の基本を使用する方法を示しています。

例

- SDK を使用した MediaStore のアクション AWS SDKs
  - AWS SDK または CLI CreateContainerで を使用する
  - AWS SDK または CLI DeleteContainerで を使用する
  - AWS SDK DeleteObjectで を使用する

- AWS SDK または CLI DescribeContainerで を使用する
- AWS SDK または CLI GetObjectで を使用する
- AWS SDK または CLI ListContainersで を使用する
- AWS SDK または CLI PutObjectで を使用する

SDK を使用した MediaStore のアクション AWS SDKs

次のコード例は、 AWS SDKs を使用して個々の MediaStore アクションを実行する方法を示しています。それぞれの例には、GitHub へのリンクがあり、そこにはコードの設定と実行に関する説明が記載されています。

以下の例には、最も一般的に使用されるアクションのみ含まれています。詳細な一覧については、「AWS Elemental MediaStore API リファレンス」を参照してください。

例

- AWS SDK または CLI CreateContainerで を使用する
- AWS SDK または CLI DeleteContainerで を使用する
- AWS SDK DeleteObjectで を使用する
- AWS SDK または CLI DescribeContainerで を使用する
- AWS SDK または CLI GetObjectで を使用する
- AWS SDK または CLI ListContainersで を使用する
- AWS SDK または CLI PutObjectで を使用する

AWS SDK または CLI **CreateContainer**で を使用する

次のサンプルコードは、`CreateContainer` を使用する方法を説明しています。

CLI

AWS CLI

コンテナを作成するには

次の `create-container` の例では、新しい空のコンテナを作成します。

```
aws mediastore create-container --container-name ExampleContainer
```

出力:

```
{
    "Container": {
        "AccessLoggingEnabled": false,
        "CreationTime": 1563557265,
        "Name": "ExampleContainer",
        "Status": "CREATING",
        "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleContainer"
    }
}
```

詳細については、「AWS Elemental MediaStore User Guide」の「Creating a Container」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「CreateContainer」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.model.CreateContainerRequest;
import software.amazon.awssdk.services.mediastore.model.CreateContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateContainer {
    public static long sleepTime = 10;

    public static void main(String[] args) {
        final String usage = """

                Usage:    <containerName>

                Where:
                   containerName - The name of the container to create.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        createMediaContainer(mediaStoreClient, containerName);
        mediaStoreClient.close();
    }


    public static void createMediaContainer(MediaStoreClient mediaStoreClient,
 String containerName) {
        try {
            CreateContainerRequest containerRequest =
 CreateContainerRequest.builder()
                    .containerName(containerName)
                    .build();

            CreateContainerResponse containerResponse =
 mediaStoreClient.createContainer(containerRequest);
            String status = containerResponse.container().status().toString();
            while (!status.equalsIgnoreCase("Active")) {
```

```
                status = DescribeContainer.checkContainer(mediaStoreClient,
 containerName);
                System.out.println("Status - " + status);
                Thread.sleep(sleepTime * 1000);
            }

            System.out.println("The container ARN value is " +
 containerResponse.container().arn());
            System.out.println("Finished ");

        } catch (MediaStoreException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「CreateContainer」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DeleteContainer**で を使用する

次のサンプルコードは、DeleteContainer を使用する方法を説明しています。

CLI

AWS CLI

コンテナを削除するには

次の delete-container の例では、指定されたコンテナを削除します。コンテナにオブジェクトが含まれていない場合に限り、コンテナを削除できます。

```
aws mediastore delete-container \
    --container-name=ExampleLiveDemo
```

このコマンドでは何も出力されません。

詳細については、「AWS Elemental MediaStore User Guide」の「Deleting a Container」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「DeleteContainer」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.model.CreateContainerRequest;
import software.amazon.awssdk.services.mediastore.model.CreateContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateContainer {
    public static long sleepTime = 10;

    public static void main(String[] args) {
        final String usage = """

                Usage:    <containerName>

                Where:
                   containerName - The name of the container to create.
                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        createMediaContainer(mediaStoreClient, containerName);
        mediaStoreClient.close();
    }


    public static void createMediaContainer(MediaStoreClient mediaStoreClient,
String containerName) {
        try {
            CreateContainerRequest containerRequest =
CreateContainerRequest.builder()
                    .containerName(containerName)
                    .build();

