aws

開発者ガイド

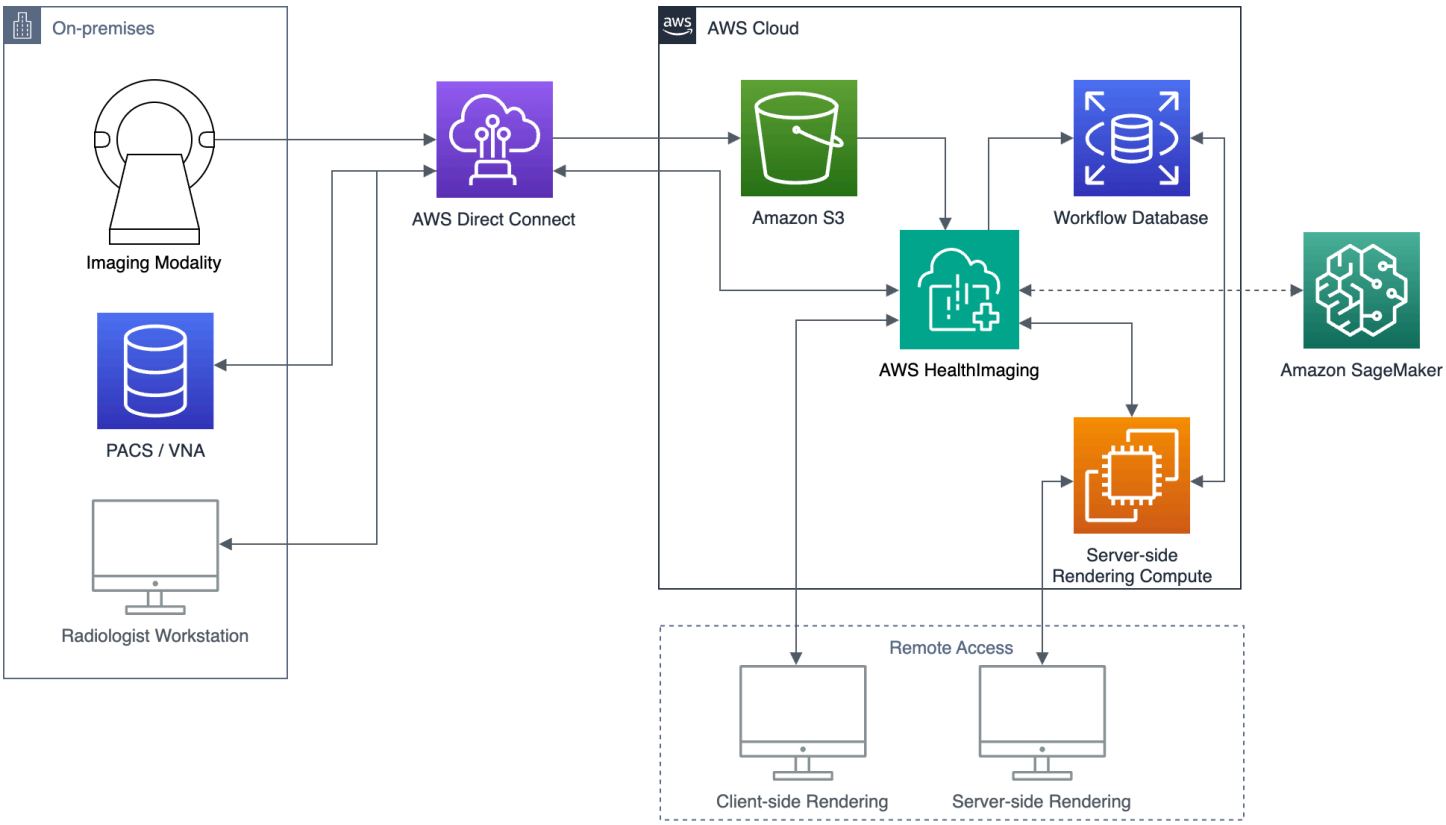
AWS HealthImaging

AWS HealthImaging: 開発者ガイド

Table of Contents

AWS HealthImagingとは

AWS HealthImaging は HIPAA 対応のサービスで、医療医療プロバイダーとその医療画像 ISV パートナーがペタバイト規模で医療画像 (X-Ray、CT、Neval、Ultrasound など) を格納、変換、適用するのに役立ちます。 は、クラウド上に構築された医療画像アプリケーションが以前はオンプレミスでのみパフォーマンスを実現できるように、画像データに 1 秒未満のアクセス HealthImaging を提供します。



トピック

重要な注意点

AWS HealthImaging は専門家による医療の助言、診断、または治療に代わるものではなく、病状や病状の是正、治療、軽減、予防、診断を目的としていません。AWS の使用の一環として人間によるレビューを置き換える責任はお客様にあります。これには HealthImaging、臨床上の意思決定を通知することを目的としたサードパーティー製品との関連付けも含まれます。AWS は、安全な医療判断を適用するトレーニングを受けた医療専門家によるレビュー後に、患者ケアまたは臨床シナリオでのみ使用 HealthImaging してください。

AWS の機能 HealthImaging

AWS HealthImaging には以下の機能があります。

開発者に使いやすい DICOM メタデータ

AWS は、DICOM メタデータをデベロッパーが使いやすい形式で返すことで、アプリケーション開発 HealthImaging を簡素化します。イメージングデータをインポートすると、馴染みのないグループ/要素の 16 進数ではなく、わかりやすいキーワードを使用して個々のメタデータ属性にアクセスできます。患者、治験、シリーズレベルのDICOM要素は標準化されるため、アプリケーション開発者はSOPインスタンス間の不一致に対処する必要がありません。さらに、ネイティブのランタイムタイプではメタデータ属性値に直接アクセスできます。

SIMD アクセラレーションによる画像デコーディング

AWS は、高度な画像圧縮コーデックである高スループット JPEG 2000 (HTJ2K) でエンコードされた画像フレーム (ピクセルデータ) HealthImaging を返します。HTJ2K は、最新のプロセッサ上の単一命令複数データ (SIMD) を活用して、新しいレベルのパフォーマンスを実現します。HTJ2K は JPEG2000 よりも桁違いに高速で、他のすべての DICOM 転送構文よりも少なくとも 2 倍高速です。WASM-SIMD を利用すれば、この極めて高速なウェブビューアーのフットプリントをゼロにすることができます。

ピクセルデータ検証

AWS HealthImaging は、インポート中にすべてのイメージの可逆エンコードとデコード状態をチェックすることで、組み込みのピクセルデータ検証を提供します。詳細については、「ピクセルデータ検証」を参照してください。

業界トップクラスのパフォーマンス

AWS は、効率的なメタデータエンコーディング、ロスレス圧縮、プログレッシブ解像度データアクセスにより、イメージロードパフォーマンスの新しい標準 HealthImaging を設定します。効

率的なメタデータエンコーディングにより、画像閲覧者と AI アルゴリズムは、画像データをロードしなくても DICOM 調査の内容を理解できます。高度な画像圧縮により、画質を損なうことなく画像の読み込みが速くなります。プログレッシブ解像度により、サムネイル、対象領域、低解像度のモバイルデバイスの画像読み込みがさらに速くなります。

スケーラブルな DICOM インポート

AWS HealthImaging インポートは、最新のクラウドネイティブテクノロジーを活用して、複数の DICOM 研究を並行してインポートします。履歴アーカイブは、新しいデータを求める臨床ワークロードに影響を与えることなく、迅速にインポートできます。サポートされている SOP インスタンスと転送構文については、「DICOM サポート」を参照してください。

関連 AWS サービス

AWS HealthImaging は、他の AWS のサービスと緊密に統合されています。以下のサービスに関する知識は、 を最大限に活用するのに役立ちます HealthImaging。

- AWS Identity and Access Management – IAM を使用して、ID と HealthImaging リソースへのアクセスを安全に管理します。
- Amazon Simple Storage Service – DICOM データを にインポートするためのステージング領域として Amazon S3 を使用します HealthImaging。
- Amazon CloudWatch – を使用して CloudWatch 、 HealthImaging リソースの監視とモニタリングを行います。
- AWS CloudTrail – を使用して CloudTrail 、 HealthImaging ユーザーのアクティビティと API の使用状況を追跡します。
- AWS CloudFormation – を使用して AWS CloudFormation 、 でリソースを作成するためのコードとしての Infrastructure as Code (IaC) テンプレートを実装します HealthImaging。
- AWS PrivateLink — データをインターネット PrivateLink に公開することなく、 を使用して HealthImaging と Amazon Virtual Private Cloud 間の接続を確立します。

AWS へのアクセス HealthImaging

AWS Management Console、、 AWS Command Line Interface AWS SDKs HealthImaging を使用して AWS にアクセスできます。このガイドでは、 AWS Management Console および AWS SDKs の AWS CLI および コード例の手順について説明します。

AWS Management Console

AWS Management Console は、 HealthImaging とそれに関連するリソースを管理するためのウェブベースのユーザーインターフェイスを提供します。 AWS アカウントにサインアップしている場合は、HealthImaging コンソールにサインインできます。

AWS Command Line Interface (AWS CLI)

AWS CLI には、さまざまな AWS 製品用のコマンドが用意されており、Windows、Mac、Linux でサポートされています。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。

AWS SDKs

AWS SDKs、ソフトウェアデベロッパー向けのライブラリ、コード例、その他のリソースを提供します。これらのライブラリには、リクエストの暗号化署名、リクエストの再試行、エラーレスポンスの処理などのタスクを自動化する基本的な機能が用意されています。詳細については、「AWSでの構築ツール」を参照してください。

HTTP リクエスト

HTTP リクエストを使用して HealthImaging アクションを呼び出すことができますが、使用するアクションのタイプに応じて 2 つの異なるエンドポイントを指定する必要があります。詳細については、「HTTP リクエストでサポートされている API アクション」を参照してください。

HIPAA の適格性とデータセキュリティ

これは HIPAA 対象サービスです。 AWS、1996 年の米国の医療保険の相互運用性と説明責任に関する法律 (HIPAA)、および AWS のサービスを使用した保護医療情報 (PHI) の処理、保存、転送の詳細については、「HIPAA 概要」を参照してください。

保護医療情報 (PHI) と個人を特定できる情報 (PII) HealthImaging を含む への接続は暗号化する必要があります。デフォルトでは、TLS 経由で HTTPS HealthImaging を使用するすべての接続。 は、暗号化されたお客様のコンテンツ HealthImaging を保存し、AWS 責任共有モデル に従って動作します。

コンプライアンスの詳細については、「AWS HealthImaging のコンプライアンス検証」を参照してください。

料金

HealthImaging は、インテリジェントな階層化による臨床データのライフサイクル管理を自動化するのに役立ちます。詳細については、「ストレージ階層を理解する」を参照してください。

一般的な料金情報については、「AWS 料金表 HealthImaging 」を参照してください。コストを見積もるには、AWS HealthImaging 料金計算ツール を使用します。

AWS HealthImaging の開始方法

AWS HealthImaging の使用を開始するには、AWS アカウントをセットアップし、AWS Identity and Access Management ユーザーを作成します。AWS CLI または AWS SDK を使用するには、それらをインストールして設定する必要があります。

HealthImaging の概念と設定について学習したら、使用開始にあたって役立つ短いチュートリアルが利用できます。

トピック

- AWS HealthImaging の概念
- AWS のセットアップ HealthImaging
- AWS HealthImaging のチュートリアル

AWS HealthImaging の概念

AWS HealthImaging を理解して使用する上で重要な用語と概念を以下に示します。

概念

- データストア
- 画像セット
- メタデータ
- 画像フレーム

データストア

データストアは、単一の場所にある医療画像データのリポジトリです。AWS リージョン1つの AWS アカウント内のデータストアはゼロでも複数でもかまいません。各データストアには独自のAWS KMS暗号化キーがあるため、1 つのデータストア内のデータを他のデータストア内のデータから物理的かつ論理的に分離できます。データストアは IAM ロール、権限、属性ベースのアクセス制御によるアクセス制御をサポートします。

詳細については、データストアの管理 および ストレージ階層を理解する を参照してください。

画像セット

画像セットは、関連する医用画像データを最適化するための抽象的なグループ化メカニズムを定義するAWSの概念です。DICOM P10 イメージングデータを AWS HealthImaging データストアにインポートすると、メタデータと画像フレーム(ピクセルデータ) で構成される画像セットに変換されます。DICOM P10 データをインポートすると、同じ DICOM シリーズの 1 つ以上のサービスオブジェクトペア (SOP) インスタンスの DICOM メタデータと画像フレームを含む画像セットが作成されます。

詳細については、画像データのインポート および 画像セットの理解 を参照してください。

メタデータ

メタデータは、画像セット内にある非ピクセル属性です。DICOM の場合、これには患者の人口統計、処置の詳細その他の取得固有パラメーターが含まれます。AWS HealthImaging は、アプリケーションがすばやくアクセスできるように、画像セットをメタデータと画像フレーム (ピクセルデータ) に分離します。これは、ピクセルデータを必要としない画像ビューア、分析、AI/ML のユースケースに役立ちます。DICOM データは患者、治験、シリーズレベルで正規化されるため、不一致がなくなります。これによりデータの使用が容易になり、安全性が向上し、アクセス性能が向上します。

詳細については、画像セットメタデータの取得 および メタデータの正規化 を参照してください。

画像フレーム

画像フレームは、2D 医療画像を構成する画像セット内にあるピクセルデータです。インポート中、AWS HealthImagingはすべての画像フレームを高スループット JPEG 2000 (HTJ2K) でエンコードします。そのため、画像フレームは表示する前にデコードする必要があります。

詳細については、画像セットのピクセルデータの取得 および HTJ2K デコーディングライブラリ を参照してください。

AWS のセットアップ HealthImaging

AWS を使用する前に、 AWS 環境を設定する必要があります HealthImaging。以下のトピックは、次のセクションにあるチュートリアルの前提条件です。

トピック

- にサインアップする AWS アカウント
- 管理ユーザーの作成

- S3 バケットを作成する
- データストアの作成
- HealthImaging フルアクセス許可を持つ IAM ユーザーを作成する
- インポート用の IAM ロールの作成
- をインストールする AWS CLI （ オプション ）

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、次のステップ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、通話呼び出しを受け取り、電話のキーパッドを使用して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、アカウント内のすべての AWS のサービス とリソースにアクセスできます。セキュリティのベストプラクティスとして、管理ユーザーに管理アクセスを割り当て、ルートユーザーのみを使用してルートユーザーアクセスが必要なタスクを実行します。

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 [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。

管理ユーザーの作成

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管理ユーザーを作成します。

のセキュリティ保護 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「Signing in as the root user」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 「ルートユーザーの仮想 MFA デバイスを有効にする (コンソール)」を参照してください。

管理ユーザーを作成する

- 日常的な管理タスクのためには、 AWS IAM Identity Centerの管理ユーザーに管理アクセスを割り当てます。

  手順については、「AWS IAM Identity Center ユーザーガイド」の「Getting started」を参照してください。

管理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。

S3 バケットを作成する

DICOM P10 データを AWS にインポートするには HealthImaging、2 つの Amazon S3 バケットが必要です。Amazon S3 入力バケットは、インポートする DICOM P10 データを保存し、このバケットから HealthImaging 読み取ります。Amazon S3 出力バケットは、インポートジョブの処理結果を格納し HealthImaging、このバケットに書き込みます。これを視覚的に示した画像データのインポートの図を参照してください。

> **Note**
>
> AWS Identity and Access Management (IAM) ポリシーにより、Amazon S3 バケット名は一意である必要があります。詳細については、「 Amazon Simple Storage Service ユーザーガイド 」の「バケットの名前付け」を参照してください。

このガイドおよび関連するコード例では、次の Amazon S3 入出力バケットを使用します。

- 入力バケット:`arn:aws:s3:::medical-imaging-dicom-input`
- 出力バケット:`arn:aws:s3:::medical-imaging-output`

詳細については、Amazon S3 ユーザーガイド のバケットの作成を参照してください。

データストアの作成

医療画像データをインポートすると、AWS HealthImaging データストアには、画像セットと呼ばれる変換された DICOM P10 ファイルの結果が保持されます。これを視覚的に示した画像データのインポートの図を参照してください。

> **Tip**
>
> `datastoreID` はデータストアを作成すると生成されます。trust relationshipが完了すると、このセクションの後半で説明するインポートで`datastoreID`を使用する必要があります。

データストアを作成するにはデータストアの作成を参照してください。

HealthImaging フルアクセス許可を持つ IAM ユーザーを作成する

> **ベストプラクティス**
>
> インポート、データアクセス、データ管理などのさまざまなニーズに合わせて、個別の IAM ユーザーを作成することをお勧めします。これはAWS Well-Architected フレームワーク の最小特権アクセスの付与と整合します。
> 次のセクションのチュートリアルでは、1 人の IAM ユーザーを使用します。

IAM ユーザーを作成するには

1. 「IAM ユーザーガイド」の AWS 「アカウントに IAM ユーザーを作成する」の手順に従います。わかりやすくするため、ユーザー `ahiadmin` (または同様のもの) などの命名法を検討してください。
2. `AWSHealthImagingFullAccess` 管理ポリシーを IAM ユーザーに適用します。詳細については、「AWS マネージドポリシー: AWSHealthImagingFullAccess」を参照してください。

> **Note**
>
> IAM の権限は絞り込むことができます。詳細については、「AWS HealthImaging の AWS マネージドポリシー」を参照してください。

インポート用の IAM ロールの作成

> **Note**
>
> 次の手順は、DICOM データをインポートするための Amazon S3 バケットへの読み取りおよび書き込みアクセスを許可する AWS Identity and Access Management (IAM) ロールを指します。このロールは次のセクションのチュートリアルで必須ですが、AWS HealthImaging の AWS マネージドポリシーを使ってユーザー、グループ、ロールに IAM アクセス権限を追加することをお勧めします。その方が自分でポリシーを作成するより簡単です。

IAM ロールは、特定の許可があり、 アカウントで作成できる IAM アイデンティティです。インポートジョブを開始するには、`StartDICOMImportJob`アクションを呼び出す IAM ロールを、DICOM P10 データの読み取りとインポートジョブの処理結果の保存に使用する Amazon S3 バケットへのアクセスを許可するユーザーポリシーにアタッチする必要があります。また、AWS がロールを HealthImaging 引き受けることができる信頼関係 (ポリシー) を割り当てる必要があります。

インポート用の IAM ロールの作成

1. IAM コンソールを使用して、`ImportJobDataAccessRole`の名称を持つ IAM ロールを作成します。このロールは、次のセクションのチュートリアルで使用します。詳細については、 IAM ユーザーガイド の IAM ロールの作成を参照してください。

   > **Tip**
   >
   > このガイドでは、インポートジョブの開始のコード例は `ImportJobDataAccessRole` IAM ロールを参考にしています。

2. この IAM ロールに次の IAM アクセス許可ポリシーをアタッチします。このアクセス許可ポリシーは Amazon S3 入出力バケットへのアクセス権を付与します。次の IAM アクセス権限ポリシーをこの IAM ロール`ImportJobDataAccessRole`にアタッチします。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::medical-imaging-dicom-input",
                "arn:aws:s3:::medical-imaging-output"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::medical-imaging-dicom-input/*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::medical-imaging-output/*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

3. `ImportJobDataAccessRole`IAM ロールに以下の信頼関係を追加します。信頼ポリシーでは、データストアの作成セクションの完了時に生成された`datastoreId`が必要です。このトピックの後のチュートリアルでは、1 つの AWS HealthImaging データストアを使用していることを前提としていますが、データストア固有の Amazon S3 バケット、IAM ロール、信頼ポリシーを使用します。

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "medical-imaging.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "aws:SourceAccount": "accountId"
                },
                "ForAllValues:ArnEquals": {
                    "aws:SourceArn": "arn:aws:medical-
imaging:region:accountId:datastore/datastoreId"
                }
            }
        }
    ]
}
```

AWS での IAM ポリシーの作成と使用の詳細については HealthImaging、「」を参照してくださいAWS の Identity and Access Management HealthImaging。

IAM ロールの詳細については、 IAM ユーザーガイド の IAM ロールを参照してください。IAM ポリシーの詳細については、 IAM ユーザーガイド の IAM の許可とポリシーを参照してください。

をインストールする AWS CLI （ オプション ）

AWS Command Line Interfaceを使用している場合は、以下の手順が必要です。 AWS Management Console または AWS SDKsを使用している場合は、次の手順をスキップできます。

を設定するには AWS CLI

1. AWS CLIをダウンロードして設定します。手順については、AWS Command Line Interface ユーザーガイド の次のトピックを参照してください。
   - の最新バージョンをインストールまたは更新する AWS CLI
   - の開始方法 AWS CLI
2. ファイルに AWS CLI config、管理者の名前付きプロファイルを追加します。このプロファイルは、 AWS CLI コマンドを実行するときに使用します。最小特権というセキュリティ原則のもと、実行中のタスクに固有の権限を持つ IAM ロールを別途作成することをお勧めします。名

前付きプロファイルの詳細については、AWS Command Line Interface ユーザーガイド の設定ファイルと認証情報ファイルの設定を参照してください。

```
[default]
aws_access_key_id = default access key ID
aws_secret_access_key = default secret access key
region = region
```

3. 次のhelpコマンドを使用して設定を確認します。

```
aws medical-imaging help
```

が正しく AWS CLI 設定されると、AWS HealthImaging の簡単な説明と使用可能なコマンドのリストが表示されます。

AWS HealthImaging のチュートリアル

目的

このチュートリアルの目的は、DICOM P10 ファイルを AWS HealthImaging データストアにインポートし、メタデータとイメージフレーム (ピクセルデータ) で構成される画像セットに変換することです。DICOM データをインポートしたら、HealthImaging のアクセス設定に基づいて画像セット、メタデータ、およびイメージフレームにアクセスします。

前提条件

このチュートリアルを完了するには、「設定」 に記載されている手順がすべて必要です。

チュートリアルのステップ

1. インポートジョブを開始する
2. インポートジョブのプロパティを取得する
3. 画像セットを検索する
4. 画像セットのプロパティを取得する
5. 画像セットのメタデータを取得する
6. 画像セットのピクセルデータを取得する
7. データストアを削除する

次のステップ

このチュートリアルの次のセクションでは、データストアの管理、イメージデータのインポート、画像セットへのアクセス、画像セットの変更を行う AWS HealthImaging アクションに関する詳細について説明します。これらのセクションではAWS Management Console、AWS CLI、および AWS SDK の手順とコード例を説明します。

AWS によるデータストアの管理 HealthImaging

AWS では HealthImaging、医療画像リソースのデータストアを作成および管理します。以下のトピックでは、 HealthImaging アクションを使用して、、、および AWS SDKs を使用してデータストアを作成 AWS Management Console、説明、一覧表示 AWS CLI、削除する方法について説明します。

> **Important**
>
> 保護対象の医療情報 (PHI)、個人を特定できる情報 (PII)、またはその他の機密情報や秘匿性の高い情報はデータストア名に使用しないでください。データストア名は、プライベートデータや機密データとして使用することを意図していません。

トピック

データストアの作成

`CreateDatastore` アクションを使用して、DICOM P10 ファイルをインポートするための AWS HealthImaging データストアを作成します。以下のタブには、 AWS Management Console および SDK の AWS CLI および コード例の手順が表示されます。 AWS SDKs 詳細については、AWS HealthImaging API リファレンスの`CreateDatastore`「」を参照してください。

データストアを作成するには

のアクセス設定に基づいてタブを選択します HealthImaging。

Console

次の手順では AWS Management Console 、 を使用してデータストアを作成します。

1. HealthImaging コンソールのデータストアの作成ページを開きます。
2. 詳細 の データストア名 に、データストアの名前を入力します。

> **Important**
>
> 保護対象の医療情報 (PHI)、個人を特定できる情報 (PII)、またはその他の機密情報や秘匿性の高い情報はデータストア名に使用しないでください。

3. データ暗号化 で、リソースを暗号化するための AWS KMS キーを選択します。詳細については、「AWS でのデータ保護 HealthImaging」を参照してください。
4. タグ - オプション では、データストアを作成するときに、データストアにタグを追加できます。詳細については、「データストアのタグ作成」を参照してください。
5. データストアの作成 を選択します。

CLI

> **Tip**
>
> データストアを作成するときに、データストアにタグを追加できます。そのためには、`--tags` オプションを含めてください。詳細については、データストアのタグ作成の CLI コードを参照してください。

次のコード例では、 AWS Command Line Interface （ AWS CLI) を使用してデータストアを作成します。

```
aws medical-imaging create-datastore \
  --datastore-name "TestDatastore123" \
  --region us-east-1
```

このアクションは次の出力を返します。

```
{
      "datastoreId": "12345678901234567890123456789012",
      "datastoreStatus": "CREATING"
}
```

JavaScript

> **Tip**
>
> データストアを作成するときに、データストアにタグを追加できます。そのためには、tags オプションを含めてください。詳細については、「」の JavaScript コード例を参照してくださいデータストアのタグ作成。

次のコード例では AWS SDK for JavaScript 、 を使用してデータストアを作成します。

```
import AWS from 'aws-sdk'

const params = {
    region:'us-east-1'
}
const medicalImaging = new AWS.MedicalImaging(params)

const main = async () => {
    try {
        const createDatastoreResponse = await medicalImaging.createDatastore({
            datastoreName: "TestDatastore123"
        }).promise()
        console.log(createDatastoreResponse)
    }
    catch(err) {
        console.log(err)
    }
}

main()
```

このアクションは次の出力を返します。

```
{
  datastoreId: '12345678901234567890123456789012',
  datastoreStatus: 'CREATING'
}
```

Python

> **Tip**
>
> データストアを作成するときに、データストアにタグを追加できます。そのためには、tags オプションを含めてください。詳細については、データストアのタグ作成 の Python コード例を参照してください。

次のコード例では AWS SDK for Python (Boto3) 、 を使用してデータストアを作成します。

```
medical_imaging.create_datastore(datastoreName='TestDatastore123')
```

このアクションは次の出力を返します。

```
{
      'datastoreId': '12345678901234567890123456789012',
      'datastoreStatus': 'CREATING'
}
```

Java

> **Tip**
>
> データストアを作成するときに、データストアにタグを追加できます。そのためには、.tags オプションを含めてください。詳細については、データストアのタグ作成 の Java コード例を参照してください。

次のコード例では AWS SDK for Java 2.x 、 を使用してデータストアを作成します。

```
final CreateDatastoreRequest createDatastoreRequest =
 CreateDatastoreRequest.builder()
                  .datastoreName("TestDatastore123")
                  .build();
final CreateDatastoreResponse createDatastoreResponseData =
 client.createDatastore(createDatastoreRequest);
```

このアクションは次の出力を返します。

```
CreateDatastoreResponse(DatastoreId=12345678901234567890123456789012,
 DatastoreStatus=CREATING)
```

データストアのプロパティの取得

GetDatastore アクションを使用して、AWS HealthImaging データストアのプロパティを取得します。以下のタブには、 AWS Management Console および SDK の AWS CLI および コード例の手順が表示されます。 AWS SDKs 詳細については、AWS HealthImaging API リファレンスのGetDatastore「」を参照してください。

データストアのプロパティを取得するには

のアクセス設定に基づいてタブを選択します HealthImaging。

Console

次の手順では AWS Management Console 、 を使用してデータストアのプロパティを取得します。

1. HealthImaging コンソールのデータストアページを開きます。
2. データストアを選択します。

   データストアの詳細 ページが開きます。詳細 セクションには、すべてのデータストアのプロパティが表示されます。関連する画像セット、インポート、タグを表示するには、該当するタブを選択します。

CLI

次のコード例では、 AWS Command Line Interface （ AWS CLI) を使用してデータストアのプロパティを取得します。

```
aws medical-imaging get-datastore \
  --datastore-id 12345678901234567890123456789012 \
  --region us-east-1
```

このアクションは次の出力を返します。

```
{
```

```
    "datastoreProperties": {
        "datastoreId": "12345678901234567890123456789012",
        "datastoreName": "TestDatastore123",
        "datastoreStatus": "ACTIVE",
        "datastoreArn": "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012",
        "createdAt": "2022-11-15T23:33:09.643000+00:00",
        "updatedAt": "2022-11-15T23:33:09.643000+00:00"
    }
}
```

JavaScript

次のコード例では AWS SDK for JavaScript 、 を使用してデータストアのプロパティを取得します。

```
import AWS from 'aws-sdk'

const params = {
    region:'us-east-1'
}
const medicalImaging = new AWS.MedicalImaging(params)

const main = async () => {
    try {
        const response = await medicalImaging.getDatastore({
            datastoreId: '12345678901234567890123456789012'
        }).promise()
        console.log(response)
    }
    catch(err) {
        console.log(err)
    }
}

main()
```

このアクションは次の出力を返します。

```
{
  datastoreProperties: {
    datastoreId: '12345678901234567890123456789012',
    datastoreName: 'TestDatastore123',
```

```
    datastoreStatus: 'CREATING',
    datastoreArn: 'arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012',
    createdAt: null,
    updatedAt: null
  }
}
```

Python

次のコード例では AWS SDK for Python (Boto3) 、 を使用してデータストアのプロパティを取得します。

```
medical_imaging.get_datastore(datastoreId='12345678901234567890123456789012')
```

このアクションは次の出力を返します。

```
{
    'datastoreProperties': {
        'datastoreId': '12345678901234567890123456789012',
        'datastoreName': 'TestDatastore123',
        'datastoreStatus': 'ACTIVE',
        'datastoreArn': 'arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012',
        'createdAt': datetime.datetime(2022, 11, 15, 14, 38, 39, 412000, tzinfo =
 tzlocal()),
        'updatedAt': datetime.datetime(2022, 11, 15, 14, 38, 39, 412000, tzinfo =
 tzlocal())
    }
}
```

Java

次のコード例では AWS SDK for Java 2.x 、 を使用してデータストアのプロパティを取得します。

```
final GetDatastoreRequest getDatastoreRequest = GetDatastoreRequest.builder()
                .datastoreId("12345678901234567890123456789012")
                .build();
final GetDatastoreResponse getDatastoreResponse =
 client.getDatastore(getDatastoreRequest);
```

このアクションは次の出力を返します。

```
GetDatastoreResponse(DatastoreProperties=DatastoreProperties
 (DatastoreId=12345678901234567890123456789012, DatastoreName=TestDatastore123,
 DatastoreStatus=ACTIVE, DatastoreArn=arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012,
 CreatedAt=2022-11-15T22:38:39.412Z, UpdatedAt=2022-11-15T22:38:39.412Z))
```

データストアの一覧表示

ListDatastores アクションを使用して、AWS で利用可能なデータストアを一覧表示します HealthImaging。以下のタブには、 AWS Management Console および SDK の AWS CLI および コード例の手順が表示されます。 AWS SDKs 詳細については、AWS HealthImaging API リファレンスのListDatastores「」を参照してください。

データストアを一覧表示するには

のアクセス設定に基づいてタブを選択します HealthImaging。

Console

次の手順では、 を使用して使用可能なデータストア AWS Management Console を一覧表示します。

- HealthImaging コンソールのデータストアページを開きます。

  すべてのデータストアは データストア セクションに一覧表示されます。

CLI

次のコード例では、 AWS Command Line Interface （ AWS CLI) を使用して使用可能なデータストアを一覧表示します。

```
aws medical-imaging list-datastores
```

このアクションは次の出力を返します。

```
{
    "datastoreSummaries": [
```

```
        {
            "datastoreId": "12345678901234567890123456789012",
            "datastoreName": "TestDatastore123",
            "datastoreStatus": "ACTIVE",
            "datastoreArn": "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012",
            "createdAt": "2022-11-15T23:33:09.643000+00:00",
            "updatedAt": "2022-11-15T23:33:09.643000+00:00"
        }
    ]
}
```

JavaScript

次のコード例では、 を使用してデータストア AWS SDK for JavaScript を一覧表示します。

```
import AWS from 'aws-sdk'

const params = {
    region:'us-east-1'
}
const medicalImaging = new AWS.MedicalImaging(params)

const main = async () => {
    try {
        const response = await medicalImaging.listDatastores({
        }).promise()
        console.log(response)
    }
    catch(err) {
        console.log(err)
    }
}

main()
```

このアクションは次の出力を返します。

```
{
  datastoreSummaries: [
    {
      datastoreId: '12345678901234567890123456789012',
      datastoreName: 'TestDatastore123',
```

```
      datastoreStatus: 'CREATING',
      datastoreArn: 'arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012',
      createdAt: null,
      updatedAt: null
    }
  ],
  nextToken: null
}
```

Python

次のコード例では、 を使用して使用可能なデータストア AWS SDK for Python (Boto3) を一覧表示します。

```
medical_imaging.list_datastores()
```

このアクションは次の出力を返します。

```
{
    'datastoreSummaries': [{
        'datastoreId': '12345678901234567890123456789012',
        'datastoreName': 'TestDatastore123',
        'datastoreStatus': 'ACTIVE',
        'datastoreArn': 'arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012',
        'createdAt': datetime.datetime(2022, 11, 15, 14, 38, 39, 412000, tzinfo =
 tzlocal()),
        'updatedAt': datetime.datetime(2022, 11, 15, 14, 38, 39, 412000, tzinfo =
 tzlocal())
    }, {
        'datastoreId': '12345678901234567890123456789013',
        'datastoreName': 'TestDatastore456',
        'datastoreStatus': 'DELETED',
        'datastoreArn': 'arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789013',
        'createdAt': datetime.datetime(2022, 11, 15, 18, 25, 48, 996000, tzinfo =
 tzlocal()),
        'updatedAt': datetime.datetime(2022, 11, 15, 18, 27, 47, 964000, tzinfo =
 tzlocal())
    }]
}
```

Java

次のコード例では、 を使用して利用可能なデータストア AWS SDK for Java 2.x を一覧表示します。

