デベロッパーガイド

AWS HealthLake

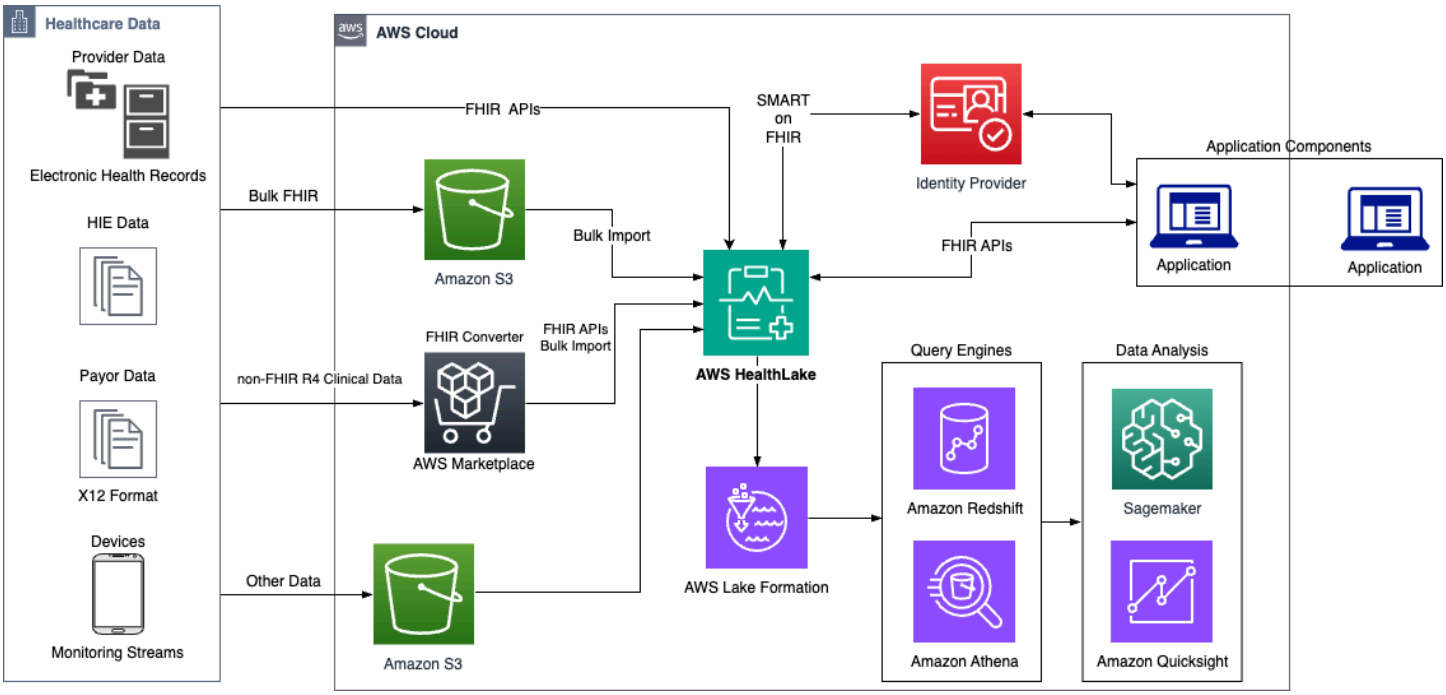
AWS HealthLake: デベロッパーガイド

Table of Contents

とは AWS HealthLake

AWS HealthLake は、高速ヘルスケア相互運用性リソース (FHIR) R4 仕様を使用して、ヘルスデータをクラウドに保存、分析、共有するための HIPAA 対応サービスです。HealthLake のユースケースは次のとおりです。

- エンタープライズヘルスデータ – FHIR R4 ヘルスデータを から直接管理および共有 AWS クラウド し、高いパフォーマンスと可用性を維持します。
- ヘルスケアの相互運用性 – フルマネージド FHIR データストアを介した患者アクセスに関する 21st Century Cures Act への顧客の準拠をサポートします。
- 自然言語処理 (NLP) – 統合された NLP モデルを使用して、非構造化ヘルスデータから意味のある医療情報を抽出します。
- マルチモーダル分析 – HealthLake データと AWS HealthImaging データとデータを組み合わせて AWS HealthOmics 、精密医療に関するインサイトを提供します。



トピック

- 重要な注意点
- の機能 AWS HealthLake
- 関連 AWS サービス

- アクセス AWS HealthLake
- HIPAA の適格性とデータセキュリティ
- 料金

重要な注意点

AWS HealthLake は、専門的な医療上の助言、診断、または治療に代わるものではなく、疾患や健康状態の修復、治療、緩和、予防、診断を目的としていません。お客様は、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に関連するものを含め AWS HealthLake、 の使用の一環として人間によるレビューを代行する責任があります。 AWS HealthLake は、適切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してください。

の機能 AWS HealthLake

AWS HealthLake には以下の機能があります。

FHIR R4 ヘルスデータのインポート

HealthLake ネイティブインポートアクションを使用すると、臨床メモ、ラボレポート、保険金請求など、FHIR データを Amazon S3 バケットから HealthLake データストアに簡単に移行できます。HealthLake は、医療データ交換の FHIR R4 仕様をサポートしています。必要に応じて、AWS HealthLake パートナーと協力してヘルスデータを FHIR R4 形式に変換できます。

ヘルスデータを安全かつ準拠し、監査可能な方法で保存する

HealthLake データストアは、クエリできるようにヘルスデータのインデックス作成に役立ちます。データストアは、各患者の医療履歴を時系列順に完全なビューを作成し、FHIR R4 仕様を使用して情報交換を容易にします。また、インデックスを最新の状態に保つために常に実行されており、耐久性のあるプライマリストレージとインデックススケーリングによる標準の FHIR R4 インタラクションを使用していつでも情報を検索できます。

トランザクション FHIR サーバーを活用する

標準リソース検証用の FHIR APIs、SMART on FHIR 認可、バルクデータ FHIR API エクスポート機能を活用して、データの統一と分析をサポートし、運用コストを削減し、意思決定を向上させます。HealthLake は、HL7 FHIR R4 APIs、FHIR Bulk Data Access、US Core IG STU、HL7

SMART App Launch Framework IG、OAuth 2.0、OpenID Connect など、最新の ONC および CMS 規制標準へのお客様の準拠をサポートしています。

NLP を使用して非構造化医療データを変換する

統合された医療自然言語処理 (NLP) は、HealthLake データストア内のすべての未加工の医療テキストデータを変換して、非構造化医療データから意味のある情報を理解し、抽出します。統合された医療 NLP を使用すると、医療テキストからエンティティ、エンティティ関係、エンティティ特性、保護医療情報 (PHI) を自動的に抽出できます。NLP で抽出されたエンティティは、HealthLake データストア内のネイティブ FHIR R4 リソースとして保存され、FHIR R4 APIs または Amazon Athena (SQL) からアクセスできます。

関連 AWS サービス

AWS HealthLake は、他の AWS サービスと緊密に統合されています。HealthLake を最大限に活用するには、以下のサービスに関する知識が役立ちます。

- AWS Identity and Access Management – IAM を使用して、ID と HealthLake リソースへのアクセスを安全に管理します。
- Amazon Simple Storage Service – Amazon S3 をステージングエリアとして使用して、DICOM データを HealthLake にインポートします。
- AWS CloudTrail – CloudTrail を使用して、HealthLake ユーザーアクティビティと API 使用状況を追跡します。
- Amazon CloudWatch – CloudWatch を使用して HealthLake リソースを監視およびモニタリングします。
- AWS CloudFormation – CloudFormation を使用して、Infrastructure as Code (IaC) テンプレートを実装し、HealthLake でリソースを作成します。
- AWS PrivateLink – Amazon VPC を使用して、インターネットにデータを公開することなく HealthLake と Amazon Virtual Private Cloud 間の接続を確立します。
- Amazon EventBridge – EventBridge を使用して、HealthLake イベントをターゲットにルーティングするルールを作成して、スケーラブルなイベント駆動型アプリケーションを作成します。
- AWS Lake Formation – Lake Formation を使用して、分析と機械学習のために HealthLake データを一元管理、保護、共有します。
- Amazon Athena – Athena を使用して SQL で HealthLake データをクエリし、より深い分析を可能にします。

アクセス AWS HealthLake

AWS マネジメントコンソール AWS Command Line Interface および AWS SDKs AWS HealthLake を使用して にアクセスできます。このガイドでは、 の手順と、 AWS マネジメントコンソール および AWS CLI SDKs の AWS コード例について説明します。

AWS Command Line Interface (AWS CLI)

AWS CLI は、さまざまな AWS 製品のコマンドを提供し、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は、ソフトウェア開発者向けのライブラリ、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、「構築するツール AWS」を参照してください。

AWS マネジメントコンソール

AWS マネジメントコンソール は、HealthLake とその関連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、HealthLake コンソールにサインインできます。

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。1996 AWS年米国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 サービスを使用して保護医療情報 (PHI) を処理、保存、および送信する方法の詳細については、「HIPAA 概要」を参照してください。

PHI と個人を特定できる情報 (PII) を含む HealthLake への接続は暗号化する必要があります。デフォルトでは、HealthLake へのすべての接続は TLS 経由の HTTPS を使用します。HealthLake は、暗号化された顧客コンテンツを保存し、AWS 責任共有モデルに従って動作します。

料金

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

の開始方法 AWS HealthLake

の使用を開始するには AWS HealthLake、 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。

> **Note**
>
> このガイドのリファレンス章では、SMART on FHIR、FHIR R4、および のサポートコンテンツを提供します AWS HealthLake。たとえば、SMART on FHIR 設定、サポートされている FHIR プロファイルの検証、HealthLake エンドポイントに関する情報を確認できます。

HealthLake の概念と設定について学習した後、使用開始に役立つコード例を含む簡単なチュートリアルを利用できます。

トピック

- AWS HealthLake の概念
- セットアップ AWS HealthLake
- AWS HealthLake チュートリアル

AWS HealthLake の概念

以下の用語と概念は、 を理解して使用する上で重要です AWS HealthLake。

概念

- データストア認可戦略
- 統合された NLP
- 統合分析

データストア認可戦略

HealthLake データストアは、単一の 内に存在する FHIR R4 ヘルスデータのリポジトリです AWS リージョン。HealthLake は、次のデータストア認可戦略をサポートしています。

- SigV4 認可 — HealthLake は、AWS 署名バージョン 4 (SigV4) 認可を使用して FHIR API コールを承認します。
- FHIR 認可に関する SMART — HealthLake は、代替医療アプリケーションと FHIR 認可に関する再利用可能なテクノロジー (SMART) を使用して FHIR API コールを承認します。

詳細については、「HealthLake データストアの作成」を参照してください。

統合された NLP

AWS HealthLake は HIPAA 対応の自然言語処理 (NLP) ライブラリと統合され、非構造化医療テキストから意味のあるヘルスデータを抽出します。NLP ライブラリは、病状、薬剤、投与量、検査、治療、処置などの医療エンティティを識別します。エンティティ間の関係を認識し、ICD-10-CM や RxNorm などの医療オントロジーライブラリにリンクします。詳細については、「HealthLake の統合自然言語処理 (NLP)」を参照してください。

統合分析

AWS HealthLake は FHIR `search`と `bundle` APIs を超えて、大量のヘルスデータをクエリおよび分析するための統合分析を提供します。インポート中、HealthLake は SQL インデックスとクエリのテーブルを自動的に生成します。これにより、広範なデータエンジニアリング作業を必要とせずに、複雑な医療データから実用的なインサイトを得ることができます。詳細については、「Amazon Athena を使用した HealthLake データのクエリ」および「AWS HealthLake サンプルプロジェクト」を参照してください。

セットアップ AWS HealthLake

この章では、 AWS マネジメントコンソール を使用して、 の使用を開始 AWS HealthLake してデータストアを作成するために必要なアクセス許可を設定します。データストアを作成するアクセス許可を設定するには、データレイク管理者および HealthLake 管理者である IAM ユーザーまたはロールを作成します。このユーザーを AWS Lake Formation のデータレイク管理者にします。データレイク管理者は、Amazon Athena を使用してデータストアをクエリするために必要なリソースへのアクセス権を Lake Formation に付与します。HealthLake データストアを作成したら、ファイルのインポートとエクスポートのアクセス許可を設定できます。

トピック

- にサインアップする AWS アカウント

- HealthLake を使用するように IAM ユーザーまたはロールを設定する (IAM 管理者)
- Lake Formation で Data Lake 管理者としてユーザーまたはロールを追加する (IAM 管理者)
- S3 バケットを作成する
- データストアの作成
- インポートジョブのアクセス許可の設定
- エクスポートジョブのアクセス許可の設定
- のインストール AWS CLI

にサインアップする AWS アカウント

の使用を開始するには AWS、 が必要です AWS アカウント。の作成の詳細については AWS アカウント、 AWS アカウント管理 リファレンスガイドの「 の開始方法 AWS アカウント」を参照してください。

HealthLake を使用するように IAM ユーザーまたはロールを設定する (IAM 管理者)

ペルソナ: IAM 管理者

IAM ユーザーとロールを作成し、データレイク管理者を追加できるユーザー。

このトピックのこれらのステップは、IAM 管理者が実行する必要があります。

HealthLake データストアを Athena に接続するには、データレイク管理者および HealthLake 管理者である IAM ユーザーまたはロールを作成する必要があります。この新しいユーザーまたはロールは、 AWS Lake Formation を介してデータストアにあるリソースへのアクセスを許可し、`AmazonHealthLakeFullAccess` AWS 管理ポリシーをユーザーまたはロールに追加します。

Important

データレイク管理者である IAM ユーザーまたはロールは、新しいデータレイク管理者を作成できません。データレイク管理者を追加するには、`AdministratorAccess`アクセス権が付与された IAM ユーザーまたはロールを使用する必要があります。

管理者を作成するには

1. IAM **AmazonHealthlakeFullAccess** AWS 管理ポリシーを組織内のユーザーまたはロールに追加します。

   IAM ユーザーの作成に慣れていない場合は、IAM ユーザーガイドの「IAM ユーザーの作成」と「IAM AWS ポリシーの概要」を参照してください。

2. AWS Lake Formation へのアクセス権を IAM ユーザーまたはロールに付与します。

   - 次の IAM AWS 管理ポリシーを組織内のユーザーまたはロールに追加します。**AWSLakeFormationDataAdmin**

   > **Note**
   >
   > このAWSLakeFormationDataAdminポリシーは、すべての AWS Lake Formation リソースへのアクセスを許可します。常に必要最小限のアクセス許可を使用してタスクを達成してください。詳細については、「IAM ユーザーガイド」の「IAM のベストプラクティス」を参照してください。

3. ユーザーまたはロールに次のインラインポリシーを追加します。詳細については、IAM ユーザーガイドのインラインポリシーを参照してください。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-source-bucket/*",
                "arn:aws:s3:::amzn-s3-demo-logging-bucket/*"
            ]
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "ram:GetResourceShareInvitations",
                "ram:AcceptResourceShareInvitation",
                "glue:CreateDatabase",
                "glue:DeleteDatabase"
            ],
            "Resource": "*"
        }
    ]
}
```

AWSLakeFormationDataAdmin ポリシーの詳細については、Lake Formation デベロッパーガイドの「Lake Formation ペルソナ」と「IAM アクセス許可リファレンス」を参照してください。 AWS

Lake Formation で Data Lake 管理者としてユーザーまたはロールを追加する (IAM 管理者)

> **Note**
>
> を統合する場合は、このステップが必要ですSQL インデックスとクエリ。

次に、IAM 管理者は、前のステップで作成したユーザーまたはロールを Lake Formation のデータレイク管理者として追加する必要があります。

データレイク管理者として IAM ユーザーまたはロールを追加するには

1. AWS Lake Formation コンソールを開きます: https://console.aws.amazon.com/lakeformation/

   > **Note**
   >
   > Lake Formation を初めて訪問する場合は、Lake Formation 管理者の定義を求める「Lake Formation へようこそ」ダイアログボックスが表示されます。

2. AWS Lake Formation データレイク管理者に新しいユーザーまたはロールを割り当てます。

   - オプション 1: Welcome to Lake Formation ダイアログボックスが表示された場合。

1. 他のユーザー AWS またはロールの追加 を選択します。
2. 下矢印 (▾) を選択します。
3. Lake Formation 管理者にもなりたい HealthLake 管理者を選択します。
4. [開始する] を選択します。

- オプション 2: ナビゲーションペイン (☰) を使用します。

1. ナビゲーションペイン (☰) を選択します。
2. アクセス許可で、管理ロールとタスクを選択します。
3. データレイク管理者セクションで、管理者の選択 を選択します。
4. データレイク管理者の管理ダイアログボックスで、下矢印 (▾) を選択します。
5. 次に、Lake Formation 管理者にする HealthLake 管理者のユーザーまたはロールを選択または検索します。
6. [保存] を選択します。

3. Lake Formation によって管理されるデフォルトのセキュリティ設定を変更します。HealthLake データストアリソースは、IAM ではなく Lake Formation によって管理される必要があります。更新するには、AWS 「Lake Formation デベロッパーガイド」の「デフォルトのアクセス許可モデルを変更する」を参照してください。

S3 バケットを作成する

FHIR R4 データを にインポートするには AWS HealthLake、2 つの Amazon S3 バケットをお勧めします。Amazon S3 入力バケットは、インポートする FHIR データを保持し、HealthLake はこのバケットから読み取ります。Amazon S3 出力バケットは、インポートジョブの処理結果と、このバケットへの HealthLake の書き込み (ログ) を保存します。

> **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。

このガイドでは、このセクションの後半でインポートアクセス許可を設定するときに、次の Amazon S3 入出力バケットを指定します。

- 入力バケット: `arn:aws:s3:::amzn-s3-demo-source-bucket`
- 出力バケット: `arn:aws:s3:::amzn-s3-demo-logging-bucket`

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

HealthLake データストアは、単一の AWS リージョン内に存在する FHIR R4 データのリポジトリです。 AWS アカウントには、ゼロまたは多数のデータストアを含めることができます。HealthLake は、2 つのデータストア認可戦略をサポートしています。

> [重要]
>
> HealthLake データストアを作成する前に、HealthLake リソースの作成または管理を制限する可能性のある AWS Organization のサービスコントロールポリシー (SCPs) を確認してください。SCPsは、IAM アクセス許可が正しく設定されていても、HealthLake データストアが正常に作成されないようにすることができます。
> HealthLake データストアを作成すると、 `datastoreID`が生成されます。このセクションの後半でインポートアクセス許可を設定する`datastoreID`ときは、 を使用する必要があります。

HealthLake データストアを作成するには、「」を参照してくださいHealthLake データストアの作成。

インポートジョブのアクセス許可の設定

データストアにファイルをインポートする前に、Amazon S3 の入出力バケットにアクセスする許可を HealthLake に付与する必要があります。 Amazon S3 HealthLake アクセスを許可するには、HealthLake の IAM サービスロールを作成し、ロールに信頼ポリシーを追加して HealthLake のロール継承アクセス許可を付与し、Amazon S3 バケットへのアクセスを許可するアクセス許可ポリシーをロールにアタッチします。

インポートジョブを作成するときは、 のこのロールの Amazon リソースネーム (ARN) を指定しますDataAccessRoleArn。IAM ロールと信頼ポリシーの詳細については、「IAM ロール」を参照してください。

アクセス許可を設定したら、インポートジョブを使用してデータストアにファイルをインポートする準備が整います。詳細については、「FHIR インポートジョブの開始」を参照してください。

インポートアクセス許可を設定するには

1. まだ作成していない場合は、出力ログファイルの送信先 Amazon S3 バケットを作成します。Amazon S3 バケットはサービスと同じ AWS リージョンにあり、すべてのオプションでブロックパブリックアクセスを有効にする必要があります。詳細については、Amazon S3のパブリックアクセスブロックの使用」を参照してください。Amazon 所有または顧客所有の KMS キーも暗号化に使用する必要があります。KMS キーの使用の詳細については、「Amazon Key Management Service」を参照してください。
2. HealthLake のデータアクセスサービスロールを作成し、次の信頼ポリシーを使用してそのロールを引き受けるアクセス許可を HealthLake サービスに付与します。HealthLake はこれを使用して出力 Amazon S3 バケットを書き込みます。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "healthlake.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "accountID"
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:healthlake:us-
west-2:111122223333:datastore/fhir/datastoreID"
                }
            }
        }
    ]
}
```

3. データアクセスロールにアクセス許可ポリシーを追加して、Amazon S3 バケットへのアクセスを許可します。をバケットの名前amzn-s3-demo-bucketに置き換えます。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Action": [
            "s3:ListBucket",
            "s3:GetBucketPublicAccessBlock",
            "s3:GetEncryptionConfiguration"
        ],
        "Resource": [
            "arn:aws:s3:::amzn-s3-demo-source-bucket"
        ],
        "Effect": "Allow"
    },
    {
        "Action": [
            "s3:PutObject"
        ],
        "Resource": [
            "arn:aws:s3:::amzn-s3-demo-logging-bucket/*"
        ],
        "Effect": "Allow"
    },
    {
        "Action": [
            "kms:DescribeKey",
            "kms:GenerateDataKey*"
        ],
        "Resource": [
            "arn:aws:kms:us-east-1:012345678910:key/d330e7fc-b56c-4216-a250-
f4c43ef46e83"
        ],
        "Effect": "Allow"
    }]
}
```

エクスポートジョブのアクセス許可の設定

データストアからファイルをエクスポートする前に、Amazon S3 の出力バケットにアクセスする権限を HealthLake に付与する必要があります。 Amazon S3 HealthLake アクセスを許可するには、HealthLake の IAM サービスロールを作成し、ロールに信頼ポリシーを追加して HealthLake のロール継承アクセス許可を付与し、Amazon S3 バケットへのアクセスを許可するアクセス許可ポリシーをロールにアタッチします。

HealthLake のロールを既に作成している場合は、そのロールを再利用して、このトピックにリストされている Amazon S3 バケットをエクスポートするための追加のアクセス許可を付与できます。IAM ロールと信頼ポリシーの詳細については、「IAM でのポリシーとアクセス許可」を参照してください。

[重要]

HealthLake は、ネイティブ SDK エクスポートリクエストと FHIR R4 `$export` オペレーションの両方をサポートしています。使用するエクスポート API に応じて、個別の IAM アクションを指定する必要があります。これにより、 `allow`および アクセス`deny`許可を個別に処理できます。HealthLake SDK と FHIR REST API の両方のエクスポートを制限する場合は、個別の IAM アクションに拒否アクセス許可を適用する必要があります。HealthLake へのフルアクセスをユーザーに付与する場合、IAM ユーザーアクセス許可の変更は必要ありません。

AWS CLI および AWS SDKsの使用:

以下のネイティブ HealthLake アクションは、 AWS CLI および AWS SDKs を使用してデータストアからデータをエクスポートするために使用できます。

- `StartFHIRExportJob`
- `DescribeFHIRExportJob`
- `ListFHIRExportJobs`

FHIR APIs の使用:

次の IAM アクションは、HealthLake データストアからデータをエクスポートしたり、FHIR `$export`オペレーションを使用してエクスポートジョブをキャンセル (削除) したりするために使用できます。
POST:

- StartFHIRExportJobWithPost

GET:

- StartFHIRExportJobWithGet
- DescribeFHIRExportJobWithGet
- GetExportedFile

DELETE:

- CancelFHIRExportJobWithDelete

アクセス許可を設定するユーザーまたはロールには、ロールの作成、ポリシーの作成、ロールへのポリシーのアタッチを行うアクセス許可が必要です。次の IAM ポリシーは、これらのアクセス許可を付与します。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "iam:CreateRole",
                "iam:CreatePolicy",
                "iam:AttachRolePolicy"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": "iam:PassRole",
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "healthlake.amazonaws.com"
                }
```

```
            }
        }
    ]
}
```

エクスポートアクセス許可を設定するには

1. まだ作成していない場合は、データストアからエクスポートするデータの送信先 Amazon S3 バケットを作成します。Amazon S3 バケットはサービスと同じ AWS リージョンにあり、すべてのオプションでブロックパブリックアクセスを有効にする必要があります。詳細については、Amazon S3のパブリックアクセスブロックの使用」を参照してください。Amazon 所有または顧客所有の KMS キーも暗号化に使用する必要があります。KMS キーの使用の詳細については、「Amazon Key Management Service」を参照してください。
2. まだ作成していない場合は、HealthLake のデータアクセスサービスロールを作成し、HealthLake サービスに次の信頼ポリシーで引き受けるアクセス許可を付与します。HealthLake はこれを使用して出力 Amazon S3 バケットを書き込みます。で作成済みの場合はインポートジョブのアクセス許可の設定、再利用して、次のステップで Amazon S3 バケットのアクセス許可を付与できます。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "healthlake.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "accountID"
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:healthlake:us-
west-2:111122223333:datastore/fhir/data store ID"
```

```
                }
            }
        }
    ]
}
```

3. 出力 Amazon S3 バケットへのアクセスを許可するアクセス許可ポリシーをデータアクセスロールに追加します。をバケットの名前amzn-s3-demo-bucketに置き換えます。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Action": [
            "s3:ListBucket",
            "s3:GetBucketPublicAccessBlock",
            "s3:GetEncryptionConfiguration"
        ],
        "Resource": [
            "arn:aws:s3:::amzn-s3-demo-source-bucket"
        ],
        "Effect": "Allow"
    },
    {
        "Action": [
            "s3:PutObject"
        ],
        "Resource": [
            "arn:aws:s3:::amzn-s3-demo-logging-bucket/*"
        ],
        "Effect": "Allow"
    },
    {
        "Action": [
            "kms:DescribeKey",
            "kms:GenerateDataKey*"
        ],
        "Resource": [
            "arn:aws:kms:us-east-1:012345678910:key/d330e7fc-b56c-4216-a250-
f4c43ef46e83"
        ],
        "Effect": "Allow"
```

```
    }]
}
```

のインストール AWS CLI

AWS CLI は、HealthLake のインポートおよびエクスポートジョブのプロパティを記述および一覧表示するために必要です。HealthLake SDKs を使用してこの情報をリクエストすることもできます。

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。

   - の最新バージョンのインストールまたは更新 AWS CLI
   - の開始方法 AWS CLI

2. AWS CLI `config` ファイルで、管理者の名前付きプロファイルを追加します。 AWS CLI コマンドを実行するときは、このプロファイルを使用します。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 healthlake help
   ```

   が正しく設定されている場合 AWS CLI は、 の AWS HealthLake 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 HealthLake チュートリアル

目的

このチュートリアルでは、ネイティブ HealthLake アクションを使用して、FHIR R4 データを HealthLake データストアにインポートします。次に、FHIR RESTful APIs。チュートリアルを終了するには、ネイティブ HealthLake アクションを使用して FHIR データをエクスポートします。

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. FHIR インポートジョブを開始する
2. FHIR インポートジョブのプロパティを取得する
3. FHIR リソースを作成する
4. FHIR リソースの読み取り
5. FHIR リソースを更新する
6. FHIR リソースを削除する
7. FHIR データのエクスポート
8. データストアを削除する

を使用したデータストアの管理 AWS HealthLake

では AWS HealthLake、FHIR R4 リソースのデータストアを作成および管理します。HealthLake データストアを作成すると、FHIR データリポジトリが RESTful API エンドポイントを介して利用可能になります。Synthea オープンソースの FHIR R4 ヘルスデータをデータストアにインポート (プリロード) することを選択できます。詳細については、「プリロードされたデータ型」を参照してください。

> [重要]
>
> HealthLake は、FHIR 上の AWS SigV4 または SMART の 2 種類の FHIR データストア認可戦略をサポートしています。HealthLake FHIR データストアを作成する前に、いずれかの認可戦略を選択する必要があります。詳細については、「データストア認可戦略」を参照してください。

アクティブな HealthLake データストアの FHIR 関連の機能 (動作) を確認するには、その機能ステートメントを取得します。

以下のトピックでは、HealthLake クラウドネイティブアクションを使用して、、 AWS SDKs AWS CLI、 を使用して FHIR データストアを作成、説明、一覧表示、タグ付け、削除する方法について説明します AWS マネジメントコンソール。

トピック

- HealthLake データストアの作成
- HealthLake データストアのプロパティの取得
- HealthLake データストアの一覧表示
- HealthLake データストアのタグ付け
- HealthLake データストアの削除

HealthLake データストアの作成

を使用して`CreateFHIRDatastore`、FHIR R4 仕様に準拠した AWS HealthLake データストアを作成します。HealthLake データストアは、FHIR データのインポート、管理、検索、エクスポートに使用されます。Synthea オープンソースの FHIR R4 ヘルスデータをデータストアにインポート (プリ

ロード) することを選択できます。詳細については、「プリロードされたデータ型」を参照してください。

> [重要]
>
> HealthLake は、FHIR 上の AWS SigV4 または SMART の 2 種類の FHIR データストア認可戦略をサポートしています。HealthLake FHIR データストアを作成する前に、いずれかの認可戦略を選択する必要があります。詳細については、「データストア認可戦略」を参照してください。

HealthLake データストアを作成すると、FHIR データリポジトリが RESTful API エンドポイントを介して利用可能になります。HealthLake データストアを作成したら、その機能ステートメントをリクエストして、関連するすべての FHIR 関連の機能 (動作) を見つけることができます。

次のメニューでは、 AWS CLI および AWS SDKsの例と、 の手順を示します AWS マネジメントコンソール。詳細については、「AWS HealthLake API リファレンス」の「`CreateFHIRDatastore`」を参照してください。

HealthLake データストアを作成するには

アクセス設定に基づいてメニューを選択します AWS HealthLake。

AWS CLI および SDKs

CLI

AWS CLI

例 1: SigV4 対応の HealthLake データストアを作成する

次の`create-fhir-datastore`例は、 AWS HealthLake で新しいデータストアを作成する方法を示しています。

```
aws healthlake create-fhir-datastore \
    --datastore-type-version R4 \
    --datastore-name "FhirTestDatastore"
```

出力:

```
{
```

```
    "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/datastore/
(Data store ID)/r4/",
    "DatastoreArn": "arn:aws:healthlake:us-east-1:(AWS Account ID):datastore/
(Data store ID)",
    "DatastoreStatus": "CREATING",
    "DatastoreId": "(Data store ID)"
}
```

例 2: SMART on FHIR 対応の HealthLake データストアを作成する

次のcreate-fhir-datastore例は、 AWS HealthLake で FHIR 対応データストアに新しい SMART を作成する方法を示しています。

```
aws healthlake create-fhir-datastore \
    --datastore-name "your-data-store-name" \
    --datastore-type-version R4 \
    --preload-data-config PreloadDataType="SYNTHEA" \
    --sse-configuration '{ "KmsEncryptionConfig": {  "CmkType":
 "CUSTOMER_MANAGED_KMS_KEY", "KmsKeyId": "arn:aws:kms:us-east-1:your-account-
id:key/your-key-id" } }' \
    --identity-provider-configuration  file://
identity_provider_configuration.json
```

identity_provider_configuration.json の内容:

```
{
    "AuthorizationStrategy": "SMART_ON_FHIR_V1",
    "FineGrainedAuthorizationEnabled": true,
    "IdpLambdaArn": "arn:aws:lambda:your-region:your-account-id:function:your-
lambda-name",
    "Metadata": "{\"issuer\":\"https://ehr.example.com\", \"jwks_uri\":
\"https://ehr.example.com/.well-known/jwks.json\",\"authorization_endpoint
\":\"https://ehr.example.com/auth/authorize\",\"token_endpoint\":\"https://
ehr.token.com/auth/token\",\"token_endpoint_auth_methods_supported\":
[\"client_secret_basic\",\"foo\"],\"grant_types_supported\":[\"client_credential
\",\"foo\"],\"registration_endpoint\":\"https://ehr.example.com/auth/
register\",\"scopes_supported\":[\"openId\",\"profile\",\"launch\"],
\"response_types_supported\":[\"code\"],\"management_endpoint\":\"https://
ehr.example.com/user/manage\",\"introspection_endpoint\":\"https://
ehr.example.com/user/introspect\",\"revocation_endpoint\":\"https://
ehr.example.com/user/revoke\",\"code_challenge_methods_supported\":[\"S256\"],
\"capabilities\":[\"launch-ehr\",\"sso-openid-connect\",\"client-public\"]}"
```

```
}
```

出力:

```
{
    "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/datastore/
(Data store ID)/r4/",
    "DatastoreArn": "arn:aws:healthlake:us-east-1:(AWS Account ID):datastore/
(Data store ID)",
    "DatastoreStatus": "CREATING",
    "DatastoreId": "(Data store ID)"
}
```

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

Python

SDK for Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def create_fhir_datastore(
        self,
        datastore_name: str,
        sse_configuration: dict[str, any] = None,
        identity_provider_configuration: dict[str, any] = None,
    ) -> dict[str, str]:
        """
        Creates a new HealthLake data store.
```

```
        When creating a SMART on FHIR data store, the following parameters are
required:
        - sse_configuration: The server-side encryption configuration for a SMART
on FHIR-enabled data store.
        - identity_provider_configuration: The identity provider configuration
for a SMART on FHIR-enabled data store.

        :param datastore_name: The name of the data store.
        :param sse_configuration: The server-side encryption configuration for a
SMART on FHIR-enabled data store.
        :param identity_provider_configuration: The identity provider
configuration for a SMART on FHIR-enabled data store.
        :return: A dictionary containing the data store information.
        """
        try:
            parameters = {"DatastoreName": datastore_name,
"DatastoreTypeVersion": "R4"}
            if (
                sse_configuration is not None
                and identity_provider_configuration is not None
            ):
                # Creating a SMART on FHIR-enabled data store
                parameters["SseConfiguration"] = sse_configuration
                parameters[
                    "IdentityProviderConfiguration"
                ] = identity_provider_configuration

            response =
self.health_lake_client.create_fhir_datastore(**parameters)
            return response
        except ClientError as err:
            logger.exception(
                "Couldn't create data store %s. Here's why %s",
                datastore_name,
                err.response["Error"]["Message"],
            )
            raise
```

次のコードは、SMART on FHIR 対応 HealthLake データストアのパラメータの例を示しています。

```
            sse_configuration = {
                "KmsEncryptionConfig": {"CmkType": "AWS_OWNED_KMS_KEY"}
            }
            # TODO: Update the metadata to match your environment.
            metadata = {
                "issuer": "https://ehr.example.com",
                "jwks_uri": "https://ehr.example.com/.well-known/jwks.json",
                "authorization_endpoint": "https://ehr.example.com/auth/
authorize",
                "token_endpoint": "https://ehr.token.com/auth/token",
                "token_endpoint_auth_methods_supported": [
                    "client_secret_basic",
                    "foo",
                ],
                "grant_types_supported": ["client_credential", "foo"],
                "registration_endpoint": "https://ehr.example.com/auth/register",
                "scopes_supported": ["openId", "profile", "launch"],
                "response_types_supported": ["code"],
                "management_endpoint": "https://ehr.example.com/user/manage",
                "introspection_endpoint": "https://ehr.example.com/user/
introspect",
                "revocation_endpoint": "https://ehr.example.com/user/revoke",
                "code_challenge_methods_supported": ["S256"],
                "capabilities": [
                    "launch-ehr",
                    "sso-openid-connect",
                    "client-public",
                ],
            }
            # TODO: Update the IdpLambdaArn.
            identity_provider_configuration = {
                "AuthorizationStrategy": "SMART_ON_FHIR_V1",
                "FineGrainedAuthorizationEnabled": True,
                "IdpLambdaArn": "arn:aws:lambda:your-region:your-account-
id:function:your-lambda-name",
                "Metadata": json.dumps(metadata),
            }
            data_store = self.create_fhir_datastore(
                datastore_name, sse_configuration,
 identity_provider_configuration
            )
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「CreateFHIRDatastore」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    TRY.
        " iv_datastore_name = 'MyHealthLakeDataStore'
        oo_result = lo_hll->createfhirdatastore(
          iv_datastorename = iv_datastore_name
          iv_datastoretypeversion = 'R4'
        ).
        MESSAGE 'Data store created successfully.' TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllinternalserverex INTO DATA(lo_internal_ex).
        lv_error = |Internal server error: { lo_internal_ex->av_err_code }-
{ lo_internal_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
      CATCH /aws1/cx_hllthrottlingex INTO DATA(lo_throttling_ex).
        lv_error = |Throttling error: { lo_throttling_ex->av_err_code }-
{ lo_throttling_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_throttling_ex.
```

```
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスの「CreateFHIRDatastore」を参照してください。

> **可用性の例**
>
> 必要なものが見つからなかった場合。このページの右側サイドバーにあるフィードバックを提供するリンクを使用して、コード例をリクエストします。

AWS コンソール

> **メモ**
>
> 次の手順では、AWS SigV4 認可を持つ HealthLake データストアを作成します。HealthLake コンソールは、FHIR データストアでの SMART の作成をサポートしていません。

AWS SigV4 認可を使用して HealthLake データストアを作成するには

1. HealthLake コンソールのデータストアの作成ページにサインインします。
2. データストアの作成を選択します。
3. データストア設定セクションで、データストア名に名前を指定します。
4. (オプション) データストア設定セクションの事前ロードサンプルデータで、チェックボックスを選択して Synthea データを事前ロードします。Synthea データはオープンソースのサンプルデータセットです。詳細については、「HealthLake の Synthea プリロードされたデータ型」を参照してください。
5. データストアの暗号化セクションで、AWS 所有キーを使用する (デフォルト) または別の AWS KMS キーを選択する (詳細) を選択します。
6. タグ - オプションセクションでは、データストアにタグを追加できます。データストアのタグ付けの詳細については、「」を参照してくださいHealthLake データストアのタグ付け。
7. データストアの作成を選択します。

   データストアのステータスは、データストアページで確認できます。

HealthLake データストアのプロパティの取得

を使用してDescribeFHIRDatastore、 AWS HealthLake データストアのプロパティを取得します。以下のメニューでは、 の手順と、 AWS マネジメントコンソール および AWS CLI SDK の AWS コード例を示します。 SDKs 詳細については、「AWS HealthLake API リファレンス」の「DescribeFHIRDatastore」を参照してください。

HealthLake データストアのプロパティを取得するには

アクセス設定に基づいてメニューを選択します AWS HealthLake。

AWS CLI および SDKs

CLI

AWS CLI

FHIR データストアの詳細を取得するには

次のdescribe-fhir-datastore例は、 AWS HealthLake のデータストアのプロパティを検索する方法を示しています。

```
aws healthlake describe-fhir-datastore \
    --datastore-id "1f2f459836ac6c513ce899f9e4f66a59"
```

出力:

```
{
    "DatastoreProperties": {
        "PreloadDataConfig": {
            "PreloadDataType": "SYNTHEA"
        },
        "SseConfiguration": {
            "KmsEncryptionConfig": {
                "CmkType": "CUSTOMER_MANAGED_KMS_KEY",
                "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
            }
        },
        "DatastoreName": "Demo",
        "DatastoreArn": "arn:aws:healthlake:us-east-1:<AWS Account ID>:datastore/
<Data store ID>",
```

```
        "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/
datastore/<Data store ID>/r4/",
        "DatastoreStatus": "ACTIVE",
        "DatastoreTypeVersion": "R4",
        "CreatedAt": 1603761064.881,
        "DatastoreId": "<Data store ID>",
        "IdentityProviderConfiguration": {
            "AuthorizationStrategy": "AWS_AUTH",
            "FineGrainedAuthorizationEnabled": false
        }
    }
}
```

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

Python

SDK for Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
 client.

        :return: An instance of HealthLakeWrapper initialized with the default
 HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def describe_fhir_datastore(self, datastore_id: str) -> dict[str, any]:
        """
        Describes a HealthLake data store.
        :param datastore_id: The data store ID.
        :return: The data store description.
        """
        try:
            response = self.health_lake_client.describe_fhir_datastore(
                DatastoreId=datastore_id
```

```
            )
            return response["DatastoreProperties"]
        except ClientError as err:
            logger.exception(
                "Couldn't describe data store with ID %s. Here's why %s",
                datastore_id,
                err.response["Error"]["Message"],
            )
            raise
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「DescribeFHIRDatastore」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        oo_result = lo_hll->describefhirdatastore(
          iv_datastoreid = iv_datastore_id
        ).
        DATA(lo_datastore_properties) = oo_result->get_datastoreproperties( ).
        IF lo_datastore_properties IS BOUND.
          DATA(lv_datastore_name) = lo_datastore_properties-
>get_datastorename( ).
```

```
          DATA(lv_datastore_status) = lo_datastore_properties-
>get_datastorestatus( ).
          MESSAGE 'Data store described successfully.' TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        DATA(lv_error) = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        lv_error = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスの「DescribeFHIRDatastore」を参照してください。

> 可用性の例
>
> 必要なものが見つからなかった場合。このページの右側サイドバーにあるフィードバックを提供するリンクを使用して、コード例をリクエストします。

AWS コンソール

1. HealthLake コンソールのデータストアページにサインインします。
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、すべての HealthLake データストアプロパティを使用できます。

HealthLake データストアの一覧表示

を使用してListFHIRDatastores、データストアのステータスに関係なく、ユーザーのアカウント内のすべての HealthLake データストアを一覧表示します。次のメニューでは、 の手順と、 AWS マ

ネジメントコンソール AWS CLI および AWS SDKs。詳細については、「AWS HealthLake API リファレンス」の「ListFHIRDatastores」を参照してください。

すべての HealthLake データストアを一覧表示するには

アクセス設定に基づいてメニューを選択します AWS HealthLake。

AWS CLI および SDKs

CLI

AWS CLI

FHIR データストアを一覧表示するには

次のlist-fhir-datastores例は、 コマンドの使用方法と、 AWS HealthLake のデータストアのステータスに基づいてユーザーが結果をフィルタリングする方法を示しています。

```
aws healthlake list-fhir-datastores \
    --filter DatastoreStatus=ACTIVE
```

出力:

```
{
    "DatastorePropertiesList": [
    {
        "PreloadDataConfig": {
            "PreloadDataType": "SYNTHEA"
        },
        "SseConfiguration": {
            "KmsEncryptionConfig": {
                "CmkType": "CUSTOMER_MANAGED_KMS_KEY",
                "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
            }
        },
        "DatastoreName": "Demo",
        "DatastoreArn": "arn:aws:healthlake:us-east-1:<AWS Account ID>:datastore/
<Data store ID>",
        "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/
datastore/<Data store ID>/r4/",
        "DatastoreStatus": "ACTIVE",
```

```
        "DatastoreTypeVersion": "R4",
        "CreatedAt": 1603761064.881,
        "DatastoreId": "<Data store ID>",
        "IdentityProviderConfiguration": {
            "AuthorizationStrategy": "AWS_AUTH",
            "FineGrainedAuthorizationEnabled": false
        }
    }
    ]
}
```

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

Python

SDK for Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def list_fhir_datastores(self) -> list[dict[str, any]]:
        """
        Lists all HealthLake data stores.
        :return: A list of data store descriptions.
        """
        try:
            next_token = None
            datastores = []

            # Loop through paginated results.
            while True:
```

```
                parameters = {}
                if next_token is not None:
                    parameters["NextToken"] = next_token
                response =
 self.health_lake_client.list_fhir_datastores(**parameters)
                datastores.extend(response["DatastorePropertiesList"])
                if "NextToken" in response:
                    next_token = response["NextToken"]
                else:
                    break

            return datastores
        except ClientError as err:
            logger.exception(
                "Couldn't list data stores. Here's why %s", err.response["Error"]
["Message"]
            )
            raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「ListFHIRDatastores」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    TRY.
        oo_result = lo_hll->listfhirdatastores( ).
```

```
        DATA(lt_datastores) = oo_result->get_datastorepropertieslist( ).
        DATA(lv_datastore_count) = lines( lt_datastores ).
        MESSAGE |Found { lv_datastore_count } data store(s).| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllthrottlingex INTO DATA(lo_throttling_ex).
        lv_error = |Throttling error: { lo_throttling_ex->av_err_code }-
{ lo_throttling_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_throttling_ex.
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスのListFHIRDatastores」を参照してください。

> 可用性の例
>
> 必要なものが見つからなかった場合。このページの右側サイドバーにあるフィードバックを提供するリンクを使用して、コード例をリクエストします。

AWS コンソール

- HealthLake コンソールのデータストアページにサインインします。

  すべての HealthLake データストアは、データストアセクションの下に表示されます。

HealthLake データストアのタグ付け

メタデータは、タグの形式で HealthLake データストアに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、データストアの管理、識別、整理、検索、フィルタリングに役立ちます。

> [重要]
>
> 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。タグは、プライベートデータまたは機密データに使用することを目的としたものではありません。

以下のトピックでは、 AWS マネジメントコンソール、、 AWS CLIおよび AWS SDKs を使用して HealthLake 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、「 ガイド」の「 AWS リソースのタグ付け」を参照してください。 AWS 全般のリファレンス

トピック

- HealthLake データストアのタグ付け
- HealthLake データストアのタグの一覧表示
- HealthLake データストアのタグ解除

HealthLake データストアのタグ付け

`TagResource` を使用して HealthLake データストアにタグを付けます。以下のメニューでは、 の手順 AWS マネジメントコンソール と、 AWS CLI および AWS SDKs。詳細については、「AWS HealthLake API リファレンス」の「`TagResource`」を参照してください。

HealthLake データストアにタグを付けるには

アクセス設定に基づいてメニューを選択します AWS HealthLake。

AWS CLI および SDKs

CLI

AWS CLI

データストアにタグを追加するには

次の `tag-resource` の例では、データストアにタグを追加する方法を示します。

```
aws healthlake tag-resource \
    --resource-arn "arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/0725c83f4307f263e16fd56b6d8ebdbe" \
```

```
    --tags '[{"Key": "key1", "Value": "value1"}]'
```

このコマンドでは何も出力されません。

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

Python

SDK for Python (Boto3)

```
   @classmethod
   def from_client(cls) -> "HealthLakeWrapper":
       """
       Creates a HealthLakeWrapper instance with a default AWS HealthLake
client.

       :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
       """
       health_lake_client = boto3.client("healthlake")
       return cls(health_lake_client)


   def tag_resource(self, resource_arn: str, tags: list[dict[str, str]]) ->
None:
       """
       Tags a HealthLake resource.
       :param resource_arn: The resource ARN.
       :param tags: The tags to add to the resource.
       """
       try:
           self.health_lake_client.tag_resource(ResourceARN=resource_arn,
Tags=tags)
       except ClientError as err:
           logger.exception(
               "Couldn't tag resource %s. Here's why %s",
               resource_arn,
               err.response["Error"]["Message"],
           )
           raise
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「TagResource」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    TRY.
        " iv_resource_arn = 'arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        lo_hll->tagresource(
          iv_resourcearn = iv_resource_arn
          it_tags = it_tags
        ).
        MESSAGE 'Resource tagged successfully.' TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスのTagResource」を参照してください。

> 可用性の例
>
> 必要なものが見つからなかった場合。このページの右側サイドバーにあるフィードバックを提供するリンクを使用して、コード例をリクエストします。

AWS コンソール

1. HealthLake コンソールのデータストアページにサインインします。
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. [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。
4. [新しいタグを追加] をクリックします。
5. [キー] を入力し、オプションで [値] を入力します。
6. [保存] を選択します。

HealthLake データストアのタグの一覧表示

`ListTagsForResource` を使用して HealthLake データストアのタグを一覧表示します。以下のメニューでは、 の手順 AWS マネジメントコンソール と、 AWS CLI および AWS SDKs。詳細については、「AWS HealthLake API リファレンス」の「`ListTagsForResource`」を参照してください。

HealthLake データストアのタグを一覧表示するには

アクセス設定に基づいてメニューを選択します AWS HealthLake。

AWS CLI および SDKs

CLI

AWS CLI

データストアのタグを一覧表示するには

次の `list-tags-for-resource` の例では、指定したデータストアに関連付けられているタグを一覧表示します。

```
aws healthlake list-tags-for-resource \
    --resource-arn "arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/0725c83f4307f263e16fd56b6d8ebdbe"
```

出力:

```
{
    "tags": {
        "key": "value",
        "key1": "value1"
    }
}
```

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

Python

SDK for Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
 client.

        :return: An instance of HealthLakeWrapper initialized with the default
 HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
```

```
        return cls(health_lake_client)


    def list_tags_for_resource(self, resource_arn: str) -> dict[str, str]:
        """
        Lists the tags for a HealthLake resource.
        :param resource_arn: The resource ARN.
        :return: The tags for the resource.
        """
        try:
            response = self.health_lake_client.list_tags_for_resource(
                ResourceARN=resource_arn
            )
            return response["Tags"]
        except ClientError as err:
            logger.exception(
                "Couldn't list tags for resource %s. Here's why %s",
                resource_arn,
                err.response["Error"]["Message"],
            )
            raise
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「ListTagsForResource」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    TRY.
        " iv_resource_arn = 'arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        DATA(lo_result) = lo_hll->listtagsforresource(
          iv_resourcearn = iv_resource_arn
        ).
        ot_tags = lo_result->get_tags( ).
        DATA(lv_tag_count) = lines( ot_tags ).
        MESSAGE |Found { lv_tag_count } tag(s).| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスのListTagsForResource」を参照してください。

> 可用性の例
>
> 必要なものが見つからなかった場合。このページの右側サイドバーにあるフィードバックを提供するリンクを使用して、コード例をリクエストします。

AWS コンソール

1. HealthLake コンソールのデータストアページにサインインします。
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。[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。

HealthLake データストアのタグ解除

UntagResource を使用して HealthLake データストアからタグを削除します。以下のメニューでは、 の手順 AWS マネジメントコンソール と、 AWS CLI および AWS SDKs。詳細については、「AWS HealthLake API リファレンス」の「UntagResource」を参照してください。

HealthLake データストアのタグを解除するには

アクセス設定に基づいてメニューを選択します AWS HealthLake。

AWS CLI および SDKs

CLI

AWS CLI

データストアからタグを削除するには

次の untag-resource の例では、データストアからタグを削除する方法を示します。

```
aws healthlake untag-resource \
    --resource-arn "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
b91723d65c6fdeb1d26543a49d2ed1fa" \
    --tag-keys '["key1"]'
```

このコマンドでは何も出力されません。

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

Python

SDK for Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
 client.
```

```
        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def untag_resource(self, resource_arn: str, tag_keys: list[str]) -> None:
        """
        Untags a HealthLake resource.
        :param resource_arn: The resource ARN.
        :param tag_keys: The tag keys to remove from the resource.
        """
        try:
            self.health_lake_client.untag_resource(
                ResourceARN=resource_arn, TagKeys=tag_keys
            )
        except ClientError as err:
            logger.exception(
                "Couldn't untag resource %s. Here's why %s",
                resource_arn,
                err.response["Error"]["Message"],
            )
            raise
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「UntagResource」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    TRY.
        " iv_resource_arn = 'arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        lo_hll->untagresource(
          iv_resourcearn = iv_resource_arn
          it_tagkeys = it_tag_keys
        ).
        MESSAGE 'Resource untagged successfully.' TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP UntagResource」を参照してください。

> **可用性の例**
>
> 必要なものが見つからなかった場合。このページの右側サイドバーにあるフィードバックを提供するリンクを使用して、コード例をリクエストします。

AWS コンソール

1. HealthLake コンソールのデータストアページにサインインします。
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. [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。
4. 削除するタグの横にある [削除] を選択します。
5. [保存] を選択します。

HealthLake データストアの削除

HealthLake データストアを削除するには、 `DeleteFHIRDatastore`を使用します。以下のメニューでは、 の手順 AWS マネジメントコンソール と、 AWS CLI および AWS SDKs。詳細については、「AWS HealthLake API リファレンス」の「`DeleteFHIRDatastore`」を参照してください。

HealthLake データストアを削除するには

アクセス設定に基づいてメニューを選択します AWS HealthLake。

AWS CLI および SDKs

CLI

AWS CLI

FHIR Data Store を削除するには

次の`delete-fhir-datastore`例は、 AWS HealthLake でデータストアとそのすべてのコンテンツを削除する方法を示しています。

```
aws healthlake delete-fhir-datastore \
    --datastore-id (Data store ID)
```

出力:

```
{
    "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/datastore/
(Data store ID)/r4/",
    "DatastoreArn": "arn:aws:healthlake:us-east-1:(AWS Account ID):datastore/
(Data store ID)",
    "DatastoreStatus": "DELETING",
    "DatastoreId": "(Data store ID)"
}
```

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

Python

SDK for Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
 client.