            CreateContainerResponse containerResponse =
mediaStoreClient.createContainer(containerRequest);
            String status = containerResponse.container().status().toString();
            while (!status.equalsIgnoreCase("Active")) {
                status = DescribeContainer.checkContainer(mediaStoreClient,
containerName);
                System.out.println("Status - " + status);
                Thread.sleep(sleepTime * 1000);
            }

            System.out.println("The container ARN value is " +
containerResponse.container().arn());
            System.out.println("Finished ");

        } catch (MediaStoreException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

```
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「DeleteContainer」を参照してください。https://docs.aws.amazon.com/goto/SdkForJavaV2/mediastore-2017-09-01/DeleteContainer

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK **DeleteObject**で を使用する

次の例は、DeleteObject を使用する方法を説明しています。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import
 software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.services.mediastoredata.model.DeleteObjectRequest;
import
 software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteObject {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = """

                Usage:    <completePath> <containerName>

                Where:
                   completePath - The path (including the container) of the item
 to delete.
                   containerName - The name of the container.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String completePath = args[0];
        String containerName = args[1];
        Region region = Region.US_EAST_1;
        URI uri = new URI(getEndpoint(containerName));

        MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
                .endpointOverride(uri)
                .region(region)
                .build();

        deleteMediaObject(mediaStoreData, completePath);
        mediaStoreData.close();
    }

    public static void deleteMediaObject(MediaStoreDataClient mediaStoreData,
 String completePath) {
        try {
            DeleteObjectRequest deleteObjectRequest =
 DeleteObjectRequest.builder()
                    .path(completePath)
                    .build();
```

```
            mediaStoreData.deleteObject(deleteObjectRequest);

        } catch (MediaStoreDataException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    private static String getEndpoint(String containerName) {
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        DescribeContainerRequest containerRequest =
 DescribeContainerRequest.builder()
                .containerName(containerName)
                .build();

        DescribeContainerResponse response =
 mediaStoreClient.describeContainer(containerRequest);
        mediaStoreClient.close();
        return response.container().endpoint();
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「DeleteObject」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DescribeContainer**で を使用する

次のサンプルコードは、DescribeContainer を使用する方法を説明しています。

CLI

AWS CLI

コンテナの詳細を表示するには

次の `describe-container` の例では、指定されたコンテナの詳細を表示します。

```
aws mediastore describe-container \
    --container-name ExampleContainer
```

出力:

```
{
    "Container": {
        "CreationTime": 1563558086,
        "AccessLoggingEnabled": false,
        "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleContainer",
        "Status": "ACTIVE",
        "Name": "ExampleContainer",
        "Endpoint": "https://aaabbbcccdddee.data.mediastore.us-
west-2.amazonaws.com"
    }
}
```

詳細については、「AWS Elemental MediaStore User Guide」の「Viewing the Details for a Container」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「DescribeContainer」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import
 software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeContainer {

    public static void main(String[] args) {
        final String usage = """

                Usage:    <containerName>

                Where:
                   containerName - The name of the container to describe.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        System.out.println("Status is " + checkContainer(mediaStoreClient,
 containerName));
        mediaStoreClient.close();
    }
```

```
    public static String checkContainer(MediaStoreClient mediaStoreClient, String
containerName) {
        try {
            DescribeContainerRequest describeContainerRequest =
DescribeContainerRequest.builder()
                    .containerName(containerName)
                    .build();

            DescribeContainerResponse containerResponse =
mediaStoreClient.describeContainer(describeContainerRequest);
            System.out.println("The container name is " +
containerResponse.container().name());
            System.out.println("The container ARN is " +
containerResponse.container().arn());
            return containerResponse.container().status().toString();

        } catch (MediaStoreException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「DescribeContainer」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **GetObject**で を使用する

次のサンプルコードは、GetObject を使用する方法を説明しています。

CLI

AWS CLI

オブジェクトをダウンロードするには

次の get-object の例では、指定されたエンドポイントにオブジェクトをダウンロードします。

```
aws mediastore-data get-object \
    --endpoint https://aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com \
    --path=/folder_name/README.md README.md
```

出力:

```
{
    "ContentLength": "2307346",
    "ContentType": "image/jpeg",
    "LastModified": "Fri, 19 Jul 2019 21:32:20 GMT",
    "ETag": "2aa333bbcc8d8d22d777e999c88d4aa9eeeeee4dd89ff7f555555555555da6d3",
    "StatusCode": 200
}
```

オブジェクトの一部をダウンロードするには

次の get-object の例では、指定されたエンドポイントにオブジェクトの一部をダウンロードします。