```
final ListDatastoresRequest listDatastoresRequest = ListDatastoresRequest.builder()
                .build();
final ListDatastoresResponse  listDatastoresResponse =
 client.listDatastores(listDatastoresRequest);
```

このアクションは次の出力を返します。

```
ListDatastoresResponse(DatastoreSummaries=[
    DatastoreSummary(DatastoreId=12345678901234567890123456789012,
 DatastoreName=TestDatastore123, DatastoreStatus=ACTIVE,
 DatastoreArn=arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012,
 CreatedAt=2022-11-15T22:38:39.412Z, UpdatedAt=2022-11-15T22:38:39.412Z),
    DatastoreSummary(DatastoreId=12345678901234567890123456789013,
 DatastoreName=TestDatastore456, DatastoreStatus=DELETED,
 DatastoreArn=arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789013,
 CreatedAt=2022-11-16T02:25:48.996Z, UpdatedAt=2022-11-16T02:27:47.964Z)
])
```

データストアの削除

DeleteDatastore アクションを使用して、AWS HealthImaging データストア を削除します。以下のタブには、 AWS Management Console および SDK の AWS CLI および コード例の手順が表示されます。 AWS SDKs 詳細については、AWS HealthImaging API リファレンスのDeleteDatastore「」を参照してください。

> **Note**
>
> データストアを削除する前に、まずデータストア内のすべての画像セットを削除する必要があります。詳細については、「画像セットの削除」を参照してください。

データストアを削除する

のアクセス設定に基づいてタブを選択します HealthImaging。

Console

次の手順では AWS Management Console 、 を使用してデータストアを削除します。

1. HealthImaging コンソールのデータストアページを開きます。
2. データストアを選択します。
3. [削除] をクリックします。

   データストアの削除 ページが開きます。
4. データストアの削除を確認するには、テキスト入力フィールドにデータストア名を入力します。
5. [データストアを削除] を選択します。

CLI

次のコード例では、 AWS Command Line Interface （ AWS CLI) を使用してデータストアを削除します。

```
aws medical-imaging delete-datastore \
  --datastore-id "12345678901234567890123456789012" \
  --region us-east-1
```

このアクションは次の出力を返します。

```
{
    "datastoreId": "12345678901234567890123456789012",
    "datastoreStatus": "DELETING"
}
```

JavaScript

次のコード例では AWS SDK for JavaScript 、 を使用してデータストアを削除します。

```
import AWS from 'aws-sdk'

const params = {
```

```
    region:'us-east-1'
}
const medicalImaging = new AWS.MedicalImaging(params)

const main = async () => {
    try {
        const result = await medicalImaging.deleteDatastore({
            datastoreId: '12345678901234567890123456789012',
        }).promise()
        console.log(result)
    }
    catch(err) {
        console.log(err)
    }
}

main()
```

このアクションは次の出力を返します。

```
{
    'datastoreId': '12345678901234567890123456789012',
    'datastoreStatus': 'DELETING'
}
```

Python

次のコード例では AWS SDK for Python (Boto3) 、 を使用してデータストアを削除します。

```
medical_imaging.delete_datastore(datastoreId='12345678901234567890123456789012')
```

このアクションは次の出力を返します。

```
{
    'datastoreId': '12345678901234567890123456789012',
    'datastoreStatus': 'DELETING'
}
```

Java

次のコード例では AWS SDK for Java 2.x 、 を使用してデータストアを削除します。

```
final DeleteDatastoreRequest deleteDatastoreRequest =
 DeleteDatastoreRequest.builder()
                .datastoreId("12345678901234567890123456789012")
                .build();
final DeleteDatastoreResponse deleteDatastoreResponse =
 client.deleteDatastore(deleteDatastoreRequest);
```

このアクションは次の出力を返します。

```
DeleteDatastoreResponse(DatastoreId=12345678901234567890123456789012,
 DatastoreStatus=DELETING)
```

ストレージ階層を理解する

AWS HealthImaging はインテリジェント階層化を使用して臨床ライフサイクルを自動管理します。これにより、新しいまたは使用中のデータと、長期間保存しているデータの両方で、摩擦のない魅力的なパフォーマンスと価格を実現できます。HealthImaging では、以下の階層を使用して 1 か月あたり 1 GB あたり のストレージ料金を請求します。

- 高頻度アクセス階層 — アクセス頻度の高いデータ用の階層。
- アーカイブインスタントアクセス階層 — アーカイブされたデータ用の階層。

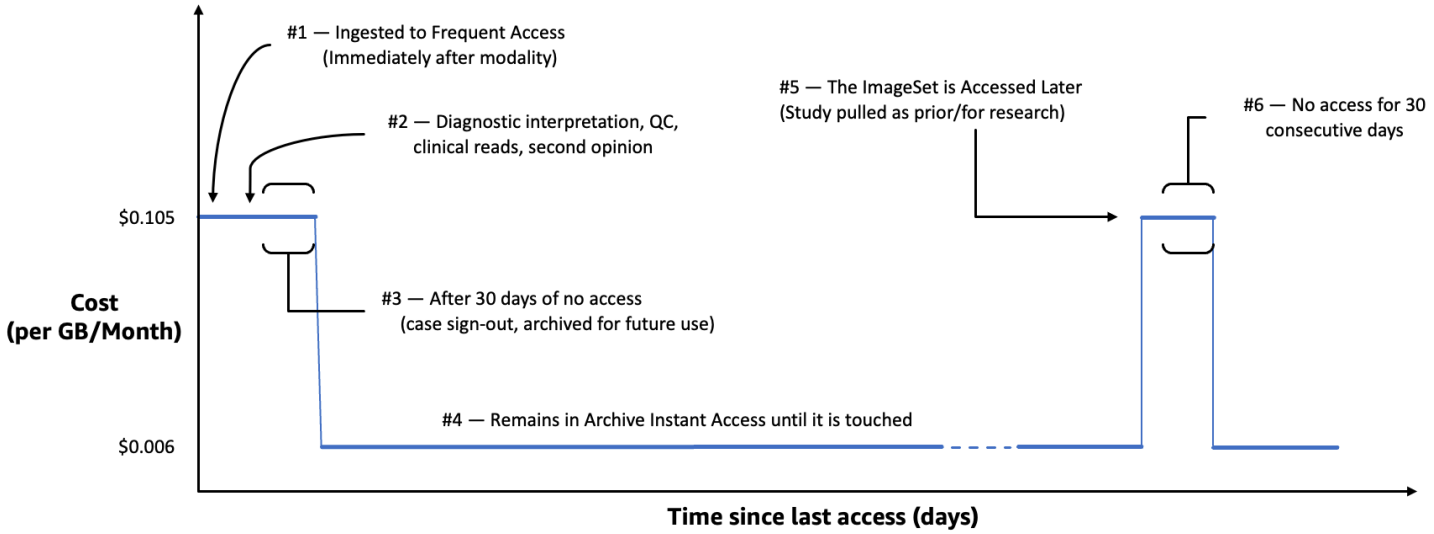
> **Note**
>
> 高頻度アクセス階層とアーカイブインスタントアクセス階層にはパフォーマンス上の違いはありません。インテリジェント階層化は特定の画像セットの API アクションに適用されます。インテリジェント階層化は、データストア、インポート、タグ付けの API アクションを認識しません。階層間の移動は API の使用状況に基づいて自動的に行われます。これについては次のセクションで説明します。

階層の移動の仕組み

- インポート後、画像セットは高頻度アクセス階層で開始されます。
- 30 日間連続して何も操作しないと 、画像セットは自動的にアーカイブインスタントアクセス階層に移動します。

- アーカイブインスタントアクセス階層の画像セットは、操作された後にのみ高頻度アクセス階層に戻ります。

次のグラフは、HealthImaging のインテリジェント階層化プロセスの概要を示しています。



操作と見なされるもの

操作とは、AWS Management Console、AWS CLI、およびAWS SDK を介した特定の API アクセスで、次の場合に発生します。

1. 新しい画像セットが 作成される (`StartDICOMImportJob` または `CopyImageSet`)
2. 画像セットが 更新される (`UpdateImageSetMetadata` または `CopyImageSet`)
3. 画像セットに関連するメタデータまたはイメージフレーム (ピクセルデータ) が読み込まれる (`GetImageSetMetaData` または `GetImageFrame`)

次の HealthImaging API アクションにより、画像セットが操作されアーカイブインスタントアクセス階層から高頻度アクセス階層に移動されます。

- `StartDICOMImportJob`
- `GetImageSetMetadata`
- `GetImageFrame`
- `CopyImageSet`
- `UpdateImageSetMetadata`

> **Note**
>
> イメージフレーム (ピクセルデータ) は UpdateImageSetMetadata アクションでは削除できませんが、請求目的のためカウントされます。

次の HealthImaging API アクションは操作と みなされません 。そのため、画像セットはアーカイブインスタントアクセス階層から高頻度アクセス階層に移動しません。

- CreateDatastore
- GetDatastore
- ListDatastores
- DeleteDatastore
- GetDICOMImportJob
- ListDICOMImportJobs
- SearchImageSets
- GetImageSet
- ListImageSetVersions
- DeleteImageSet
- TagResource
- ListTagsForResource
- UntagResource

AWS HealthImaging による画像データのインポート

インポートとは、医療画像データを Amazon S3 入力バケットから AWS HealthImaging データストアに移動するプロセスです。インポート中、AWS HealthImaging はピクセルデータの検証チェックを実行してから、DICOM P10 ファイルをメタデータとイメージフレーム (ピクセルデータ) で構成される画像セットに変換します。

> **Tip**
>
> HealthImaging に慣れたら、ぜひ AWS HealthImaging のサンプルプロジェクト にアクセスして、インポートと表示のプロジェクトを使用して実装をすぐに始められるようにすることをお勧めします。

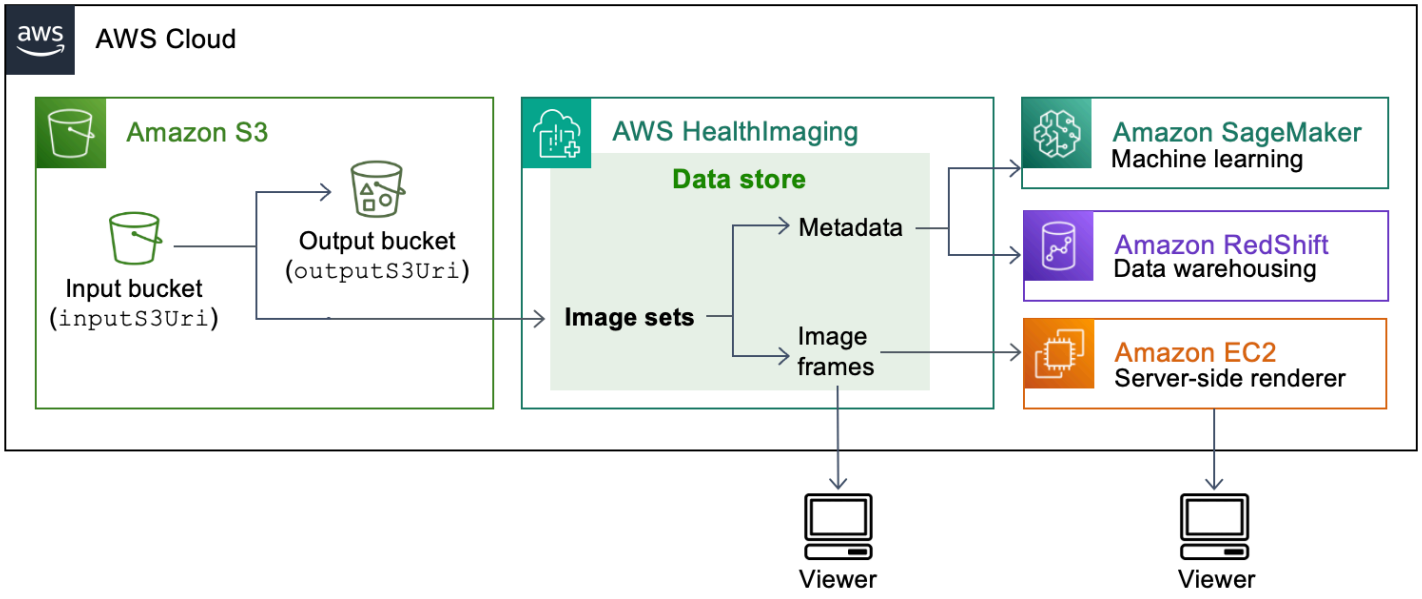
以下のトピックでは、AWS Management Console、AWS CLI、AWS SDK を使用して医療画像データを HealthImaging データストアにインポートする方法について説明します。

トピック

インポートジョブを理解する

AWS でデータストアを作成したら HealthImaging、Amazon S3 入力バケットからデータストアに医療画像データをインポートして、画像セットを作成する必要があります。 AWS Management Console AWS CLI、、および AWS SDKs を使用して、インポートジョブを開始、説明、および一覧表示できます。

次の図は、 が DICOM データをデータストアに HealthImaging インポートし、それを画像セットに変換する方法の概要を示しています。インポートジョブの処理結果は Amazon S3 出力バケット (`outputS3Uri`) に保存され、画像セットは AWS HealthImaging データストアに保存されます。



Amazon S3 から AWS HealthImaging データストアに医療画像ファイルをインポートするときは、次の点に注意してください。

- インポートジョブでは SOP インスタンスと特定の転送構文がサポートされています。詳細については、「DICOM サポート」を参照してください。
- インポート中、特定の DICOM 要素には長さの制限が適用されます。インポートジョブを正常に実行するには、医療画像データが長さの制限を超えていないことを確認してください。詳細については、「DICOM 要素の制約」を参照してください。
- インポートジョブの開始時に、ピクセルデータの検証チェックが実行されます。詳細については、「ピクセルデータ検証」を参照してください。
- HealthImaging インポートアクションに関連するエンドポイント、クォータ、スロットリング制限があります。詳細については、エンドポイントとクォータおよびスロットリングの制限を参照してください。
- インポートジョブごとに、処理結果は `outputS3Uri` の場所に保存されます。処理結果は、`job-output-manifest.json`ファイル、`SUCCESS` および `FAILURE` フォルダで整理されます。

> **Note**
>
> ネストされたフォルダーはインポートジョブではサポートされていません。

- この job-output-manifest.json ファイルには、jobSummary 出力と処理されたデータに関する追加情報が含まれます。次の例は、job-output-manifest.json ファイルからの出力を示しています。

```
{
    "jobSummary": {
        "jobId": "09876543210987654321098765432109",
        "datastoreId": "12345678901234567890123456789012",
        "inputS3Uri": "s3://medical-imaging-dicom-input/dicom_input/",
        "outputS3Uri": "s3://medical-imaging-output/
job_output/12345678901234567890123456789012-
DicomImport-09876543210987654321098765432109/",
        "successOutputS3Uri": "s3://medical-imaging-
output/job_output/12345678901234567890123456789012-
DicomImport-09876543210987654321098765432109/SUCCESS/",
        "failureOutputS3Uri": "s3://medical-imaging-
output/job_output/12345678901234567890123456789012-
DicomImport-09876543210987654321098765432109/FAILURE/",
        "numberOfScannedFiles": 5,
        "numberOfImportedFiles": 3,
        "numberOfFilesWithCustomerError": 2,
        "numberOfFilesWithServerError": 0,
        "numberOfGeneratedImageSets": 2,
        "imageSetsSummary": [{
                "imageSetId": "12345612345612345678907890789012",
                "numberOfMatchedSOPInstances": 2
            },
            {
                "imageSetId": "123456123456123456789017891789012",
                "numberOfMatchedSOPInstances": 1
            }
        ]
    }
}
```

- この SUCCESS フォルダーには、正常にインポートされたすべての画像ファイルの結果を含む success.ndjson ファイルが格納されます。次の例は、success.ndjson ファイルからの出力を示しています。

```
{"inputFile":"dicomInputFolder/1.3.51.5145.5142.20010109.1105620.1.0.1.dcm","importResponse
{"imageSetId":"12345612345612345678907890789012"}}
```

```
{"inputFile":"dicomInputFolder/1.3.51.5145.5142.20010109.1105630.1.0.1.dcm","importResponse
{"imageSetId":"12345612345612345678917891789012"}}
```

- この FAILURE フォルダーには、正常にインポートされなかったすべての画像ファイルの結果を含む failure.ndjson ファイルが格納されます。次の例は、failure.ndjson ファイルからの出力を示しています。

```
{"inputFile":"dicom_input/invalidDicomFile1.dcm","exception":
{"exceptionType":"ValidationException","message":"DICOM attribute TransferSyntaxUID
 does not exist"}}
{"inputFile":"dicom_input/invalidDicomFile2.dcm","exception":
{"exceptionType":"ValidationException","message":"DICOM attributes does not
 exist"}}
```

インポートジョブの開始

StartDICOMImportJob アクションを使用して、ピクセルデータ検証チェックと AWS HealthImaging データストアへの一括データのインポートを開始します。インポートジョブは、inputS3Uri パラメータで指定された Amazon S3 入力バケットにある DICOM P10 ファイルをインポートします。インポートジョブの処理結果は、outputS3Uri パラメータで指定された Amazon S3 出力バケットに保存されます。

> **Note**
>
> インポート中、特定の DICOM 要素には長さの制限が適用されます。詳細については、「DICOM 要素の制約」を参照してください。

以下のタブには、 AWS Management Console および SDK の AWS CLI および コード例の手順が表示されます。 AWS SDKs 詳細については、AWS HealthImaging API リファレンスのStartDICOMImportJob「」を参照してください。

インポートジョブを開始するには

のアクセス設定に基づいてタブを選択します HealthImaging。

Console

次の手順では AWS Management Console 、 を使用してインポートジョブを開始します。

1. HealthImaging コンソールのデータストアページを開きます。
2. データストアを選択します。
3. DICOM データをインポート を選択します。

   DICOM データをインポート ページが開きます。
4. 詳細 セクションで、インポートジョブの名前、S3 のインポート元の場所、暗号化キー (オプション)、および S3 の出力先を入力します。
5. サービスアクセス セクションで 既存のサービスロールを使用する を選択し、サービスロール名 メニューからロールを選択するか、新しいサービスロールを作成して使用する を選択します。
6. Import (インポート) を選択します。

CLI

次のコード例では、 AWS Command Line Interface （ AWS CLI) を使用してインポートジョブを開始します。

```
aws medical-imaging start-dicom-import-job \
  --job-name "test-1" \
  --datastore-id "12345678901234567890123456789012" \
  --input-s3-uri "s3://medical-imaging-dicom-input/dicom_input/" \
  --output-s3-uri "s3://medical-imaging-output/job_output/" \
  --data-access-role-arn "arn:aws:iam::123456789012:role/ImportJobDataAccessRole" \
  --region us-east-1
```

このアクションは次の出力を返します。

```
{
    "datastoreId": "12345678901234567890123456789012",
    "jobId": "09876543210987654321098765432109",
    "jobStatus": "SUBMITTED",
    "submittedAt": "2022-08-12T11:28:11.152000+00:00"
}
```

JavaScript

次のコード例では AWS SDK for JavaScript 、 を使用してインポートジョブを開始します。

```
import AWS from 'aws-sdk'
```

```
const params = {
    region:'us-east-1'
}
const medicalImaging = new AWS.MedicalImaging(params)

const main = async () => {
    try {
        const result = await medicalImaging.startDICOMImportJob({
            jobName: 'test-1',
            datastoreId: '12345678901234567890123456789012',
            dataAccessRoleArn: 'arn:aws:iam::123456789012:role/
ImportJobDataAccessRole',
            inputS3Uri: 's3://medical-imaging-dicom-input/dicom_input/',
            outputS3Uri: 's3://medical-imaging-output/job_output/'
        }).promise()
        console.log(result)
    }
    catch(err) {
        console.log(err)
    }
}

main()
```

このアクションは次の出力を返します。

```
{
  datastoreId: '12345678901234567890123456789012',
  jobId: 'd47974f9e9c8c8d5ef56765ca0ffd7b9',
  jobStatus: 'SUBMITTED',
  submittedAt: 2023-03-17T17:00:31.586Z
}
```

Python

次のコード例では AWS SDK for Python (Boto3) 、 を使用してインポートジョブを開始します。

```
medical_imaging.start_dicom_import_job(
    jobName="test-1",
    datastoreId="12345678901234567890123456789012",
    dataAccessRoleArn="arn:aws:iam::123456789012:role/ImportJobDataAccessRole",
    inputS3Uri="s3://medical-imaging-dicom-input/dicom_input/",
```

```
    outputS3Uri="s3://medical-imaging-output/job_output/",
)
```

このアクションは次の出力を返します。

```
{
    'datastoreId': '12345678901234567890123456789012',
    'jobId': '09876543210987654321098765432109',
    'jobStatus': 'SUBMITTED',
    'submittedAt': datetime.datetime(2022, 8, 10, 4, 19, 51, 676000, tzinfo =
 tzlocal())
}
```

Java

次のコード例では AWS SDK for Java 2.x 、 を使用してインポートジョブを開始します。

```
final StartDicomImportJobRequest startDicomImportJobRequest =
 StartDicomImportJobRequest.builder()
                .jobName("test-1")
                .datastoreId("12345678901234567890123456789012")
                .dataAccessRoleArn("arn:aws:iam::123456789012:role/
ImportJobDataAccessRole")
                .inputS3Uri("s3://medical-imaging-dicom-input/dicom_input/")
                .outputS3Uri("s3://medical-imaging-output/job_output/")
                .build();
final StartDicomImportJobResponse startDicomImportJobResponse =
 client.startDICOMImportJob(startDicomImportJobRequest);
```

このアクションは次の出力を返します。

```
StartDicomImportJobResponse(DatastoreId=12345678901234567890123456789012,
 JobId=09876543210987654321098765432109, JobStatus=SUBMITTED,
 SubmittedAt=2022-08-10T21:44:48.285Z)
```

インポートジョブプロパティの取得

GetDICOMImportJob アクションを使用して、AWS HealthImaging インポートジョブのプロパティの詳細を確認します。たとえば、インポートジョブを開始した後に、GetDICOMImportJob を実行

してジョブのステータスを確認できます。jobStatus が COMPLETED に戻ったら、画像セットにアクセスする準備は完了です。

以下のタブには、 AWS Management Console および SDK の AWS CLI および コード例の手順が表示されます。 AWS SDKs 詳細については、AWS HealthImaging API リファレンスのGetDICOMImportJob「」を参照してください。

インポートジョブのプロパティを取得するには

のアクセス設定に基づいてタブを選択します HealthImaging。

Console

次の手順では AWS Management Console 、 を使用してインポートジョブのプロパティを取得します。

1. HealthImaging コンソールのデータストアページを開きます。
2. データストアを選択します。

   データストアの詳細 ページが開きます。画像セット タブはデフォルトで選択されています。
3. インポート タブを選択します。
4. インポートジョブを選択します。

   インポートジョブの詳細 ページが開き、インポートジョブに関するプロパティが表示されます。

CLI

次のコード例では、 AWS Command Line Interface （ AWS CLI) を使用してインポートジョブのプロパティを取得します。

```
aws medical-imaging get-dicom-import-job \
  --datastore-id "12345678901234567890123456789012" \
  --job-id "09876543210987654321098765432109" \
  --region us-east-1
```

このアクションは次の出力を返します。

```
{
```

```
    "jobProperties": {
        "jobId": "09876543210987654321098765432109",
        "jobName": "test-1",
        "jobStatus": "COMPLETED",
        "datastoreId": "12345678901234567890123456789012",
        "dataAccessRoleArn": "arn:aws:iam::123456789012:role/
ImportJobDataAccessRole",
        "endedAt": "2022-08-12T11:29:42.285000+00:00",
        "submittedAt": "2022-08-12T11:28:11.152000+00:00",
        "inputS3Uri": "s3://medical-imaging-dicom-input/dicom_input/",
        "outputS3Uri": "s3://medical-imaging-output/
job_output/12345678901234567890123456789012-
DicomImport-09876543210987654321098765432109/"
    }
}
```

JavaScript

次のコード例では AWS SDK for JavaScript 、 を使用して HealthImaging インポートジョブのプロパティを取得します。

```
import AWS from 'aws-sdk'

const params = {
    region:'us-east-1'
}
const medicalImaging = new AWS.MedicalImaging(params)

const main = async () => {
    try {
        const result = await medicalImaging.getDICOMImportJob({
            datastoreId: '12345678901234567890123456789012',
            jobId: '09876543210987654321098765432109'
        }).promise()
        console.log(result)
    }
    catch(err) {
        console.log(err)
    }
}

main()
```

このアクションは次の出力を返します。

```
{
  jobProperties: {
    jobId: '09876543210987654321098765432109',
    jobName: 'test-1',
    jobStatus: 'IN_PROGRESS',
    datastoreId: '12345678901234567890123456789012',
    dataAccessRoleArn: 'arn:aws:iam::123456789012:role/ImportJobDataAccessRole',
    endedAt: null,
    submittedAt: 2023-03-17T17:00:31.586Z,
    inputS3Uri: 's3://medical-imaging-dicom-input/dicom_input/',
    outputS3Uri: 's3://medical-imaging-output/
job_output/12345678901234567890123456789012-
DicomImport-09876543210987654321098765432109/',
    message: null
  }
}
```

Python

次のコード例では AWS SDK for Python (Boto3) 、 を使用してインポートジョブのプロパティを取得します。

```
medical_imaging.get_dicom_import_job(
    jobId="09876543210987654321098765432109",
    datastoreId="12345678901234567890123456789012",
)
```

このアクションは次の出力を返します。

```
{
    'jobProperties': {
        'jobId': '09876543210987654321098765432109',
        'jobName': 'test-1',
        'jobStatus': 'SUBMITTED',
        'datastoreId': '12345678901234567890123456789012',
        'dataAccessRoleArn': 'arn:aws:iam::123456789012:role/
ImportJobDataAccessRole',
        'submittedAt': datetime.datetime(2022, 8, 10, 4, 19, 51, 676000, tzinfo =
 tzlocal()),
        'inputS3Uri': 's3://medical-imaging-dicom-input/dicom_input/',
```

```
        'outputS3Uri': 's3://medical-imaging-output/
job_output/123456789012345678901234567890 12-
DicomImport-09876543210987654321098765432109/'
    }
}
```

Java

次のコード例では AWS SDK for Java 2.x 、 を使用してインポートジョブのプロパティを取得します。

```
final GetDicomImportJobRequest getDicomImportJobRequest =
 GetDicomImportJobRequest.builder()
                .datastoreId("1234567890123456789012345678 9012")
                .jobId("09876543210987654321098765432109")
                .build();
final GetDicomImportJobResponse getDicomImportJobResponse =
 client.getDICOMImportJob(getDicomImportJobRequest);
```

このアクションは次の出力を返します。

```
GetDicomImportJobResponse(JobProperties=DICOMImportJobProperties
 (JobId=09876543210987654321098765432109, JobName=Test_Job_Name_1,
 JobStatus=SUBMITTED, DatastoreId=123456789012345678901234567890 12,
 DataAccessRoleArn=arn:aws:iam::123456789012:role/ImportJobDataAccessRole",
 SubmittedAt=2022-08-10T21:44:48.285Z, InputS3Uri=s3://medical-
imaging-dicom-input/dicom_input/, OutputS3Uri=s3://medical-
imaging-output/job_output/123456789012345678901234567890 12-
DicomImport-09876543210987654321098765432109/))
```

インポートジョブの一覧表示

ListDICOMImportJobs アクションを使用して、特定の HealthImaging データストア 用に作成されたインポートジョブを一覧表示します。以下のタブには、 および SDK AWS Management Console の AWS CLI および コード例の手順が表示されます。 AWS SDKs 詳細については、AWS HealthImaging API リファレンスのListDICOMImportJobs「」を参照してください。

インポートジョブを一覧表示するには

のアクセス設定に基づいてタブを選択します HealthImaging。

Console

次の手順では、 を使用してインポートジョブ AWS Management Console を一覧表示します。

1. HealthImaging コンソールのデータストアページを開きます。
2. データストアを選択します。

   データストアの詳細 ページが開きます。画像セット タブはデフォルトで選択されています。
3. インポート タブを選択すると、関連するすべてのインポートジョブが表示されます。

CLI

次のコード例では、 AWS Command Line Interface （ AWS CLI) を使用してインポートジョブを一覧表示します。

```
aws medical-imaging list-dicom-import-jobs \
  --datastore-id "12345678901234567890123456789012" \
  --region us-east-1
```

このアクションは次の出力を返します。

```
{
    "jobSummaries": [
        {
            "jobId": "09876543210987654321098765432109",
            "jobName": "test-1",
            "jobStatus": "COMPLETED",
            "datastoreId": "12345678901234567890123456789012",
            "dataAccessRoleArn": "arn:aws:iam::123456789012:role/
ImportJobDataAccessRole",
            "endedAt": "2022-08-12T11:21:56.504000+00:00",
            "submittedAt": "2022-08-12T11:20:21.734000+00:00"
        }
    ]
}
```

JavaScript

次のコード例では、 を使用してインポートジョブ AWS SDK for JavaScript を一覧表示します。

```
import AWS from 'aws-sdk'

const params = {
    region:'us-east-1'
}
const medicalImaging = new AWS.MedicalImaging(params)

const main = async () => {
    try {
        const result = await medicalImaging.listDICOMImportJobs({
            datastoreId: '12345678901234567890123456789012'
        }).promise()
        console.log(result)
    }
    catch(err) {
        console.log(err)
    }
}

main()
```

このアクションは次の出力を返します。

```
{
  jobSummaries: [
    {
      jobId: '9bd538b59232116b611972d62f9bedb0',
      jobName: 'test',
      jobStatus: 'COMPLETED',
      datastoreId: '12345678901234567890123456789012',
      dataAccessRoleArn: 'arn:aws:iam::123456789012:role/service-role/
MedicalImagingDatastores-20230308T175204',
      endedAt: 2023-03-08T23:54:38.399Z,
      submittedAt: 2023-03-08T23:53:36.807Z,
      message: null
    }
  ],
  nextToken: null
}
```

Python

次のコード例では、 を使用してインポートジョブ AWS SDK for Python (Boto3) を一覧表示します。

```
medical_imaging.list_dicom_import_jobs(datastoreId='12345678901234567890123456789012')
```

このアクションは次の出力を返します。

```
{
    'jobSummaries': [{
                'jobId': '09876543210987654321098765432109',
                'jobName': 'test-1',
                'jobStatus': 'SUBMITTED',
                'datastoreId': '12345678901234567890123456789012',
                'dataAccessRoleArn': 'arn:aws:iam::123456789012:role/
ImportJobDataAccessRole',
                'submittedAt ': datetime.datetime(2022, 8, 10, 4, 19, 51, 676000,
 tzinfo=tzlocal())
    }]
}
```

Java

次のコード例では、 を使用してインポートジョブ AWS SDK for Java 2.x を一覧表示します。

```
final ListDicomImportJobsRequest listDicomImportJobsRequest =
 ListDicomImportJobsRequest.builder()
                .datastoreId("12345678901234567890123456789012")
                .build();
final ListDicomImportJobsResponse listDicomImportJobsResponse =
 client.listDICOMImportJobs(listDicomImportJobsRequest);
```

このアクションは次の出力を返します。

```
ListDicomImportJobsResponse(JobSummaries=[
    DICOMImportJobSummary(JobId=09876543210987654321098765432109,
 JobName=Test_Job_Name_1, JobStatus=SUBMITTED,
 DatastoreId=12345678901234567890123456789012,
 DataAccessRoleArn=arn:aws:iam::123456789012:role/ImportJobDataAccessRole,
 SubmittedAt=2022-08-10T21:44:48.285Z)
    ]
```