        :return: An instance of HealthLakeWrapper initialized with the default
 HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def delete_fhir_datastore(self, datastore_id: str) -> None:
        """
        Deletes a HealthLake data store.
        :param datastore_id: The data store ID.
        """
        try:

self.health_lake_client.delete_fhir_datastore(DatastoreId=datastore_id)
        except ClientError as err:
            logger.exception(
                "Couldn't delete data store with ID %s. Here's why %s",
                datastore_id,
                err.response["Error"]["Message"],
```

```
            )
            raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「DeleteFHIRDatastore」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        oo_result = lo_hll->deletefhirdatastore(
          iv_datastoreid = iv_datastore_id
        ).
        MESSAGE 'Data store deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_hllaccessdeniedex INTO DATA(lo_access_ex).
        DATA(lv_error) = |Access denied: { lo_access_ex->av_err_code }-
{ lo_access_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_access_ex.
      CATCH /aws1/cx_hllconflictexception INTO DATA(lo_conflict_ex).
        lv_error = |Conflict error: { lo_conflict_ex->av_err_code }-
{ lo_conflict_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_conflict_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
```

```
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスのDeleteFHIRDatastore」を参照してください。

> 可用性の例
>
> 必要なものが見つからなかった場合。このページの右側サイドバーにあるフィードバックを提供するリンクを使用して、コード例をリクエストします。

AWS コンソール

1. HealthLake コンソールのデータストアページにサインインします。
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. [削除] を選択します。

   [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. [Delete] (削除) をクリックします。

での FHIR サブスクリプションの管理 AWS HealthLake

AWS HealthLake は FHIR サブスクリプションをサポートしているため、特定の医療データの変更が発生したときにリアルタイムの通知を受け取ることができます。この機能は、FHIR R5 Backport トピックベースのサブスクリプションモデルを実装し、従来の FHIR R4 サブスクリプションモデルよりもスケーラビリティと柔軟性を向上させます。

FHIR サブスクリプションを使用すると、臨床データの変化に即座に対応するイベント駆動型の医療アプリケーションを構築でき、タイムリーな介入、自動化されたワークフロー、強化されたケア調整が可能になります。

トピック

FHIR サブスクリプションの仕組み

HealthLake の FHIR サブスクリプションは、以下のトピックベースのモデルで動作します。

1. イベントを定義するトピックを作成する: 通知をトリガーできるイベントを指定するサブスクリプショントピックを作成する
2. サブスクライブ: 特定のフィルタリング条件でこれらのトピックへのサブスクリプションを作成する
3. HealthLake モニター: サービスは、条件に一致するイベントを継続的にモニタリングします。
4. 通知の配信: CWhen一致するイベントが発生すると、HealthLake は選択したチャネルを通じて通知を配信します。

主要コンポーネント

FHIR サブスクリプションは、次のコンポーネントで構成されます。

サブスクリプショントピック

サブスクリプショントピックは、通知システムの基盤であり、以下を定義します。

- トリガーイベント: 通知をトリガーする変更 (リソースの作成、更新、削除など)
- 使用可能なフィルター: サブスクライバーが利用できるフィルターオプション
- 通知コンテンツ: 通知に含まれるデータ

次の表に、一般的なトピックタイプを示します。

イベントタイプ	説明	一般的なユースケース
リソースの作成	リソースの作成時にトリガーされます	新しい患者登録、新しい観察記録
リソースの更新	リソースが変更されるとトリガーされます	ステータスの変更、臨床更新
リソースの削除	リソースが削除されるとトリガーされます	監査とコンプライアンスの追跡

サブスクリプション

サブスクリプションは、サブスクリプショントピックで定義された特定のイベントの通知を受信するリクエストです。各サブスクリプションには以下が含まれます。

- トピックリファレンス: サブスクライブするサブスクリプショントピックを指定します
- フィルター: 通知を生成するイベントを選択する条件
- チャネル設定: 通知を配信する場所と方法

- ペイロード設定: 通知に含める詳細レベル

通知チャネル

HealthLake は、次の通知チャネルをサポートしています。

チャネルタイプ	ユースケース	
EventBridge	エンタープライズ統合、サーバーレスワークフロー、クロスAWS サービスオーケストレーション	
REST Hook	ダイレクトエンドポイント通知、サードパーティーのシステム統合	

通知ペイロード

必要に応じて適切なペイロードタイプを選択します。

ペイロードタイプ	説明	セキュリティに関する考慮事項
ID のみ	リソース識別子のみが含まれます	最小限の PHI 露出
フルリソース	最大サイズが 256 KB の完全なリソースコンテンツが含まれます。サイズが 256KB を超える場合は、ID のみに戻ります。	PHI を含む。安全な処理を検証する

ベストプラクティス

パフォーマンスの最適化

- 焦点を絞ったフィルターを使用する: 必須の通知のみを受信するように基準を絞り込む
- 適切なペイロードタイプを選択する: ID のみのペイロードを可能な限り使用してパフォーマンスを向上させる
- 効率的なレシーバーを実装する: 通知レシーバーがメッセージを迅速に処理するようにする

セキュリティに関する考慮事項

- セキュアエンドポイント: REST Hook エンドポイントに適切な認証を実装する
- PHI 保護: PHI を含むフルリソースペイロードには注意してください
- アクセスコントロール: サブスクリプションの作成を承認されたユーザーのみに制限する

オペレーショナルエクセレンス

- 適切な終了日を設定する: 一時的なサブスクリプションの終了日を使用する
- サブスクリプションステータスのモニタリング: サブスクリプションのステータスを定期的にチェックする
- エラー処理の実装: 通知配信の失敗を処理するようにアプリケーションを設計する

を使用した FHIR サブスクリプションのライフサイクル AWS HealthLake

FHIR サブスクリプションのライフサイクルを理解するには、次の手順に従います。

1. SubscriptionTopic を作成する
   - ステータスSubscriptionTopicで を作成する "unknown"
2. Subscription を作成する
   - ステータスSubscriptionで を作成する "requested"
   - HealthLake がSubscription設定を検証します
   - Subscription は既存のトピックを参照する必要があります (トピックのステータスは unknown、draft、)active。
3. Activation
   - 有効な場合、HealthLake は のステータスSubscriptionを に更新します。 "active"

- の作成中にSubscription、指定されたトピックのステータスが の場合"unknown"、サブスクリプションもアクティブ"active"になると、HealthLake はステータスを に更新します。
- サブスクリプションが正常に作成されるまでに通常 5～10 分かかります
- が正常に作成Subscriptionされない場合、ステータスは DELETE オペレーションを実行するerror場所に変更され、サブスクリプションの作成を再試行します。サブスクリプションリソースの "error"フィールドを表示して、サブスクリプションが正常に作成されなかった理由を確認できます。

4. サブスクリプション中の取り込み active
5. Subscription が の場合 active
   - HealthLake は、条件に一致するイベントをモニタリングします
   - 一致が発生すると、設定されたエンドポイントに通知が送信されます
6. エラー処理
   - HealthLake は 14 日間再試行し、それらのイベントの再試行を停止します。
7. 非アクティブ化
   - は、次の方法で非アクティブ化Subscriptionできます。

     終了日の設定 (自動非アクティブ化)

```
{
  "resourceType": "Subscription",
  "meta": {
    "profile": [
      "http://hl7.org/fhir/uv/subscriptions-backport/StructureDefinition/backport-
subscription"
    ]
  },
  "status": "requested",
  "end": "2026-07-31T05:38:17.2404292+00:00",
  "reason": "Test subscription for walkthrough",
  "criteria": "https://healthlake.<AWS_REGION>.amazonaws.com/datastore/
<datastoreId>/r4/SubscriptionTopic/<your topic id>",
  "_criteria": {
    "extension": [
      {
        "url": "http://hl7.org/fhir/uv/subscriptions-backport/StructureDefinition/
backport-filter-criteria",
        "valueString": "Encounter?subject=Patient/<patient id>"
      }
```

```
    ]
  },
  "channel": {
    "type": "event-bridge",
    "endpoint": "<your event bus arn>",
    "payload": "application/fhir+json",
    "_payload": {
      "extension": [
        {
          "url": "http://hl7.org/fhir/uv/subscriptions-backport/
StructureDefinition/backport-payload-content",
          "valueCode": "id-only"
        }
      ]
    }
  }
}
```

Subscription リソースの削除

```
DELETE https://<baseHealthLakeURL>/Subscription/<your subscription resource id>
```

を使用した FHIR サブスクリプションの作成 AWS HealthLake

次のガイドでは、 を使用して FHIR サブスクリプションを作成する方法を示します AWS HealthLake。

FHIR サブスクリプションを作成するには

1. SubscriptionTopic を作成します。

   サブスクリプショントピックリソースの例:

```
{
  "resourceType": "SubscriptionTopic",
  "url": "http://example.org/FHIR/SubscriptionTopic/encounter-create",
  "version": "1.0.0-fhir.r4b",
  "title": "encounter-create",
  "status": "unknown",
  "description": "Example topic for new encounters",
  "resourceTrigger": [
```

```
    {
      "description": "Encounter Create",
      "resource": "Encounter",
      "supportedInteraction": ["create", "update"]
    }
  ]
}
```

2. 通知エンドポイント (カスタムチャネル) を準備します。エンドポイントが通知を受信するようにするには、次の手順が必要です。

   REST フックを使用する場合

   - CM_CMK データストアを使用する場合は、KMS キーポリシーを信頼`events.amazonaws.com`します。
   - CM_CMK データストアを使用する場合は、 の値を使用して `EventBridgeApiDestinations` タグを KMS キーに追加する必要があります。 `true`
   - HealthLake は OAuth を使用して REST Hook エンドポイントを認証します。したがって、REST フックサブスクリプションを作成するときは、client-id、client-secret、および oAuth-endpoint-url を channel._type.extension[*] に渡す必要があります。

   CM_CMK データストアを使用する場合の KMS キーポリシーの例:

```
{
  "Sid": "AllowEventBridgeToUseKMSKey",
  "Effect": "Allow",
  "Principal": {
    "Service": ["events.amazonaws.com", "healthlake.amazonaws.com"]
  },
  "Action": ["kms:GenerateDataKey*", "kms:Decrypt", "kms:DescribeKey"],
  "Resource": "*"
}
```

   EventBridge を使用する場合

   - CM_CMK データストアを使用する場合は、KMS キーポリシーを信頼`events.amazonaws.com`します。
   - EventBridge リソースポリシーがサービスプリンシパル`healthlake.amazonaws.com`として信頼されていることを確認します。
   - CM_CMK を使用し、EventBridge がエンドポイントである場合は、データストア KMS キーと同じ KMS キーを使用して EventBridge バスを暗号化していることを確認します。

- EventBridge バスに、HealthLake によって生成されたイベントに一致するルールが少なくとも 1 つあることを確認します。

EventBridge チャネルバスのリソースポリシーの例:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "allowHealthlakeToPutEvents",
      "Effect": "Allow",
      "Principal": {
        "Service": "healthlake.amazonaws.com"
      },
      "Action": "events:PutEvents",
      "Resource": "arn:aws:healthlake:us-east-1:111122223333:event-bus/
FhirSubscriptions-bus"
    }
  ]
  }
```

HealthLake からイベントを受信するための EventBridge ルールイベントパターンの例:

```
{
    "detail-type": ["FHIR Subscription Notification"],
    "source": ["healthlake"]
}
```

> **Note**
>
> HealthLake は 2 つのソースをサポートしています。
>
> - “healthlake”: サブスクリプションの場合のみ。
> - “aws.healthlake”: HealthLake サービスイベントを受信するには。
>
> FHIR サブスクリプションイベントバスのルールを作成するときは、 をソース“healthlake”として使用します。

3. Subscription が作成してください。

以下を使用してサブスクリプションリソースを送信します。

- ステータス: "requested"
- 選択した ID SubscriptionTopic への参照
- フィルター条件。詳細については、「サポートされているフィルターの通知のフィルタリング」を参照してください。
- チャンネル設定

サブスクリプションペイロードの例

次のコード例は、サブスクリプションペイロードを作成する方法を示しています。

EventBridge

event-bridge チャネルとid-onlyペイロードタイプのサブスクリプション。

```
{
  "resourceType": "Subscription",
  "meta": {
    "profile": [
      "http://hl7.org/fhir/uv/subscriptions-backport/StructureDefinition/backport-
subscription"
    ]
  },
  "status": "requested",
  "end": "2026-07-31T05:38:17.2404292+00:00",
  "reason": "Test subscription for walkthrough",
  "criteria": "https://healthlake.<AWS_REGION>.amazonaws.com/datastore/<datastoreId/r4/
SubscriptionTopic/<your topic id>",
  "_criteria": {
    "extension": [
      {
        "url": "http://hl7.org/fhir/uv/subscriptions-backport/StructureDefinition/
backport-filter-criteria",
        "valueString": "Encounter?subject=Patient/<patient id>"
      }
    ]
  },
  "channel": {
    "type": "event-bridge",
    "endpoint": "arn:aws:healthlake:eu-west-2:111122223333:event-bus/FhirSubscriptions-
bus",
    "payload": "application/fhir+json",
```

```
    "_payload": {
      "extension": [
        {
          "url": "http://hl7.org/fhir/uv/subscriptions-backport/StructureDefinition/
backport-payload-content",
          "valueCode": "id-only"
        }
      ]
    }
  }
}
```

event-bridge エンドポイントとfull-resourceペイロードタイプのサブスクリプション。

```
{
  "resourceType": "Subscription",
  "meta": {
    "profile": [
      "http://hl7.org/fhir/uv/subscriptions-backport/StructureDefinition/backport-
subscription"
    ]
  },
  "status": "requested",
  "end": "2026-07-31T05:38:17.2404292+00:00",
  "reason": "Test subscription for walkthrough",
  "criteria": "https://healthlake.<AWS_REGION>.amazonaws.com/datastore/<datastoreId>/
r4/SubscriptionTopic/<your topic id>",
  "_criteria": {
    "extension": [
      {
        "url": "http://hl7.org/fhir/uv/subscriptions-backport/StructureDefinition/
backport-filter-criteria",
        "valueString": "Encounter?subject=Patient/<patient id>"
      }
    ]
  },
  "channel": {
    "type": "event-bridge",
    "endpoint": "<your event bus arn>",
    "payload": "application/fhir+json",
    "_payload": {
      "extension": [
        {
```

```
          "url": "http://hl7.org/fhir/uv/subscriptions-backport/StructureDefinition/
backport-payload-content",
          "valueCode": "full-resource"
        }
      ]
    }
  }
}
```

静止フック

rest-hook エンドポイントとid-onlyペイロードタイプのサブスクリプション。

```
{
  "resourceType": "Subscription",
  "meta": {
    "profile": [
      "http://hl7.org/fhir/uv/subscriptions-backport/StructureDefinition/backport-
subscription"
    ]
  },
  "status": "requested",
  "end": "2026-07-31T05:38:17.2404292+00:00",
  "reason": "Test subscription for walkthrough",
  "criteria": "https://healthlake.<AWS_REGION>.amazonaws.com/datastore/<datastoreId>/
r4/SubscriptionTopic/<your topic id>",
  "_criteria": {
    "extension": [
      {
        "url": "http://hl7.org/fhir/uv/subscriptions-backport/StructureDefinition/
backport-filter-criteria",
        "valueString": "Encounter?subject=Patient/<your patient id>"
      }
    ]
  },
  "channel": {
    "type": "rest-hook",
    "_type": {
      "extension": [
        {
          "url": "http://healthlake.amazonaws.com/channel-type-clientId",
          "valueString": "<CLIENT_ID>"
        },
        {
```

```
          "url": "http://healthlake.amazonaws.com/channel-type-clientSecret",
          "valueString": "<CLIENT_SECRET>"
        },
        {
          "url": "http://healthlake.amazonaws.com/channel-type-oauth-endpoint",
          "valueUri": "<OAUTH_ENDPOINT_URL>"
        }
      ]
    },
    "endpoint": "<YOUR_REST_HOOK_ENDPOINT>",
    "payload": "application/fhir+json",
    "_payload": {
      "extension": [
        {
          "url": "http://hl7.org/fhir/uv/subscriptions-backport/StructureDefinition/
backport-payload-content",
          "valueCode": "id-only"
        }
      ]
    }
  }
}
```

rest-hook チャネルとfull-resourceペイロードタイプのサブスクリプション。

```
{
  "resourceType": "Subscription",
  "meta": {
    "profile": [
      "http://hl7.org/fhir/uv/subscriptions-backport/StructureDefinition/backport-
subscription"
    ]
  },
  "status": "requested",
  "end": "2026-07-31T05:38:17.2404292+00:00",
  "reason": "Test subscription for walkthrough",
  "criteria": "https://healthlake.<AWS_REGION>.amazonaws.com/datastore/<datastoreId>/
r4/SubscriptionTopic/<your topic id>",
  "_criteria": {
    "extension": [
      {
        "url": "http://hl7.org/fhir/uv/subscriptions-backport/StructureDefinition/
backport-filter-criteria",
```

```
        "valueString": "Encounter?subject=Patient/test-patient-id"
      }
    ]
  },
  "channel": {
    "type": "rest-hook",
    "_type": {
      "extension": [
        {
          "url": "http://healthlake.amazonaws.com/channel-type-clientId",
          "valueString": "<CLIENT_ID>"
        },
        {
          "url": "http://healthlake.amazonaws.com/channel-type-clientSecret",
          "valueString": "<CLIENT_SECRET>"
        },
        {
          "url": "http://healthlake.amazonaws.com/channel-type-oauth-endpoint",
          "valueUri": "<OAUTH_ENDPOINT_URL>"
        }
      ]
    },
    "endpoint": "<YOUR_REST_HOOK_ENDPOINT>",
    "payload": "application/fhir+json",
    "_payload": {
      "extension": [
        {
          "url": "http://hl7.org/fhir/uv/subscriptions-backport/StructureDefinition/
backport-payload-content",
          "valueCode": "full-resource"
        }
      ]
    }
  }
}
```

通知ペイロードの例

サブスクリプションの作成中に、HealthLake は設定されたチャネルにハンドシェイクバンドルを送信して、サブスクリプション設定が成功したかどうかを確認します。次のペイロードは、ハンドシェイクバンドルの例です。

```
{
```

```
  "version": "0",
  "id": "<your-id>",
  "detail-type": "FHIR Subscription Notification",
  "source": "healthlake",
  "account": "436845984719",
  "time": "2025-09-04T23:43:50Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "subscriptionUrl": "https://healthlake.<AWS_REGION>.amazonaws.com/datastore/
<DS_ID>/r4/Subscription/<SUBSCRIPTION_ID>",
    "notificationBundlePayload": {
      "resourceType": "Bundle",
      "id": "<BUNDLE_ID>",
      "type": "history",
      "timestamp": "2025-09-04T23:43:50.341791934Z",
      "status": "requested",
      "entry": [
        {
          "fullUrl": "urn:uuid:<HANDSHAKE_RESOURCE_ID>",
          "resource": {
            "resourceType": "SubscriptionStatus",
            "id": "<HANDSHAKE_RESOURCE_ID>",
            "status": "requested",
            "type": "handshake",
            "eventsSinceSubscriptionStart": "0",
            "subscription": {
              "reference": "https://healthlake.<AWS_REGION>.amazonaws.com/datastore/
<DS_ID>/r4/Subscription/<SUBSCRIPTION_ID>"
            },
            "topic": "https://healthlake.<AWS_REGION>.amazonaws.com/datastore/<DS_ID>/
r4/SubscriptionTopic/<TOPIC_ID>"
          }
        }
      ]
    }
  }
}
```

ID のみの通知バンドルの例。

```
{
  "version": "0",
```

```
  "id": "<your-id>",
  "detail-type": "FHIR Subscription Notification",
  "source": "healthlake",
  "account": "436845984719",
  "time": "2025-09-05T00:18:43Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "subscriptionUrl": "https://healthlake.<AWS_REGION>.amazonaws.com/datastore/
<DS_ID>/r4/Subscription/<SUBSCRIPTION_ID>",
    "notificationBundlePayload": {
      "resourceType": "Bundle",
      "id": "c74ea02a-9c69-4e34-85d6-e72720189574",
      "type": "history",
      "timestamp": "2025-09-05T00:18:43.393688851Z",
      "status": "requested",
      "entry": [
        {
          "fullUrl": "urn:uuid:173135e3-3c80-4b90-a10a-e01a1420fdea",
          "resource": {
            "resourceType": "SubscriptionStatus",
            "id": "173135e3-3c80-4b90-a10a-e01a1420fdea",
            "status": "active",
            "type": "event-notification",
            "eventsSinceSubscriptionStart": "-1",
            "subscription": {
              "reference": "https://healthlake.<AWS_REGION>.amazonaws.com/datastore/
<DS_ID>/r4/Subscription/<SUBSCRIPTION_ID>"
            },
            "topic": "https://healthlake.<AWS_REGION>.amazonaws.com/datastore/<DS_ID>/
r4/SubscriptionTopic/<TOPIC_ID>",
            "notificationEvent": [
              {
                "eventNumber": "0",
                "timestamp": "2025-09-05T00:18:43.393775234Z",
                "focus": "Encounter/c5ae898f-bd96-44dd-a509-87fdbcf23b19",
                "additionalContext": "Encounter/c5ae898f-bd96-44dd-a509-87fdbcf23b19/
_history/1",
                "id": "8f4e9c1a-2b3d-4e5f-6a7b-8c9d0e1f2a3b"
              }
            ]
          }
        }
      ]
```

```
    }
  }
}
```

フルリソース通知バンドルの例。

```
{
  "version": "0",
  "id": "d142bed8-db3f-445f-c4db-a843ad84121a",
  "detail-type": "FHIR Subscription Notification",
  "source": "healthlake",
  "account": "436845984719",
  "time": "2025-09-05T00:18:43Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "subscriptionUrl": "https://healthlake.<AWS_REGION>.amazonaws.com/datastore/
<DS_ID>/r4/Subscription/<SUBSCRIPTION_ID>",
    "notificationBundlePayload": {
      "resourceType": "Bundle",
      "id": "3d42c70f-4fa9-4b1a-98a7-43c0d0441115",
      "type": "history",
      "timestamp": "2025-09-05T00:18:43.845821667Z",
      "status": "requested",
      "entry": [
        {
          "fullUrl": "urn:uuid:1d005a09-a15c-4010-9675-1e8043ce08a8",
          "resource": {
            "resourceType": "SubscriptionStatus",
            "id": "1d005a09-a15c-4010-9675-1e8043ce08a8",
            "status": "active",
            "type": "event-notification",
            "eventsSinceSubscriptionStart": "-1",
            "subscription": {
              "reference": "https://healthlake.<AWS_REGION>.amazonaws.com/datastore/
<DS_ID>/r4/Subscription/<SUBSCRIPTION_ID>"
            },
            "topic": "https://healthlake.<AWS_REGION>.amazonaws.com/datastore/<DS_ID>/
r4/SubscriptionTopic/<TOPIC_ID>",
            "notificationEvent": [
              {
```

```
                "eventNumber": "0",
                "timestamp": "2025-09-05T00:18:43.845970754Z",
                "focus": "Encounter/82776529-59a0-4d63-bedb-82f6726d65b5",
                "additionalContext": "Encounter/82776529-59a0-4d63-bedb-82f6726d65b5/
_history/1",
                "id": "7a8b9c0d-1e2f-3a4b-5c6d-7e8f9a0b1c2d"
              }
            ]
          }
        },
        {
          "fullUrl": "Encounter/82776529-59a0-4d63-bedb-82f6726d65b5",
          "resource": {
            "resourceType": "Encounter",
            "id": "82776529-59a0-4d63-bedb-82f6726d65b5",
            "status": "finished",
            "class": {
              "system": "http://terminology.hl7.org/CodeSystem/v3-ActCode",
              "code": "AMB",
              "display": "ambulatory"
            },
            "subject": {
              "reference": "Patient/test-patient-id"
            },
            "meta": {
              "lastUpdated": "2025-09-05T00:18:43.219652906Z",
              "versionId": "1"
            }
          },
          "request": {
            "method": "CREATE",
            "url": "Encounter/82776529-59a0-4d63-bedb-82f6726d65b5"
          }
        }
      ]
    }
  }
}
```

イベントのバージョニング

HealthLake はデフォルトで FHIR 履歴をサポートしています。

通知バンドルで受信したリソースのバージョンを確認するには:

- full-resource: フルリソースバンドルにはリソース全体が含まれているため、バージョンはバンドル内の各リソース`entry[*]`の に含まれます。
- id-only: バンドルにはリソース情報は含まれません。HealthLake には、`entry[0].notificationEvent[*].additionalContext`フィールドを通じて一致し、バンドルに含まれていたバージョンが含まれます。このフィールドは `<ResourceType>/<ResourceId>/_history/<Version Id>` の形式です。詳細については、id-only ペイロードの例の additionalContext フィールドを参照してください。

イベント重複検出

HealthLake の FHIR サブスクリプション機能は、少なくとも 1 つの配信を保証します。つまり、同じバンドルまたは別のバンドルのいずれかで、同じイベントが複数回受信される場合があります。重複を識別するために、HealthLake は の通知バンドル内のイベントごとに一意の ID を提供しますentry[0].notificationEvent[*].id。

この ID は、一致して配信されたイベントの特定のバージョンに固有です。たとえば、同じ Encounter が 2 回更新され、両方の更新がフィルター条件と一致した場合、同じ Encounter リファレンスを持つ 2 つの個別のイベントを受け取ります。これらは同じ を持ちますがnotificationEvent[*].focus、一意の を持ちますnotificationEvent[*].id。さらに、これらのイベントは別々のバンドルで送信することも、同じ通知バンドル内で送信することもできます。

を使用した FHIR サブスクリプションの検索 AWS HealthLake

Subscription および SubscriptionTopicリソースも検索可能です。HealthLake は、サブスクリプションリソースと SubscriptionTopic リソースのすべての一般的な検索パラメータをサポートしています。

さらに、以下のパラメータを使用して追加の検索機能をサポートしています。

サブスクリプション

検索パラメータ	説明	例
問い合わせ	R4 コア仕様の Subscription.contact フィールドで検索する	`Subscription?contact=phone`

検索パラメータ	説明	例
criteria	R4 コア仕様の Subscription.criteria フィールドで検索する	`Subscription?criteria=[baseUrl]/datastore/[datastoreId]/r4/SubscriptionTopic/[topicId]`
payload	R4 コア仕様の Subscription.channel.payload フィールドで検索する	`Subscription?payload=application/fhir+json`
ステータス	R4 コア仕様の Subscription.status フィールドで検索する	`Subscription?status=error`
型	R4 コア仕様の Subscription.channel.type フィールドで検索する	`Subscription?topic=event-bridge`
url	R4 コア仕様の Subscription.channel.endpoint フィールドで検索する	`Subscription?url=[Subscription.channel.endpoint]`
filter-criteria	URL "http://hl7.org/fhir/uv/subscriptions-backport/StructureDefinition/backport-filter-criteria" を文字列として条件拡張フィールドを検索する	`Subscription?filter-criteria=Encounter?`

検索パラメータ	説明	例
カスタムチャネル	URL が「http://hl7.org/fhir/uv/subscriptions-backport/StructureDefinition/backport-channel-type」のカスタムチャネルタイプ拡張フィールドをコーディングとして検索する サブスクリプションペイロードの例	`Subscription?custom-channel=[System]|[code]`
payload-type	ペイロードのフォーマット方法 (`id-only`または `full-resource`) を指定するペイロードタイプの拡張フィールドを検索します。	`Subscription?payload-type=full-resource`
トピック	トピックが追加された Subscription.criteria フィールドで検索する	`Subscription?topic=[topicId]`

SubscriptionTopic

検索パラメータ	説明	例
date	SubscriptionTopic リソースの日付フィールドで検索する	`SubscriptionTopic?date=[SubscriptionTopic.date]`
derived-or-self	SubscriptionTopic リソースの `url`または `derivedFrom` フィールドのいずれかを検索する	`SubscriptionTopic?derived-or-self=[SubscriptionTopic.url | SubscriptionTopic.derivedFrom]`

検索パラメータ	説明	例
識別子	SubscriptoinTopic.identifier フィールドで検索する	`SubscriptionTopic?identifier=[SubscriptionTopic.identifier]`
リソース	SubscriptionTopic.resourceTrigger.resource フィールドで検索する	`SubscriptionTopic?resource=Encounter`
ステータス	SubscriptionTopic のステータスを検索する	`SubscriptionTopic?status=unknown`
title	SubscriptionTopic のタイトルを検索する	`SubscriptionTopic?title=admission`
トリガーの説明	SubscriptionTopic.resourceTrigger.description で検索する	`SubscriptionTopic.trigger-description=resource moving to state 'in-progress'`
url	URL で SubscriptionTopic を検索する	`SubscriptionTopic?url=[SubscriptionTopic.url]`
バージョン	SubscriptionTopic のバージョンを検索する	`SubscriptionTopic?version=1`

を使用した通知のフィルタリング AWS HealthLake

`Subscription` 基準の標準 FHIR 検索パラメータを使用して通知を絞り込みます。

サポートされているフィルター

次の表は、HealthLake がサブスクリプションでサポートするサポートされているフィルター条件のリストを示しています。

検索パラメータタイプ	FHIR データ型	Modifiers (修飾子)	サポートされているプレフィックス
String	string HumanName Address	完全、欠落を含む	
トークン	boolean コード string コーディング CodeableConcept 識別子 ContactPoint id	なし、テキスト、欠落	
Number	integer decimal positiveInt unsignedInt	missing	「eq」、「ne」、「gt」、「lt」、「ge」、「le」、「sa」、「eb」、「ap」
日付	date dateTime 即時 Period [Timing] (タイミング)		「eq」、「ne」、「gt」、「lt」、「ge」、「le」、「sa」、「eb」、「ap」

検索パラメータタイプ	FHIR データ型	Modifiers (修飾子)	サポートされているプレフィックス
数量	数量 金額 Range SimpleQuantity		「eq」、「ne」、「gt」、「lt」、「ge」、「le」、「sa」、「eb」、「ap」
参照資料	リファレンス	欠落、識別子、タイプ	
[URI]	uri url canonical uid oid	missing	

サポートされるフィルターの例

次の表は、HealthLake がサブスクリプションでサポートするサポートされているフィルター条件のサンプルを示しています。

目的	フィルター条件	説明
患者固有の観察	`Observation?patient=Patient/[id]&status=final`	特定の患者の観察が確定されたときに通知を受け取る
患者固有の観察	`Patient?birthdate=gt2021`	2021 年以降に生まれる患者が登録または更新されたときに通知を受け取る

目的	フィルター条件	説明
患者固有の観察	`Condition?code=http://snomed.info/sct|39065001`	特定の SNOMED コードを持つ条件の通知を取得する
患者固有の観察	`Observation?code=http://loinc.org|8480-6&value-quantity=gt160`	収縮期血圧の高い測定値の通知を受け取る

を使用した FHIR データのインポート AWS HealthLake

HealthLake データストアを作成したら、次のステップとして Amazon Simple Storage Service (S3) バケットからファイルをインポートします。FHIR インポートジョブは AWS マネジメントコンソール、、 AWS CLI、または AWS SDKs を使用して開始できます。ネイティブ AWS HealthLake アクションを使用して、FHIR インポートジョブを開始、説明、一覧表示します。

> [重要]
>
> HealthLake は、医療データ交換の FHIR R4 仕様をサポートしています。必要に応じて、AWS HealthLake パートナーと協力して、インポート前にヘルスデータを FHIR R4 形式に変換できます。

FHIR インポートジョブを開始するときは、Amazon S3 バケット入力の場所、Amazon S3 バケット出力の場所 (ジョブ処理結果用) 、Amazon S3 バケットへの HealthLake アクセスを許可する IAM ロール、およびカスタマー所有またはカスタマー AWS 所有の AWS Key Management Service キーを指定します。詳細については、「インポートジョブのアクセス許可の設定」を参照してください。

> Note
>
> インポートジョブをキューに入れることができます。非同期インポートジョブは、FIFO (先入れ先出し) 方式で処理されます。インポートジョブを開始するのと同じ方法でジョブをキューに入れることができます。1 つが進行中の場合は、単にキューに入れられます。インポートジョブの進行中に、FHIR リソースを作成、読み取り、更新、または削除できます。

HealthLake は、FHIR インポートジョブごとに `manifest.json` ファイルを生成します。ファイルは、FHIR インポートジョブの成功と失敗の両方を記述します。HealthLake は、FHIR インポートジョブの開始時に指定された Amazon S3 バケットに`manifest.json`ファイルを出力します。ログファイルは、 `SUCCESS`と という名前の 2 つのフォルダに編成されています`FAILURE`。各`manifest.json`ファイルの詳細が提供されるため、失敗したインポートジョブのトラブルシューティングの最初のステップとして ファイルを使用します。

```
{
    "inputDataConfig": {
        "s3Uri": "s3://amzn-s3-demo-source-bucket/healthlake-input/invalidInput/"
    },
```

```
    "outputDataConfig": {
        "s3Uri": "s3://amzn-s3-demo-logging-bucket/32839038a2f47f17c2fe0f53f0c3a0ba-
FHIR_IMPORT-19dd7bb7bcc8ee12a09bf6d322744a3d/",
        "encryptionKeyID": "arn:aws:kms:us-west-2:123456789012:key/fbbbfee3-20b3-42a5-
a99d-c48c655ed545"
    },
    "successOutput": {
        "successOutputS3Uri": "s3://amzn-s3-demo-logging-
bucket/32839038a2f47f17c2fe0f53f0c3a0ba-FHIR_IMPORT-19dd7bb7bcc8ee12a09bf6d322744a3d/
SUCCESS/"
    },
    "failureOutput": {
        "failureOutputS3Uri": "s3://amzn-s3-demo-logging-
bucket/32839038a2f47f17c2fe0f53f0c3a0ba-FHIR_IMPORT-19dd7bb7bcc8ee12a09bf6d322744a3d/
FAILURE/"
    },
    "numberOfScannedFiles": 1,
    "numberOfFilesImported": 1,
    "sizeOfScannedFilesInMB": 0.023627,
    "sizeOfDataImportedSuccessfullyInMB": 0.011232,
    "numberOfResourcesScanned": 9,
    "numberOfResourcesImportedSuccessfully": 4,
    "numberOfResourcesWithCustomerError": 5,
    "numberOfResourcesWithServerError": 0
}
```

インポートの検証レベルの設定

FHIR インポートジョブを開始するときに、オプションValidationLevelで各リソースに適用するを指定できます。 AWS HealthLake は現在、次の検証レベルをサポートしています。

- strict: リソースは、リソースのプロファイル要素、またはプロファイルが存在しない場合は R4 仕様に従って検証されます。これはデフォルトの検証レベルです AWS HealthLake。
- structure-only: リソースは R4 に対して検証され、参照されるプロファイルは無視されます。
- minimal: リソースは、特定の R4 ルールを無視して、最小限検証されます。検索/分析に必要な構造チェックに失敗したリソースは、監査の警告を含むように更新されます。

minimal 検証レベルを使用してインポートする場合、 という名前のフォルダに追加のログファイルが生成されますSUCCESS_WITH_SEARCH_VALIDATION_FAILURES。このフォルダのログファイル内のリソースは、検索関連の検証チェックに失敗したにもかかわらず、データストアに取り込ま

れました。これは、FHIR リソースの特定の側面が FHIR に従って無効であり、不正な形式のフィールドが検索できない可能性があることを意味します。これらのリソースには、その障害を説明する `extension` が追加されます。

トピック

FHIR インポートジョブの開始

を使用して`StartFHIRImportJob`、HealthLake データストアへの FHIR インポートジョブを開始します。以下のメニューでは、 の手順 AWS マネジメントコンソール と、 AWS CLI および AWS SDKs。詳細については、「AWS HealthLake API リファレンス」の「`StartFHIRImportJob`」を参照してください。

> [重要]
>
> HealthLake は、医療データ交換の FHIR R4 仕様をサポートしています。必要に応じて、AWS HealthLake パートナーと協力して、インポート前にヘルスデータを FHIR R4 形式に変換できます。

FHIR インポートジョブを開始するには

アクセス設定に基づいてメニューを選択します AWS HealthLake。

AWS CLI および SDKs

CLI

AWS CLI

FHIR インポートジョブを開始するには

次の`start-fhir-import-job`例は、 AWS HealthLake を使用して FHIR インポートジョブを開始する方法を示しています。

```
aws healthlake start-fhir-import-job \
```

```
    --input-data-config S3Uri="s3://(Bucket Name)/(Prefix Name)/" \
    --job-output-data-config '{"S3Configuration": {"S3Uri":"s3://(Bucket Name)/
(Prefix Name)/","KmsKeyId":"arn:aws:kms:us-east-1:012345678910:key/d330e7fc-
b56c-4216-a250-f4c43ef46e83"}}' \
    --datastore-id (Data store ID) \
    --data-access-role-arn "arn:aws:iam::(AWS Account ID):role/(Role Name)"
```

出力:

```
{
    "DatastoreId": "(Data store ID)",
    "JobStatus": "SUBMITTED",
    "JobId": "c145fbb27b192af392f8ce6e7838e34f"
}
```

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

Python

SDK for Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
 client.

        :return: An instance of HealthLakeWrapper initialized with the default
 HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def start_fhir_import_job(
        self,
        job_name: str,
        datastore_id: str,
        input_s3_uri: str,
        job_output_s3_uri: str,
        kms_key_id: str,
```

```
        data_access_role_arn: str,
    ) -> dict[str, str]:
        """
        Starts a HealthLake import job.
        :param job_name: The import job name.
        :param datastore_id: The data store ID.
        :param input_s3_uri: The input S3 URI.
        :param job_output_s3_uri: The job output S3 URI.
        :param kms_key_id: The KMS key ID associated with the output S3 bucket.
        :param data_access_role_arn: The data access role ARN.
        :return: The import job.
        """
        try:
            response = self.health_lake_client.start_fhir_import_job(
                JobName=job_name,
                InputDataConfig={"S3Uri": input_s3_uri},
                JobOutputDataConfig={
                    "S3Configuration": {
                        "S3Uri": job_output_s3_uri,
                        "KmsKeyId": kms_key_id,
                    }
                },
                DataAccessRoleArn=data_access_role_arn,
                DatastoreId=datastore_id,
            )
            return response
        except ClientError as err:
            logger.exception(
                "Couldn't start import job. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「StartFHIRImportJob」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    TRY.
        " iv_job_name = 'MyImportJob'
        " iv_input_s3_uri = 's3://my-bucket/import/data.ndjson'
        " iv_job_output_s3_uri = 's3://my-bucket/import/output/'
        " iv_kms_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012'
        " iv_data_access_role_arn = 'arn:aws:iam::123456789012:role/
HealthLakeImportRole'
        oo_result = lo_hll->startfhirimportjob(
          iv_jobname = iv_job_name
          io_inputdataconfig = NEW /aws1/cl_hllinputdataconfig( iv_s3uri =
 iv_input_s3_uri )
          io_joboutputdataconfig = NEW /aws1/cl_hlloutputdataconfig(
            io_s3configuration = NEW /aws1/cl_hlls3configuration(
              iv_s3uri = iv_job_output_s3_uri
              iv_kmskeyid = iv_kms_key_id
            )
          )
          iv_dataaccessrolearn = iv_data_access_role_arn
          iv_datastoreid = iv_datastore_id
        ).
        DATA(lv_job_id) = oo_result->get_jobid( ).
        MESSAGE |Import job started with ID { lv_job_id }.| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllthrottlingex INTO DATA(lo_throttling_ex).
        lv_error = |Throttling error: { lo_throttling_ex->av_err_code }-
{ lo_throttling_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
```

```
        RAISE EXCEPTION lo_throttling_ex.
      CATCH /aws1/cx_hllaccessdeniedex INTO DATA(lo_access_ex).
        lv_error = |Access denied: { lo_access_ex->av_err_code }-{ lo_access_ex-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_access_ex.
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスのStartFHIRImportJob」を参照してください。

> 可用性の例
>
> 必要なものが見つからなかった場合。このページの右側サイドバーにあるフィードバックを提供するリンクを使用して、コード例をリクエストします。

AWS コンソール

1. HealthLake コンソールのデータストアページにサインインします。
2. データストアを選択します。
3. [インポート] を選択します。

   インポートページが開きます。
4. 入力データセクションで、次の情報を入力します。
   - Amazon S3 の入力データの場所
5. 出力ファイルのインポート セクションで、次の情報を入力します。
   - Amazon S3 で出力ファイルの場所をインポートする
   - 出力ファイルの暗号化をインポートする
6. アクセス許可セクションで、既存の IAM 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、IAM ロールの作成 を選択します。
7. [データをインポート] を選択します。

> **Note**
>
> インポート中に、ページ上部のバナーでジョブ ID をコピーを選択します。を使用してインポートジョブのプロパティを`JobID`リクエストできます AWS CLI。詳細については、「FHIR インポートジョブのプロパティの取得」を参照してください。

FHIR インポートジョブのプロパティの取得

`DescribeFHIRImportJob` を使用して FHIR インポートジョブのプロパティを取得します。以下のメニューでは、 の手順 AWS マネジメントコンソール と、 AWS CLI および AWS SDKs。詳細については、「AWS HealthLake API リファレンス」の「`DescribeFHIRImportJob`」を参照してください。

FHIR インポートジョブのプロパティを取得するには

アクセス設定に基づいてメニューを選択します AWS HealthLake。

AWS CLI および SDKs

CLI

AWS CLI

FHIR インポートジョブを記述するには

次の`describe-fhir-import-job`例は、 AWS HealthLake を使用して FHIR インポートジョブのプロパティを学習する方法を示しています。

```
aws healthlake describe-fhir-import-job \
    --datastore-id (Data store ID) \
    --job-id c145fbb27b192af392f8ce6e7838e34f
```

出力:

```
{
    "ImportJobProperties": {
    "InputDataConfig": {
        "S3Uri": "s3://(Bucket Name)/(Prefix Name)/"
```

```
        { "arrayitem2": 2 }
    },
    "DataAccessRoleArn": "arn:aws:iam::(AWS Account ID):role/(Role Name)",
    "JobStatus": "COMPLETED",
    "JobId": "c145fbb27b192af392f8ce6e7838e34f",
    "SubmitTime": 1606272542.161,
    "EndTime": 1606272609.497,
    "DatastoreId": "(Data store ID)"
    }
}
```

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

Python

SDK for Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
 client.

        :return: An instance of HealthLakeWrapper initialized with the default
 HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def describe_fhir_import_job(
        self, datastore_id: str, job_id: str
    ) -> dict[str, any]:
        """
        Describes a HealthLake import job.
        :param datastore_id: The data store ID.
        :param job_id: The import job ID.
        :return: The import job description.
        """
        try:
            response = self.health_lake_client.describe_fhir_import_job(
```

```
                DatastoreId=datastore_id, JobId=job_id
            )
            return response["ImportJobProperties"]
        except ClientError as err:
            logger.exception(
                "Couldn't describe import job with ID %s. Here's why %s",
                job_id,
                err.response["Error"]["Message"],
            )
            raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「DescribeFHIRImportJob」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
TRY.
    " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
    " iv_job_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
    oo_result = lo_hll->describefhirimportjob(
      iv_datastoreid = iv_datastore_id
      iv_jobid = iv_job_id
    ).
    DATA(lo_import_job_properties) = oo_result->get_importjobproperties( ).
    IF lo_import_job_properties IS BOUND.
```

```
          DATA(lv_job_status) = lo_import_job_properties->get_jobstatus( ).
          MESSAGE |Import job status: { lv_job_status }.| TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        DATA(lv_error) = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        lv_error = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスの「DescribeFHIRImportJob」を参照してください。

> **可用性の例**
>
> 必要なものが見つからなかった場合。このページの右側サイドバーにあるフィードバックを提供するリンクを使用して、コード例をリクエストします。

AWS コンソール

> **Note**
>
> FHIR インポートジョブ情報は HealthLake コンソールでは使用できません。代わりに、AWS CLI で を使用してDescribeFHIRImportJob、 などのインポートジョブのプロパティをリクエストしますJobStatus。詳細については、このページ AWS CLI の例を参照してください。

FHIR インポートジョブの一覧表示

を使用してListFHIRImportJobs、アクティブな HealthLake データストアの FHIR インポートジョブを一覧表示します。次のメニューでは、 の手順と、 AWS マネジメントコンソール および

AWS CLI SDK の AWS コード例を示します。 SDKs 詳細については、「AWS HealthLake API リファレンス」の「ListFHIRImportJobs」を参照してください。

FHIR インポートジョブを一覧表示するには

アクセス設定に基づいてメニューを選択します AWS HealthLake。

AWS CLI および SDKs

CLI

AWS CLI

すべての FHIR インポートジョブを一覧表示するには

次の `list-fhir-import-jobs` の例は、コマンドを使用して、アカウントに関連付けられているすべてのインポートジョブのリストを表示する方法を示しています。

```
aws healthlake list-fhir-import-jobs \
    --datastore-id (Data store ID) \
    --submitted-before (DATE like 2024-10-13T19:00:00Z) \
    --submitted-after (DATE like 2020-10-13T19:00:00Z ) \
    --job-name "FHIR-IMPORT" \
    --job-status SUBMITTED  \
    -max-results (Integer between 1 and 500)
```

出力:

```
{
    "ImportJobPropertiesList": [
        {
            "JobId": "c0fddbf76f238297632d4aebdbfc9ddf",
            "JobStatus": "COMPLETED",
            "SubmitTime": "2024-11-20T10:08:46.813000-05:00",
            "EndTime": "2024-11-20T10:10:09.093000-05:00",
            "DatastoreId": "(Data store ID)",
            "InputDataConfig": {
                "S3Uri": "s3://(Bucket Name)/(Prefix Name)/"
            },
            "JobOutputDataConfig": {
                "S3Configuration": {
```

```
                    "S3Uri": "s3://(Bucket Name)/
import/6407b9ae4c2def3cb6f1a46a0c599ec0-FHIR_IMPORT-
c0fddbf76f238297632d4aebdbfc9ddf/",
                    "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/b7f645cb-
e564-4981-8672-9e012d1ff1a0"
                }
            },
            "JobProgressReport": {
                "TotalNumberOfScannedFiles": 1,
                "TotalSizeOfScannedFilesInMB": 0.001798,
                "TotalNumberOfImportedFiles": 1,
                "TotalNumberOfResourcesScanned": 1,
                "TotalNumberOfResourcesImported": 1,
                "TotalNumberOfResourcesWithCustomerError": 0,
                "TotalNumberOfFilesReadWithCustomerError": 0,
                "Throughput": 0.0
            },
            "DataAccessRoleArn": "arn:aws:iam::(AWS Account ID):role/(Role Name)"
        }
    ]
}
```

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

Python

SDK for Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
 client.

        :return: An instance of HealthLakeWrapper initialized with the default
 HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)
```

```
    def list_fhir_import_jobs(
        self,
        datastore_id: str,
        job_name: str = None,
        job_status: str = None,
        submitted_before: datetime = None,
        submitted_after: datetime = None,
    ) -> list[dict[str, any]]:
        """
        Lists HealthLake import jobs satisfying the conditions.
        :param datastore_id: The data store ID.
        :param job_name: The import job name.
        :param job_status: The import job status.
        :param submitted_before: The import job submitted before the specified
date.
        :param submitted_after: The import job submitted after the specified
date.
        :return: A list of import jobs.
        """
        try:
            parameters = {"DatastoreId": datastore_id}
            if job_name is not None:
                parameters["JobName"] = job_name
            if job_status is not None:
                parameters["JobStatus"] = job_status
            if submitted_before is not None:
                parameters["SubmittedBefore"] = submitted_before
            if submitted_after is not None:
                parameters["SubmittedAfter"] = submitted_after
            next_token = None
            jobs = []
            # Loop through paginated results.
            while True:
                if next_token is not None:
                    parameters["NextToken"] = next_token
                response =
self.health_lake_client.list_fhir_import_jobs(**parameters)
                jobs.extend(response["ImportJobPropertiesList"])
                if "NextToken" in response:
                    next_token = response["NextToken"]
                else:
                    break
            return jobs
        except ClientError as err:
```

```
            logger.exception(
                "Couldn't list import jobs. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「ListFHIRImportJobs」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        IF iv_submitted_after IS NOT INITIAL.
          oo_result = lo_hll->listfhirimportjobs(
            iv_datastoreid = iv_datastore_id
            iv_submittedafter = iv_submitted_after
          ).
        ELSE.
          oo_result = lo_hll->listfhirimportjobs(
            iv_datastoreid = iv_datastore_id
          ).
        ENDIF.
        DATA(lt_import_jobs) = oo_result->get_importjobpropertieslist( ).
        DATA(lv_job_count) = lines( lt_import_jobs ).
```

```
        MESSAGE |Found { lv_job_count } import job(s).| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスのListFHIRImportJobs」を参照してください。

可用性の例

必要なものが見つからなかった場合。このページの右側サイドバーにあるフィードバックを提供するリンクを使用して、コード例をリクエストします。

AWS コンソール

Note

FHIR インポートジョブ情報は HealthLake コンソールでは使用できません。代わりに、 とを使用してすべての FHIR インポートジョブListFHIRImportJobsを一覧表示します AWS CLI 。詳細については、このページ AWS CLI の例を参照してください。

での FHIR リソースの管理 AWS HealthLake

FHIR R4 RESTful API インタラクションを使用して、HealthLake データストア内の FHIR リソースを管理します。以下のセクションでは、FHIR リソース管理で使用できる HealthLake がサポートするすべての FHIR R4 RESTful API インタラクションについて説明します。HealthLake データストアの機能およびサポートされる FHIR 仕様の部分については、「」を参照してくださいの FHIR R4 機能ステートメント AWS HealthLake。

> **Note**
>
> この章に記載されている FHIR インタラクションは、医療データ交換のための HL7 FHIR R4 標準に準拠して構築されています。これらは HL7 FHIR サービスの表現であるため、 AWS CLI および AWS SDKsでは提供されません。詳細については、FHIR R4 RESTful API ドキュメントの「RESTful API」を参照してください。 R4 RESTful

次の表に、 でサポートされている FHIR R4 インタラクションを示します AWS HealthLake。HealthLake でサポートされている FHIR リソースタイプについては、「」を参照してくださいリソースタイプ:。

でサポートされている FHIR R4 インタラクション AWS HealthLake

相互作用	説明
システム全体のインタラクション	
`capabilities`	システムの機能ステートメントを取得します。「の FHIR R4 機能ステートメント AWS HealthLake」を参照してください。
`batch`	1 回の操作で一連のリソースを更新、作成、または削除します。「FHIR リソースのバンドル」を参照してください。
タイプレベルのインタラクション	
`create`	サーバーが割り当てた ID を使用して新しいリソースを作成します。「FHIR リソースの作成」を参照してください。

相互作用	説明
search	一部のフィルター条件に基づいてリソースタイプを検索します。「FHIR リソースの検索」を参照してください。
history	特定のリソースタイプの変更履歴を取得します。「FHIR リソース履歴の読み取り」を参照してください。
インスタンスレベルのインタラクション	
read	リソースの現在の状態を読み取ります。「FHIR リソースの読み取り」を参照してください。
history	特定のリソースの変更履歴を読み取ります。「FHIR リソース履歴の読み取り」を参照してください。
vread	リソースの特定のバージョンの状態を読み取ります。「バージョン固有の FHIR リソース履歴の読み取り」を参照してください。
update	ID でリソースを更新します (新しい場合は作成します)。「FHIR リソースの更新」を参照してください。
delete	リソースを削除します。「FHIR リソースの削除」を参照してください。

トピック

- FHIR リソースの作成
- FHIR リソースの読み取り
- FHIR リソース履歴の読み取り
- FHIR リソースの更新
- PATCH オペレーションによるリソースの変更
- FHIR リソースのバンドル
- FHIR リソースの削除
- べき等性と同時実行性

FHIR リソースの作成

FHIR createインタラクションは、HealthLake データストアに新しい FHIR リソースを作成します。詳細については、FHIR R4 RESTful API ドキュメントcreateの「」を参照してください。 R4 RESTful

FHIR リソースを作成するには

1. HealthLake regionと datastoreIdの値を収集します。詳細については、「データストアのプロパティの取得」を参照してください。
2. Resource 作成する FHIR のタイプを決定します。詳細については、「リソースタイプ:」を参照してください。
3. HealthLake regionと の収集された値を使用して、リクエストの URL を作成しますdatastoreId。作成する FHIR Resourceタイプも含めます。次の例の URL パス全体を表示するには、コピーボタンにスクロールします。

```
POST https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Resource
```

4. リクエストの JSON 本文を作成し、新しいリソースの FHIR データを指定します。この手順では、FHIR Patientリソースを使用しているため、ファイルを として保存しますcreate-patient.json。