```
aws mediastore-data get-object \
    --endpoint https://aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com \
    --path /folder_name/README.md \
    --range="bytes=0-100" README2.md
```

出力:

```
{
    "StatusCode": 206,
    "ContentRange": "bytes 0-100/2307346",
    "ContentLength": "101",
    "LastModified": "Fri, 19 Jul 2019 21:32:20 GMT",
    "ContentType": "image/jpeg",
    "ETag": "2aa333bbcc8d8d22d777e999c88d4aa9eeeeee4dd89ff7f555555555555da6d3"
}
```

詳細については、「AWS Elemental MediaStore User Guide」の「Downloading an Object」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「GetObject」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.core.ResponseInputStream;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import
 software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.services.mediastoredata.model.GetObjectRequest;
import software.amazon.awssdk.services.mediastoredata.model.GetObjectResponse;
import
 software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetObject {
```

```
   public static void main(String[] args) throws URISyntaxException {
       final String usage = """

               Usage:    <completePath> <containerName> <savePath>

               Where:
                  completePath - The path of the object in the container (for
example, Videos5/sampleVideo.mp4).
                  containerName - The name of the container.
                  savePath - The path on the local drive where the file is
saved, including the file name (for example, C:/AWS/myvid.mp4).
               """;

       if (args.length != 3) {
           System.out.println(usage);
           System.exit(1);
       }

       String completePath = args[0];
       String containerName = args[1];
       String savePath = args[2];

       Region region = Region.US_EAST_1;
       URI uri = new URI(getEndpoint(containerName));
       MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
               .endpointOverride(uri)
               .region(region)
               .build();

       getMediaObject(mediaStoreData, completePath, savePath);
       mediaStoreData.close();
   }

   public static void getMediaObject(MediaStoreDataClient mediaStoreData, String
completePath, String savePath) {

       try {
           GetObjectRequest objectRequest = GetObjectRequest.builder()
                   .path(completePath)
                   .build();

           // Write out the data to a file.
           ResponseInputStream<GetObjectResponse> data =
mediaStoreData.getObject(objectRequest);
```

```
            byte[] buffer = new byte[data.available()];
            data.read(buffer);

            File targetFile = new File(savePath);
            OutputStream outStream = new FileOutputStream(targetFile);
            outStream.write(buffer);
            System.out.println("The data was written to " + savePath);

        } catch (MediaStoreDataException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    private static String getEndpoint(String containerName) {
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        DescribeContainerRequest containerRequest =
 DescribeContainerRequest.builder()
                .containerName(containerName)
                .build();

        DescribeContainerResponse response =
 mediaStoreClient.describeContainer(containerRequest);
        return response.container().endpoint();
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「GetObject」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **ListContainers**で を使用する

次のサンプルコードは、ListContainers を使用する方法を説明しています。

CLI

AWS CLI

コンテナのリストを表示するには

次の list-containers の例では、アカウントに関連付けられているすべてのコンテナのリストを表示します。

```
aws mediastore list-containers
```

出力:

```
{
    "Containers": [
        {
            "CreationTime": 1505317931,
            "Endpoint": "https://aaabbbcccdddee.data.mediastore.us-
west-2.amazonaws.com",
            "Status": "ACTIVE",
            "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleLiveDemo",
            "AccessLoggingEnabled": false,
            "Name": "ExampleLiveDemo"
        },
        {
            "CreationTime": 1506528818,
            "Endpoint": "https://fffggghhhiiijj.data.mediastore.us-
west-2.amazonaws.com",
            "Status": "ACTIVE",
            "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleContainer",
            "AccessLoggingEnabled": false,
            "Name": "ExampleContainer"
        }
    ]
}
```

詳細については、「AWS Elemental MediaStore User Guide」の「Viewing a List of Containers」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「ListContainers」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.Container;
import software.amazon.awssdk.services.mediastore.model.ListContainersResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListContainers {

    public static void main(String[] args) {

        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        listAllContainers(mediaStoreClient);
        mediaStoreClient.close();
    }

    public static void listAllContainers(MediaStoreClient mediaStoreClient) {
        try {
```

```
            ListContainersResponse containersResponse =
 mediaStoreClient.listContainers();
            List<Container> containers = containersResponse.containers();
            for (Container container : containers) {
                System.out.println("Container name is " + container.name());
            }

        } catch (MediaStoreException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「ListContainers」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **PutObject**で を使用する

次のサンプルコードは、PutObject を使用する方法を説明しています。

CLI

AWS CLI

オブジェクトをアップロードするには

次の put-object の例では、指定されたコンテナにオブジェクトをアップロードします。オブジェクトをコンテナ内に保存するフォルダパスを指定できます。フォルダが既に存在する場合、 AWS Elemental MediaStore はオブジェクトをフォルダに保存します。フォルダが存在しない場合は、フォルダが自動的に作成されて、そのフォルダにオブジェクトが保存されます。