```
)
```

AWS HealthImaging による画像セットへのアクセス

AWS HealthImaging で医療画像データにアクセスするには、通常、一意のキーを持つ画像セットを検索し、関連するメタデータとイメージフレーム (ピクセルデータ) を取得する必要があります。

> **Tip**
>
> AWS HealthImaging に慣れてきたら、ぜひ AWS HealthImaging のサンプルプロジェクト アクセスして、表示プロジェクトを使用して実装をすぐに始められるようにすることをお勧めします。

以下のトピックでは、画像セットとは何か、AWS Management Console、AWS CLI、および AWS SDK を使用して画像セットを検索し、関連するプロパティ、メタデータ、イメージフレームを取得する方法について説明します。

トピック

- 画像セットの理解
- 画像セットの検索
- 画像セットのプロパティの取得
- 画像セットメタデータの取得
- 画像セットのピクセルデータの取得

画像セットの理解

画像セットは、AWS の基盤となる AWS 概念です HealthImaging。画像セットは DICOM データをにインポートするときに作成されるため HealthImaging、サービスを使用する場合は画像セットを十分に理解する必要があります。

画像セットが導入された理由は次のとおりです。

- 関連データのみをグループ化することで、患者の安全性を最大限に高めることができます。
- 不一致の可視性を高めるために、データのクリーニングを促します。詳細については、「画像セットの変更」を参照してください。

> **Note**
>
> クリーニング前に DICOM データを臨床使用すると、患者に害を及ぼす可能性があります。

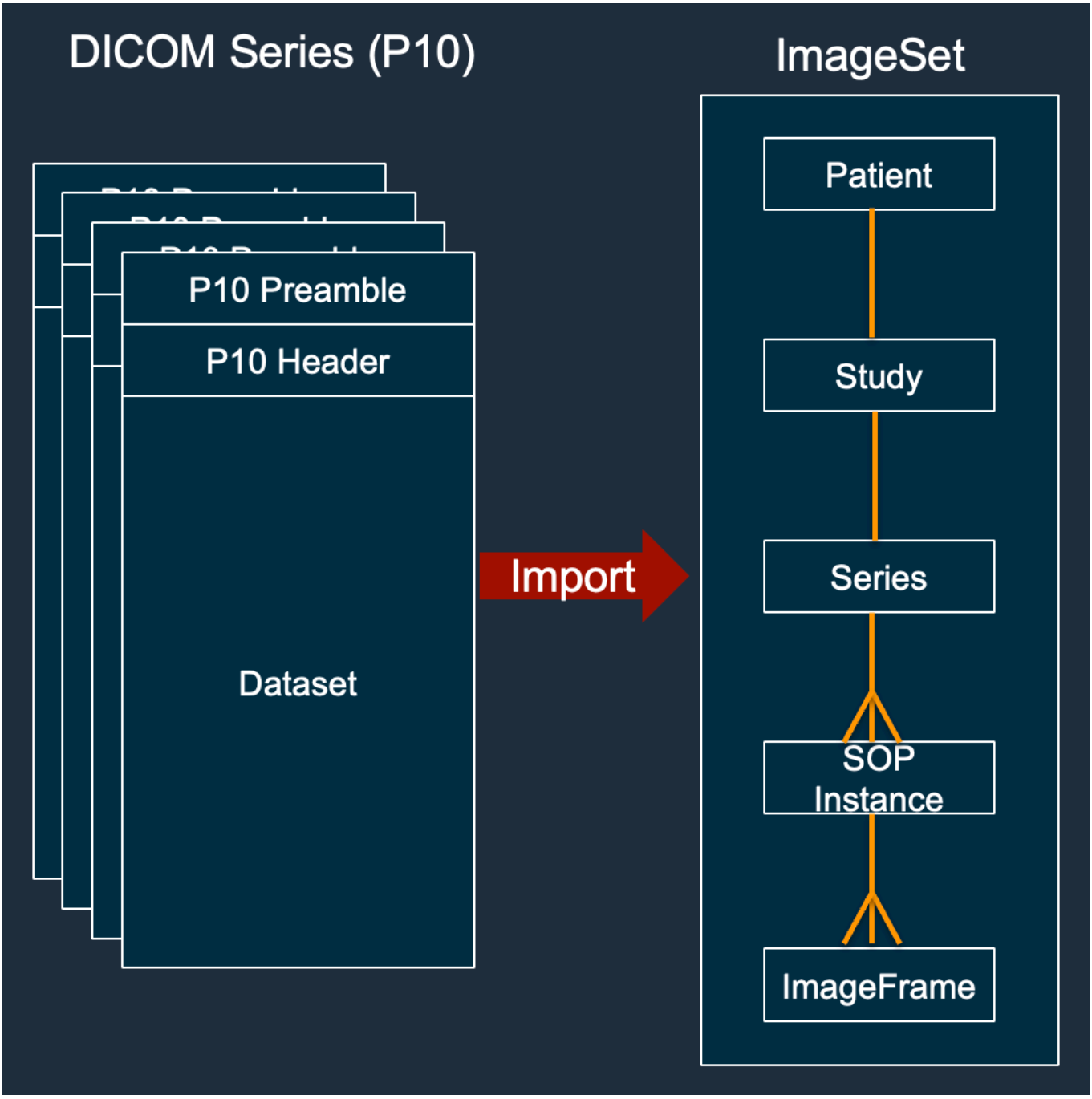
- 柔軟な API により、さまざまな医療画像ワークフロー (臨床および非臨床) をサポートします。

次のリストでは、画像セットへのアクセスについて詳しく説明し、 でその機能と目的を理解するのに役立つ例と図を示します HealthImaging。

画像セットとは

画像セットは関連する画像データをグループ化したものです。DICOM では、これは不一致のない、DICOM シリーズ内の1つ以上の SOP インスタンスです。画像セットは正規化された属性で構成され、これは、患者、治験、シリーズレベルの DICOM レジストリ要素に対応する属性と値の共通セットです。さらに、画像セットはバージョン管理されているため、すべての変更は保存され、以前のバージョンにもアクセスできます。画像セットは AWS リソースであるため、Amazon リソースネーム (ARNsが割り当てられます。

> **Note**
>
> 理想的には、DICOM シリーズが単一の画像セットにマッピングされ、患者のレベル、患者のレベル、シリーズレベルで一貫した共通の属性を持ちます。



Note

DICOM インポートジョブ:

- 常に新しい画像セットを作成し、既存の画像セットは更新しないでください。
- 同じ SOP インスタンスをインポートするたびに追加のストレージが使用されるため、SOP インスタンスのストレージを重複排除しないでください。

- PatientName 不一致などの正規化されたメタデータ属性のバリアントがある場合など、単一の DICOM に対して複数の画像セットを作成する場合があります。

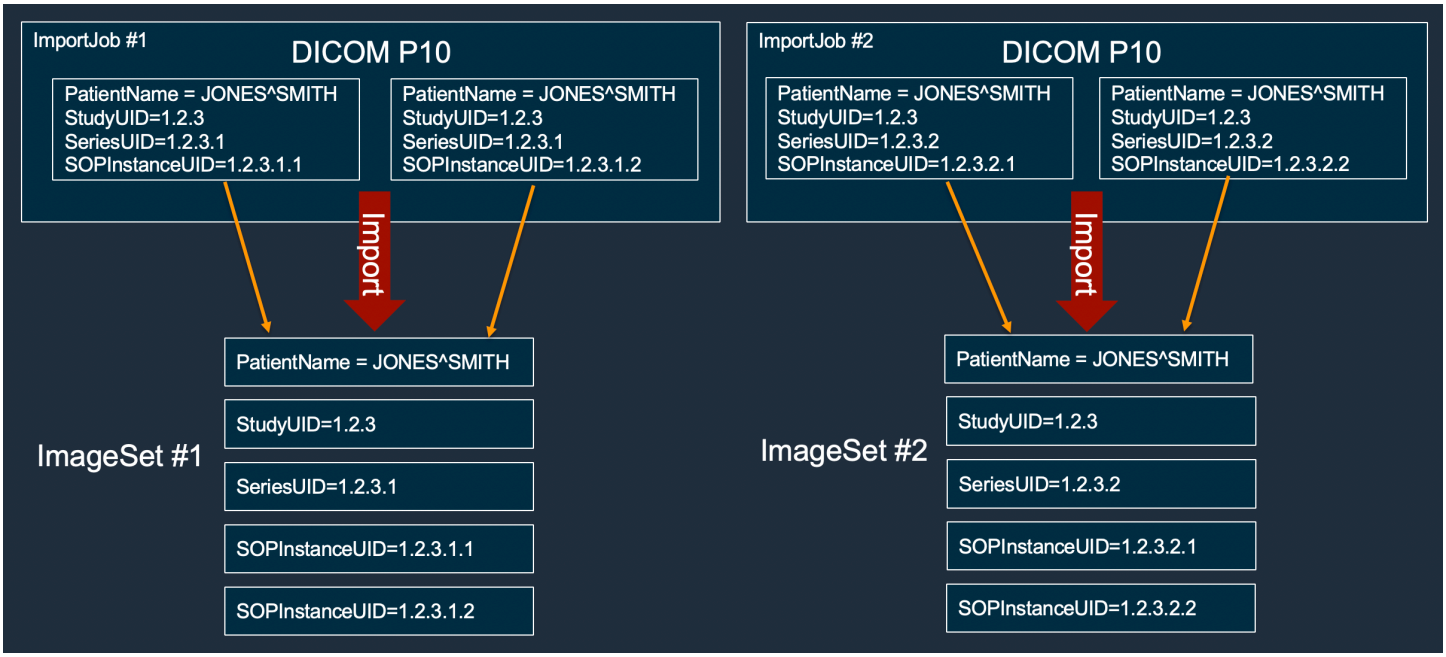
画像セットはどのように表示されるか

画像セットは、一意の Amazon AWS リソースネーム (ARN) を持つリソースです。キーと値のペアを 50 個までタグ付けでき、IAM を通じてロールベースのアクセス制御 (RBAC) と属性ベースのアクセス制御 (ABAC) が可能です。

以下の画像は、画像セットのコードとその他の機能の一覧です。

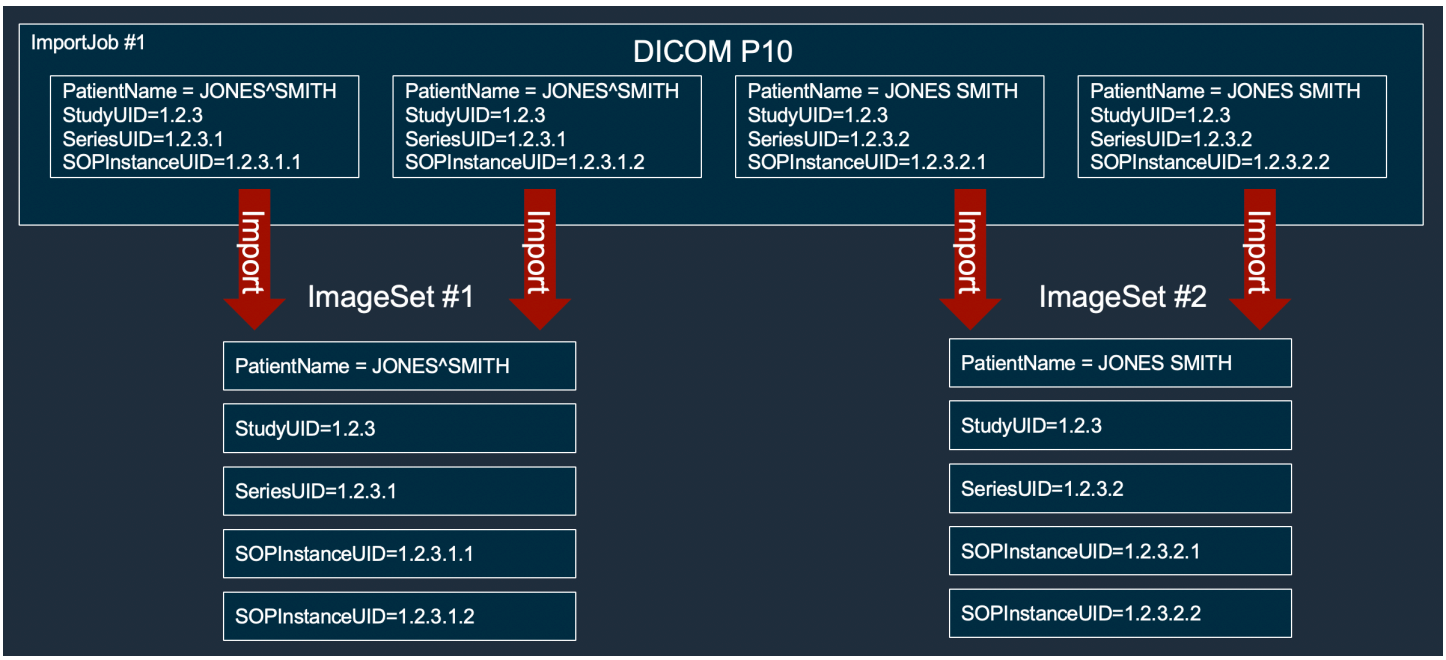
画像セットの作成例: 複数のインポートジョブ

次の例は、複数のインポートジョブが常に新しいイメージセットを作成し、既存のイメージセットに絶対に追加しないことを示しています。



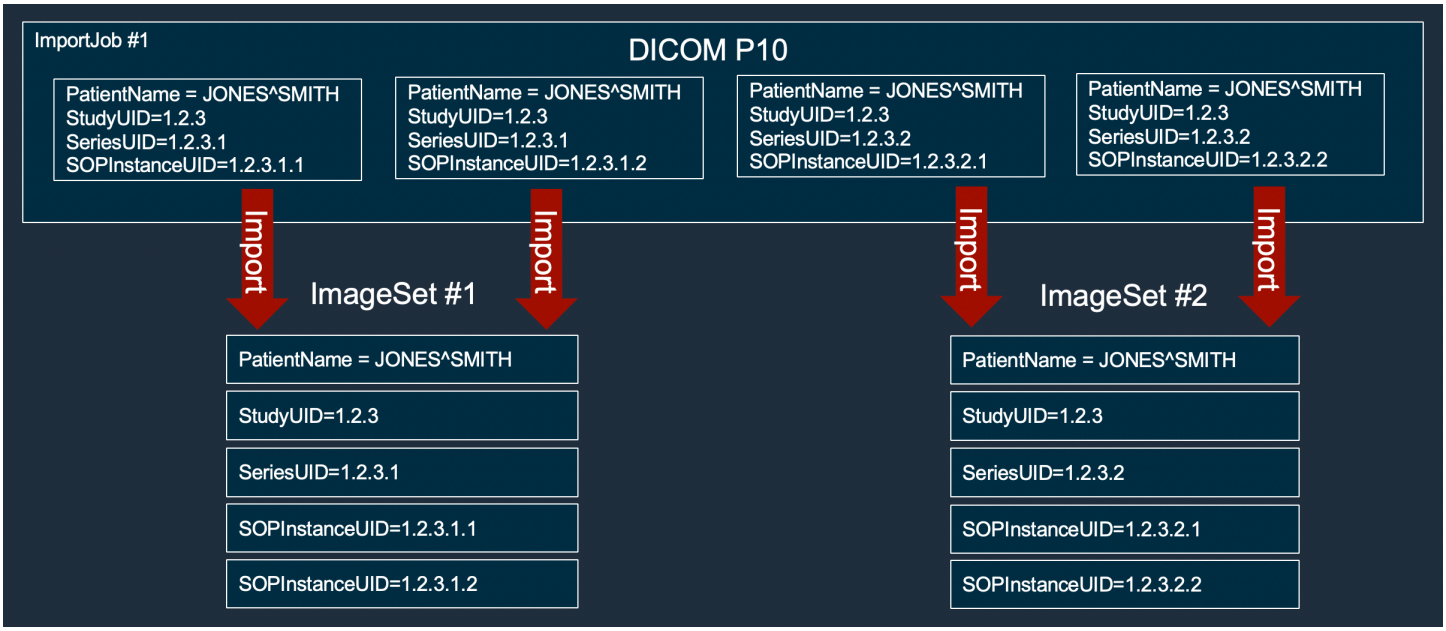
画像セットの作成例: 2 つのバリアントを含む 1 つのインポートジョブ

次の例は、インスタンス 1 と 2 の患者名がインスタンス 3 と 4 と異なるために、1 つのインポートジョブで 2 つの画像セットが作成されることを示しています。



画像セットの作成例: 最適化を含む単一のインポートジョブ

以下の例は、患者名が一致していても 1 つのインポートジョブで 2 つの画像セットを作成して、スループットを向上させる様子を示しています。



画像セットの検索

`SearchImageSets` アクションを使用して、 `ACTIVE` HealthImaging データストア内のすべての画像セットに対して検索クエリを実行します。以下のタブには、 AWS Management Console および SDK の AWS CLI および コード例の手順が表示されます。 AWS SDKs 詳細については、AWS HealthImaging API リファレンスの`SearchImageSets`「」を参照してください。

> **Note**
>
> `SearchImageSets` は 1 つの検索クエリパラメータを受け入れ、条件に一致するすべての画像セットについて、ページ分割されたレスポンスを返します。範囲クエリはすべて `(lowerBound, upperBound)` として入力する必要があります。
> デフォルトでは、`SearchImageSets` は `updatedAt` フィールドを使用して最新のものから古いものへと降順でソートします。
> 顧客所有の AWS KMS キーを使用してデータストアを作成した場合は、画像セットを操作する前に AWS KMS キーポリシーを更新する必要があります。詳細については、「カスタマーマネージドキーの作成」を参照してください。

画像セットを検索するには

の検索設定に基づいてタブを選択します HealthImaging。

Console

次の手順では AWS Management Console 、 を使用して画像セットを検索します。

コンソールのユースケース:**EQUAL** 演算子

以下の検索条件では、EQUAL 演算子を使用し、特定の値に基づいて画像セットを検索します。

1. HealthImaging コンソールのデータストアページを開きます。
2. データストアを選択します。

   データストアの詳細ページが開き、デフォルトで [画像セット] タブが選択されます。
3. プロパティフィルターメニューを選択し、`Created =` を選択します。
4. メニューからデータストアの作成日を選択し、検索条件に基づいてフィルタリングします。

CLI

次のコード例では、 AWS Command Line Interface （ AWS CLI) を使用して、複数のユースケースシナリオに基づいて画像セットを検索します。

CLI ユースケース #1: **EQUAL** 演算子

以下の検索条件では、EQUAL 演算子を使用し、特定の値に基づいて画像セットを検索します。

```
aws medical-imaging search-image-sets \
  --datastore-id 12345678901234567890123456789012 \
  --search-criteria '{
"filters": [{
    "values": [{"DICOMPatientId" : "SUBJECT08701"}],
    "operator": "EQUAL"
    }]
}'
```

このアクションは次の出力を返します。

```
{
    "imageSetsMetadataSummaries": [{
```

```
            "imageSetId": "09876543210987654321098765432109",
            "createdAt": "2022-12-06T21:40:59.429000+00:00",
            "version": 1,
            "DICOMTags": {
                "DICOMStudyId": "2011201407",
                "DICOMStudyDate": "19991122",
                "DICOMPatientSex": "F",
                "DICOMStudyInstanceUID":
 "1.2.840.99999999.84710745.943275268089",
                "DICOMPatientBirthDate": "19201120",
                "DICOMStudyDescription": "UNKNOWN",
                "DICOMPatientId": "SUBJECT08701",
                "DICOMPatientName": "Melissa844 Huel628",
                "DICOMNumberOfStudyRelatedInstances": 1,
                "DICOMStudyTime": "140728",
                "DICOMNumberOfStudyRelatedSeries": 1
            },
            "updatedAt": "2022-12-06T21:40:59.429000+00:00"
        }
    ]
}
```

CLI ユースケース #2: **DICOMStudyDate** と **DICOMStudyTime** を使用する**BETWEEN** 演算子

次の検索条件では、1990 年 1 月 1 日 (午前 0 時) から 2023 年 1 月 1 日 (午前 0 時) の間に生成された DICOM スタディを含む画像セットを検索します。

> **Note**
>
> DICOMStudyTime は省略可能です。入力されていない場合は、フィルターで指定された日付けの時間値は午前 0 時 (1 日の始まり) になります。

```
aws medical-imaging search-image-sets \
  --datastore-id 12345678901234567890123456789012 \
  --search-criteria '{
    "filters": [{
    "values": [{"DICOMStudyDateAndTime": {"DICOMStudyDate": "19900101",
 "DICOMStudyTime":"000000"}}, {"DICOMStudyDateAndTime": {"DICOMStudyDate":
 "20230101", "DICOMStudyTime":"000000"}}],
    "operator": "BETWEEN"
  }]
```

```
}'
```

このアクションは次の出力を返します。

```
{
    "imageSetsMetadataSummaries": [{
            "imageSetId": "09876543210987654321098765432109",
            "createdAt": "2022-12-06T21:40:59.429000+00:00",
            "version": 1,
            "DICOMTags": {
                "DICOMStudyId": "2011201407",
                "DICOMStudyDate": "19991122",
                "DICOMPatientSex": "F",
                "DICOMStudyInstanceUID":
 "1.2.840.99999999.84710745.943275268089",
                "DICOMPatientBirthDate": "19201120",
                "DICOMStudyDescription": "UNKNOWN",
                "DICOMPatientId": "SUBJECT08701",
                "DICOMPatientName": "Melissa844 Huel628",
                "DICOMNumberOfStudyRelatedInstances": 1,
                "DICOMStudyTime": "140728",
                "DICOMNumberOfStudyRelatedSeries": 1
            },
            "updatedAt": "2022-12-06T21:40:59.429000+00:00"
        }
    ]
}
```

CLI ユースケース #3: **`createdAt`** を使用した **`BETWEEN`** 演算子 (タイムスタディは以前に保持)

次の検索条件では、UTC タイムゾーンの時間範囲 HealthImaging 間で DICOM 値が保持されている画像セットを検索します。

> **Note**
>
> サンプルの形式 (`"1985-04-12T23:20:50.52Z"`) で `createdAt` を入力します。

```
aws medical-imaging search-image-sets \
  --datastore-id 12345678901234567890123456789012 \
  --search-criteria '{
```

```
    "filters": [{
    "values": [{"createdAt": "1985-04-12T23:20:50.52Z"}}, {"createdAt":
 "2022-04-12T23:20:50.52Z"}}],
    "operator": "BETWEEN"
  }]
}'
```

このアクションは次の出力を返します。

```
{
    "imageSetsMetadataSummaries": [{
            "imageSetId": "09876543210987654321098765432109",
            "createdAt": "2022-12-06T21:40:59.429000+00:00",
            "version": 1,
            "DICOMTags": {
                "DICOMStudyId": "2011201407",
                "DICOMStudyDate": "19991122",
                "DICOMPatientSex": "F",
                "DICOMStudyInstanceUID":
 "1.2.840.99999999.84710745.943275268089",
                "DICOMPatientBirthDate": "19201120",
                "DICOMStudyDescription": "UNKNOWN",
                "DICOMPatientId": "SUBJECT08701",
                "DICOMPatientName": "Melissa844 Huel628",
                "DICOMNumberOfStudyRelatedInstances": 1,
                "DICOMStudyTime": "140728",
                "DICOMNumberOfStudyRelatedSeries": 1
            },
            "lastUpdatedAt": "2022-12-06T21:40:59.429000+00:00"
        }
    ]
}
```

JavaScript

次のコード例では、 を使用して AWS SDK for JavaScript 、複数のユースケースシナリオに基づいて画像セットを検索します。

JavaScript ユースケース #1: **EQUAL**オペレーター

以下の検索条件では、EQUAL 演算子を使用し、特定の値に基づいて画像セットを検索します。

```
import AWS from 'aws-sdk'
import util from 'util'

const params = {
    region:'us-east-1'
}
const medicalImaging = new AWS.MedicalImaging(params)

const main = async () => {
    try {
        const result = await medicalImaging.searchImageSets({
            datastoreId: '12345678901234567890123456789012',
            searchCriteria: {
                filters: [{
                    values: [{DICOMPatientId: "2CT2"}],
                    operator: "EQUAL"
                }]
            }
        }).promise()
        console.log(util.inspect(result, false, null, true /* enable colors */))
    }
    catch(err) {
        console.log(err)
    }
}

main()
```

このアクションは次の出力を返します。

```
{
    imageSetsMetadataSummaries: [
    {
      imageSetId: 'd6a08ab9fa25326afa3392b8f3c9ca50',
      version: 1,
      createdAt: 2023-03-09T14:55:04.086Z,
      updatedAt: 2023-03-09T14:55:04.086Z,
      DICOMTags: {
        DICOMPatientId: '2CT2',
        DICOMPatientName: 'CompressedSamples^CT2',
        DICOMPatientBirthDate: null,
        DICOMPatientSex: null,
        DICOMStudyInstanceUID: '1.3.6.1.4.1.5962.1.2.2.20040826185059.5457',
```

```
        DICOMStudyId: '2CT2',
        DICOMStudyDescription: null,
        DICOMNumberOfStudyRelatedSeries: 1,
        DICOMNumberOfStudyRelatedInstances: 1,
        DICOMAccessionNumber: null,
        DICOMStudyDate: '20040826',
        DICOMStudyTime: '185059'
      }
    }
    ],
    nextToken: null
}
```

JavaScript ユースケース #2: **DICOMStudyDate**と を使用する **BETWEEN** 演算子 **DICOMStudyTime**

次の検索条件では、1990 年 1 月 1 日 (午前 0 時) から 2023 年 1 月 1 日 (午前 0 時) の間に生成された DICOM スタディを含む画像セットを検索します。

> **Note**
>
> DICOMStudyTime は省略可能です。入力されていない場合は、フィルターで指定された日付けの時間値は午前 0 時 (1 日の始まり) になります。

```
import AWS from 'aws-sdk'
import util from 'util'

const params = {
    region:'us-east-1'
}
const medicalImaging = new AWS.MedicalImaging(params)

const main = async () => {
    try {
        const result = await medicalImaging.searchImageSets({
            datastoreId: '12345678901234567890123456789012',
            searchCriteria: {
                filters: [{
                    values: [{DICOMStudyDateAndTime: {
                        DICOMStudyDate: '19900101',
                        DICOMStudyTime: '000000'
```

```
                    }},
                    {DICOMStudyDateAndTime: {
                        DICOMStudyDate: '20230101',
                        DICOMStudyTime: '000000'
                    }}],
                    operator: "BETWEEN"
                }]
            }
        }).promise()
        console.log(util.inspect(result, false, null, true /* enable colors */))
    }
    catch(err) {
        console.log(err)
    }
}

main()
```

このアクションは次の出力を返します。

```
{
    imageSetsMetadataSummaries: [
    {
      imageSetId: 'd6a08ab9fa25326afa3392b8f3c9ca50',
      version: 1,
      createdAt: 2023-03-09T14:55:04.086Z,
      updatedAt: 2023-03-09T14:55:04.086Z,
      DICOMTags: {
        DICOMPatientId: '2CT2',
        DICOMPatientName: 'CompressedSamples^CT2',
        DICOMPatientBirthDate: null,
        DICOMPatientSex: null,
        DICOMStudyInstanceUID: '1.3.6.1.4.1.5962.1.2.2.20040826185059.5457',
        DICOMStudyId: '2CT2',
        DICOMStudyDescription: null,
        DICOMNumberOfStudyRelatedSeries: 1,
        DICOMNumberOfStudyRelatedInstances: 1,
        DICOMAccessionNumber: null,
        DICOMStudyDate: '20040826',
        DICOMStudyTime: '185059'
      }
    }
    ],
```

```
    nextToken: null
}
```

JavaScript ユースケース #3: を使用する **BETWEEN** 演算子 **createdAt** (タイムトライアルは以前に保持されていた)

次の検索条件では、UTC タイムゾーンの時間範囲 HealthImaging 間で DICOM 値が保持されている画像セットを検索します。

> **Note**
>
> サンプルの形式 ("1985-04-12T23:20:50.52Z") で createdAt を入力します。

```
import AWS from 'aws-sdk'
import util from 'util'

const params = {
    region:'us-east-1'
}
const medicalImaging = new AWS.MedicalImaging(params)

const main = async () => {
    try {
        const result = await medicalImaging.searchImageSets({
            datastoreId: '12345678901234567890123456789012',
            searchCriteria: {
                filters: [{
                    values: [
                        {createdAt: "1985-04-12T23:20:50.52Z"},
                        {createdAt: "2023-05-12T23:20:50.52Z"},
                    ],
                    operator: "BETWEEN"
                }]
            }
        }).promise()
        console.log(util.inspect(result, false, null, true /* enable colors */))
    }
    catch(err) {
        console.log(err)
    }
}
```

```
main()
```

このアクションは次の出力を返します。

```
{
    imageSetsMetadataSummaries: [
    {
      imageSetId: 'd6a08ab9fa25326afa3392b8f3c9ca50',
      version: 1,
      createdAt: 2023-03-09T14:55:04.086Z,
      updatedAt: 2023-03-09T14:55:04.086Z,
      DICOMTags: {
        DICOMPatientId: '2CT2',
        DICOMPatientName: 'CompressedSamples^CT2',
        DICOMPatientBirthDate: null,
        DICOMPatientSex: null,
        DICOMStudyInstanceUID: '1.3.6.1.4.1.5962.1.2.2.20040826185059.5457',
        DICOMStudyId: '2CT2',
        DICOMStudyDescription: null,
        DICOMNumberOfStudyRelatedSeries: 1,
        DICOMNumberOfStudyRelatedInstances: 1,
        DICOMAccessionNumber: null,
        DICOMStudyDate: '20040826',
        DICOMStudyTime: '185059'
      }
    }
    ],
    nextToken: null
}
```

Python

次のコード例では、 を使用して AWS SDK for Python (Boto3) 、複数のユースケースシナリオに基づいて画像セットを検索します。

Python ユースケース #1: **EQUAL** 演算子

以下の検索条件では、EQUAL 演算子を使用し、特定の値に基づいて画像セットを検索します。

```
medical_imaging_client.search_image_sets(
```

```
    datastoreId='12345678901234567890123456789012',
    searchCriteria= {"filters" : [{ "operator": "EQUAL", "values":
 [{"DICOMPatientId": "3524578"}]}]}
)
```

このアクションは次の出力を返します。

```
{
    "imageSetsMetadataSummaries": [{
        "imageSetId": "09876543210987654321098765432109",
        "version": 1,
        "createdAt": datetime.datetime(2022, 11, 15, 14, 47, 39, 387000,
 "tzinfo=tzlocal())"),
        "updatedAt": datetime.datetime(2022, 11, 15, 14, 47, 39, 387000,
 "tzinfo=tzlocal())"),
        "DICOMTags": {
            "DICOMPatientId": "3524578",
            "DICOMPatientName": "MISTER^DX",
            "DICOMStudyInstanceUID":
 "1.2.840.113619.2.67.2158294438.15745010109084247.20000",
            "DICOMStudyId": "300018363",
            "DICOMStudyDescription": "CHEST",
            "DICOMNumberOfStudyRelatedSeries": 1,
            "DICOMNumberOfStudyRelatedInstances": 1,
            "DICOMStudyDate": "20010109",
            "DICOMStudyTime": "084247.000000"
        }
    }]
}
```

Python ユースケース #2: **DICOMStudyDate** と**DICOMStudyTime** を使用する **BETWEEN** 演算子

次の検索条件では、1990 年 1 月 1 日 (午前 0 時) から 2023 年 1 月 1 日 (午前 0 時) の間に生成された DICOM スタディを含む画像セットを検索します。

> **Note**
>
> DICOMStudyTime は省略可能です。入力されていない場合は、フィルターで指定された日付けの時間値は午前 0 時 (1 日の始まり) になります。

```
medical_imaging_client.search_image_sets(
```

```
    datastoreId='12345678901234567890123456789012',
    searchCriteria= {"filters" : [{"operator": "BETWEEN",
 "values":[{"DICOMStudyDateAndTime": {"DICOMStudyDate": "19900101",
 "DICOMStudyTime":"000000"}}, {"DICOMStudyDateAndTime": {"DICOMStudyDate":
 "20230101", "DICOMStudyTime":"000000"}}]}]}
)
```

次の出力が返されます。

```
{
    "imageSetsMetadataSummaries": [{
        "imageSetId": "09876543210987654321098765432109",
        "version": 1,
        "createdAt": datetime.datetime(2022, 11, 15, 14, 47, 39, 387000,
 "tzinfo=tzlocal())"),
        "updatedAt": datetime.datetime(2022, 11, 15, 14, 47, 39, 387000,
 "tzinfo=tzlocal())"),
        "DICOMTags": {
            "DICOMPatientId": "3524578",
            "DICOMPatientName": "MISTER^DX",
            "DICOMStudyInstanceUID":
 "1.2.840.113619.2.67.2158294438.15745010109084247.20000",
            "DICOMStudyId": "300018363",
            "DICOMStudyDescription": "CHEST",
            "DICOMNumberOfStudyRelatedSeries": 1,
            "DICOMNumberOfStudyRelatedInstances": 1,
            "DICOMStudyDate": "20010109",
            "DICOMStudyTime": "084247.000000"
        }
    }]
}
```

Python ユースケース #3: **`createdAt`** を使用した **`BETWEEN`** 演算子 (タイムスタディは以前に保持)

次の検索条件では、UTC タイムゾーンの時間範囲 HealthImaging 間で DICOM 値が保持されている画像セットを検索します。

> **Note**
>
> サンプルの形式 (`"1985-04-12T23:20:50.52Z"`) で `createdAt` を入力します。

```
medical_imaging_client.search_image_sets(
    datastoreId='12345678901234567890123456789012',
    searchCriteria= {"filters" : [{"operator": "BETWEEN", "values":[{"createdAt":
 "1985-04-12T23:20:50.52Z"},{"createdAt": "2023-04-12T23:20:50.52Z"}]}]}
)
```

このアクションは次の出力を返します。

```
{
    "imageSetsMetadataSummaries": [{
        "imageSetId": "09876543210987654321098765432109",
        "version": 1,
        "createdAt": datetime.datetime(2022, 11, 15, 14, 47, 39, 387000,
 "tzinfo=tzlocal())"),
        "updatedAt": datetime.datetime(2022, 11, 15, 14, 47, 39, 387000,
 "tzinfo=tzlocal())"),
        "DICOMTags": {
            "DICOMPatientId": "3524578",
            "DICOMPatientName": "MISTER^DX",
            "DICOMStudyInstanceUID":
 "1.2.840.113619.2.67.2158294438.15745010109084247.20000",
            "DICOMStudyId": "300018363",
            "DICOMStudyDescription": "CHEST",
            "DICOMNumberOfStudyRelatedSeries": 1,
            "DICOMNumberOfStudyRelatedInstances": 1,
            "DICOMStudyDate": "20010109",
            "DICOMStudyTime": "084247.000000"
        }
    }]
}
```

Java

次のコード例では、 を使用して AWS SDK for Java 2.x 、複数のユースケースシナリオに基づいて画像セットを検索します。

Java ユースケース #1: **EQUAL** 演算子

以下の検索条件では、EQUAL 演算子を使用し、特定の値に基づいて画像セットを検索します。