```
{
    "resourceType": "Patient",
    "identifier": [
        {
            "system": "urn:oid:1.2.36.146.595.217.0.1",
            "value": "12345"
        }
    ],
    "name": [
        {
            "family": "Silva",
            "given": [
                "Ana",
                "Carolina"
            ]
        }
    ],
    "gender": "female",
```

```
    "birthDate": "1992-02-10"
}
```

5. リクエストを送信します。FHIR createインタラクションは、AWS 署名バージョン 4 または SMART on FHIR 認可のPOSTリクエストを使用します。次の例では、curl または HealthLake コンソールを使用して HealthLake に FHIR Patientリソースを作成します。例全体を表示するには、コピーボタンをスクロールします。

   SigV4

   SigV4 認可

   ```
   curl --request POST \
     'https://healthlake.region.amazonaws.com/datastore/datastore-id/r4/Patient' \
     --aws-sigv4 'aws:amz:region:healthlake' \
     --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
     --header "x-amz-security-token:$AWS_SESSION_TOKEN" \
     --header 'Accept: application/json' \
     --data @create-patient.json
   ```

   SMART on FHIR

   IdentityProviderConfiguration データ型の FHIR 認可の SMART の例。

   ```
   {
       "AuthorizationStrategy": "SMART_ON_FHIR",
       "FineGrainedAuthorizationEnabled": true,
       "IdpLambdaArn": "arn:aws:lambda:your-region:your-account-id:function:your-
   lambda-name",
       "Metadata": "{\"issuer\":\"https://ehr.example.com\", \"jwks_uri\":
   \"https://ehr.example.com/.well-known/jwks.json\",\"authorization_endpoint
   \":\"https://ehr.example.com/auth/authorize\",\"token_endpoint\":\"https://
   ehr.token.com/auth/token\",\"token_endpoint_auth_methods_supported\":
   [\"client_secret_basic\",\"foo\"],\"grant_types_supported\":[\"client_credential
   \",\"foo\"],\"registration_endpoint\":\"https://ehr.example.com/auth/
   register\",\"scopes_supported\":[\"openId\",\"profile\",\"launch\"],
   \"response_types_supported\":[\"code\"],\"management_endpoint\":\"https://
   ehr.example.com/user/manage\",\"introspection_endpoint\":\"https://
   ehr.example.com/user/introspect\",\"revocation_endpoint\":\"https://
   ehr.example.com/user/revoke\",\"code_challenge_methods_supported\":[\"S256\"],
   \"capabilities\":[\"launch-ehr\",\"sso-openid-connect\",\"client-public\",
   \"permission-v2\"]}"
   ```

```
}
```

発信者は認可 Lambda でアクセス許可を割り当てることができます。詳細については、「OAuth 2.0 スコープ」を参照してください。

AWS Console

> **Note**
>
> HealthLake コンソールは AWS SigV4 認可のみをサポートします。

1. HealthLake コンソールのクエリの実行ページにサインインします。
2. クエリ設定セクションで、次の選択を行います。
   - データストア ID — データストア ID を選択してクエリ文字列を生成します。
   - クエリタイプ — を選択しますCreate。
   - リソースタイプ — 作成する FHIR リソースタイプを選択します。
   - リクエスト本文 — リクエストの JSON 本文を作成し、新しいリソースの FHIR データを指定します。
3. [Run query] (クエリの実行) を選択します。

リソース作成の検証レベルの設定

FHIR リソースを作成するときに、オプションで HTTP `x-amzn-healthlake-fhir-validation-level` ヘッダーを指定して、リソースの検証レベルを設定できます。 AWS HealthLake は現在、次の検証レベルをサポートしています。

- `strict`: リソースは、リソースのプロファイル要素、またはプロファイルが存在しない場合は R4 仕様に従って検証されます。これはデフォルトの検証レベルです AWS HealthLake。
- `structure-only`: リソースは R4 に対して検証され、参照されるプロファイルは無視されます。
- `minimal`: リソースは、特定の R4 ルールを無視して、最小限検証されます。検索/分析に必要な構造チェックに失敗したリソースは、監査の警告を含むように更新されます。

最小限の検証レベルで作成されたリソースは、検索インデックス作成に必要な検証に失敗しても、Datastore に取り込むことができます。この場合、リソースが更新されて Healthlake 固有の拡張機能が含まれ、その障害が記録されます。

```
{
    "url": "http://healthlake.amazonaws.com/fhir/StructureDefinition/validation-issue",
    "valueString": "{\"resourceType\":\"OperationOutcome\",\"issue\":[{\"severity\":
\"error\",\"code\":\"processing\",\"details\":{\"text\":\"FHIR resource in payload
 failed FHIR validation rules.\"},\"diagnostics\":\"FHIR resource in payload failed
 FHIR validation rules.\"}]}"
}
```

さらに、次の HTTP レスポンスヘッダーが「true」の値に含まれます。

```
x-amzn-healthlake-validation-issues : true
```

> **Note**
>
> R4 仕様に従って不正な形式を取り込んだデータは、これらのエラーが存在する場合、期待どおりに検索できない場合があります。

FHIR リソースの読み取り

FHIR `read`インタラクションは、HealthLake データストア内のリソースの現在の状態を読み取ります。詳細については、FHIR R4 RESTful API ドキュメント`read`の「」を参照してください。 R4 RESTful

FHIR リソースを読み取るには

1. HealthLake `region`と `datastoreId`の値を収集します。詳細については、「データストアのプロパティの取得」を参照してください。
2. 関連するid値`Resource`を読み取って収集する FHIR のタイプを決定します。詳細については、「リソースタイプ:」を参照してください。
3. HealthLake `region`と の収集された値を使用して、リクエストの URL を作成しますdatastoreId。FHIR `Resource`タイプと関連する も含めますid。次の例の URL パス全体を表示するには、コピーボタンをスクロールします。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Resource/id
```

4. リクエストを送信します。FHIR readインタラクションは、AWS 署名バージョン 4 または SMART on FHIR 認可のいずれかのGETリクエストを使用します。次の のcurl例では、HealthLake の FHIR Patientリソースの現在の状態を読み取ります。例全体を表示するには、コピーボタンをスクロールします。

   SigV4

   SigV4 認可

   ```
   curl --request GET \
     'https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id'
    \
     --aws-sigv4 'aws:amz:region:healthlake' \
     --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
     --header "x-amz-security-token:$AWS_SESSION_TOKEN" \
     --header 'Accept: application/json'
   ```

   SMART on FHIR

   IdentityProviderConfiguration データ型の FHIR 認可の SMART の例。

   ```
   {
       "AuthorizationStrategy": "SMART_ON_FHIR",
       "FineGrainedAuthorizationEnabled": true,
       "IdpLambdaArn": "arn:aws:lambda:your-region:your-account-id:function:your-
   lambda-name",
       "Metadata": "{\"issuer\":\"https://ehr.example.com\", \"jwks_uri\":
   \"https://ehr.example.com/.well-known/jwks.json\",\"authorization_endpoint
   \":\"https://ehr.example.com/auth/authorize\",\"token_endpoint\":\"https://
   ehr.token.com/auth/token\",\"token_endpoint_auth_methods_supported\":
   [\"client_secret_basic\",\"foo\"],\"grant_types_supported\":[\"client_credential
   \",\"foo\"],\"registration_endpoint\":\"https://ehr.example.com/auth/
   register\",\"scopes_supported\":[\"openId\",\"profile\",\"launch\"],
   \"response_types_supported\":[\"code\"],\"management_endpoint\":\"https://
   ehr.example.com/user/manage\",\"introspection_endpoint\":\"https://
   ehr.example.com/user/introspect\",\"revocation_endpoint\":\"https://
   ehr.example.com/user/revoke\",\"code_challenge_methods_supported\":[\"S256\"],
   \"capabilities\":[\"launch-ehr\",\"sso-openid-connect\",\"client-public\",
   \"permission-v2\"]}"
   ```

```
}
```

発信者は認可 Lambda でアクセス許可を割り当てることができます。詳細については、「OAuth 2.0 スコープ」を参照してください。

AWS Console

1. HealthLake コンソールのクエリの実行ページにサインインします。
2. クエリ設定セクションで、次の選択を行います。
   - データストア ID — データストア ID を選択してクエリ文字列を生成します。
   - クエリタイプ — を選択しますRead。
   - リソースタイプ — 読み取る FHIR リソースタイプを選択します。
   - リソース ID — FHIR リソース ID を入力します。
3. [Run query] (クエリの実行) を選択します。

FHIR リソース履歴の読み取り

FHIR historyインタラクションは、HealthLake データストア内の特定の FHIR リソースの履歴を取得します。このインタラクションを使用して、FHIR リソースの内容が時間の経過とともにどのように変化したかを判断できます。また、監査ログと連携して、変更前後のリソースの状態を確認するのにも役立ちます。FHIR インタラクション create、update、および deleteでは、リソースの履歴バージョンが保存されます。詳細については、FHIR R4 RESTful API ドキュメントhistoryの「」を参照してください。 R4 RESTful

> **Note**
>
> 特定の FHIR リソースタイプhistoryに対して をオプトアウトできます。オプトアウトするには、 を使用してケースを作成しますAWS Support Center Console。ケースを作成するには、 にログイン AWS アカウント し、ケースの作成を選択します。

FHIR リソース履歴を読み取るには

1. HealthLake regionと datastoreIdの値を収集します。詳細については、「データストアのプロパティの取得」を参照してください。
2. 関連するid値Resourceを読み取って収集する FHIR のタイプを決定します。詳細については、「リソースタイプ:」を参照してください。
3. HealthLake regionと の収集された値を使用して、リクエストの URL を作成しますdatastoreId。また、FHIR Resourceタイプ、関連付けられた id、およびオプションの検索パラメータを含めます。次の例の URL パス全体を表示するには、コピーボタンをスクロールします。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Resource/id/
_history{?[parameters]}
```

HealthLake が FHIR **history**インタラクションでサポートする検索パラメータ

パラメータ	説明
_count : integer	ページ上の検索結果の最大数。サーバーは、リクエストされた数またはデータストアでデフォルトで許可される検索結果の最大数のいずれか低い方を返します。
_since : instant	特定の時点以降に作成されたリソースバージョンのみを含めます。
_at : date(Time)	日時値で指定された期間のある時点で最新であったリソースバージョンのみを含めます。詳細については、HL7 FHIR RESTful API date ドキュメントの「」を参照してください。 HL7 RESTful

4. リクエストを送信します。FHIR historyインタラクションは、AWS 署名バージョン 4 または SMART on FHIR 認可のGETリクエストを使用します。次のcurl例では、 _count検索パラメータを使用して、HealthLake の FHIR Patientリソースの 1 ページあたり 100 件の履歴検索結果を返します。例全体を表示するには、コピーボタンをスクロールします。

SigV4

SigV4 認可

```
curl --request GET \
  'https://healthlake.region.amazonaws.com/datastore/datastore-id/r4/Patient/id/
_history?_count=100' \
  --aws-sigv4 'aws:amz:region:healthlake' \
  --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
  --header "x-amz-security-token:$AWS_SESSION_TOKEN" \
  --header 'Accept: application/json'
```

SMART on FHIR

IdentityProviderConfiguration データ型の FHIR 認可の SMART の例。

```
{
    "AuthorizationStrategy": "SMART_ON_FHIR",
    "FineGrainedAuthorizationEnabled": true,
    "IdpLambdaArn": "arn:aws:lambda:your-region:your-account-id:function:your-
lambda-name",
    "Metadata": "{\"issuer\":\"https://ehr.example.com\", \"jwks_uri\":
\"https://ehr.example.com/.well-known/jwks.json\",\"authorization_endpoint
\":\"https://ehr.example.com/auth/authorize\",\"token_endpoint\":\"https://
ehr.token.com/auth/token\",\"token_endpoint_auth_methods_supported\":
[\"client_secret_basic\",\"foo\"],\"grant_types_supported\":[\"client_credential
\",\"foo\"],\"registration_endpoint\":\"https://ehr.example.com/auth/
register\",\"scopes_supported\":[\"openId\",\"profile\",\"launch\"],
\"response_types_supported\":[\"code\"],\"management_endpoint\":\"https://
ehr.example.com/user/manage\",\"introspection_endpoint\":\"https://
ehr.example.com/user/introspect\",\"revocation_endpoint\":\"https://
ehr.example.com/user/revoke\",\"code_challenge_methods_supported\":[\"S256\"],
\"capabilities\":[\"launch-ehr\",\"sso-openid-connect\",\"client-public\",
\"permission-v2\"]}"
}
```

発信者は認可 Lambda でアクセス許可を割り当てることができます。詳細については、「OAuth 2.0 スコープ」を参照してください。

history インタラクションの戻りコンテンツは FHIR リソース に含まれBundle、タイプは に設定されますhistory。これには、指定されたバージョン履歴が含まれ、最後に最も古いバージョンでソートされ、削除されたリソースが含まれます。詳細については、FHIR R4 ドキュメントResource Bundleの「」を参照してください。

バージョン固有の FHIR リソース履歴の読み取り

FHIR vreadインタラクションは、HealthLake データストア内のリソースのバージョン固有の読み取りを実行します。このインタラクションを使用すると、FHIR リソースのコンテンツは、過去の特定の時点と同じように表示できます。

> **Note**
>
> なしで FHIR historyインタラクションを使用する場合vread、HealthLake は常にリソースのメタデータの最新バージョンを返します。

HealthLake は、サポートされている各リソースの でのバージョニングのサポートCapabilityStatement.rest.resource.versioningを宣言します。すべての HealthLake データストアには、すべてのリソースに Resource.meta.versionId (vid) が含まれます。

FHIR historyインタラクションが有効になっている場合 (10/25/2024 以降に作成されたデータストアの場合はデフォルトで、古いデータストアの場合はリクエストによって)、Bundleレスポンスvidには location要素の一部として が含まれます。次の例では、 は数値 としてvid表示されます1。完全な例を表示するには、「バンドル/バンドルレスポンス (JSON) の例」を参照してください。

```
"response" : {
    "status" : "201 Created",
    "location" : "Patient/12423/_history/1",
    ...}
```

バージョン固有の FHIR リソース履歴を読み取るには

1. HealthLake regionと datastoreIdの値を収集します。詳細については、「データストアのプロパティの取得」を参照してください。
2. 関連する とidvid値を読み取って収集する FHIR Resourceタイプを決定します。詳細については、「リソースタイプ:」を参照してください。
3. HealthLake と FHIR 用に収集された値を使用して、リクエストの URL を作成します。次の例の URL パス全体を表示するには、コピーボタンをスクロールします。

   ```
   GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Resource/id/
   _history/vid
   ```

4. リクエストを送信します。FHIR historyインタラクションは、AWS 署名バージョン 4 またはSMART on FHIR 認可のいずれかのGETリクエストを使用します。次のvreadインタラクションは、 で指定されたPatientリソースメタデータのバージョンについて、FHIR リソースに指定されたコンテンツを含む単一のインスタンスを返しますvid。次の例の URL パス全体を表示するには、コピーボタンをスクロールします。

   SigV4

   SigV4 認可

```
curl --request GET \
  'https://healthlake.region.amazonaws.com/datastore/datastore-id/r4/Patient/id/
_history/vid' \
  --aws-sigv4 'aws:amz:region:healthlake' \
  --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
  --header "x-amz-security-token:$AWS_SESSION_TOKEN" \
  --header 'Accept: application/json'
```

   SMART on FHIR

   IdentityProviderConfiguration データ型の FHIR 認可の SMART の例。

```
{
    "AuthorizationStrategy": "SMART_ON_FHIR",
    "FineGrainedAuthorizationEnabled": true,
    "IdpLambdaArn": "arn:aws:lambda:your-region:your-account-id:function:your-
lambda-name",
    "Metadata": "{\"issuer\":\"https://ehr.example.com\", \"jwks_uri\":
\"https://ehr.example.com/.well-known/jwks.json\",\"authorization_endpoint
\":\"https://ehr.example.com/auth/authorize\",\"token_endpoint\":\"https://
ehr.token.com/auth/token\",\"token_endpoint_auth_methods_supported\":
[\"client_secret_basic\",\"foo\"],\"grant_types_supported\":[\"client_credential
\",\"foo\"],\"registration_endpoint\":\"https://ehr.example.com/auth/
register\",\"scopes_supported\":[\"openId\",\"profile\",\"launch\"],
\"response_types_supported\":[\"code\"],\"management_endpoint\":\"https://
ehr.example.com/user/manage\",\"introspection_endpoint\":\"https://
ehr.example.com/user/introspect\",\"revocation_endpoint\":\"https://
ehr.example.com/user/revoke\",\"code_challenge_methods_supported\":[\"S256\"],
\"capabilities\":[\"launch-ehr\",\"sso-openid-connect\",\"client-public\",
\"permission-v2\"]}"
}
```

発信者は認可 Lambda でアクセス許可を割り当てることができます。詳細については、「OAuth 2.0 スコープ」を参照してください。

FHIR リソースの更新

FHIR updateインタラクションは、既存のリソースの新しい最新バージョンを作成するか、特定の にリソースがまだ存在しない場合は初期バージョンを作成しますid。詳細については、FHIR R4 RESTful API ドキュメントupdateの「」を参照してください。 R4 RESTful

FHIR リソースを更新するには

1. HealthLake regionと datastoreIdの値を収集します。詳細については、「データストアのプロパティの取得」を参照してください。
2. 更新Resourceする FHIR のタイプを決定し、関連するid値を収集します。詳細については、「リソースタイプ:」を参照してください。
3. HealthLake regionと の収集された値を使用して、リクエストの URL を作成しますdatastoreId。また、FHIR Resourceタイプとそれに関連する を含めますid。次の例の URL パス全体を表示するには、コピーボタンにスクロールします。

```
PUT https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Resource/id
```

4. リクエストのJSON本文を作成し、実行する FHIR データの更新を指定します。この手順では、ファイルを として保存しますupdate-patient.json。

```
{
    "id": "2de04858-ba65-44c1-8af1-f2fe69a977d9",
    "resourceType": "Patient",
    "active": true,
    "name": [
        {
            "use": "official",
            "family": "Doe",
            "given": [
                "Jane"
            ]
        },
        {
            "use": "usual",
```

```
            "given": [
                "Jane"
            ]
        }
    ],
    "gender": "female",
    "birthDate": "1985-12-31"
}
```

5. リクエストを送信します。FHIR updateインタラクションは、AWS 署名バージョン 4 または SMART on FHIR 認可のPUTリクエストを使用します。次の のcurl例では、HealthLakeのPatientリソースを更新します。例全体を表示するには、コピーボタンをスクロールします。

   SigV4

   SigV4 認可

   ```
   curl --request PUT \
     'https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id'
    \
     --aws-sigv4 'aws:amz:region:healthlake' \
     --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
     --header "x-amz-security-token:$AWS_SESSION_TOKEN" \
     --header 'Accept: application/json' \
     --data @update-patient.json
   ```

   リクエストは、既存のリソースが更新された場合は 200 HTTP ステータスコード、新しいリソースが作成された場合は 201 HTTP ステータスコードを返します。

   SMART on FHIR

   IdentityProviderConfiguration データ型の FHIR 認可の SMART の例。

   ```
   {
       "AuthorizationStrategy": "SMART_ON_FHIR",
       "FineGrainedAuthorizationEnabled": true,
       "IdpLambdaArn": "arn:aws:lambda:your-region:your-account-id:function:your-
   lambda-name",
       "Metadata": "{\"issuer\":\"https://ehr.example.com\", \"jwks_uri\":
   \"https://ehr.example.com/.well-known/jwks.json\",\"authorization_endpoint
   \":\"https://ehr.example.com/auth/authorize\",\"token_endpoint\":\"https://
   ehr.token.com/auth/token\",\"token_endpoint_auth_methods_supported\":
   ```

```
[\"client_secret_basic\",\"foo\"],\"grant_types_supported\":[\"client_credential
\",\"foo\"],\"registration_endpoint\":\"https://ehr.example.com/auth/
register\",\"scopes_supported\":[\"openId\",\"profile\",\"launch\"],
\"response_types_supported\":[\"code\"],\"management_endpoint\":\"https://
ehr.example.com/user/manage\",\"introspection_endpoint\":\"https://
ehr.example.com/user/introspect\",\"revocation_endpoint\":\"https://
ehr.example.com/user/revoke\",\"code_challenge_methods_supported\":[\"S256\"],
\"capabilities\":[\"launch-ehr\",\"sso-openid-connect\",\"client-public\",
\"permission-v2\"]}"
}
```

発信者は認可 Lambda でアクセス許可を割り当てることができます。詳細については、「OAuth 2.0 スコープ」を参照してください。

AWS Console

1. HealthLake コンソールのクエリの実行ページにサインインします。
2. クエリ設定セクションで、次の選択を行います。

- データストア ID — データストア ID を選択してクエリ文字列を生成します。
- クエリタイプ — を選択しますUpdate（PUT）。
- リソースタイプ — 更新または作成する FHIR リソースタイプを選択します。
- リクエスト本文 — リクエストの JSON 本文を作成し、リソースを更新する FHIR データを指定します。

3. [Run query] (クエリの実行) を選択します。

条件に基づく FHIR リソースの更新

条件付き更新では、論理 FHIR ではなく、一部の識別検索条件に基づいて既存のリソースを更新できますid。サーバーが更新を処理すると、idリクエストの単一の論理を解決するために、リソースタイプの標準検索機能を使用して検索が実行されます。

サーバーが実行するアクションは、見つかった一致の数によって異なります。

- 一致なし、リクエスト本文に**id**指定なし: サーバーは FHIR リソースを作成します。
- 一致する**id**ものがなく、リソースが にまだ存在しない**id**場合: サーバーはインタラクションをアップデートとして処理し、インタラクションを作成します。

- 一致なし、**id**指定済み、既に存在: サーバーは`409 Conflict`エラーで更新を拒否します。
- One Match, no resource **id** provided OR (resource **id** provided and it match the found resource): サーバーは上記のように、一致するリソースに対して更新を実行します。リソースが更新された場合、サーバーは を返します`200 OK`。
- 1 つの一致。リソース**id**が指定されていますが、リソースと一致しません: サーバーは、クライアント ID の仕様が で問題である可能性を示す`409 Conflict`エラーを返します。`OperationOutcome`
- 複数の一致: サーバーは、クライアントの基準が OperationOutcome で十分に選択されていないことを示す`412 Precondition Failed`エラーを返します。

次の例では、名前が peter、生年月日が 2000 年 1 月 1 日、電話番号が 1234567890 の`Patient`リソースを更新します。

```
PUT https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?
name=peter&birthdate=2000-01-01&phone=1234567890
```

リソース更新の検証レベルの設定

FHIR リソースを更新する場合、オプションで HTTP `x-amzn-healthlake-fhir-validation-level` ヘッダーを指定して、リソースの検証レベルを設定できます。 AWS HealthLake は現在、次の検証レベルをサポートしています。

- `strict`: リソースは、リソースのプロファイル要素、またはプロファイルが存在しない場合は R4 仕様に従って検証されます。これはデフォルトの検証レベルです AWS HealthLake。
- `structure-only`: リソースは R4 に対して検証され、参照されるプロファイルは無視されます。
- `minimal`: リソースは、特定の R4 ルールを無視して、最小限検証されます。検索/分析に必要な構造チェックに失敗したリソースは、監査の警告を含むように更新されます。

最小限の検証レベルで更新されたリソースは、検索インデックス作成に必要な検証に失敗しても、Datastore に取り込まれる可能性があります。この場合、リソースが更新されて Healthlake 固有の拡張機能が含まれ、その障害が記録されます。

```
{
    "url": "http://healthlake.amazonaws.com/fhir/StructureDefinition/validation-issue",
    "valueString": "{\"resourceType\":\"OperationOutcome\",\"issue\":[{\"severity\":
\"error\",\"code\":\"processing\",\"details\":{\"text\":\"FHIR resource in payload
```

```
 failed FHIR validation rules.\"},\"diagnostics\":\"FHIR resource in payload failed
 FHIR validation rules.\"}]}"
}
```

さらに、次の HTTP レスポンスヘッダーは「true」の値に含まれます。

```
x-amzn-healthlake-validation-issues : true
```

> **Note**
>
> これらのエラーが存在する場合、R4 仕様に従って誤った形式で取り込まれたデータは期待どおりに検索できない可能性があることに注意してください。

PATCH オペレーションによるリソースの変更

AWS HealthLake は FHIR リソースの PATCH オペレーションをサポートしているため、リソース全体を更新することなく、特定の要素をターゲットにして追加、置換、または削除することで、リソースを変更できます。このオペレーションは、以下が必要な場合に特に役立ちます。

- 大規模なリソースをターゲットに更新する
- ネットワーク帯域幅の使用量を減らす
- 特定のリソース要素に対してアトミック変更を実行する
- 同時変更を上書きするリスクを最小限に抑える
- バッチワークフローとトランザクションワークフローの一部としてリソースを更新する

サポートされているパッチ形式

AWS HealthLake は 2 つの標準 PATCH 形式をサポートしています。

JSON パッチ (RFC 6902)

JSON ポインタ構文を使用して、リソース構造内の要素の位置で要素をターゲットにします。

Content-Type: `application/json-patch+json`

FHIRPath パッチ (FHIR R4 仕様)

FHIRPath 式を使用して、コンテンツと関係によって要素をターゲットにし、パッチ適用に対する FHIR ネイティブアプローチを提供します。

Content-Type: `application/fhir+json`

使用方法

Direct PATCH オペレーション

PATCH オペレーションは、PATCH HTTP メソッドを使用して FHIR リソースで直接呼び出すことができます。

```
PATCH [base]/[resource-type]/[id]{?_format=[mime-type]}
```

バンドル内のパッチ

パッチオペレーションは、タイプ `batch`または の FHIR バンドル内のエントリとして含めることができるため`transaction`、1 つのリクエストでパッチオペレーションを他の FHIR インタラクション (作成、読み取り、更新、削除) と組み合わせることができます。

- トランザクションバンドル: すべてのエントリがアトミックに成功または失敗する
- バッチバンドル: 各エントリは個別に処理されます

JSON パッチ形式

サポートされているオペレーション

運用	説明
`add`	リソースに新しい値を追加する
`remove`	リソースから値を削除する
`replace`	リソース内の既存の値を置き換える
`move`	ある場所から値を削除し、別の場所に追加する

運用	説明
copy	ある場所から別の場所に値をコピーする
test	ターゲットロケーションの値が指定された値と等しいことをテストする

パス構文

JSON パッチは JSON ポインタ構文 (RFC 6901) を使用します。

パスの例	説明
/name/0/family	名前のファミリー要素
/telecom/-	テレコム配列に追加
/active	ルートレベルのアクティブ要素
/address/0/line/1	最初のアドレスの 2 行目

例

複数のオペレーションによる直接 JSON パッチリクエスト

```
PATCH [base]/Patient/example
Content-Type: application/json-patch+json
If-Match: W/"1"

[
  {
    "op": "replace",
    "path": "/name/0/family",
    "value": "Smith"
  },
  {
    "op": "add",
    "path": "/telecom/-",
    "value": {
```

```
      "system": "phone",
      "value": "555-555-5555",
      "use": "home"
    }
  },
  {
    "op": "remove",
    "path": "/address/0"
  },
  {
    "op": "move",
    "from": "/name/0/family",
    "path": "/name/1/family"
  },
  {
    "op": "test",
    "path": "/gender",
    "value": "male"
  },
  {
    "op": "copy",
    "from": "/name/0",
    "path": "/name/1"
  }
]
```

単一のオペレーションによる直接 JSON パッチリクエスト

```
PATCH [base]/Patient/example
Content-Type: application/json-patch+json

[
  {
    "op": "replace",
    "path": "/active",
    "value": false
  }
]
```

バンドルの JSON パッチ

base64 でエンコードされた JSON パッチペイロードを含むバイナリリソースを使用します。

```
{
  "resourceType": "Bundle",
  "type": "transaction",
  "entry": [{
    "resource": {
      "resourceType": "Binary",
      "contentType": "application/json-patch+json",
      "data":
 "W3sib3AiOiJhZGQiLCJwYXRoIjoiL2JpcnRoRGF0ZSIsInZhbHVlIjoiMTk5MC0wMS0wMSJ9XQ=="
    },
    "request": {
      "method": "PATCH",
      "url": "Patient/123"
    }
  }]
}
```

FHIRPath パッチ形式

サポートされているオペレーション

運用	説明
add	リソースに新しい要素を追加する
insert	リスト内の特定の位置に要素を挿入する
delete	リソースから 要素を削除する
replace	既存の要素の値を置き換える
move	リスト内の要素の順序を変更する

パス構文

FHIRPath Patch は、以下をサポートする FHIRPath 式を使用します。

- インデックスベースのアクセス: `Patient.name[0]`
- を使用したフィルタリング**`where()`**: `Patient.name.where(use = 'official')`

- ブールロジック: `Patient.telecom.where(system = 'phone' and use = 'work')`
- サブセット関数: `first()`、 `last()`
- 存在チェック: `exists()`、 `count()`
- 多形ナビゲーション: `Observation.value`

例

Direct FHIRPath パッチリクエスト

```
PATCH [base]/Patient/123
Content-Type: application/fhir+json
Authorization: ...

{
  "resourceType": "Parameters",
  "parameter": [{
    "name": "operation",
    "part": [
      { "name": "type", "valueCode": "add" },
      { "name": "path", "valueString": "Patient" },
      { "name": "name", "valueString": "birthDate" },
      { "name": "value", "valueDate": "1990-01-01" }
    ]
  }]
}
```

バンドルの FHIRPath パッチ

Parameters リソースを のエントリリソースとして使用しますmethod: PATCH。

```
{
  "resourceType": "Bundle",
  "type": "transaction",
  "entry": [{
    "resource": {
      "resourceType": "Parameters",
      "parameter": [{
        "name": "operation",
        "part": [
```

```
          { "name": "type", "valueCode": "add" },
          { "name": "path", "valueString": "Patient" },
          { "name": "name", "valueString": "birthDate" },
          { "name": "value", "valueDate": "1990-01-01" }
        ]
      }]
    },
    "request": {
      "method": "PATCH",
      "url": "Patient/123"
    }
  }]
}
```

リクエストヘッダー

ヘッダー	必要	説明
Content-Type	はい	application/json-patch+json for JSON パッチ または application/fhir+json for FHIRPath パッチ
If-Match	いいえ	ETag を使用したバージョン固有の条件付き更新

レスポンス例

オペレーションは、更新されたリソースに新しいバージョン情報を返します。

```
HTTP/1.1 200 OK
Content-Type: application/fhir+json
ETag: W/"2"
Last-Modified: Mon, 05 May 2025 10:10:10 GMT

{
  "resourceType": "Patient",
  "id": "example",
  "active": true,
  "name": [
    {
      "family": "Smith",
```

```
      "given": ["John"]
    }
  ],
  "telecom": [
    {
      "system": "phone",
      "value": "555-555-5555",
      "use": "home"
    }
  ],
  "meta": {
    "versionId": "2",
    "lastUpdated": "2025-05-05T10:10:10Z"
  }
}
```

行動

PATCH オペレーション:

- 適切な仕様に従ってパッチ構文を検証します (JSON パッチの場合は RFC 6902、FHIRPath パッチの場合は FHIR R4)
- オペレーションをアトミックに適用 - すべてのオペレーションが成功または失敗する
- リソースバージョン ID を更新し、新しい履歴エントリを作成します。
- 変更を適用する前に、元のリソースを履歴に保持します
- パッチを適用した後、FHIR リソースの制約を検証します
- ETag で If-Match ヘッダーを使用した条件付き更新をサポート

エラー処理

オペレーションは、次のエラー条件を処理します。

- 400 不正なリクエスト: 無効なパッチ構文 (非準拠のリクエストまたは不正な形式のパッチドキュメント)
- 404 Not Found: リソースが見つかりません (指定された ID は存在しません)
- 409 競合: バージョン競合 (同時更新または以前のバージョン ID の提供)
- 422 未処理のエンティティ: パッチオペレーションを指定されたリソース要素に適用できません

機能の概要

機能	JSON パッチ	FHIRPath パッチ
コンテンツタイプ	`application/json-patch+json`	`application/fhir+json`
パス形式	JSON ポインタ (RFC 6901)	FHIRPath 式
Direct PATCH API	サポート対象	サポート
バンドルバッチ	サポートされている (バイナリ経由)	サポートされている (パラメータ経由)
バンドルトランザクション	サポートされている (バイナリ経由)	サポートされている (パラメータ経由)
オペレーション	追加、削除、置換、移動、コピー、テスト	追加、挿入、削除、置換、移動

制限事項

- 検索条件を使用した条件付き PATCH オペレーションはサポートされていません
- バンドル内の JSON パッチは、base64 でエンコードされたコンテンツを含むバイナリリソースを使用する必要があります
- バンドルの FHIRPath パッチは Parameters リソースを使用する必要があります

その他のリソース

PATCH オペレーションの詳細については、以下を参照してください。

- FHIR R4 パッチドキュメント
- FHIR R4 FHIRPath パッチ仕様
- RFC 6902 - JSON パッチ
- RFC 6901 - JSON ポインタ

FHIR リソースのバンドル

FHIR Bundleは、 の FHIR リソースのコレクション用のコンテナです AWS HealthLake。 は、処理動作が異なる 2 種類のバンドル AWS HealthLake をサポートしています。

Batch バンドルは、各リソースを個別に処理します。1 つのリソースが失敗しても、残りのリソースは成功します。各オペレーションは個別に処理され、一部のオペレーションが失敗しても処理が続行されます。複数の無関係な患者レコードのアップロードなど、部分的な成功が許容される一括操作にはバッチバンドルを使用します。

Transaction バンドルは、すべてのリソースを 1 つのユニットとしてアトミックに処理します。すべてのリソースオペレーションが成功するか、コミット AWS HealthLake しません。トランザクションバンドルは、すべてのデータを一緒に記録する必要がある、関連する観測値と条件を持つ患者を作成するなど、関連するリソース全体で参照整合性を保証する必要がある場合に使用します。

バッチバンドルとトランザクションバンドルの違い

機能	バッチ	トランザクション
処理モデル	各オペレーションは個別に成功または失敗します。	すべてのオペレーションは 1 つのアトミックユニットとして成功または失敗します。
障害処理	個々のオペレーションが失敗しても処理は続行されます。	1 回のオペレーションが失敗すると、バンドル全体が失敗します。
実行順序	実行順序は保証されません。	オペレーションは指定された順序で処理されます。
参照整合性	オペレーション間では適用されません。	バンドル内のローカルに参照されるリソースに対して強制されます。
最適なケース	部分的な成功が許容される一括オペレーション。	一緒に作成または更新する必要がある関連リソース。

同じタイプまたは異なるタイプの FHIR リソースをバンドルでき、、、create、readupdate、deleteなどの FHIR オペレーションの組み合わせを含めることができますpatch。詳細については、FHIR R4 ドキュメントの「リソースバンドル」を参照してください。

各バンドルタイプのユースケースの例を次に示します。

バッチバンドル

- 夜間のデータ同期中に、異なる施設から複数の無関係な患者レコードをアップロードします。
- 一部のレコードに検証の問題がある可能性のある過去の薬剤レコードを一括アップロードします。
- 個々の障害が他のエントリに影響を与えない組織や実務者などのリファレンスデータをロードします。

トランザクションバンドル

- すべてのデータを一緒に記録する必要がある緊急部門への入院中に、関連する観察結果と状態を持つ患者を作成します。
- 一貫性を保つ必要がある患者の薬剤リストおよび関連するアレルギー情報を更新します。
- 患者、観察結果、処置、請求情報との完全な遭遇を単一のアトミックユニットとして記録します。

> **Important**
>
> バッチバンドルとトランザクションバンドルの両方が同じBundleリソース構造を使用します。唯一の違いは、 typeフィールドの値です。

次の例は、複数のリソースタイプとオペレーションを持つトランザクションバンドルを示しています。

```
{
  "resourceType": "Bundle",
  "type": "transaction",
  "entry": [
    {
      "fullUrl": "urn:uuid:4f6a30fb-cd3c-4ab6-8757-532101f72065",
      "resource": {
        "resourceType": "Patient",
        "id": "new-patient",
        "active": true,
        "name": [
          {
            "family": "Johnson",
            "given": [
```

```
            "Sarah"
          ]
        }
      ],
      "gender": "female",
      "birthDate": "1985-08-12",
      "telecom": [
        {
          "system": "phone",
          "value": "555-123-4567",
          "use": "home"
        }
      ]
    },
    "request": {
      "method": "POST",
      "url": "Patient"
    }
  },
  {
    "fullUrl": "urn:uuid:7f83f473-d8cc-4a8d-86d3-9d9876a3248b",
    "resource": {
      "resourceType": "Observation",
      "id": "blood-pressure",
      "status": "final",
      "code": {
        "coding": [
          {
            "system": "http://loinc.org",
            "code": "85354-9",
            "display": "Blood pressure panel"
          }
        ],
        "text": "Blood pressure panel"
      },
      "subject": {
        "reference": "urn:uuid:4f6a30fb-cd3c-4ab6-8757-532101f72065"
      },
      "effectiveDateTime": "2023-10-15T09:30:00Z",
      "component": [
        {
          "code": {
            "coding": [
              {
```

```
                "system": "http://loinc.org",
                "code": "8480-6",
                "display": "Systolic blood pressure"
              }
            ]
          },
          "valueQuantity": {
            "value": 120,
            "unit": "mmHg",
            "system": "http://unitsofmeasure.org",
            "code": "mm[Hg]"
          }
        },
        {
          "code": {
            "coding": [
              {
                "system": "http://loinc.org",
                "code": "8462-4",
                "display": "Diastolic blood pressure"
              }
            ]
          },
          "valueQuantity": {
            "value": 80,
            "unit": "mmHg",
            "system": "http://unitsofmeasure.org",
            "code": "mm[Hg]"
          }
        }
      ]
    },
    "request": {
      "method": "POST",
      "url": "Observation"
    }
  },
  {
    "resource": {
      "resourceType": "Appointment",
      "id": "appointment-123",
      "status": "booked",
      "description": "Annual physical examination",
      "start": "2023-11-15T09:00:00Z",
```

```
        "end": "2023-11-15T09:30:00Z",
        "participant": [
          {
            "actor": {
              "reference": "urn:uuid:4f6a30fb-cd3c-4ab6-8757-532101f72065"
            },
            "status": "accepted"
          }
        ]
      },
      "request": {
        "method": "PUT",
        "url": "Appointment/appointment-123"
      }
    },
    {
      "request": {
        "method": "DELETE",
        "url": "MedicationRequest/med-request-456"
      }
    }
  ]
}
```

独立したエンティティとしての FHIR リソースのバンドル

FHIR リソースを独立したエンティティとしてバンドルするには

1. HealthLake `region`と `datastoreId`の値を収集します。詳細については、「データストアのプロパティの取得」を参照してください。
2. HealthLake `region`と の収集された値を使用して、リクエストの URL を作成しますdatastoreId。URL に FHIR リソースタイプを指定しないでください。次の例の URL パス全体を表示するには、コピーボタンにスクロールします。

   ```
   POST https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
   ```

3. 各 HTTP 動詞を `method`要素の一部として指定して、リクエストの JSON 本文を作成します。次の例では、 `Bundle`リソースと`batch`のタイプインタラクションを使用して、新しい `Patient` および `Medication`リソースを作成します。必要なセクションはすべて、

それに応じてコメントされます。この手順では、ファイルを として保存しますbatch-independent.json。

```
{
    "resourceType": "Bundle",
    "id": "bundle-batch",
    "meta": {
        "lastUpdated": "2014-08-18T01:43:30Z"
    },
    "type": "batch",
    "entry": [
        {
            "resource": {
                "resourceType": "Patient",
                "meta": {
                    "lastUpdated": "2022-06-03T17:53:36.724Z"
                },
                "text": {
                    "status": "generated",
                    "div": "Some narrative"
                },
                "active": true,
                "name": [
                    {
                        "use": "official",
                        "family": "Jackson",
                        "given": [
                            "Mateo",
                            "James"
                        ]
                    }
                ],
                "gender": "male",
                "birthDate": "1974-12-25"
            },
            "request": {
                "method": "POST",
                "url": "Patient"
            }
        },
        {
            "resource": {
                "resourceType": "Medication",
```

```
                "id": "med0310",
                "contained": [
                    {
                        "resourceType": "Substance",
                        "id": "sub03",
                        "code": {
                            "coding": [
                                {
                                    "system": "http://snomed.info/sct",
                                    "code": "55452001",
                                    "display": "Oxycodone (substance)"
                                }
                            ]
                        }
                    }
                ],
                "code": {
                    "coding": [
                        {
                            "system": "http://snomed.info/sct",
                            "code": "430127000",
                            "display": "Oral Form Oxycodone (product)"
                        }
                    ]
                },
                "form": {
                    "coding": [
                        {
                            "system": "http://snomed.info/sct",
                            "code": "385055001",
                            "display": "Tablet dose form (qualifier value)"
                        }
                    ]
                },
                "ingredient": [
                    {
                        "itemReference": {
                            "reference": "#sub03"
                        },
                        "strength": {
                            "numerator": {
                                "value": 5,
                                "system": "http://unitsofmeasure.org",
                                "code": "mg"
```

```
                            },
                            "denominator": {
                                "value": 1,
                                "system": "http://terminology.hl7.org/CodeSystem/
v3-orderableDrugForm",
                                "code": "TAB"
                            }
                        }
                    }
                ]
            },
            "request": {
                "method": "POST",
                "url": "Medication"
            }
        }
    ]
}
```

4. リクエストを送信します。FHIR Bundleバッチタイプは、AWS 署名バージョン 4 またはSMART on FHIR 認可のPOSTリクエストを使用します。次のコード例では、デモ目的で curl コマンドラインツールを使用しています。

   SigV4

   SigV4 認可

   ```
   curl --request POST \
     'https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/' \
     --aws-sigv4 'aws:amz:region:healthlake' \
     --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
     --header "x-amz-security-token:$AWS_SESSION_TOKEN" \
     --header 'Accept: application/json' \
     --data @batch-type.json
   ```

   SMART on FHIR

   IdentityProviderConfiguration データ型の FHIR 認可の SMART の例。

   ```
   {
       "AuthorizationStrategy": "SMART_ON_FHIR",
       "FineGrainedAuthorizationEnabled": true,
   ```

```
    "IdpLambdaArn": "arn:aws:lambda:your-region:your-account-id:function:your-
lambda-name",
    "Metadata": "{\"issuer\":\"https://ehr.example.com\", \"jwks_uri\":
\"https://ehr.example.com/.well-known/jwks.json\",\"authorization_endpoint
\":\"https://ehr.example.com/auth/authorize\",\"token_endpoint\":\"https://
ehr.token.com/auth/token\",\"token_endpoint_auth_methods_supported\":
[\"client_secret_basic\",\"foo\"],\"grant_types_supported\":[\"client_credential
\",\"foo\"],\"registration_endpoint\":\"https://ehr.example.com/auth/
register\",\"scopes_supported\":[\"openId\",\"profile\",\"launch\"],
\"response_types_supported\":[\"code\"],\"management_endpoint\":\"https://
ehr.example.com/user/manage\",\"introspection_endpoint\":\"https://
ehr.example.com/user/introspect\",\"revocation_endpoint\":\"https://
ehr.example.com/user/revoke\",\"code_challenge_methods_supported\":[\"S256\"],
\"capabilities\":[\"launch-ehr\",\"sso-openid-connect\",\"client-public\",
\"permission-v2\"]}"
}
```

発信者は認可 Lambda でアクセス許可を割り当てることができます。詳細については、「OAuth 2.0 スコープ」を参照してください。

サーバーは、Bundleバッチタイプリクエストの結果として作成された Patientおよび Medicationリソースを示すレスポンスを返します。

バンドルの条件付き PUTs

AWS HealthLake は、次のクエリパラメータを使用してバンドル内の条件付き更新をサポートします。

> **Note**
>
> 条件付き PUTs はbatchバンドルでのみサポートされます。 Transactionバンドルは条件付き PUTsをサポートしていません。

- _id (スタンドアロン)
- _id を次のいずれかと組み合わせて使用します。
  - _tag
  - _createdAt

- `_lastUpdated`

バンドルで条件付き PUTs を使用する場合、 は既存のリソースに対してクエリパラメータ AWS HealthLake を評価し、一致結果に基づいてアクションを実行します。

条件付き更新動作

シナリオ	HTTP ステータス	実行されたアクション
ID が指定されていないリソース	201 作成	常に新しいリソースを作成します。
新しい ID を持つリソース (一致なし)	201 作成	指定された ID で新しいリソースを作成します。
既存の ID を持つリソース (単一一致)	200 OK	一致するリソースを更新します。
既存の ID を持つリソース (競合が検出されました)	409 Conflict	エラーを返します。変更は行われません。
既存の ID を持つリソース (ID の不一致)	400 Bad Request	エラーを返します。変更は行われません。
複数のリソースの一致条件	412 Precondition Failed	エラーを返します。変更は行われません。

次の条件付き更新のバンドル例では、FHIR ID を持つ`Patient`リソース`_lastUpdated=lt2025-04-20`は、条件が満たされた場合にのみ`476`更新されます。

```
{
    "resourceType": "Bundle",
    "id": "bundle-batch",
    "meta": {
        "lastUpdated": "2014-08-18T01:43:30Z"
    },
    "type": "batch",
    "entry": [
        {
            "resource": {
```

```
                "resourceType": "Patient",
                "id": "476",
                "meta": {
                    "lastUpdated": "2022-06-03T17:53:36.724Z"
                },
                "active": true,
                "name": [
                    {
                        "use": "official",
                        "family": "Jackson",
                        "given": [
                            "Mateo",
                            "James"
                        ]
                    }
                ],
                "gender": "male",
                "birthDate": "1974-12-25"
            },
            "request": {
                "method": "PUT",
                "url": "Patient?_id=476&_lastUpdated=lt2025-04-20"
            }
        },
        {
            "resource": {
                "resourceType": "Medication",
                "id": "med0310",
                "contained": [
                    {
                        "resourceType": "Substance",
                        "id": "sub03",
                        "code": {
                            "coding": [
                                {
                                    "system": "http://snomed.info/sct",
                                    "code": "55452001",
                                    "display": "Oxycodone (substance)"
                                }
                            ]
                        }
                    }
                ],
                "code": {
```

```
                    "coding": [
                        {
                            "system": "http://snomed.info/sct",
                            "code": "430127000",
                            "display": "Oral Form Oxycodone (product)"
                        }
                    ]
                },
                "form": {
                    "coding": [
                        {
                            "system": "http://snomed.info/sct",
                            "code": "385055001",
                            "display": "Tablet dose form (qualifier value)"
                        }
                    ]
                },
                "ingredient": [
                    {
                        "itemReference": {
                            "reference": "#sub03"
                        },
                        "strength": {
                            "numerator": {
                                "value": 5,
                                "system": "http://unitsofmeasure.org",
                                "code": "mg"
                            },
                            "denominator": {
                                "value": 1,
                                "system": "http://terminology.hl7.org/CodeSystem/v3-
orderableDrugForm",
                                "code": "TAB"
                            }
                        }
                    }
                ]
            },
            "request": {
                "method": "POST",
                "url": "Medication"
            }
        }
    ]
```

```
}
```

FHIR リソースを単一のエンティティとしてバンドルする

FHIR リソースを単一のエンティティとしてバンドルするには

1. HealthLake regionと datastoreIdの値を収集します。詳細については、「データストアのプロパティの取得」を参照してください。
2. HealthLake regionと の収集された値を使用して、リクエストの URL を作成しますdatastoreId。URL Bundleの一部として FHIR リソースタイプを含めます。次の例の URL パス全体を表示するには、コピーボタンをスクロールします。

```
POST https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Bundle
```

3. リクエストの JSON 本文を作成し、グループ化する FHIR リソースを指定します。次の例では、HealthLake で 2 つのPatientリソースをグループ化します。この手順では、ファイルを としてbatch-single.json保存しますbatch-single.json。

```
{
    "resourceType": "Bundle",
    "id": "bundle-minimal",
    "language": "en-US",
    "identifier": {
        "system": "urn:oid:1.2.3.4.5",
        "value": "28b95815-76ce-457b-b7ae-a972e527db4f"
    },
    "type": "document",
    "timestamp": "2020-12-11T14:30:00+01:00",
    "entry": [
        {
            "fullUrl": "urn:uuid:f40b07e3-37e8-48c3-bf1c-ae70fe12dabf",
            "resource": {
                "resourceType": "Composition",
                "id": "f40b07e3-37e8-48c3-bf1c-ae70fe12dabf",
                "status": "final",
                "type": {
                    "coding": [
                        {
                            "system": "http://loinc.org",
                            "code": "60591-5",
                            "display": "Patient summary Document"
```

```
                        }
                    ]
                },
                "date": "2020-12-11T14:30:00+01:00",
                "author": [
                    {
                        "reference":
 "urn:uuid:45271f7f-63ab-4946-970f-3daaaa0663ff"
                    }
                ],
                "title": "Patient Summary as of December 7, 2020 14:30"
            }
        },
        {
            "fullUrl": "urn:uuid:45271f7f-63ab-4946-970f-3daaaa0663ff",
            "resource": {
                "resourceType": "Practitioner",
                "id": "45271f7f-63ab-4946-970f-3daaaa0663ff",

                "active": true,
                "name": [
                    {
                        "family": "Doe",
                        "given": [
                            "John"
                        ]
                    }
                ]
            }
        }
    ]
}
```

4. リクエストを送信します。FHIR Bundleドキュメントタイプは、AWS 署名バージョン 4 の署名プロトコルを持つPOSTリクエストを使用します。次のコード例では、デモ目的で curl コマンドラインツールを使用しています。

   SigV4

   SigV4 認可

```
curl --request POST \
  'https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Bundle' \
```

```
  --aws-sigv4 'aws:amz:region:healthlake' \
  --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
  --header "x-amz-security-token:$AWS_SESSION_TOKEN" \
  --header 'Accept: application/json' \
  --data @document-type.json
```

SMART on FHIR

IdentityProviderConfiguration データ型の FHIR 認可の SMART の例。

```
{
    "AuthorizationStrategy": "SMART_ON_FHIR",
    "FineGrainedAuthorizationEnabled": true,
    "IdpLambdaArn": "arn:aws:lambda:your-region:your-account-id:function:your-
lambda-name",
    "Metadata": "{\"issuer\":\"https://ehr.example.com\", \"jwks_uri\":
\"https://ehr.example.com/.well-known/jwks.json\",\"authorization_endpoint
\":\"https://ehr.example.com/auth/authorize\",\"token_endpoint\":\"https://
ehr.token.com/auth/token\",\"token_endpoint_auth_methods_supported\":
[\"client_secret_basic\",\"foo\"],\"grant_types_supported\":[\"client_credential
\",\"foo\"],\"registration_endpoint\":\"https://ehr.example.com/auth/
register\",\"scopes_supported\":[\"openId\",\"profile\",\"launch\"],
\"response_types_supported\":[\"code\"],\"management_endpoint\":\"https://
ehr.example.com/user/manage\",\"introspection_endpoint\":\"https://
ehr.example.com/user/introspect\",\"revocation_endpoint\":\"https://
ehr.example.com/user/revoke\",\"code_challenge_methods_supported\":[\"S256\"],
\"capabilities\":[\"launch-ehr\",\"sso-openid-connect\",\"client-public\",
\"permission-v2\"]}"
}
```

発信者は認可 Lambda でアクセス許可を割り当てることができます。詳細については、「OAuth 2.0 スコープ」を参照してください。

サーバーは、Bundleドキュメントタイプリクエストの結果として作成された 2 つのPatientリソースを示すレスポンスを返します。

バンドルの検証レベルの設定

FHIR リソースをバンドルする場合、オプションで HTTP x-amzn-healthlake-fhir-validation-level ヘッダーを指定して、リソースの検証レベルを設定できます。この検証レベル

は、バンドル内のすべての作成および更新リクエストに対して設定されます。 AWS HealthLake は現在、次の検証レベルをサポートしています。

- `strict`: リソースは、リソースのプロファイル要素、またはプロファイルが存在しない場合は R4 仕様に従って検証されます。これは のデフォルトの検証レベルです AWS HealthLake。
- `structure-only`: リソースは R4 に対して検証され、参照されるプロファイルは無視されます。
- `minimal`: リソースは、特定の R4 ルールを無視して、最小限検証されます。検索/分析に必要な構造チェックに失敗したリソースは、監査の警告を含むように更新されます。

最小限の検証レベルにバンドルされたリソースは、検索インデックス作成に必要な検証に失敗しても、Datastore に取り込むことができます。この場合、リソースが更新されて Healthlake 固有の拡張機能が含まれ、上記の失敗が記録されます。バンドルレスポンス内のエントリには、次のように OperationOutcome リソースが含まれます。