```
aws mediastore-data put-object \
    --endpoint https://aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com \
    --body README.md \
    --path /folder_name/README.md \
    --cache-control "max-age=6, public" \
```

```
    --content-type binary/octet-stream
```

出力:

```
{
    "ContentSHA256":
 "74b5fdb517f423ed750ef214c44adfe2be36e37d861eafe9c842cbe1bf387a9d",
    "StorageClass": "TEMPORAL",
    "ETag": "af3e4731af032167a106015d1f2fe934e68b32ed1aa297a9e325f5c64979277b"
}
```

詳細については、「AWS Elemental MediaStore User Guide」の「Uploading an Object」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「PutObject」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.mediastoredata.model.PutObjectRequest;
import
 software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import software.amazon.awssdk.services.mediastoredata.model.PutObjectResponse;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import
 software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import java.io.File;
import java.net.URI;
import java.net.URISyntaxException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PutObject {
    public static void main(String[] args) throws URISyntaxException {
        final String USAGE = """

                To run this example, supply the name of a container, a file
 location to use, and path in the container\s

                Ex: <containerName> <filePath> <completePath>
                """;

        if (args.length < 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String containerName = args[0];
        String filePath = args[1];
        String completePath = args[2];

        Region region = Region.US_EAST_1;
        URI uri = new URI(getEndpoint(containerName));
        MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
                .endpointOverride(uri)
                .region(region)
                .build();

        putMediaObject(mediaStoreData, filePath, completePath);
        mediaStoreData.close();
    }

    public static void putMediaObject(MediaStoreDataClient mediaStoreData, String
 filePath, String completePath) {
        try {
            File myFile = new File(filePath);
```

```
            RequestBody requestBody = RequestBody.fromFile(myFile);

            PutObjectRequest objectRequest = PutObjectRequest.builder()
                    .path(completePath)
                    .contentType("video/mp4")
                    .build();

            PutObjectResponse response = mediaStoreData.putObject(objectRequest,
 requestBody);
            System.out.println("The saved object is " +
 response.storageClass().toString());

        } catch (MediaStoreDataException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getEndpoint(String containerName) {

        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        DescribeContainerRequest containerRequest =