```
final SearchImageSetsRequest request = SearchImageSetsRequest.builder()
                .datastoreId("12345678901234567890123456789012")
                .searchCriteria(SearchCriteria.builder()
                    .filters(List.of(SearchFilter.builder()
                        .withOperator("EQUAL").build()))
                        .withValues(List.of(SearchByAttributeValue.builder()
                            .withDICOMPatientId("SUBJECT08701").build()))
                        .build())
                .build();
final SearchImageSetsResponse response = client
                .searchImageSets(request);
```

このアクションは次の出力を返します。

```
{
    "imageSetsMetadataSummaries": [{
            "imageSetId": "09876543210987654321098765432109",
            "createdAt": "2022-12-06T21:40:59.429000+00:00",
            "version": 1,
            "DICOMTags": {
                "DICOMStudyId": "2011201407",
                "DICOMStudyDate": "19991122",
                "DICOMPatientSex": "F",
                "DICOMStudyInstanceUID":
 "1.2.840.99999999.84710745.943275268089",
                "DICOMPatientBirthDate": "19201120",
                "DICOMStudyDescription": "UNKNOWN",
                "DICOMPatientId": "SUBJECT08701",
                "DICOMPatientName": "Melissa844 Huel628",
                "DICOMNumberOfStudyRelatedInstances": 1,
                "DICOMStudyTime": "140728",
                "DICOMNumberOfStudyRelatedSeries": 1
            },
            "updatedAt": "2022-12-06T21:40:59.429000+00:00"
        }
    ]
}
```

Java ユースケース #2: **DICOMStudyDate** と **DICOMStudyTime** を使用する演算子 **BETWEEN**

次の検索条件では、1990 年 1 月 1 日 (午前 0 時) から 2023 年 1 月 1 日 (午前 0 時) の間に生成された DICOM スタディを含む画像セットを検索します。

> **Note**
>
> DICOMStudyTime は省略可能です。入力されていない場合は、フィルターで指定された日付けの時間値は午前 0 時 (1 日の始まり) になります。

```
final SearchImageSetsRequest request = SearchImageSetsRequest.builder()
                .datastoreId("12345678901234567890123456789012")
                .searchCriteria(SearchCriteria.builder()
                    .filters(List.of(SearchFilter.builder()
                         .withOperator("BETWEEN")
                         .withValues(List.of(SearchByAttributeValue.builder()

 .withDICOMStudyDateAndTime(DICOMStudyDateAndTime.builder()
                            .withDICOMStudyDate("19990101")
                            .withDICOMStudyTime("000000.000").build()
                            ).build(),SearchByAttributeValue.builder()

 .withDICOMStudyDateAndTime(DICOMStudyDateAndTime.builder()
                            .withDICOMStudyDate("20230101")
                            .withDICOMStudyTime("000000.000").build()
                            ).build()))
                        .build())
                .build();

final SearchImageSetsResponse response = client
                .searchImageSets(request);
```

このアクションは次の出力を返します。

```
{
    "imageSetsMetadataSummaries": [{
            "imageSetId": "09876543210987654321098765432109",
            "createdAt": "2022-12-06T21:40:59.429000+00:00",
            "version": 1,
            "DICOMTags": {
                "DICOMStudyId": "2011201407",
                "DICOMStudyDate": "19991122",
                "DICOMPatientSex": "F",
                "DICOMStudyInstanceUID":
 "1.2.840.99999999.84710745.943275268089",
                "DICOMPatientBirthDate": "19201120",
```

```
                "DICOMStudyDescription": "UNKNOWN",
                "DICOMPatientId": "SUBJECT08701",
                "DICOMPatientName": "Melissa844 Huel628",
                "DICOMNumberOfStudyRelatedInstances": 1,
                "DICOMStudyTime": "140728",
                "DICOMNumberOfStudyRelatedSeries": 1
            },
            "updatedAt": "2022-12-06T21:40:59.429000+00:00"
        }
    ]
}
```

Java ユースケース #3: **`createdAt`** を使用した **`BETWEEN`** 演算子 (タイムスタディは以前に保持)

次の検索条件では、UTC タイムゾーンの時間範囲 HealthImaging 間で DICOM 値が保持されている画像セットを検索します。

> **Note**
>
> サンプルの形式 (`"1985-04-12T23:20:50.52Z"`) で `createdAt` を入力します。

```
final SearchImageSetsRequest request = SearchImageSetsRequest.builder()
                .datastoreId("12345678901234567890123456789012")
                .searchCriteria(SearchCriteria.builder()
                    .filters(List.of(SearchFilter.builder()
                        .withValues(List.of(SearchByAttributeValue.builder()

 .withCreatedAt(Instant.parse("1985-04-12T23:20:50.52Z")).build(),
                            SearchByAttributeValue.builder()

 .withCreatedAt(Instant.parse("2022-04-12T23:20:50.52Z")).build())
                        .withOperator("BETWEEN").build()))
                        .build()))
                .build();
final SearchImageSetsResponse response = client
                .searchImageSets(request);
```

このアクションは次の出力を返します。

```
{
```

```
    "imageSetsMetadataSummaries": [{
            "imageSetId": "09876543210987654321098765432109",
            "createdAt": "2022-12-06T21:40:59.429000+00:00",
            "version": 1,
            "DICOMTags": {
                "DICOMStudyId": "2011201407",
                "DICOMStudyDate": "19991122",
                "DICOMPatientSex": "F",
                "DICOMStudyInstanceUID":
 "1.2.840.99999999.84710745.943275268089",
                "DICOMPatientBirthDate": "19201120",
                "DICOMStudyDescription": "UNKNOWN",
                "DICOMPatientId": "SUBJECT08701",
                "DICOMPatientName": "Melissa844 Huel628",
                "DICOMNumberOfStudyRelatedInstances": 1,
                "DICOMStudyTime": "140728",
                "DICOMNumberOfStudyRelatedSeries": 1
            },
            "updatedAt": "2022-12-06T21:40:59.429000+00:00"
        }
    ]
}
```


画像セットのプロパティの取得

GetImageSet アクションを使用して、 で特定のイメージセットのプロパティを返します HealthImaging。以下のタブには、 AWS Management Console および SDK の AWS CLI および コード例の手順が表示されます。 AWS SDKs 詳細については、AWS HealthImaging API リファレンスのGetImageSet「」を参照してください。

> **Note**
>
> デフォルトでは、 はイメージセットの最新バージョンのプロパティ HealthImaging を返します。古いバージョンの画像セットのプロパティを表示するには、 versionId をリクエストに入力します。

画像セットのプロパティを取得するには

のアクセス設定に基づいてタブを選択します HealthImaging。

Console

次の手順では AWS Management Console 、 を使用して画像セットのプロパティを取得します。

1. HealthImaging コンソールのデータストアページを開きます。
2. データストアを選択します。

   データストアの詳細ページが開き、デフォルトで [画像セット] タブが選択されます。
3. 画像セットを選択します。

   画像セットの詳細 ページが開き、画像セットのプロパティが表示されます。

CLI

次のコード例では、 AWS Command Line Interface （ AWS CLI) を使用して画像セットのプロパティを取得します。

```
aws medical-imaging get-image-set \
  --datastore-id 12345678901234567890123456789012 \
  --image-set-id 18f88ac7870584f58d56256646b4d92b \
  --version-id 1
```

このアクションは次の出力を返します。

```
{
    "versionId": "1",
    "imageSetWorkflowStatus": "COPIED",
    "updatedAt": 1680027253.471,
    "imageSetId": "18f88ac7870584f58d56256646b4d92b",
    "imageSetState": "ACTIVE",
    "createdAt": 1679592510.753,
    "datastoreId": "12345678901234567890123456789012"
}
```

JavaScript

次のコード例では AWS SDK for JavaScript 、 を使用して画像セットのプロパティを取得します。

```
var params = {
  datastoreId: '12345678901234567890123456789012', /* required */
```

```
  imageSetId: 'ea92b0d8838c72a3f25d00d13616f87e', /* required */
  versionId: `1` /* optional */
};

var medicalImaging = new AWS.MedicalImaging();
medicalImaging.getImageSet(params, function(err,data) {
     if (err)
        console.log(err, err.stack); // an error occurred
     else
        console.log(data); // successful response
});
```

このアクションは次の出力を返します。

```
{
    datastoreId: "12345678901234567890123456789012";
    imageSetId: "ea92b0d8838c72a3f25d00d13616f87e";
    versionId: "1";
    imageSetState: "ACTIVE";
    imageSetWorkflowStatus: "COPIED";
    createdAt: 1679592510.753;
    updatedAt: 1680027253.471;
}
```

Python

次のコード例では AWS SDK for Python (Boto3) 、 を使用して画像セットのプロパティを取得します。

```
import boto3
hi_client= boto3.client('medical-imaging')
hi_client.get_image_set(datastoreId = "12345678901234567890123456789012", imageSetId
 = "18f88ac7870584f58d56256646b4d92b", versionId = "1")
```

このアクションは次の出力を返します。

```
{'ResponseMetadata': {'RequestId': '27421581-2412-40e5-860c-aca16d07098f',
 'HTTPStatusCode': 200, 'HTTPHeaders': {'date': 'Tue, 28 Mar 2023 17:52:51 GMT',
 'content-type': 'application/json', 'content-length': '259', 'connection':
 'keep-alive', 'x-amzn-requestid': '27421581-2412-40e5-860c-aca16d07098f'},
 'RetryAttempts': 0}, 'datastoreId': '12345678901234567890123456789012',
 'imageSetId': '18f88ac7870584f58d56256646b4d92b', 'versionId': '1',
```

```
 'imageSetState': 'ACTIVE', 'createdAt': datetime.datetime(2023, 3, 23, 10, 28, 30,
 753000, tzinfo=tzlocal()), 'updatedAt': datetime.datetime(2023, 3, 23, 10, 28, 30,
 753000, tzinfo=tzlocal())}
```

Java

次のコード例では AWS SDK for Java 2.x 、 を使用して画像セットのプロパティを取得します。

```
final GetImageSetRequest request = GetImageSetRequest.builder()
        .datastoreId(12345678901234567890123456789012)
        .imageSetId(18f88ac7870584f58d56256646b4d92b)
        .versionId(1)
        .build();
```

このアクションは次の出力を返します。

```
GetImageSetResponse(DatastoreId=12345678901234567890123456789012,
 ImageSetId=18f88ac7870584f58d56256646b4d92b, VersionId=1, ImageSetState=ACTIVE,
 CreatedAt=2022-11-02T06:33:26.115Z, UpdatedAt=2022-11-02T06:33:26.115Z)
```

画像セットメタデータの取得

GetImageSetMetadata アクションを使用して、 で特定の画像セットのメタデータを取得します HealthImaging。以下のタブには、 AWS Management Console および SDK の AWS CLI およびコード例の手順が表示されます。 AWS SDKs 詳細については、AWS HealthImaging API リファレンスのGetImageSetMetadata「」を参照してください。

> **Note**
>
> デフォルトでは、 はイメージセットの最新バージョンのメタデータ属性 HealthImaging を返します。古いバージョンの画像セットのメタデータを表示するには、リクエストにversionIdを付けてください。
> 画像セットのメタデータはgzipで圧縮され、JSON オブジェクトとして返されます。そのため、メタデータを表示する前に JSON オブジェクトを解凍する必要があります。

画像セットのメタデータを取得するには

のアクセス設定に基づいてタブを選択します HealthImaging。

Console

次の手順では AWS Management Console 、 を使用して画像セットのメタデータを取得します。

1. HealthImaging コンソールのデータストアページを開きます。
2. データストアを選択します。

   データストアの詳細ページが開き、デフォルトで [画像セット] タブが選択されます。
3. 画像セットを選択します。

   画像セットの詳細ページが開き、画像セットのメタデータが「画像セットメタデータビューア」セクションの下に表示されます。

CLI

次のコード例では、 AWS Command Line Interface （ AWS CLI) を使用して画像セットのメタデータを取得します。

> **Note**
>
> `--outfile` は必須パラメータです。

バージョンなし:

```
aws medical-imaging get-image-set-metadata \
  --datastore-id 12345678901234567890123456789012 \
  --image-set-id ea92b0d8838c72a3f25d00d13616f87e \
  --outfile studymetadata.json.gz
```

返されたメタデータはgzipで圧縮され、studymetadata.json.gzファイルに保存されます。返された JSON オブジェクトの内容を表示するには、まずオブジェクトを解凍する必要があります。

```
{
    "contentType": "application/json",
    "contentEncoding": "gzip"
}
```

バージョン付き:

```
aws medical-imaging get-image-set-metadata \
  --datastore-id 12345678901234567890123456789012 \
  --image-set-id ea92b0d8838c72a3f25d00d13616f87e \
  --version-id 1 \
  --outfile studymetadata.json.gz
```

返されたメタデータはgzipで圧縮され、studymetadata.json.gzファイルに保存されます。返された JSON オブジェクトの内容を表示するには、まずオブジェクトを解凍する必要があります。

```
{
    "contentType": "application/json",
    "contentEncoding": "gzip"
}
```

JavaScript

次のコード例では AWS SDK for JavaScript 、 を使用して画像セットのメタデータを取得します。

バージョンなし:

```
var params = {
  datastoreId: '12345678901234567890123456789012', /* required */
  imageSetId: 'ea92b0d8838c72a3f25d00d13616f87e' /* required */
};

var medicalImaging = new AWS.MedicalImaging();
medicalImaging.getImageSetMetadata(params, function(err,data) {
     if (err)
        console.log(err, err.stack); // an error occurred
     else
        console.log(data); // successful response
});
```

このアクションは次の出力を返します。

```
{
    imageSetMetadataBlob: ImageSetMetadataBlob;
```

```
    contentType: "application/json";
    contentEncoding: "gzip";
}
```

ImageSetMetadataBlobのタイプはすぐには解析できません。メタデータレスポンスを次のようにバッファリングし、解析可能な形式に変換することをお勧めします。

```
var jsonMetadata = Buffer.from(imageSetMetadataBlob).toString("utf8");
```

変換を実行すると、JSON は次のようになるはずです。

```
{
    'SchemaVersion': '1.0',
    'DatastoreID': '12345678901234567890123456789012',
    'ImageSetID': 'ea92b0d8838c72a3f25d00d13616f87e',
    'Patient': {
        'DICOM': {
            'PatientBirthDate': None,
            'PatientSex': None,
            'PatientID': '9227465',
            'PatientName': 'MISTER^CR'
        }
    },
    'Study': {
        'DICOM': {
            'StudyTime': '100821',
            'ReferringPhysicianName': None,
            'StudyID': None,
            'NameOfPhysiciansReadingStudy': None,
            'StudyDate': '20010109',
            'StudyDescription': 'pelvis',
            'AccessionNumber': '0000000006',
            'StudyInstanceUID': '1.3.51.0.7.633918642.633920010109.6339100821'
        },
        'Series': {
            '1.3.51.5145.15142.20010109.1105627': {
                'DICOM': {
                    'PerformingPhysicianName': None,
                    'SeriesNumber': '1',
                    'StudyInstanceUID':
 '1.3.51.0.7.633918642.633920010109.6339100821',
                    'Laterality': None,
                    'Modality': 'CR',
```

```
                    'BodyPartExamined': 'PELVIS',
                    'SeriesInstanceUID': '1.3.51.5145.15142.20010109.1105627',
                    'SeriesDescription': 'Pelvis'
                },
                'Instances': {
                    '1.3.51.5145.5142.20010109.1105627.1.0.1': {
                        'DICOM': {
                            'SpecificCharacterSet': 'ISO_IR 100',
                            'InstanceNumber': '1',
                            'WindowWidth': '2.80000000E+03',
                            'AcquisitionDeviceProcessingCode': '60141Ia713Ra',
                            'ImageType': ['DERIVED', 'PRIMARY'],
                            'InstitutionalDepartmentName': None,
                            'BitsAllocated': 16,
                            '00191015': '2.33',
                            '00191014': '1.33/2.40',
                            '00191013': 'RP1KT',
                            'MediaStorageSOPClassUID':
'1.2.840.10008.5.1.4.1.1.1',
                            'InstanceCreationTime': '095618',
                            'ImagesInAcquisition': '1',
                            'PrivateCreatorID': 'AGFA',
                            'ContentDate': '20010109',
                            'PlateType': 'AGFATEST',
                            'Manufacturer': 'AGFA',
                            '00191011': 'P BOOGERT_0',
                            'SOPInstanceUID':
'1.3.51.5145.5142.20010109.1105627.1.0.1',
                            '00191010': 'MENU=60141 CC=0 MC=3.00 EC=1.00 LR=2.00
NR=1.00',
                            'SourceApplicationEntityTitle': None,
                            'SOPClassUID': '1.2.840.10008.5.1.4.1.1.1',
                            'SoftwareVersions': 'VIPS1009',
                            'HighBit': 11,
                            'PixelData': None,
                            'ImplementationVersionName': 'OFFIS_DCMTK_354',
                            'ExposuresOnPlate': 1031,
                            'AcquisitionTime': '100821',
                            'InstanceCreationDate': '20010109',
                            'WindowCenter': '1.60000000E+03',
                            'RescaleSlope': '6.83760684E-01',
                            'ViewPosition': 'AP',
                            'SamplesPerPixel': 1,
                            'BitsStored': 12,
```

```
                            'PixelRepresentation': 0,
                            'DeviceSerialNumber': '5142',
                            'ImagerPixelSpacing': ['1.14000000E-01',
 '1.14000000E-01'],
                            'FileMetaInformationVersion': ['1', '1'],
                            'StationName': 'STATION_NAME',
                            'ImplementationClassUID':
 '1.2.276.0.7230010.3.0.3.5.4',
                            'PhotometricInterpretation': 'MONOCHROME1',
                            'MediaStorageSOPInstanceUID':
 '1.3.51.5145.5142.20010109.1105627.1.0.1',
                            'AcquisitionDate': '20010109',
                            'PlateID': 'F8DBBT',
                            'Rows': 3062,
                            'TransferSyntaxUID': '1.2.840.10008.1.2',
                            'InstitutionName': None,
                            'ManufacturerModelName': 'ADC_51xx',
                            'CassetteOrientation': 'LANDSCAPE',
                            'ImageComments': 'JAN 09 2001',
                            'Columns': 3730,
                            'RescaleIntercept': '2.00000000E+02',
                            'CassetteSize': '35CMX43CM',
                            'ContentTime': '100821',
                            'Sensitivity': '4.00000000E+02',
                            'PixelSpacing': ['1.14000000E-01', '1.14000000E-01'],
                            'RescaleType': 'OD'
                        },
                        'ImageFrames': [{
                            'ID': '678906789067890123451234512'
                        }]
                    }
                }
            }
        }
    }
}
```

バージョン付き:

```
var params = {
  datastoreId: '12345678901234567890123456789012', /* required */
  imageSetId: 'ea92b0d8838c72a3f25d00d13616f87e', /* required */
  versionId: '1'
```

```
};

var medicalImagingRuntimeOperations = new AWS.MedicalImagingRuntimeOperations();
medicalImagingRuntimeOperations.getImageSetMetadata(params, function(err,data) {
     if (err)
        console.log(err, err.stack); // an error occurred
     else
        console.log(data); // successful response
});
```

このアクションは次の出力を返します。

```
{
    imageSetMetadataBlob: ImageSetMetadataBlob;
    contentType: "application/json";
    contentEncoding: "gzip";
}
```

ImageSetMetadataBlobのタイプはすぐには解析できません。メタデータレスポンスを次のようにバッファリングし、解析可能な形式に変換することをお勧めします。

```
var jsonMetadata = Buffer.from(imageSetMetadataBlob).toString("utf8");
```

変換を実行すると、JSON は次のようになるはずです。

```
{
    'SchemaVersion': '1.0',
    'DatastoreID': '12345678901234567890123456789012',
    'ImageSetID': 'ea92b0d8838c72a3f25d00d13616f87e',
    'Patient': {
        'DICOM': {
            'PatientBirthDate': None,
            'PatientSex': None,
            'PatientID': '9227465',
            'PatientName': 'MISTER^CR'
        }
    },
    'Study': {
        'DICOM': {
            'StudyTime': '100821',
            'ReferringPhysicianName': None,
            'StudyID': None,
```

```
            'NameOfPhysiciansReadingStudy': None,
            'StudyDate': '20010109',
            'StudyDescription': 'pelvis',
            'AccessionNumber': '0000000006',
            'StudyInstanceUID': '1.3.51.0.7.633918642.633920010109.6339100821'
        },
        'Series': {
            '1.3.51.5145.15142.20010109.1105627': {
                'DICOM': {
                    'PerformingPhysicianName': None,
                    'SeriesNumber': '1',
                    'StudyInstanceUID':
'1.3.51.0.7.633918642.633920010109.6339100821',
                    'Laterality': None,
                    'Modality': 'CR',
                    'BodyPartExamined': 'PELVIS',
                    'SeriesInstanceUID': '1.3.51.5145.15142.20010109.1105627',
                    'SeriesDescription': 'Pelvis'
                },
                'Instances': {
                    '1.3.51.5145.5142.20010109.1105627.1.0.1': {
                        'DICOM': {
                            'SpecificCharacterSet': 'ISO_IR 100',
                            'InstanceNumber': '1',
                            'WindowWidth': '2.80000000E+03',
                            'AcquisitionDeviceProcessingCode': '60141Ia713Ra',
                            'ImageType': ['DERIVED', 'PRIMARY'],
                            'InstitutionalDepartmentName': None,
                            'BitsAllocated': 16,
                            '00191015': '2.33',
                            '00191014': '1.33/2.40',
                            '00191013': 'RP1KT',
                            'MediaStorageSOPClassUID':
'1.2.840.10008.5.1.4.1.1.1',
                            'InstanceCreationTime': '095618',
                            'ImagesInAcquisition': '1',
                            'PrivateCreatorID': 'AGFA',
                            'ContentDate': '20010109',
                            'PlateType': 'AGFATEST',
                            'Manufacturer': 'AGFA',
                            '00191011': 'P BOOGERT_0',
                            'SOPInstanceUID':
'1.3.51.5145.5142.20010109.1105627.1.0.1',
```

```
                            '00191010': 'MENU=60141 CC=0 MC=3.00 EC=1.00 LR=2.00
NR=1.00',
                            'SourceApplicationEntityTitle': None,
                            'SOPClassUID': '1.2.840.10008.5.1.4.1.1.1',
                            'SoftwareVersions': 'VIPS1009',
                            'HighBit': 11,
                            'PixelData': None,
                            'ImplementationVersionName': 'OFFIS_DCMTK_354',
                            'ExposuresOnPlate': 1031,
                            'AcquisitionTime': '100821',
                            'InstanceCreationDate': '20010109',
                            'WindowCenter': '1.60000000E+03',
                            'RescaleSlope': '6.83760684E-01',
                            'ViewPosition': 'AP',
                            'SamplesPerPixel': 1,
                            'BitsStored': 12,
                            'PixelRepresentation': 0,
                            'DeviceSerialNumber': '5142',
                            'ImagerPixelSpacing': ['1.14000000E-01',
'1.14000000E-01'],
                            'FileMetaInformationVersion': ['1', '1'],
                            'StationName': 'STATION_NAME',
                            'ImplementationClassUID':
'1.2.276.0.7230010.3.0.3.5.4',
                            'PhotometricInterpretation': 'MONOCHROME1',
                            'MediaStorageSOPInstanceUID':
'1.3.51.5145.5142.20010109.1105627.1.0.1',
                            'AcquisitionDate': '20010109',
                            'PlateID': 'F8DBBT',
                            'Rows': 3062,
                            'TransferSyntaxUID': '1.2.840.10008.1.2',
                            'InstitutionName': None,
                            'ManufacturerModelName': 'ADC_51xx',
                            'CassetteOrientation': 'LANDSCAPE',
                            'ImageComments': 'JAN 09 2001',
                            'Columns': 3730,
                            'RescaleIntercept': '2.00000000E+02',
                            'CassetteSize': '35CMX43CM',
                            'ContentTime': '100821',
                            'Sensitivity': '4.00000000E+02',
                            'PixelSpacing': ['1.14000000E-01', '1.14000000E-01'],
                            'RescaleType': 'OD'
                        },
                        'ImageFrames': [{
```

```
                            'ID': '678906789067890123451234512345 12'
                        }]
                    }
                }
            }
        }
    }
}
```

Python

次のコード例では AWS SDK for Python (Boto3) 、 を使用して画像セットのメタデータを取得します。

バージョンなし:

```
import boto3
hi_client= boto3.client('medical-imaging')
hi_client.get_image_set_metadata(datastoreId =
 "12345678901234567890123456789012", imageSetId =
 "ea92b0d8838c72a3f25d00d13616f87e")
```

返されたメタデータはgzipで圧縮され、studymetadata.json.gzファイルに保存されます。返された JSON オブジェクトの内容を表示するには、まずオブジェクトを解凍する必要があります。

```
{'ResponseMetadata': {'RequestId': '7b2b43d4-46ec-4e93-b4a0-3d7eb9f56c9d',
 'HTTPStatusCode': 200, 'HTTPHeaders': {'date': 'Tue, 28 Mar 2023 23:01:37 GMT',
 'content-type': 'application/json', 'content-length': '2810', 'connection':
 'keep-alive', 'x-amzn-requestid': '7b2b43d4-46ec-4e93-b4a0-3d7eb9f56c9d',
 'content-encoding': 'gzip'}, 'RetryAttempts': 0}, 'contentType':
 'application/json', 'contentEncoding': 'gzip', 'imageSetMetadataBlob':
 <botocore.response.StreamingBody object at 0x107c62a10>}
```

バージョン付き:

```
import boto3
hi_client= boto3.client('medical-imaging')
hi_client.get_image_set_metadata(datastoreId =
 "12345678901234567890123456789012", imageSetId =
 "ea92b0d8838c72a3f25d00d13616f87e", versionId = "1")
```

返されたメタデータはgzipで圧縮され、studymetadata.json.gzファイルに保存されます。返された JSON オブジェクトの内容を表示するには、まずオブジェクトを解凍する必要があります。

```
{'ResponseMetadata': {'RequestId': '7b1e6711-c10f-482f-b396-fad17c88dd95',
 'HTTPStatusCode': 200, 'HTTPHeaders': {'date': 'Tue, 28 Mar 2023 23:02:20
 GMT', 'content-type': 'application/json', 'content-length': '2804',
 'connection': 'keep-alive', 'x-amzn-requestid': '7b1e6711-c10f-482f-b396-
fad17c88dd95', 'content-encoding': 'gzip'}, 'RetryAttempts': 0}, 'contentType':
 'application/json', 'contentEncoding': 'gzip', 'imageSetMetadataBlob':
 <botocore.response.StreamingBody object at 0x107c62a40>}
```

Java

次のコード例では AWS SDK for Java 2.x 、 を使用して画像セットのメタデータを取得します。

バージョンなし:

```
final GetImageSetMetadataRequest request = GetImageSetMetadataRequest.builder()
                .datastoreId(1234567890123456789012345678901 2)
                .imageSetId(ea92b0d8838c72a3f25d00d13616f87e)
                .build();
```

返されたメタデータはgzipで圧縮され、studymetadata.json.gzファイルに保存されます。圧縮解除後、次の出力が JSON に返されます。

```
{
    'SchemaVersion': '1.0',
    'DatastoreID': '1234567890123456789012345678901 2',
    'ImageSetID': 'ea92b0d8838c72a3f25d00d13616f87e',
    'Patient': {
        'DICOM': {
            'PatientBirthDate': None,
            'PatientSex': None,
            'PatientID': '9227465',
            'PatientName': 'MISTER^CR'
        }
    },
    'Study': {
        'DICOM': {
            'StudyTime': '100821',
```

```
            'ReferringPhysicianName': None,
            'StudyID': None,
            'NameOfPhysiciansReadingStudy': None,
            'StudyDate': '20010109',
            'StudyDescription': 'pelvis',
            'AccessionNumber': '0000000006',
            'StudyInstanceUID': '1.3.51.0.7.633918642.633920010109.6339100821'
        },
        'Series': {
            '1.3.51.5145.15142.20010109.1105627': {
                'DICOM': {
                    'PerformingPhysicianName': None,
                    'SeriesNumber': '1',
                    'StudyInstanceUID':
'1.3.51.0.7.633918642.633920010109.6339100821',
                    'Laterality': None,
                    'Modality': 'CR',
                    'BodyPartExamined': 'PELVIS',
                    'SeriesInstanceUID': '1.3.51.5145.15142.20010109.1105627',
                    'SeriesDescription': 'Pelvis'
                },
                'Instances': {
                    '1.3.51.5145.5142.20010109.1105627.1.0.1': {
                        'DICOM': {
                            'SpecificCharacterSet': 'ISO_IR 100',
                            'InstanceNumber': '1',
                            'WindowWidth': '2.80000000E+03',
                            'AcquisitionDeviceProcessingCode': '60141Ia713Ra',
                            'ImageType': ['DERIVED', 'PRIMARY'],
                            'InstitutionalDepartmentName': None,
                            'BitsAllocated': 16,
                            '00191015': '2.33',
                            '00191014': '1.33/2.40',
                            '00191013': 'RP1KT',
                            'MediaStorageSOPClassUID':
'1.2.840.10008.5.1.4.1.1.1',
                            'InstanceCreationTime': '095618',
                            'ImagesInAcquisition': '1',
                            'PrivateCreatorID': 'AGFA',
                            'ContentDate': '20010109',
                            'PlateType': 'AGFATEST',
                            'Manufacturer': 'AGFA',
                            '00191011': 'P BOOGERT_0',
```

```
                            'SOPInstanceUID':
'1.3.51.5145.5142.20010109.1105627.1.0.1',
                            '00191010': 'MENU=60141 CC=0 MC=3.00 EC=1.00 LR=2.00
NR=1.00',
                            'SourceApplicationEntityTitle': None,
                            'SOPClassUID': '1.2.840.10008.5.1.4.1.1.1',
                            'SoftwareVersions': 'VIPS1009',
                            'HighBit': 11,
                            'PixelData': None,
                            'ImplementationVersionName': 'OFFIS_DCMTK_354',
                            'ExposuresOnPlate': 1031,
                            'AcquisitionTime': '100821',
                            'InstanceCreationDate': '20010109',
                            'WindowCenter': '1.60000000E+03',
                            'RescaleSlope': '6.83760684E-01',
                            'ViewPosition': 'AP',
                            'SamplesPerPixel': 1,
                            'BitsStored': 12,
                            'PixelRepresentation': 0,
                            'DeviceSerialNumber': '5142',
                            'ImagerPixelSpacing': ['1.14000000E-01',
'1.14000000E-01'],
                            'FileMetaInformationVersion': ['1', '1'],
                            'StationName': 'STATION_NAME',
                            'ImplementationClassUID':
'1.2.276.0.7230010.3.0.3.5.4',
                            'PhotometricInterpretation': 'MONOCHROME1',
                            'MediaStorageSOPInstanceUID':
'1.3.51.5145.5142.20010109.1105627.1.0.1',
                            'AcquisitionDate': '20010109',
                            'PlateID': 'F8DBBT',
                            'Rows': 3062,
                            'TransferSyntaxUID': '1.2.840.10008.1.2',
                            'InstitutionName': None,
                            'ManufacturerModelName': 'ADC_51xx',
                            'CassetteOrientation': 'LANDSCAPE',
                            'ImageComments': 'JAN 09 2001',
                            'Columns': 3730,
                            'RescaleIntercept': '2.00000000E+02',
                            'CassetteSize': '35CMX43CM',
                            'ContentTime': '100821',
                            'Sensitivity': '4.00000000E+02',
                            'PixelSpacing': ['1.14000000E-01', '1.14000000E-01'],
                            'RescaleType': 'OD'
```

```
                        },
                        'ImageFrames': [{
                            'ID': '678906789067890123451234512'
                        }]
                    }
                }
            }
        }
}
```

バージョン付き:

```
final GetImageSetMetadataRequest request = GetImageSetMetadataRequest.builder()
            .datastoreId(12345678901234567890123456789012)
            .imageSetId(ea92b0d8838c72a3f25d00d13616f87e)
            .versionId(1)
            .build();
```

返されたメタデータはgzipで圧縮され、studymetadata.json.gzファイルに保存されます。圧縮解除後、次の出力が JSON に返されます。

```
{
    'SchemaVersion': '1.0',
    'DatastoreID': '12345678901234567890123456789012',
    'ImageSetID': 'ea92b0d8838c72a3f25d00d13616f87e',
    'Patient': {
        'DICOM': {
            'PatientBirthDate': None,
            'PatientSex': None,
            'PatientID': '9227465',
            'PatientName': 'MISTER^CR'
        }
    },
    'Study': {
        'DICOM': {
            'StudyTime': '100821',
            'ReferringPhysicianName': None,
            'StudyID': None,
            'NameOfPhysiciansReadingStudy': None,
            'StudyDate': '20010109',
            'StudyDescription': 'pelvis',
            'AccessionNumber': '0000000006',
```