```
{
    "resourceType": "Bundle",
    "type": "batch-response",
    "timestamp": "2025-08-25T22:58:48.846287342Z",
    "entry": [
        {
            "response": {
                "status": "201",
                "location": "Patient/195abc49-ba8e-4c8b-95c2-abc88fef7544/_history/1",
                "etag": "W/\"1\"",
                "lastModified": "2025-08-25T22:58:48.801245445Z",
                "outcome": {
                    "resourceType": "OperationOutcome",
                    "issue": [
                        {
                            "severity": "error",
                            "code": "processing",
                            "details": {
                                "text": "FHIR resource in payload failed FHIR
 validation rules."
                            },
                            "diagnostics": "FHIR resource in payload failed FHIR
 validation rules."
                        }
                    ]
                }
            }
```

```
        }
    ]
}
```

さらに、次の HTTP レスポンスヘッダーは「true」の値に含まれます。

```
x-amzn-healthlake-validation-issues : true
```

> **Note**
>
> これらのエラーが存在する場合、R4 仕様に従って誤った形式で取り込まれたデータは期待どおりに検索できない可能性があることに注意してください。

バンドルタイプ「メッセージ」の限定サポート

HealthLake では、内部変換プロセス`message`を通じて FHIR バンドルタイプのサポートが制限されています。このサポートは、レガシーの病院システムから ADT (入室、退室、転送) フィードを取り込むなど、メッセージバンドルをソースで再フォーマットできないシナリオ向けに設計されています。

> **Warning**
>
> この機能には明示的な AWS アカウント許可リストが必要であり、FHIR R4 メッセージセマンティクスや参照整合性は適用されません。メッセージバンドルを使用する前に、 AWS サポートに連絡してアカウントの有効化をリクエストしてください。

標準メッセージ処理との主な違い

- メッセージバンドル (FHIR 仕様): 最初のエントリは、他のリソースを参照`MessageHeader`するである必要があります。リソースには個々の`request`オブジェクトがなく、MessageHeader イベントによって処理アクションが決まります。
- HealthLake 処理: 各リソースエントリに PUT オペレーションを自動的に割り当てることで、メッセージのバンドルをバッチバンドルに変換します。リソースは、メッセージセマンティクスや参照整合性を適用せずに個別に処理されます。

重要な制限事項

- FHIR R4 メッセージ固有の処理ルールは適用されません
- リソース間でトランザクションの整合性がない
- リソース間の参照は検証されません
- 明示的なアカウント許可リストが必要です

メッセージバンドル構造の例

```
{
              "resourceType": "Bundle",
              "type": "message",
              "entry": [
                {
                  "resource": {
                    "resourceType": "MessageHeader",
                    "eventCoding": {
                      "system": "http://hl7.org/fhir/us/davinci-alerts/CodeSystem/
notification-event",
                      "code": "notification-admit"
                    },
                    "focus": [{"reference": "Encounter/example-id"}]
                  }
                },
                {
                  "resource": {"resourceType": "Patient", "id": "example-id"}
                },
                {
                  "resource": {"resourceType": "Encounter", "id": "example-id"}
                }
              ]
            }
```

> **Note**
>
> 各リソースは、個別の PUT オペレーションを介して送信されたかのように個別に保存されます。完全な FHIR メッセージングセマンティクスまたは参照整合性検証が必要な場合は、

送信前にメッセージバンドルを前処理するか、アプリケーションレベルの検証を実装します。

非同期バンドルトランザクション

AWS HealthLake は、最大 500 `transaction`個のリソースを持つトランザクションを送信できる非同期`Bundle`タイプをサポートしています。非同期トランザクションを送信すると、HealthLake はそれを処理のためにキューに入れ、すぐにポーリング URL を返します。この URL を使用してステータスを確認し、レスポンスを取得できます。これは FHIR 非同期バンドルパターンに従います。

非同期トランザクションを使用するタイミング

- 1 つのトランザクションで 100 を超えるリソース (同期制限) を送信する必要があります。
- トランザクション処理の完了を待っている間は、アプリケーションをブロックしないようにします。
- 大量の関連リソースをより良いスループットで処理する必要があります。

Important

ポーリング結果は、トランザクション完了後 90 日間使用できます。この 90 日間の期間が過ぎると、ポーリング URL は結果を返さなくなります。このウィンドウ内で結果を取得して保存する統合を設計します。

Note

同期`Bundle`型`transaction`は引き続き最大 100 個のリソースをサポートし、デフォルトの処理モードです。`Prefer: respond-async` ヘッダーなしで 100 を超えるリソース`transaction`を持つ`Bundle`タイプを送信すると、HealthLake は`422 Unprocessable Entity`エラーを返します。タイプのバンドル`batch`は非同期処理ではサポートされていません。`Bundle`タイプのみを非同期で送信`transaction`できます (最大 500 オペレーション) 。

> **Note**
>
> PATCH オペレーションと条件付き PUTsは、非同期バンドルトランザクションではサポートされていません。

非同期トランザクションの送信

非同期トランザクションを送信するには、 `Prefer: respond-async`ヘッダーを使用してデータストアエンドポイントにPOSTリクエストを送信します。バンドルのタイプは である必要がありますtransaction。タイプのバンドルbatchは、非同期バンドル処理ではサポートされていません。

HealthLake は、送信時にバンドルの初期検証を行います。検証が成功すると、HealthLake はポーリング URL を含むcontent-locationレスポンスヘッダーで HTTP 202 Accepted を返します。

非同期Bundle型を送信するには transaction

1. HealthLake データストアエンドポイントにPOSTリクエストを送信します。

```
POST https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
```

2. バンドルタイプ でリクエストの JSON 本文を作成しますtransaction。この手順では、ファイルを として保存しますasync-transaction.json。

```
{
    "resourceType": "Bundle",
    "type": "transaction",
    "entry": [
        {
            "resource": {
                "resourceType": "Patient",
                "active": true,
                "name": [
                    {
                        "use": "official",
                        "family": "Smith",
                        "given": ["Jane"]
                    }
                ],
                "gender": "female",
                "birthDate": "1990-01-15"
```

```
            },
            "request": {
                "method": "POST",
                "url": "Patient"
            }
        },
        {
            "resource": {
                "resourceType": "Observation",
                "status": "final",
                "code": {
                    "coding": [
                        {
                            "system": "http://loinc.org",
                            "code": "85354-9",
                            "display": "Blood pressure panel"
                        }
                    ]
                },
                "subject": {
                    "reference": "urn:uuid:example-patient-id"
                }
            },
            "request": {
                "method": "POST",
                "url": "Observation"
            }
        }
    ]
}
```

3. `Prefer: respond-async` ヘッダーを使用してリクエストを送信します。FHIR Bundleトランザクションタイプは、AWS 署名バージョン 4 または SMART on FHIR 認可のいずれかのPOSTリクエストを使用します。次のコード例では、デモンストレーションの目的で `curl` コマンドラインツールを使用しています。

   SigV4

   SigV4 認可

```
curl --request POST \
  'https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/' \
  --aws-sigv4 'aws:amz:region:healthlake' \
```

```
  --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
  --header "x-amz-security-token:$AWS_SESSION_TOKEN" \
  --header 'Accept: application/json' \
  --header 'Prefer: respond-async' \
  --data @async-transaction.json
```

SMART on FHIR

IdentityProviderConfiguration データ型の FHIR 認可の SMART の例。

```
{
    "AuthorizationStrategy": "SMART_ON_FHIR",
    "FineGrainedAuthorizationEnabled": true,
    "IdpLambdaArn": "arn:aws:lambda:your-region:your-account-id:function:your-
lambda-name",
    "Metadata": "{\"issuer\":\"https://ehr.example.com\", \"jwks_uri\":
\"https://ehr.example.com/.well-known/jwks.json\",\"authorization_endpoint
\":\"https://ehr.example.com/auth/authorize\",\"token_endpoint\":\"https://
ehr.token.com/auth/token\",\"token_endpoint_auth_methods_supported\":
[\"client_secret_basic\",\"foo\"],\"grant_types_supported\":[\"client_credential
\",\"foo\"],\"registration_endpoint\":\"https://ehr.example.com/auth/
register\",\"scopes_supported\":[\"openId\",\"profile\",\"launch\"],
\"response_types_supported\":[\"code\"],\"management_endpoint\":\"https://
ehr.example.com/user/manage\",\"introspection_endpoint\":\"https://
ehr.example.com/user/introspect\",\"revocation_endpoint\":\"https://
ehr.example.com/user/revoke\",\"code_challenge_methods_supported\":[\"S256\"],
\"capabilities\":[\"launch-ehr\",\"sso-openid-connect\",\"client-public\",
\"permission-v2\"]}"
}
```

発信者は認可 Lambda でアクセス許可を割り当てることができます。詳細については、「OAuth 2.0 スコープ」を参照してください。

4. 送信が成功すると、サーバーは HTTP 202 Accepted を返します。content-location レスポンスヘッダーにはポーリング URL が含まれます。レスポンス本文は OperationOutcomeリソースです。

```
HTTP/1.1 202 Accepted
content-location: https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
Transaction/transactionId
```

```
{
```

```
    "resourceType": "OperationOutcome",
    "issue": [
        {
            "severity": "information",
            "code": "informational",
            "diagnostics": "Submitted Asynchronous Bundle Transaction",
            "location": [
                "https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
Transaction/transactionId"
            ]
        }
    ]
}
```

トランザクションステータスのポーリング

非同期トランザクションを送信したら、content-locationレスポンスヘッダーのポーリング URL を使用してトランザクションのステータスを確認します。ポーリング URL にGETリクエストを送信します。

> **Note**
>
> SMART on FHIR 対応データストアの場合、認可トークンにはトランザクションステータスをポーリングするTransactionリソースタイプのreadアクセス許可が含まれている必要があります。SMART on FHIR スコープの詳細については、「」を参照してくださいHealthLake でサポートされている FHIR OAuth 2.0 スコープでの SMART。

ポーリング URL にGETリクエストを送信します。次の例では、 curl コマンドラインツールを使用します。

SigV4

SigV4 認可

```
curl --request GET \
  'https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
Transaction/transactionId' \
  --aws-sigv4 'aws:amz:region:healthlake' \
  --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
```

```
  --header "x-amz-security-token:$AWS_SESSION_TOKEN" \
  --header 'Accept: application/json'
```

SMART on FHIR

FHIR 認可に関する SMART。認可トークンには、Transactionリソースタイプのreadアクセス許可が含まれている必要があります。

```
curl --request GET \
  'https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
Transaction/transactionId' \
  --header 'Authorization: Bearer $SMART_ACCESS_TOKEN' \
  --header 'Accept: application/json'
```

次の表に、考えられるレスポンスを示します。

ポーリングレスポンスコード

HTTP ステータス	意味	レスポンス本文
202 Accepted	トランザクションはキューに入れられます	OperationOutcome 診断「SUBMITTED」を使用する
202 Accepted	トランザクションは処理中です	OperationOutcome 診断「IN_PROGRESS」を使用する
200 OK	トランザクションが正常に完了しました	Bundle タイプ を持つ transaction-response
4xx/5xx	トランザクションが失敗しました	OperationOutcome エラーの詳細を含む

次の例は、各レスポンスタイプを示しています。

キューに入れられたトランザクション (202)

```
{
```

```
    "resourceType": "OperationOutcome",
    "id": "transactionId",
    "issue": [
        {
            "severity": "information",
            "code": "informational",
            "diagnostics": "SUBMITTED"
        }
    ]
}
```

トランザクション処理 (202)

```
{
    "resourceType": "OperationOutcome",
    "id": "transactionId",
    "issue": [
        {
            "severity": "information",
            "code": "informational",
            "diagnostics": "IN_PROGRESS"
        }
    ]
}
```

トランザクション完了 (200)

```
{
    "resourceType": "Bundle",
    "type": "transaction-response",
    "entry": [
        {
            "response": {
                "status": "201",
                "location": "Patient/example-id/_history/1",
                "etag": "W/\"1\"",
                "lastModified": "2024-01-15T10:30:00.000Z"
            }
        },
        {
            "response": {
```

```
                "status": "201",
                "location": "Observation/example-id/_history/1",
                "etag": "W/\"1\"",
                "lastModified": "2024-01-15T10:30:00.000Z"
            }
        }
    ]
}
```

トランザクションが失敗しました (4xx/5xx)

```
{
    "resourceType": "OperationOutcome",
    "issue": [
        {
            "severity": "error",
            "code": "exception",
            "diagnostics": "Transaction failed: conflict detected on resource Patient/
example-id"
        }
    ]
}
```

処理順序

タイプの非同期バンドルtransactionはキューに入れられますが、厳密な送信順序では処理されません。HealthLake は、使用可能な容量とシステム負荷に基づいて処理を最適化します。

> **Important**
>
> 送信された順序で処理されているトランザクションに依存しないでください。たとえば、トランザクション A を午前 10:00 に送信し、トランザクション B を午前 10:01 に送信すると、トランザクション B はトランザクション A の前に完了する可能性があります。アプリケーションを設計する目的は次のとおりです。
>
> - out-of-orderの完了を処理します。
> - ポーリング URL を使用して、各トランザクションを個別に追跡します。
> - ユースケースで注文が重要な場合は、アプリケーションレベルのシーケンスを実装します。

クォータとスロットリング

非同期トランザクションには、次のクォータとレート制限が適用されます。

非同期トランザクションクォータ

クォータ	値	引き上げ可能
非同期トランザクションあたりの最大オペレーション数	500	いいえ
データストアあたりの保留中の最大トランザクション数	500	はい

- 非同期トランザクションは、 で定義されているのと同じ API レート制限を共有しますService Quotas。
- トランザクションステータスのポーリングは、FHIR リソースの読み取り (GET) オペレーションと同じ API レート制限を共有します。
- 保留中のトランザクション制限に達すると、後続の送信は既存のトランザクションが完了するまでエラーを返します。

エラー処理

「トランザクション」バンドルの場合、バンドルに含まれるすべての FHIR リソースはアトミックオペレーションとして処理されます。オペレーションのすべてのリソースは成功する必要があります。成功しない場合、バンドル内のオペレーションは処理されません。

エラーは、HealthLake が同期的に返す送信エラーと、ポーリングによって取得する処理エラーの 2 つのカテゴリに分類されます。

送信エラー

HealthLake は送信時にバンドルを検証し、トランザクションがキューに入る前にエラーを同期的に返します。送信エラーには、無効な FHIR リソース検証エラー、サポートされていないリソースタイプ、500 オペレーション制限の超過、バッチバンドルでの `Prefer: respond-async`ヘッダーの使用が含まれます。データストアの保留中のトランザクション制限に達した場合、HealthLake は を返しますThrottlingException。送信エラーが発生すると、トランザクションはキューに入れられません。

処理エラー

処理エラーは、トランザクションがキューに入れられ、ポーリング URL を介して返された後に発生します。これには、トランザクションの一部であるリソースを別のオペレーションで変更したトランザクションの競合や、処理中のサーバーエラーが含まれます。処理エラーが発生した場合、トランザクション内のリソースに対してリソースミューテーションは行われません。ポーリング URL は、エラーの詳細OperationOutcomeを含む を返します。

FHIR リソースの削除

FHIR deleteインタラクションは、HealthLake データストアから既存の FHIR リソースを削除します。詳細については、FHIR R4 RESTful API ドキュメントdeleteの「」を参照してください。 R4 RESTful

FHIR リソースを削除するには

1. HealthLake regionと datastoreIdの値を収集します。詳細については、「データストアのプロパティの取得」を参照してください。
2. 削除する FHIR のタイプを決定しResource、関連するid値を収集します。詳細については、「リソースタイプ:」を参照してください。
3. HealthLake regionと の収集された値を使用して、リクエストの URL を作成しますdatastoreId。FHIR Resourceタイプと関連する も含めますid。次の例の URL パス全体を表示するには、コピーボタンをスクロールします。

   ```
   DELETE https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Resource/id
   ```

4. リクエストを送信します。FHIR deleteインタラクションは、AWS 署名バージョン 4 または SMART on FHIR 認可のDELETEリクエストを使用します。次の のcurl例では、HealthLake データストアから既存の FHIR Patientリソースを削除します。例全体を表示するには、コピーボタンをスクロールします。

   SigV4

   SigV4 認可

   ```
   curl --request DELETE \
     'https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id'
    \
   ```

```
  --aws-sigv4 'aws:amz:region:healthlake' \
  --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
  --header "x-amz-security-token:$AWS_SESSION_TOKEN" \
  --header 'Accept: application/json'
```

サーバーは、リソースが HealthLake データストアから削除されたことを確認する 204 HTTP ステータスコードを返します。削除リクエストが失敗した場合、リクエストが失敗した理由を示す400一連の HTTP ステータスコードを受け取ります。

SMART on FHIR

IdentityProviderConfiguration データ型の FHIR 認可の SMART の例。

```
{
    "AuthorizationStrategy": "SMART_ON_FHIR",
    "FineGrainedAuthorizationEnabled": true,
    "IdpLambdaArn": "arn:aws:lambda:your-region:your-account-id:function:your-
lambda-name",
    "Metadata": "{\"issuer\":\"https://ehr.example.com\", \"jwks_uri\":
\"https://ehr.example.com/.well-known/jwks.json\",\"authorization_endpoint
\":\"https://ehr.example.com/auth/authorize\",\"token_endpoint\":\"https://
ehr.token.com/auth/token\",\"token_endpoint_auth_methods_supported\":
[\"client_secret_basic\",\"foo\"],\"grant_types_supported\":[\"client_credential
\",\"foo\"],\"registration_endpoint\":\"https://ehr.example.com/auth/
register\",\"scopes_supported\":[\"openId\",\"profile\",\"launch\"],
\"response_types_supported\":[\"code\"],\"management_endpoint\":\"https://
ehr.example.com/user/manage\",\"introspection_endpoint\":\"https://
ehr.example.com/user/introspect\",\"revocation_endpoint\":\"https://
ehr.example.com/user/revoke\",\"code_challenge_methods_supported\":[\"S256\"],
\"capabilities\":[\"launch-ehr\",\"sso-openid-connect\",\"client-public\",
\"permission-v2\"]}"
}
```

発信者は認可 Lambda でアクセス許可を割り当てることができます。詳細については、「OAuth 2.0 スコープ」を参照してください。

AWS Console

1. HealthLake コンソールのクエリの実行ページにサインインします。

2. クエリ設定セクションで、次の選択を行います。

- データストア ID — データストア ID を選択してクエリ文字列を生成します。

- クエリタイプ — を選択しますDelete。
- リソースタイプ — 削除する FHIR リソースタイプを選択します。
- リソース ID — FHIR リソース ID を入力します。

3. [Run query] (クエリの実行) を選択します。

条件に基づく FHIR リソースの削除

条件付き削除は、特定の FHIR リソース ID がわからないが、削除するリソースに関するその他の識別情報がある場合に特に便利です。

条件付き削除を使用すると、論理 FHIR ID ではなく検索条件に基づいて既存のリソースを削除できます。サーバーが削除リクエストを処理すると、リソースタイプの標準検索機能を使用して検索を実行し、リクエストの単一の論理 ID を解決します。

条件付き削除の仕組み

サーバーのアクションは、見つかった一致の数によって異なります。

1. 一致なし: サーバーは通常の削除を試みて適切に応答します (存在しないリソースの場合は 404 Not Found、既に削除されているリソースの場合は 204 No Content)
2. 1 つの一致: サーバーは一致するリソースに対して通常の削除を実行します
3. 複数の一致: クライアントの基準が十分に選択されていないことを示す 412 Precondition Failed エラーを返します

対応シナリオ

AWS HealthLake は、以下のレスポンスパターンで条件付き削除オペレーションを処理します。

正常なオペレーション

- 検索条件が 1 つのアクティブなリソースを正常に識別すると、システムは標準の削除オペレーションと同様に、削除の完了後に 204 No Content を返します。

ID ベースの条件付き削除

追加のパラメータ (createdAt、tag、または _lastUpdated) idを使用して に基づいて条件付き削除を実行する場合:

- 204 コンテンツなし: リソースは既に削除されています
- 404 Not Found: リソースが存在しません
- 409 競合: ID が一致するが、他のパラメータが一致しない

Non-ID-Based条件付き削除

が指定されidていない場合、または createdAt、、tagまたは 以外のパラメータを使用する場合_lastUpdated:

- 404 Not Found: 一致が見つかりません

競合状況

いくつかのシナリオでは、412 Precondition Failed レスポンスが発生します。

- 検索条件に一致するリソースが複数ある (基準が十分に具体的ではない)
- で ETag ヘッダーを使用する場合のバージョン競合 If-Match
- 検索オペレーションと削除オペレーションの間で発生するリソースの更新

条件付き削除が成功した例

次の の例では、特定の基準に基づいて患者リソースを削除します。

```
DELETE https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?
name=peter&birthdate=2000-01-01&phone=1234567890
```

このリクエストは、次の場合に患者リソースを削除します。

- 名前は「peter」です
- 生年月日は 2000 年 1 月 1 日です
- 電話番号は 1234567890 です

ベストプラクティス

1. 特定の検索条件を使用して、複数の一致を回避し、412 エラーを防止します。
2. 同時変更を処理するために必要な場合は、バージョン管理に ETag ヘッダーを検討してください。
3. エラーレスポンスを適切に処理します。
   - 404 の場合: 検索条件を絞り込む
   - 412 の場合: 条件をより具体的にするか、バージョン競合を解決する
4. 検索操作と削除操作の間でリソースが変更される可能性のある、同時実行性の高い環境でのタイミングの競合に備えます。

べき等性と同時実行性

べき等性キー

AWS HealthLake は FHIR POSTオペレーションのべき等性キーをサポートし、リソースの作成中にデータの整合性を確保するための堅牢なメカニズムを提供します。リクエストヘッダーに一意の UUID をべき等性キーとして含めることで、ヘルスケアアプリケーションは、ネットワークの不安定性や自動再試行を伴うシナリオでも、各 FHIR リソースが 1 回だけ作成されることを保証できます。

この機能は、重複する医療記録が重大な結果をもたらす可能性のある医療システムにとって特に重要です。以前のリクエストと同じべき等性キーを使用してリクエストを受信すると、HealthLake は重複を作成する代わりに元のリソースを返します。たとえば、再試行ループ中や冗長なリクエストパイプラインが原因で発生する可能性があります。べき等性キーを使用すると、HealthLake は断続的な接続問題を処理するクライアントアプリケーションにシームレスなエクスペリエンスを提供しながら、データの一貫性を維持できます。

実装

```
POST /<baseURL>/Patient
x-amz-fhir-idempotency-key: 123e4567-e89b-12d3-a456-426614174000
{
    "resourceType": "Patient",
    "name": [...]
}
```

レスポンスシナリオ

最初のリクエスト (201 作成)

- 新しいリソースが正常に作成されました
- レスポンスにはリソース ID が含まれます

リクエストの重複 (409 競合)

- 同じべき等性キーが検出されました
- 元のリソースが返されました
- 新しいリソースは作成されていません

無効なリクエスト (400 Bad Request)

- 不正な形式の UUID
- 必須フィールドがありません

ベストプラクティス

- 新しいリソースの作成ごとに一意の UUID を生成する
- 再試行ロジックのべき等性キーを保存する
- 一貫したキー形式を使用する: UUID v4 を推奨
- リソース作成を処理するクライアントアプリケーションに を実装する

> **Note**
>
> この機能は、厳密なデータ精度を必要とし、重複する医療記録を防ぐ医療システムにとって特に重要です。

AWS HealthLake の ETag

AWS HealthLake は、FHIR リソースのオプティミスティック同時実行制御に ETags を使用し、同時変更を管理し、データ整合性を維持する信頼性の高いメカニズムを提供します。ETag は、リソースの特定のバージョンを表す一意の識別子であり、HTTP ヘッダーを介してバージョン管理システムとして機能します。リソースを読み取りまたは変更する場合、アプリケーションは ETags を使用して意図しない上書きを防止し、データの整合性を確保できます。特に、更新が同時に行われる可能性があるシナリオではそうです。

実装例

```
// Initial Read
GET /fhir/Patient/123
Response:
ETag: W/"1"

// Update with If-Match
PUT /fhir/Patient/123
If-Match: W/"1"
{resource content}

// Create with If-None-Match
PUT /fhir/Patient/123
If-None-Match: *
{resource content}
// Succeeds only if resource doesn't exist
// Fails with 412 if resource exists
```

レスポンスシナリオ

正常なオペレーション (200 OK または 204 コンテンツなし)

- ETag が最新バージョンと一致する
- オペレーションは意図したとおりに進行します

バージョン競合 (412 前提条件失敗)

- ETag が現在のバージョンと一致しません
- データ損失を防ぐための更新が拒否されました

ベストプラクティス

- すべての更新および削除オペレーションに ETagsを含める
- バージョンの競合を処理するための再試行ロジックを実装する
- If-None-Match を使用する: create-if-not-exists シナリオには *
- 変更する前に必ず ETag の鮮度を確認する

この同時実行管理システムは、特に複数のユーザーまたはシステムが同じリソースにアクセスして変更する環境において、医療データの整合性を維持するために不可欠です。

での FHIR リソースの検索 AWS HealthLake

FHIR searchインタラクションを使用して、いくつかのフィルター条件に基づいて HealthLake データストア内の一連の FHIR リソースを検索します。search インタラクションは、 GETまたは POSTリクエストを使用して実行できます。個人を特定できる情報 (PII) または保護された医療情報 (PHI) を含む検索では、PII と PHI がPOSTリクエスト本文の一部として追加され、転送中に暗号化されるため、リクエストを使用することをお勧めします。

> **Note**
>
> この章で説明する FHIR searchインタラクションは、医療データ交換のための HL7 FHIR R4 標準に準拠して構築されています。HL7 FHIR サービスの表現であるため、 AWS CLI および AWS SDKs では提供されません。詳細については、FHIR R4 RESTful API ドキュメントsearchの「」を参照してください。 R4 RESTful

HealthLake は、FHIR R4 検索パラメータのサブセットをサポートしています。詳細については、「HealthLake の FHIR R4 検索パラメータ」を参照してください。

トピック

- GET を使用した FHIR リソースの検索
- POST を使用した FHIR リソースの検索
- FHIR 検索整合性レベル

GET を使用した FHIR リソースの検索

GET リクエストを使用して HealthLake データストアを検索できます。を使用する場合GET、HealthLake は検索パラメータを URL の一部として提供できますが、リクエスト本文の一部として提供することはできません。詳細については、「HealthLake の FHIR R4 検索パラメータ」を参照してください。

> **[重要]**
>
> 個人を特定できる情報 (PII) または保護された医療情報 (PHI) を含む検索の場合、PII と PHI はPOSTリクエスト本文の一部として追加され、転送中に暗号化されるため、セキュリティの

ベストプラクティスではリクエストの使用が要求されます。詳細については、「POST を使用した FHIR リソースの検索」を参照してください。

次の手順に続いて、 を使用して HealthLake データストアGETを検索する例を示します。

で HealthLake データストアを検索するには **GET**

1. HealthLake regionと datastoreIdの値を収集します。詳細については、「データストアのプロパティの取得」を参照してください。
2. 関連するid値を検索して収集する FHIR リソースのタイプを決定します。詳細については、「リソースタイプ:」を参照してください。
3. HealthLake regionと の収集された値を使用して、リクエストの URL を作成しますdatastoreId。FHIR Resourceタイプとサポートされている検索パラメータも含めます。次の例の URL パス全体を表示するには、コピーボタンをスクロールします。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Resource{?[parameters]{&_format=[mime-type]}}
```

4. AWS 署名バージョン 4 または SMART on FHIR 認可を使用してGETリクエストを送信します。次のcurl例では、HealthLake データストア内のPatientリソースの総数を返します。例全体を表示するには、コピーボタンをスクロールします。

   SigV4

   SigV4 認可

```
curl --request GET \
  'https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?_total=accurate' \
  --aws-sigv4 'aws:amz:region:healthlake' \
  --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
  --header "x-amz-security-token:$AWS_SESSION_TOKEN" \
  --header 'Accept: application/json'
```

   SMART on FHIR

   IdentityProviderConfiguration データ型の FHIR 認可の SMART の例。

```
{
    "AuthorizationStrategy": "SMART_ON_FHIR",
    "FineGrainedAuthorizationEnabled": true,
    "IdpLambdaArn": "arn:aws:lambda:your-region:your-account-id:function:your-
lambda-name",
    "Metadata": "{\"issuer\":\"https://ehr.example.com\", \"jwks_uri\":
\"https://ehr.example.com/.well-known/jwks.json\",\"authorization_endpoint
\":\"https://ehr.example.com/auth/authorize\",\"token_endpoint\":\"https://
ehr.token.com/auth/token\",\"token_endpoint_auth_methods_supported\":
[\"client_secret_basic\",\"foo\"],\"grant_types_supported\":[\"client_credential
\",\"foo\"],\"registration_endpoint\":\"https://ehr.example.com/auth/
register\",\"scopes_supported\":[\"openId\",\"profile\",\"launch\"],
\"response_types_supported\":[\"code\"],\"management_endpoint\":\"https://
ehr.example.com/user/manage\",\"introspection_endpoint\":\"https://
ehr.example.com/user/introspect\",\"revocation_endpoint\":\"https://
ehr.example.com/user/revoke\",\"code_challenge_methods_supported\":[\"S256\"],
\"capabilities\":[\"launch-ehr\",\"sso-openid-connect\",\"client-public\",
\"permission-v2\"]}"
}
```

発信者は認可 Lambda でアクセス許可を割り当てることができます。詳細については、「OAuth 2.0 スコープ」を参照してください。

AWS Console

> **Note**
>
> HealthLake コンソールは SigV4 認可のみをサポートします。SMART on FHIR 認可は、 AWS CLI および AWS SDKs。

1. HealthLake コンソールのクエリの実行ページにサインインします。

2. クエリ設定セクションで、次の選択を行います。

- データストア ID — データストア ID を選択してクエリ文字列を生成します。
- クエリタイプ — を選択しますSearch with GET。
- リソースタイプ — 検索する FHIR リソースタイプを選択します。
- 検索パラメータ — 検索パラメータまたは検索パラメータの組み合わせを選択して、クエリを特定のレコードに集中させます。

3. [Run query] (クエリの実行) を選択します。

例: GET で検索する

次のタブでは、 で特定の FHIR リソースタイプを検索するための例を示しますGET。この例では、リクエスト URLs で検索パラメータを指定する方法を示します。

> **Note**
>
> HealthLake コンソールは SigV4 認可のみをサポートします。SMART on FHIR 認可は、AWS CLI および AWS SDKs。
> HealthLake は、FHIR R4 検索パラメータのサブセットをサポートしています。詳細については、「検索パラメータ」を参照してください。

Patient (age)

age は FHIR で定義されたリソースタイプではありませんが、Patientリソースタイプの 要素としてキャプチャされます。次の例を使用して、birthDate 要素を使用してPatientリソースタイプに対して GETベースの検索リクエストを行い、eq検索コンパレータを使用して 1997 年生まれの個人を検索します。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?
birthdate=eq1997
```

Condition

次の例を使用して、 Conditionリソースタイプに対してGETリクエストを行います。検索では、SNOMED 医療コード を含む HealthLake データストア内の条件が検索され72892002、 に変換されますNormal pregnancy。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Condition?
code=72892002
```

DocumentationReference

次の例は、レンサ球菌の診断を受け、アモキシシリンも処方された Patient(s) のDocumentReferenceリソースタイプに対してGETリクエストを作成する方法を示しています。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
DocumentReference?_lastUpdated=le2021-12-19&infer-icd10cm-entity-text-concept-
score;=streptococcal|0.6&infer-rxnorm-entity-text-concept-score=Amoxicillin|0.8
```

Location

次の例を使用して、 Locationリソースタイプに対してGETリクエストを行います。次の検索では、住所の一部としてボストンという都市名を含む HealthLake データストア内の場所を検索します。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Location?
address=boston
```

Observation

次の例を使用して、Observationリソースタイプに対して GETベースの検索リクエストを行います。この検索では、value-concept検索パラメータを使用して266919005、 に変換される医療コード を検索しますNever smoker。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Observation?
value-concept=266919005
```

POST を使用した FHIR リソースの検索

POST リクエストとの FHIR searchインタラクションを使用して HealthLake データストアを検索できます。を使用する場合POST、HealthLake は URL またはリクエスト本文の検索パラメータをサポートしますが、両方を 1 つのリクエストで使用することはできません。

> [重要]
> 個人を特定できる情報 (PII) または保護された医療情報 (PHI) を含む検索の場合、PII と PHI はPOSTリクエスト本文の一部として追加され、転送中に暗号化されるため、セキュリティのベストプラクティスではリクエストの使用が要求されます。

次の手順に続いて、 との FHIR R4 searchインタラクションを使用して HealthLake データストアPOSTを検索する例を示します。この例では、JSON リクエスト本文で検索パラメータを指定する方法を示します。

で HealthLake データストアを検索するには **POST**

1. HealthLake regionと datastoreIdの値を収集します。詳細については、「データストアのプロパティの取得」を参照してください。
2. 関連するid値を検索して収集する FHIR リソースのタイプを決定します。詳細については、「リソースタイプ:」を参照してください。
3. HealthLake regionと の収集された値を使用して、リクエストの URL を作成しますdatastoreId。FHIR Resourceタイプと_searchインタラクションも含めます。次の例の URL パス全体を表示するには、コピーボタンをスクロールします。

   ```
   POST https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Resource/
   _search
   ```

4. 検索する FHIR データを指定して、リクエストの JSON 本文を作成します。この手順では、Observationリソースを検索して、一度も喫煙したことのない患者を検出します。医療コードのステータス を指定するにはNever smoker、JSON リクエスト本文value-concept=266919005で を設定します。search-observation.json という名前でファイルを保存します。

   ```
   value-concept=266919005
   ```

5. リクエストを送信します。FHIR searchインタラクションは、AWS 署名バージョン 4 または SMART on FHIR 認可のいずれかでGETリクエストを使用します。

> **Note**
>
> POST リクエスト本文の検索パラメータを使用してリクエストを行う場合は、 ヘッダーContent-Type: application/x-www-form-urlencodedの一部として を使用します。

次のcurl例では、 Observationリソースタイプに対して POST ベースの検索リクエストを作成します。リクエストはvalue-concept検索パラメータを使用して、値 266919005を示す医療コードを検索しますNever smoker。例全体を表示するには、コピーボタンをスクロールします。

SigV4

SigV4 認可

```
curl --request POST \
  'https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Observation/
_search' \
  --aws-sigv4 'aws:amz:region:healthlake' \
  --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
  --header "x-amz-security-token:$AWS_SESSION_TOKEN" \
  --header "Content-Type: application/x-www-form-urlencoded"
  --header "Accept: application/json"
  --data @search-observation.json
```

SMART on FHIR

IdentityProviderConfiguration データ型の FHIR 認可の SMART の例。

```
{
    "AuthorizationStrategy": "SMART_ON_FHIR",
    "FineGrainedAuthorizationEnabled": true,
    "IdpLambdaArn": "arn:aws:lambda:your-region:your-account-id:function:your-
lambda-name",
    "Metadata": "{\"issuer\":\"https://ehr.example.com\", \"jwks_uri\":
\"https://ehr.example.com/.well-known/jwks.json\",\"authorization_endpoint
\":\"https://ehr.example.com/auth/authorize\",\"token_endpoint\":\"https://
ehr.token.com/auth/token\",\"token_endpoint_auth_methods_supported\":
[\"client_secret_basic\",\"foo\"],\"grant_types_supported\":[\"client_credential
```

```
\",\"foo\"],\"registration_endpoint\":\"https://ehr.example.com/auth/
register\",\"scopes_supported\":[\"openId\",\"profile\",\"launch\"],
\"response_types_supported\":[\"code\"],\"management_endpoint\":\"https://
ehr.example.com/user/manage\",\"introspection_endpoint\":\"https://
ehr.example.com/user/introspect\",\"revocation_endpoint\":\"https://
ehr.example.com/user/revoke\",\"code_challenge_methods_supported\":[\"S256\"],
\"capabilities\":[\"launch-ehr\",\"sso-openid-connect\",\"client-public\",
\"permission-v2\"]}"
}
```

発信者は認可 Lambda でアクセス許可を割り当てることができます。詳細については、「OAuth 2.0 スコープ」を参照してください。

例: POST で検索する

次のタブでは、 で特定の FHIR リソースタイプを検索するための例を示しますPOST。この例では、URLs でリクエストを指定する方法を示します。

> **Note**
>
> HealthLake コンソールは SigV4 認可のみをサポートします。SMART on FHIR 認可は、AWS CLI および AWS SDKs。
> HealthLake は、FHIR R4 検索パラメータのサブセットをサポートしています。詳細については、「検索パラメータ」を参照してください。

Patient (age)

age は FHIR で定義されたリソースタイプではありませんが、Patientリソースタイプの 要素としてキャプチャされます。次の例を使用して、Patientリソースタイプに対して POSTベースの検索リクエストを行います。次の検索例では、eq検索コンパレータを使用して 1997 年生まれの個人を検索します。

```
POST https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/
_search
```

検索で 1997 年を指定するには、リクエスト本文に次の要素を追加します。

```
birthdate=eq1997
```

Condition

以下を使用して、Conditionリソースタイプに対してPOSTリクエストを行います。この検索では、医療コード を含む HealthLake データストア内の場所を検索します72892002。

リクエスト URL とリクエスト本文を指定する必要があります。リクエスト URL の例を次に示します。

```
POST https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Condition/
_search
```

検索する医療コードを指定するには、次の JSON 要素をリクエスト本文に追加します。

```
code=72892002
```

DocumentReference

DocumentReference リソースタイプでPOSTリクエストを行うときに HealthLake の統合自然言語処理 (NLP) の結果を表示するには、次のようにリクエストをフォーマットします。

```
POST https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
DocumentReference/_search
```

参照するDocumentReference検索パラメータを指定するには、「」を参照してください検索パラメータタイプ。次のクエリ文字列は、複数の検索パラメータを使用して、統合された NLP 結果の生成に使用される Amazon Comprehend Medical API オペレーションを検索します。

```
_lastUpdated=le2021-12-19&infer-icd10cm-entity-text-concept-score;=streptococcal|
0.6&infer-rxnorm-entity-text-concept-score=Amoxicillin|0.8
```

Location

次の例を使用して、 Locationリソースタイプに対してPOSTリクエストを行います。検索では、住所の一部としてボストンという都市名を含む HealthLake データストア内の場所を検索します。

リクエスト URL とリクエスト本文を指定する必要があります。リクエスト URL の例を次に示します。

```
POST https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Location/
_search
```

検索Bostonで を指定するには、リクエスト本文に次の要素を追加します。

```
address=Boston
```

Observation

次の例を使用して、Observationリソースタイプに対して POSTベースの検索リクエストを行います。検索では、value-concept検索パラメータを使用して、 266919005を示す医療コードを検索しますNever smoker。

```
POST https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Observation/
_search
```

ステータスを指定するには、JSON Never smoker の本文value-concept=266919005に を設定します。

```
value-concept=266919005
```

FHIR 検索整合性レベル

AWS HealthLake の検索インデックスは、SEARCH オペレーションの GETおよび の結果整合性モデルで動作POSTします。結果整合性では、リソースの検索インデックスの更新が保留中の場合、インデックスの更新が完了するまで、検索結果はリソースのバージョン N-1 を除外します。

AWS HealthLake に、更新されたリソースに対する整合性モデルの動作を選択できるようになりました。デベロッパーには、「結果整合性」、上記のデフォルトの動作、または「強度整合性」を含めることができます。強力な整合性により、検索インデックスの更新が保留中のリソースのリソースの N-1 バージョンを検索結果に含めることができます。これは、検索インデックスの更新がまだ完了していない場合でも、結果にすべてのリソースが必要なユースケースシナリオに使用できます。お客様は、 x-amz-fhir-history-consistency-levelリクエストヘッダーを使用してこの動作を制御できます。

整合性レベル

強力な整合性

検索インデックスの更新が保留中のレコードを含む、一致するすべてのレコードを返すx-amz-fhir-history-consistency-level: strongように を設定します。更新直後にリソースを検索する必要がある場合は、このオプションを使用します。

結果整合性

検索インデックスの更新が完了したレコードのみを返すx-amz-fhir-history-consistency-level: eventualように を設定します。これは、整合性レベルが指定されていない場合のデフォルトの動作です。

使用例

1. リソースを更新する場合:

```
POST <baseURL>/Patient
Content-Type: application/fhir+json
x-amz-fhir-history-consistency-level: strong

{
  "resourceType": "Patient",
  "id": "123",
  "meta": {
    "profile": ["http://hl7.org/fhir/us/core/StructureDefinition/us-core-patient"]
  },
  "identifier": [
    {
      "system": "http://example.org/identifiers",
      "value": "123"
    }
  ],
  "active": true,
  "name": [
    {
      "family": "Smith",
      "given": ["John"]
    }
  ],
  "gender": "male",
```

```
  "birthDate": "1970-01-01"
}
```

2. 後続の検索:

```
GET <baseURL>/Patient?_id=123
```

ベストプラクティス

- 最近更新されたリソースをすぐに検索する必要がある場合は、強力な整合性を使用します。
- 即時の可視性が重要でない一般的なクエリに結果整合性を使用する
- 即時の可視性と潜在的なパフォーマンスへの影響のトレードオフを考慮する

> **Note**
>
> 整合性レベルの設定は、更新されたリソースが検索結果に表示される速度に影響しますが、リソースの実際のストレージには影響しません。
> オプションの`x-amz-fhir-history-consistency-level`ヘッダーを「強力な」に設定すると、リソースあたりの書き込みキャパシティ消費量が 2 倍になります。
> この機能は、バージョン履歴が有効になっているデータストアにのみ適用されます (2024 年 10 月 25 日以降に作成されたすべてのデータストアはデフォルトで有効になっています) 。

を使用した FHIR データのエクスポート AWS HealthLake

ネイティブ AWS HealthLake アクションを使用して、FHIR エクスポートジョブを開始、説明、一覧表示します。エクスポートジョブをキューに入れることができます。非同期エクスポートジョブは FIFO (先入れ先出し) 方式で処理されます。エクスポートジョブを開始するのと同じ方法でジョブをキューに入れることができます。進行中であれば、キューに入れられます。エクスポートジョブの進行中に FHIR リソースを作成、読み取り、更新、または削除できます。

> **Note**
>
> FHIR R4 `$export`オペレーションを使用して HealthLake データストアから FHIR データをエクスポートすることもできます。詳細については、「FHIR を使用した HealthLake データのエクスポート `$export`」を参照してください。

トピック

FHIR エクスポートジョブの開始

`StartFHIRExportJob` を使用して HealthLake データストアから FHIR エクスポートジョブを開始します。次のメニューでは、 の手順と、 AWS マネジメントコンソール および AWS CLI AWS SDKs。詳細については、「AWS HealthLake API リファレンス」の「`StartFHIRExportJob`」を参照してください。

> **メモ**
>
> HealthLake は、医療データ交換の FHIR R4 仕様をサポートしています。したがって、すべてのヘルスデータは FHIR R4 形式でエクスポートされます。

FHIR エクスポートジョブを開始するには

アクセス設定に基づいてメニューを選択します AWS HealthLake。

AWS CLI および SDKs

CLI

AWS CLI

FHIR エクスポートジョブを開始するには

次のstart-fhir-export-job例は、 AWS HealthLake を使用して FHIR エクスポートジョブを開始する方法を示しています。

```
aws healthlake start-fhir-export-job \
    --output-data-config '{"S3Configuration": {"S3Uri":"s3://(Bucket Name)/(Prefix Name)/","KmsKeyId":"arn:aws:kms:us-east-1:012345678910:key/d330e7fc-b56c-4216-a250-f4c43ef46e83"}}' \
    --datastore-id (Data store ID) \
    --data-access-role-arn arn:aws:iam::(AWS Account ID):role/(Role Name)
```

出力:

```
{
    "DatastoreId": "(Data store ID)",
    "JobStatus": "SUBMITTED",
    "JobId": "9b9a51943afaedd0a8c0c26c49135a31"
}
```

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

Python

SDK for Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
 client.

        :return: An instance of HealthLakeWrapper initialized with the default
 HealthLake client.
```

```
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def start_fhir_export_job(
        self,
        job_name: str,
        datastore_id: str,
        output_s3_uri: str,
        kms_key_id: str,
        data_access_role_arn: str,
    ) -> dict[str, str]:
        """
        Starts a HealthLake export job.
        :param job_name: The export job name.
        :param datastore_id: The data store ID.
        :param output_s3_uri: The output S3 URI.
        :param kms_key_id: The KMS key ID associated with the output S3 bucket.
        :param data_access_role_arn: The data access role ARN.
        :return: The export job.
        """
        try:
            response = self.health_lake_client.start_fhir_export_job(
                OutputDataConfig={
                    "S3Configuration": {"S3Uri": output_s3_uri, "KmsKeyId":
kms_key_id}
                },
                DataAccessRoleArn=data_access_role_arn,
                DatastoreId=datastore_id,
                JobName=job_name,
            )

            return response
        except ClientError as err:
            logger.exception(
                "Couldn't start export job. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「StartFHIRExportJob」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    TRY.
        " iv_job_name = 'MyExportJob'
        " iv_output_s3_uri = 's3://my-bucket/export/output/'
        " iv_kms_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012'
        " iv_data_access_role_arn = 'arn:aws:iam::123456789012:role/
HealthLakeExportRole'
        oo_result = lo_hll->startfhirexportjob(
          iv_jobname = iv_job_name
          io_outputdataconfig = NEW /aws1/cl_hlloutputdataconfig(
            io_s3configuration = NEW /aws1/cl_hlls3configuration(
              iv_s3uri = iv_output_s3_uri
              iv_kmskeyid = iv_kms_key_id
            )
          )
          iv_dataaccessrolearn = iv_data_access_role_arn
          iv_datastoreid = iv_datastore_id
        ).
        DATA(lv_job_id) = oo_result->get_jobid( ).
        MESSAGE |Export job started with ID { lv_job_id }.| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
```

```
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllthrottlingex INTO DATA(lo_throttling_ex).
        lv_error = |Throttling error: { lo_throttling_ex->av_err_code }-
{ lo_throttling_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_throttling_ex.
      CATCH /aws1/cx_hllaccessdeniedex INTO DATA(lo_access_ex).
        lv_error = |Access denied: { lo_access_ex->av_err_code }-{ lo_access_ex-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_access_ex.
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスのStartFHIRExportJob」を参照してください。

> 可用性の例
>
> 必要なものが見つからなかった場合。このページの右側サイドバーにあるフィードバックを提供するリンクを使用して、コード例をリクエストします。

AWS コンソール

1. HealthLake コンソールのデータストアページにサインインします。
2. データストアを選択します。
3. [エクスポート] を選択します。

   エクスポートページが開きます。
4. 出力データセクションで、次の情報を入力します。

   - Amazon S3 の出力データの場所
   - 出力の暗号化
5. アクセス許可セクションで、既存の IAM サービスロールを使用する を選択し、ロール名メニューからロールを選択するか、IAM ロールの作成を選択します。

6. [エクスポートの開始] を選択します。

> **Note**
>
> エクスポート中に、ページ上部のバナーでジョブ ID をコピーを選択します。を使用してエクスポートジョブのプロパティを`JobID`リクエストできます AWS CLI。詳細については、「FHIR エクスポートジョブのプロパティの取得」を参照してください。

FHIR エクスポートジョブのプロパティの取得

を使用して`DescribeFHIRExportJob`、HealthLake データストアからエクスポートジョブのプロパティを取得します。以下のメニューでは、 の手順と、 AWS マネジメントコンソール および AWS CLI AWS SDKs。詳細については、「AWS HealthLake API リファレンス」の「`DescribeFHIRExportJob`」を参照してください。

> **メモ**
>
> HealthLake は、医療データ交換の FHIR R4 仕様をサポートしています。したがって、すべてのヘルスデータは FHIR R4 形式でエクスポートされます。

FHIR エクスポートジョブを記述するには

アクセス設定に基づいてメニューを選択します AWS HealthLake。

AWS CLI および SDKs

CLI

AWS CLI

FHIR エクスポートジョブを記述するには

次の`describe-fhir-export-job`例は、 AWS HealthLake で FHIR エクスポートジョブのプロパティを検索する方法を示しています。

```
aws healthlake describe-fhir-export-job \
    --datastore-id (Data store ID) \
    --job-id 9b9a51943afaedd0a8c0c26c49135a31
```

出力:

```
{
    "ExportJobProperties": {
        "DataAccessRoleArn": "arn:aws:iam::(AWS Account ID):role/(Role Name)",
        "JobStatus": "IN_PROGRESS",
        "JobId": "9009813e9d69ba7cf79bcb3468780f16",
        "SubmitTime": "2024-11-20T11:31:46.672000-05:00",
        "EndTime": "2024-11-20T11:34:01.636000-05:00",
        "OutputDataConfig": {
            "S3Configuration": {
            "S3Uri": "s3://(Bucket Name)/(Prefix Name)/",
            "KmsKeyId": "arn:aws:kms:us-east-1:012345678910:key/d330e7fc-
b56c-4216-a250-f4c43ef46e83"
        }

        },
        "DatastoreId": "(Data store ID)"
    }
}
```

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

Python

SDK for Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
 client.

        :return: An instance of HealthLakeWrapper initialized with the default
 HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def describe_fhir_export_job(
```

```
        self, datastore_id: str, job_id: str
    ) -> dict[str, any]:
        """
        Describes a HealthLake export job.
        :param datastore_id: The data store ID.
        :param job_id: The export job ID.
        :return: The export job description.
        """
        try:
            response = self.health_lake_client.describe_fhir_export_job(
                DatastoreId=datastore_id, JobId=job_id
            )
            return response["ExportJobProperties"]
        except ClientError as err:
            logger.exception(
                "Couldn't describe export job with ID %s. Here's why %s",
                job_id,
                err.response["Error"]["Message"],
            )
            raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「DescribeFHIRExportJob」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        " iv_job_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        oo_result = lo_hll->describefhirexportjob(
          iv_datastoreid = iv_datastore_id
          iv_jobid = iv_job_id
        ).
        DATA(lo_export_job_properties) = oo_result->get_exportjobproperties( ).
        IF lo_export_job_properties IS BOUND.
          DATA(lv_job_status) = lo_export_job_properties->get_jobstatus( ).
          MESSAGE |Export job status: { lv_job_status }.| TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        DATA(lv_error) = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        lv_error = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスの「DescribeFHIRExportJob」を参照してください。

> **可用性の例**
>
> 必要なものが見つからなかった場合。このページの右側サイドバーにあるフィードバックを提供するリンクを使用して、コード例をリクエストします。

AWS コンソール

> **Note**
>
> FHIR エクスポートジョブ情報は HealthLake コンソールでは使用できません。代わりに、AWS CLI で を使用してDescribeFHIRExportJob、 などのエクスポートジョブのプロパ

ティをリクエストしますJobStatus。詳細については、このページ AWS CLI の例を参照してください。

FHIR エクスポートジョブの一覧表示

を使用してListFHIRExportJobs、HealthLake データストアの FHIR エクスポートジョブを一覧表示します。次のメニューでは、 の手順と、 AWS マネジメントコンソール および AWS CLI SDK の AWS コード例を示します。 SDKs 詳細については、「AWS HealthLake API リファレンス」の「ListFHIRExportJobs」を参照してください。

> **メモ**
>
> HealthLake は、医療データ交換の FHIR R4 仕様をサポートしています。したがって、すべてのヘルスデータは FHIR R4 形式でエクスポートされます。

FHIR エクスポートジョブを一覧表示するには

アクセス設定に基づいてメニューを選択します AWS HealthLake。

AWS CLI および SDKs

CLI

AWS CLI

すべての FHIR エクスポートジョブを一覧表示するには

次の list-fhir-export-jobs の例は、コマンドを使用して、アカウントに関連付けられたエクスポートジョブのリストを表示する方法を示しています。

```
aws healthlake list-fhir-export-jobs \
    --datastore-id (Data store ID) \
    --submitted-before (DATE like 2024-10-13T19:00:00Z)\
    --submitted-after (DATE like 2020-10-13T19:00:00Z )\
    --job-name "FHIR-EXPORT" \
    --job-status SUBMITTED  \
    --max-results (Integer between 1 and 500)
```

出力:

```
{
    "ExportJobPropertiesList": [
        {
            "ExportJobProperties": {
                "OutputDataConfig": {
                    "S3Uri": "s3://(Bucket Name)/(Prefix Name)/",
                    "S3Configuration": {
                        "S3Uri": "s3://(Bucket Name)/(Prefix Name)/",
                        "KmsKeyId": "(KmsKey Id)"
                    }
                },
                "DataAccessRoleArn": "arn:aws:iam::(AWS Account ID):role/(Role
 Name)",
                "JobStatus": "COMPLETED",
                "JobId": "c145fbb27b192af392f8ce6e7838e34f",
                "JobName": "FHIR-EXPORT",
                "SubmitTime": "2024-11-20T11:31:46.672000-05:00",
                "EndTime": "2024-11-20T11:34:01.636000-05:00",
                "DatastoreId": "(Data store ID)"
            }
        }
    ]
}
```

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

Python

SDK for Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
 client.

        :return: An instance of HealthLakeWrapper initialized with the default
 HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)
```

```
   def list_fhir_export_jobs(
       self,
       datastore_id: str,
       job_name: str = None,
       job_status: str = None,
       submitted_before: datetime = None,
       submitted_after: datetime = None,
   ) -> list[dict[str, any]]:
       """
       Lists HealthLake export jobs satisfying the conditions.
       :param datastore_id: The data store ID.
       :param job_name: The export job name.
       :param job_status: The export job status.
       :param submitted_before: The export job submitted before the specified
date.
       :param submitted_after: The export job submitted after the specified
date.
       :return: A list of export jobs.
       """
       try:
           parameters = {"DatastoreId": datastore_id}
           if job_name is not None:
               parameters["JobName"] = job_name
           if job_status is not None:
               parameters["JobStatus"] = job_status
           if submitted_before is not None:
               parameters["SubmittedBefore"] = submitted_before
           if submitted_after is not None:
               parameters["SubmittedAfter"] = submitted_after
           next_token = None
           jobs = []
           # Loop through paginated results.
           while True:
               if next_token is not None:
                   parameters["NextToken"] = next_token
               response =
self.health_lake_client.list_fhir_export_jobs(**parameters)
               jobs.extend(response["ExportJobPropertiesList"])
               if "NextToken" in response:
                   next_token = response["NextToken"]
               else:
                   break
```

```
            return jobs
        except ClientError as err:
            logger.exception(
                "Couldn't list export jobs. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「ListFHIRExportJobs」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
TRY.
    " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
    IF iv_submitted_after IS NOT INITIAL.
      oo_result = lo_hll->listfhirexportjobs(
        iv_datastoreid = iv_datastore_id
        iv_submittedafter = iv_submitted_after
      ).
    ELSE.
      oo_result = lo_hll->listfhirexportjobs(
        iv_datastoreid = iv_datastore_id
      ).
    ENDIF.
```

```
        DATA(lt_export_jobs) = oo_result->get_exportjobpropertieslist( ).
        DATA(lv_job_count) = lines( lt_export_jobs ).
        MESSAGE |Found { lv_job_count } export job(s).| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスの[ListFHIRExportJobs]」を参照してください。

> **可用性の例**
>
> 必要なものが見つからなかった場合。このページの右側サイドバーにあるフィードバックを提供するリンクを使用して、コード例をリクエストします。

AWS コンソール

> **Note**
>
> FHIR エクスポートジョブ情報は HealthLake コンソールでは使用できません。代わりに、AWS CLI と を使用してすべての FHIR エクスポートジョブ`ListFHIRExportJobs`を一覧表示します。詳細については、このページ AWS CLI の例を参照してください。

SDK を使用した HealthLake のコード例 AWS SDKs

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

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

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

コードの例

- SDK を使用した HealthLake の基本的な例 AWS SDKs
  - SDK を使用した HealthLake のアクション AWS SDKs
    - AWS SDK または CLI CreateFHIRDatastoreで を使用する
    - AWS SDK または CLI DeleteFHIRDatastoreで を使用する
    - AWS SDK または CLI DescribeFHIRDatastoreで を使用する
    - AWS SDK または CLI DescribeFHIRExportJobで を使用する
    - AWS SDK または CLI DescribeFHIRImportJobで を使用する
    - AWS SDK または CLI ListFHIRDatastoresで を使用する
    - AWS SDK または CLI ListFHIRExportJobsで を使用する
    - AWS SDK または CLI ListFHIRImportJobsで を使用する
    - AWS SDK または CLI ListTagsForResourceで を使用する
    - AWS SDK または CLI StartFHIRExportJobで を使用する
    - AWS SDK または CLI StartFHIRImportJobで を使用する
    - AWS SDK または CLI TagResourceで を使用する
    - AWS SDK または CLI UntagResourceで を使用する

SDK を使用した HealthLake の基本的な例 AWS SDKs

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

例

- SDK を使用した HealthLake のアクション AWS SDKs
  - AWS SDK または CLI CreateFHIRDatastoreで を使用する
  - AWS SDK または CLI DeleteFHIRDatastoreで を使用する
  - AWS SDK または CLI DescribeFHIRDatastoreで を使用する
  - AWS SDK または CLI DescribeFHIRExportJobで を使用する
  - AWS SDK または CLI DescribeFHIRImportJobで を使用する
  - AWS SDK または CLI ListFHIRDatastoresで を使用する
  - AWS SDK または CLI ListFHIRExportJobsで を使用する
  - AWS SDK または CLI ListFHIRImportJobsで を使用する
  - AWS SDK または CLI ListTagsForResourceで を使用する
  - AWS SDK または CLI StartFHIRExportJobで を使用する
  - AWS SDK または CLI StartFHIRImportJobで を使用する
  - AWS SDK または CLI TagResourceで を使用する
  - AWS SDK または CLI UntagResourceで を使用する

SDK を使用した HealthLake のアクション AWS SDKs

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

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

例

- AWS SDK または CLI CreateFHIRDatastoreで を使用する
- AWS SDK または CLI DeleteFHIRDatastoreで を使用する
- AWS SDK または CLI DescribeFHIRDatastoreで を使用する
- AWS SDK または CLI DescribeFHIRExportJobで を使用する
- AWS SDK または CLI DescribeFHIRImportJobで を使用する
- AWS SDK または CLI ListFHIRDatastoresで を使用する
- AWS SDK または CLI ListFHIRExportJobsで を使用する

- AWS SDK または CLI ListFHIRImportJobsで を使用する
- AWS SDK または CLI ListTagsForResourceで を使用する
- AWS SDK または CLI StartFHIRExportJobで を使用する
- AWS SDK または CLI StartFHIRImportJobで を使用する
- AWS SDK または CLI TagResourceで を使用する
- AWS SDK または CLI UntagResourceで を使用する

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

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

CLI

AWS CLI

例 1: SigV4 対応の HealthLake データストアを作成する

次のcreate-fhir-datastore例は、 AWS HealthLake で新しいデータストアを作成する方法を示しています。

```
aws healthlake create-fhir-datastore \
    --datastore-type-version R4 \
    --datastore-name "FhirTestDatastore"
```

出力:

```
{
    "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/datastore/
(Data store ID)/r4/",
    "DatastoreArn": "arn:aws:healthlake:us-east-1:(AWS Account ID):datastore/
(Data store ID)",
    "DatastoreStatus": "CREATING",
    "DatastoreId": "(Data store ID)"
}
```

例 2: SMART on FHIR 対応の HealthLake データストアを作成する

次のcreate-fhir-datastore例は、 AWS HealthLake で FHIR 対応データストアに新しい SMART を作成する方法を示しています。

```
aws healthlake create-fhir-datastore \
    --datastore-name "your-data-store-name" \
    --datastore-type-version R4 \
    --preload-data-config PreloadDataType="SYNTHEA" \
    --sse-configuration '{ "KmsEncryptionConfig": {  "CmkType":
 "CUSTOMER_MANAGED_KMS_KEY", "KmsKeyId": "arn:aws:kms:us-east-1:your-account-
id:key/your-key-id" } }' \
    --identity-provider-configuration  file://
identity_provider_configuration.json
```

identity_provider_configuration.json の内容:

```
{
    "AuthorizationStrategy": "SMART_ON_FHIR_V1",
    "FineGrainedAuthorizationEnabled": true,
    "IdpLambdaArn": "arn:aws:lambda:your-region:your-account-id:function:your-
lambda-name",
    "Metadata": "{\"issuer\":\"https://ehr.example.com\", \"jwks_uri\":
\"https://ehr.example.com/.well-known/jwks.json\",\"authorization_endpoint
\":\"https://ehr.example.com/auth/authorize\",\"token_endpoint\":\"https://
ehr.token.com/auth/token\",\"token_endpoint_auth_methods_supported\":
[\"client_secret_basic\",\"foo\"],\"grant_types_supported\":[\"client_credential
\",\"foo\"],\"registration_endpoint\":\"https://ehr.example.com/auth/
register\",\"scopes_supported\":[\"openId\",\"profile\",\"launch\"],
\"response_types_supported\":[\"code\"],\"management_endpoint\":\"https://
ehr.example.com/user/manage\",\"introspection_endpoint\":\"https://
ehr.example.com/user/introspect\",\"revocation_endpoint\":\"https://
ehr.example.com/user/revoke\",\"code_challenge_methods_supported\":[\"S256\"],
\"capabilities\":[\"launch-ehr\",\"sso-openid-connect\",\"client-public\"]}"
}
```

出力:

```
{
    "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/datastore/
(Data store ID)/r4/",
    "DatastoreArn": "arn:aws:healthlake:us-east-1:(AWS Account ID):datastore/
(Data store ID)",
    "DatastoreStatus": "CREATING",
    "DatastoreId": "(Data store ID)"
}
```

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

Python

SDK for Python (Boto3)

```
   @classmethod
   def from_client(cls) -> "HealthLakeWrapper":
       """
       Creates a HealthLakeWrapper instance with a default AWS HealthLake
client.

       :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
       """
       health_lake_client = boto3.client("healthlake")
       return cls(health_lake_client)


   def create_fhir_datastore(
       self,
       datastore_name: str,
       sse_configuration: dict[str, any] = None,
       identity_provider_configuration: dict[str, any] = None,
   ) -> dict[str, str]:
       """
       Creates a new HealthLake data store.
       When creating a SMART on FHIR data store, the following parameters are
required:
       - sse_configuration: The server-side encryption configuration for a SMART
on FHIR-enabled data store.
       - identity_provider_configuration: The identity provider configuration
for a SMART on FHIR-enabled data store.

       :param datastore_name: The name of the data store.
       :param sse_configuration: The server-side encryption configuration for a
SMART on FHIR-enabled data store.
       :param identity_provider_configuration: The identity provider
configuration for a SMART on FHIR-enabled data store.
       :return: A dictionary containing the data store information.
       """
       try:
```

```
            parameters = {"DatastoreName": datastore_name,
 "DatastoreTypeVersion": "R4"}
            if (
                sse_configuration is not None
                and identity_provider_configuration is not None
            ):
                # Creating a SMART on FHIR-enabled data store
                parameters["SseConfiguration"] = sse_configuration
                parameters[
                    "IdentityProviderConfiguration"
                ] = identity_provider_configuration

            response =
 self.health_lake_client.create_fhir_datastore(**parameters)
            return response
        except ClientError as err:
            logger.exception(
                "Couldn't create data store %s. Here's why %s",
                datastore_name,
                err.response["Error"]["Message"],
            )
            raise
```

次のコードは、SMART on FHIR 対応 HealthLake データストアのパラメータの例を示しています。

```
            sse_configuration = {
                "KmsEncryptionConfig": {"CmkType": "AWS_OWNED_KMS_KEY"}
            }
            # TODO: Update the metadata to match your environment.
            metadata = {
                "issuer": "https://ehr.example.com",
                "jwks_uri": "https://ehr.example.com/.well-known/jwks.json",
                "authorization_endpoint": "https://ehr.example.com/auth/
authorize",
                "token_endpoint": "https://ehr.token.com/auth/token",
                "token_endpoint_auth_methods_supported": [
                    "client_secret_basic",
                    "foo",
                ],
                "grant_types_supported": ["client_credential", "foo"],
```

```
                "registration_endpoint": "https://ehr.example.com/auth/register",
                "scopes_supported": ["openId", "profile", "launch"],
                "response_types_supported": ["code"],
                "management_endpoint": "https://ehr.example.com/user/manage",
                "introspection_endpoint": "https://ehr.example.com/user/
introspect",
                "revocation_endpoint": "https://ehr.example.com/user/revoke",
                "code_challenge_methods_supported": ["S256"],
                "capabilities": [
                    "launch-ehr",
                    "sso-openid-connect",
                    "client-public",
                ],
            }
            # TODO: Update the IdpLambdaArn.
            identity_provider_configuration = {
                "AuthorizationStrategy": "SMART_ON_FHIR_V1",
                "FineGrainedAuthorizationEnabled": True,
                "IdpLambdaArn": "arn:aws:lambda:your-region:your-account-
id:function:your-lambda-name",
                "Metadata": json.dumps(metadata),
            }
            data_store = self.create_fhir_datastore(
                datastore_name, sse_configuration,
 identity_provider_configuration
            )
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「CreateFHIRDatastore」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    TRY.
        " iv_datastore_name = 'MyHealthLakeDataStore'
        oo_result = lo_hll->createfhirdatastore(
          iv_datastorename = iv_datastore_name
          iv_datastoretypeversion = 'R4'
        ).
        MESSAGE 'Data store created successfully.' TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllinternalserverex INTO DATA(lo_internal_ex).
        lv_error = |Internal server error: { lo_internal_ex->av_err_code }-
{ lo_internal_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
      CATCH /aws1/cx_hllthrottlingex INTO DATA(lo_throttling_ex).
        lv_error = |Throttling error: { lo_throttling_ex->av_err_code }-
{ lo_throttling_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_throttling_ex.
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスの「CreateFHIRDatastore」を参照してください。

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

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

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

CLI

AWS CLI

FHIR Data Store を削除するには

次のdelete-fhir-datastore例は、 AWS HealthLake でデータストアとそのすべてのコンテンツを削除する方法を示しています。

```
aws healthlake delete-fhir-datastore \
    --datastore-id (Data store ID)
```

出力:

```
{
    "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/datastore/
(Data store ID)/r4/",
    "DatastoreArn": "arn:aws:healthlake:us-east-1:(AWS Account ID):datastore/
(Data store ID)",
    "DatastoreStatus": "DELETING",
    "DatastoreId": "(Data store ID)"
}
```

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

Python

SDK for Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
 client.
```

```
        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def delete_fhir_datastore(self, datastore_id: str) -> None:
        """
        Deletes a HealthLake data store.
        :param datastore_id: The data store ID.
        """
        try:

self.health_lake_client.delete_fhir_datastore(DatastoreId=datastore_id)
        except ClientError as err:
            logger.exception(
                "Couldn't delete data store with ID %s. Here's why %s",
                datastore_id,
                err.response["Error"]["Message"],
            )
            raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「DeleteFHIRDatastore」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        oo_result = lo_hll->deletefhirdatastore(
          iv_datastoreid = iv_datastore_id
        ).
        MESSAGE 'Data store deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_hllaccessdeniedex INTO DATA(lo_access_ex).
        DATA(lv_error) = |Access denied: { lo_access_ex->av_err_code }-
{ lo_access_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_access_ex.
      CATCH /aws1/cx_hllconflictexception INTO DATA(lo_conflict_ex).
        lv_error = |Conflict error: { lo_conflict_ex->av_err_code }-
{ lo_conflict_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_conflict_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスのDeleteFHIRDatastore」を参照してください。

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

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

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

CLI

AWS CLI

FHIR データストアの詳細を取得するには

次のdescribe-fhir-datastore例は、 AWS HealthLake でデータストアのプロパティを検索する方法を示しています。

```
aws healthlake describe-fhir-datastore \
    --datastore-id "1f2f459836ac6c513ce899f9e4f66a59"
```

出力:

```
{
    "DatastoreProperties": {
        "PreloadDataConfig": {
            "PreloadDataType": "SYNTHEA"
        },
        "SseConfiguration": {
            "KmsEncryptionConfig": {
                "CmkType": "CUSTOMER_MANAGED_KMS_KEY",
                "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
            }
        },
        "DatastoreName": "Demo",
        "DatastoreArn": "arn:aws:healthlake:us-east-1:<AWS Account ID>:datastore/
<Data store ID>",
        "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/
datastore/<Data store ID>/r4/",
        "DatastoreStatus": "ACTIVE",
        "DatastoreTypeVersion": "R4",
        "CreatedAt": 1603761064.881,
        "DatastoreId": "<Data store ID>",
        "IdentityProviderConfiguration": {
            "AuthorizationStrategy": "AWS_AUTH",
            "FineGrainedAuthorizationEnabled": false
```

```
        }
    }
}
```

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

Python

SDK for Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def describe_fhir_datastore(self, datastore_id: str) -> dict[str, any]:
        """
        Describes a HealthLake data store.
        :param datastore_id: The data store ID.
        :return: The data store description.
        """
        try:
            response = self.health_lake_client.describe_fhir_datastore(
                DatastoreId=datastore_id
            )
            return response["DatastoreProperties"]
        except ClientError as err:
            logger.exception(
                "Couldn't describe data store with ID %s. Here's why %s",
                datastore_id,
                err.response["Error"]["Message"],
            )
            raise
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「DescribeFHIRDatastore」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        oo_result = lo_hll->describefhirdatastore(
          iv_datastoreid = iv_datastore_id
        ).
        DATA(lo_datastore_properties) = oo_result->get_datastoreproperties( ).
        IF lo_datastore_properties IS BOUND.
          DATA(lv_datastore_name) = lo_datastore_properties-
>get_datastorename( ).
          DATA(lv_datastore_status) = lo_datastore_properties-
>get_datastorestatus( ).
          MESSAGE 'Data store described successfully.' TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        DATA(lv_error) = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
```

```
        lv_error = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスの「DescribeFHIRDatastore」を参照してください。

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

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

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

CLI

AWS CLI

FHIR エクスポートジョブを記述するには

次のdescribe-fhir-export-job例は、 AWS HealthLake で FHIR エクスポートジョブのプロパティを検索する方法を示しています。

```
aws healthlake describe-fhir-export-job \
    --datastore-id (Data store ID) \
    --job-id 9b9a51943afaedd0a8c0c26c49135a31
```

出力:

```
{
    "ExportJobProperties": {
        "DataAccessRoleArn": "arn:aws:iam::(AWS Account ID):role/(Role Name)",
        "JobStatus": "IN_PROGRESS",
        "JobId": "9009813e9d69ba7cf79bcb3468780f16",
        "SubmitTime": "2024-11-20T11:31:46.672000-05:00",
        "EndTime": "2024-11-20T11:34:01.636000-05:00",
        "OutputDataConfig": {
```

```
            "S3Configuration": {
            "S3Uri": "s3://(Bucket Name)/(Prefix Name)/",
            "KmsKeyId": "arn:aws:kms:us-east-1:012345678910:key/d330e7fc-
b56c-4216-a250-f4c43ef46e83"
        }

        },
        "DatastoreId": "(Data store ID)"
    }
}
```

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

Python

SDK for Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
 client.

        :return: An instance of HealthLakeWrapper initialized with the default
 HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def describe_fhir_export_job(
        self, datastore_id: str, job_id: str
    ) -> dict[str, any]:
        """
        Describes a HealthLake export job.
        :param datastore_id: The data store ID.
        :param job_id: The export job ID.
        :return: The export job description.
        """
        try:
            response = self.health_lake_client.describe_fhir_export_job(
```

```
                DatastoreId=datastore_id, JobId=job_id
            )
            return response["ExportJobProperties"]
        except ClientError as err:
            logger.exception(
                "Couldn't describe export job with ID %s. Here's why %s",
                job_id,
                err.response["Error"]["Message"],
            )
            raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「DescribeFHIRExportJob」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        " iv_job_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        oo_result = lo_hll->describefhirexportjob(
          iv_datastoreid = iv_datastore_id
          iv_jobid = iv_job_id
        ).
        DATA(lo_export_job_properties) = oo_result->get_exportjobproperties( ).
        IF lo_export_job_properties IS BOUND.
```

```
          DATA(lv_job_status) = lo_export_job_properties->get_jobstatus( ).
          MESSAGE |Export job status: { lv_job_status }.| TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        DATA(lv_error) = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        lv_error = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスの「DescribeFHIRExportJob」を参照してください。

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

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

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

CLI

AWS CLI

FHIR インポートジョブを記述するには

次のdescribe-fhir-import-job例は、 AWS HealthLake を使用して FHIR インポートジョブのプロパティを学習する方法を示しています。

```
aws healthlake describe-fhir-import-job \
    --datastore-id (Data store ID) \
    --job-id c145fbb27b192af392f8ce6e7838e34f
```

出力:

```
{
    "ImportJobProperties": {
    "InputDataConfig": {
        "S3Uri": "s3://(Bucket Name)/(Prefix Name)/"
        { "arrayitem2": 2 }
    },
    "DataAccessRoleArn": "arn:aws:iam::(AWS Account ID):role/(Role Name)",
    "JobStatus": "COMPLETED",
    "JobId": "c145fbb27b192af392f8ce6e7838e34f",
    "SubmitTime": 1606272542.161,
    "EndTime": 1606272609.497,
    "DatastoreId": "(Data store ID)"
    }
}
```

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

Python

SDK for Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
 client.

        :return: An instance of HealthLakeWrapper initialized with the default
 HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def describe_fhir_import_job(
        self, datastore_id: str, job_id: str
    ) -> dict[str, any]:
        """
        Describes a HealthLake import job.
        :param datastore_id: The data store ID.
        :param job_id: The import job ID.
```

```
        :return: The import job description.
        """
        try:
            response = self.health_lake_client.describe_fhir_import_job(
                DatastoreId=datastore_id, JobId=job_id
            )
            return response["ImportJobProperties"]
        except ClientError as err:
            logger.exception(
                "Couldn't describe import job with ID %s. Here's why %s",
                job_id,
                err.response["Error"]["Message"],
            )
            raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「DescribeFHIRImportJob」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
TRY.
    " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
    " iv_job_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
    oo_result = lo_hll->describefhirimportjob(
      iv_datastoreid = iv_datastore_id
```

```
          iv_jobid = iv_job_id
        ).
        DATA(lo_import_job_properties) = oo_result->get_importjobproperties( ).
        IF lo_import_job_properties IS BOUND.
          DATA(lv_job_status) = lo_import_job_properties->get_jobstatus( ).
          MESSAGE |Import job status: { lv_job_status }.| TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        DATA(lv_error) = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        lv_error = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスの「DescribeFHIRImportJob」を参照してください。

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

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

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

CLI

AWS CLI

FHIR データストアを一覧表示するには

次のlist-fhir-datastores例は、 コマンドの使用方法と、 AWS HealthLake のデータストアのステータスに基づいてユーザーが結果をフィルタリングする方法を示しています。

```
aws healthlake list-fhir-datastores \
    --filter DatastoreStatus=ACTIVE
```

出力:

```
{
    "DatastorePropertiesList": [
    {
        "PreloadDataConfig": {
            "PreloadDataType": "SYNTHEA"
        },
        "SseConfiguration": {
            "KmsEncryptionConfig": {
                "CmkType": "CUSTOMER_MANAGED_KMS_KEY",
                "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
            }
        },
        "DatastoreName": "Demo",
        "DatastoreArn": "arn:aws:healthlake:us-east-1:<AWS Account ID>:datastore/
<Data store ID>",
        "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/
datastore/<Data store ID>/r4/",
        "DatastoreStatus": "ACTIVE",
        "DatastoreTypeVersion": "R4",
        "CreatedAt": 1603761064.881,
        "DatastoreId": "<Data store ID>",
        "IdentityProviderConfiguration": {
            "AuthorizationStrategy": "AWS_AUTH",
            "FineGrainedAuthorizationEnabled": false
        }
    }
    ]
}
```

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

Python

SDK for Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
```

```
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
 client.

        :return: An instance of HealthLakeWrapper initialized with the default
 HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def list_fhir_datastores(self) -> list[dict[str, any]]:
        """
        Lists all HealthLake data stores.
        :return: A list of data store descriptions.
        """
        try:
            next_token = None
            datastores = []

            # Loop through paginated results.
            while True:
                parameters = {}
                if next_token is not None:
                    parameters["NextToken"] = next_token
                response =
 self.health_lake_client.list_fhir_datastores(**parameters)
                datastores.extend(response["DatastorePropertiesList"])
                if "NextToken" in response:
                    next_token = response["NextToken"]
                else:
                    break

            return datastores
        except ClientError as err:
            logger.exception(
                "Couldn't list data stores. Here's why %s", err.response["Error"]
["Message"]
            )
            raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「ListFHIRDatastores」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    TRY.
        oo_result = lo_hll->listfhirdatastores( ).
        DATA(lt_datastores) = oo_result->get_datastorepropertieslist( ).
        DATA(lv_datastore_count) = lines( lt_datastores ).
        MESSAGE |Found { lv_datastore_count } data store(s).| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllthrottlingex INTO DATA(lo_throttling_ex).
        lv_error = |Throttling error: { lo_throttling_ex->av_err_code }-
{ lo_throttling_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_throttling_ex.
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスのListFHIRDatastores」を参照してください。

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

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

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

CLI

AWS CLI

すべての FHIR エクスポートジョブを一覧表示するには

次の list-fhir-export-jobs の例は、コマンドを使用して、アカウントに関連付けられたエクスポートジョブのリストを表示する方法を示しています。

```
aws healthlake list-fhir-export-jobs \
    --datastore-id (Data store ID) \
    --submitted-before (DATE like 2024-10-13T19:00:00Z)\
    --submitted-after (DATE like 2020-10-13T19:00:00Z )\
    --job-name "FHIR-EXPORT" \
    --job-status SUBMITTED  \
    --max-results (Integer between 1 and 500)
```

出力:

```
{
    "ExportJobPropertiesList": [
        {
            "ExportJobProperties": {
                "OutputDataConfig": {
                    "S3Uri": "s3://(Bucket Name)/(Prefix Name)/",
                    "S3Configuration": {
                        "S3Uri": "s3://(Bucket Name)/(Prefix Name)/",
                        "KmsKeyId": "(KmsKey Id)"
                    }
                },
                "DataAccessRoleArn": "arn:aws:iam::(AWS Account ID):role/(Role
 Name)",
                "JobStatus": "COMPLETED",
                "JobId": "c145fbb27b192af392f8ce6e7838e34f",
                "JobName": "FHIR-EXPORT",
                "SubmitTime": "2024-11-20T11:31:46.672000-05:00",
                "EndTime": "2024-11-20T11:34:01.636000-05:00",
                "DatastoreId": "(Data store ID)"
            }
```

```
        }
    ]
}
```

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

Python

SDK for Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def list_fhir_export_jobs(
        self,
        datastore_id: str,
        job_name: str = None,
        job_status: str = None,
        submitted_before: datetime = None,
        submitted_after: datetime = None,
    ) -> list[dict[str, any]]:
        """
        Lists HealthLake export jobs satisfying the conditions.
        :param datastore_id: The data store ID.
        :param job_name: The export job name.
        :param job_status: The export job status.
        :param submitted_before: The export job submitted before the specified
date.
        :param submitted_after: The export job submitted after the specified
date.
        :return: A list of export jobs.
```

```
        """
        try:
            parameters = {"DatastoreId": datastore_id}
            if job_name is not None:
                parameters["JobName"] = job_name
            if job_status is not None:
                parameters["JobStatus"] = job_status
            if submitted_before is not None:
                parameters["SubmittedBefore"] = submitted_before
            if submitted_after is not None:
                parameters["SubmittedAfter"] = submitted_after
            next_token = None
            jobs = []
            # Loop through paginated results.
            while True:
                if next_token is not None:
                    parameters["NextToken"] = next_token
                response =
self.health_lake_client.list_fhir_export_jobs(**parameters)
                jobs.extend(response["ExportJobPropertiesList"])
                if "NextToken" in response:
                    next_token = response["NextToken"]
                else:
                    break
            return jobs
        except ClientError as err:
            logger.exception(
                "Couldn't list export jobs. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「ListFHIRExportJobs」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        IF iv_submitted_after IS NOT INITIAL.
          oo_result = lo_hll->listfhirexportjobs(
            iv_datastoreid = iv_datastore_id
            iv_submittedafter = iv_submitted_after
          ).
        ELSE.
          oo_result = lo_hll->listfhirexportjobs(
            iv_datastoreid = iv_datastore_id
          ).
        ENDIF.
        DATA(lt_export_jobs) = oo_result->get_exportjobpropertieslist( ).
        DATA(lv_job_count) = lines( lt_export_jobs ).
        MESSAGE |Found { lv_job_count } export job(s).| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスのListFHIRExportJobs」を参照してください。

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

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

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

CLI

AWS CLI

すべての FHIR インポートジョブを一覧表示するには

次の list-fhir-import-jobs の例は、コマンドを使用して、アカウントに関連付けられているすべてのインポートジョブのリストを表示する方法を示しています。

```
aws healthlake list-fhir-import-jobs \
    --datastore-id (Data store ID) \
    --submitted-before (DATE like 2024-10-13T19:00:00Z) \
    --submitted-after (DATE like 2020-10-13T19:00:00Z ) \
    --job-name "FHIR-IMPORT" \
    --job-status SUBMITTED  \
    -max-results (Integer between 1 and 500)
```

出力:

```
{
    "ImportJobPropertiesList": [
        {
            "JobId": "c0fddbf76f238297632d4aebdbfc9ddf",
            "JobStatus": "COMPLETED",
            "SubmitTime": "2024-11-20T10:08:46.813000-05:00",
            "EndTime": "2024-11-20T10:10:09.093000-05:00",
            "DatastoreId": "(Data store ID)",
            "InputDataConfig": {
                "S3Uri": "s3://(Bucket Name)/(Prefix Name)/"
            },
            "JobOutputDataConfig": {
                "S3Configuration": {
                    "S3Uri": "s3://(Bucket Name)/
import/6407b9ae4c2def3cb6f1a46a0c599ec0-FHIR_IMPORT-
c0fddbf76f238297632d4aebdbfc9ddf/",
```

```
                    "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/b7f645cb-
e564-4981-8672-9e012d1ff1a0"
                }
            },
            "JobProgressReport": {
                "TotalNumberOfScannedFiles": 1,
                "TotalSizeOfScannedFilesInMB": 0.001798,
                "TotalNumberOfImportedFiles": 1,
                "TotalNumberOfResourcesScanned": 1,
                "TotalNumberOfResourcesImported": 1,
                "TotalNumberOfResourcesWithCustomerError": 0,
                "TotalNumberOfFilesReadWithCustomerError": 0,
                "Throughput": 0.0
            },
            "DataAccessRoleArn": "arn:aws:iam::(AWS Account ID):role/(Role Name)"
        }
    ]
}
```

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

Python

SDK for Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
 client.

        :return: An instance of HealthLakeWrapper initialized with the default
 HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def list_fhir_import_jobs(
        self,
        datastore_id: str,
```

```
        job_name: str = None,
        job_status: str = None,
        submitted_before: datetime = None,
        submitted_after: datetime = None,
    ) -> list[dict[str, any]]:
        """
        Lists HealthLake import jobs satisfying the conditions.
        :param datastore_id: The data store ID.
        :param job_name: The import job name.
        :param job_status: The import job status.
        :param submitted_before: The import job submitted before the specified
date.
        :param submitted_after: The import job submitted after the specified
date.
        :return: A list of import jobs.
        """
        try:
            parameters = {"DatastoreId": datastore_id}
            if job_name is not None:
                parameters["JobName"] = job_name
            if job_status is not None:
                parameters["JobStatus"] = job_status
            if submitted_before is not None:
                parameters["SubmittedBefore"] = submitted_before
            if submitted_after is not None:
                parameters["SubmittedAfter"] = submitted_after
            next_token = None
            jobs = []
            # Loop through paginated results.
            while True:
                if next_token is not None:
                    parameters["NextToken"] = next_token
                response =
self.health_lake_client.list_fhir_import_jobs(**parameters)
                jobs.extend(response["ImportJobPropertiesList"])
                if "NextToken" in response:
                    next_token = response["NextToken"]
                else:
                    break
            return jobs
        except ClientError as err:
            logger.exception(
                "Couldn't list import jobs. Here's why %s",
                err.response["Error"]["Message"],
```

```
            )
            raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「ListFHIRImportJobs」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        IF iv_submitted_after IS NOT INITIAL.
          oo_result = lo_hll->listfhirimportjobs(
            iv_datastoreid = iv_datastore_id
            iv_submittedafter = iv_submitted_after
          ).
        ELSE.
          oo_result = lo_hll->listfhirimportjobs(
            iv_datastoreid = iv_datastore_id
          ).
        ENDIF.
        DATA(lt_import_jobs) = oo_result->get_importjobpropertieslist( ).
        DATA(lv_job_count) = lines( lt_import_jobs ).
        MESSAGE |Found { lv_job_count } import job(s).| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
```

```
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスのListFHIRImportJobs」を参照してください。

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

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

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

CLI

AWS CLI

データストアのタグを一覧表示するには

次の list-tags-for-resource の例では、指定したデータストアに関連付けられているタグを一覧表示します。

```
aws healthlake list-tags-for-resource \
    --resource-arn "arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/0725c83f4307f263e16fd56b6d8ebdbe"
```

出力:

```
{
    "tags": {
        "key": "value",
```

```
        "key1": "value1"
    }
}
```

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

Python

SDK for Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def list_tags_for_resource(self, resource_arn: str) -> dict[str, str]:
        """
        Lists the tags for a HealthLake resource.
        :param resource_arn: The resource ARN.
        :return: The tags for the resource.
        """
        try:
            response = self.health_lake_client.list_tags_for_resource(
                ResourceARN=resource_arn
            )
            return response["Tags"]
        except ClientError as err:
            logger.exception(
                "Couldn't list tags for resource %s. Here's why %s",
                resource_arn,
                err.response["Error"]["Message"],
            )
            raise
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「ListTagsForResource」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    TRY.
        " iv_resource_arn = 'arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        DATA(lo_result) = lo_hll->listtagsforresource(
          iv_resourcearn = iv_resource_arn
        ).
        ot_tags = lo_result->get_tags( ).
        DATA(lv_tag_count) = lines( ot_tags ).
        MESSAGE |Found { lv_tag_count } tag(s).| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
```

```
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスのListTagsForResource」を参照してください。

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

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

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

CLI

AWS CLI

FHIR エクスポートジョブを開始するには

次のstart-fhir-export-job例は、 AWS HealthLake を使用して FHIR エクスポートジョブを開始する方法を示しています。

```
aws healthlake start-fhir-export-job \
    --output-data-config '{"S3Configuration": {"S3Uri":"s3://(Bucket Name)/(Prefix Name)/","KmsKeyId":"arn:aws:kms:us-east-1:012345678910:key/d330e7fc-b56c-4216-a250-f4c43ef46e83"}}' \
    --datastore-id (Data store ID) \
    --data-access-role-arn arn:aws:iam::(AWS Account ID):role/(Role Name)
```

出力:

```
{
    "DatastoreId": "(Data store ID)",
    "JobStatus": "SUBMITTED",
    "JobId": "9b9a51943afaedd0a8c0c26c49135a31"
}
```

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

Python

SDK for Python (Boto3)

```
   @classmethod
   def from_client(cls) -> "HealthLakeWrapper":
       """
       Creates a HealthLakeWrapper instance with a default AWS HealthLake
client.

       :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
       """
       health_lake_client = boto3.client("healthlake")
       return cls(health_lake_client)


   def start_fhir_export_job(
       self,
       job_name: str,
       datastore_id: str,
       output_s3_uri: str,
       kms_key_id: str,
       data_access_role_arn: str,
   ) -> dict[str, str]:
       """
       Starts a HealthLake export job.
       :param job_name: The export job name.
       :param datastore_id: The data store ID.
       :param output_s3_uri: The output S3 URI.
       :param kms_key_id: The KMS key ID associated with the output S3 bucket.
       :param data_access_role_arn: The data access role ARN.
       :return: The export job.
       """
       try:
           response = self.health_lake_client.start_fhir_export_job(
               OutputDataConfig={
                   "S3Configuration": {"S3Uri": output_s3_uri, "KmsKeyId":
kms_key_id}
               },
               DataAccessRoleArn=data_access_role_arn,
               DatastoreId=datastore_id,
               JobName=job_name,
           )
```

```
            return response
        except ClientError as err:
            logger.exception(
                "Couldn't start export job. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「StartFHIRExportJob」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    TRY.
        " iv_job_name = 'MyExportJob'
        " iv_output_s3_uri = 's3://my-bucket/export/output/'
        " iv_kms_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012'
        " iv_data_access_role_arn = 'arn:aws:iam::123456789012:role/
HealthLakeExportRole'
        oo_result = lo_hll->startfhirexportjob(
          iv_jobname = iv_job_name
          io_outputdataconfig = NEW /aws1/cl_hlloutputdataconfig(
            io_s3configuration = NEW /aws1/cl_hlls3configuration(
```

```
              iv_s3uri = iv_output_s3_uri
              iv_kmskeyid = iv_kms_key_id
            )
          )
          iv_dataaccessrolearn = iv_data_access_role_arn
          iv_datastoreid = iv_datastore_id
        ).
        DATA(lv_job_id) = oo_result->get_jobid( ).
        MESSAGE |Export job started with ID { lv_job_id }.| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllthrottlingex INTO DATA(lo_throttling_ex).
        lv_error = |Throttling error: { lo_throttling_ex->av_err_code }-
{ lo_throttling_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_throttling_ex.
      CATCH /aws1/cx_hllaccessdeniedex INTO DATA(lo_access_ex).
        lv_error = |Access denied: { lo_access_ex->av_err_code }-{ lo_access_ex-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_access_ex.
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスのStartFHIRExportJob」を参照してください。

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

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

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

CLI

AWS CLI

FHIR インポートジョブを開始するには

次のstart-fhir-import-job例は、 AWS HealthLake を使用して FHIR インポートジョブを開始する方法を示しています。

```
aws healthlake start-fhir-import-job \
    --input-data-config S3Uri="s3://(Bucket Name)/(Prefix Name)/" \
    --job-output-data-config '{"S3Configuration": {"S3Uri":"s3://(Bucket Name)/(Prefix Name)/","KmsKeyId":"arn:aws:kms:us-east-1:012345678910:key/d330e7fc-b56c-4216-a250-f4c43ef46e83"}}' \
    --datastore-id (Data store ID) \
    --data-access-role-arn "arn:aws:iam::(AWS Account ID):role/(Role Name)"
```

出力:

```
{
    "DatastoreId": "(Data store ID)",
    "JobStatus": "SUBMITTED",
    "JobId": "c145fbb27b192af392f8ce6e7838e34f"
}
```

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

Python

SDK for Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
 client.

        :return: An instance of HealthLakeWrapper initialized with the default
 HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
```

```
        return cls(health_lake_client)


    def start_fhir_import_job(
        self,
        job_name: str,
        datastore_id: str,
        input_s3_uri: str,
        job_output_s3_uri: str,
        kms_key_id: str,
        data_access_role_arn: str,
    ) -> dict[str, str]:
        """
        Starts a HealthLake import job.
        :param job_name: The import job name.
        :param datastore_id: The data store ID.
        :param input_s3_uri: The input S3 URI.
        :param job_output_s3_uri: The job output S3 URI.
        :param kms_key_id: The KMS key ID associated with the output S3 bucket.
        :param data_access_role_arn: The data access role ARN.
        :return: The import job.
        """
        try:
            response = self.health_lake_client.start_fhir_import_job(
                JobName=job_name,
                InputDataConfig={"S3Uri": input_s3_uri},
                JobOutputDataConfig={
                    "S3Configuration": {
                        "S3Uri": job_output_s3_uri,
                        "KmsKeyId": kms_key_id,
                    }
                },
                DataAccessRoleArn=data_access_role_arn,
                DatastoreId=datastore_id,
            )
            return response
        except ClientError as err:
            logger.exception(
                "Couldn't start import job. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「StartFHIRImportJob」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    TRY.
        " iv_job_name = 'MyImportJob'
        " iv_input_s3_uri = 's3://my-bucket/import/data.ndjson'
        " iv_job_output_s3_uri = 's3://my-bucket/import/output/'
        " iv_kms_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012'
        " iv_data_access_role_arn = 'arn:aws:iam::123456789012:role/
HealthLakeImportRole'
        oo_result = lo_hll->startfhirimportjob(
          iv_jobname = iv_job_name
          io_inputdataconfig = NEW /aws1/cl_hllinputdataconfig( iv_s3uri =
 iv_input_s3_uri )
          io_joboutputdataconfig = NEW /aws1/cl_hlloutputdataconfig(
            io_s3configuration = NEW /aws1/cl_hlls3configuration(
              iv_s3uri = iv_job_output_s3_uri
              iv_kmskeyid = iv_kms_key_id
            )
          )
          iv_dataaccessrolearn = iv_data_access_role_arn
          iv_datastoreid = iv_datastore_id
```

```
        ).
        DATA(lv_job_id) = oo_result->get_jobid( ).
        MESSAGE |Import job started with ID { lv_job_id }.| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllthrottlingex INTO DATA(lo_throttling_ex).
        lv_error = |Throttling error: { lo_throttling_ex->av_err_code }-
{ lo_throttling_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_throttling_ex.
      CATCH /aws1/cx_hllaccessdeniedex INTO DATA(lo_access_ex).
        lv_error = |Access denied: { lo_access_ex->av_err_code }-{ lo_access_ex-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_access_ex.
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスのStartFHIRImportJob」を参照してください。

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

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

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

CLI

AWS CLI

データストアにタグを追加するには

次の tag-resource の例では、データストアにタグを追加する方法を示します。

```
aws healthlake tag-resource \
    --resource-arn "arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/0725c83f4307f263e16fd56b6d8ebdbe" \
```

```
    --tags '[{"Key": "key1", "Value": "value1"}]'
```

このコマンドでは何も出力されません。

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

Python

SDK for Python (Boto3)

```
   @classmethod
   def from_client(cls) -> "HealthLakeWrapper":
       """
       Creates a HealthLakeWrapper instance with a default AWS HealthLake
client.

       :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
       """
       health_lake_client = boto3.client("healthlake")
       return cls(health_lake_client)


   def tag_resource(self, resource_arn: str, tags: list[dict[str, str]]) ->
None:
       """
       Tags a HealthLake resource.
       :param resource_arn: The resource ARN.
       :param tags: The tags to add to the resource.
       """
       try:
           self.health_lake_client.tag_resource(ResourceARN=resource_arn,
Tags=tags)
       except ClientError as err:
           logger.exception(
               "Couldn't tag resource %s. Here's why %s",
               resource_arn,
               err.response["Error"]["Message"],
           )
           raise
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「TagResource」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    TRY.
        " iv_resource_arn = 'arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        lo_hll->tagresource(
          iv_resourcearn = iv_resource_arn
          it_tags = it_tags
        ).
        MESSAGE 'Resource tagged successfully.' TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスのTagResource」を参照してください。

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

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

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

CLI

AWS CLI

データストアからタグを削除するには

次の untag-resource の例では、データストアからタグを削除する方法を示します。

```
aws healthlake untag-resource \
    --resource-arn "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
b91723d65c6fdeb1d26543a49d2ed1fa" \
    --tag-keys '["key1"]'
```

このコマンドでは何も出力されません。

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

Python

SDK for Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
client.
```

```
        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def untag_resource(self, resource_arn: str, tag_keys: list[str]) -> None:
        """
        Untags a HealthLake resource.
        :param resource_arn: The resource ARN.
        :param tag_keys: The tag keys to remove from the resource.
        """
        try:
            self.health_lake_client.untag_resource(
                ResourceARN=resource_arn, TagKeys=tag_keys
            )
        except ClientError as err:
            logger.exception(
                "Couldn't untag resource %s. Here's why %s",
                resource_arn,
                err.response["Error"]["Message"],
            )
            raise
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「UntagResource」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    TRY.
        " iv_resource_arn = 'arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        lo_hll->untagresource(
          iv_resourcearn = iv_resource_arn
          it_tagkeys = it_tag_keys
        ).
        MESSAGE 'Resource untagged successfully.' TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP UntagResource」を参照してください。

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

統合 AWS HealthLake

以下の AWS サービスは と直接統合 AWS HealthLake され、統合された自然言語処理、SQL クエリ、およびデータウェアハウスを有効にします。

- Amazon Comprehend Medical は、HIPAA 対応の自然言語処理サービス (NLP) であり、機械学習ライブラリを使用して HealthLake の非構造化医療テキストから意味のある医療データを抽出します。詳細については、Amazon Comprehend Medical デベロッパーガイド」を参照してください。
- Amazon Athena は、標準 SQL を使用して Amazon Simple Storage Service (Amazon S3) バケットで HealthLake データを直接分析できるインタラクティブなクエリサービスです。詳細については、Amazon Athena デベロッパーガイド」を参照してください。

トピック

HealthLake の統合自然言語処理 (NLP)

AWS HealthLake は、FHIR `DocumentReference`リソースタイプに保存されている非構造化データを解析、識別、マッピングするための統合された自然言語処理 (NLP) ライブラリを提供します。

> [重要]
>
> HealthLake の統合 NLP はデフォルトでオフになっています。有効にするには、 を使用してサポートケースを送信しますAWS Support Center Console。ケースを作成するには、 にログイン AWS アカウント し、ケースの作成を選択します。

HealthLake 統合 NLP は、Amazon Comprehend Medical `DetectEntities-V2`、`InferICD10-CM`、および `InferRxNorm` API アクションを呼び出すことで機能します。アクションは、結果を`DocumentReference`リソースの拡張機能として追加します。Amazon Comprehend Medical API アクションが `SIGN`、`SYMPTOM`、および/または である特性を検出すると`DIAGNOSIS`、FHIR `Linkage`リソースが生成されます。新しい `Condition` および `Observation`リソースは、`SIGN`、、または の特性で識別されたエンティティから作成`DIAGNOSIS`され`SYMPTOM`、`Linkage`リソースを使用してソースドキュメントにリンクされます。

> **Note**
>
> HealthLake 統合 NLP によって生成された FHIR リソースではGETリクエストがサポートされていますが、FHIR API search機能はサポートされていません。HealthLake と Athena の統合を使用した NLP 拡張機能の検索の詳細については、「」を参照してくださいSQL インデックスとクエリ。

目次

HealthLake 統合 NLP ライブラリ

HealthLake は、Amazon Comprehend Medical ライブラリを使用してDocumentReference、リソースタイプで見つかったデータを推測します。Amazon Comprehend Medical API オペレーション DetectEntities-V2、InferICD10-CM、 は、病状を特性としてInferRxNorm検出します。オペレーションごとに異なるインサイトが得られます。

> **⚠ 言語サポート**
>
> Amazon Comprehend Medical API オペレーションは、英語テキストの医療エンティティのみを検出します。

- DetectEntities-V2: さまざまな医療エンティティの臨床テキストを検査し、エンティティカテゴリ、場所、信頼スコアなど、それらに関する特定の情報を返します。
- InferICD10-CM: 患者レコード内の病状をエンティティとして検出し、世界保健機関 (WHO) の承認の下で、CDC の National Center for Health Statistics から ICD-10-CM ナレッジベースの正規化された概念識別子にそれらのエンティティをリンクします。
- InferRxNorm: 患者レコードにリストされているエンティティとして薬剤を検出し、国立医学図書館の RxNorm データベース内の正規化された概念識別子にリンクします。

各 API オペレーションでサポートされている特性は、SIGN、SYMPTOM、および ですDIAGNOSIS。特性が検出されると、それらは HealthLake データストア内の異なる場所に FHIR 準拠の拡張機能として追加されます。

拡張機能が追加される場所。

- DocumentReference: Amazon Comprehend Medical API オペレーションの結果は、DocumentReferenceリソースタイプ内の各ドキュメントextensionに として追加されます。拡張機能の結果は 2 つのグループに分割されます。これらは、 に基づいて結果で確認できますURL。
  - http://healthlake.amazonaws.com/system-generated-resources/
    - これらは、HealthLake によって作成または追加されたリソースタイプです。
  - http://healthlake.amazonaws.com/aws-cm/
    - ここで、Amazon Comprehend Medical API オペレーションの raw 出力が HealthLake データストアに追加されます。
- Linkage: このリソースタイプは、統合された NLP の結果として追加または作成されます。特定の に対するGETリクエストは、リンクされたリソースのリストLinkageを返します。Linkage が HealthLake によって追加されたかどうかを確認するには、追加された"tag": [{"display": "SYSTEM_GENERATED"}]キーと値のペアを探します。Linkage の FHIR 仕様の詳細については、FHIR R4 ドキュメントLinkageの「」を参照してください。
- Amazon Comprehend Medical オペレーションの結果として生成された FHIR リソースタイプ。
  - Observation: には、Amazon Comprehend Medical API アクションの結果DetectEntities-V2と、特性が SIGNまたは InferICD10-CMの場合が含まれますSYMPTOM。
  - Condition: には、Amazon Comprehend Medical API アクションの結果DetectEntities-V2と、特性が InferICD10-CMの場合の結果が含まれますDIAGNOSIS。
  - MedicationStatement: には、Amazon Comprehend Medical API アクション の結果が含まれますInferRxNorm。

FHIR REST API インタラクションの使用

デフォルトでは、Amazon Comprehend Medical API オペレーションによって検出された特性は、GETリクエストの実行時に返されません。統合された NLP オペレーションの結果を表示するには、次の FHIR リソースタイプIDに既知の を指定する必要があります。

- Linkage

- `Observation`
- `Condition`
- `MedicationStatement`

`DocumentReference` リソースタイプ外の HealthLake 統合 NLP アクションの結果は、指定された `ID` に Amazon Comprehend Medical API オペレーションの結果が含まれていることがわかっているGETリクエストを使用して使用できます。

HealthLake 統合 NLP の検索パラメータ

次の表に、HealthLake 統合 NLP の検索可能な属性を示します。

HealthLake NLP の検索パラメータ

検索パラメータ	の一致を検索します
detectEntities-entity-category	CM 拡張機能内の DetectEntities AWS サブ拡張機能内のエンティティカテゴリ
detectEntities-entity-text	CM 拡張機能内の DetectEntities AWS サブ拡張機能内のエンティティテキスト
detectEntities-entity-type	CM 拡張機能内の DetectEntities AWS サブ拡張機能内のエンティティタイプ
detectEntities-entity-score	CM 拡張機能内の DetectEntities AWS サブ拡張機能内のエンティティスコア
infer-icd10cm-entity-text	AWS CM 拡張機能内の InferICD10CM サブ拡張内のエンティティテキスト
infer-icd10cm-entity-score	AWS CM 拡張内の InferICD10CM サブ拡張内のエンティティスコア
infer-icd10cm-entity-concept-code	AWS CM 拡張機能内の InferICD10CM サブ拡張機能内のエンティティ概念コード
infer-icd10cm-entity-concept-description	CM 拡張機能内の InferICD10CM サブ拡張機能内の AWS エンティティ概念の説明

検索パラメータ	の一致を検索します
infer-icd10cm-entity-concept-score	AWS CM 拡張機能内の InferICD10CM サブ拡張機能内のエンティティ概念スコア
infer-rxnorm-entity-score	AWS CM 拡張機能内の InferRxNorm サブ拡張内のエンティティスコア
infer-rxnorm-entity-text	AWS CM 拡張機能内の InferRxNorm サブ拡張内のエンティティテキスト
infer-rxnorm-entity-concept-code	AWS CM 拡張機能内の InferRxNorm サブ拡張内のエンティティ概念コード
infer-rxnorm-entity-concept-description	CM 拡張機能内の InferRxNorm サブ拡張内の AWS エンティティ概念の説明
infer-rxnorm-entity-concept-score	AWS CM 拡張機能内の InferRxNorm サブ拡張内のエンティティ概念スコア

HealthLake は、 `EntityText`と が同じエンティティの一部`EntityCategory`である条件に一致するように特別な検索を提供します。次の表は、HealthLake でサポートされている特別な検索パラメータを示しています。

検索パラメータ

検索パラメータ	返される一致
detectEntities-entity-text-category	DetectEntities サブ拡張機能に entityText と entityCategory の両方に一致するエンティティが少なくとも 1 つある場合。
detectEntities-entity-type-score	DetectEntities サブ拡張機能に entityType と entityScore の両方に一致するエンティティが少なくとも 1 つある場合。
detectEntities-entity-text-score	DetectEntities サブ拡張機能に entityText と entityScore の両方に一致するエンティティが少なくとも 1 つある場合。