 DescribeContainerRequest.builder()
                .containerName(containerName)
                .build();

        DescribeContainerResponse response =
 mediaStoreClient.describeContainer(containerRequest);
        return response.container().endpoint();
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「PutObject」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS Elemental MediaStore のクォータ

Service Quotas コンソールには、AWS Elemental MediaStore のクォータに関する情報が表示されます。デフォルトのクォータの表示に加えて、Service Quotas コンソールを使用して、調整可能なクォータの引き上げをリクエストできます。

次の表に、AWS Elemental MediaStore でのクォータ (以前は制限と呼ばれていた) を示します。クォータは、AWS アカウントのサービスリソースまたはオペレーションの最大数です。

> **Note**
>
> アカウント内の個々のコンテナにクォータを割り当てるには、AWS Support またはアカウントマネージャーにお問い合わせください。このオプションは、コンテナ間でアカウントレベルの制限を分割して、1 つのコンテナがクォータ全体を使い果たさないようにするのに役立ちます。

リソースまたはオペレーション	デフォルトのクォータ	コメント
コンテナ	100	このアカウントで作成できるコンテナの最大数。
フォルダレベル	10	コンテナ内に作成できるフォルダレベルの最大数。コンテナ内で 10 レベルを超えて入れ子にしない限り、無制限の数のフォルダを作成できます。
フォルダ	無制限	コンテナ内で 10 レベルを超えて入れ子にしない限り、無制限の数のフォルダを作成できます。
オブジェクトのサイズ	25 MB	1 つのオブジェクトの最大ファイルサイズ。
オブジェクト	無制限	アカウントのフォルダまたはコンテナに必要な数のオブジェクトをアップロードできます。
DeleteObject API リクエストのレート	100	1 秒あたりに可能なオペレーションリクエストの最大数。追加のリクエストは調整されます。

リソースまたはオペレーション	デフォルトのクォータ	コメント
		クォータの引き上げをリクエストできます。
DescribeObject API リクエストのレート	1,000	1 秒あたりに可能なオペレーションリクエストの最大数。追加のリクエストは調整されます。 クォータの引き上げをリクエストできます。
標準のアップロード可用性の GetObject API リクエストのレート	1,000	1 秒あたりに可能なオペレーションリクエストの最大数。追加のリクエストは調整されます。 クォータの引き上げをリクエストできます。
ストリーミングアップロード可用性の GetObject API リクエストのレート	25	1 秒あたりに可能なオペレーションリクエストの最大数。追加のリクエストは調整されます。 クォータの引き上げをリクエストできます。
ListItems API リクエストのレート	5	1 秒あたりに可能なオペレーションリクエストの最大数。追加のリクエストは調整されます。 クォータの引き上げをリクエストできます。
チャンク転送エンコーディング用の PutObject API リクエストのレート (ストリーミングアップロード可用性とも呼ばれます)	10	1 秒あたりに可能なオペレーションリクエストの最大数。追加のリクエストは調整されます。 クォータの引き上げをリクエストできます。リクエストで、リクエストされた TPS と平均オブジェクトサイズを指定します。

リソースまたはオペレーション	デフォルトのクォータ	コメント
標準のアップロード可用性の PutObject API リクエストのレート	100	1 秒あたりに可能なオペレーションリクエストの最大数。追加のリクエストは調整されます。 クォータの引き上げをリクエストできます。リクエストで、リクエストされた TPS と平均オブジェクトサイズを指定します。
メトリクスポリシーのルール	10	メトリクスポリシーに含めることができるルールの最大数。
オブジェクトのライフサイクルポリシーのルール	10	オブジェクトのライフサイクルポリシーに含めることのできるルールの最大数。

AWS Elemental MediaStore 関連情報

以下の表は、AWS Elemental MediaStore を使用する際に役立つ関連リソースのリストです。

- クラスとワークショップ – AWS スキルを磨き、実践的な経験を積むために役立つ、セルフペースラボに加えて、ロールベースおよび専門コースへのリンク。
- AWS デベロッパーセンター – チュートリアル、ダウンロードツール、デ AWS ベロッパーイベントについて説明します。
- AWS デベロッパーツール – AWS アプリケーションを開発および管理するためのデベロッパーツール、SDKs、IDE ツールキット、コマンドラインツールへのリンク。
- 入門リソースセンター – をセットアップし AWS アカウント、 AWS コミュニティに参加して、最初のアプリケーションを起動する方法について説明します。
- ハンズオンチュートリアル - ステップ バイ ステップのチュートリアルに従って、最初のアプリケーションを AWSで起動します。
- AWS ホワイトペーパー – ソリューションアーキテクトや他の技術エキスパートが AWS 作成した、アーキテクチャ、セキュリティ、経済などのトピックを網羅した技術 AWS ホワイトペーパーの包括的なリストへのリンク。
- AWS サポート センター – AWS サポート ケースを作成および管理するためのハブ。フォーラム、技術的なFAQs、サービスのヘルスステータスなど、その他の役立つリソースへのリンクも含まれています AWS Trusted Advisor。
- サポート – クラウドでのアプリケーションの構築と実行に役立つ サポート one-on-one の高速応答サポートチャネルに関する情報のプライマリウェブページ。