```
            'StudyInstanceUID': '1.3.51.0.7.633918642.633920010109.6339100821'
        },
        'Series': {
            '1.3.51.5145.15142.20010109.1105627': {
                'DICOM': {
                    'PerformingPhysicianName': None,
                    'SeriesNumber': '1',
                    'StudyInstanceUID':
'1.3.51.0.7.633918642.633920010109.6339100821',
                    'Laterality': None,
                    'Modality': 'CR',
                    'BodyPartExamined': 'PELVIS',
                    'SeriesInstanceUID': '1.3.51.5145.15142.20010109.1105627',
                    'SeriesDescription': 'Pelvis'
                },
                'Instances': {
                    '1.3.51.5145.5142.20010109.1105627.1.0.1': {
                        'DICOM': {
                            'SpecificCharacterSet': 'ISO_IR 100',
                            'InstanceNumber': '1',
                            'WindowWidth': '2.80000000E+03',
                            'AcquisitionDeviceProcessingCode': '60141Ia713Ra',
                            'ImageType': ['DERIVED', 'PRIMARY'],
                            'InstitutionalDepartmentName': None,
                            'BitsAllocated': 16,
                            '00191015': '2.33',
                            '00191014': '1.33/2.40',
                            '00191013': 'RP1KT',
                            'MediaStorageSOPClassUID':
'1.2.840.10008.5.1.4.1.1.1',
                            'InstanceCreationTime': '095618',
                            'ImagesInAcquisition': '1',
                            'PrivateCreatorID': 'AGFA',
                            'ContentDate': '20010109',
                            'PlateType': 'AGFATEST',
                            'Manufacturer': 'AGFA',
                            '00191011': 'P BOOGERT_0',
                            'SOPInstanceUID':
'1.3.51.5145.5142.20010109.1105627.1.0.1',
                            '00191010': 'MENU=60141 CC=0 MC=3.00 EC=1.00 LR=2.00
NR=1.00',
                            'SourceApplicationEntityTitle': None,
                            'SOPClassUID': '1.2.840.10008.5.1.4.1.1.1',
                            'SoftwareVersions': 'VIPS1009',
```

```
                            'HighBit': 11,
                            'PixelData': None,
                            'ImplementationVersionName': 'OFFIS_DCMTK_354',
                            'ExposuresOnPlate': 1031,
                            'AcquisitionTime': '100821',
                            'InstanceCreationDate': '20010109',
                            'WindowCenter': '1.60000000E+03',
                            'RescaleSlope': '6.83760684E-01',
                            'ViewPosition': 'AP',
                            'SamplesPerPixel': 1,
                            'BitsStored': 12,
                            'PixelRepresentation': 0,
                            'DeviceSerialNumber': '5142',
                            'ImagerPixelSpacing': ['1.14000000E-01',
'1.14000000E-01'],
                            'FileMetaInformationVersion': ['1', '1'],
                            'StationName': 'STATION_NAME',
                            'ImplementationClassUID':
'1.2.276.0.7230010.3.0.3.5.4',
                            'PhotometricInterpretation': 'MONOCHROME1',
                            'MediaStorageSOPInstanceUID':
'1.3.51.5145.5142.20010109.1105627.1.0.1',
                            'AcquisitionDate': '20010109',
                            'PlateID': 'F8DBBT',
                            'Rows': 3062,
                            'TransferSyntaxUID': '1.2.840.10008.1.2',
                            'InstitutionName': None,
                            'ManufacturerModelName': 'ADC_51xx',
                            'CassetteOrientation': 'LANDSCAPE',
                            'ImageComments': 'JAN 09 2001',
                            'Columns': 3730,
                            'RescaleIntercept': '2.00000000E+02',
                            'CassetteSize': '35CMX43CM',
                            'ContentTime': '100821',
                            'Sensitivity': '4.00000000E+02',
                            'PixelSpacing': ['1.14000000E-01', '1.14000000E-01'],
                            'RescaleType': 'OD'
                        },
                        'ImageFrames': [{
                            'ID': '678906789067890123451234512345 12'
                        }]
                    }
                }
            }
```

```
        }
    }
}
```

画像セットのピクセルデータの取得

画像フレームは、2D 医療画像を構成する画像セット内にあるピクセルデータです。GetImageFrame アクションを使用して、 で設定された特定の画像に対して HTJ2K-encoded画像フレームを取得します HealthImaging。以下のタブには、 AWS Management Console および SDK の AWS CLI および コード例の手順が表示されます。 AWS SDKs 詳細については、AWS HealthImaging API リファレンスのGetImageFrame「」を参照してください。

画像フレームを取得するには

のアクセス設定に基づいてタブを選択します HealthImaging。

Console

> **Note**
>
> AWS Management Consoleには画像ビューアが組み込まれていないため、画像フレームをデコードしてプログラムでアクセスする必要があります。
> 画像フレームのデコードと表示の詳細については、HTJ2K デコーディングライブラリを参照してください。

CLI

次のコード例では、 AWS Command Line Interface （ AWS CLI) を使用してイメージフレームを取得します。

```
aws medical-imaging get-image-frame \
  --region us-east-1 \
  --datastore-id "12345678901234567890123456789012" \
  --image-set-id "98765412345612345678907890789012" \
  --image-frame-information imageFrameId=3abf5d5d7ae72f80a0ec81b2c0de3ef4 \
  imageframe.jph
```

> **Note**
>
> このコード例には出力は含まれていません。GetImageFrameアクションは、ピクセルデータのストリームをimageframe.jphファイルに返すからです。画像フレームのデコードと表示については、HTJ2K デコーディングライブラリを参照してください。

JavaScript

次のコード例では AWS SDK for JavaScript 、 を使用してイメージフレームを取得します。

```
import AWS from 'aws-sdk'

const params = {
    region:'us-east-1'
}
const medicalImaging = new AWS.MedicalImaging(params)

const main = async () => {
    try {
        const result = await medicalImaging.getImageFrame({
            datastoreId: '12345678901234567890123456789012',
            imageSetId: 'd6a08ab9fa25326afa3392b8f3c9ca50',
            imageFrameInformation: {imageFrameId :
 '7a3d2b7157d1c2fd575f8992c5061f00'}
        }).promise()
        console.log(result)
    }
    catch(err) {
        console.log(err)
    }
}
```

> **Note**
>
> このコード例には出力が含まれていません。GetImageFrameアクションは、ピクセルデータのストリームを返すため、表示する前にデコードする必要があるからです。画像フレームのデコードと表示については、HTJ2K デコーディングライブラリを参照してください。

Python

次のコード例では AWS SDK for Python (Boto3) 、 を使用してイメージフレームを取得します。

```
medical_imaging.get_image_frame(
    datastoreId="12345678901234567890123456789012",
    imageSetId="98765412345612345678907890789012",
    imageFrameInformation={imageFrameId : "67890678906789012345123451234512"},
)
```

> **Note**
>
> このコード例には出力が含まれていません。GetImageFrameアクションは、ピクセルデータのストリームを返すため、表示する前にデコードする必要があるからです。画像フレームのデコードと表示については、HTJ2K デコーディングライブラリを参照してください。

Java

次のコード例では AWS SDK for Java 2.x 、 を使用してイメージフレームを取得します。

```
final GetImageFrameRequest getImageFrameRequest = GetImageFrameRequest.builder()
            .datastoreId("12345678901234567890123456789012")
            .imageSetId("98765412345612345678907890789012")

 .imageFrameInformation(ImageFrameInformation.builder().imageFrameId("678906789067890123451
            .build();
```

> **Note**
>
> このコード例には出力が含まれていません。GetImageFrameアクションは、ピクセルデータのストリームを返すため、表示する前にデコードする必要があるからです。画像フレームのデコードと表示については、HTJ2K デコーディングライブラリを参照してください。

AWS HealthImaging による画像セットの変更

DICOM インポートジョブでは、通常、以下の理由で画像セットを変更する必要があります。

- 患者の安全
- データ整合性
- ストレージコストの削減

HealthImaging には、画像セットの変更プロセスを簡素化するためにいくつかの API が用意されています。以下のトピックでは、AWS CLI および AWS SDK を使用して画像セットを変更する方法について説明します。

トピック

画像セットのバージョンを一覧表示する

`ListImageSetVersions` アクションを使用して、 で画像セットのバージョン履歴を一覧表示します HealthImaging。以下のタブには、 AWS Management Console および SDK の AWS CLI およびコード例の手順が表示されます。 AWS SDKs 詳細については、AWS HealthImaging API リファレンスの`ListImageSetVersions`「」を参照してください。

> **Note**
>
> HealthImaging は、画像セットに加えられたすべての変更を記録します。画像セットのメタデータを更新すると、画像セット履歴に新しいバージョンが作成されます。詳細については、「画像セットメタデータの更新」を参照してください。

画像セットのバージョンを一覧表示するには

のアクセス設定に基づいてタブを選択します HealthImaging。

Console

次の手順では AWS Management Console 、 を使用して画像セットのバージョンを一覧表示します。

1. HealthImaging コンソールのデータストアページを開きます。
2. データストアを選択します。

   データストアの詳細ページが開き、デフォルトで [画像セット] タブが選択されます。
3. 画像セットを選択します。

   画像セットの詳細ページが開きます。

   画像セットのバージョンは、「画像セットの詳細」セクションに表示されます。

CLI

次のコード例では、 AWS Command Line Interface （ AWS CLI) を使用して画像セットのバージョン履歴を一覧表示します。

```
aws medical-imaging list-image-set-versions \
  --datastore-id 12345678901234567890123456789012 \
  --image-set-id ea92b0d8838c72a3f25d00d13616f87e
```

このアクションは次の出力を返します。

```
{
    "imageSetPropertiesList": [
        {
            "ImageSetWorkflowStatus": "UPDATED",
            "versionId": "4",
            "updatedAt": 1680029436.304,
            "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
            "imageSetState": "ACTIVE",
            "createdAt": 1680027126.436
        },
        {
            "ImageSetWorkflowStatus": "UPDATED",
            "versionId": "3",
            "updatedAt": 1680029163.325,
            "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
```

```
            "imageSetState": "ACTIVE",
            "createdAt": 1680027126.436
        },
        {
            "ImageSetWorkflowStatus": "COPY_FAILED",
            "versionId": "2",
            "updatedAt": 1680027455.944,
            "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
            "imageSetState": "ACTIVE",
            "message": "INVALID_REQUEST:  Series of SourceImageSet and
 DestinationImageSet don't match.",
            "createdAt": 1680027126.436
        },
        {
            "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
            "imageSetState": "ACTIVE",
            "versionId": "1",
            "ImageSetWorkflowStatus": "COPIED",
            "createdAt": 1680027126.436
        }
    ]
}
```

JavaScript

次のコード例では AWS SDK for JavaScript 、 を使用して画像セットのバージョン履歴を一覧表示します。

```
var params = {
  datastoreId: '12345678901234567890123456789012', /* required */
  imageSetId: 'ea92b0d8838c72a3f25d00d13616f87e' /* required */
};

var medicalImaging = new AWS.MedicalImaging();
medicalImaging.listImageSetVersions(params, function(err,data) {
     if (err)
        console.log(err, err.stack); // an error occurred
     else
        console.log(data); // successful response
});
```

このアクションは次の出力を返します。

```
{
    "imageSetPropertiesList": [
        {
            "ImageSetWorkflowStatus": "UPDATED",
            "versionId": "4",
            "updatedAt": 1680029436.304,
            "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
            "imageSetState": "ACTIVE",
            "createdAt": 1680027126.436
        },
        {
            "ImageSetWorkflowStatus": "UPDATED",
            "versionId": "3",
            "updatedAt": 1680029163.325,
            "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
            "imageSetState": "ACTIVE",
            "createdAt": 1680027126.436
        },
        {
            "ImageSetWorkflowStatus": "COPY_FAILED",
            "versionId": "2",
            "updatedAt": 1680027455.944,
            "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
            "imageSetState": "ACTIVE",
            "message": "INVALID_REQUEST:  Series of SourceImageSet and
 DestinationImageSet don't match.",
            "createdAt": 1680027126.436
        },
        {
            "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
            "imageSetState": "ACTIVE",
            "versionId": "1",
            "ImageSetWorkflowStatus": "COPIED",
            "createdAt": 1680027126.436
        }
    ]
}
```

Python

次のコード例では AWS SDK for Python (Boto3) 、 を使用して画像セットのバージョン履歴を一覧表示します。

```
import boto3
hi_client = boto3.client('medical-imaging')
hi_client.list_image_set_versions(datastoreId = "12345678901234567890123456789012",
 imageSetId = "ea92b0d8838c72a3f25d00d13616f87e")
```

このアクションは次の出力を返します。

```
{'ResponseMetadata': {'RequestId': 'd978afe1-4b58-4da6-bdc3-171304a7b047',
 'HTTPStatusCode': 200, 'HTTPHeaders': {'date': 'Tue, 28 Mar 2023 18:34:50
 GMT', 'content-type': 'application/json', 'content-length': '601',
 'connection': 'keep-alive', 'x-amzn-requestid': 'd978afe1-4b58-4da6-
bdc3-171304a7b047'}, 'RetryAttempts': 0}, 'imageSetPropertiesList': [{'imageSetId':
 'ea92b0d8838c72a3f25d00d13616f87e', 'versionId': '2', 'imageSetState': 'ACTIVE',
 'ImageSetWorkflowStatus': 'COPY_FAILED', 'createdAt': datetime.datetime(2023, 3,
 28, 11, 12, 6, 436000, tzinfo=tzlocal()), 'updatedAt': datetime.datetime(2023,
 3, 28, 11, 17, 35, 944000, tzinfo=tzlocal()), 'message': "INVALID_REQUEST:
 Series of SourceImageSet and DestinationImageSet don't match."}, {'imageSetId':
 'ea92b0d8838c72a3f25d00d13616f87e', 'versionId': '1', 'imageSetState': 'ACTIVE',
 'ImageSetWorkflowStatus': 'COPIED', 'createdAt': datetime.datetime(2023, 3, 28, 11,
 12, 6, 436000, tzinfo=tzlocal())}]}
```

Java

次のコード例では AWS SDK for Java 2.x 、 を使用して画像セットのバージョン履歴を一覧表示します。

```
final ListImageSetVersionsRequest request = ListImageSetVersionsRequest.builder()
        .datastoreId(12345678901234567890123456789012)
        .imageSetId(ea92b0d8838c72a3f25d00d13616f87e)
        .build();
```

このアクションは次の出力を返します。

```
ListImageSetVersionsResponse(ImageSetPropertiesList=[ImageSetProperties
 (ImageSetId=ea92b0d8838c72a3f25d00d13616f87e, VersionId=1, ImageSetState=ACTIVE,
 CreatedAt=2022-11-02T06:33:26.115Z, UpdatedAt=2022-11-02T06:33:26.115Z)])
```

画像セットメタデータの更新

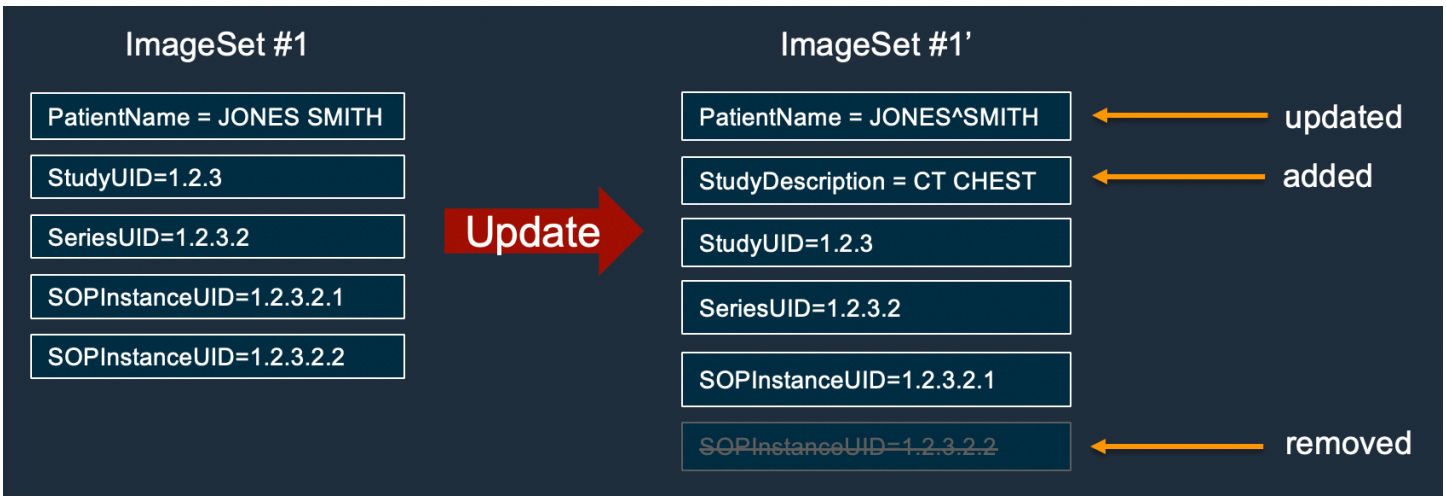
UpdateImageSetMetadata アクションを使用して、AWS のイメージセットメタデータを更新します HealthImaging。この非同期プロセスを使用して、インポート中に作成される DICOM 正規化要素の兆候である画像セットメタデータ属性を追加、更新、削除できます。UpdateImageSetMetadataアクションを使用して、シリーズインスタンスとSOPインスタンスを削除し、画像セットを外部システムと同期させたり、画像セットのメタデータを匿名化したりすることもできます。

画像セットのメタデータの更新について

実際の DICOM インポートでは、画像セットメタデータの属性を更新、追加、削除する必要があります。

> **Note**
>
> 画像セットのメタデータを更新すると、画像セット履歴に新しいバージョンが作成されます。詳細については、「画像セットのバージョンを一覧表示する」を参照してください。
> 画像セットメタデータの更新は非同期プロセスです。そのため、state (imageSetState) フィールドと status (imageSetStatus) フィールドを使用して、ロックされた画像セットに対してどのような操作が行われているかを知ることができます。ロックされた画像セットには他の書き込み操作は実行できません。
> DICOM 要素の制約はメタデータの更新に適用されます。詳細については、「DICOM メタデータの制約」を参照してください。

次の図は、 で更新されている画像セットメタデータを示しています HealthImaging。



次のタブには、 AWS CLI および AWS SDKs。詳細については、AWS HealthImaging API リファレンスのUpdateImageSetMetadata「」を参照してください。

画像セットのメタデータを更新するには

へのアクセス設定に基づいてタブを選択します HealthImaging。

CLI

次のコード例では、 AWS Command Line Interface （ AWS CLI) を使用して画像セットメタデータを更新します。

```
aws medical-imaging update-image-set-metadata \
  --datastore-id 12345678901234567890123456789012 \
  --image-set-id ea92b0d8838c72a3f25d00d13616f87e \
  --latest-version-id 4 \
  --update-image-set-metadata-updates '{"DICOMUpdates": {"updatableAttributes":
 "{\"SchemaVersion\":1.1,\"Patient\":{\"DICOM\":{\"PatientName\":\"Garcia^Gloria
\"}}}"}}'
```

このアクションは次の出力を返します。

```
{
    "latestVersionId": "5",
    "imageSetWorkflowStatus": "UPDATING",
    "updatedAt": 1680042257.908,
    "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
    "imageSetState": "LOCKED",
    "createdAt": 1680027126.436,
    "datastoreId": "12345678901234567890123456789012"
}
```

JavaScript

次のコード例では AWS SDK for JavaScript 、 を使用して画像セットのメタデータを更新します。

```
var params = {
  datastoreId: '12345678901234567890123456789012', /* required */
  imageSetId: 'ea92b0d8838c72a3f25d00d13616f87e', /* required */
  latestVersionId: '4', /* required */
  updateImageSetMetadataUpdates:
    {
```

```
        DICOMUpdates: {
            updatableAttributes: "{
                \"SchemaVersion\": 1.1,
                \"Patient\": {
                    \"DICOM\": {
                        \"PatientName\": \"Garcia^Gloria\"
                    }
                }
            }"
        }
    } /* required */
};

var medicalImaging = new AWS.MedicalImaging();
medicalImaging.updateImageSetMetadata(params, function(err,data) {
     if (err)
        console.log(err, err.stack); // an error occurred
     else
        console.log(data); // successful response
});
```

このアクションは次の出力を返します。

```
{
    "latestVersionId": "5",
    "imageSetWorkflowStatus": "UPDATING",
    "updatedAt": 1680042257.908,
    "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
    "imageSetState": "LOCKED",
    "createdAt": 1680027126.436,
    "datastoreId": "12345678901234567890123456789012"
}
```

Python

次のコード例では AWS SDK for Python (Boto3) 、 を使用して画像セットのメタデータを更新します。

```
import boto3
hi_client= boto3.client('medical-imaging')

updatable_string = "{\"SchemaVersion\":1.1,\"Patient\":{\"DICOM\":{\"PatientName\":
\"MX^MX\"}}}"
```

```
removable_string = "{\"SchemaVersion\":1.1,\"Patient\":{\"DICOM\":{\"PatientSex\":
\"M\"}}}";
hi_client.update_image_set_metadata(datastoreId =
 "12345678901234567890123456789012", imageSetId =
 "ea92b0d8838c72a3f25d00d13616f87e", latestVersionId = "2",
 updateImageSetMetadataUpdates = { "DICOMUpdates" : { "updatableAttributes" :
 updatable_string , "removableAttributes" : removable_string}})
```

このアクションは次の出力を返します。

```
{'ResponseMetadata': {'RequestId': '27259785-3562-48a0-808e-49dc0f1bbbc2',
 'HTTPStatusCode': 200, 'HTTPHeaders': {'date': 'Tue, 28 Mar 2023 18:46:03 GMT',
 'content-type': 'application/json', 'content-length': '254', 'connection':
 'keep-alive', 'x-amzn-requestid': '27259785-3562-48a0-808e-49dc0f1bbbc2'},
 'RetryAttempts': 0}, 'datastoreId': '12345678901234567890123456789012',
 'imageSetId': 'ea92b0d8838c72a3f25d00d13616f87e', 'latestVersionId': '3',
 'imageSetState': 'LOCKED', 'imageSetWorkflowStatus': 'UPDATING', 'createdAt':
 datetime.datetime(2023, 3, 28, 11, 12, 6, 436000, tzinfo=tzlocal()), 'updatedAt':
 datetime.datetime(2023, 3, 28, 11, 46, 3, 325000, tzinfo=tzlocal())}
```

Java

次のコード例では AWS SDK for Java 2.x 、 を使用して画像セットのメタデータを更新します。

```
final String updateAttributes = "{\"SchemaVersion\":1.1,\"Patient\":{\"DICOM\":
{\"PatientName\":\"MX^MX\"}}}";
final String removeAttributes = "{\"SchemaVersion\":1.1,\"Patient\":{\"DICOM\":
{\"PatientSex\":\"X\"}}}";
final UpdateImageSetMetadataRequest request =
 UpdateImageSetMetadataRequest.builder()
                .datastoreId(12345678901234567890123456789012)
                .imageSetId(ea92b0d8838c72a3f25d00d13616f87e)
                .latestVersionId(2)
                .updateImageSetMetadataUpdates(MetadataUpdates.builder()
                        .dicomUpdates(DICOMUpdates.builder()
                                .updatableAttributes(SdkBytes.fromByteBuffer(

 ByteBuffer.wrap(updateAttributes.getBytes(StandardCharsets.UTF_8))))
                                .removableAttributes(SdkBytes.fromByteBuffer(

 ByteBuffer.wrap(removeAttributes.getBytes(StandardCharsets.UTF_8))))
                                .build())
                        .build())
```

```
                .build();
```

このアクションは次の出力を返します。

```
UpdateImageSetMetadataResponse(DatastoreId=12345678901234567890123456789012,
 ImageSetId=ea92b0d8838c72a3f25d00d13616f87e, LatestVersionId=2,
 ImageSetState=LOCKED, ImageSetWorkflowStatus=UPDATING,
 CreatedAt=2023-03-09T05:07:30.057Z, UpdatedAt=2023-03-16T01:24:22.606Z)
```

画像セットのコピー

CopyImageSet アクションを使用して、 で画像セットをコピーします HealthImaging。この非同期プロセスを使用して、画像セットの内容を新規または既存の画像セットにコピーします。新しいイメージにコピーして画像セットを分割したり、別のコピーを作成したりできます。既存の画像セットにコピーして 2 つの画像セットを結合することもできます。

> **Note**
>
> 画像セットをコピーすると、画像セットの履歴に新しいバージョンが作成されます。詳細については、「画像セットのバージョンを一覧表示する」を参照してください。
> 画像セットのコピーは非同期プロセスです。そのため、状態 (imageSetState) フィールドと ステータス (imageSetStatus) フィールドを使用して、ロックされた画像セットに対してどのような操作が行われているかを知ることができます。ロックされた画像セットには他の書き込み操作は実行できません。

実際に DICOM をインポートすると、DICOM シリーズごとに複数の画像セットが作成される可能性があります。CopyImageSet アクションを使用する際には、以下の点を考慮してください。

- ある画像セットから別の画像セットにインスタンスをコピーする
- コピーでは、両方のメタデータ ImageSets が一貫している必要があります

次のタブには、 AWS CLI および AWS SDKs。詳細については、AWS HealthImaging API リファレンスのCopyImageSet「」を参照してください。

画像セットをコピーするには

へのアクセス設定に基づいてタブを選択します HealthImaging。

CLI

次のコード例では、 AWS Command Line Interface （ AWS CLI) を使用して画像セットのコピーを作成します。

送信先なし:

```
aws medical-imaging copy-image-set \
  --datastore-id 12345678901234567890123456789012 \
  --source-image-set-id ea92b0d8838c72a3f25d00d13616f87e\
  --copy-image-set-information '{"sourceImageSet": {"latestVersionId": "1" } }'
```

このアクションは次の出力を返します。

```
{
    "destinationImageSetProperties": {
        "latestVersionId": "1",
        "imageSetWorkflowStatus": "COPYING",
        "updatedAt": 1680042357.432,
        "imageSetId": "b9a06fef182a5f992842f77f8e0868e5",
        "imageSetState": "LOCKED",
        "createdAt": 1680042357.432
    },
    "sourceImageSetProperties": {
        "latestVersionId": "5",
        "imageSetWorkflowStatus": "COPYING_WITH_READ_ONLY_ACCESS",
        "updatedAt": 1680042357.432,
        "imageSetState": "LOCKED",
        "createdAt": 1680027126.436
    },
    "datastoreId": "12345678901234567890123456789012"
}
```

送信先あり:

```
aws medical-imaging copy-image-set \
  --datastore-id 12345678901234567890123456789012 \
  --source-image-set-id ea92b0d8838c72a3f25d00d13616f87e\
  --copy-image-set-information '{"sourceImageSet": {"latestVersionId": "5" },
 "destinationImageSet": { "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
 "latestVersionId": "1"} }'
```

このアクションは次の出力を返します。

```
{
    "destinationImageSetProperties": {
        "latestVersionId": "2",
        "imageSetWorkflowStatus": "COPYING",
        "updatedAt": 1680042505.135,
        "imageSetId": "b9a06fef182a5f992842f77f8e0868e5",
        "imageSetState": "LOCKED",
        "createdAt": 1680042357.432
    },
    "sourceImageSetProperties": {
        "latestVersionId": "5",
        "imageSetWorkflowStatus": "COPYING_WITH_READ_ONLY_ACCESS",
        "updatedAt": 1680042505.135,
        "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
        "imageSetState": "LOCKED",
        "createdAt": 1680027126.436
    },
    "datastoreId": "12345678901234567890123456789012"
}
```

JavaScript

次のコード例では AWS SDK for JavaScript 、 を使用して画像セットの複製コピーを作成します。

送信先なし:

```
var params = {
  datastoreId: '12345678901234567890123456789012', /* required */
  sourceImageSetId: 'ea92b0d8838c72a3f25d00d13616f87e', /* required */
  copyImageSetInformation: {
    sourceImageSet: { latestVersionId: "1" }
  } /* required */
};

var medicalImaging = new AWS.MedicalImaging();
medicalImaging.copyImageSet(params, function(err,data) {
     if (err)
        console.log(err, err.stack); // an error occurred
     else
```

```
        console.log(data); // successful response
});
```

このアクションは次の出力を返します。

```
{
    "destinationImageSetProperties": {
    "latestVersionId": "1",
        "imageSetWorkflowStatus": "COPYING",
        "updatedAt": 1680042357.432,
        "imageSetId": "b9a06fef182a5f992842f77f8e0868e5",
        "imageSetState": "LOCKED",
        "createdAt": 1680042357.432
    },
    "sourceImageSetProperties": {
    "latestVersionId": "5",
        "imageSetWorkflowStatus": "COPYING_WITH_READ_ONLY_ACCESS",
        "updatedAt": 1680042357.432,
        "imageSetState": "LOCKED",
        "createdAt": 1680027126.436
    },
    "datastoreId": "12345678901234567890123456789012"
}
```

送信先あり:

```
var params = {
    datastoreId: '12345678901234567890123456789012', /* required */
    sourceImageSetId: 'ea92b0d8838c72a3f25d00d13616f87e', /* required */
    copyImageSetInformation: {
        sourceImageSet: { latestVersionId: '5' },
        destinationImageSet: {
            imageSetId: '92b0d8838c72a3f25d00d13616f87e',
             latestVersionId: '1'
        }
    } /* required */
};

var medicalImaging = new AWS.MedicalImaging();
medicalImaging.copyImageSet(params, function(err,data) {
     if (err)
        console.log(err, err.stack); // an error occurred
     else
```

```
        console.log(data); // successful response
});
```

このアクションは次の出力を返します。

```
{
    "destinationImageSetProperties": {
    "latestVersionId": "2",
        "imageSetWorkflowStatus": "COPYING",
        "updatedAt": 1680042505.135,
        "imageSetId": "b9a06fef182a5f992842f77f8e0868e5",
        "imageSetState": "LOCKED",
        "createdAt": 1680042357.432
    },
    "sourceImageSetProperties": {
    "latestVersionId": "5",
        "imageSetWorkflowStatus": "COPYING_WITH_READ_ONLY_ACCESS",
        "updatedAt": 1680042505.135,
        "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
        "imageSetState": "LOCKED",
        "createdAt": 1680027126.436
    },
    "datastoreId": "12345678901234567890123456789012"
}
```

Python

次のコード例では AWS SDK for Python (Boto3) 、 を使用して画像セットの複製コピーを作成します。

送信先なし:

```
import boto3
hi_client= boto3.client('medical-imaging')
hi_client.copy_image_set(datastoreId = "12345678901234567890123456789012",
 sourceImageSetId = "ea92b0d8838c72a3f25d00d13616f87e", copyImageSetInformation=
 { "sourceImageSet": {  "latestVersionId": "1" } })
```

このアクションは次の出力を返します。

```
{'ResponseMetadata': {'RequestId': '5e45d0cc-b971-40a2-91d1-3b0dd62869bb',
 'HTTPStatusCode': 200, 'HTTPHeaders': {'date': 'Tue, 28 Mar 2023 18:12:06 GMT',
 'content-type': 'application/json', 'content-length': '511', 'connection':
 'keep-alive', 'x-amzn-requestid': '5e45d0cc-b971-40a2-91d1-3b0dd62869bb'},
 'RetryAttempts': 0}, 'datastoreId': '12345678901234567890123456789012',
 'sourceImageSetProperties': {'imageSetId': '18f88ac7870584f58d56256646b4d92b',
 'latestVersionId': '1', 'imageSetState': 'LOCKED', 'imageSetWorkflowStatus':
 'COPYING_WITH_READ_ONLY_ACCESS', 'createdAt': datetime.datetime(2023, 3, 23,
 10, 28, 30, 753000, tzinfo=tzlocal()), 'updatedAt': datetime.datetime(2023,
 3, 28, 11, 12, 6, 436000, tzinfo=tzlocal())}, 'destinationImageSetProperties':
 {'imageSetId': 'ea92b0d8838c72a3f25d00d13616f87e', 'latestVersionId': '1',
 'imageSetState': 'LOCKED', 'imageSetWorkflowStatus': 'COPYING', 'createdAt':
 datetime.datetime(2023, 3, 28, 11, 12, 6, 436000, tzinfo=tzlocal()),
 'updatedAt': datetime.datetime(2023, 3, 28, 11, 12, 6, 436000,
 tzinfo=tzlocal())}}
```

送信先あり:

```
import boto3
hi_client= boto3.client('medical-imaging')
hi_client.copy_image_set(datastoreId = "12345678901234567890123456789012",
 sourceImageSetId = "ea92b0d8838c72a3f25d00d13616f87e", copyImageSetInformation=
 { "sourceImageSet": {  "latestVersionId": "1" }, "destinationImageSet":
 { "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e", "latestVersionId": "1"}})
```

このアクションは次の出力を返します。