検索パラメータ	返される一致
detectEntities-entity-text-type	DetectEntities サブ拡張機能に entityText と entityType の両方に一致するエンティティが少なくとも 1 つある場合。
detectEntities-entity-category-score	entityCategory と entityScore の両方に一致するエンティティが少なくとも 1 つある場合。
infer-icd10cm-entity-text-concept-code	InferICD10CM サブ拡張に entityText に一致するエンティティが少なくとも 1 つあり、そのエンティティのコードに一致する conceptCode が少なくとも 1 つある場合。 entityText
infer-icd10cm-entity-text-concept-score	InferICD10CM サブ拡張に entityText に一致するエンティティが少なくとも 1 つあり、そのエンティティの conceptScore がスコアに一致するエンティティが少なくとも 1 つある場合。 entityText
infer-icd10cm-entity-concept-description-concept-score	InferICD10CM サブ拡張のエンティティ内に、概念の説明と conceptScore に一致する概念が少なくとも 1 つある場合。 conceptScore
infer-rxnorm-entity-text-concept-code	InferRxNorm サブ拡張に entityText に一致するエンティティが少なくとも 1 つあり、そのエンティティのコードに一致する conceptCode が少なくとも 1 つある場合。 entityText
infer-rxnorm-entity-text-concept-score	InferRxNorm サブ拡張機能に entityText に一致するエンティティが少なくとも 1 つあり、そのエンティティにスコアに一致する conceptScore が少なくとも 1 つある場合。 entityText
infer-rxnorm-entity-concept-description-concept-score	InferRxNorm サブ拡張のエンティティ内に、概念の説明と conceptScore に一致する概念が少なくとも 1 つある場合。

HealthLake 統合 NLP リクエストの例

例 1: HealthLake データストアに取り込まれた**Patient**レコード

以下は、医療専門家とのPatient遭遇に基づく臨床記録の例です。

> **合成データ**
>
> 次の例のテキストは合成コンテンツであり、保護された医療情報 (PHI) は含まれていません。

```
1991-08-31

# Chief Complaint
- Headache
- Sinus Pain
- Nasal Congestion
- Sore Throat
- Pain with Bright Lights
- Nasal Discharge
- Cough

# History of Present Illness
Jerónimo599 is a 4 month-old non-hispanic white male.

# Social History
Patient has never smoked.

Patient comes from a middle socioeconomic background.

Patient currently has Aetna.

# Allergies
No Known Allergies.

# Medications
No Active Medications.

# Assessment and Plan
```

```
Patient is presenting with bee venom (substance), mold (organism), house dust
 mite (organism), animal dander (substance), grass pollen (substance), tree pollen
 (substance), lisinopril, sulfamethoxazole / trimethoprim, fish (substance).

## Plan

The patient was prescribed the following medications:
- astemizole 10 mg oral tablet
- nda020800 0.3 ml epinephrine 1 mg/ml auto-injector
The patient was placed on a careplan:
- self-care interventions (procedure)
```

注意点として、この情報は DocumentReferenceリソースの base64 形式でエンコードされます。このドキュメントが HealthLake に取り込まれ、Amazon Comprehend Medical API オペレーションが完了すると、結果を表示するには、DocumentReferenceリソースタイプのGETリクエストから開始できます。

```
GET https://https://healthlake.region.amazonaws.com/datastore/datastoreId/
r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/DocumentReference
```

Amazon Comprehend Medical API オペレーションが成功したら、以下extensionにリンクされた内でこれらのキーと値のペアを探します。 "url": "http://healthlake.amazonaws.com/aws-cm/"

```
{
  "url": "http://healthlake.amazonaws.com/aws-cm/status/",
  "valueString": "SUCCESS"
 },
 {
  "url": "http://healthlake.amazonaws.com/aws-cm/message/",
  "valueString": "The AWS HealthLake integrated medical NLP operation was successful."
 }
```

次のタブは、リソースタイプに基づいて、取り込まれた医療記録が HealthLake データストアにどのように報告されるかを示しています。

DocumentReference

単一のDocumentReferenceリソースタイプの結果を表示するには、特定のリソースidの が提供されているGETリクエストを行います。

```
GET https://https://healthlake.region.amazonaws.com/datastore/datastoreId/
r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/DocumentReference/0e938f03-da7f-4178-acd8-
eea9586c46ed
```

成功すると、200HTTP レスポンスコードと次の JSON レスポンス (わかりやすくするために切り捨てられています) を取得します。

以下にそのhttp://healthlake.amazonaws.com/system-generated-resources/部分を示します。新しい Linkage/*e366d29f-2c22-4c19-866e-09603937935a* が追加されたことを確認できます。また、HealthLake が特定の ObservationおよびConditionリソースタイプに推論ベースの検出結果を追加した場所を確認することもできます。

これらのリソースタイプがどのように修正されたかを確認するには、関連するタブを選択します。


```
{
  "extension": [
   {
    "url": "http://healthlake.amazonaws.com/linkage",
    "valueReference": {
     "reference": "Linkage/e366d29f-2c22-4c19-866e-09603937935a"
    }
   },
   {
    "url": "http://healthlake.amazonaws.com/nlp-entity",
    "valueReference": {
     "reference": "Observation/c6e0a3ff-7a17-4d8b-bfd0-d02d7da090c5"
    }
   },
   {
    "url": "http://healthlake.amazonaws.com/nlp-entity",
    "valueReference": {
     "reference": "Condition/0854e1f3-894d-448e-a8d9-3af5b9902baf"
    }
   }
  ],
  "url": "http://healthlake.amazonaws.com/system-generated-resources/"
 }
```

Linkage

単一のLinkageリソースタイプの結果を表示するには、特定のリソースIDの が提供されているGETリクエストを行います。

```
GET https://https://healthlake.region.amazonaws.com/datastore/datastoreId/
r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/Linkage/e366d29f-2c22-4c19-866e-09603937935a
```

成功すると、HTTP 200 レスポンスコードと、次の切り捨てられた JSON レスポンスを取得します。

レスポンスには item要素が含まれています。キーと値のペアは、 の変更に使用され、"type": "alternate"キーConditionと値のペアの下にObservationsリストされている特定のDocumentReferenceエントリ"type": "source"を示します。

また、 *meta*要素、およびこれらのリソースが HealthLake によって作成された*"tag": [{"display": "SYSTEM_GENERATED"}]*ことを示す対応するキーと値のペア も表示されます。

```
{
  "resourceType": "Linkage",
  "id": "e366d29f-2c22-4c19-866e-09603937935a",
  "active": true,
  "item":
  [
   {
    "type": "alternate",
    "resource": {
     "reference": "Observation/c6e0a3ff-7a17-4d8b-bfd0-d02d7da090c5",
     "type": "Observation"
    }
   },
   {
    "type": "alternate",
    "resource": {
     "reference": "Condition/9d5c1ef6-f822-4faf-b55f-7c70f2a4aa8d",
     "type": "Condition"
    }
   },
   {
    "type": "source",
    "resource": {
```

```
    "reference": "DocumentReference/0e938f03-da7f-4178-acd8-eea9586c46ed",
    "type": "DocumentReference"
   }
  }
 ],
 "meta": {
  "lastUpdated": "2022-10-21T19:38:31.327Z",
  "tag": [{
   "display": "SYSTEM_GENERATED"
  }]
 }
}
```

Resource type: Observation

単一のObservationリソースタイプの結果を表示するには、特定のリソースIDの が提供されているGETリクエストを行います。

```
GET https://https://healthlake.region.amazonaws.com/
datastore/datastoreId/r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/
Observation/e366d29f-2c22-4c19-866e-09603937935a
```

Amazon Comprehend Medical API オペレーションの結果はcode、、meta、および の要素に変更されますmodifierExtension。

code

タイプの 要素CodeableConcept。詳細については、FHIR R4 ドキュメントCodeableConceptの「」を参照してください。

HealthLake は、次の 3 つのキーと値のペアを追加します。

- "system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/": URL は特定の Amazon Comprehend Medical API オペレーションを参照します。この場合、InferICD10CM。
- "code": "A52.06": A52.06は、Centers for Disease Control のナレッジベースにある概念を識別する ICD-10-CM コードです。
- "display": "Other syphilitic heart involvement": "Other syphilitic heart involvement"はオントロジー内の ICD-10-CM コードの長い説明です。

次の切り捨てられた JSON レスポンスには、 code要素のみが含まれます。

```
"code": {
  "coding":
  [
    {
   "system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/",
   "code": "A52.06",
   "display": "Other syphilitic heart involvement"
    }
  ],
  "text": "Other syphilitic heart involvement"
 }
```

割り当てられた ICD-10-CM コードが正しいというモデルの信頼度を理解するには、modifierExtension要素を使用します。

meta

meta 要素には、Amazon Comprehend Medical API オペレーションによって追加された詳細がcode要素に含まれているかどうかを示すメタデータが含まれています。

次の切り捨てられた JSON レスポンスには、 meta要素のみが含まれます。

```
"meta": {
  "lastUpdated": "2022-10-21T19:38:30.879Z",
  "tag": [{
   "display": "SYSTEM_GENERATED"
  }]
 }
```

modifierExtension

modifierExtension 要素には、 code要素で見つかった割り当てられたコードの信頼度に関する詳細が含まれます。また、結果の生成に使用される元の DocumentReference へのリンクと関連する Linkage リソースタイプを提供するキーと値のペアもあります。

追加されるcoding要素ごとに、modifierExtension に entity-scoreと entity-Concept-Scoreが追加されます。キーと値のペアの値ごとに、スコアが表示されます。の場合entity-score、このスコアは Amazon Comprehend Medical が検出の精度に対して持っている信頼度です。の場合entity-Concept-Score、このスコアは、エンティティが ICD-10-CM 概念に正確にリンクされているという Amazon Comprehend Medical の信頼度です。

次の切り捨てられた JSON レスポンスには、 modifierExtension要素のみが含まれます。

```
"modifierExtension": [{
   "url": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/aws-cm-icd10-entity-
score",
   "valueDecimal": 0.45005733
  },
  {
   "url": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/aws-cm-icd10-entity-
Concept-Score",
   "valueDecimal": 0.1111792
  },
  {
   "url": "http://healthlake.amazonaws.com/system-generated-linkage",
   "valueReference": {
    "reference": "Linkage/e366d29f-2c22-4c19-866e-09603937935a"
   }
  },
  {
   "url": "http://healthlake.amazonaws.com/source-document-reference",
   "valueReference": {
    "reference": "DocumentReference/0e938f03-da7f-4178-acd8-eea9586c46ed"
   }
  }
 ]
```

完全な JSON レスポンス

```
{
  "subject": {
   "reference": "Patient/0679b7b7-937d-488a-b48d-6315b8e7003b"
  },
  "resourceType": "Observation",
  "status": "unknown",
  "code": {
   "coding": [{
    "system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/",
    "code": "A52.06",
    "display": "Other syphilitic heart involvement"
   }],
   "text": "Other syphilitic heart involvement"
  },
  "meta": {
   "lastUpdated": "2022-10-21T19:38:30.879Z",
   "tag": [{
```

```
    "display": "SYSTEM_GENERATED"
   }]
  },
  "modifierExtension": [{
    "url": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/aws-cm-icd10-entity-
score",
    "valueDecimal": 0.45005733
   },
   {
    "url": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/aws-cm-icd10-entity-
Concept-Score",
    "valueDecimal": 0.1111792
   },
   {
    "url": "http://healthlake.amazonaws.com/system-generated-linkage",
    "valueReference": {
     "reference": "Linkage/e366d29f-2c22-4c19-866e-09603937935a"
    }
   },
   {
    "url": "http://healthlake.amazonaws.com/source-document-reference",
    "valueReference": {
     "reference": "DocumentReference/0e938f03-da7f-4178-acd8-eea9586c46ed"
    }
   }
  ],
  "id": "7e88c7c5-21a5-4dd7-8fc2-a02474fba583"
 }
```


Condition

単一のConditionリソースタイプの結果を表示するには、特定のリソースIDの が提供されているGETリクエストを行います。

```
GET https://https://healthlake.region.amazonaws.com/datastore/datastoreId/
r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/Condition/b06d343d-
ddb8-4f36-82cb-853fcd434dfd
```

Amazon Comprehend Medical API オペレーションの結果はcode、、meta、および の要素に変更されますmodifierExtension。

code

タイプの 要素CodeableConcept。詳細については、FHIR R4 ドキュメントCodeableConceptの「」を参照してください。

HealthLake は、次の 3 つのキーと値のペアを追加します。

- "system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/": URL は特定の Amazon Comprehend Medical API オペレーションを参照します。この場合、InferICD10CM。
- "code": "I70.0": A52.06は、Centers for Disease Control のナレッジベースにある概念を識別する ICD-10-CM コードです。
- "display": "Atherosclerosis of aorta": "Other syphilitic heart involvement"はオントロジー内の ICD-10-CM コードの長い説明です。

次の切り捨てられた JSON レスポンスには、 code要素のみが含まれます。

```
"code": {
  "coding":
  [
    {
   "system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/",
   "code": "I70.0",
   "display": "Atherosclerosis of aorta"
    }
  ],
  "text": "Atherosclerosis of aorta"
 }
```

割り当てられた ICD-10-CM コードが正しいというモデルの信頼度を理解するには、modifierExtension要素を使用します。

meta

meta 要素には、Amazon Comprehend Medical API オペレーションによって追加された詳細がcode要素に含まれているかどうかを示すメタデータが含まれています。

次の切り捨てられた JSON レスポンスには、 meta要素のみが含まれます。

```
"meta": {
  "lastUpdated": "2022-10-21T19:38:30.877Z",
```

```
  "tag": [{
   "display": "SYSTEM_GENERATED"
  }]
 }
```

modifierExtension

modifierExtension 要素には、 code要素で見つかった割り当てられたコードの信頼度に関する詳細が含まれます。また、結果の生成に使用される元の DocumentReference へのリンクと関連する Linkage リソースタイプを提供するキーと値のペアもあります。

追加されるcoding要素ごとに、modifierExtension に entity-scoreと entity-Concept-Scoreが追加されます。キーと値のペアの値ごとに、スコアが表示されます。の場合entity-score、このスコアは Amazon Comprehend Medical が検出の精度に対して持っている信頼度です。の場合entity-Concept-Score、このスコアは、エンティティが ICD-10-CM 概念に正確にリンクされているという Amazon Comprehend Medical の信頼度です。

次の切り捨てられた JSON レスポンスには、 modifierExtension要素のみが含まれます。

```
"modifierExtension": [{
   "url": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/aws-cm-icd10-entity-
score",
   "valueDecimal": 0.94417894
  },
  {
   "url": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/aws-cm-icd10-entity-
Concept-Score",
   "valueDecimal": 0.8458298
  },
  {
   "url": "http://healthlake.amazonaws.com/system-generated-linkage",
   "valueReference": {
    "reference": "Linkage/e366d29f-2c22-4c19-866e-09603937935a"
   }
  },
  {
   "url": "http://healthlake.amazonaws.com/source-document-reference",
   "valueReference": {
    "reference": "DocumentReference/0e938f03-da7f-4178-acd8-eea9586c46ed"
   }
  }
 ]
```

完全な JSON レスポンス


```
{
  "subject": {
   "reference": "Patient/0679b7b7-937d-488a-b48d-6315b8e7003b"
  },
  "resourceType": "Condition",
  "code": {
   "coding": [{
    "system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/",
    "code": "I70.0",
    "display": "Atherosclerosis of aorta"
   }],
   "text": "Atherosclerosis of aorta"
  },
  "meta": {
   "lastUpdated": "2022-10-21T19:38:30.877Z",
   "tag": [{
    "display": "SYSTEM_GENERATED"
   }]
  },
  "modifierExtension": [{
    "url": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/aws-cm-icd10-entity-
score",
    "valueDecimal": 0.94417894
   },
   {
    "url": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/aws-cm-icd10-entity-
Concept-Score",
    "valueDecimal": 0.8458298
   },
   {
    "url": "http://healthlake.amazonaws.com/system-generated-linkage",
    "valueReference": {
     "reference": "Linkage/e366d29f-2c22-4c19-866e-09603937935a"
    }
   },
   {
    "url": "http://healthlake.amazonaws.com/source-document-reference",
    "valueReference": {
     "reference": "DocumentReference/0e938f03-da7f-4178-acd8-eea9586c46ed"
    }
   }
  ],
```

```
  "id": "b06d343d-ddb8-4f36-82cb-853fcd434dfd"
 }
```

例 2: **MedicationStatement**リソースタイプ**DocumentReference**を含む

以下は、患者の医療専門家との出会いに基づく臨床メモの例です。

> ⚠ 合成データ
>
> この例のテキストは合成コンテンツであり、保護された医療情報 (PHI) は含まれていません。

```
Tom is not prescribed Advil
```

次のタブは、リソースタイプに基づいて、取り込まれた医療記録が HealthLake データストアにどのように報告されるかを示しています。

DocumentReference

単一のDocumentReferenceリソースタイプの結果を表示するには、特定のリソースIDの が提供されているGETリクエストを行います。

```
GET https://https://healthlake.region.amazonaws.com/datastore/datastoreId/
r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/DocumentReference/c549125d-a218-421f-
b8bf-23614c5e796c
```

成功すると、HTTP 200 レスポンスコードと、次の切り捨てられた JSON レスポンスを取得します。

キーと値のペア は"url": "http://healthlake.amazonaws.com/system-generated-resources/"、この中のリソースタイプextensionが Amazon Comprehend Medical API オペレーションによって追加されたことを示します。新しいLinkageリソースタイプと複数のMedicationStatementリソースを表示できます。

```
"extension": [{
   "extension": [{
     "url": "http://healthlake.amazonaws.com/linkage",
     "valueReference": {
      "reference": "Linkage/394bb244-177b-4409-8657-26b20ed56dd7"
```

```
      }
     },
     {
      "url": "http://healthlake.amazonaws.com/nlp-entity",
      "valueReference": {
       "reference": "MedicationStatement/cbf6af10-b0b9-451c-bdde-99611e3498a8"
      }
     },
     {
      "url": "http://healthlake.amazonaws.com/nlp-entity",
      "valueReference": {
       "reference": "MedicationStatement/9a89b0d3-6681-45ca-9926-27951edce5c7"
      }
     },
     {
      "url": "http://healthlake.amazonaws.com/nlp-entity",
      "valueReference": {
       "reference": "MedicationStatement/4a01f6c8-5f3a-4122-80ab-405312f96aa2"
      }
     },
     {
      "url": "http://healthlake.amazonaws.com/nlp-entity",
      "valueReference": {
       "reference": "MedicationStatement/fbfb77d8-70cf-4579-b4c0-d6fe3c01656b"
      }
     },
     {
      "url": "http://healthlake.amazonaws.com/nlp-entity",
      "valueReference": {
       "reference": "MedicationStatement/1340c9ce-9c48-4bf9-9b2f-d0ab027f5e0b"
      }
     }
    ],
    "url": "http://healthlake.amazonaws.com/system-generated-resources/"
   }
```

Linkage

単一のLinkageリソースタイプの結果を表示するには、特定のリソースIDの が提供されているGETリクエストを行います。

```
GET https://https://healthlake.region.amazonaws.com/datastore/datastoreId/
r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/Linkage/394bb244-177b-4409-8657-26b20ed56dd7
```

成功すると、HTTP 200 レスポンスコードと次の JSON レスポンスを取得します。

レスポンスには item要素が含まれています。キーと値のペアは、MedicationStatementリソースタイプの変更に使用される特定のDocumentReferenceエントリ"type": "source"を示します。

また、 meta要素と対応するキーと値のペア を表示して"tag": [{"display": "SYSTEM_GENERATED"}]、これらのリソースが HealthLake によって作成されたことを示すこともできます。


```
{
  "resourceType": "Linkage",
  "id": "394bb244-177b-4409-8657-26b20ed56dd7",
  "active": true,
  "item": [{
    "type": "alternate",
    "resource": {
     "reference": "MedicationStatement/cbf6af10-b0b9-451c-bdde-99611e3498a8",
     "type": "MedicationStatement"
    }
   },
   {
    "type": "alternate",
    "resource": {
     "reference": "MedicationStatement/9a89b0d3-6681-45ca-9926-27951edce5c7",
     "type": "MedicationStatement"
    }
   },
   {
    "type": "alternate",
    "resource": {
     "reference": "MedicationStatement/4a01f6c8-5f3a-4122-80ab-405312f96aa2",
     "type": "MedicationStatement"
    }
   },
   {
    "type": "alternate",
    "resource": {
     "reference": "MedicationStatement/fbfb77d8-70cf-4579-b4c0-d6fe3c01656b",
     "type": "MedicationStatement"
    }
   },
   {
```

```
    "type": "alternate",
    "resource": {
     "reference": "MedicationStatement/1340c9ce-9c48-4bf9-9b2f-d0ab027f5e0b",
     "type": "MedicationStatement"
    }
   },
   {
    "type": "source",
    "resource": {
     "reference": "DocumentReference/c549125d-a218-421f-b8bf-23614c5e796c",
     "type": "DocumentReference"
    }
   }
  ],
  "meta": {
   "lastUpdated": "2022-10-24T20:05:03.501Z",
   "tag": [{
    "display": "SYSTEM_GENERATED"
   }]
  }
 }
```

MedicationStatement

単一のMedicationStatementリソースタイプの結果を表示するには、特定のリソースIDの が提供されているGETリクエストを行います。

```
GET https://https://healthlake.region.amazonaws.com/
datastore/datastoreId/r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/
MedicationStatement/9a89b0d3-6681-45ca-9926-27951edce5c7
```

MedicationStatement リソースタイプは、Amazon Comprehend Medical InferRxNorm API オペレーションの結果が見つかった場所です。結果は、、medicationCodeableConcept、metaおよび の要素に修正されますmodifierExtension。

medicationCodeableConcept

タイプの 要素CodeableConcept。詳細については、FHIR R4 ドキュメントCodeableConceptの「」を参照してください。

HealthLake は、次の 3 つのキーと値のペアを追加します。

- "system": ""http://healthlake.amazonaws.com/aws-cm/infer-rxnorm/: URL は特定の Amazon Comprehend Medical API オペレーションを参照します。この場合、InferRxNorm。
- "code": "731533": 731533は、RxCUI とも呼ばれる RxNorm 概念 ID です。 RxCUI
- "display": "ibuprofen 200 MG Oral Capsule [Advil]": ここで、 ibuprofen 200 MG Oral Capsule [Advil]は RxNorm 概念の説明です。

次の切り捨てられた JSON レスポンスには、 MedicationStatement要素のみが含まれます。

```
"medicationCodeableConcept": {
  "coding": [
   {
    "system": "http://healthlake.amazonaws.com/aws-cm/infer-rxnorm/",
    "code": "731533",
    "display": "ibuprofen 200 MG Oral Capsule [Advil]"
   }
  ]
 }
```

meta

meta 要素には、Amazon Comprehend Medical API オペレーションによって追加された詳細がcode要素に含まれているかどうかを示すメタデータが含まれています。

次の切り捨てられた JSON レスポンスには、 meta要素のみが含まれます。

```
"meta": {
  "lastUpdated": "2022-10-24T20:05:02.800Z",
  "tag": [
   {
    "display": "SYSTEM_GENERATED"
   }
  ]
 }
```

modifierExtension

modifierExtension 要素には、結果の生成DocumentReferenceに使用される元の および関連する Linkage リソースタイプへのリンクを提供するキーと値のペアが含まれています。

```
"modifierExtension": [
   {
    "url": "http://healthlake.amazonaws.com/system-generated-linkage",
    "valueReference": {
     "reference": "Linkage/394bb244-177b-4409-8657-26b20ed56dd7"
    }
   },
   {
    "url": "http://healthlake.amazonaws.com/source-document-reference",
    "valueReference": {
     "reference": "DocumentReference/c549125d-a218-421f-b8bf-23614c5e796c"
    }
   }
 ]
```

Amazon Athena を使用した HealthLake データのクエリ

HealthLake インポートジョブ中、ネストされた FHIR JSON データは ETL プロセスを受け、Apache Iceberg オープンテーブル形式で保存されます。ここで、各 FHIR リソースタイプは Athena の個々のテーブルとして表されます。これにより、ユーザーは SQL を使用して FHIR データをクエリできますが、最初にエクスポートする必要はありません。これは、臨床医や科学者が FHIR データをクエリして決定を検証したり、研究を進めたりできるため、価値があります。Athena での Apache Iceberg テーブルの機能の詳細については、「Athena ユーザーガイド」の「Query Apache Iceberg tables」を参照してください。

> **Note**
>
> HealthLake は、Athena の HealthLake データでの FHIR R4 `read`インタラクションをサポートしています。詳細については、「FHIR リソースの読み取り」を参照してください。

このセクションのトピックでは、HealthLake データストアを Athena に接続する方法、SQL を使用してクエリを実行する方法、結果を他の AWS のサービスに接続して詳細な分析を行う方法について説明します。

トピック

- Amazon Athena の開始方法
- SQL を使用した HealthLake データのクエリ

- 複雑なフィルタリングを使用した SQL クエリの例

Amazon Athena の開始方法

HealthLake を Amazon Athena と統合するには、アクセス許可を設定する必要があります。これを行うには、Athena ユーザー、グループ、またはロールを作成し、HealthLake データストア内にある FHIR リソースへのアクセスを許可します。

- ユーザー、グループ、またはロールに HealthLake データストアへのアクセスを許可する (AWS Lake Formation コンソール)
- Athena アカウントのセットアップ

ユーザー、グループ、またはロールに HealthLake データストアへのアクセスを許可する (AWS Lake Formation コンソール)

> ペルソナ: HealthLake 管理者
>
> HealthLake 管理者ペルソナは、 AWS Lake Formation のデータレイク管理者です。Lake Formation の HealthLake データストアへのアクセスを許可します。

作成されたデータストアごとに、 AWS Lake Formation コンソールに 2 つのエントリが表示されます。1 つのエントリはリソースリンクです。リソースリンク名は常に斜体で表示されます。各リソースリンクには、リンクされた共有リソースの名前と所有者が表示されます。すべての HealthLake データストアで、共有リソース所有者は HealthLake サービスアカウントです。もう 1 つのエントリは、HealthLake サービスアカウントの HealthLake データストアです。この手順のステップでは、リソースリンクであるデータストアを使用します。

リソースリンクの詳細については、Lake Formation デベロッパーガイドの「Lake Formation でのリソースリンクの仕組みAWS 」を参照してください。

ユーザー、グループ、またはロールが Athena でデータをクエリできるようにするには、リソースデータベースに対する Describe アクセス許可を付与する必要があります。次に、テーブルに Select と Describe を付与する必要があります。

ステップ 1: HealthLake データストアリソースリンクデータベースに DESCRIBE アクセス許可を付与するには

1. AWS Lake Formation コンソールを開きます: https://console.aws.amazon.com/lakeformation/
2. プライマリナビゲーションバーで、データベースを選択します。
3. データベースページで、斜体のデータストアの名前の横にあるラジオボタンを選択します。
4. アクション (▾) を選択します。
5. [付与] を選択します。
6. データ許可の付与ページのプリンシパルで、IAM ユーザーまたはロールを選択します。
7. IAM ユーザーまたはロールで、下矢印 (▾) を使用するか、Athena でクエリを実行できるようにする IAM ユーザー、ロール、またはグループを検索します。
8. LF タグまたはカタログリソースカードで、名前付きデータカタログリソースオプションを選択します。
9. Databases で、下矢印 (▾) を使用して、アクセスを共有する HealthLake データストアデータベースを選択します。
10. リソースリンクのアクセス許可カードで、リソースリンクのアクセス許可で、説明を選択します。

権限が成功すると、権限付与の成功バナーが表示されます。先ほど付与したアクセス許可を表示するには、データレイクのアクセス許可を選択します。テーブルでユーザー、グループ、ロールを見つけます。アクセス許可列の下に、Describe が表示されます。

次に、ターゲットで Grant を使用して、データベース内のすべてのテーブルで Select と Describe を付与する必要があります。

ステップ 2: HealthLake データストアリソースリンク内のすべてのテーブルへのアクセスを許可する

1. AWS Lake Formation コンソールを開きます: https://console.aws.amazon.com/lakeformation/
2. プライマリナビゲーションバーで、データベースを選択します。
3. データベースページで、斜体のデータストアの名前の横にあるラジオボタンを選択します。
4. アクション (▾) を選択します。
5. ターゲットの付与を選択します。
6. データ許可の付与ページのプリンシパルで、IAM ユーザーまたはロールを選択します。
7. IAM ユーザーまたはロールで、下矢印 (▾) を使用するか、Athena でクエリを実行できるようにする IAM ユーザー、グループ、またはロールを検索します。

8. LF タグまたはカタログリソースカードで、名前付きデータカタログリソースオプションを選択します。
9. Databases で、下矢印 (▾) を使用して、アクセス権を付与する HealthLake データストアデータベースを選択します。
10. Tables で、すべてのテーブルを選択して、すべてのテーブルを HealthLake ユーザーと共有します。
11. テーブルのアクセス許可カードで、テーブルのアクセス許可で、説明と選択を選択します。
12. [付与] を選択します。

grant を選択すると、Grant permissions success バナーが表示されます。指定されたユーザーは、Athena の HealthLake データストアに対してクエリを実行できるようになりました。

Athena の開始方法

> **HealthLake ユーザー**
>
> HealthLake ユーザーは、Athena コンソール AWS CLI、または AWS SDKs を使用して、HealthLake 管理者によって共有されている HealthLake データストアをクエリします。

Athena を使用してデータストアをクエリするには、次の 3 つの操作を行う必要があります。

- Lake Formation 経由で HealthLake データストアへのアクセス権を IAM ユーザーまたはロールに付与します。詳細については[ユーザー、グループ、またはロールに HealthLake データストアへのアクセスを許可する (AWS Lake Formation コンソール)]を参照してください。
- HealthLake データストアのワークグループを作成します。
- クエリ結果を保存する Amazon S3 バケットを指定します。

Athena の使用を開始するには、AmazonAthenaFullAccess および AmazonS3FullAccess AWS 管理ポリシーをユーザー、グループ、またはロールに追加します。 AWS マネージドポリシーを使用すると、新しいサービスの使用を開始できます。AWS 管理ポリシーは、すべての AWS のお客様が使用できるため、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注意してください。IAM ポリシーでアクセス許可を設定するときは、タスクの実行に必要なアクセス許可のみを付与します。IAM と最小特権の適用の詳細については、「IAM ユーザーガイド」の[「最小特権のアクセス許可の適用」]を参照してください。

> **Important**
>
> Athena で HealthLake データストアをクエリするには、Athena エンジンバージョン 3 を使用する必要があります。

ワークグループはリソースであるため、IAM ベースのポリシーを使用して特定のワークグループへのアクセスを制御できます。詳細については、Athena ユーザーガイドの「ワークグループを使用してクエリのアクセスとコストを制御する」を参照してください。

ワークグループの設定の詳細については、Athena ユーザーガイドhttps://docs.aws.amazon.com/athena/latest/ug/workgroups-procedure.htmlの「」を参照してください。

> **Note**
>
> Amazon S3 バケットがあるリージョンで、Athena コンソールが一致している必要があります。

クエリを実行する前に、Amazon S3 のクエリ結果バケットの場所を指定しておく、または指定されたバケットがあり、その設定がクライアント設定を上書きするワークグループを使用する必要があります。出力ファイルは、実行されるすべてのクエリに対して自動的に保存されます。

Athena コンソールでのクエリ結果の場所の指定の詳細については、Amazon Athena ユーザーガイド Amazon Athenaの「Athena コンソールを使用したクエリ結果の場所の指定」を参照してください。

Athena で HealthLake データストアをクエリする方法の例については、「」を参照してくださいSQL を使用した HealthLake データのクエリ。

SQL を使用した HealthLake データのクエリ

FHIR データを HealthLake データストアにインポートすると、ネストされた JSON FHIR データは同時に ETL プロセスを実行し、Amazon S3 の Apache Iceberg オープンテーブル形式で保存されます。HealthLake データストアの各 FHIR リソースタイプはテーブルに変換され、Amazon Athena を使用してクエリできます。SQL ベースのクエリを使用して、テーブルを個別にクエリすることも、グループとしてクエリすることもできます。データストアの構造のため、データは複数の異なるデータ型として Athena にインポートされます。これらのデータ型にアクセスできる SQL クエリの作成の詳細については、「Amazon Athena ユーザーガイド」の「複雑な型とネストされた構造を持つ配列のクエリ」を参照してください。 Amazon Athena

> **Note**
>
> このトピックのすべての例では、Synthea を使用して作成された架空のデータを使用します。Synthea データでプリロードされたデータストアの作成の詳細については、「」を参照してくださいHealthLake データストアの作成。

リソースタイプの各要素について、FHIR 仕様はカーディナリティを定義します。要素の基数により、この要素を表示できる回数の下限と上限が定義されます。SQL クエリを構築するときは、これを考慮する必要があります。たとえば、リソースタイプ: 患者の一部の要素を見てみましょう。

- 要素: Name FHIR 仕様はカーディナリティを に設定します`0..*`。

  要素は配列としてキャプチャされます。

```
[{
 id = null,
 extension = null,
 use = official,
 _use = null,
 text = null,
 _text = null,
 family = Wolf938,
 _family = null,
 given = [Noel608],
 _given = null,
 prefix = null,
 _prefix = null,
 suffix = null,
 _suffix = null,
 period = null
}]
```

  Athena では、リソースタイプがどのように取り込まれたかを確認するには、テーブルとビューでリソースタイプを検索します。この配列の要素にアクセスするには、ドット表記を使用できます。`given` と の値にアクセスする簡単な例を次に示します`family`。

```
SELECT
    name[1].given as FirstName,
    name[1].family as LastName
```

```
FROM Patient
```

- 要素: MaritalStatus FHIR 仕様では、基数を に設定します`0..1`。

  この要素は JSON としてキャプチャされます。

```
{
 id = null,
 extension = null,
 coding = [
  {
   id = null,
   extension = null,
   system = http: //terminology.hl7.org/CodeSystem/v3-MaritalStatus,
    _system = null,
   version = null,
   _version = null,
   code = S,
   _code = null,
   display = Never Married,
   _display = null,
   userSelected = null,
   _userSelected = null
  }

 ],
 text = Never Married,
 _text = null
}
```

  Athena では、リソースタイプがどのように取り込まれたかを確認するには、テーブルとビューでリソースタイプを検索します。JSON のキーと値のペアにアクセスするには、ドット表記を使用できます。配列ではないため、配列インデックスは必要ありません。の値にアクセスする簡単な例を次に示します`text`。

```
SELECT
    maritalstatus.text as MaritalStatus
FROM Patient
```

JSON へのアクセスと検索の詳細については、Athena ユーザーガイドの「JSON のクエリ」を参照してください。

Athena データ操作言語 (DML) クエリステートメントは Trino に基づいています。Athena は Trino のすべての機能をサポートしているわけではなく、大きな違いがあります。詳細については、Amazon Athena ユーザーガイド」の「DML クエリ、関数、演算子」を参照してください。

さらに、Athena は HealthLake データストアのクエリを作成するときに発生する可能性のある複数のデータ型をサポートしています。Athena のデータ型の詳細については、Amazon Athena ユーザーガイド」の「Amazon Athena のデータ型」を参照してください。 Amazon Athena

Athena での SQL クエリの仕組みの詳細については、Amazon Athena ユーザーガイド」の「Amazon Athena の SQL リファレンス」を参照してください。 Amazon Athena

各タブには、Athena を使用して指定されたリソースタイプと関連する要素を検索する方法の例が表示されます。

Element: Extension

要素は、データストアにカスタムフィールドを作成するextensionために使用されます。

この例では、 Patientリソースタイプにある extension要素の機能にアクセスする方法を示します。

HealthLake データストアが Athena にインポートされると、リソースタイプの要素の解析は異なります。の構造elementは可変であるため、スキーマで完全に指定することはできません。この変動性を処理するために、配列内の要素は文字列として渡されます。

の表の説明にはPatient、 とextension記述されている 要素が表示されます。つまりarray<string>、インデックス値を使用して配列の 要素にアクセスできます。ただし、文字列の要素にアクセスするには、 を使用する必要がありますjson_extract。

以下は、患者テーブルにある extension要素からの 1 つのエントリです。

```
[{
  "valueString": "Kerry175 Cummerata161",
  "url": "http://hl7.org/fhir/StructureDefinition/patient-mothersMaidenName"
 },
 {
  "valueAddress": {
   "country": "DE",
   "city": "Hamburg",
   "state": "Hamburg"
  },
```

```
  "url": "http://hl7.org/fhir/StructureDefinition/patient-birthPlace"
 },
 {
  "valueDecimal": 0.0,
  "url": "http://synthetichealth.github.io/synthea/disability-adjusted-life-years"
 },
 {
  "valueDecimal": 5.0,
  "url": "http://synthetichealth.github.io/synthea/quality-adjusted-life-years"
 }
]
```

これは有効な JSON ですが、Athena はそれを文字列として扱います。

この SQL クエリの例は、 要素patient-mothersMaidenNameと patient-birthPlace要素を含むテーブルを作成する方法を示しています。これらの要素にアクセスするには、異なる配列インデックスと json_extract.

```
SELECT
    extension[1],
    json_extract(extension[1], '$.valueString') AS MothersMaidenName,
    extension[2],
    json_extract(extension[2], '$.valueAddress.city') AS birthPlace
FROM patient
```

JSON を含むクエリの詳細については、Amazon Athena ユーザーガイド」の「JSON からのデータの抽出」を参照してください。

Element: birthDate (Age)

Age は、FHIR 内の患者リソースタイプの要素ではありません。年齢に基づいてフィルタリングする検索の 2 つの例を次に示します。

age は要素ではないため、SQL クエリbirthDateには を使用します。要素が FHIR にどのように取り込まれたかを確認するには、テーブルとビューでテーブル名を検索します。文字列型であることがわかります。

例 1: 年齢の値を計算する

このサンプル SQL クエリでは、組み込みの SQL ツール current_dateと を使用して、これらのコンポーネントyearを抽出します。次に、これらを減算して、患者の実際の年齢を という列として返しますage。

```
SELECT
 (year(current_date) - year(date(birthdate))) as age
FROM patient
```

例 2: 以前に生まれ2019-01-01、 である患者のフィルタリングmale。

SQL クエリでは、 CAST関数を使用してbirthDate要素をタイプ としてキャストする方法とDATE、 WHERE句の 2 つの条件に基づいてフィルタリングする方法を示します。要素はデフォルトで型文字列として取り込まれるCASTため、型として取り込む必要がありますDATE。その後、 <演算子を使用して、別の日付 と比較できます2019-01-01。を使用するとAND、 WHERE句に 2 番目の条件を追加できます。

```
SELECT birthdate
FROM patient
-- we convert birthdate (varchar) to date  > cast that as date too
WHERE CAST(birthdate AS DATE) < CAST('2019-01-01' AS DATE) AND gender = 'male'
```

Resource type: Location

この例では、都市名がアトルボロである Location リソースタイプ内の場所を検索します。

```
SELECT *
FROM Location
WHERE address.city='ATTLEBORO'
LIMIT 10;
```

Element: Age

```
SELECT birthdate
FROM patient
-- we convert birthdate (varchar) to date  > cast that as date too
WHERE CAST(birthdate AS DATE) < CAST('2019-01-01' AS DATE) AND gender = 'male'
```

Resource type: Condition

リソースタイプ条件は、懸念レベルに達した問題に関連する診断データを保存します。HealthLake の統合医療自然言語処理 (NLP) は、DocumentReference Conditionリソースタイプで見つかった詳細に基づいて新しいリソースを生成します。新しいリソースが生成されると、HealthLake は タグを SYSTEM_GENERATED meta要素に追加します。このサンプル SQL ク

エリは、条件テーブルを検索し、結果が削除されたSYSTEM_GENERATED結果を返す方法を示しています。

HealthLake の統合自然言語処理 (NLP) の詳細については、「」を参照してくださいHealthLakeの統合自然言語処理 (NLP)。

```
SELECT *
FROM condition
WHERE meta.tag[1] is NULL
```

指定された文字列要素内で検索して、クエリをさらにフィルタリングすることもできます。modifierextension 要素には、一連の条件の生成に使用されたDocumentReferenceリソースに関する詳細が含まれています。ここでも、 json_extractを使用して、文字列として Athena に取り込まれるネストされた JSON 要素にアクセスする必要があります。

このサンプル SQL クエリは、特定の に基づいてCondition生成されたすべての を検索する方法を示していますDocumentReference。CAST を使用して JSON 要素を文字列として設定し、LIKEを使用して比較できるようにします。

```
SELECT
    meta.tag[1].display as SystemGenerated,
    json_extract(modifierextension[4], '$.valueReference.reference') as
 DocumentReference
FROM condition
WHERE meta.tag[1].display = 'SYSTEM_GENERATED'

AND CAST(json_extract(modifierextension[4], '$.valueReference.reference') as
 VARCHAR) LIKE '%DocumentReference/67aa0278-8111-40d0-8adc-43055eb9d18d%'
```

Resource type: Observation

リソースタイプ「オブザベーション」は、患者、デバイス、またはその他のサブジェクトについて行われた測定値と簡単なアサーションを保存します。HealthLake の統合自然言語処理 (NLP) は、Observationリソースで見つかった詳細に基づいて新しいDocumentReferenceリソースを生成します。このサンプル SQL クエリにはWHERE meta.tag[1] is NULLコメントアウトが含まれています。つまり、SYSTEM_GENERATED結果が含まれます。

```
SELECT valueCodeableConcept.coding[1].code
FROM Observation
WHERE  valueCodeableConcept.coding[1].code = '266919005'
```

```
-- WHERE meta.tag[1] is NULL
```

この列は としてインポートされましたstruct。したがって、ドット表記を使用して内部の要素にアクセスできます。

Resource type: MedicationStatement

MedicationStatement は、患者が使用した、使用している、または今後使用する薬剤の詳細を保存するために使用できる FHIR リソースタイプです。HealthLake の統合医療自然言語処理 (NLP) は、DocumentReference リソースタイプで見つかったドキュメントに基づいて、新しい MedicationStatement リソースを生成します。新しいリソースが生成されると、HealthLake は タグを SYSTEM_GENERATED meta要素に追加します。このサンプル SQL クエリは、識別子を使用して単一の患者に基づいてフィルタリングし、HealthLake の統合 NLP によって追加されたリソースを検索するクエリを作成する方法を示しています。

```
SELECT *
FROM medicationstatement
WHERE meta.tag[1].display = 'SYSTEM_GENERATED' AND subject.reference =
 'Patient/0679b7b7-937d-488a-b48d-6315b8e7003b';
```

HealthLake の統合自然言語処理 (NLP) の詳細については、「」を参照してくださいHealthLake の統合自然言語処理 (NLP)。

複雑なフィルタリングを使用した SQL クエリの例

次の例は、複雑なフィルタリングで Amazon Athena SQL クエリを使用して HealthLake データストアから FHIR データを検索する方法を示しています。

Example人口統計データに基づいてフィルタリング基準を作成する

患者コホートを作成するときは、正しい患者属性を特定することが重要です。このサンプルクエリは、Trino ドット表記と json_extract を使用して HealthLake データストア内のデータをフィルタリングする方法を示しています。

```
SELECT
    id
    , CONCAT(name[1].family, ' ', name[1].given[1]) as name
    , (year(current_date) - year(date(birthdate))) as age
    , gender as gender
```

```
    , json_extract(extension[1], '$.valueString') as MothersMaidenName
    , json_extract(extension[2], '$.valueAddress.city') as birthPlace
    , maritalstatus.coding[1].display as maritalstatus
    , address[1].line[1] as addressline
    , address[1].city as city
    , address[1].district as district
    , address[1].state as state
    , address[1].postalcode as postalcode
    , address[1].country as country
    , json_extract(address[1].extension[1], '$.extension[0].valueDecimal') as latitude
    , json_extract(address[1].extension[1], '$.extension[1].valueDecimal') as longitude
    , telecom[1].value as telNumber
    , deceasedboolean as deceasedIndicator
    , deceaseddatetime
FROM database.patient;
```

Athena コンソールを使用すると、結果をさらにソートしてダウンロードできます。

Example患者とその関連条件のフィルターを作成する

次のクエリ例は、HealthLake データストアで見つかった患者のすべての関連条件を検索してソートする方法を示しています。

```
SELECT
 patient.id as patientId
    , condition.id  as conditionId
    , CONCAT(name[1].family, ' ', name[1].given[1]) as name
    , condition.meta.tag[1].display
    , json_extract(condition.modifierextension[1], '$.valueDecimal') AS confidenceScore
    , category[1].coding[1].code as categoryCode
    , category[1].coding[1].display as categoryDescription
    , code.coding[1].code as diagnosisCode
    , code.coding[1].display as diagnosisDescription
    , onsetdatetime
    , severity.coding[1].code as severityCode
    , severity.coding[1].display as severityDescription
    , verificationstatus.coding[1].display as verificationStatus
    , clinicalstatus.coding[1].display as clinicalStatus
    , encounter.reference as encounterId
    , encounter.type as encountertype
FROM database.patient, condition
WHERE CONCAT('Patient/', patient.id) = condition.subject.reference
ORDER BY name;
```

Athena コンソールを使用して、結果をさらにソートしたり、さらに分析するためにダウンロードしたりできます。

Example患者とそれに関連する観測値のフィルターを作成する

次のクエリ例は、HealthLake データストアで見つかった患者の関連するすべての観測値を検索してソートする方法を示しています。

```
SELECT
 patient.id as patientId
    , observation.id as observationId
    , CONCAT(name[1].family, ' ', name[1].given[1]) as name
    , meta.tag[1].display
    , json_extract(modifierextension[1], '$.valueDecimal') AS confidenceScore
    , status
    , category[1].coding[1].code as categoryCode
    , category[1].coding[1].display as categoryDescription
    , code.coding[1].code as observationCode
    , code.coding[1].display as observationDescription
    , effectivedatetime
    , CASE
  WHEN valuequantity.value IS NOT NULL THEN CONCAT(CAST(valuequantity.value AS
 VARCHAR),' ',valuequantity.unit)
       WHEN valueCodeableConcept.coding [ 1 ].code IS NOT NULL THEN
 CAST(valueCodeableConcept.coding [ 1 ].code AS VARCHAR)
       WHEN valuestring IS NOT NULL THEN CAST(valuestring AS VARCHAR)
       WHEN valueboolean IS NOT NULL THEN CAST(valueboolean AS VARCHAR)
       WHEN valueinteger IS NOT NULL THEN CAST(valueinteger AS VARCHAR)
       WHEN valueratio IS NOT NULL THEN CONCAT(CAST(valueratio.numerator.value AS
 VARCHAR),'/',CAST(valueratio.denominator.value AS VARCHAR))
       WHEN valuerange IS NOT NULL THEN CONCAT(CAST(valuerange.low.value AS
 VARCHAR),'-',CAST(valuerange.high.value AS VARCHAR))
       WHEN valueSampledData IS NOT NULL THEN CAST(valueSampledData.data AS VARCHAR)
       WHEN valueTime IS NOT NULL THEN CAST(valueTime AS VARCHAR)
       WHEN valueDateTime IS NOT NULL THEN CAST(valueDateTime AS VARCHAR)
       WHEN valuePeriod IS NOT NULL THEN valuePeriod.start
       WHEN component[1] IS NOT NULL THEN CONCAT(CAST(component[2].valuequantity.value
 AS VARCHAR),' ',CAST(component[2].valuequantity.unit AS VARCHAR),
 '/', CAST(component[1].valuequantity.value AS VARCHAR),'
 ',CAST(component[1].valuequantity.unit AS VARCHAR))
    END AS observationvalue
 , encounter.reference as encounterId
    , encounter.type as encountertype
```

```
FROM database.patient, observation
WHERE CONCAT('Patient/', patient.id) = observation.subject.reference
ORDER BY name;
```

Example患者とその関連手順のフィルタリング条件を作成する

手順を患者に接続することは、医療の重要な側面です。次の SQL サンプルクエリは、FHIR PatientとProcedureリソースタイプを使用してこれを実現する方法を示しています。次の SQL クエリは、HealthLake データストアにあるすべての患者とその関連手順を返します。

```
SELECT
 patient.id  as patientId
 , PROCEDURE.id as procedureId
 , CONCAT(name[1].family, ' ', name[1].given[1]) as name
 , status
 , category.coding[1].code as categoryCode
 , category.coding[1].display as categoryDescription
 , code.coding[1].code as procedureCode
 , code.coding[1].display as procedureDescription
 , performeddatetime
 , performer[1]
 , encounter.reference as encounterId
 , encounter.type as encountertype
FROM database.patient, procedure
WHERE CONCAT('Patient/', patient.id) = procedure.subject.reference
ORDER BY name;
```

Athena コンソールを使用して、結果をダウンロードしてさらに分析したり、結果をよりよく理解するためにソートしたりできます。

Example患者および関連する処方のフィルタリング条件を作成する

患者が現在使用している薬剤のリストを確認することは重要です。Athena を使用すると、HealthLake データストアにある Patientと の両方のMedicationRequestリソースタイプを使用する SQL クエリを記述できます。

次の SQL クエリは、Athena にインポートされた テーブルPatientと MedicationRequestテーブルを結合します。また、ドット表記を使用して、処方を個々のエントリに整理します。

```
SELECT
 patient.id  as patientId
 , medicationrequest.id  as medicationrequestid
```

```
 , CONCAT(name[1].family, ' ', name[1].given[1]) as name
 , status
 , statusreason.coding[1].code as categoryCode
 , statusreason.coding[1].display as categoryDescription
 , category[1].coding[1].code as categoryCode
 , category[1].coding[1].display as categoryDescription
 , priority
 , donotperform
 , encounter.reference as encounterId
 , encounter.type as encountertype
 , medicationcodeableconcept.coding[1].code as medicationCode
 , medicationcodeableconcept.coding[1].display as medicationDescription
 , dosageinstruction[1].text as dosage
FROM database.patient, medicationrequest
WHERE CONCAT('Patient/', patient.id ) = medicationrequest.subject.reference
ORDER BY name
```

Athena コンソールを使用して結果をソートしたり、ダウンロードしてさらに分析したりできます。

Example **MedicationStatement** リソースタイプで見つかった薬剤を参照する

次のクエリ例は、SQL を使用して Athena にインポートされたネストされた JSON を整理する方法を示しています。クエリは FHIR meta要素を使用して、HealthLake の統合自然言語処理 (NLP) によって薬剤がいつ追加されたかを示します。また、 を使用して json_extract JSON 文字列の配列内のデータを検索します。詳細については、「自然言語処理」を参照してください。

```
SELECT
 medicationcodeableconcept.coding[1].code as medicationCode
 , medicationcodeableconcept.coding[1].display as medicationDescription
 , meta.tag[1].display
 , json_extract(modifierextension[1], '$.valueDecimal') AS confidenceScore
FROM medicationstatement;
```

Athena コンソールを使用して、これらの結果をダウンロードまたはソートできます。

Example特定の疾患タイプのフィルタリング

この例では、18～75 歳で、糖尿病と診断されている患者のグループを見つける方法を示しています。

```
SELECT patient.id as patientId,
 condition.id as conditionId,
 CONCAT(name [ 1 ].family, ' ', name [ 1 ].given [ 1 ]) as name,
```

```
(year(current_date) - year(date(birthdate))) AS age,
CASE
 WHEN condition.encounter.reference IS NOT NULL THEN condition.encounter.reference
 WHEN observation.encounter.reference IS NOT NULL THEN observation.encounter.reference
END as encounterId,
CASE
 WHEN condition.encounter.type IS NOT NULL THEN observation.encounter.type
 WHEN observation.encounter.type IS NOT NULL THEN observation.encounter.type
END AS encountertype,
condition.code.coding [ 1 ].code as diagnosisCode,
condition.code.coding [ 1 ].display as diagnosisDescription,
observation.category [ 1 ].coding [ 1 ].code as categoryCode,
observation.category [ 1 ].coding [ 1 ].display as categoryDescription,
observation.code.coding [ 1 ].code as observationCode,
observation.code.coding [ 1 ].display as observationDescription,
effectivedatetime AS observationDateTime,
CASE
     WHEN valuequantity.value IS NOT NULL THEN CONCAT(CAST(valuequantity.value AS
VARCHAR),' ',valuequantity.unit)
     WHEN valueCodeableConcept.coding [ 1 ].code IS NOT NULL THEN
CAST(valueCodeableConcept.coding [ 1 ].code AS VARCHAR)
     WHEN valuestring IS NOT NULL THEN CAST(valuestring AS VARCHAR)
     WHEN valueboolean IS NOT NULL THEN CAST(valueboolean AS VARCHAR)
     WHEN valueinteger IS NOT NULL THEN CAST(valueinteger AS VARCHAR)
     WHEN valueratio IS NOT NULL THEN CONCAT(CAST(valueratio.numerator.value AS
VARCHAR),'/',CAST(valueratio.denominator.value AS VARCHAR))
     WHEN valuerange IS NOT NULL THEN CONCAT(CAST(valuerange.low.value AS
VARCHAR),'-',CAST(valuerange.high.value AS VARCHAR))
     WHEN valueSampledData IS NOT NULL THEN CAST(valueSampledData.data AS VARCHAR)
     WHEN valueTime IS NOT NULL THEN CAST(valueTime AS VARCHAR)
     WHEN valueDateTime IS NOT NULL THEN CAST(valueDateTime AS VARCHAR)
     WHEN valuePeriod IS NOT NULL THEN valuePeriod.start
     WHEN component[1] IS NOT NULL THEN CONCAT(CAST(component[2].valuequantity.value
AS VARCHAR),' ',CAST(component[2].valuequantity.unit AS VARCHAR),
'/', CAST(component[1].valuequantity.value AS VARCHAR),'
',CAST(component[1].valuequantity.unit AS VARCHAR))
   END AS observationvalue,
CASE
 WHEN condition.meta.tag [ 1 ].display = 'SYSTEM GENERATED' THEN 'YES'
 WHEN condition.meta.tag [ 1 ].display IS NULL THEN 'NO'
 WHEN observation.meta.tag [ 1 ].display = 'SYSTEM GENERATED' THEN 'YES'
 WHEN observation.meta.tag [ 1 ].display IS NULL THEN 'NO'
  END AS IsSystemGenerated,
 CAST(
```

```
    json_extract(
      condition.modifierextension [ 1 ],
      '$.valueDecimal'
    ) AS int
  ) AS confidenceScore
FROM database.patient,
 database.condition,
 database.observation
WHERE CONCAT('Patient/', patient.id) = condition.subject.reference
 AND CONCAT('Patient/', patient.id) = observation.subject.reference
   AND (year(current_date) - year(date(birthdate))) >= 18
   AND (year(current_date) - year(date(birthdate))) <= 75
   AND condition.code.coding [ 1 ].display like ('%diabetes%');
```