- お問い合わせ - AWS の請求、アカウント、イベント、不正使用、その他の問題などに関するお問い合わせの受付窓口です。
- AWS サイト規約 – 当社の著作権と商標、お客様のアカウント、ライセンス、サイトアクセス、およびその他のトピックに関する詳細情報。

ユーザーガイドのドキュメント履歴

次の表は、AWS Elemental MediaStore の今回のリリースの内容をまとめたものです。このドキュメントの更新に関する通知を受け取るには、RSS フィードにサブスクライブできます。

変更	説明	日付
サポート終了通知	サポート終了通知: 2025 年 11 月 13 日、 AWS は AWS Elemental MediaStore のサポートを終了します。2025 年 11 月 13 日以降、MediaStore コンソールまたは MediaStore リソースにアクセスできなくなります。詳細については、こちらのブログ記事をご覧ください。	2024 年 11 月 12 日
オリジンアクセスコントロール (OAC) の改善	AWS Elemental MediaStore を使用したの OAC の使用方法に関する情報を追加しました。	2023 年 4 月 17 日
クォータの更新	`Rules in a Metric Policy` のクォータ値と説明を修正しました。	2022 年 10 月 25 日
ExpiresAt フィールド	アクセスログに、コンテナのライフサイクルポリシー内の一時データのルールに基づいてオブジェクトの有効期限を示す `ExpiresAt` フィールドが含まれるようになりました。	2020 年 7 月 16 日
ライフサイクル移行ルール	オブジェクトライフサイクルポリシーにライフサイクル移行ルールを追加して、オブ	2020 年 4 月 20 日

	ジェクトが一定の期間に達した後に低頻度アクセス (IA) ストレージクラスに移動されるように設定できるようになりました。	
コンテナを空にする	コンテナ内のすべてのオブジェクトを一度に削除できるようになりました。	2020 年 4 月 7 日
Amazon CloudWatch のメトリクスのサポート	メトリクスポリシーを設定して、MediaStore が CloudWatch に送信するメトリクスを指定できます。	2020 年 3 月 30 日
オブジェクトの削除ルールのワイルドカード	オブジェクトのライフサイクルポリシーで、オブジェクトの削除ルールにワイルドカードを使用できるようになりました。これにより、ファイル名や拡張子に基づいて特定の日数後にサービスによって削除されるファイルを指定できます。	2019 年 12 月 20 日
オブジェクトのライフサイクルポリシー	有効期限を秒単位で示すルールをオブジェクトのライフサイクルポリシーに追加できるようになりました。	2019 年 9 月 13 日

AWS CloudFormation のサポート	AWS CloudFormation テンプレートを使用して、コンテナを自動的に作成できるようになりました。 AWS CloudFormation テンプレートは 5 つの API アクションのデータを管理し、コンテナの作成、アクセスのログ記録の設定を追加、デフォルトのコンテナポリシーの更新、Cross-Origin Resource Sharing (CORS) ポリシーの追加、およびライフサイクルポリシーオブジェクトを追加します。	2019 年 5 月 17 日
ストリーミングアップロードの可用性に対するクォータ	ストリーミングアップロードの可用性に従うオブジェクト (チャンク転送されるオブジェクト) の場合、PutObject オペレーションは 10 TPS を、GetObject オペレーションは 25 TPS を超えることはできません。	2019 年 4 月 8 日
オブジェクトのチャンク転送	オブジェクトのチャンク転送のサポートが追加されました。この機能を使用すると、オブジェクトが完全にアップロードされる前にダウンロードできるように設定することができます。	2019 年 4 月 5 日

アクセスのログ記録	AWS Elemental MediaStore では、アクセスのログ記録をサポートするようになりました。これにより、コンテナ内でオブジェクトに対して行われたリクエストの詳細が記録されます。	2019 年 2 月 25 日
オブジェクトのライフサイクルポリシー	現在のコンテナ内のオブジェクトの有効期限を管理する、オブジェクトのライフサイクルポリシーのサポートが追加されました。	2018 年 12 月 12 日
オブジェクトサイズのクォータ (引き上げ後)	オブジェクトサイズのクォータは 25 MB になりました。	2018 年 10 月 10 日
オブジェクトサイズのクォータ (引き上げ後)	オブジェクトサイズのクォータは 20 MB になりました。	2018 年 9 月 6 日
AWS CloudTrail 統合	CloudTrail 統合コンテンツは、CloudTrail サービスの最近の変更に合わせて更新されました。	2018 年 7 月 12 日
CDN コラボレーション	Amazon CloudFront などのコンテンツ配信ネットワーク (CDN) で AWS Elemental MediaStore を使用する方法に関する情報を追加しました。	2018 年 4 月 14 日

CORS 設定	AWS Elemental MediaStore でクロスオリジンリソース共有 (CORS) がサポートされました。CORS では、特定のドメインにロードされたクライアントウェブアプリケーションが別のドメインのリソースと通信できます。	2018 年 2 月 7 日
新しいサービスとガイド	これは、動画の配信およびストレージサービスである AWS Elemental MediaStore と「AWS Elemental MediaStore ユーザーガイド」の最初のリリースです。	2017 年 11 月 27 日

Note

- AWS メディアサービスは、ライフセーフティオペレーション、ナビゲーションまたは通信システム、航空交通コントロール、ライフサポートマシンなど、サービスの可用性、中断、障害が死亡、人身事故、物的損害、環境損害につながる可能性がある、フェイルセーフなパフォーマンスを必要とするアプリケーションや状況での使用を目的として設計または意図されていません。

AWS 用語集

最新の AWS 用語については、「 AWS の用語集 リファレンス」のAWS 「 用語集」を参照してください。