```
{'ResponseMetadata': {'RequestId': '9ac647fe-7a65-41c8-97a6-3823f961a555',
 'HTTPStatusCode': 200, 'HTTPHeaders': {'date': 'Tue, 28 Mar 2023 18:17:36 GMT',
 'content-type': 'application/json', 'content-length': '511', 'connection':
 'keep-alive', 'x-amzn-requestid': '9ac647fe-7a65-41c8-97a6-3823f961a555'},
 'RetryAttempts': 0}, 'datastoreId': '12345678901234567890123456789012',
 'sourceImageSetProperties': {'imageSetId': '8748350b8719b51639e151867306353a',
 'latestVersionId': '1', 'imageSetState': 'LOCKED', 'imageSetWorkflowStatus':
 'COPYING_WITH_READ_ONLY_ACCESS', 'createdAt': datetime.datetime(2023, 3, 23,
 10, 28, 30, 676000, tzinfo=tzlocal()), 'updatedAt': datetime.datetime(2023, 3,
 28, 11, 17, 35, 944000, tzinfo=tzlocal())}, 'destinationImageSetProperties':
 {'imageSetId': 'ea92b0d8838c72a3f25d00d13616f87e', 'latestVersionId': '2',
 'imageSetState': 'LOCKED', 'imageSetWorkflowStatus': 'COPYING', 'createdAt':
 datetime.datetime(2023, 3, 28, 11, 12, 6, 436000, tzinfo=tzlocal()),
 'updatedAt': datetime.datetime(2023, 3, 28, 11, 17, 35, 944000,
 tzinfo=tzlocal())}}
```

Java

次のコード例では AWS SDK for Java 2.x 、 を使用して画像セットの複製コピーを作成します。を使用してそれらをマージします。

送信先なし:

```
final CopyImageSetRequest copyImageSetRequest = CopyImageSetRequest.builder()
            .datastoreId(this.datastoreId)
            .sourceImageSetId(this.sourceImageSetId)
            .copyImageSetInformation(CopyImageSetInformation.builder()
                .sourceImageSet(CopySourceImageSetInformation.builder()
                    .latestVersionId(this.sourceImageSetLatestVersion)
                    .build())
            .build();
```

このアクションは次の出力を返します。

```
CopyImageSetResponse(DatastoreId=1234567890123456789012345678901234,
 SourceImageSetProperties=CopySourceImageSetProperties
 (ImageSetId=532db456704ebc636036ff8e8c53a0ef, LatestVersionId=1,
 ImageSetState=LOCKED, ImageSetWorkflowStatus=COPYING_WITH_READ_ONLY_ACCESS,
 CreatedAt=2023-03-06T13:47:29.268Z, UpdatedAt=2023-03-15T23:45:13.260Z),
 DestinationImageSetProperties=CopyDestinationImageSetProperties
 (ImageSetId=22c41f83a35ee6be68d22f6ba337accd, LatestVersionId=1,
 ImageSetState=LOCKED, ImageSetWorkflowStatus=COPYING,
 CreatedAt=2023-03-15T23:45:13.260Z, UpdatedAt=2023-03-15T23:45:13.260Z))
```

送信先あり:

```
final CopyImageSetRequest copyImageSetRequest = CopyImageSetRequest.builder()
            .datastoreId(this.datastoreId)
            .sourceImageSetId(this.sourceImageSetId)
            .copyImageSetInformation(CopyImageSetInformation.builder()
                .sourceImageSet(CopySourceImageSetInformation.builder()
                    .latestVersionId(this.sourceImageSetLatestVersion)
                    .build())
                .destinationImageSet(CopyDestinationImageSet.builder()
                    .imageSetId(this.destinationImageSetId)
                    .latestVersionId(getImageSetLatestVersion(this.datastoreId,
 this.destinationImageSetId)).build())
                .build())
```

```
            .build();
```

このアクションは次の出力を返します。

```
CopyImageSetResponse(DatastoreId=12345678901234567890123456789012,
 SourceImageSetProperties=CopySourceImageSetProperties
 (ImageSetId=532db456704ebc636036ff8e8c53a0ef, LatestVersionId=1,
 ImageSetState=LOCKED, ImageSetWorkflowStatus=COPYING_WITH_READ_ONLY_ACCESS,
 CreatedAt=2023-03-06T13:47:29.268Z, UpdatedAt=2023-03-15T23:45:13.260Z),
 DestinationImageSetProperties=CopyDestinationImageSetProperties
 (ImageSetId=22c41f83a35ee6be68d22f6ba337accd, LatestVersionId=1,
 ImageSetState=LOCKED, ImageSetWorkflowStatus=COPYING,
 CreatedAt=2023-03-15T23:45:13.260Z, UpdatedAt=2023-03-15T23:45:13.260Z))
```

画像セットの削除

`DeleteImageSet` アクションを使用して、 で画像セットを削除します HealthImaging。以下のタブには、 AWS Management Console および SDK の AWS CLI および コード例の手順が表示されます。 AWS SDKs 詳細については、AWS HealthImaging API リファレンスの`DeleteImageSet`「」を参照してください。

画像セットを削除するには

へのアクセス設定に基づいてタブを選択します HealthImaging。

Console

次の手順では AWS Management Console 、 を使用して画像セットを削除します。

1. HealthImaging コンソールのデータストアページを開きます。
2. データストアを選択します。

   データストアの詳細ページが開き、デフォルトで [画像セット] タブが選択されます。
3. 画像セットを選択し、削除 を選択します。

   画像セットを削除 モーダルが開きます。
4. 画像セットの ID を入力し、画像セットを削除を選択します。

CLI

次のコード例では、 AWS Command Line Interface （ AWS CLI) を使用して画像セットを削除します。

```
aws medical-imaging delete-image-set \
  --datastore-id 12345678901234567890123456789012 \
  --image-set-id ea92b0d8838c72a3f25d00d13616f87e
```

このアクションは次の出力を返します。

```
{
    "imageSetWorkflowStatus": "DELETING",
    "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
    "imageSetState": "LOCKED",
    "datastoreId": "12345678901234567890123456789012"
}
```

JavaScript

次のコード例では AWS SDK for JavaScript 、 を使用して画像セットを削除します。

```
var params = {
  datastoreId: '12345678901234567890123456789012', /* required */
  imageSetId: 'ea92b0d8838c72a3f25d00d13616f87e' /* required */
};

var medicalImagingRuntimeOperations = new AWS.MedicalImagingRuntimeOperations();
medicalImagingRuntimeOperations.deleteImageSet(params, function(err,data) {
     if (err)
        console.log(err, err.stack); // an error occurred
     else
        console.log(data); // successful response
});
```

このアクションは次の出力を返します。

```
{
    "imageSetWorkflowStatus": "DELETING",
    "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
    "imageSetState": "LOCKED",
    "datastoreId": "12345678901234567890123456789012"
```

```
}
```

Python

次のコード例では AWS SDK for Python (Boto3) 、 を使用して画像セットを削除します。

```
import boto3
hi_client = boto3.client('medical-imaging')
hi_client.delete_image_set(datastoreId = "12345678901234567890123456789012",
imageSetId = "ea92b0d8838c72a3f25d00d13616f87e")
```

このアクションは次の出力を返します。

```
{'ResponseMetadata': {'RequestId': '27259785-3562-48a0-808e-49dc0f1bbbc2',
'HTTPStatusCode': 200, 'HTTPHeaders': {'date': 'Tue, 28 Mar 2023 18:46:03 GMT',
'content-type': 'application/json', 'content-length': '254', 'connection':
'keep-alive', 'x-amzn-requestid': '27259785-3562-48a0-808e-49dc0f1bbbc2'},
'RetryAttempts': 0}, 'datastoreId': '12345678901234567890123456789012',
 'imageSetId':
'ea92b0d8838c72a3f25d00d13616f87e', 'imageSetState': 'LOCKED',
'imageSetWorkflowStatus': 'DELETING'}
```

Java

次のコード例では AWS SDK for Java 2.x 、 を使用して画像セットを削除します。

```
final DeleteImageSetRequest request = DeleteImageSetRequest.builder()
    .datastoreId(12345678901234567890123456789012)
    .imageSetId(ea92b0d8838c72a3f25d00d13616f87e)
```

このアクションは次の出力を返します。

```
DeleteImageSetResponse(DatastoreId=12345678901234567890123456789012,
ImageSetId=ea92b0d8838c72a3f25d00d13616f87e, ImageSetState=LOCKED,
ImageSetWorkflowStatus=DELETING)
```

AWS によるリソースのタグ付け HealthImaging

タグ形式で AWS HealthImaging リソースにメタデータを割り当てることができます。各タグは、ユーザー定義のキーと値で構成されるラベルです。タグは、リソースの管理、識別、整理、検索、フィルタリングに役立ちます。

> **Important**
>
> 保健情報 (PHI)、個人を特定できる情報 (PII)などの機密情報や秘匿性の高い情報はタグに格納しないでください。タグを使用して、課金および管理サービスを提供します。タグは、プライベートデータや機密データに使用することを意図していません。

以下のトピックでは、 AWS Management Console、 AWS CLI、および AWS SDKs を使用して、データストアと画像セットで HealthImaging タグ付けオペレーションを使用する方法について説明します。詳細については、「 AWS 全般のリファレンス ガイド」の「 AWS リソースのタグ付け」を参照してください。

トピック

- データストアのタグ作成
- 画像セットのタグ付け

データストアのタグ作成

AWS HealthImaging のデータストアを管理しやすくするには、`TagResource`、`ListTagsForResource`、`UntagResource`アクションを使用します。以下のコード例は、AWS Management Console、AWS CLI、AWS の SDK を使用して、データストアで HealthImaging タグ付けアクションを使用する方法を示しています。詳細については、 AWS 全般のリファレンスユーザーガイド のリソースへのタグ付けAWSを参照してください。

> **Note**
>
> HealthImaging データストアの作成時にタグを付けることもできます。詳細については、「データストアの作成」を参照してください。

データストアにタグを付けたり、タグを一覧表示したり、データストアからタグを解除したりするには

HealthImaging のアクセス設定に基づいてタブを選択します。

Console

以下の手順では、AWS Management Consoleを使用してデータストアのタグ付け、タグの一覧表示、タグ解除を行います。

データストアにタグ付けするには

1. HealthImaging コンソールのデータストアページを開きます。
2. データストアを選択します。

   データストアの詳細ページが開きます。
3. 詳細 タブを選択します。
4. タグ セクションで、[タグの管理] を選択します。

   タグの管理 ページが開きます。
5. [新しいタグを追加] をクリックします。
6. キー を入力し、オプションで 値 を入力します。
7. [変更の保存] をクリックします。

データストアのタグを一覧表示するには

1. HealthImaging コンソールのデータストアページを開きます。
2. データストアを選択します。

   データストアの詳細ページが開きます。
3. 詳細 タブを選択します。

   タグセクションに、すべてのデータストアタグが一覧表示されます。

データストアのタグを解除するには

1. HealthImaging コンソールのデータストアページを開きます。
2. データストアを選択します。

データストアの詳細ページが開きます。

3. 詳細 タブを選択します。
4. タグ セクションで、[タグの管理] を選択します。

   タグの管理 ページが開きます。
5. 削除するタグの横にある 削除 を選択します。
6. 変更の保存 をクリックします。

CLI

次のコード例では、AWS Command Line Interface (AWS CLI) を使用してデータストアのタグ付け、タグの一覧表示、タグ解除を行います。

データストアにタグ付けするには

```
aws medical-imaging tag-resource \
  --resource-arn "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012" \
  --tags '{"Deployment":"Development"}'
```

このアクションは次の出力を返します。

```
{}
```

データストアのタグを一覧表示するには

```
aws medical-imaging list-tags-for-resource \
  --resource-arn "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012"
```

このアクションは次の出力を返します。

```
{"tags":{"Deployment":"Development"}}
```

データストアのタグを解除するには

```
aws medical-imaging untag-resource \
```

```
  --resource-arn "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012" \
  --tag-keys '["Deployment"]'
```

このアクションは次の出力を返します。

```
{}
```

JavaScript

次のコード例では、AWS SDK for JavaScriptを使用してデータストアのタグ付け、タグの一覧表示、タグ解除を行います。

データストアにタグ付けするには

```
import AWS from 'aws-sdk'

const params = {
    region:'us-east-1'
}
const medicalImaging = new AWS.MedicalImaging(params)

const main = async () => {
    try {
        const result = await medicalImaging.tagResource({
            resourceArn: 'arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012',
            tags: {
                "Deployment" : "Development"
            }
        }).promise()
        console.log(result)
    }
    catch(err) {
        console.log(err)
    }
}

main()
```

このアクションは次の出力を返します。

```
{}
```

データストアのタグを一覧表示するには

```
import AWS from 'aws-sdk'

const params = {
    region:'us-east-1'
}
const medicalImaging = new AWS.MedicalImaging(params)

const main = async () => {
    try {
        const result = await medicalImaging.listTagsForResource({
            resourceArn: 'arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012'

        }).promise()
        console.log(result)
    }
    catch(err) {
        console.log(err)
    }
}

main()
```

このアクションは次の出力を返します。

```
{ tags: { Deployment: 'Development' } }
```

データストアのタグを解除するには

```
import AWS from 'aws-sdk'

const params = {
    region:'us-east-1'
}
const medicalImaging = new AWS.MedicalImaging(params)

const main = async () => {
    try {
```

```
        const result = await medicalImaging.untagResource({
            resourceArn: 'arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012',
            tagKeys: ["Development"]

        }).promise()
        console.log(result)
    }
    catch(err) {
        console.log(err)
    }
}

main()
```

このアクションは次の出力を返します。

```
{}
```

Python

次のコード例では、AWS SDK for Python (Boto3)を使用してデータストアのタグ付け、タグの一覧表示、タグ解除を行います。

データストアにタグ付けするには

```
medical_imaging_client.tag_resource(
  resourceArn="arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012",
  tags={"Deployment":"Development"}
)
```

このアクションは次の出力を返します。

```
{}
```

データストアのタグを一覧表示するには

```
medical_imaging_client.list-tags-for-resource(
  resourceArn="arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012"
```

```
)
```

このアクションは次の出力を返します。

```
{"tags":{"Deployment":"Development"}}
```

データストアのタグを解除するには

```
medical_imaging_client.untag_resource(
  resourceArn="arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012",
  tagKeys=["Deployment"]
)
```

このアクションは次の出力を返します。

```
{}
```

Java

次のコード例では、AWS SDK for Java 2.xを使用してデータストアのタグ付け、タグの一覧表示、タグ解除を行います。

データストアにタグ付けするには

```
Map tags = new HashMapString, String{...};
TagResourceRequest request = TagResourceRequest.builder()
                .resourceArn("arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012")
                .tags("Deployment":"Development")
                .build();
TagResourceResponse response = client.tagResource(request);
```

このアクションは次の出力を返します。

```
TagResourceResponse object
```

データストアのタグを一覧表示するには

```
ListTagsForResourceRequest request = ListTagsForResourceRequest.builder()
```

```
                .resourceArn("arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012")
                .build();
ListTagsForResourceResponse response = client.listTagsForResource(request);
```

このアクションは次の出力を返します。

```
ListTagsForResourceResponse object
```

データストアのタグを解除するには

```
List tagKeys = new ArrayListString{...};
UntagResourceRequest request = UntagResourceRequest.builder()
                .resourceArn("arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012")
                .tagKeys("Deployment")
                .build();
UntagResourceResponse response = client.untagResource(request);
```

このアクションは次の出力を返します。

```
UntagResourceResponse object
```

画像セットのタグ付け

AWS HealthImaging で画像セットを管理しやすくするには TagResource、ListTagsForResource、および UntagResource アクションを使用します。以下のコード例は、AWS CLI および AWS SDK を使用して画像セットで HealthImaging タグ付けアクションを使用する方法を示しています。詳細については、「AWS 全般のリファレンス ガイド」の「AWS リソースへのタグ付け」を参照してください。

画像セットのタグ付け、タグの一覧表示、タグ解除を行うには

HealthImaging のアクセス設定に基づいてタブを選択します。

CLI

以下のコード例では、AWS Command Line Interface (AWS CLI) を使用して画像セットのタグ付け、タグの一覧表示、およびタグ解除を行います。

画像セットのタグ付けをするには

```
aws medical-imaging tag-resource \
  --resource-arn "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/18f88ac7870584f58d56256646b4d92b" \
  --tags '{"Deployment":"Development"}'
```

このアクションは次の出力を返します。

```
{}
```

画像セットのタグを一覧表示するには

```
aws medical-imaging list-tags-for-resource \
  --resource-arn "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/18f88ac7870584f58d56256646b4d92b"
```

このアクションは次の出力を返します。

```
{"tags":{"Deployment":"Development"}}
```

画像セットのタグを解除するには

```
aws medical-imaging untag-resource \
  --resource-arn "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/18f88ac7870584f58d56256646b4d92b" \
  --tag-keys '["Deployment"]'
```

このアクションは次の出力を返します。

```
{}
```

JavaScript

以下のコード例では、AWS SDK for JavaScript を使用して画像セットにタグを付け、タグを一覧表示し、タグを解除します。

画像セットのタグ付けをするには

```
import AWS from 'aws-sdk'

const params = {
    region:'us-east-1'
}
const medicalImaging = new AWS.MedicalImaging(params)

const main = async () => {
    try {
        const result = await medicalImaging.tagResource({
            resourceArn: 'arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/18f88ac7870584f58d56256646b4d92b',
            tags: {
                "Deployment" : "Development"
            }
        }).promise()
        console.log(result)
    }
    catch(err) {
        console.log(err)
    }
}

main()
```

このアクションは次の出力を返します。

```
{}
```

画像セットのタグを一覧表示するには

```
import AWS from 'aws-sdk'

const params = {
    region:'us-east-1'
}
const medicalImaging = new AWS.MedicalImaging(params)

const main = async () => {
    try {
```

```
        const result = await medicalImaging.listTagsForResource({
            resourceArn: 'arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/18f88ac7870584f58d56256646b4d92b'

        }).promise()
        console.log(result)
    }
    catch(err) {
        console.log(err)
    }
}

main()
```

このアクションは次の出力を返します。

```
{ tags: { Deployment: 'Development' } }
```

画像セットのタグを解除するには

```
import AWS from 'aws-sdk'

const params = {
    region:'us-east-1'
}
const medicalImaging = new AWS.MedicalImaging(params)

const main = async () => {
    try {
        const result = await medicalImaging.untagResource({
            resourceArn: 'arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/18f88ac7870584f58d56256646b4d92b',
            tagKeys: ["Development"]

        }).promise()
        console.log(result)
    }
    catch(err) {
        console.log(err)
    }
}
```

```
main()
```

このアクションは次の出力を返します。

```
{}
```

Python

以下のコード例では、AWS SDK for Python (Boto3) を使用して画像セットにタグを付け、タグを一覧表示し、タグを解除します。

画像セットのタグ付けをするには

```
medical_imaging_client.tag_resource(
  resourceArn="arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/18f88ac7870584f58d56256646b4d92b",
  tags={"Deployment":"Development"}
)
```

このアクションは次の出力を返します。

```
{}
```

画像セットのタグを一覧表示するには

```
medical_imaging_client.list-tags-for-resource(
  resourceArn="arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/18f88ac7870584f58d56256646b4d92b"
)
```

このアクションは次の出力を返します。

```
{"tags":{"Deployment":"Development"}}
```

画像セットのタグを解除するには

```
medical_imaging_client.untag_resource(
```

```
  resourceArn="arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/18f88ac7870584f58d56256646b4d92b",
  tagKeys=["Deployment"]
)
```

このアクションは次の出力を返します。

```
{}
```

Java

以下のコード例では、AWS SDK for Java 2.x を使用して画像セットにタグを付け、タグを一覧表示し、タグを解除します。

画像セットのタグ付けをするには

```
Map tags = new HashMapString, String{...};
TagResourceRequest request = TagResourceRequest.builder()
                .resourceArn("arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/18f88ac7870584f58d56256646b4d92b")
                .tags("Deployment":"Development")
                .build();
TagResourceResponse response = client.tagResource(request);
```

このアクションは次の出力を返します。

```
TagResourceResponse object
```

画像セットのタグを一覧表示するには

```
ListTagsForResourceRequest request = ListTagsForResourceRequest.builder()
                .resourceArn("arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/18f88ac7870584f58d56256646b4d92b")
                .build();
ListTagsForResourceResponse response = client.listTagsForResource(request);
```

このアクションは次の出力を返します。

```
ListTagsForResourceResponse object
```

画像セットのタグを解除するには

```
List tagKeys = new ArrayListString{...};
UntagResourceRequest request = UntagResourceRequest.builder()
                .resourceArn("arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/18f88ac7870584f58d56256646b4d92b")
                .tagKeys("Deployment")
                .build();
UntagResourceResponse response = client.untagResource(request);
```

このアクションは次の出力を返します。

```
UntagResourceResponse object
```

AWS HealthImaging のセキュリティ

AWS では、クラウドセキュリティを最優先事項としています。AWS のユーザーは、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を利用できます。

セキュリティは、AWS とユーザーの間の責任共有です。責任共有モデル では、これをクラウドのセキュリティおよびクラウド内のセキュリティとして説明しています。

- クラウドのセキュリティ - AWS は、AWS クラウド で AWS の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。また、AWS は、ユーザーが安全に使用できるサービスも提供します。AWSコンプライアンスプログラム[g11]AWSコンプライアンスプログラム[/g11] [g10]AWSコンプライアンスプログラム[/g10]の一環として、サードパーティーの監査が定期的にセキュリティの有効性をテストおよび検証しています。AWS HealthImaging に適用す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による対象範囲内の AWS のサービスをご参照ください。
- クラウド内のセキュリティ - ユーザーの責任は、使用する AWS のサービスに応じて異なります。またお客様は、データの機密性、企業要件、適用法令と規制などのその他の要因に対しても責任を担います。

このドキュメントは、HealthImaging を使用する際に責任共有モデルを適用する方法を理解するのに役立ちます。次のトピックでは、セキュリティとコンプライアンスの目的を達成するために HealthImaging を設定する方法を示します。また、HealthImaging リソースのモニタリングや保護に役立つ他の AWS のサービスの使用方法についても説明します。

トピック

- AWS でのデータ保護 HealthImaging
- AWS の Identity and Access Management HealthImaging
- AWS HealthImaging でのログ記録とモニタリング
- AWS HealthImaging のコンプライアンス検証
- AWS HealthImaging のレジリエンス
- AWS HealthImaging のインフラストラクチャセキュリティ
- AWS CloudFormation による AWS HealthImaging リソースの作成
- AWS HealthImaging とインターフェース VPC エンドポイント (AWS PrivateLink)

AWS でのデータ保護 HealthImaging

AWS 責任共有モデル、AWS でのデータ保護に適用されます HealthImaging。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を担います AWS クラウド。このインフラストラクチャでホストされているコンテンツに対する管理を維持する責任はユーザーにあります。このコンテンツには、使用される AWS のサービス のセキュリティ構成と管理タスクが含まれます。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のよくある質問」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。こうすると、それぞれのジョブを遂行するために必要なアクセス許可のみを各ユーザーに付与できます。また、以下の方法でデータを保護することをお勧めします。

- 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、できれば TLS 1.3 が必要です。
- で 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。
- AWS 暗号化ソリューションを、 内のすべてのデフォルトのセキュリティコントロールとともに使用します AWS のサービス。
- Amazon Macie などの高度なマネージド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 140-2 検証済みの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) 140-2」を参照してください。

お客様の E メールアドレスなどの機密情報やセンシティブ情報は、タグや名前フィールド などの自由形式のフィールドに配置しないことを強くお勧めします。これは、コンソール HealthImaging 、API、または SDK で AWS CLIまたは他の AWS のサービス を使用する場合も同様です。 AWS SDKs 名前に使用する自由記述のテキストフィールドやタグに入力したデータは、課金や診断ログに使用される場合があります。外部サーバーへの URL を提供する場合は、そのサーバーへのリクエストを検証するための認証情報を URL に含めないように強くお勧めします。

トピック

データ暗号化

AWS を使用すると HealthImaging、クラウド内の保管中のデータにセキュリティレイヤーを追加し、スケーラブルで効率的な暗号化機能を提供できます。具体的には次のとおりです。

- ほとんどの AWS サービスで利用できる保管中のデータの暗号化機能
- を含む柔軟なキー管理オプション。暗号化キー AWS を管理するか AWS Key Management Service、独自のキーを完全に制御するかを選択できます。
- AWS 所有の AWS KMS 暗号化キー
- Amazon SQS に対するサーバー側の暗号化 (SSE) を使用した、機密データを送信するための暗号化メッセージキュー

さらに、 は、暗号化とデータ保護を AWS 環境で開発またはデプロイする のサービスと統合するための APIs AWS を提供します。

保管中の暗号化

HealthImaging は、サービス所有の AWS KMS キーを使用して保管中の顧客の機密データを保護するための暗号化をデフォルトで提供します。

転送中の暗号化

HealthImaging は TLS 1.2 を使用して、パブリックエンドポイントおよびバックエンドサービスを介して転送中のデータを暗号化します。

キー管理

AWS KMS キー (KMS キー) は、 のプライマリリソースです AWS Key Management Service。の外部で使用するデータキーを生成することもできます AWS KMS。

AWS 所有の KMS キー

HealthImaging は、デフォルトでこれらのキーを使用して、保管中の個人を特定できるデータやプライベート健康情報 (PHI) データなど、機密性の高い情報を自動的に暗号化します。 AWS が所有す

る KMS キーは、 アカウントに保存されません。これらは、複数の AWS accounts. AWS services で使用するために が AWS 所有および管理する KMS キーのコレクションの一部です。 サービスは、AWS 所有の KMS キーを使用してデータを保護できます。 AWS 所有の KMS キーを表示、管理、使用したり、その使用を監査したりすることはできません。ただし、データを暗号化するキーを保護するための作業やプログラムを操作したり変更したりする必要はありません。

AWS 所有の KMS キーを使用する場合、月額料金や使用料金は請求されません。また、アカウントの AWS KMS クォータにも影響しません。詳細については、AWS Key Management Service デベロッパーガイド の「AWS 所有キー」を参照してください。

カスタマーマネージド KMS キー

HealthImaging は、ユーザーが作成、所有、管理する対称カスタマーマネージド KMS キーの使用をサポートし、既存の AWS 所有の暗号化に 2 番目の暗号化レイヤーを追加します。この暗号化レイヤーはユーザーが完全に制御できるため、次のようなタスクを実行できます。

- キーポリシー、IAM ポリシー、許可の確立と維持
- キー暗号化マテリアルのローテーション
- キーポリシーの有効化と無効化
- タグの追加
- キーエイリアスの作成
- 削除のためのキースケジューリング

CloudTrail を使用して、 が AWS KMS ユーザーに代わって HealthImaging に送信するリクエストを追跡することもできます。 AWS KMS 追加料金が適用されます。詳細については、AWS Key Management Service デベロッパーガイドの「 カスタマーマネージドキー 」を参照してください。

カスタマーマネージドキーの作成

対称カスタマーマネージドキーは、 AWS Management Console または AWS KMS APIsを使用して作成できます。詳細については、「AWS Key Management Service デベロッパーガイド」の「対称暗号化 KMS キーを作成する」を参照してください。

キーポリシーは、カスタマーマネージドキーへのアクセスを制御します。すべてのカスタマーマネージドキーには、キーポリシーが 1 つだけ必要です。このポリシーには、そのキーを使用できるユーザーとその使用方法を決定するステートメントが含まれています。カスタマーマネージドキーを作成する際に、キーポリシーを指定することができます。詳細については、AWS Key Management

Service デベロッパーガイド の「カスタマーマネージドキーへのアクセスの管理」を参照してください。

カスタマーマネージドキーを HealthImaging リソースで使用するには、kms:CreateGrant キーポリシーでオペレーションを許可する必要があります。これにより、指定された KMS キーへのアクセスを制御するカスタマーマネージドキーに許可が追加され、これにより、付与オペレーション HealthImaging に必要なアクセス権がユーザーに付与されます。詳細については、「AWS Key Management Service デベロッパーガイド」の「AWS KMSの許可」を参照してください。

カスタマーマネージド KMS キーを HealthImaging リソースで使用するには、キーポリシーで次の API オペレーションを許可する必要があります。

- kms:DescribeKey は、キーの検証に必要なカスタマーマネージドキーの詳細を提供します。これはすべてのオペレーションに必要です。
- kms:GenerateDataKey は、すべての書き込み操作で保存中の暗号化リソースにアクセスできるようにします。
- kms:Decrypt は暗号化されたリソースの読み取りまたは検索操作へのアクセスを提供します。
- kms:ReEncrypt* はリソースを再暗号化するためのアクセスを提供します。

以下は、ユーザーがそのキーで暗号化 HealthImaging されたデータストアを作成して操作できるようにするポリシーステートメントの例です。

```
{
    "Sid": "Allow access to create data stores and perform CRUD and search in
 HealthImaging",
    "Effect": "Allow",
    "Principal": {
        "Service": [
            "medical-imaging.amazonaws.com"
        ]
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey*"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "kms:EncryptionContext:kms-arn": "arn:aws:kms:us-east-1:123456789012:key/
bec71d48-3462-4cdd-9514-77a7226e001f",
```

```
            "kms:EncryptionContext:aws:medical-imaging:datastoreId": "datastoreId"
        }
    }
}
```

カスタマーマネージド KMS キーの使用に必要な IAM アクセス許可

カスタマーマネージド KMS キーを使用して AWS KMS 暗号化を有効にしてデータストアを作成する場合、 HealthImaging データストアを作成するユーザーまたはロールのキーポリシーと IAM ポリシーの両方に必要なアクセス許可があります。

キーポリシーの詳細については、「AWS Key Management Service デベロッパーガイド」の「IAM ポリシーの有効化」を参照してください。

リポジトリを作成する IAM ユーザー、IAM ロール、または AWS アカウントには、kms:CreateGrant、、kms:GenerateDataKey、kms:RetireGrantkms:Decryptおよび のアクセス許可に加えてkms:ReEncrypt*、AWS に必要なアクセス許可が必要です HealthImaging。

が で許可 HealthImaging を使用する方法 AWS KMS

HealthImaging には、カスタマーマネージド KMS キーを使用するための許可が必要です。カスタマーマネージド KMS キーで暗号化されたデータストアを作成すると、 は にCreateGrantリクエストを送信することで、ユーザーに代わって許可 HealthImaging を作成します AWS KMS。の許可 AWS KMS は、顧客アカウントの KMS キー HealthImaging へのアクセスを許可するために使用されます。

がユーザーに代わって HealthImaging 作成する許可は、取り消したり、廃止したりしないでください。アカウントでキーを使用する AWS KMS アクセス HealthImaging 許可を付与するグラントを取り消すか廃止した場合、このデータにアクセスしたり、データストアにプッシュされた新しいイメージリソースを暗号化したり、プル時に復号したり HealthImaging することはできません。の許可を取り消したり、廃止したりすると HealthImaging、変更はすぐに行われます。アクセス権限を取り消すには、許可を取り消すのではなく、データストアを削除します。データストアが削除されると、 はユーザーに代わって許可を HealthImaging 廃止にします。

HealthImaging の暗号化キーのモニタリング

を使用して CloudTrail 、カスタマーマネージド KMS キーを使用するときに、 AWS KMS がユーザーに代わって HealthImaging に送信するリクエストを追跡できます。 CloudTrail ログのログエン

トリは userAgentフィールドmedical-imaging.amazonaws.comに表示され、 によって行われたリクエストを明確に区別します HealthImaging。

次の例はCreateGrant、カスタマーマネージドキーによって暗号化されたデータにアクセス HealthImaging するために によって呼び出される AWS KMS オペレーションをモニタリングDescribeKeyするための GenerateDataKey、Decrypt、、および の CloudTrail イベントです。

以下は、 CreateGrantを使用して HealthImaging が顧客提供の KMS キーにアクセスできるようにする方法を示しています。これにより、 はその KMS キー HealthImaging を使用して、保管中のすべての顧客データを暗号化できるようになります。

ユーザーは、独自の grants を作成する必要はありません。 は、 にCreateGrantリクエストを送信することで、ユーザーに代わって grant HealthImaging を作成します AWS KMS。の許可 AWS KMS は、顧客アカウントの AWS KMS キー HealthImaging へのアクセスを許可するために使用されます。

```
{
    "Grants": [
        {
            "Operations": [
                "Decrypt",
                "Encrypt",
                "GenerateDataKey",
                "GenerateDataKeyWithoutPlaintext",
                "DescribeKey"
            ],
            "KeyId": "arn:aws:kms:us-west-2:824333766656:key/2fe3c119-792d-4b99-822f-
b5841e1181d1",
            "Name": "0a74e6ad2aa84b74a22fcd3efac1eaa8",
            "RetiringPrincipal": "AWS Internal",
            "GranteePrincipal": "AWS Internal",
            "GrantId":
 "0da169eb18ffd3da8c0eebc9e74b3839573eb87e1e0dce893bb544a34e8fbaaf",
            "IssuingAccount": "AWS Internal",
            "CreationDate": 1685050229.0,
            "Constraints": {
                "EncryptionContextSubset": {
                    "kms-arn": "arn:aws:kms:us-
west-2:824333766656:key/2fe3c119-792d-4b99-822f-b5841e1181d1"
                }
            }
```