これで、Athena コンソールを使用して結果をソートしたり、ダウンロードしてさらに分析したりできます。

モニタリング AWS HealthLake

モニタリングとログ記録は、 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な部分です AWS HealthLake。 は、HealthLake を監視し、問題が発生したときに報告し、必要に応じて自動アクションを実行するために、以下のサービス AWS を提供します。

- AWS CloudTrail は、 AWS アカウントによって、またはアカウントに代わって行われた API コールおよび関連イベントをキャプチャし、指定した Amazon S3 バケットにログファイルを配信します。呼び出し元のユーザーとアカウント AWS、呼び出し元の送信元 IP アドレス、呼び出しの発生日時を特定できます。詳細については、「AWS CloudTrail ユーザーガイド」を参照してください。
- Amazon CloudWatch は、 AWS リソースと で実行しているアプリケーションを AWS リアルタイムでモニタリングします。メトリクスの収集と追跡、カスタマイズしたダッシュボードの作成、および指定したメトリクスが指定したしきい値に達したときに通知またはアクションを実行するアラームの設定を行うことができます。例えば、CloudWatch で Amazon EC2 インスタンスの CPU 使用率などのメトリクスを追跡し、必要に応じて新しいインスタンスを自動的に起動できます。詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。
- Amazon EventBridge は、アプリケーションをさまざまなイベントソースのデータに簡単に接続できるようにするサーバーレスイベントバスサービスです。EventBridge は、お客様独自のアプリケーション、Software as a Service (SaaS) アプリケーション、 AWS のサービスからのリアルタイムデータをストリーム配信し、そのデータを Lambda などのターゲットにルーティングします。これにより、サービスで発生したイベントをモニタリングし、イベント駆動型アーキテクチャを構築できます。詳細については、「Amazon EventBridge ユーザーガイド」を参照してください。

トピック

- を使用した HealthLake API コールのログ記録 AWS CloudTrail
- Amazon CloudWatch を使用した HealthLake メトリクスのモニタリング
- Amazon EventBridge を使用した HealthLake イベントのモニタリング EventBridge

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

AWS HealthLake は AWS CloudTrail、HealthLake のユーザー、ロール、または のサービスによって実行されたアクションを記録する AWS サービスである と統合されています。CloudTrail

は、HealthLake のすべての API コールをイベントとしてキャプチャします。キャプチャされた呼び出しには、HealthLake コンソールからの呼び出しと HealthLake API オペレーションへのコード呼び出しが含まれます。証跡を作成する場合は、HealthLake のイベントなど、Amazon S3 バケットへの CloudTrail イベントの継続的な配信を有効にすることができます。証跡を設定しない場合でも、CloudTrail コンソールの [イベント履歴] で最新のイベントを表示できます。CloudTrail で収集された情報を使用して、HealthLake に対するリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

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。

AWS HealthLake CloudTrail の情報

CloudTrail は、 AWS アカウントの作成時にアカウントで有効になります。HealthLake 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。

HealthLake のイベントなど、 AWS アカウントのイベントの継続的な記録については、証跡を作成します。追跡により、CloudTrail はログファイルを Simple Storage Service (Amazon S3) バケットに配信できます。デフォルトでは、コンソールで作成した証跡がすべての AWS リージョンに適用されます。証跡は、 AWS パーティション内のすべてのリージョンからのイベントを口グに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、CloudTrail ログで収集されたイベントデータをさらに分析して処理するように他の AWS サービスを設定できます。詳細については、次を参照してください:

- 証跡の作成のための概要
- CloudTrail がサポートするサービスと統合
- CloudTrail 用 Amazon SNS 通知の構成
- 複数のリージョンから CloudTrail ログファイルを受け取るおよび複数のアカウントから CloudTrail ログファイルを受け取る

すべての HealthLake アクションは CloudTrail によって口グに記録され、HealthLake API リファレンス」と「このデベロッパーガイド」に、FHIR REST API を使用して実行されるアクションが記載されています。たとえば、次の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。

- `DescribeFHIRImportJob`

- DescribeFHIRExportJob
- StartFHIRImportJob
- ListFHIRImportJobs
- StartFHIRExportJob
- ListFHIRExportJobs
- CreateFHIRDatastore
- ListFHIRDatastores
- DeleteFHIRDatastore
- DescribeFHIRDatastore
- UpdateResource
- CreateResource
- DeleteResource
- ReadResource
- GetCapabilities
- SearchWithGet
- SearchWithPost

各イベントまたはログエントリには、誰がリクエストを生成したかという情報が含まれます。アイデンティティ情報は、以下を判別するのに役立ちます。

- リクエストがルートまたは AWS Identity and Access Management (IAM) ユーザー認証情報を使用して行われたかどうか。
- リクエストがロールまたはフェデレーション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 サービスによって行われたかどうか。

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

AWS HealthLake ログファイルエントリについて

「トレイル」は、指定した Amazon S3 バケットにイベントをログファイルとして配信するように設定できます。CloudTrail のログファイルは、単一か複数のログエントリを含みます。イベントは、任意の出典からの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエ

ストパラメータなどに関する情報が含まれます。CloudTrail ログファイルは、パブリック API 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。

以下の例は、CreateFHIRDatastore アクションを示す CloudTrail ログエントリです。


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA2B3ZHOADD2OJ4AHJX:git
 full_access_iam_role580074395690222150",
        "arn": "arn:aws:sts::691207106566:assumed-role/
colossusfrontend_full_access_iam_role/_iam_role580074395690222150",
        "accountId": "AccountID",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROA2B3ZHOADD2OJ4AHJX",
                "arn": "arn:aws:iam::691207106566:role/full_access_iam_role",
                "accountId": "AccountID",
                "userName": "full_access_iam_role"
            },
            "webIdFederationData": {

            },
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-11-20T00:08:15Z"
            }
        }
    },
    "eventTime": "2020-11-20T00:08:16Z",
    "eventSource": "healthlake.amazonaws.com",
    "eventName": "CreateFHIRDatastore",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "3.213.247.1",
    "userAgent": "Coral/Netty4",
    "requestParameters": {
        "datastoreName":
 "testCreateFHIRDatastore_GBYAZFCLLBLSUTOYYFQZRLBLQJNFOYQVRPZBOJAIIUAHICAEAGIWLNVQEYAMSXVWMBLXC
        "datastoreTypeVersion": "R4",
        "clientToken": "d737ffe0-14dd-44cc-9f0a-fdf59b26c66b"
    },
```

```
    "responseElements": {
        "datastoreId": "datastoreID",
        "datastoreArn": "arn:aws:healthlake:us-
east-1:691207106566:datastore/55576c487ff4975262b10d1d65eb4509",
        "datastoreStatus": "CREATING",
        "datastoreEndpoint": "datastore_endpoint/"
    },
    "requestID": "68e62bdd-d2d4-44c1-af69-e6f055a69f99",
    "eventID": "7ef483dc-5dca-469e-823a-7d9e3a7fe924",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "691207106566"
}
```


Amazon CloudWatch を使用した HealthLake メトリクスのモニタリング

Amazon CloudWatch を使用して HealthLake Amazon CloudWatch は raw データを収集し、読み取り可能なほぼリアルタイムのメトリクスに加工します。これらの統計は 15 か月間保持されるため、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、特定のしきい値を監視するアラームを設定し、これらのしきい値に達したときに通知を送信したりアクションを実行したりできます。詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。

> **Note**
>
> メトリクスは、すべてのネイティブ HealthLake アクションについてレポートされます。

次の表に、CloudWatch に報告された HealthLake メトリクスとディメンションを示します。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。

次の HealthLake メトリクスが CloudWatch に報告されます。

CloudWatch にレポートされる HealthLake 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されたデータストアについて報告できます。 単位: 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
成功したリクエスト	成功した API リクエストの数。 単位: カウント 有効な統計: Sum、Average 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
ユーザーエラー	ユーザーエラーが原因で失敗したリクエストの数。 単位: カウント 有効な統計: Sum、Average 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
サーバーエラー	サーバーエラーのために失敗したリクエストの数。 単位: カウント 有効な統計: Sum、Average

メトリクス	説明
	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
スロットルリクエスト	スロットリングされたリクエストの数。このメトリクスは、ユーザーまたはサーバーのエラー数に含まれません。 単位: カウント 有効な統計: Sum、Average 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
レイテンシー	ユーザーリクエストの処理にかかったミリ秒単位の時間。 単位: ミリ秒 有効な統計: Minimum、Maximum、Average 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

次の HealthLake ディメンションが CloudWatch に報告されます。

CloudWatch にレポートされる HealthLake ディメンション

ディメンション	説明
運用	リクエストで使用される API オペレーション
DataStoreID	リクエストで使用されるデータストア ID
DataStoreType	リクエストで使用されるデータストアのタイプ

HealthLake のメトリクスは、AWS マネジメントコンソール、 AWS CLI、または CloudWatch API を使用して取得できます。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。HealthLake 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 HealthLake をモニタリングするには、適切な CloudWatch アクセス許可が必要です。詳細については、Amazon CloudWatch ユーザーガイド」の「Amazon CloudWatch の Identity and Access Management」を参照してください。 Amazon CloudWatch

HealthLake 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. にサインイン AWS マネジメントコンソール し、CloudWatch コンソールを開きます。
2. メトリクスを選択し、すべてのメトリクスを選択し、AWS/HealthLake を選択します。
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、指定された期間にわたって単一のメトリクスを監視し、Amazon Simple Notification Service (SNS) トピックまたは Auto Scaling ポリシーに通知を送信する 1 つ以上のアクションを実行します。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。CloudWatch は、アラームの状態が変わったときに SNS メッセージを送信することもできます。

> **Note**
>
> 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。

メトリクスを表示する方法 (CloudWatch コンソール)

1. CloudWatchコンソールにサインインします。
2. [Alarms]、[Create Alarm] の順に選択します。
3. AWS/HealthLake を選択し、メトリクスを選択します。
4. [Time Range] (時間の範囲) で、モニタリングする期間を選択し、[Next] (次へ) を選択します。
5. [Name] (名前) と [Description] (説明) を入力します。

6. 常に >= を選択し、最大値を入力します。
7. アラーム状態に達したときに CloudWatch から E メールを送信する場合は、「アクション」セクションの「このアラームが発生するたびに、「状態は ALARM」を選択します。通知の送信先として、メーリングリストを選択するか、新しいリストを選択して新しいメーリングリストを作成します。
8. [Alarm Preview] (アラームの確認) セクションでアラームをプレビューします。アラームに問題がなければ、[Create Alarm] (アラームの作成) を選択します。

Amazon EventBridge を使用した HealthLake イベントのモニタリング EventBridge

Amazon EventBridge は、イベントを使用してアプリケーションコンポーネント同士を接続するサーバーレスサービスです。これにより、スケーラブルなイベント駆動型アプリケーションを簡単に構築できます。EventBridge の基礎は、イベントをターゲットにルーティングするルールを作成することです。 AWS HealthLake はEventBridge に状態変更を永続的に配信します。詳細については、「Amazon EventBridge ユーザーガイド」の「Amazon EventBridge とは」を参照してください。

> **Note**
>
> HealthLake イベントを Amazon EventBridge に送信する方法については、 for AWS Things ブログの「 の Amazon EventBridge 統合 AWS HealthLake」を参照してください。

トピック

- EventBridge に送信される HealthLake イベント EventBridge
- HealthLake イベント構造

EventBridge に送信される HealthLake イベント EventBridge

次の表に、処理のために EventBridge に送信されたすべての HealthLake イベントを示します。

HealthLake イベントタイプ	State
データストアイベント	

HealthLake イベントタイプ	State
データストアの作成	CREATING
データストアがアクティブ	ACTIVE
データストアの削除	DELETING
データストアの削除	DELETED
データストアの作成に失敗しました	CREATE_FAILED
詳細については、「AWS HealthLake API リファレンス」の「datastoreStatus 」を参照してください。	
ジョブイベントをインポートする	
インポートジョブが送信されました	SUBMITTED
インポート中のジョブ	IN_PROGRESS
エラーで完了したジョブのインポート	COMPLETED_WITH_ERRORS
インポートジョブが完了しました	COMPLETED
ジョブのインポートに失敗しました	FAILED
詳細については、「AWS HealthLake API リファレンス」の「jobStatus 」を参照してください。	
ジョブイベントのエクスポート	
エクスポートジョブが送信されました	SUBMITTED
ジョブのエクスポート中	IN_PROGRESS
エラーで完了したジョブのエクスポート	COMPLETED_WITH_ERRORS
エクスポートジョブが完了しました	COMPLETED
エクスポートジョブが失敗しました	FAILED

HealthLake イベントタイプ	State
詳細については、「AWS HealthLake API リファレンス」の「jobStatus 」を参照してください。	

HealthLake イベント構造

HealthLake イベントは、メタデータの詳細を含む JSON 構造を持つオブジェクトです。メタデータを入力として使用して、イベントを再作成するか、詳細情報を確認できます。関連するすべてのメタデータフィールドは、次のメニューのコード例の下の表に一覧表示されます。詳細については、「Amazon EventBridge ユーザーガイド」のAWS 「サービスイベントメタデータ」を参照してください。

> **Note**
>
> HealthLake イベントを Amazon EventBridge に送信する方法については、 for AWS Industrys ブログの「 の Amazon EventBridge 統合 AWS HealthLake」を参照してください。

データストアイベント

Data Store Creating

状態 - **CREATING**

```
{
    "version": "0",
    "id": "514ad836-bb8a-4523-a10b-fa2756c3bdb0",
    "detail-type": "Data Store Creating",
    "source": "aws.healthlake",
    "account": "123456789012",
    "time": "2023-12-08T08:58:12Z",
    "region": "us-east-1",
    "resources":
    [
        "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
eeb8005725ae22b35b4edbdc68cf2dfd"
    ],
    "detail":
    {
```

```
        "datastoreId": "eeb8005725ae22b35b4edbdc68cf2dfd",
        "datastoreName": "your-data-store-name",
        "datastoreTypeVersion": "R4",
        "datastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/datastore/
eeb8005725ae22b35b4edbdc68cf2dfd/r4/"
    }
}
```

Data Store Active

状態 - **ACTIVE**

```
{
    "version": "0",
    "id": "d57105bc-0d2d-4009-b34d-453e2567c599",
    "detail-type": "Data Store Active",
    "source": "aws.healthlake",
    "account": "123456789012",
    "time": "2023-12-08T09:16:51Z",
    "region": "us-east-1",
    "resources":
    [
        "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
eeb8005725ae22b35b4edbdc68cf2dfd"
    ],
    "detail":
    {
        "datastoreId": "eeb8005725ae22b35b4edbdc68cf2dfd",
        "datastoreName": "your-data-store-name",
        "datastoreTypeVersion": "R4",
        "datastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/datastore/
eeb8005725ae22b35b4edbdc68cf2dfd/r4/"
    }
}
```

Data Store Deleting

状態 - **DELETING**

```
{
    "version": "0",
    "id": "d135ee1f-e14a-4730-8766-7b98f822c94a",
    "detail-type": "Data Store Deleting",
```

```
    "source": "aws.healthlake",
    "account": "123456789012",
    "time": "2023-12-08T12:44:47Z",
    "region": "us-east-1",
    "resources":
    [
        "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
eeb8005725ae22b35b4edbdc68cf2dfd"
    ],
    "detail":
    {
        "datastoreId": "eeb8005725ae22b35b4edbdc68cf2dfd",
        "datastoreName": "your-data-store-name",
        "datastoreTypeVersion": "R4",
        "datastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/datastore/
eeb8005725ae22b35b4edbdc68cf2dfd/r4/"
    }
}
```

Data Store Deleted

状態 - **DELETED**

```
{
    "version": "0",
    "id": "6d880b86-e115-4947-81a9-494db704571a",
    "detail-type": "Data Store Deleted",
    "source": "aws.healthlake",
    "account": "123456789012",
    "time": "2023-05-12T12:58:03Z",
    "region": "us-east-1",
    "resources":
    [
        "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
eeb8005725ae22b35b4edbdc68cf2dfd"
    ],
    "detail":
    {
        "datastoreId": "eeb8005725ae22b35b4edbdc68cf2dfd",
        "datastoreName": "your-data-store-name",
        "datastoreTypeVersion": "R4",
        "datastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/datastore/
eeb8005725ae22b35b4edbdc68cf2dfd/r4/"
    }
```

```
}
```

データストアイベント - メタデータの説明

名前	型	説明
version	string	EventBridge イベントスキーマのバージョン。
id	string	イベントごとに生成されたバージョン 4 UUID。
detail-type	string	送信されるイベントのタイプ。
source	string	イベントを発生させたサービスを識別します。
account	string	データストア所有者の 12 桁の AWS アカウント ID。
time	string	イベントが発生した時刻。
region	string	データストアの AWS リージョンを識別します。
resources	配列 (文字列)	データストアの ARN を含む JSON 配列。
detail	オブジェクト	イベントに関する情報を含む JSON オブジェクト。
detail.datastoreId	string	ステータス変更イベントに関連付けられたデータストア ID。
detail.datastoreName	string	データストア名。
detail.datastoreTypeVersion	string	データストアの FHIR バージョン。

名前	型	説明
detail.datastoreEndpoint	string	データストアエンドポイント。

ジョブイベントをインポートする

Import Job Submitted

状態 - **SUBMITTED**

```
{
    "version": "0",
    "id": "25e606f7-800c-da41-45df-0e68587250c9",
    "detail-type": "Import Job Submitted",
    "source": "aws.healthlake",
    "account": "123456789012",
    "time": "2023-12-08T01:50:51Z",
    "region": "us-east-1",
    "resources":
    [
        "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
eeb8005725ae22b35b4edbdc68cf2dfd"
    ],
    "detail":
    {
        "jobId": "08c60716d6321710893ff88410e902c2",
        "submitTime": "2023-12-08T01:50:50.986Z",
        "datastoreId": "eeb8005725ae22b35b4edbdc68cf2dfd",
        "inputDataConfig":
        {
            "s3Uri": "s3://amzn-s3-demo-source-bucket/input/"
        }
    }
}
```

Import Job In Progress

状態 - **IN_PROGRESS**

```
{
    "version": "0",
```

```
    "id": "cc886b49-2737-19c4-7c4e-84ac9429ab73",
    "detail-type": "Import Job In Progress",
    "source": "aws.healthlake",
    "account": "123456789012",
    "time": "2023-12-08T01:51:23Z",
    "region": "us-east-1",
    "resources":
    [
        "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
eeb8005725ae22b35b4edbdc68cf2dfd"
    ],
    "detail":
    {
        "jobId": "08c60716d6321710893ff88410e902c2",
        "submitTime": "2023-12-08T01:50:50.986Z",
        "datastoreId": "eeb8005725ae22b35b4edbdc68cf2dfd",
        "inputDataConfig":
        {
            "s3Uri": "s3://amzn-s3-demo-source-bucket/input/"
        }
    }
}
```

Import Job Completed

状態 - **COMPLETED**

```
{
    "version": "0",
    "id": "36c865ef-da41-76ef-c882-3ba8dad8656b",
    "detail-type": "Import Job Completed",
    "source": "aws.healthlake",
    "account": "123456789012",
    "time": "2023-12-08T02:14:42Z",
    "region": "us-east-1",
    "resources":
    [
        "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
eeb8005725ae22b35b4edbdc68cf2dfd"
    ],
    "detail":
    {
        "jobId": "08c60716d6321710893ff88410e902c2",
        "submitTime": "2023-12-08T01:50:50.986Z",
```

```
        "datastoreId": "eeb8005725ae22b35b4edbdc68cf2dfd",
        "inputDataConfig":
        {
            "s3Uri": "s3://amzn-s3-demo-source-bucket/input/"
        }
    }
}
```

Import Job Completed With Errors

状態 - **COMPLETED_WITH_ERRORS**

```
{
    "version": "0",
    "id": "b61387d7-bffe-4f01-8291-65dc4be52cc1",
    "detail-type": "Import Job Completed With Errors",
    "source": "aws.healthlake",
    "account": "123456789012",
    "time": "2023-12-08T02:14:42Z",
    "region": "us-east-1",
    "resources":
    [
        "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
eeb8005725ae22b35b4edbdc68cf2dfd"
    ],
    "detail":
    {
        "jobId": "08c60716d6321710893ff88410e902c2",
        "submitTime": "2023-12-08T01:50:50.986Z",
        "datastoreId": "eeb8005725ae22b35b4edbdc68cf2dfd",
        "inputDataConfig":
        {
            "s3Uri": "s3://amzn-s3-demo-source-bucket/input/"
        }
    }
}
```

Import Job Failed

状態 - **FAILED**

```
{
    "version": "0",
```

```
    "id": "c4d65575-d1a7-4040-9c6c-c225bf6723c5",
    "detail-type": "Import Job Failed",
    "source": "aws.healthlake",
    "account": "123456789012",
    "time": "2023-12-08T02:14:42Z",
    "region": "us-east-1",
    "resources":
    [
        "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
eeb8005725ae22b35b4edbdc68cf2dfd"
    ],
    "detail":
    {
        "jobId": "08c60716d6321710893ff88410e902c2",
        "submitTime": "2023-12-08T01:50:50.986Z",
        "datastoreId": "eeb8005725ae22b35b4edbdc68cf2dfd",
        "inputDataConfig":
        {
            "s3Uri": "s3://amzn-s3-demo-source-bucket/input/"
        }
    }
}
```

ジョブイベントのインポート - メタデータの説明

名前	型	説明
version	string	EventBridge イベントスキーマのバージョン。
id	string	イベントごとに生成されたバージョン 4 UUID。
detail-type	string	送信されるイベントのタイプ。
source	string	イベントを発生させたサービスを識別します。
account	string	データストア所有者の 12 桁の AWS アカウント ID。

名前	型	説明
time	string	イベントが発生した時刻。
region	string	データストアの AWS リージョンを識別します。
resources	配列 (文字列)	データストアの ARN を含む JSON 配列。
detail	オブジェクト	イベントに関する情報を含む JSON オブジェクト。
detail.jobId	string	ステータス変更イベントに関連付けられたインポートジョブ ID。
detail.submitTime	string	インポートジョブが送信された時刻。
detail.datastoreId	string	ステータス変更イベントを生成したデータストア。
detail.inputDataConfig	string	インポートする FHIR ファイルを含む Amazon S3 バケットの入力プレフィックスパス。

ジョブイベントのエクスポート

Export Job Submitted

状態 - **SUBMITTED**

```
{
    "version": "0",
    "id": "f8af7d04-2221-4f02-a01a-6fc3ae403bab",
    "detail-type": "Export Job Submitted",
    "source": "aws.healthlake",
    "account": "123456789012",
```

```
    "time": "2023-12-08T01:50:51Z",
    "region": "us-east-1",
    "resources":
    [
        "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
eeb8005725ae22b35b4edbdc68cf2dfd"
    ],
    "detail":
    {
        "jobId": "45e899e545bf774710388260fc60b143",
        "submitTime": "2023-12-08T01:50:50.986Z",
        "datastoreId": "eeb8005725ae22b35b4edbdc68cf2dfd",
        "outputDataConfig":
        {
            "s3Uri": "s3://amzn-s3-demo-source-bucket/output/"
        }
    }
}
```

Export Job In Progress

状態 - **IN_PROGRESS**

```
{
    "version": "0",
    "id": "7bb7e39c-707d-4a83-8532-cee015299100",
    "detail-type": "Export Job In Progress",
    "source": "aws.healthlake",
    "account": "123456789012",
    "time": "2023-12-08T01:51:23Z",
    "region": "us-east-1",
    "resources":
    [
        "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
eeb8005725ae22b35b4edbdc68cf2dfd"
    ],
    "detail":
    {
        "jobId": "45e899e545bf774710388260fc60b143",
        "submitTime": "2023-12-08T01:50:50.986Z",
        "datastoreId": "eeb8005725ae22b35b4edbdc68cf2dfd",
        "outputDataConfig":
        {
            "s3Uri": "s3://amzn-s3-demo-source-bucket/output/"
```

```
        }
    }
}
```

Export Job Completed

状態 - **COMPLETED**

```
{
    "version": "0",
    "id": "d7629aa7-e63a-4b84-858c-96a62b57ebc8",
    "detail-type": "Export Job Completed",
    "source": "aws.healthlake",
    "account": "123456789012",
    "time": "2023-12-08T02:14:42Z",
    "region": "us-east-1",
    "resources":
    [
        "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
eeb8005725ae22b35b4edbdc68cf2dfd"
    ],
    "detail":
    {
        "jobId": "45e899e545bf774710388260fc60b143",
        "submitTime": "2023-12-08T01:50:50.986Z",
        "datastoreId": "eeb8005725ae22b35b4edbdc68cf2dfd",
        "outputDataConfig":
        {
            "s3Uri": "s3://amzn-s3-demo-source-bucket/output/"
        }
    }
}
```

Export Job Completed With Errors

状態 - **COMPLETED_WITH_ERRORS**

```
{
    "version": "0",
    "id": "5fa50bc5-50e3-4bc4-b66a-1b1d2f7b07a7",
    "detail-type": "Export Job Completed With Errors",
    "source": "aws.healthlake",
    "account": "123456789012",
```

```
    "time": "2023-12-08T02:14:42Z",
    "region": "us-east-1",
    "resources":
    [
        "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
eeb8005725ae22b35b4edbdc68cf2dfd"
    ],
    "detail":
    {
        "jobId": "45e899e545bf774710388260fc60b143",
        "submitTime": "2023-12-08T01:50:50.986Z",
        "datastoreId": "eeb8005725ae22b35b4edbdc68cf2dfd",
        "outputDataConfig":
        {
            "s3Uri": "s3://amzn-s3-demo-source-bucket/output/"
        }
    }
}
```

Export Job Failed

状態 - **FAILED**

```
{
    "version": "0",
    "id": "49fce45e-7e02-4846-8582-e7f19ca039cb",
    "detail-type": "Export Job Failed",
    "source": "aws.healthlake",
    "account": "123456789012",
    "time": "2023-12-08T02:14:42Z",
    "region": "us-east-1",
    "resources":
    [
        "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
eeb8005725ae22b35b4edbdc68cf2dfd"
    ],
    "detail":
    {
        "jobId": "45e899e545bf774710388260fc60b143",
        "submitTime": "2023-12-08T01:50:50.986Z",
        "datastoreId": "eeb8005725ae22b35b4edbdc68cf2dfd",
        "outputDataConfig":
        {
            "s3Uri": "s3://amzn-s3-demo-source-bucket/output/"
```

```
        }
    }
}
```

ジョブイベントのエクスポート - メタデータの説明

名前	型	説明
version	string	EventBridge イベントスキーマのバージョン。
id	string	イベントごとに生成されたバージョン 4 UUID。
detail-type	string	送信されるイベントのタイプ。
source	string	イベントを発生させたサービスを識別します。
account	string	データストア所有者の 12 桁の AWS アカウント ID。
time	string	イベントが発生した時刻。
region	string	データストアの AWS リージョンを識別します。
resources	配列 (文字列)	データストアの ARN を含む JSON 配列。
detail	オブジェクト	イベントに関する情報を含む JSON オブジェクト。
detail.jobId	string	ステータス変更イベントに関連付けられたエクスポートジョブ ID。
detail.submitTime	string	エクスポートジョブが送信された時刻。

名前	型	説明
detail.datastoreId	string	ステータス変更イベントを生成したデータストア。
detail.outputDataConfig	string	エクスポートする FHIR ファイルを含む Amazon S3 バケットの出力プレフィックスパス。

AWS HealthLake のセキュリティ

のクラウドセキュリティが最優先事項 AWS です。お客様は AWS 、最もセキュリティの影響を受けやすい組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャを活用できます。

セキュリティは、 AWS とお客様の間の責任共有です。責任共有モデルではこれをクラウドのセキュリティおよびクラウド内のセキュリティと説明しています。

- クラウドのセキュリティ AWS は、 AWS クラウドで AWS サービスを実行するインフラストラクチャを保護する責任を担います。 は、安全に使用できるサービス AWS も提供します。サードパーティーの監査者は、AWS コンプライアンスプログラムコンプライアンスプログラムの一環として、当社のセキュリティの有効性を定期的にテストおよび検証。HealthLake に適用されるコンプライアンスプログラムの詳細については、「コンプライアンスプログラムによる AWS 対象範囲内のサービスコンプライアンスプログラム」を参照してください。
- クラウド内のセキュリティ – お客様の責任は、使用する AWS サービスによって決まります。また、ユーザーは、データの機密性、会社の要件、適用される法律や規制など、その他の要因についても責任を負います。

このドキュメントは、HealthLake を使用する際の責任共有モデルの適用方法を理解するのに役立ちます。以下のトピックでは、セキュリティおよびコンプライアンスの目的を達成するように HealthLake を設定する方法について説明します。また、HealthLake 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 HealthLake でのデータ保護
- REST for AWS HealthLake での暗号化
- AWS HealthLake の転送中の暗号化
- AWS HealthLake の Identity and Access Management
- AWS HealthLake のコンプライアンス検証
- AWS HealthLake のインフラストラクチャセキュリティ
- を使用した AWS HealthLake リソースの作成 AWS CloudFormation
- AWS HealthLake とインターフェイス VPC エンドポイント (AWS PrivateLink)
- AWS HealthLake のセキュリティのベストプラクティス

- AWS HealthLake の耐障害性

AWS HealthLake でのデータ保護

責任 AWS 共有モデル、 AWS HealthLake でのデータ保護に適用されます。このモデルで説明されているように、 AWS はすべての を実行するグローバルインフラストラクチャを保護する責任があります AWS クラウド。ユーザーは、このインフラストラクチャでホストされるコンテンツに対する管理を維持する責任があります。また、使用する「 AWS のサービス 」のセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、「Data Privacy FAQChina」を参照してください。 欧州におけるデータ保護に関する情報については、General Data Protection Regulation (GDPR) Center 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 アイデンティティセンターまたは AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

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

お客様の E メールアドレスなどの極秘または機密情報を、タグ、または [名前] フィールドなどの自由形式のテキストフィールドに含めないことを強くお勧めします。これは、コンソール、API、または SDK を使用して HealthLake 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 タグ、または名前に使用される自由記述のテキストフィールドに入力したデータは、請

求または診断ログに使用される場合があります。外部サーバーに URL を提供する場合、そのサーバーへのリクエストを検証できるように、認証情報を URL に含めないことを強くお勧めします。

REST for AWS HealthLake での暗号化

HealthLake はデフォルトで暗号化を提供し、サービス所有の AWS Key Management Service (AWS KMS) キーを使用して保管中の機密データを保護します。カスタマーマネージド KMS キーもサポートされており、データストアからのファイルのインポートとエクスポートの両方に必要です。カスタマーマネージド KMS キーの詳細については、「Amazon Key Management Service」を参照してください。お客様は、データストアの作成時に AWS 所有の KMS キーまたはカスタマー管理の KMS キーを選択できます。データストアの作成後に暗号化設定を変更することはできません。データストアが AWS 所有の KMS キーを使用している場合、AWS_OWNED_KMS_KEY と表示され、保管時の暗号化に使用される特定のキーは表示されません。

AWS 所有の KMS キー

HealthLake 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ヘルス情報 (PHI) データなどの機密情報を自動的に暗号化します。AWS 所有の 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 アカウントで使用するために AWS が所有および管理する KMS キーのコレクションの一部です。AWS のサービスでは、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、それら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ず、アカウントの AWS KMS クォータにはカウントされません。詳細については、「AWS 所有のキー」を参照してください。

カスタマーマネージド KMS キー

HealthLake は、作成、所有、管理する対称カスタマーマネージド KMS キーを使用して、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

- タグを追加する
- 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、HealthLake が AWS KMS ユーザーに代わって 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、「カスタマー所有のキー」を参照してください。

カスタマーマネージドキーを作成する

AWS マネジメントコンソールまたは AWS KMS APIs を使用して、対称カスタマーマネージドキーを作成できます。

AWS Key Management Service デベロッパーガイドの「対称カスタマーマネージドキーの作成」の手順に従います。

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。キーポリシーは、カスタマーマネージドキーの作成時に指定できます。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。

HealthLake リソースでカスタマーマネージドキーを使用するには、キーポリシーで kms:CreateGrant 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、ユーザーは HealthLake が必要とする kms:grant オペレーションにアクセスできます。詳細については、「許可の使用」を参照してください。

HealthLake リソースでカスタマーマネージド KMS キーを使用するには、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:CreateGrant は、特定のカスタマーマネージド KMS キーに許可を追加し、許可オペレーションへのアクセスを許可します。
- 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、すべての書き込みオペレーションで保管中のリソースを暗号化するためのアクセスを提供します。

- kms:Decrypt は、暗号化されたリソースの読み取りまたは検索オペレーションへのアクセスを提供します。

以下は、ユーザーがそのキーによって暗号化された AWS HealthLake の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。

```
"Statement": [
    {
        "Sid": "Allow access to create data stores and do CRUD/search in AWS
 HealthLake",
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::111122223333:HealthLakeFullAccessRole"
        },
        "Action": [
            "kms:DescribeKey",
            "kms:CreateGrant",
            "kms:GenerateDataKey",
            "kms:Decrypt"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "kms:ViaService": "healthlake.amazonaws.com",
                "kms:CallerAccount": "111122223333"
            }
        }
    }
]
```

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、HealthLake 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。

kms:ViaService 条件キーを使用して、KMS キーの使用を HealthLake からのリクエストのみに制限できます。

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、および kms:DescribeKey アクセス許可と、必要な HealthLake アクセス許可が必要です。

HealthLake が AWS KMS で許可を使用する方法

HealthLake では、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、HealthLake は CreateGrant リクエストを AWS KMS に送信して、ユーザーに代わって許可を作成します。AWS KMS の許可は、顧客アカウントの KMS キーへのアクセスを HealthLake に許可するために使用されます。

HealthLake がユーザーに代わって作成する許可は、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 AWS KMS キーを使用するアクセス許可を HealthLake に付与する許可を取り消しまたは廃止した場合、HealthLake は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 FHIR リソースを暗号化したり、プル時に復号したりすることはできません。HealthLake の許可を取り消すか廃止すると、変更はすぐに行われます。アクセス権を取り消すには、許可を取り消すのではなく、データストアを削除する必要があります。データストアが削除されると、HealthLake はユーザーに代わって許可を廃止します。

HealthLake の暗号化キーのモニタリング

CloudTrail を使用して、カスタマーマネージド KMS キーを使用するときに HealthLake が AWS KMS ユーザーに代わって 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には、healthlake.amazonaws.com が userAgent フィールドに表示され、HealthLake によって行われたリクエストを明確に区別できます。

次の例は、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するために HealthLake 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するための CreateGrant、GenerateDataKey、Decrypt、および DescribeKey の CloudTrail イベントです。

以下は、CreateGrant を使用して HealthLake が顧客提供の KMS キーにアクセスすることを許可し、HealthLake がその KMS キー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する方法を示しています。

ユーザーは自分で許可を作成する必要はありません。HealthLake は、CreateGrant リクエストを AWS KMS に送信することで、ユーザーに代わって許可を作成します。の許可 AWS KMS は、顧客アカウントの AWS KMS キーへのアクセスを HealthLake に許可するために使用されます。

```
         {
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EXAMPLEROLE:Sampleuser01",
        "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": "2021-06-30T19:33:37Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "healthlake.amazonaws.com"
    },
    "eventTime": "2021-06-30T20:31:15Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "healthlake.amazonaws.com",
    "userAgent": "healthlake.amazonaws.com",
    "requestParameters": {
        "operations": [
            "CreateGrant",
            "Decrypt",
            "DescribeKey",
            "Encrypt",
            "GenerateDataKey",
            "GenerateDataKeyWithoutPlaintext",
            "ReEncryptFrom",
            "ReEncryptTo",
            "RetireGrant"
        ],
```

```
        "granteePrincipal": "healthlake.us-east-1.amazonaws.com",
        "keyId": "arn:aws:kms:us-east-1:111122223333:key/EXAMPLE_KEY_ARN",
        "retiringPrincipal": "healthlake.us-east-1.amazonaws.com"
    },
    "responseElements": {
        "grantId": "EXAMPLE_ID_01"
    },
    "requestID": "EXAMPLE_ID_02",
    "eventID": "EXAMPLE_ID_03",
    "readOnly": false,
    "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、GenerateDataKey を使用して、ユーザーが保存する前にデータを暗号化するために必要なアクセス許可を持っていることを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": "healthlake.amazonaws.com"
    },
    "eventTime": "2021-06-30T21:17:37Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "healthlake.amazonaws.com",
    "userAgent": "healthlake.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、HealthLake が Decrypt オペレーションを呼び出して、保存された暗号化された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": "healthlake.amazonaws.com"
    },
    "eventTime": "2021-06-30T21:21:59Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "healthlake.amazonaws.com",
    "userAgent": "healthlake.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、HealthLake が DescribeKey オペレーションを使用して、 AWS KMS 顧客所有の AWS KMS キーが使用可能な状態であるかどうかを確認し、機能していない場合のユーザー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": "healthlake.amazonaws.com"
    },
    "eventTime": "2021-07-01T18:36:36Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DescribeKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "healthlake.amazonaws.com",
    "userAgent": "healthlake.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"
}
```

詳細情報

以下のリソースは、保管時のデータの暗号化に関する詳細情報を提供します。

AWS Key Management Service の基本概念の詳細については、 AWS KMS ドキュメントを参照してください。

AWS KMS ドキュメントのセキュリティのベストプラクティスの詳細については、「」を参照してください。

AWS HealthLake の転送中の暗号化

AWS HealthLake 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。

AWS HealthLake の Identity and Access Management

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

トピック

オーディエンス

AWS Identity and Access Management (IAM) の使用方法は、ロールによって異なります。

- サービスユーザー - 機能にアクセスできない場合は、管理者にアクセス許可をリクエストします (「AWS HealthLake 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)。
- サービス管理者 - ユーザーアクセスを決定し、アクセス許可リクエストを送信します (「IAM での AWS HealthLake の仕組み」を参照)
- IAM 管理者 - アクセスを管理するためのポリシーを作成します (「AWS HealthLake のアイデンティティベースのポリシーの例」を参照)

アイデンティティを使用した認証

認証とは、ID 認証情報 AWS を使用して にサインインする方法です。、IAM ユーザー 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で認証される必要があります。

AWS IAM アイデンティティセンター (IAM Identity Center)、シングルサインオン認証、Google/Facebook 認証情報などの ID ソースからの認証情報を使用して、フェデレーティッド ID としてサインインできます。サインインの詳細については、「AWS サインイン ユーザーガイド」の「AWS アカウントにサインインする方法」を参照してください。

プログラムによるアクセスの場合、 は SDK と CLI AWS を提供してリクエストを暗号化して署名します。詳細については、「IAM ユーザーガイド」の「API リクエストに対するAWS 署名バージョン 4」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、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 AWS アカウント ルートユーザーと呼ばれる 1 つのサインインアイデンティティから始めます。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。ルートユーザー認証情報を必要とするタスクについては、「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。

フェデレーテッドアイデンティティ

ベストプラクティスとして、人間のユーザーが一時的な認証情報 AWS のサービス を使用して にアクセスするには、ID プロバイダーとのフェデレーションを使用する必要があります。

フェデレーション ID は、エンタープライズディレクトリ、ウェブ ID プロバイダー、または ID Directory Service ソースの認証情報 AWS のサービス を使用して にアクセスするユーザーです。フェデレーテッドアイデンティティは、一時的な認証情報を提供するロールを引き受けます。

アクセスを一元管理する場合は、 AWS IAM アイデンティティセンターをお勧めします。詳細については、「AWS IAM アイデンティティセンター 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、特定の個人やアプリケーションに対する特定のアクセス許可を持つアイデンティティです。長期認証情報を持つ IAM ユーザーの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。詳細については、IAM ユーザーガイドの「ID プロバイダーとのフェデレーションを使用して にアクセスする必要がある AWS」を参照してください。

IAM グループは、IAM ユーザーの集合を指定し、大量のユーザーに対するアクセス許可の管理を容易にします。詳細については、「IAM ユーザーガイド」の「IAM ユーザーに関するユースケース」を参照してください。

IAM ロール

IAM ロールは、特定のアクセス許可を持つアイデンティであり、一時的な認証情報を提供します。ユーザーから IAM ロール (コンソール) に切り替えるか、 または API オペレーションを呼び出すことで、ロールを引き受けることができます。 AWS CLI AWS 詳細については、「IAM ユーザーガイド」の「ロールを引き受けるための各種方法」を参照してください。

IAM ロールは、フェデレーションユーザーアクセス、一時的な IAM ユーザーのアクセス許可、クロスアカウントアクセス、クロスサービスアクセス、および Amazon EC2 で実行するアプリケーショ

ンに役立ちます。詳細については、IAM ユーザーガイド の IAM でのクロスアカウントリソースアクセス を参照してください。

ポリシーを使用したアクセスの管理

でアクセスを制御する AWS には、ポリシーを作成し、ID AWS またはリソースにアタッチします。ポリシーは、アイデンティティまたはリソースに関連付けられたときにアクセス許可を定義します。は、プリンシパルがリクエストを行うときにこれらのポリシー AWS を評価します。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、ポリシーを使用して、どのプリンシパルがどのリソースに対して、どのような条件でアクションを実行できるかを定義することで、誰が何にアクセスできるかを指定します。

デフォルトでは、ユーザーやロールにアクセス許可はありません。IAM 管理者は IAM ポリシーを作成してロールに追加し、このロールをユーザーが引き受けられるようにします。IAM ポリシーは、オペレーションの実行方法を問わず、アクセス許可を定義します。

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のポリシーは、アイデンティティ (ユーザー、グループ、またはロール) にアタッチできる 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、アイデンティティがどのリソースに対してどのような条件下でどのようなアクションを実行できるかを制御します。アイデンティティベースポリシーの作成方法については、IAM ユーザーガイド の カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する を参照してください。

アイデンティティベースのポリシーは、インラインポリシー (単一の ID に直接埋め込む) または管理ポリシー (複数の ID にアタッチされたスタンドアロンポリシー) にすることができます。管理ポリシーとインラインポリシーのいずれかを選択する方法については、「IAM ユーザーガイド」の「管理ポリシーとインラインポリシーのいずれかを選択する」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。例としては、IAM ロール信頼ポリシーや Amazon S3 バケットポリシーなどがあります。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。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。

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

その他のポリシータイプ

AWS は、より一般的なポリシータイプによって付与されるアクセス許可の最大数を設定できる追加のポリシータイプをサポートしています。

- アクセス許可の境界 – アイデンティティベースのポリシーで IAM エンティティに付与することのできるアクセス許可の数の上限を設定します。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 (SCP) - AWS Organizations内の組織または組織単位の最大のアクセス許可を指定します。詳細については、「AWS Organizations ユーザーガイド」の「サービスコントロールポリシー」を参照してください。
- リソースコントロールポリシー (RCP) – は、アカウント内のリソースで利用できる最大数のアクセス許可を定義します。詳細については、「AWS Organizations ユーザーガイド」の「リソースコントロールポリシー (RCP)」を参照してください。
- セッションポリシー – ロールまたはフェデレーションユーザーの一時セッションを作成する際にパラメータとして渡される高度なポリシーです。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アクセス許可を理解するのがさらに難しくなります。が複数のポリシータイプが関与する場合にリクエストを許可するかどうか AWS を決定する方法については、「IAM ユーザーガイド」の「ポリシー評価ロジック」を参照してください。

IAM での AWS HealthLake の仕組み

IAM を使用して HealthLake へのアクセスを管理する前に、HealthLake で使用できる IAM 機能を確認してください。

AWS HealthLake で使用できる IAM 機能

IAM 機能	HealthLake のサポート
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	なし

IAM 機能	HealthLake のサポート
ポリシーアクション	あり
ポリシーリソース	あり
ポリシー条件キー	あり
ACL	なし
ABAC (ポリシー内のタグ)	あり
一時的な認証情報	あり
プリンシパルアクセス権限	あり
サービスロール	あり
サービスリンクロール	いいえ

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

AWS HealthLake のアイデンティティベースのポリシー

アイデンティティベースのポリシーのサポート: あり

アイデンティティベースポリシーは、IAM ユーザー、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、および条件をコントロールします。アイデンティティベースポリシーの作成方法については、「IAM ユーザーガイド」の「カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する」を参照してください。

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。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。

AWS HealthLake のアイデンティティベースのポリシーの例

HealthLake 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 HealthLake のアイデンティティベースのポリシーの例。

AWS HealthLake 内のリソースベースのポリシー

リソースベースのポリシーのサポート: 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あげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスをコントロールできます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーで、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、全体のアカウント、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。詳細については、IAM ユーザーガイドのIAM でのクロスアカウントリソースアクセスを参照してください。

AWS HealthLake のポリシーアクション

ポリシーアクションのサポート: あり

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

JSON ポリシーの `Action` 要素にはポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。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。

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

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

```
healthlake
```

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、各アクションをカンマで区切ります。

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

HealthLake 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 HealthLake のアイデンティティベースのポリシーの例。

AWS HealthLake のポリシーリソース

ポリシーリソースのサポート: あり

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

`Resource` JSON ポリシー要素はアクションが適用されるオブジェクトを指定します。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。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": "*"
```

HealthLake リソースタイプとその ARNsAWS HealthLake で定義されるリソース」を参照してください。 各リソースの ARN を指定できるアクションについては、AWS HealthLake で定義されるアクション」を参照してください。

HealthLake 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 HealthLake のアイデンティティベースのポリシーの例。

AWS HealthLake のポリシー条件キー

サービス固有のポリシー条件キーのサポート: あり

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

Condition 要素は、定義された基準に基づいてステートメントが実行される時期を指定します。イコールや未満などの条件演算子を使用して条件式を作成して、ポリシーの条件とリクエスト内の値を一致させることができます。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。

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

HealthLake 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 HealthLake のアイデンティティベースのポリシーの例。

AWS HealthLake のアクセスコントロールリスト (ACLs)

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アクセス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。

AWS HealthLake を使用した属性ベースのアクセスコントロール (ABAC)

ABAC (ポリシー内のタグ) のサポート: あり

属性ベースのアクセス制御 (ABAC) は、タグと呼ばれる属性に基づいてアクセス許可を定義する認可戦略です。IAM エンティティと AWS リソースにタグをアタッチし、プリンシパルのタグがリソースのタグと一致するときにオペレーションを許可するように ABAC ポリシーを設計できます。

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

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

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

AWS HealthLake での一時的な認証情報の使用

一時的な認証情報のサポート: あり

一時的な認証情報は、 AWS リソースへの短期的なアクセスを提供し、フェデレーションまたはスイッチロールの使用時に自動的に作成されます。長期的なアクセスキーを使用する代わりに、一時的な認証情報を動的に生成 AWS することをお勧めします。詳細については、「IAM ユーザーガイド」の「IAM の一時的な認証情報」および「AWS のサービス と IAM との連携」を参照してください。

AWS HealthLake のクロスサービスプリンシパル許可

転送アクセスセッション (FAS) のサポート: あり

転送アクセスセッション (FAS) は、 を呼び出すプリンシパルのアクセス許可と AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストをリクエストする を使用します。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。

AWS HealthLake のサービスロール

サービスロールのサポート: あり

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

AWS HealthLake へのフルアクセスに必要なサービスロールとインラインポリシーについては、「」を参照してくださいセットアップ AWS HealthLake。

> **Warning**
>
> サービスロールのアクセス許可を変更すると、HealthLake の機能が破損する可能性があります。HealthLake が指示する場合にのみ、サービスロールを編集します。

AWS HealthLake のサービスにリンクされたロール

サービスにリンクされたロールのサポート: なし

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

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

AWS HealthLake のアイデンティティベースのポリシーの例

デフォルトでは、ユーザーとロールには HealthLake リソースを作成または変更するアクセス許可はありません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。

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

各リソースタイプの ARN の形式など、HealthLake で定義されるアクションとリソースタイプの詳細については、「サービス認可リファレンス」のAWS HealthLake 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。 ARNs

トピック

- ポリシーに関するベストプラクティス
- AWS HealthLake コンソールの使用
- での AWS HealthLake データストアへのアクセス Amazon Athena
- ユーザー自身のアクセス許可を表示することをユーザーに許可する

ポリシーに関するベストプラクティス

ID ベースのポリシーは、アカウント内で HealthLake リソースを作成、アクセス、または削除できるかどうかを決定します。これらのアクションでは、 AWS アカウントに費用が発生する場合があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS 管理ポリシーを開始し、最小特権のアクセス許可に移行 – ユーザーとワークロードにアクセス許可の付与を開始するには、多くの一般的なユースケースにアクセス許可を付与するAWS 管理ポリシーを使用します。これらは で使用できます AWS アカウント。ユースケースに固有の AWS カスタマー管理ポリシーを定義することで、アクセス許可をさらに減らすことをお勧めします。詳細については、IAM ユーザーガイド の AWS マネージドポリシー または ジョブ機能のAWS マネージドポリシー を参照してください。
- 最小特権を適用する – IAM ポリシーでアクセス許可を設定する場合は、タスクの実行に必要な許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権アクセス許可とも呼ばれています。IAM を使用して許可を適用する方法の詳細については、IAM ユーザーガイド の IAM でのポリシーとアクセス許可 を参照してください。
- IAM ポリシーで条件を使用してアクセスをさらに制限する - ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。たとえば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。条件を使用して、サービスアクションが などの特定の を通じて使用されている場合に AWS のサービス、サービスアクションへのアクセスを許可することもできます 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 アクセスアナライザー を使用して IAM ポリシーを検証し、安全で機能的な権限を確保する - IAM アクセスアナライザー は、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、IAM ユーザーガイド の IAM Access Analyzer でポリシーを検証する を参照してください。
- 多要素認証 (MFA) を要求する – で IAM ユーザーまたはルートユーザーを必要とするシナリオがある場合は AWS アカウント、MFA をオンにしてセキュリティを強化します。API オペレーションが呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細については、IAM ユーザーガイド の MFA を使用した安全な API アクセス を参照してください。

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

AWS HealthLake コンソールの使用

AWS HealthLake コンソールにアクセスするには、最小限のアクセス許可のセットが必要です。これらのアクセス許可により、 の HealthLake リソースの詳細を一覧表示および表示できます AWS アカウント。最小限必要な許可よりも制限が厳しいアイデンティティベースのポリシーを作成すると、そ

のポリシーを持つエンティティ (ユーザーまたはロール) に対してコンソールが意図したとおりに機能しません。

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

HealthLake へのフルアクセスについては、IAM ユーザーまたはロールに次のポリシーをアタッチします: `AmazonHealthLakeFullAccess`および `AWSLakeFormationDataAdmin`。また、サービスロールである HealthLake インラインポリシーをアタッチする必要があります。サービスロールとは、サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、IAM ユーザーガイドの AWS のサービスに許可を委任するロールを作成するを参照してください。必要なサービスロールを作成するインラインポリシーの詳細については、「」を参照してくださいセットアップ AWS HealthLake。また、 AWS Lake Formation コンソールまたは CLI を使用してHealthLake 管理者を AWS Lake Formation Data Lake 管理者として割り当てる必要があります。詳細については、「セットアップ AWS HealthLake」を参照してください。