```
        },
        {
            "Operations": [
                "GenerateDataKey",
                "CreateGrant",
                "RetireGrant",
                "DescribeKey"
            ],
            "KeyId": "arn:aws:kms:us-west-2:824333766656:key/2fe3c119-792d-4b99-822f-
b5841e1181d1",
            "Name": "2023-05-25T21:30:17",
            "RetiringPrincipal": "AWS Internal",
            "GranteePrincipal": "AWS Internal",
            "GrantId":
 "8229757abbb2019555ba64d200278cedac08e5a7147426536fcd1f4270040a31",
            "IssuingAccount": "AWS Internal",
            "CreationDate": 1685050217.0,
        }
    ]
}
```

以下の例は、GenerateDataKey を使用して、ユーザーがデータを暗号化するのに必要な許可を持っているか、データを保存する前に確認する方法を示しています。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EXAMPLEUSER",
        "arn": "arn:aws:sts::111122223333:assumed-role/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLEKEYID",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "EXAMPLEROLE",
                "arn": "arn:aws:iam::111122223333:role/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Sampleuser01"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2021-06-30T21:17:06Z",
```

```
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "medical-imaging.amazonaws.com"
    },
    "eventTime": "2021-06-30T21:17:37Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "medical-imaging.amazonaws.com",
    "userAgent": "medical-imaging.amazonaws.com",
    "requestParameters": {
        "keySpec": "AES_256",
        "keyId": "arn:aws:kms:us-east-1:111122223333:key/EXAMPLE_KEY_ARN"
    },
    "responseElements": null,
    "requestID": "EXAMPLE_ID_01",
    "eventID": "EXAMPLE_ID_02",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-east-1:111122223333:key/EXAMPLE_KEY_ARN"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

次の例は、 が Decryptオペレーションを HealthImaging 呼び出して、保存された暗号化データキーを使用して暗号化されたデータにアクセスする方法を示しています。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EXAMPLEUSER",
        "arn": "arn:aws:sts::111122223333:assumed-role/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLEKEYID",
```

```
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "EXAMPLEROLE",
                "arn": "arn:aws:iam::111122223333:role/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Sampleuser01"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2021-06-30T21:17:06Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "medical-imaging.amazonaws.com"
    },
    "eventTime": "2021-06-30T21:21:59Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "medical-imaging.amazonaws.com",
    "userAgent": "medical-imaging.amazonaws.com",
    "requestParameters": {
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "keyId": "arn:aws:kms:us-east-1:111122223333:key/EXAMPLE_KEY_ARN"
    },
    "responseElements": null,
    "requestID": "EXAMPLE_ID_01",
    "eventID": "EXAMPLE_ID_02",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-east-1:111122223333:key/EXAMPLE_KEY_ARN"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

次の例は、 が DescribeKeyオペレーション HealthImaging を使用して、 AWS KMS 顧客所有 AWS KMS キーが使用可能な状態であるかどうかを確認し、ユーザーが機能していない場合のトラブルシューティングを支援する方法を示しています。


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EXAMPLEUSER",
        "arn": "arn:aws:sts::111122223333:assumed-role/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLEKEYID",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "EXAMPLEROLE",
                "arn": "arn:aws:iam::111122223333:role/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Sampleuser01"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2021-07-01T18:36:14Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "medical-imaging.amazonaws.com"
    },
    "eventTime": "2021-07-01T18:36:36Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DescribeKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "medical-imaging.amazonaws.com",
    "userAgent": "medical-imaging.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-east-1:111122223333:key/EXAMPLE_KEY_ARN"
    },
    "responseElements": null,
    "requestID": "EXAMPLE_ID_01",
    "eventID": "EXAMPLE_ID_02",
    "readOnly": true,
    "resources": [
        {
```

```
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-east-1:111122223333:key/EXAMPLE_KEY_ARN"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

詳細はこちら

以下のリソースは、保管中のデータの暗号化に関する詳細を説明しており、「AWS Key Management Service デベロッパーガイド」にあります。

- AWS KMS の概念
- のセキュリティのベストプラクティス AWS KMS

ネットワークトラフィックのプライバシー

トラフィックは、 HealthImaging とオンプレミスアプリケーション間、および と Amazon S3 間 HealthImagingの両方で保護されます。 HealthImaging と 間のトラフィックは、デフォルトで HTTPS AWS Key Management Service を使用します。

- AWS HealthImaging は、米国東部 (バージニア北部) 、米国西部 (オレゴン) 、欧州 (アイルランド) 、アジアパシフィック (シドニー) の各リージョンで利用できるリージョンサービスです。
- HealthImaging と Amazon S3 バケット間のトラフィックの場合、Transport Layer Security (TLS) は、 HealthImaging と Amazon S3 間、および HealthImaging とそれにアクセスするカスタマーアプリケーション間で転送中のオブジェクトを暗号化します。Amazon S3 バケット IAM ポリシー`aws:SecureTransport condition`で を使用して、HTTPS (TLS) 経由の暗号化された接続のみを許可する必要があります。 HealthImaging 現在、 はパブリックエンドポイントを使用して Amazon S3 バケットのデータにアクセスしますが、データがパブリックインターネットを通過するわけではありません。 HealthImaging と Amazon S3 間のすべてのトラフィックは AWS ネットワーク経由でルーティングされ、TLS を使用して暗号化されます。

AWS の Identity and Access Management HealthImaging

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 HealthImaging リソースの使用を承認する (アクセス許可を付与する) かを制御します。IAM は、追加料金なしで AWS のサービス 使用できる です。

トピック

対象者

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

サービスユーザー – HealthImaging サービスを使用してジョブを実行する場合は、管理者から必要な認証情報とアクセス許可が与えられます。さらに多くの HealthImaging 機能を使用して作業を行う場合は、追加のアクセス許可が必要になることがあります。アクセスの管理方法を理解しておくと、管理者に適切な許可をリクエストするうえで役立ちます。 HealthImaging 機能にアクセスできない場合は、「AWS HealthImaging アイデンティティとアクセスの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 HealthImaging リソースを担当している場合は、通常、 へのフルアクセスがあります HealthImaging。サービスのユーザーがどの HealthImaging 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。お客様の会社で で IAM を利用する方法の詳細については HealthImaging、「」を参照してくださいAWS と IAM の HealthImaging 連携方法。

IAM 管理者 - 管理者は、 HealthImaging へのアクセスを管理するポリシーの書き込み方法の詳細について確認する場合があります。IAM で使用できる HealthImaging アイデンティティベースのポリ

シーの例を表示するには、「」を参照してくださいAWS のアイデンティティベースのポリシーの例 HealthImaging。

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 (AWS IAM Identity Center) から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。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 の認証情報は、フェデレーティッド ID の例です。フェデレーティッドアイデンティティ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。フェデレーションを使用して にアクセスすると、間接的 AWS にロールを引き受けます。

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「 にサインインする方法 AWS アカウント」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 Software Development Kit (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に暗号で署名します。 AWS ツールを使用しない場合は、リクエストを自分で署名する必要があります。推奨される方法を使用してリクエストを自分で署名する方法の詳細については、「IAM ユーザーガイド」のAWS 「 API リクエストの署名」を参照してください。

使用する認証方法を問わず、セキュリティ情報の提供を追加でリクエストされる場合もあります。例えば、 では、多要素認証 (MFA) を使用してアカウントのセキュリティを高めることを AWS 推奨しています。詳細については、「AWS IAM Identity Center ユーザーガイド」の「多要素認証 (MFA)」および「IAM ユーザーガイド」の「AWSでの多要素認証 (MFA) の使用」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 AWS のサービス およびリソースへの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。このアイデンティティは 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を保護し、それらを使用してルートユーザーのみが実行できるタスクを実行してください。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、一時的な AWS のサービス 認証情報を使用して にアクセスする ID プロバイダーとのフェデレーションの使用を要求します。

フェデレーティッド ID とは、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、、Identity Center ディレクトリのユーザー、または ID ソースから提供された認証情報 AWS のサービス を使用して にアクセスするすべてのユーザーです。フェデレーティッド ID が にアクセスすると 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ことも、すべての とアプリケーションで使用する独自の ID ソースのユーザー AWS アカウント とグループのセットに接続して同期することもできます。IAM アイデンティティセンターの詳細については、[AWS IAM Identity Center ユーザーガイド」の「IAM アイデンティティセンターとは」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 内のアイデンティティです。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アクセス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、ロールでは一時的な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 (ロールではなく) の作成が適している場合」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 内のアイデンティティです。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。ロールを切り替える AWS Management Console ことで、 で IAM ロールを一時的に引き受けることができます。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「IAM ロールの使用」を参照してください。

一時的な認証情報を持った IAM ロールは、以下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アイデンティティにアクセス許可を割り当てるには、ロールを作成してそのロールのアクセス許可を定義します。フェデレーティッドアイデンティティが認証されると、そのアイデンティティはロールに関連付けられ、ロールで定義されている権限が付与されます。フェデレーションの詳細については、「IAM ユーザーガイド」の「サードパーティーアイデンティティプロバイダー向けロールの作成」を参照してください。IAM Identity Center を使用する場合、アクセス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 アイデンティティセンターは、アクセス許可セットを 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 ユーザーガイド」の「アクセス許可セット」を参照してください。
- 一時的な IAM ユーザーアクセス許可 - 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アクセス許可を一時的に 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、 (ロールをプロキシとして使用する代わりに) リソースにポリシーを直接アタッチできます。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 ロールとリソースベースのポリシーとの相違点」を参照してください。
- クロスサービスアクセス — 一部の では、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権限、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。

- プリンシパル許可 – 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場合 AWS、プリンシパルと見なされます。ポリシーによって、プリンシパルに許可が付与されます。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により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。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。アクションがポリシーで追加の依存アクションを必要とするかどうかを確認するには、「サービス認証リファレンス」のAWS 「Awesome 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。
- 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 のサービスに権限を委任するロールの作成」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。

- 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 リクエストを作成している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。AWS ロールを EC2 インスタンスに割り当て、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 ロールを使用してアクセス許可を付与する」を参照してください。

IAM ロールと IAM ユーザーのどちらを使用するかについては、「IAM ユーザーガイド」の「IAM ロールの作成が適している場合 (ユーザーではなく)」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、 AWS ID またはリソースにアタッチします。ポリシーは のオブジェクト AWS であり、ID またはリソースに関連付けて、これらのアクセス許可を定義します。 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、これらのポリシー AWS を評価します。ポリシーでのアクセス許可により、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 AWS JSON 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アクセス許可をユーザーに付与するため、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。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションのアクセス許可を定義します。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。こ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、ユーザーグループ、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、および条件を制御します。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 ポリシーの作成」を参照してください。

アイデンティティベースポリシーは、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、 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。管理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比較」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挙げられます。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。ポリシーが添付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 IAM の AWS マネージドポリシーを使用することはできません。

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 と 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、追加の一般的でない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る最大の許可を設定できます。

- 権限の境界 - 権限の境界は、アイデンティティベースのポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる許可の上限を設定する高度な機能です。エンティティに権限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権限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、権限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) に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、 AWS アカウント 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。組織と SCP の詳細については、「AWS Organizations ユーザーガイド」の「SCP の仕組み」を参照してください。
- 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の範囲は、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。複数のポリシータイプが関連する場合に が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「ポリシーの評価ロジック」を参照してください。

AWS と IAM の HealthImaging 連携方法

IAM を使用して へのアクセスを管理する前に HealthImaging、 で使用できる IAM の機能について学びます HealthImaging。

AWS で使用できる IAM の機能 HealthImaging

IAM 機能	HealthImaging サポート
アイデンティティベースのポリシー	あり
リソースベースのポリシー	いいえ
ポリシーアクション	あり
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	なし
ABAC (ポリシー内のタグ)	部分的
一時的な認証情報	あり
プリンシパル権限	あり
サービスロール	あり

IAM 機能	HealthImaging サポート
サービスリンクロール	いいえ

HealthImaging 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を把握するには、「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。

のアイデンティティベースのポリシー HealthImaging

アイデンティティベースポリシーをサポートする	あり

アイデンティティベースポリシーは、IAM ユーザー、ユーザーグループ、ロールなど、アイデンティティに添付できる JSON アクセス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、および条件を制御します。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 ポリシーの作成」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。

のアイデンティティベースのポリシーの例 HealthImaging

HealthImaging アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいAWS のアイデンティティベースのポリシーの例 HealthImaging。

内のリソースベースのポリシー HealthImaging

リソースベースのポリシーのサポート	なし

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挙げ

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。ポリシーが添付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 にある場合 AWS アカウント、信頼されたアカウントの IAM 管理者は、リソースへのアクセス許可をプリンシパルエンティティ (ユーザーまたはロール) に付与する必要もあります。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添付することで許可を付与します。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追加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 ロールとリソースベースのポリシーとの相違点」を参照してください。

のポリシーアクション HealthImaging

ポリシーアクションに対するサポート	はい

管理者は AWS JSON 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んなリソースにどんな条件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、関連する 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アクセス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。

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

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 HealthImaging を使用します。

```
AWS
```

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": [
      "AWS:action1",
      "AWS:action2"
         ]
```

HealthImaging アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいAWS のアイデンティティベースのポリシーの例 HealthImaging。

のポリシーリソース HealthImaging

ポリシーリソースに対するサポート	はい

管理者は AWS JSON 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 ポリシーの要素は、オブジェクトあるい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は、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、リソースレベルのアクセス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、リソースレベルのアクセス許可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": "*"
```

HealthImaging リソースタイプとその ARNs」の「AWS で定義されるリソースタイプ HealthImaging」を参照してください。 ARN を使用できるアクションとリソースについては、「AWS で定義されるアクション HealthImaging」を参照してください。

HealthImaging アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいAWS のアイデンティティベースのポリシーの例 HealthImaging。

のポリシー条件キー HealthImaging

サービス固有のポリシー条件キーのサポート	はい

管理者は AWS JSON 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んなリソースにどんな条件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。equal や less than などの条件演算子を使用して条件式を作成することによって、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 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演算を使用して条件 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許可を付与できます。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。

HealthImaging 条件キーのリストを確認するには、「サービス認証リファレンス」の「AWS の条件キー HealthImaging」を参照してください。条件キーを使用できるアクションとリソースについては、「AWS で定義されるアクション HealthImaging」を参照してください。

HealthImaging アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいAWS のアイデンティティベースのポリシーの例 HealthImaging。

ACLs HealthImaging

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権限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。

での RBAC HealthImaging

RBAC のサポート	はい

IAM で使用される従来の認証モデルは、ロールベースのアクセスコントロール (RBAC) と呼ばれます。RBAC は、 の外部で AWS ロール と呼ばれる、個人の職務機能に基づいてアクセス許可を定義します。詳細については、 IAM ユーザーガイド の「ABAC と従来の RBAC モデルの比較」を参照してください。

での ABAC HealthImaging

ABAC (ポリシー内のタグ) のサポート	部分的

> **Warning**
>
> ABAC は `SearchImageSets`API アクションでは適用されません。`SearchImageSets`アクションにアクセスできるユーザーであれば、データストア内の画像セットのすべてのメタデータにアクセスできます。

> **Note**
>
> 画像セットはデータストアの子リソースです。ABAC を使用するには、画像セットにデータストアと同じタグが必要です。詳細については、「AWS によるリソースのタグ付け HealthImaging」を参照してください。

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。次に、プリンシパルのタグがアクセスを試行するリソースのタグと一致したときにオペレーションを許可するよう、ABAC ポリシーを設計します。

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制御するには、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 Yes です。サービスが一部のリソースタイプに対してのみ 3 つの条件キーすべてをサポートする場合、値は Partial です。

ABAC の詳細については、「IAM ユーザーガイド」の [ABAC とは?」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制御 (ABAC) を使用する」を参照してください。

での一時的な認証情報の使用 HealthImaging

一時的な認証情報のサポート	はい

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 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ています。例えば、会社の Single Sign-On (SSO) リンク AWS を使用して にアクセスすると、そのプロセスは自動的に一時的な認証情報を作成します。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「ロールへの切り替え (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、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する AWS. AWS recommends にアクセスできます。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。

のクロスサービスプリンシパル許可 HealthImaging

プリンシパル権限のサポート	はい

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場合 AWS、ユーザーはプリンシパルと見なされます。ポリシーによって、プリンシパルに許可が付与されます。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により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。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。アクションがポリシーで追加の依存アクションを必要とするかどうかを確認するには、「サービス認証リファレンス」の「AWS のアクション、リソース、および条件キー HealthImaging」を参照してください。

HealthImaging のサービスロール

サービスロールに対するサポート	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 のサービスに権限を委任するロールの作成」を参照してください。

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

のサービスにリンクされたロール HealthImaging

サービスにリンクされたロールのサポート	いいえ

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 サービス」を参照してください。表の中から、[サービスにリンクされたロール] 列から Yes を含むサービスを検索します。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 のアイデンティティベースのポリシーの例 HealthImaging

デフォルトでは、 ユーザーおよびロールには、 HealthImaging リソースを作成または変更する権限はありません。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アクセス許可をユーザーに付与するため、IAM ポリシーを作成できます。その後、管理者がロールに IAM ポリシーを追加すると、ユーザーはロールを引き受けることができます。

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 ポリシーの作成」を参照してください。

各リソースタイプの ARNs「サービス認証リファレンス」の「Awesome AWS 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

- ポリシーのベストプラクティス
- HealthImaging コンソールを使用する
- 自分の許可の表示をユーザーに許可する

ポリシーの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かが HealthImaging リソースを作成、アクセス、または削除できるどうかを決定します。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の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 管理ポリシーの使用を開始し、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を付与するには、多くの一般的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して、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「職務機能の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 ポリシーでアクセス許可を設定する場合、タスクの実行に必要なアクセス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、最小特権権限とも呼ばれています。IAM を使用してアクセス

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、 などの特定の を通じてサービスアクションを使用する場合 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、安全で機能的なアクセス許可を確保する - 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 ポリシーの検証」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、セキュリティを強化するために MFA を有効にします。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 保護 API アクセスの設定」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。

HealthImaging コンソールを使用する

AWS HealthImaging コンソールにアクセスするには、一連の最小限のアクセス許可が必要です。これらのアクセス許可により、 の HealthImaging リソースの詳細をリスト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 HealthImaging コンソールを使用できるようにするには、エンティティに HealthImaging *ConsoleAccess*または *ReadOnly* AWS 管理ポリシーもアタッチします。詳細については、『IAM ユーザーガイド』の「ユーザーへの権限の追加」を参照してください。

自分の許可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添付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を作成する方法を示します。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 HealthImaging の AWS マネージドポリシー

AWSマネージドポリシーは、AWSが作成および管理するスタンドアロンポリシーです。AWSマネージドポリシーは、多くの一般的なユースケースでアクセス許可を提供できるように設計されているため、ユーザー、グループ、ロールへのアクセス許可の割り当てを開始できます。

AWSマネージドポリシーは、ご利用の特定のユースケースに対して最小特権のアクセス許可を付与しない場合があることにご注意ください。AWS のすべてのお客様が使用できるようになるのを避けるためです。ユースケース別にカスタマーマネージドポリシーを定義することで、アクセス許可を絞り込むことをお勧めします。

AWS マネージドポリシーで定義したアクセス権限は変更できません。AWSがAWSマネージドポリシーに定義されているアクセス許可を更新すると、更新はポリシーがアタッチされているすべてのプリンシパルアイデンティティ (ユーザー、グループ、ロール) に影響します。新しいAWS のサービスを起動するか、既存のサービスで新しい API オペレーションが使用可能になると、AWSがAWSマネージドポリシーを更新する可能性が最も高くなります。

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」を参照してください。

トピック

- AWS マネージドポリシー: AWSHealthImagingFullAccess
- AWS マネージドポリシー: AWSHealthImagingReadOnlyAccess
- HealthImaging のAWS マネージドポリシーに対する更新

AWS マネージドポリシー: AWSHealthImagingFullAccess

`AWSHealthImagingFullAccess` ポリシーは IAM ID に添付できます。

このポリシーは、HealthImaging のすべてのアクションに管理者権限を付与します。

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "medical-imaging:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "medical-imaging.amazonaws.com"
                }
            }
        }
    ]
}
```

AWS マネージドポリシー: AWSHealthImagingReadOnlyAccess

AWSHealthImagingReadOnlyAccess ポリシーは IAM ID に添付できます。

このポリシーは、特定の AWS HealthImaging アクションに対し読み取り専用アクセス許可を付与します。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "medical-imaging:GetDICOMImportJob",
            "medical-imaging:GetDatastore",
            "medical-imaging:GetImageFrame",
            "medical-imaging:GetImageSet",
            "medical-imaging:GetImageSetMetadata",
            "medical-imaging:ListDICOMImportJobs",
            "medical-imaging:ListDatastores",
            "medical-imaging:ListImageSetVersions",
```

```
            "medical-imaging:ListTagsForResource",
            "medical-imaging:SearchImageSets"
        ],
        "Resource": "*"
    }]
}
```

HealthImaging のAWS マネージドポリシーに対する更新

このサービスがこれらの変更の追跡を開始してからの、HealthImaging の AWS マネージドポリシーに対する更新に関する詳細を表示します。このページの変更に関する自動通知については、リリース ページの RSS フィードを購読してください。

変更	説明	日付
HealthImaging が変更の追跡を開始	HealthImaging が AWS マネージドポリシーの変更の追跡を開始しました。	2023 年 7 月 19 日

AWS HealthImaging アイデンティティとアクセスのトラブルシューティング

次の情報は、 と IAM の使用に伴って発生する可能性がある一般的な問題の診断 HealthImaging や修復に役立ちます。

トピック

- でアクションを実行する権限がない HealthImaging
- iam を実行する権限がありません。PassRole
- 自分の 以外のユーザーに HealthImaging リソース AWS アカウント へのアクセスを許可したい

でアクションを実行する権限がない HealthImaging

「I am not authorized to perform an action in Amazon Bedrock」というエラーが表示された場合、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。

次の例は、mateojackson という IAM ユーザーがコンソールを使用して架空の *my-example-widget* リソースに関する詳細を表示しようとしたとき、架空の AWS:*GetWidget* アクセス許可がない場合に発生するエラーを示しています。

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

この場合、AWS:*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。サインイン資格情報を提供した担当者が管理者です。

iam を実行する権限がありません。PassRole

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 HealthImaging 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、既存のロールをそのサービスに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アクセス許可が必要です。

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 HealthImaging 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。Mary には、ロールをサービスに渡すアクセス許可がありません。

```
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 Mary 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 HealthImaging 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:

- がこれらの機能 HealthImaging をサポートしているかどうかを確認するには、「」を参照してくださいAWS と IAM の HealthImaging 連携方法。
- 所有 AWS アカウント する 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 する別の の IAM ユーザーへのアクセスを許可する」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の 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「第三者 AWS アカウント が所有する へのアクセスの許可」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権を付与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) にアクセス権を付与する」を参照してください。
- クロスアカウントアクセスでのロールとリソースベースのポリシーの使用の違いの詳細については、「IAM ユーザーガイド」の「IAM ロールとリソースベースのポリシーとの相違点」を参照してください。

AWS HealthImaging でのログ記録とモニタリング

ログの作成とモニタリングは、AWS HealthImaging のセキュリティ、信頼性、可用性、パフォーマンスを維持する上で重要です。AWSには、AWS HealthImaging をモニタリングし、問題発生時に報告を行い、必要に応じて自動アクションを実行するための以下のようなログ記録とモニタリング用のツールがあります。

- AWS CloudTrail は、AWSアカウントにより、またはそのアカウントに代わって行われた API コールや関連イベントを取得し、指定した Amazon S3 バケットにログファイルを配信します。AWS を呼び出したユーザーとアカウント、呼び出し元の 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 のリソースおよび AWS で実行しているアプリケーションを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。

トピック

- AWS CloudTrailを使用した AWS HealthImaging API コールのログ作成
- Amazon CloudWatch での AWS HealthImagingのモニタリング

AWS CloudTrailを使用した AWS HealthImaging API コールのログ作成

AWS HealthImaging は、HealthImaging のユーザー、ロール、またはAWSサービスが実行したアクションの記録を提供するサービスである AWS CloudTrail と統合されています。CloudTrail は、HealthImaging のすべての API コールをイベントとしてキャプチャします。キャプチャされたコールには、HealthImaging コンソールからのコールと、HealthImaging API オペレーションへのコードコールが含まれます。証跡を作成する場合は、HealthImaging のイベントを含む Amazon S3 バケットへの CloudTrail イベントの継続的な 配信を有効にすることができます。追跡を設定しない場合でも、CloudTrail コンソールの [Event history] (イベント履歴) で最新のイベントを表示できます。CloudTrail で収集された情報を使用して、 HealthImaging に対するリクエスト、リクエスト元の IP アドレス、リクエスト元のユーザー、リクエスト日時などの詳細を確認できます。

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。

証跡の作成

CloudTrail は、アカウント作成時に AWS アカウント で有効になります。HealthImaging でアクティビティが発生すると、そのアクティビティは [イベント履歴] の他の AWS サービスのイベントとともに CloudTrail イベントに記録されます。最近のイベントは、AWS アカウント で表示、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。

HealthImaging のイベントなど、AWS アカウントで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。追跡 により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、コンソールで証跡を作成するときに、証跡がすべての AWS リージョン に適用されます 証跡は、AWS パーティションのすべてのリージョンからのイベントをログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、CloudTrail ログで収集したイベントデータをより詳細に分析し、それに基づく対応するためにその他の AWS のサービスを設定できます。詳細については、次を参照してください。

- 「追跡を作成するための概要」

- CloudTrail がサポートされているサービスと統合
- CloudTrail の Amazon SNS 通知の設定
- 複数のリージョンから CloudTrail ログファイルを受け取るおよび複数のアカウントから CloudTrail ログファイルを受け取る

大半の HealthImaging アクションは CloudTrail によってログに記録され、AWS HealthImaging、API リファレンス で文書化されます。例えば、`CreateDatastore`、`GetDatastore`、`DeleteDatastore` の各アクションを呼び出すと、CloudTrail ログファイルにエントリが生成されます。

各イベントまたはログエントリには、リクエストの生成者に関する情報が含まれます。アイデンティティ情報は、以下を判別するのに役立ちます。

- リクエストが、ルート認証情報と AWS Identity and Access Management (IAM) ユーザー認証情報のどちらを使用して送信されたか。
- リクエストがロールまたはフェデレーティッドユーザーのテンポラリなセキュリティ認証情報を使用して行われたかどうか。
- リクエストが、別の AWS サービスによって送信されたかどうか。

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

ログエントリについて

「トレイル」は、指定した Simple Storage Service (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 のログファイルには、単一か複数のログエントリがあります。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、パブリック 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。

次の例は、`GetDICOMImportJob` アクションを示す HealthImaging のログエントリです。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "XXXXXXXXXXXXXXXXXXXXX:ce6d90ba-5fba-4456-a7bc-f9bc877597c3",
```

```
        "arn": "arn:aws:sts::123456789012:assumed-role/TestAccessRole/
ce6d90ba-5fba-4456-a7bc-f9bc877597c3"
        "accountId": "123456789012",
        "accessKeyId": "XXXXXXXXXXXXXXXXXXX",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "XXXXXXXXXXXXXXXXXXXXX",
                "arn": "arn:aws:iam::123456789012:role/TestAccessRole",
                "accountId": "123456789012",
                "userName": "TestAccessRole"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-10-28T15:52:42Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-10-28T16:02:30Z",
    "eventSource": "medical-imaging.amazonaws.com",
    "eventName": "GetDICOMImportJob",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "aws-sdk-java/2.18.1 Linux/5.4.209-129.367.amzn2int.x86_64 OpenJDK_64-
Bit_Server_VM/11.0.17+9-LTS Java/11.0.17 vendor/Amazon.com_Inc. md/internal io/sync
 http/Apache cfg/retry-mode/standard",
    "requestParameters": {
        "jobId": "5d08d05d6aab2a27922d6260926077d4",
        "datastoreId": "12345678901234567890123456789012"
    },
    "responseElements": null,
    "requestID": "922f5304-b39f-4034-9d2e-f062de092a44",
    "eventID": "26307f73-07f4-4276-b379-d362aa303b22",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "824333766656",
    "eventCategory": "Management"
}
```

Amazon CloudWatch での AWS HealthImagingのモニタリング

CloudWatch を使用して AWS HealthImaging をモニタリングできます。CloudWatch は生データを収集し、それを読み取り可能なほぼリアルタイムのメトリクスに処理します。これらの統計は 15 か月間保持されるため、その履歴情報を利用して、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、特定のしきい値をモニタリング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。

> **Note**
>
> メトリクスはすべての HealthImaging API について報告されます。

以下の表は、HealthImaging のメトリクスとディメンションの一覧です。それぞれは、ユーザーが指定したデータ範囲の頻度カウントとして表示されます。

メトリクス

メトリクス	説明
コールカウント	API の呼び出し回数。これはアカウントまたは指定したデータストアについて報告できます。 単位はカウント 有効な統計: Sum、Count ディメンション:オペレーション、データストア ID、データストアタイプ

HealthImaging のメトリクスは、AWS Management Console、AWS CLI、または CloudWatch API で取得できます。CloudWatch API は、いずれかの Amazon AWS Software Development Kit (SDK) または Amazon CloudWatch API ツールでも使用できます。HealthImaging コンソールには、CloudWatch API の raw データに基づくグラフが表示されます。

CloudWatch で HealthImaging をモニタリングするには、適切な CloudWatch アクセス権限が必要です。詳細については、「 Amazon Aurora ユーザーガイド 」の「CloudWatch のアイデンティティとアクセス管理」を参照してください。

HealthImaging メトリクスの表示

メトリクスを表示する方法 (CloudWatch コンソール)

1. AWS Management Console にサインインして、CloudWatch コンソールを開きます。
2. メトリクス で、すべてのメトリクス、[AWS/Medical Imaging] の順に選択します。
3. ディメンションを選択してメトリクスの名前を選んだら、グラフに追加 を選択します。
4. 日付範囲の値を選択します。選択した日付範囲のメトリクスカウントがグラフに表示されます。

CloudWatch を使用したアラームの作成

CloudWatch アラームは指定期間中に単一のメトリクスを監視し、1 つ以上のアクションを実行して Amazon Simple Notification Service (Amazon SNS) トピックまたは Auto Scaling ポリシーに通知を送信します。アクションは、複数の指定期間にわたって特定のしきい値を基準としたメトリクスの値に応じて実行されます。アラームの状態が変わったときにも、CloudWatch は Amazon SNS メッセージを送信できます。

CloudWatch アラームがアクションを呼び出すのは、状態が変わってから指定期間が経過するまで、その新しい状態が続いた場合に限ります。詳細については、「CloudWatch アラームの使用」を参照してください。

AWS HealthImaging のコンプライアンス検証

サードパーティーの監査者は、さまざまな AWS コンプライアンスプログラムの一環として AWS HealthImaging のセキュリティとコンプライアンスを評価します。HealthImagingの場合、これには HIPAA が含まれます。

特定のコンプライアンスプログラムの対象となる AWS サービスのリストについては、「コンプライアンスプログラムによる AWS 対象範囲内のサービス」を参照してください。一般的な情報については、AWS コンプライアンスプログラムを参照してください。

AWS Artifact 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。詳細については、「AWS Artifact にレポートをダウンロードする」を参照してください。

AWS HealthImaging を使用する際のユーザーのコンプライアンス責任は、ユーザーのデータの機密性や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。AWS では、コンプライアンスに役立つ以下のリソースを提供しています。

- AWSパートナーソリューション – セキュリティおよびコンプライアンスの自動化リファレンスデプロイガイドは、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に重点を置いたベースライン環境を AWS にデプロイするための手順を説明します。
- Architecting for HIPAA Security and Compliance Whitepaper (HIPAA のセキュリティとコンプライアンスのためのアーキテクチャの設計に関するホワイトペーパー) - このホワイトペーパーは、企業が AWS を使用して HIPAA 準拠のアプリケーションを作成する方法について説明します。
- AWS における GxP システム — このホワイトペーパーでは、GxP 関連のコンプライアンスとセキュリティに対する AWS の取り組み方についての情報を提供し、GxP の観点における AWS サービスの利用に関するガイダンスを提供します。
- AWS コンプライアンスのリソース - このワークブックとガイドのコレクションは、お客様の業界や所在地に適用される場合があります。
- ルールを使用したリソースの評価 – AWS Config がリソース設定が社内のプラクティス、業界のガイドライン、規制にどの程度適合しているかを評価します。
- AWS Security Hub - この AWS サービスでは、AWS 内のセキュリティ状態を包括的に表示しており、セキュリティ業界の標準およびベストプラクティスへの準拠を確認するのに役立ちます。

AWS HealthImaging のレジリエンス

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。AWS リージョン には、低レイテンシー、高いスループット、そして高度の冗長ネットワークで接続されている物理的に独立・隔離された複数のアベイラビリティーゾーンがあります。アベイラビリティーゾーンを使用すると、中断することなくゾーン間で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 とアベイラビリティーゾーンの詳細については、「AWS グローバルインフラストラクチャ」を参照してください。

AWS HealthImaging では、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に対応できるように複数の機能を提供しています。

AWS HealthImaging のインフラストラクチャセキュリティ

マネージド型サービスである AWS HealthImaging は、ホワイトペーパー「Amazon Web Services: のセキュリティプロセスの概要」に記載されている AWS グローバルネットワークセキュリティ対策により保護されています。

AWS が発行している APIコールを使用して、ネットワーク経由で HealthImaging にアクセスします。クライアントは、Transport Layer Security (TLS) 1.3 以降をサポートする必要があります。また、Ephemeral Diffie-Hellman (DHE) や Elliptic Curve Ephemeral Diffie-Hellman (ECDHE) などの Perfect Forward Secrecy (PFS) を使用した暗号スイートもクライアントでサポートされている必要があります。これらのモードは、Java 7 以降など、最近のほとんどのシステムでサポートされています。

また、リクエストには、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。AWS Security Token Service (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

AWS CloudFormation による AWS HealthImaging リソースの作成

AWS HealthImaging は、リソースとインフラストラクチャの作成と管理の所要時間を短縮できるように AWS リソースをモデル化して設定するためのサービスである AWS CloudFormation と統合されています。必要なすべての AWS リソースを記述したテンプレートを作成すれば、AWS CloudFormation がこれらのリソースのプロビジョニングや設定を処理します。

AWS CloudFormation を使用すると、テンプレートを再利用して HealthImaging リソースをいつでも繰り返しセットアップできます。リソースを一度記述するだけで、同じリソースを複数の AWS アカウント とリージョンで何度でもプロビジョニングできます。

HealthImaging と AWS CloudFormation テンプレート

HealthImaging および関連サービスのリソースをプロビジョニングして設定するには、AWS CloudFormation テンプレートについて理解しておく必要があります。テンプレートは、JSON またはYAMLでフォーマットされたテキストファイルです。これらのテンプレートには、AWS CloudFormation スタックにプロビジョニングしたいリソースを記述します。JSON や YAML に不慣れな方は、AWS CloudFormation Designer を使えば、AWS CloudFormation テンプレートを使いこなすことができます。詳細については、「AWS CloudFormation ユーザーガイド」の「AWS CloudFormation Designer とは」を参照してください。

AWS HealthImaging は、AWS CloudFormation を使用したデータストアの作成をサポートしています。HealthImaging データストアのプロビジョニング用の JSON テンプレートと YAML テンプレートの例を含む詳細情報については、「AWS CloudFormationユーザーガイド」の「AWS HealthImagingリソースタイプのリファレンス」を参照してください。

AWS CloudFormation の詳細はこちら

AWS CloudFormation の詳細については、以下のリソースを参照してください。

- AWS CloudFormation
- AWS CloudFormation ユーザーガイド
- AWS CloudFormation API リファレンス
- AWS CloudFormation コマンドラインインターフェイスユーザーガイド

AWS HealthImaging とインターフェース VPC エンドポイント (AWS PrivateLink)

VPC と AWS HealthImaging とのプライベート接続を確立するには、 インターフェース VPC エンドポイント を作成します。インターフェースエンドポイントは、インターネットゲートウェイ、NAT デバイス、VPN 接続、AWS Direct Connect 接続のいずれも必要とせずに DevOps Guru API にプライベートにアクセスできるテクノロジーである AWS PrivateLink を利用します。VPC のインスタンスは、パブリック IP アドレスがなくても HealthImaging API と通信できます。VPC と HealthImaging 間のトラフィックは、Amazon ネットワークから離れません。

各インターフェースエンドポイントは、サブネット内の 1 つ以上の Elastic Network Interface によって表されます。

詳細については、「 Amazon VPC ユーザーガイド 」の「インターフェース VPC エンドポイント (AWS PrivateLink)」を参照してください。

トピック

- HealthImaging VPC エンドポイントに関する考慮事項
- HealthImaging 用のインターフェース VPC エンドポイントの作成
- HealthImaging 用の VPC エンドポイントポリシーの作成

HealthImaging VPC エンドポイントに関する考慮事項

HealthImaging のインターフェース VPC エンドポイントを設定する前に、 Amazon VPC ユーザーガイド のインターフェースエンドポイントのプロパティと制限を確認してください。

HealthImaging は、VPC からのすべてのAWS HealthImaging API アクション呼び出しをサポートしています。

HealthImaging 用のインターフェース VPC エンドポイントの作成

HealthImaging サービス用の VPC エンドポイントは、Amazon VPC コンソールまたは AWS Command Line Interface (AWS CLI) を使用して作成できます。詳細については、 Amazon VPC ユーザーガイド のインターフェースエンドポイントの作成を参照してください。

HealthImaging 用の VPC エンドポイントは、以下のサービス名を使用して作成します。

- com.amazonaws.*region*.medical-imaging
- com.amazonaws.*region*.runtime-medical-imaging

> **Note**
>
> PrivateLink を使用するには、プライベート DNS を有効にする必要があります。

リージョンのデフォルト DNS 名を使用して、HealthImaging に API リクエストを作成できます。`medical-imaging.us-east-1.amazonaws.com`

詳細については、「Amazon VPC ユーザーガイド」の「インターフェースエンドポイントを介したサービスへのアクセス」を参照してください。

HealthImaging 用の VPC エンドポイントポリシーの作成

VPC エンドポイントには、HealthImaging へのアクセスを制御するエンドポイントポリシーをアタッチできます。このポリシーでは、以下の情報を指定します。

- アクションを実行できるプリンシパル
- 実行可能なアクション
- アクションを実行できるリソース

詳細については、「 Amazon VPC ユーザーガイド 」の「VPC エンドポイントによるサービスのアクセスコントロール」を参照してください。

例: HealthImaging アクション用の VPC エンドポイントポリシー

以下は、HealthImaging 用のエンドポイントポリシーの例です。エンドポイントにアタッチされると、このポリシーは、すべてのリソースですべてのプリンシパルに、HealthImaging アクションへのアクセス権を付与します。

API

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "medical-imaging:*"
         ],
         "Resource":"*"
      }
   ]
}
```

CLI

```
aws ec2 modify-vpc-endpoint \
    --vpc-endpoint-id vpce-id
    --region us-west-2 \
    --private-dns-enabled \
    --policy-document \
    "{\"Statement\":[{\"Principal\":\"*\",\"Effect\":\"Allow\",\"Action\":
[\"medical-imaging:*\"],\"Resource\":\"*\"}]}"
```

AWS HealthImaging リファレンスマテリアル

AWS では、以下のリファレンス資料を使用できます HealthImaging。

> **Note**
>
> すべての HealthImaging API アクションとデータ型は、別のガイドに記載されています。詳細については、AWS HealthImaging API リファレンスを参照してください。

トピック

AWS HealthImaging の DICOM サポート

AWS HealthImaging は特定の DICOM 要素と転送構文をサポートしています。HealthImaging メタデータキーはこれらに基づいているため、サポートされている患者レベル、治験、シリーズレベルの DICOM データ要素に精通してください。インポートを開始する前に、医療画像データが HealthImaging がサポートする転送構文と DICOM の要素制約に準拠していることを確認してください。

トピック

サポートされている SOP クラス

AWS HealthImaging では、廃止済みやプライベートなものを含め、あらゆる SOP クラス UID でエンコードされた DICOM P10 サービスオブジェクトペア (SOP) インスタンスをインポートできます。プライベート属性もすべて保持されます。

メタデータの正規化

DICOM P10 データを AWS HealthImaging にインポートすると、メタデータと画像フレーム (ピクセルデータ) で構成される画像セットに変換されます。変換プロセス中に、HealthImaging メタデータキーは特定のバージョンの DICOM 標準に基づいて生成されます。ヘルスイメージングは現在、DICOM PS3.6 2022b データディクショナリに基づいて、メタデータキーを生成しサポートしています。

AWS HealthImaging は、患者、治験、シリーズレベルで次の DICOM データ要素をサポートしています。

患者レベルの要素

> **Note**
>
> 各患者レベル要素の詳細については、DICOM データ要素のレジストリを参照してください。

AWS HealthImaging は、以下の患者レベルの要素をサポートしています。

```
Patient Module Elements

  (0010,0010) - Patient's Name
  (0010,0020) - Patient ID

Issuer of Patient ID Macro Elements

  (0010,0021) - Issuer of Patient ID
  (0010,0024) - Issuer of Patient ID Qualifiers Sequence
  (0010,0022) - Type of Patient ID
  (0010,0030) - Patient's Birth Date
  (0010,0033) - Patient's Birth Date in Alternative Calendar
  (0010,0034) - Patient's Death Date in Alternative Calendar
  (0010,0035) - Patient's Alternative Calendar Attribute
```

```
  (0010,0040) - Patient's Sex
  (0010,1100) - Referenced Patient Photo Sequence
  (0010,0200) - Quality Control Subject
  (0008,1120) - Referenced Patient Sequence
  (0010,0032) - Patient's Birth Time
  (0010,1002) - Other Patient IDs Sequence
  (0010,1001) - Other Patient Names
  (0010,2160) - Ethnic Group
  (0010,4000) - Patient Comments
  (0010,2201) - Patient Species Description
  (0010,2202) - Patient Species Code Sequence Attribute
  (0010,2292) - Patient Breed Description
  (0010,2293) - Patient Breed Code Sequence
  (0010,2294) - Breed Registration Sequence Attribute
  (0010,0212) - Strain Description
  (0010,0213) - Strain Nomenclature Attribute
  (0010,0219) - Strain Code Sequence
  (0010,0218) - Strain Additional Information Attribute
  (0010,0216) - Strain Stock Sequence
  (0010,0221) - Genetic Modifications Sequence Attribute
  (0010,2297) - Responsible Person
  (0010,2298) - Responsible Person Role Attribute
  (0010,2299) - Responsible Organization
  (0012,0062) - Patient Identity Removed
  (0012,0063) - De-identification Method
  (0012,0064) - De-identification Method Code Sequence

Patient Group Macro Elements

  (0010,0026) - Source Patient Group Identification Sequence
  (0010,0027) - Group of Patients Identification Sequence

Clinical Trial Subject Module

  (0012,0010) - Clinical Trial Sponsor Name
  (0012,0020) - Clinical Trial Protocol ID
  (0012,0021) - Clinical Trial Protocol Name Attribute
  (0012,0030) - Clinical Trial Site ID
  (0012,0031) - Clinical Trial Site Name
  (0012,0040) - Clinical Trial Subject ID
  (0012,0042) - Clinical Trial Subject Reading ID
  (0012,0081) - Clinical Trial Protocol Ethics Committee Name
  (0012,0082) - Clinical Trial Protocol Ethics Committee Approval Number
```

治験レベルの要素

> **Note**
>
> 各治験レベル要素の詳細については、DICOM データ要素のレジストリを参照してください。

AWS HealthImaging は、以下の治験レベルの要素をサポートしています。

```
General Study Module

  (0020,000D) - Study Instance UID
  (0008,0020) - Study Date
  (0008,0030) - Study Time
  (0008,0090) - Referring Physician's Name
  (0008,0096) - Referring Physician Identification Sequence
  (0008,009C) - Consulting Physician's Name
  (0008,009D) - Consulting Physician Identification Sequence
  (0020,0010) - Study ID
  (0008,0050) - Accession Number
  (0008,0051) - Issuer of Accession Number Sequence
  (0008,1030) - Study Description
  (0008,1048) - Physician(s) of Record
  (0008,1049) - Physician(s) of Record Identification Sequence
  (0008,1060) - Name of Physician(s) Reading Study
  (0008,1062) - Physician(s) Reading Study Identification Sequence
  (0032,1033) - Requesting Service
  (0032,1034) - Requesting Service Code Sequence
  (0008,1110) - Referenced Study Sequence
  (0008,1032) - Procedure Code Sequence
  (0040,1012) - Reason For Performed Procedure Code Sequence

Patient Study Module

  (0008,1080) - Admitting Diagnoses Description
  (0008,1084) - Admitting Diagnoses Code Sequence
  (0010,1010) - Patient's Age
  (0010,1020) - Patient's Size
  (0010,1030) - Patient's Weight
  (0010,1022) - Patient's Body Mass Index
  (0010,1023) - Measured AP Dimension
  (0010,1024) - Measured Lateral Dimension
```

```
  (0010,1021) - Patient's Size Code Sequence
  (0010,2000) - Medical Alerts
  (0010,2110) - Allergies
  (0010,21A0) - Smoking Status
  (0010,21C0) - Pregnancy Status
  (0010,21D0) - Last Menstrual Date
  (0038,0500) - Patient State
  (0010,2180) - Occupation
  (0010,21B0) - Additional Patient History
  (0038,0010) - Admission ID
  (0038,0014) - Issuer of Admission ID Sequence
  (0032,1066) - Reason for Visit
  (0032,1067) - Reason for Visit Code Sequence
  (0038,0060) - Service Episode ID
  (0038,0064) - Issuer of Service Episode ID Sequence
  (0038,0062) - Service Episode Description
  (0010,2203) - Patient's Sex Neutered

Clinical Trial Study Module

  (0012,0050) - Clinical Trial Time Point ID
  (0012,0051) - Clinical Trial Time Point Description
  (0012,0052) - Longitudinal Temporal Offset from Event
  (0012,0053) - Longitudinal Temporal Event Type
  (0012,0083) - Consent for Clinical Trial Use Sequence
```

シリーズレベルの要素

> **Note**
>
> 各シリーズレベル要素の詳細については、DICOM データ要素のレジストリを参照してください。

AWS HealthImaging は以下のシリーズレベルの要素をサポートしています。

```
General Series Module

  (0008,0060) - Modality
  (0020,000E) - Series Instance UID
  (0020,0011) - Series Number
  (0020,0060) - Laterality
  (0008,0021) - Series Date
```

```
  (0008,0031) - Series Time
  (0008,1050) - Performing Physician's Name
  (0008,1052) - Performing Physician Identification Sequence
  (0018,1030) - Protocol Name
  (0008,103E) - Series Description
  (0008,103F) - Series Description Code Sequence
  (0008,1070) - Operators' Name
  (0008,1072) - Operator Identification Sequence
  (0008,1111) - Referenced Performed Procedure Step Sequence
  (0008,1250) - Related Series Sequence
  (0018,0015) - Body Part Examined
  (0018,5100) - Patient Position
  (0028,0108) - Smallest Pixel Value in Series
  (0028,0109) - Largest Pixel Value in Series
  (0040,0275) - Request Attributes Sequence
  (0010,2210) - Anatomical Orientation Type
  (300A,0700) - Treatment Session UID

Clinical Trial Series Module

  (0012,0060) - Clinical Trial Coordinating Center Name
  (0012,0071) - Clinical Trial Series ID
  (0012,0072) - Clinical Trial Series Description

General Equipment Module

  (0008,0070) - Manufacturer
  (0008,0080) - Institution Name
  (0008,0081) - Institution Address
  (0008,1010) - Station Name
  (0008,1040) - Institutional Department Name
  (0008,1041) - Institutional Department Type Code Sequence
  (0008,1090) - Manufacturer's Model Name
  (0018,100B) - Manufacturer's Device Class UID
  (0018,1000) - Device Serial Number
  (0018,1020) - Software Versions
  (0018,1008) - Gantry ID
  (0018,100A) - UDI Sequence
  (0018,1002) - Device UID
  (0018,1050) - Spatial Resolution
  (0018,1200) - Date of Last Calibration
  (0018,1201) - Time of Last Calibration
  (0028,0120) - Pixel Padding Value
```

```
Frame of Reference Module

  (0020,0052) - Frame of Reference UID
  (0020,1040) - Position Reference Indicator
```

サポートされる転送構文

AWS HealthImaging は、次の表にある転送構文でエンコードされた DICOM P10 SOP インスタンスをインポートします。HealthImaging は、SOP インスタンスのストレージに加えて、以下の転送構文でエンコードされた SOP インスタンスの画像フレーム (ピクセルデータ) を HTJ2K にトランスコードします。

転送構文 UID	転送構文名
1.2.840.10008.1.2	インプリシット VR エンディアン：DICOM のデフォルト転送構文
1.2.840.10008.1.2.1	エクスプリシット VR リトルエンディアン
1.2.840.10008.1.2.1.99	デフレートされたエクスプリシット VR リトルエンディアン
1.2.840.10008.1.2.2	エクスプリシット VR ビッグエンディアン
1.2.840.10008.1.2.4.50	JPEG ベースライン (プロセス 1)：不可逆JPEG 8 ビット画像圧縮のデフォルト転送構文
1.2.840.10008.1.2.4.51	JPEG ベースライン (プロセス 2 と 4)：不可逆 JPEG 12 ビット画像圧縮のデフォルト転送構文 (プロセス 4 のみ)
1.2.840.10008.1.2.4.70	JPEG 可逆性、非階層型、一次予測 (プロセス 14 [選択値 1])：可逆 JPEG 画像圧縮のデフォルト転送構文
1.2.840.10008.1.2.4.80	JPEG-LS 可逆画像圧縮
1.2.840.10008.1.2.4.81	JPEG-LS 不可逆 (ほぼ可逆性の) 画像圧縮
1.2.840.10008.1.2.4.90	JPEG 2000 画像圧縮 (可逆のみ)

転送構文 UID	転送構文名
1.2.840.10008.1.2.4.91	JPEG 2000 イメージ圧縮
1.2.840.10008.1.2.5	RLE 可逆性

DICOM 要素の制約

AWS HealthImaging は、データをインポートして画像セットのメタデータ属性を更新するときに DICOM 要素の制約を適用します。インポートとメタデータの更新の両方の制約を以下に示します。

医療画像データを AWS HealthImaging にインポートすると、以下の DICOM 要素に最大長制約が適用されます。インポートを正常に行うには、データが最大制限長を超えないようにしてください。

DICOM インポートの制約

HealthImaging キーワード	DICOM キーワード	DICOM キー	長さ制限
DICOMPatientId	PatientID	(0010,0020)	(最小: 0、最大: 64)
DICOMPatientName	PatientName	(0010,0010)	(最小: 0、最大: 256)
DICOMPatientBirthDate	PatientBirthDate	(0010,0030)	最小: 0、最大: 18
DICOMPatientSex	PatientSex	(0010,0040)	最小: 0、最大: 16
DICOMStudyInstanceUID	StudyInstanceUID	(0020,000D)	(最小: 0、最大: 64)
DICOMStudyId	StudyID	(0020,0010)	最小: 0、最大: 16
DICOMStudyDescription	StudyDescription	(0008,1030)	(最小: 0、最大: 64)
DICOMNumberOfStudy RelatedSeries	NumberOfStudyRelatedSeries	(0020,1206)	最小: 0、最大: 10000

HealthImaging キーワード	DICOM キーワード	DICOM キー	長さ制限
DICOMNumberOfStudyRelatedInstances	NumberOfStudyRelated Instances	(0020,1208)	最小: 0、最大: 10000
DICOMAccessionNumber	AccessionNumber	(0008,0050)	最小: 0、最大: 16
DICOMStudyDate	StudyDate	(0008,0020)	最小: 0、最大: 18
DICOMStudyTime	StudyTime	(0008,0030)	最小: 0、最大: 28

AWS HealthImaging でメタデータ属性を更新すると、次の DICOM 要素の制約が適用されます。メタデータ属性を更新する前に、データが以下の制約一覧のいずれにも該当しないことを確認してください。

DICOM メタデータの制約

AWS HealthImagingによって適用される DICOM メタデータの更新制約は次のとおりです。

- 患者/治験/シリーズ/インスタンスレベルの属性のプライベート属性は更新できません
- 次の AWS HealthImaging で生成された属性は更新できません:`SchemaVersion`、`DatastoreID`、`ImageSetID`、`PixelData`、`Checksum`、`Width`、`Height`、`Mir`
- 次の DICOM 属性は更新できません:`Tag.PixelData`、`Tag.StudyInstanceUID`、`Tag.SeriesInstanceUID`、`Tag.SOPInstanceUI`
- VR タイプ SQ の属性 (ネスト属性) は更新できません
- 複数値を持つ属性は更新できません
- 属性 VR タイプと互換性のない値の属性は更新できません
- DICOM 標準で有効属性と見なされない属性は更新できません
- モジュール間で属性を更新することはできません たとえば、顧客ペイロードリクエストの治験レベルで患者レベルの属性が指定されている場合、そのリクエストは無効になる可能性があります。
- 関連する属性モジュールが`ImageSetMetadata`にない場合、属性を更新できません。たとえば、`seriesInstanceUID`のシリーズが既存の画像セットにない場合、`seriesInstanceUID`の属性を更新できません。

AWS HealthImaging ピクセルデータの検証

AWS HealthImaging には、各イメージの可逆圧縮エンコードとデコード状態をチェックするピクセルデータ検証が組み込まれています。

- 画像のオンボーディングプロセスは、インポート前に、インポートジョブが画像の元のピクセルの品質状態をキャプチャすると開始されます。CRC32 アルゴリズムを使用して、画像ごとに一意の変更不可能な画像フレーム解像度チェックサム (IFRC) が生成されます。
- IFRC は、各画像フレームの解像度レベルごとに計算されます。チェックサム値は、ベース解像度から最大解像度までがソートされたリストで、メタデータドキュメントに表示されます。
- 画像はインポートされるとすぐにデコードされ、新しい IFRC が計算されます。HealthImaging は、元の画像のフル解像度の IFRC を、インポートされた画像の新しい IFRC と比較して、正確性を検証します。
- 対応する各画像のエラー状態がインポートジョブの出力ログに記録され、確認と検証が可能になります。

ピクセルデータを検証するには

1. 医療画像データをインポートしたら、インポートジョブの出力ログである `job-output-manifest.json` に記録された、各画像セットの成功 (またはエラー状態) を確認できます。`job-output-manifest.json` の詳細については、「インポートジョブを理解する」を参照してください。
2. `GetImageSetMetadata` アクションを使用して、画像セットに関連するメタデータを取得します。詳細については、「画像セットメタデータの取得」を参照してください。

   画像セットのメタデータには、イメージフレーム (ピクセルデータ) に関する情報が含まれています。`PixelDataChecksumFromBaseToFullResolution` には解像度レベルごとの IFRC (チェックサム) が含まれています。以下は、インポートジョブプロセスの一環として生成される IFRC のメタデータ出力の例です。

```
"ImageFrames": [{
  "ID": "678906789067890123451234512345 12",
  "PixelDataChecksumFromBaseToFullResolution": [
    {
      "Width": 128,
      "Height": 128,
      "Checksum": 2928338830
    },
```

```
    {
      "Width": 256,
      "Height": 256,
      "Checksum": 1362274918
    },
    {
      "Width": 512,
      "Height": 512,
      "Checksum": 2510355201
    }
  ]
```

3. ピクセルデータを検証するには、GitHub のピクセルデータ検証手順にアクセスし、README.md ファイル内の指示に従って、HealthImaging が使用するさまざまな HTJ2K デコーディングライブラリ より可逆画像処理を個別に検証します。データは解像度レベルごとに段階的に読み込まれるので、未加工の入力データの IFRC をユーザー側で計算し、HealthImaging メタデータで提供されている同じ解像度の IFRC 値と比較して、ピクセルデータを検証できます。

AWS 用 HTJ2K デコードライブラリ HealthImaging

HTJ2K はハイスループット JPEG 2000 の略語で、JPEG2000 規格 (ISO/IEC 15444-15:2019) のパート 15 で定義されています。HTJ2K は、解像度のスケーラビリティ、プリシンクト、タイリング、高ビット深度、複数チャネル、色空間のサポートなど、JPEG2000 の高度な機能を引き継いでいます。

インポート中、AWS はすべてのイメージフレームを HTJ2K 損失なし形式で HealthImaging エンコードし、一貫した高速なイメージ表示と HTJ2K の高度な機能へのユニバーサルアクセスを実現します。プログラミング言語によっては、以下の画像フレーム (ピクセルデータ) のデコードライブラリをお勧めします。

- OpenJPH — オープンソース、C++、WASM
- OpenJPEG — オープンソース、C/C++、Java
- openjphpy — オープンソース、Python
- pylibjpeg-openjpeg — オープンソース、Python
- カカドゥ・ソフトウェア — 商用、Java および .NET バインディングを含む C++

画像フレームをデコードすると、表示できます。AWS HealthImaging APIs、次のようなさまざまなオープンソースのウェブビューワーをサポートします。

- Cornerstone.js
- Open Health Imaging Foundation (OHIF)

AWS HealthImaging エンドポイントとクォータ

以下のトピックでは、AWS HealthImaging サービスエンドポイントとクォータについて説明します。

トピック

- サービスエンドポイント
- Service Quotas

サービスエンドポイント

サービスポイントとは、ホストとポートをウェブサービスのエンドポイントとして識別する URL のことです。ウェブサービスの各リクエストには、1 つずつエンドポイントが含まれています。ほとんどの AWS サービスは、特定のリージョンのエンドポイントを提供し、より高速な接続を可能にします。次の表に、AWS のサービスエンドポイントを示します HealthImaging。

リージョン名	リージョン	エンドポイント	プロトコル	
米国東部 (バージニア北部)	us-east-1	medical-imaging.us-east-1.amazonaws.com	HTTPS	
米国西部 (オレゴン)	us-west-2	medical-imaging.us-west-2.amazonaws.com	HTTPS	
アジアパシフィック (シドニー)	ap-southeast-2	medical-imaging.ap-southeast-2.amazonaws.com	HTTPS	

リージョン名	リージョン	エンドポイント	プロトコル	
欧州 (アイルランド)	eu-west-1	medical-imaging.eu-west-1.amazonaws.com	HTTPS	

HTTP リクエストを使用して AWS HealthImaging アクションを呼び出す場合は、呼び出されるアクションに応じて 2 つの異なるエンドポイントを使用する必要があります。次のメニューには、HTTP リクエストで使用可能なサービスエンドポイントと、それらがサポートするアクションがリストされます。

HTTP リクエストでサポートされている API アクション

HTTP リクエストを使用すると、次のデータストア、インポート、タグ付けアクションにエンドポイント経由でアクセスできます。

`https://medical-imaging.region.amazonaws.com`

- CreateDatastore
- GetDatastore
- ListDatastores
- DeleteDatastore
- StartDICOMImportJob
- GetDICOMImportJob
- ListDICOMImportJobs
- TagResource

- ListTagsForResource
- UntagResource

HTTP リクエストを使用すると、次のランタイムアクションにエンドポイント経由でアクセスできます。

`https://runtime-medical-imaging.region.amazonaws.com`

- SearchImageSets
- GetImageSet
- GetImageSetMetadata
- GetImageFrame
- ListImageSetVersions
- UpdateImageSetMetadata
- CopyImageSet
- DeleteImageSet

Service Quotas

Service Quotas は、アカウント内のリソース、アクション、および項目の最大値として定義されます AWS 。

> **Note**
>
> 調整可能なクォータの場合、Service Quotas コンソールを使用してクォータの引き上げをリクエストできます。詳細については、「Service Quotas ユーザーガイド」の「クォータ引き上げのリクエスト」を参照してください。

次の表に、AWS のデフォルトのクォータを示します HealthImaging。

名前	デフォルト	引き上げ可能	説明
データストアあたりの同時 CopyImageSet リクエストの最大数	サポートされている各リージョン: 100	はい	現在の AWS リージョン内のデータストアあたりの最大同時 CopyImageSet リクエスト数
データストアあたりの同時 DeleteImageSet リクエストの最大数	サポートされている各リージョン: 100	はい	現在の AWS リージョン内のデータストアあたりの最大同時 DeleteImageSet リクエスト数
データストアあたりの同時 UpdateImageSetMetadata リクエストの最大数	サポートされている各リージョン: 100	はい	現在の AWS リージョン内のデータストアあたりの最大同時 UpdateImageSetMetadata リクエスト数
データストアあたりの最大同時インポートジョブ	ap-southeast-2: 20 他のサポートされている各リージョン: 100	はい	現在の AWS リージョン内のデータストアあたりの同時インポートジョブの最大数

名前	デフォルト	引き上げ可能	説明
最大データストア	サポートされている各リージョン: 10	はい	現在の AWS リージョン内のアクティブなデータストアの最大数
CopyImageSet リクエストごとにコピー ImageFrames できる の最大数	サポートされている各リージョン: 1,000	はい	現在の AWS リージョンで CopyImageSet リクエストごとにコピー ImageFrames できる の最大数
DICOM インポートジョブ内のファイルの最大数	サポートされている各リージョン: 3,000	はい	現在の AWS リージョンの DICOM インポートジョブ内のファイルの最大数
で許容される最大ペイロードサイズ制限 (KB) UpdateImageSetMetadata	サポートされている各リージョン: 10 KB	はい	現在の AWS リージョン UpdateImageSetMetadata で によって許容される最大ペイロードサイズ制限 (KB)
DICOM インポートジョブ内のすべてのファイルの最大サイズ (GB 単位)	サポートされている各リージョン: 30 GB	いいえ	現在の AWS リージョンの DICOM インポートジョブ内のすべてのファイルの最大サイズ (GB)
DICOM インポートジョブ内の各 DICOM P10 ファイルの最大サイズ (GB 単位)	サポートされている各リージョン: 2 GB	いいえ	現在の AWS リージョンの DICOM インポートジョブの各 DICOM P10 ファイルの最大サイズ (GB)

名前	デフォルト	引き上げ可能	説明
インポート、コピー、および ImageSetMetadata あたりの の最大サイズ制限 (MB) UpdateImageSet	サポートされている各リージョン: 50 MB	はい	UpdateImageSet 現在の AWS リージョンにおけるインポート、コピー、および ImageSetMetadata あたりの の最大サイズ制限 (MB)

AWS HealthImaging スロットリング制限

AWS アカウントには、AWS HealthImaging API アクションに適用されるスロットリング制限があります。すべてのアクションで、スロットリング制限を超えた場合に `ThrottlingException` エラーが発生します。詳細については、AWS HealthImaging API リファレンスを参照してください。

次の表に、AWS HealthImaging API アクションのスロットリング制限を示します。

AWS HealthImaging スロットリング制限

アクション	スロットリングレート	スロットリングバースト
CreateDatastore	0.085 tps	1 tps
GetDatastore	10 tps	20 tps
ListDatastores	5 tps	10 tps
DeleteDatastore	0.085 tps	1 tps
StartDICOMImportJob	0.25 tps	1 tps
GetDICOMImportJob	25 tps	50 tps
ListDICOMImportJobs	10 tps	20 tps

アクション	スロットリングレート	スロットリングバースト
SearchImageSets	25 tps	50 tps
GetImageSet	25 tps	50 tps
GetImageSetMetadata	50 tps	100 tps
GetImageFrame	1,000 tps	2,000 tps
ListImageSetVersions	25 tps	50 tps
UpdateImageSetMetadata	0.25 tps	1 tps
CopyImageSet	0.25 tps	1 tps
DeleteImageSet	0.25 tps	1 tps
TagResource	10 tps	20 tps
ListTagsForResource	10 tps	20 tps
UntagResource	10 tps	20 tps

AWS HealthImaging のサンプルプロジェクト

AWS HealthImagingは GitHub で以下のサンプルプロジェクトを提供しています。

[オンプレミスから AWS HealthImaging への DICOM の取り込み]

DICOM DIMSE ソース (PACS、VNA、CT スキャナー) から DICOM ファイルを受信し、安全な Amazon S3 バケットに保存する、IoT エッジソリューションをデプロイするための AWS サーバーレスプロジェクト。このソリューションは、データベース内の DICOM ファイルのインデックスを作成し、各 DICOM シリーズを AWS HealthImaging にインポートするキューに入れます。これは、AWS IoT Greengrass によって管理されるエッジで実行されるコンポーネントと、AWS クラウドで実行される DICOM 取り込みパイプラインで構成されています。

[タイルレベルマーカー (TLM) プロキシ]

High-Throughput JPEG 2000 (HTJ2K) の機能であるタイルレベルマーカー (TLM) を使用して AWS HealthImaging からイメージフレームを取得する AWS Cloud Development Kit (AWS CDK)

プロジェクト。これにより、低解像度の画像では取得時間が短縮されます。考えられるワークフローには、サムネイルの生成や画像の段階的な読み込みなどがあります。

Amazon CloudFront 配信

エッジからイメージフレームをキャッシュして (GET を使用) 配信する HTTPS エンドポイントで Amazon CloudFront ディストリビューションを作成するための AWS サーバーレスプロジェクト。デフォルトでは、エンドポイントは Amazon Cognito JSON Web トークン (JWT) を使用してリクエストを認証します。認証とリクエスト署名はどちらも、Lambda @Edge を使用してエッジで行われます。このサービスは Amazon CloudFront の機能で、アプリケーションのユーザーの近くでコードを実行できるため、パフォーマンスが向上し、レイテンシーが減少します。インフラストラクチャを管理する必要はありません。

AWS HealthImaging ビューワー UI

AWS HealthImaging に保存されている画像セットのメタデータ属性とイメージフレーム (ピクセルデータ) をプログレッシブデコードで表示できる、バックエンド認証付きのフロントエンド UI をデプロイする AWS Amplify プロジェクト。オプションで、上記のタイルレベルマーカー (TLM) プロキシや Amazon CloudFront 配信プロジェクトを統合して、別の方法でイメージフレームを読み込むことができます。

その他のサンプルプロジェクトを表示するには、GitHub の「AWS HealthImaging のサンプル」を参照してください。

AWS HealthImaging リリース

次の表は、AWS HealthImaging のサービスおよびドキュメントの機能と更新がいつリリースされたかを示しています。

変更	説明	日付
CloudFormation サポート	AWS は、データストアをプロビジョニングするためのコードとしてのインフラストラクチャ (IaC) をサポートする HealthImaging ようになりました。詳細については、「AWS CloudFormation による AWS HealthImaging リソースの作成」を参照してください。	2023 年 9 月 21 日
一般提供	AWS HealthImaging は、米国東部 (バージニア北部)、米国西部 (オレゴン)、欧州 (アイルランド)、アジアパシフィック (シドニー) の各リージョンですべてのお客様にご利用いただけます。	2023 年 7 月 26 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。