での AWS HealthLake データストアへのアクセス Amazon Athena

ユーザーとロールに の HealthLake データストアへのアクセスを許可する場合は Amazon Athena、ロールまたはユーザーに次の IAM ポリシーをアタッチします。 `AmazonAthenaFullAccess` および `AmazonS3FullAccess`。 `Select` および アクセス`Describe`許可は、 によって管理されるテーブルにも必要です AWS Lake Formation。 AWS Lake Formation テーブルアクセス許可は、 AWS Lake Formation コンソールまたは CLI を介して AWS Lake Formation 管理者によって付与されます。詳細については、セットアップ AWS HealthLakeを参照してください。

ユーザー自身のアクセス許可を表示することをユーザーに許可する

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

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

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

AWS AWS HealthLake の マネージドポリシー

AWS 管理ポリシーは、 によって作成および管理されるスタンドアロンポリシーです AWS。 AWS 管理ポリシーは、ユーザー、グループ、ロールにアクセス許可の割り当てを開始できるように、多くの一般的なユースケースにアクセス許可を付与するように設計されています。

AWS 管理ポリシーは、すべての AWS お客様が使用できるため、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注意してください。ユースケースに固有のカスタマー管理ポリシーを定義して、アクセス許可を絞り込むことをお勧めします。

AWS 管理ポリシーで定義されているアクセス許可は変更できません。が AWS マネージドポリシーで定義されたアクセス許可 AWS を更新すると、ポリシーがアタッチされているすべてのプリンシパ

ル ID (ユーザー、グループ、ロール) に影響します。 AWS は、新しい が起動されるか、新しい API オペレーション AWS のサービス が既存のサービスで使用できるようになったときに、 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。

AWS マネージドポリシー: AmazonHealthLakeFullAccess

このAmazonHealthLakeFullAccessポリシーは HealthLake へのフルアクセスを提供します。このポリシーをユーザーまたはロールにアタッチすると、ユーザーは HealthLake を使用して HealthLake のデータにアクセス、クエリ、インポート、エクスポートできます。HealthLake で多くの一般的なアクションを実行するには、ユーザーまたはロールにポリシーを追加する必要があります。詳細については、セットアップ AWS HealthLake「」およびHealthLake オペレーションとアクセス許可」を参照してください。

AmazonHealthLakeFullAccess ポリシーを IAM アイデンティティにアタッチできます。

このポリシーは、ユーザーとロールが HealthLake でクエリ、検索、インポート、エクスポートできるようにする管理アクセス許可と寄稿者アクセス許可を付与します。また、これらのアクセス許可を持つユーザーとロールに代わって HealthLake がアクションを実行できるようにします。

アクセス許可の詳細

このポリシーには、次のステートメントが含まれます。

JSON

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

```
    "Action": [
     "healthlake:*",
     "s3:ListAllMyBuckets",
     "s3:ListBucket",
     "s3:GetBucketLocation",
     "iam:ListRoles"
    ],
    "Resource": "*",
    "Effect": "Allow"
   },
   {
    "Effect": "Allow",
    "Action": "iam:PassRole",
    "Resource": "*",
    "Condition": {
     "StringEquals": {
      "iam:PassedToService": "healthlake.amazonaws.com"
     }
    }
   }
  ]
}
```

AWS マネージドポリシー: AmazonHealthLakeReadOnlyAccess

AmazonHealthLakeReadOnlyAccess ポリシーは、HealthLake および他の AWS サービスの関連リソースへの読み取り専用アクセスとアクセス許可を付与します。このポリシーは、HealthLake データストアをクエリおよび表示する権限を付与するユーザーに適用しますが、作成または変更を行う権限は付与しません。

AmazonHealthLakeReadOnlyAccess ポリシーを IAM アイデンティティにアタッチできます。

このポリシーは、ユーザーとロールが HealthLake をクエリできるようにする######アクセス許可を付与します。

アクセス許可の詳細

このポリシーには、次のステートメントが含まれます。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "healthlake:ListFHIRDatastores",
                "healthlake:DescribeFHIRDatastore",
                "healthlake:DescribeFHIRImportJob",
                "healthlake:DescribeFHIRExportJob",
                "healthlake:GetCapabilities",
                "healthlake:ReadResource",
                "healthlake:SearchWithGet",
                "healthlake:SearchWithPost",
                "healthlake:SearchEverything"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

HealthLake オペレーションとアクセス許可

次の表に、HealthLake の一般的なオペレーションとその実行に必要なアクセス許可を示します。

HealthLake オペレーション	必要なアクセス許可
HealthLake でデータストアを作成する	AmazonHealthLakeFullAccess 、AmazonLakeFormationDataAdmin 、インラインポリシー、および によって管理される AWS Lake Formation 管理者権限 AWS Lake Formation
HealthLake でデータストアを削除する	AmazonHealthLakeFullAccess に よって管理される AmazonLakeFormatio

HealthLake オペレーション	必要なアクセス許可
	nDataAdmin 、、インラインポリシー、管理者 AWS Lake Formation 権限 AWS Lake Formation
HealthLake のデータストアを一覧表示、検索、またはクエリする	AmazonHealthLakeReadOnlyAccess
を使用してデータストアをクエリする Amazon Athena	AmazonAthenaFullAccess によって管理されるテーブルに対する 、AmazonS3FullAccess 、 AWS Lake Formation Selectおよび アクセスDescribe許可 AWS Lake Formation
HealthLake からデータをインポートする	「インポートジョブのアクセス許可の設定」を参照してください。
HealthLake からデータをエクスポートする	「エクスポートジョブのアクセス許可の設定」を参照してください。

AWS 管理ポリシーに対する HealthLake の更新

HealthLake の AWS マネージドポリシーの更新に関する詳細を、このサービスがこれらの変更の追跡を開始した時点から表示します。このページの変更に関する自動アラートについては、HealthLake ドキュメント履歴ページの RSS フィードにサブスクライブしてください。

変更	説明	日付
AmazonHealthLakeFullAccess	AmazonHealthLakeFullAccess HealthLake へのフルアクセスを許可するために必要な ポリシー。	2022 年 11 月 14 日
AmazonHealthLakeReadOnlyAccess	AmazonHealthLakeReadOnlyAccess HealthLake への読み取り専用アクセスに必要な ポリシー。	2022 年 11 月 14 日

変更	説明	日付
HealthLake が変更の追跡を開始しました	HealthLake は、 AWS 管理ポリシーの変更の追跡を開始しました。	2022 年 11 月 14 日

AWS HealthLake 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

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

トピック

- AWS HealthLake でアクションを実行する権限がない
- iam:PassRole を実行する権限がない
- 自分の AWS アカウント以外のユーザーに my AWS HealthLake リソースへのアクセスを許可したい

AWS HealthLake でアクションを実行する権限がない

でアクションを実行する権限がないと AWS マネジメントコンソール 通知された場合は、管理者に連絡してサポートを依頼する必要があります。管理者とは、ユーザーにユーザー名とパスワードを提供した人です。

以下のエラー例は、`mateojackson` 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情報を表示しようとしているが、架空の `healthlake:`*`GetWidget`* アクセス許可がないという場合に発生します。

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

この場合、Mateo は、`healthlake:`*`GetWidget`* アクションを使用して *`my-example-widget`* リソースにアクセスできるように、ポリシーの更新を管理者に依頼します。

iam:PassRole を実行する権限がない

`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 HealthLake にロールを渡すことができるようにする必要があります。

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

次の例のエラーは、 という IAM ユーザーがコンソールを使用して HealthLake `marymajor` でアクションを実行しようとすると発生します。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アクセス許可が必要です。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メアリーに `iam:PassRole` アクションの実行を許可する必要があります。

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

自分の AWS アカウント以外のユーザーに my AWS HealthLake リソース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:

- HealthLake がこれらの機能をサポートしているかどうかを確認するには、「」を参照してくださいIAM での AWS HealthLake の仕組み。
- 所有 AWS アカウント している のリソースへのアクセスを提供する方法については、「IAM ユーザーガイド」の「所有 AWS アカウント している別の の IAM ユーザーへのアクセスを提供する」を参照してください。
- リソースへのアクセスをサードパーティーに提供する方法については AWS アカウント、IAM ユーザーガイドの「サードパーティー AWS アカウント が所有する へのアクセスを提供する」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、IAM ユーザーガイド の 外部で認証されたユーザー (ID フェデレーション) へのアクセスの許可 を参照してください。

- クロスアカウントアクセスにおけるロールとリソースベースのポリシーの使用方法の違いについては、「IAM ユーザーガイド」の「IAM でのクロスアカウントのリソースへのアクセス」を参照してください。

AWS HealthLake のコンプライアンス検証

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

AWS のサービス が特定のコンプライアンスプログラムの範囲内にあるかどうかを確認するには、「コンプライアンスAWS のサービス プログラムによる対象範囲内」の「コンプライアンス」を参照して、関心のあるコンプライアンスプログラムを選択します。一般的な情報については、AWS 「コンプライアンスプログラム」を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、「Downloading Reports in AWS Artifact」を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS のサービス は、お客様のデータの機密性、貴社のコンプライアンス目的、適用される法律および規制によって決まります。を使用する際のコンプライアンス責任の詳細については AWS のサービス、AWS 「 セキュリティドキュメント」を参照してください。

AWS HealthLake のインフラストラクチャセキュリティ

マネージドサービスである AWS HealthLake は、ホワイトペーパー「Amazon Web Services: セキュリティプロセスの概要」に記載されている AWS グローバルネットワークセキュリティ手順で保護されています。

AWS 公開された API コールを使用して、ネットワーク経由で HealthLake にアクセスします。クライアントで Transport Layer Security (TLS) 1.0 以降がサポートされている必要があります。TLS 1.2 以降が推奨されています。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。または、AWS Security Token Service (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

を使用した AWS HealthLake リソースの作成 AWS CloudFormation

AWS HealthLake は と統合されています。これは AWS CloudFormation、 AWS リソースとインフラストラクチャの作成と管理に費やす時間を短縮できるように、リソースのモデル化とセットアップに役立つサービスです。必要なすべての AWS リソースを記述するテンプレートを作成し、それらのリソースを CloudFormation プロビジョニングして設定します。

を使用すると CloudFormation、テンプレートを再利用して HealthLake リソースを一貫して繰り返しセットアップできます。リソースを 1 回記述し、複数の AWS アカウント およびリージョンで同じリソースを何度もプロビジョニングします。

HealthLake と CloudFormation テンプレート

HealthLake および関連サービスのリソースをプロビジョニングおよび設定するには、CloudFormation テンプレートを理解する必要があります。テンプレートは、JSON や 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は、 CloudFormation スタックでプロビジョニングするリソースを記述します。JSON または YAML に慣れていない場合は、CloudFormation デザイナー を使用して CloudFormation テンプレートの使用を開始できます。詳細については、「AWS CloudFormation ユーザーガイド」の「CloudFormation Designer とは」を参照してください。

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

の詳細 CloudFormation

詳細については CloudFormation、次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
- 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 HealthLake とインターフェイス VPC エンドポイント (AWS PrivateLink)

インターフェイス VPC エンドポイントを作成することで、VPC と AWS HealthLake 間のプライベート接続を確立できます。インターフェイス VPC エンドポイントは、HealthLake、インターネットゲートウェイAWS PrivateLink、NAT デバイス、VPN 接続、または Direct Connect 接続のない APIs にプライベートにアクセスするために使用できるテクノロジーである を利用しています。VPC 内のインスタンスは、パブリック IP アドレスがなくても HealthLake; APIs。VPC と HealthLake 間のトラフィック。 は Amazon ネットワークを離れません。

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「インターフェイス VPC エンドポイント (AWS PrivateLink)」を参照してください。

HealthLake VPC エンドポイントに関する考慮事項

HealthLake のインターフェイス 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イ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。

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

HealthLake 用のインターフェイス VPC エンドポイントの作成

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

次のサービス名を使用して、HealthLake の VPC エンドポイントを作成します。

- com.amazonaws.*region*.healthlake

エンドポイントのプライベート DNS を有効にすると、リージョンのデフォルトの DNS 名を使用して HealthLake に API リクエストを行うことができます。例えば、*healthlake.us-east-1.amazonaws.com*。

詳細については、「Amazon VPC ユーザーガイド」の「インターフェイ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。

HealthLake の VPC エンドポイントポリシーの作成

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

- アクションを実行できるプリンシパル。
- 実行可能なアクション。
- 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 エンドポイントでサービスへのアクセスを制御する」を参照してください。

例: HealthLake アクションの VPC エンドポイントポリシー

HealthLake のエンドポイントポリシーの例を次に示します。エンドポイントにアタッチすると、このポリシーはすべてのリソースのすべてのプリンシパルに HealthLake `CreateFHIRDatastore`アクションへのアクセスを許可します。

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "healthlake:create-fhir-datastore"
         ],
         "Resource":"*"
      }
   ]
}
```

AWS HealthLake のセキュリティのベストプラクティス

AWS HealthLake には、独自のセキュリティポリシーを開発および実装する際に考慮すべき多くのセキュリティ機能が用意されています。以下のベストプラクティスは一般的なガイドラインであり、完全なセキュリティソリューションを説明するものではありません。これらのベストプラクティスはお

客様の環境に適切ではないか、十分ではない場合があるため、これらは指示ではなく、有用な考慮事項と見なしてください。

- 最小特権アクセスの実装
- 可能な限り、Customer-Managed-Keys (CMKs) を使用してデータを暗号化します。CMKs「Amazon Key Management Service」を参照してください。
- データストアで PHI または PII をクエリするときは、GET ではなく POST で検索を使用します。
- 機密性の高い重要な監査機能へのアクセスを制限します。
- 更新 API または一括インポート APIs を使用してリソースを作成するときは、データストアとジョブの名前を含む PHI または PII を、表示可能なフィールドまたは論理 FHIR ID (LID) で使用しないでください。
- 作成、読み取り、更新、削除、または検索リクエストを送信するときは、HTTP ヘッダーで PHI を使用しないでください。
- AWS CloudTrail が Audit AWS HealthLake の使用と を有効にして、予期しないアクティビティがないことを確認します。
- Amazon S3 バケットを安全に使用するためのベストプラクティスを確認します。詳細については、Amazon S3ユーザーガイド」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。

AWS HealthLake の耐障害性

AWS グローバルインフラストラクチャは、 AWS リージョンとアベイラビリティーゾーンを中心に構築されています。 AWS リージョンは、低レイテンシー、高スループット、高度に冗長なネットワークで接続された複数の物理的に分離されたアベイラビリティーゾーンと分離されたアベイラビリティーゾーンを提供します。アベイラビリティーゾーンでは、ゾーン間で中断することなく自動的にフェイルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性、フォールトトレランス、および拡張性が優れています。

データまたはアプリケーションを遠方でレプリケートする必要がある場合は、 AWS ローカルリージョンを使用します。 AWS ローカルリージョンは、既存の AWS リージョンを補完するように設計された単一のデータセンターです。すべての AWS リージョンと同様に、 AWS ローカルリージョンは他の AWS リージョンから完全に分離されています。

AWS リージョンとアベイラビリティーゾーンの詳細については、AWS 「グローバルインフラストラクチャ」を参照してください。

AWS HealthLake リファレンス

以下のサポートリファレンスマテリアルは、SMART on FHIR、FHIR、および で利用できます AWS HealthLake。

> **Note**
>
> すべてのネイティブ HealthLake アクションとデータ型については、別のリファレンスで説明します。詳細については、「AWS HealthLake APIリファレンス」を参照してください。

トピック

- の FHIR での SMART サポート AWS HealthLake
- の FHIR R4 サポート AWS HealthLake
- のコンプライアンスリファレンス AWS HealthLake
- のサポートリファレンス AWS HealthLake

の FHIR での SMART サポート AWS HealthLake

FHIR 対応 HealthLake データストアの代替医療アプリケーションと再利用可能なテクノロジー (SMART) を使用すると、FHIR 準拠アプリケーション上の SMART にアクセスできます。HealthLake データにアクセスするには、サードパーティーの認可サーバーを使用してリクエストを認証および承認します。したがって、 を介してユーザー認証情報を管理する代わりに AWS Identity and Access Management、FHIR 準拠の認可サーバーで SMART を使用して管理します。

> **Note**
>
> HealthLake は、FHIR バージョン 1.0 および 2.0 で SMART をサポートしています。これらのフレームワークの詳細については、FHIR R4 ドキュメントの「SMART App Launch」を参照してください。
> HealthLake データストアは、FHIR リクエストでの SMART の以下の認証および認可フレームワークをサポートしています。
>
> - OpenID (AuthN): 個人またはクライアントアプリケーションを認証するには、誰 (または何) であると主張します。

- OAuth 2.0 (AuthZ): 認証されたリクエストが読み書きできる HealthLake データストアの FHIR リソースを承認します。これは、認可サーバーで設定されたスコープによって定義されます。

SMART on FHIR 対応データストアは、 AWS CLI または AWS SDKs を使用して作成できます。詳細については、「HealthLake データストアの作成」を参照してください。

トピック

FHIR での SMART の開始方法

以下のトピックでは、 AWS HealthLake の FHIR 認可で SMART の使用を開始する方法について説明します。これには、 AWS アカウントでプロビジョニングする必要があるリソース、FHIR 対応 HealthLake データストアでの SMART の作成、および FHIR クライアントアプリケーションが認可サーバーと HealthLake データストアとやり取りする方法の例が含まれます。

トピック

FHIR での SMART のリソースの設定

次のステップでは、HealthLake が FHIR リクエストで SMART を処理する方法と、それらを成功させるために必要なリソースを定義します。以下の要素は、ワークフロー内で連携して SMART on FHIR リクエストを行います。

- エンドユーザー: 一般的に、患者または臨床医は、サードパーティーの SMART on FHIR アプリケーションを使用して HealthLake データストアのデータにアクセスします。
- SMART on FHIR アプリケーション (クライアントアプリケーションと呼ばれます): HealthLake データストアにあるデータにアクセスしたいアプリケーション。
- 認可サーバー: ユーザーを認証し、アクセストークンを発行できる OpenID Connect 準拠サーバー。
- HealthLake データストア: Lambda 関数を使用してベアラートークンを提供する FHIR REST リクエストに応答する、SMART on FHIR 対応 HealthLake データストア。

これらの要素を連携させるには、次のリソースを作成する必要があります。

> **Note**
>
> 認可サーバーを設定して必要なスコープを定義し、トークンイントロスペクションを処理する AWS Lambda 関数を作成したら、FHIR 対応 HealthLake データストアで SMART を作成することをお勧めします。

1. 認可サーバーエンドポイントを設定する

SMART on FHIR フレームワークを使用するには、データストアで行われた FHIR REST リクエストを検証できるサードパーティー認可サーバーを設定する必要があります。詳細については、「SMART on FHIR の HealthLake 認証要件」を参照してください。

2. HealthLake データストアのアクセスレベルを制御するための認可サーバーのスコープを定義する

SMART on FHIR フレームワークは、OAuth スコープを使用して、認証されたリクエストがアクセスできる FHIR リソースとその範囲を決定します。スコープの定義は、最小特権を設計する方法です。詳細については、「HealthLake でサポートされている FHIR OAuth 2.0 スコープでの SMART」を参照してください。

3. トークンイントロスペクションを実行できる AWS Lambda 関数を設定する

クライアントアプリケーションから SMART on FHIR 対応データストアに送信された FHIR REST リクエストには、JSON ウェブトークン (JWT) が含まれています。詳細については、「JWT のデコード」を参照してください。

4. FHIR 対応 HealthLake データストアでの SMART の作成

FHIR HealthLake データストアで SMART を作成するには、 を指定する必要がありますIdentityProviderConfiguration。詳細については、「HealthLake データストアの作成」を参照してください。

SMART on FHIR のクライアントアプリケーションワークフロー

次のセクションでは、クライアントアプリケーションを起動し、SMART on FHIR のコンテキスト内で HealthLake データストアで FHIR REST リクエストを正常に実行する方法について説明します。

1. クライアントアプリケーションを使用して Well-Known Uniform Resource Identifier に**GET**リクエストを行う

SMART 対応クライアントアプリケーションは、HealthLake データストアの承認エンドポイントを検索するGETリクエストを行う必要があります。これは、Well-Known Uniform Resource Identifier (URI) リクエストを介して行われます。詳細については、「FHIR 検出ドキュメントの SMART の取得」を参照してください。

2. アクセスとスコープのリクエスト

クライアントアプリケーションは、ユーザーがログインできるように、認可サーバーの認可エンドポイントを使用します。このプロセスはユーザーを認証します。スコープは、クライアントアプリケーションがアクセスできる HealthLake データストア内の FHIR リソースを定義するために使用されます。詳細については、「HealthLake でサポートされている FHIR OAuth 2.0 スコープでの SMART」を参照してください。

3. アクセストークン

ユーザーが認証されたので、クライアントアプリケーションは認可サーバーから JWT アクセストークンを受け取ります。このトークンは、クライアントアプリケーションが HealthLake に FHIR REST リクエストを送信するときに提供されます。詳細については、「トークンの検証」を参照してください。

4. SMART on FHIR 対応 HealthLake データストアで FHIR REST API リクエストを行う

クライアントアプリケーションは、認可サーバーによって提供されるアクセストークンを使用して HealthLake データストアエンドポイントに FHIR REST API リクエストを送信できるようになりました。詳細については、「SMART 対応 HealthLake データストアでの FHIR REST API リクエストの実行」を参照してください。

5. JWT アクセストークンを検証する

FHIR REST リクエストで送信されたアクセストークンを検証するには、Lambda 関数を使用します。詳細については、「を使用したトークンの検証 AWS Lambda」を参照してください。

SMART on FHIR の HealthLake 認証要件

SMART on FHIR 対応 HealthLake データストアの FHIR リソースにアクセスするには、クライアントアプリケーションが OAuth 2.0 準拠の認可サーバーによって承認され、FHIR REST API リクエストの一部として OAuth ベアラートークンを提示する必要があります。認可サーバーのエンドポイントを検索するには、Uniform Resource Identifier を介して FHIR Discovery Document の HealthLake SMART `Well-Known` を使用します。このプロセスについては、「FHIR 検出ドキュメントの SMART の取得」を参照してください。

SMART on FHIR HealthLake データストアを作成するときは、`CreateFHIRDatastore`リクエストの `metadata`要素で認可サーバーのエンドポイントとトークンエンドポイントを定義する必要があります。`metadata` 要素の定義の詳細については、「」を参照してくださいHealthLake データストアの作成。

認可サーバーエンドポイントを使用して、クライアントアプリケーションは認可サービスでユーザーを認証します。承認および認証されると、認可サービスによって JSON ウェブトークン (JWT) が生成され、クライアントアプリケーションに渡されます。このトークンには、クライアントアプリケーションが使用できる FHIR リソーススコープが含まれています。これにより、ユーザーがアクセスできるデータが制限されます。オプションで、起動スコープが指定されている場合、レスポンスにはそれらの詳細が含まれます。HealthLake でサポートされている FHIR スコープの SMART の詳細については、「」を参照してくださいHealthLake でサポートされている FHIR OAuth 2.0 スコープでの SMART。

認可サーバーによって付与された JWT を使用して、クライアントアプリケーションは FHIR が有効な HealthLake データストアで SMART への FHIR REST API 呼び出しを行います。JWT を検証してデコードするには、Lambda 関数を作成する必要があります。HealthLake は、FHIR REST API リクエストを受信すると、ユーザーに代わってこの Lambda 関数を呼び出します。スターター Lambda 関数の例については、「」を参照してくださいを使用したトークンの検証 AWS Lambda。

FHIR 対応 HealthLake データストアで SMART を作成するために必要な認可サーバー要素

`CreateFHIRDatastore` リクエストでは、 `IdentityProviderConfiguration` オブジェクトの `metadata`要素の一部として認可エンドポイントとトークンエンドポイントを指定する必要があります。認可エンドポイントとトークンエンドポイントの両方が必要です。これ

をCreateFHIRDatastoreリクエストで指定する方法の例については、「」を参照してくださいHealthLake データストアの作成。

SMART on FHIR 対応 HealthLake データストアで FHIR REST API リクエストを完了するために必要なクレーム

AWS Lambda 関数が SMART on FHIR 対応 HealthLake データストアでの有効な FHIR REST API リクエストであるためには、関数に次のクレームが含まれている必要があります。

- nbf: (前ではない) クレーム — 「nbf」クレーム (前ではない) は、JWT を処理に受け入れることができない時刻を識別します。「nbf」クレームの処理では、現在の日付/時刻が「nbf」クレームに記載されている以前の日付/時刻以降である必要があります。提供されているサンプル Lambda 関数は、サーバーレスポンスiatから に変換しますnbf。
- exp: (有効期限) クレーム — 「有効期限」 (有効期限) クレームは、JWT を処理に受け入れてはならない 以降の有効期限を識別します。
- isAuthorized: ブール値を に設定しますTrue。認可サーバーでリクエストが承認されたことを示します。
- aud: (対象者) クレーム — 「aud」 (対象者) クレームは、JWT の対象となる受取人を識別します。これは、FHIR 対応 HealthLake データストアエンドポイントの SMART である必要があります。
- scope: これは少なくとも 1 つの FHIR リソース関連のスコープである必要があります。このスコープは認可サーバーで定義されます。HealthLake で受け入れられる FHIR リソース関連のスコープの詳細については、「」を参照してくださいHealthLake の FHIR リソーススコープに関する SMART。

HealthLake でサポートされている FHIR OAuth 2.0 スコープでの SMART

HealthLake は認可プロトコルとして OAuth 2.0 を使用します。認可サーバーでこのプロトコルを使用すると、クライアントアプリケーションがアクセスできる FHIR リソースの HealthLake データストアのアクセス許可 (作成、読み取り、更新、削除、検索) を定義できます。

SMART on FHIR フレームワークは、認可サーバーからリクエストできる一連のスコープを定義します。たとえば、患者が検査結果を表示したり、連絡先の詳細を表示したりできるようにのみ設計されたクライアントアプリケーションには、readスコープをリクエストする権限のみが必要です。

> **Note**
>
> HealthLake は、以下で説明するように、FHIR V1 と V2 の両方で SMART をサポートします。SMART on FHIR `AuthorizationStrategy`は、データストアの作成時に次の 3 つの値のいずれかに設定されます。
>
> - `SMART_ON_FHIR_V1` – (読み取り/検索) および `read` (`write`create/update/delete) アクセス許可を含む FHIR V1 での SMART のみをサポートします。
> - `SMART_ON_FHIR` – 、、、`create`、および アクセス`search`許可を含む FHIR V1 `delete`および V2 での SMART `read` `update`のサポート。
> - `AWS_AUTH` – デフォルトの AWS HealthLake 認可戦略。FHIR 上の SMART とは関連していません。

スタンドアロン起動スコープ

HealthLake は、スタンドアロン起動モードスコープ をサポートしています`launch/patient`。

スタンドアロン起動モードでは、ユーザーと患者がクライアントアプリケーションに知られていないため、クライアントアプリケーションは患者の臨床データへのアクセスをリクエストします。したがって、クライアントアプリケーションの認可リクエストは、患者スコープが返されることを明示的に要求します。認証が成功すると、認可サーバーはリクエストされた起動患者スコープを含むアクセストークンを発行します。必要な患者コンテキストは、認可サーバーのレスポンスでアクセストークンとともに提供されます。

サポートされている起動モードスコープ

スコープ	説明
`launch/patient`	OAuth 2.0 認可リクエストのパラメータ。認可レスポンスで患者データが返されるように要求します。

HealthLake の FHIR リソーススコープに関する SMART

HealthLake は、FHIR リソーススコープで 3 つのレベルの SMART を定義します。

- `patient` スコープは、単一の患者に関する特定のデータへのアクセスを許可します。
- `user` スコープは、ユーザーがアクセスできる特定のデータへのアクセスを許可します。

- `system` スコープは、HealthLake データストアにあるすべての FHIR リソースへのアクセスを許可します。

以下のセクションでは、SMART on FHIR V1 または SMART on FHIR V2 を使用して FHIR リソーススコープを構築するための構文を一覧表示します。

> **Note**
>
> SMART on FHIR 認可戦略は、データストアの作成時に設定されます。詳細については、AWS HealthLake API リファレンス`AuthorizationStrategy`の「」を参照してください。

HealthLake でサポートされている FHIR V1 スコープの SMART

FHIR V1 で SMART を使用する場合、HealthLake の FHIR リソーススコープを構築するための一般的な構文は次のとおりです。次の例の URL パス全体を表示するには、コピーボタンをスクロールします。

```
('patient' | 'user' | 'system') '/' (fhir-resource | '*') '.' ('read' | 'write' | '*')
```

FHIR v1 でサポートされる認可スコープでの SMART

スコープ構文	スコープの例	結果
`patient/(fhir-resource | '*').('read' | 'write' | '*')`	`patient/AllergyIntolerance.*`	患者クライアントアプリケーションには、記録されたすべてのアレルギーに対するインスタンスレベルの読み取り/書き込みアクセス権があります。
`user/(fhir-resource | '*').('read' | 'write' | '*')`	`user/Observation.read`	ユーザークライアントアプリケーションには、記録されたすべての観測値へのイ

スコープ構文	スコープの例	結果
		ンスタンスレベルの読み取り/書き込みアクセス権があります。
`system/('read' | 'write' | *)`	`system/*.*`	システムクライアントアプリケーションには、すべての FHIR リソースデータへの読み取り/書き込みアクセス権があります。

HealthLake でサポートされている FHIR V2 スコープの SMART

FHIR V2 で SMART を使用する場合、HealthLake の FHIR リソーススコープを構築するための一般的な構文は次のとおりです。次の例の URL パス全体を表示するには、コピーボタンをスクロールします。

```
('patient' | 'user' | 'system') '/' (fhir-resource | '*') '.' ('c' | 'r' | 'u' | 'd' |
 's')
```

> **Note**
>
> FHIR V2 で SMART を使用するには、 値を`IdentityProviderConfiguration`データ型のメンバーであるメタデータ`capabilities`文字列`permission-v2`に渡す必要があります。
>
> HealthLake はきめ細かなスコープをサポートしています。詳細については、「FHIR 米国コア実装ガイド」の「サポートされている詳細なスコープ」を参照してください。

FHIR V2 でサポートされる SMART 認可スコープ

スコープ構文	V1 スコープの例	結果
`patient/Observation.rs`	`user/Observation.read`	現在の患者のObservation リソースを読み取って検索するアクセス許可。
`system/*.cruds`	`system/*.*`	システムクライアントアプリケーションには、すべての FHIR リソースデータへの完全なcreate/read/update/削除/検索アクセス権があります。

を使用したトークンの検証 AWS Lambda

FHIR 対応データストアで HealthLake SMART を作成する場合は、`CreateFHIRDatastore`リクエストで AWS Lambda 関数の ARN を指定する必要があります。Lambda 関数の ARN は、`IdpLambdaArn`パラメータを使用して `IdentityProviderConfiguration` オブジェクトで指定されます。

SMART on FHIR 対応データストアを作成する前に、Lambda 関数を作成する必要があります。データストアを作成すると、Lambda ARN を変更することはできません。データストアの作成時に指定した Lambda ARN を表示するには、 `DescribeFHIRDatastore` API アクションを使用します。

FHIR REST リクエストが SMART on FHIR 対応データストアで成功するには、Lambda 関数が以下を実行する必要があります。

- HealthLake データストアエンドポイントに 1 秒未満でレスポンスを返します。
- クライアントアプリケーションによって送信された REST API リクエストの承認ヘッダーで提供されるアクセストークンをデコードします。
- FHIR REST API リクエストを実行するのに十分なアクセス許可を持つ IAM サービスロールを割り当てます。

- FHIR REST API リクエストを完了するには、次のクレームが必要です。詳細については必要なクレームを参照してください。
  - nbf
  - exp
  - isAuthorized
  - aud
  - scope

Lambda を使用する場合は、Lambda 関数に加えて実行ロールとリソースベースのポリシーを作成する必要があります。Lambda 関数の実行ロールは、実行時に必要な AWS のサービスおよびリソースにアクセスするためのアクセス許可を関数に付与する IAM ロールです。指定するリソースベースのポリシーでは、HealthLake がユーザーに代わって関数を呼び出すことを許可する必要があります。

このトピックのセクションでは、クライアントアプリケーションからのリクエスト例とデコードされたレスポンス、 AWS Lambda 関数の作成に必要なステップ、および HealthLake が引き受けることができるリソースベースのポリシーの作成方法について説明します。

- パート 1: Lambda 関数の作成
- パート 2: AWS Lambda 関数で使用される HealthLake サービスロールの作成
- パート 3: Lambda 関数の実行ロールの更新
- パート 4: Lambda 関数へのリソースポリシーの追加
- パート 5: Lambda 関数の同時実行のプロビジョニング

AWS Lambda 関数の作成

このトピックで作成された Lambda 関数は、HealthLake が SMART on FHIR 対応データストアへのリクエストを受信するとトリガーされます。クライアントアプリケーションからのリクエストには、REST API コールと、アクセストークンを含む認可ヘッダーが含まれています。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
Authorization: Bearer i8hweunweunweofiwweoijewiwe
```

このトピックの Lambda 関数の例では AWS Secrets Manager 、 を使用して認可サーバーに関連する認証情報を隠します。Lambda 関数で認可サーバーログインの詳細を直接提供しないことを強くお勧めします。

Example認可ベアラートークンを含む FHIR REST リクエストの検証

Lambda 関数の例は、SMART on FHIR 対応データストアに送信された FHIR REST リクエストを検証する方法を示しています。この Lambda 関数を実装する手順については、 step-by-steps 「」を参照してくださいを使用した Lambda 関数の作成 AWS マネジメントコンソール。

FHIR REST API リクエストに有効なデータストアエンドポイント、アクセストークン、REST オペレーションが含まれていない場合、Lambda 関数は失敗します。必要な認可サーバー要素の詳細については、「」を参照してください必要なクレーム。

```
import base64
import boto3
import logging
import json
import os
from urllib import request, parse

logger = logging.getLogger()
logger.setLevel(logging.INFO)

## Uses Secrets manager to gain access to the access key ID and secret access key for
 the authorization server
client = boto3.client('secretsmanager', region_name="region-of-datastore")
response = client.get_secret_value(SecretId='name-specified-by-customer-in-
secretsmanager')
secret = json.loads(response['SecretString'])
client_id = secret['client_id']
client_secret = secret['client_secret']


unencoded_auth = f'{client_id}:{client_secret}'
headers = {
  'Authorization': f'Basic {base64.b64encode(unencoded_auth.encode()).decode()}',
  'Content-Type': 'application/x-www-form-urlencoded'
}

auth_endpoint = os.environ['auth-server-base-url'] # Base URL of the Authorization
 server
user_role_arn = os.environ['iam-role-arn'] # The IAM role client application will use
 to complete the HTTP request on the datastore

def lambda_handler(event, context):
```

```
    if 'datastoreEndpoint' not in event or 'operationName' not in event or
 'bearerToken' not in event:
    return {}

    datastore_endpoint = event['datastoreEndpoint']
    operation_name = event['operationName']
    bearer_token = event['bearerToken']
    logger.info('Datastore Endpoint [{}], Operation Name:
 [{}]'.format(datastore_endpoint, operation_name))

    ## To validate the token
    auth_response = auth_with_provider(bearer_token)
    logger.info('Auth response: [{}]'.format(auth_response))
    auth_payload = json.loads(auth_response)
    ## Required parameters needed to be sent to the datastore endpoint for the HTTP
 request to go through
    auth_payload["isAuthorized"] = bool(auth_payload["active"])
    auth_payload["nbf"] = auth_payload["iat"]
    return {"authPayload": auth_payload, "iamRoleARN": user_role_arn}

## access the server
def auth_with_provider(token):
    data = {'token': token, 'token_type_hint': 'access_token'}
    req = request.Request(url=auth_endpoint + '/v1/introspect',
 data=parse.urlencode(data).encode(), headers=headers)
    with request.urlopen(req) as resp:
    return resp.read().decode()
```

を使用した Lambda 関数の作成 AWS マネジメントコンソール

次の手順では、SMART on FHIR 対応データストアで FHIR REST API リクエストを処理するときに HealthLake が引き受けるサービスロールが既に作成されていることを前提としています。サービスロールを作成していない場合でも、Lambda 関数を作成できます。Lambda 関数が機能する前に、サービスロールの ARN を追加する必要があります。サービスロールの作成と Lambda 関数での指定の詳細については、「」を参照してください。 JWT のデコードに使用される AWS Lambda 関数で使用する HealthLake サービスロールの作成

Lambda 関数を作成するには (AWS マネジメントコンソール)

1. Lambda コンソールの [関数] ページを開きます。
2. [関数の作成] を選択してください。
3. [一から作成] を選択します。

4. 基本情報 に関数名を入力します。Runtime で、Python ベースのランタイムを選択します。
5. [Execution role] (実行ロール) で [Create a new role with basic Lambda permissions] (基本的な Lambda アクセス権限で新しいロールを作成) を選択します。

   Lambda が、Amazon CloudWatch にログをアップロードする関数許可を付与する実行ロールを作成します。Lambda 関数は、関数を呼び出すときに実行ロールを引き受け、実行ロールを使用して AWS SDK の認証情報を作成します。
6. Code タブを選択し、サンプル Lambda 関数を追加します。

   Lambda 関数が使用するサービスロールをまだ作成していない場合は、サンプルの Lambda 関数が機能する前に作成する必要があります。Lambda 関数のサービスロールの作成の詳細については、「」を参照してくださいJWT のデコードに使用される AWS Lambda 関数で使用する HealthLake サービスロールの作成。

```
import base64
import boto3
import logging
import json
import os
from urllib import request, parse

logger = logging.getLogger()
logger.setLevel(logging.INFO)

## Uses Secrets manager to gain access to the access key ID and secret access key
 for the authorization server
client = boto3.client('secretsmanager', region_name="region-of-datastore")
response = client.get_secret_value(SecretId='name-specified-by-customer-in-
secretsmanager')
secret = json.loads(response['SecretString'])
client_id = secret['client_id']
client_secret = secret['client_secret']


unencoded_auth = f'{client_id}:{client_secret}'
headers = {
  'Authorization': f'Basic {base64.b64encode(unencoded_auth.encode()).decode()}',
  'Content-Type': 'application/x-www-form-urlencoded'
}
```

```
auth_endpoint = os.environ['auth-server-base-url'] # Base URL of the Authorization
 server
user_role_arn = os.environ['iam-role-arn'] # The IAM role client application will
 use to complete the HTTP request on the datastore

def lambda_handler(event, context):
    if 'datastoreEndpoint' not in event or 'operationName' not in event or
 'bearerToken' not in event:
    return {}

    datastore_endpoint = event['datastoreEndpoint']
    operation_name = event['operationName']
    bearer_token = event['bearerToken']
    logger.info('Datastore Endpoint [{}], Operation Name:
 [{}]'.format(datastore_endpoint, operation_name))

    ## To validate the token
    auth_response = auth_with_provider(bearer_token)
    logger.info('Auth response: [{}]'.format(auth_response))
    auth_payload = json.loads(auth_response)
    ## Required parameters needed to be sent to the datastore endpoint for the HTTP
 request to go through
    auth_payload["isAuthorized"] = bool(auth_payload["active"])
    auth_payload["nbf"] = auth_payload["iat"]
    return {"authPayload": auth_payload, "iamRoleARN": user_role_arn}

## Access the server
def auth_with_provider(token):
    data = {'token': token, 'token_type_hint': 'access_token'}
    req = request.Request(url=auth_endpoint + '/v1/introspect',
 data=parse.urlencode(data).encode(), headers=headers)
    with request.urlopen(req) as resp:
    return resp.read().decode()
```

Lambda 関数の実行ロールの変更

Lambda 関数を作成したら、Secrets Manager を呼び出すために必要なアクセス許可を含めるように実行ロールを更新する必要があります。Secrets Manager では、作成する各シークレットに ARN があります。最小権限を適用するには、実行ロールが Lambda 関数の実行に必要なリソースにのみアクセスできる必要があります。

Lambda 関数の実行ロールを変更するには、IAM コンソールで検索するか、Lambda コンソールで設定を選択します。Lambda 関数の実行ロールの管理の詳細については、「」を参照してくださいLambda 実行ロール。

Exampleへのアクセスを許可する Lambda 関数実行ロール GetSecretValue

IAM アクションGetSecretValueを実行ロールに追加すると、サンプルの Lambda 関数が機能するために必要なアクセス許可が付与されます。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:secret-name-DKodTA"
        }
    ]
}
```

この時点で、FHIR 対応データストアで SMART に送信される FHIR REST リクエストの一部として提供されるアクセストークンを検証するために使用できる Lambda 関数を作成しました。

JWT のデコードに使用される AWS Lambda 関数で使用する HealthLake サービスロールの作成

ペルソナ: IAM 管理者

IAM ポリシーを追加または削除し、新しい IAM ID を作成できるユーザー。

サービスロール

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

JSON ウェブトークン (JWT) がデコードされると、認可 Lambda は IAM ロール ARN も返す必要があります。このロールには、REST API リクエストを実行するために必要なアクセス許可が必要です。そうしないと、アクセス許可が不十分であるために失敗します。

IAM を使用してカスタムポリシーを設定する場合は、必要な最小限のアクセス許可を付与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「最小特権のアクセス許可を適用する」を参照してください。

認可 Lambda 関数で指定する HealthLake サービスロールを作成するには、2 つのステップが必要です。

- まず、IAM ポリシーを作成する必要があります。ポリシーは、認可サーバーでスコープを指定した FHIR リソースへのアクセスを指定する必要があります。
- 次に、サービスロールを作成する必要があります。ロールを作成するときは、信頼関係を指定し、ステップ 1 で作成したポリシーをアタッチします。信頼関係はHealthLake をサービスプリンシパルとして指定します。このステップでは、HealthLake データストア ARN と AWS アカウント ID を指定する必要があります。

新しい IAM ポリシーの作成

認可サーバーで定義するスコープによって、認証されたユーザーが HealthLake データストアでアクセスできる FHIR リソースが決まります。

作成した IAM ポリシーは、定義したスコープに合わせて調整できます。

IAM ポリシーステートメントの `Action`要素で以下のアクションを定義できます。テーブル`Action`内の ごとに、 を定義できます`Resource types`。HealthLake では、データストアは、IAM アクセス許可ポリシーステートメントの `Resource`要素で定義できるサポートされている唯一のリソースタイプです。

個々の FHIR リソースは、IAM アクセス許可ポリシーの 要素として定義できるリソースではありません。

HealthLake によって定義されたアクション

アクション	説明	アクセスレベル	リソースタイプ (必須)
CreateResource	リソースの作成にアクセス許可を付与します	書き込み	データストア ARN: arn:aws:healthlake:**your-region**:**111122223333**:datastore/fhir/**your-datastore-id**
DeleteResource	リソースを削除する許可を付与	書き込み	データストア ARN: arn:aws:healthlake:**your-region**:**111122223333**:datastore/fhir/**your-datastore-id**
ReadResource	リソースを読み取るアクセス許可を付与します	読み取り	データストア ARN: arn:aws:healthlake:**your-region**:**111122223333**:datastore/fhir/**your-datastore-id**
SearchWithGet	GET メソッドでリソースを検索する許可を付与	読み取り	データストア ARN: arn:aws:healthlake:**your-region**:**111122223333**:datastore/fhir/**your-datastore-id**
SearchWithPost	POST メソッドでリソースを検索する許可を付与	読み取り	データストア ARN: arn:aws:healthlake:**your-region**:**111122223333**:datastore/fhir/**your-datastore-id**
StartFHIRExportJobWithPost	GET で FHIR エクスポートジョブを開始する許可を付与	書き	データストア ARN: arn:aws:healthlake:**your-region**:**111122223333**:datastore/fhir/**your-datastore-id**

アクション	説明	アクセスレベル	リソースタイプ (必須)
		込み	
UpdateResource	リソースを更新する許可を付与	書き込み	データストア ARN: arn:aws:healthlake:**your-region**:**111122223333**:datastore/fhir/**your-datastore-id**

開始するには、 を使用できますAmazonHealthLakeFullAccess。このポリシーは、データストアで見つかったすべての FHIR リソースの読み取り、書き込み、検索、エクスポートを許可します。データストアに読み取り専用アクセス許可を付与するには、 を使用しますAmazonHealthLakeReadOnlyAccess。

、、または IAM SDK を使用したカスタムポリシーの作成の詳細については AWS マネジメントコンソール AWS CLI、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。 SDKs

HealthLake のサービスロールの作成 (IAM コンソール)

この手順を使用して、サービスロールを作成します。サービスを作成するときは、IAM ポリシーも指定する必要があります。

HealthLake のサービスロールを作成するには (IAM コンソール)

1. にサインイン AWS マネジメントコンソール し、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。
2. IAM コンソールのナビゲーションペインで [ロール] を選択します。
3. 続いて、[ロールの作成] を選択します。
4. 信頼エンティティの選択ページで、カスタム信頼ポリシーを選択します。

5. 次に、カスタム信頼ポリシーで、次のようにサンプルポリシーを更新します。をアカウント番号**your-account-id**に置き換え、インポートジョブまたはエクスポートジョブで使用するデータストアの ARN を追加します。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Principal": {
                "Service": "healthlake.amazonaws.com"
            },
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:healthlake:us-
east-1:123456789012:datastore/fhir/your-datastore-id"
                }
            }
        }
    ]
}
```

6. その後、[Next] を選択します。
7. アクセス許可の追加ページで、HealthLake サービスが引き受けるポリシーを選択します。ポリシーを検索するには、アクセス許可ポリシーで検索します。
8. 次に、ポリシーのアタッチを選択します。
9. 次に、名前、レビュー、作成のページで、ロール名に名前を入力します。
10. (オプション) 説明 に、ロールの簡単な説明を追加します。
11. 可能な場合は、このロールの目的を識別するのに役立つロール名またはロール名サフィックスを入力します。ロール名は AWS アカウントアカウント内で一意である必要があります。大文字と小文字は区別されません。例えば、**PRODROLE** と **prodrole** というロール名を両方作成することはできません。多くのエンティティによりロールが参照されるため、作成後にロール名を変更することはできません。

12. ロールの詳細を確認し、ロールの作成を選択します。

サンプル Lambda 関数でロール ARN を指定する方法については、「」を参照してくださいAWS Lambda 関数の作成。

Lambda 実行ロール

Lambda 関数の実行ロールは、 AWS サービスとリソースへのアクセス許可を関数に付与する IAM ロールです。このページでは、Lambda 関数の実行ロールを作成、表示、および管理する方法について説明します。

デフォルトでは、 を使用して新しい Lambda 関数を作成すると、Lambda は最小限のアクセス許可で実行ロールを作成します AWS マネジメントコンソール。実行ロールで付与されたアクセス許可を管理するには、Lambda デベロッパーガイドの「IAM コンソールでの実行ロールの作成」を参照してください。

このトピックで提供されるサンプル Lambda 関数は、Secrets Manager を使用して認可サーバーの認証情報を隠します。

作成した IAM ロールと同様に、最小特権のベストプラクティスに従うことが重要です。開発フレーズ中に、必要以上のアクセス許可を付与することがあります。ベストプラクティスとしては、本番環境に関数を公開する前に、ポリシーを調整して必要なアクセス許可のみを含めるようにします。詳細については、IAM ユーザーガイドの「最小権限の適用」を参照してください。

HealthLake に Lambda 関数のトリガーを許可する

HealthLake がユーザーに代わって Lambda 関数を呼び出すには、以下を実行する必要があります。

- HealthLake が`CreateFHIRDatastore`リクエストで呼び出す Lambda 関数の ARN と`IdpLambdaArn`等しく設定する必要があります。
- HealthLake がユーザーに代わって Lambda 関数を呼び出すことを許可するリソースベースのポリシーが必要です。

HealthLake が SMART on FHIR 対応データストアで FHIR REST API リクエストを受信すると、ユーザーに代わってデータストアの作成時に指定された Lambda 関数を呼び出すアクセス許可が必要です。HealthLake アクセスを許可するには、リソースベースのポリシーを使用します。Lambda 関数のリソースベースのポリシーの作成の詳細については、「 AWS Lambda デベロッパーガイド」の「Lambda 関数を呼び出すことを AWS に許可する」を参照してください。

Lambda 関数の同時実行のプロビジョニング

> **Important**
>
> HealthLake では、Lambda 関数の最大実行時間が 1 秒 (1000 ミリ秒) 未満である必要があります。
> Lambda 関数が実行時間の制限を超えると、TimeOut 例外が発生します。

この例外が発生しないように、プロビジョニングされた同時実行を設定することをお勧めします。呼び出しの増加前にプロビジョニングされた同時実行数を割り当てることにより、すべてのリクエストが、短いレイテンシーで、初期化されたインスタンスによって処理されるようにできます。プロビジョニングされた同時実行の設定の詳細については、Lambda デベロッパーガイドの「プロビジョニングされた同時実行の設定」を参照してください。

現在、Lambda 関数の平均実行時間を確認するには、Lambda コンソールの Lambda 関数のモニタリングページを使用します。デフォルトでは、Lambda コンソールには、関数コードがイベントの処理に費やした平均、最小、最大時間を示す期間グラフが表示されます。Lambda 関数のモニタリングの詳細については、Lambda デベロッパーガイドの「Lambda コンソールでの関数のモニタリング」を参照してください。

Lambda 関数の同時実行を既にプロビジョニングしていて、それをモニタリングする場合は、Lambda デベロッパーガイドの「同時実行のモニタリング」を参照してください。

SMART on FHIR 対応 HealthLake データストアでのきめ細かな認可の使用

スコープだけでは、リクエスタがデータストア内のどのデータにアクセスすることを許可されているかについて必要な具体性は得られません。きめ細かな認可を使用すると、FHIR 対応の HealthLake データストアで SMART へのアクセスを許可するときに、より高いレベルの特異度を実現できます。きめ細かな認可を使用するには、`CreateFHIRDatastore`リクエストの `IdentityProviderConfiguration`パラメータ`True`で を に`FineGrainedAuthorizationEnabled`等しく設定します。

きめ細かな認可を有効にした場合、認可サーバーはアクセストークン`id_token`とともに で`fhirUser`スコープを返します。これにより、クライアントアプリケーションでユーザーに関する情報を取得できます。クライアントアプリケーションは、`fhirUser`クレームを現在のユーザーを表す FHIR リソースの URI として扱う必要があります。これは、`Patient`、`Practitioner`、または `RelatedPerson` です。認可サーバーのレスポンスには、ユーザーがアクセスできるデータを定義

するuser/スコープも含まれています。これは、FHIR リソース固有のスコープに関連するスコープに定義された構文を使用します。

```
user/(fhir-resource | '*').('read' | 'write' | '*')
```

以下は、きめ細かな認可を使用してデータアクセス関連の FHIR リソースタイプをさらに指定する方法の例です。

- fhirUser が の場合Practitioner、きめ細かな認可によって、ユーザーがアクセスできる患者のコレクションが決まります。へのアクセスfhirUserは、患者が一般臨床医fhirUserとして を参照している患者に対してのみ許可されます。

  ```
  Patient.generalPractitioner : [{Reference(Practitioner)}]
  ```

- fhirUser が Patientまたは RelatedPersonで、リクエストで参照される患者が と異なる場合fhirUser、きめ細かな承認により、リクエストされた患者の fhirUser へのアクセスが決まります。リクエストされたPatientリソースで関係が指定されている場合、アクセスが許可されます。

  ```
  Patient.link.other : {Reference(Patient|RelatedPerson)}
  ```

FHIR 検出ドキュメントの SMART の取得

SMART は、HealthLake データストアがサポートする認可エンドポイント URLsと機能をクライアントが学習できるようにする Discovery Document を定義します。この情報は、クライアントが認可リクエストを適切なエンドポイントに転送し、HealthLake データストアがサポートする認可リクエストを構築するのに役立ちます。

クライアントアプリケーションが HealthLake への FHIR REST リクエストを成功させるには、HealthLake データストアで定義されている認可要件を収集する必要があります。このリクエストを成功させるには、ベアラートークン (認可) は必要ありません。

HealthLake データストアの検出ドキュメントをリクエストするには

1. HealthLake regionと datastoreIdの値を収集します。詳細については、「データストアのプロパティの取得」を参照してください。

2. HealthLake regionと の収集された値を使用して、リクエストの URL を作成しますdatastoreId。URL のエンドポイント/.well-known/smart-configurationに追加します。次の例の URL パス全体を表示するには、コピーボタンをスクロールします。

```
https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/.well-known/smart-configuration
```

3. AWS 署名バージョン 4 の署名プロトコルGETを使用してリクエストを送信します。例全体を表示するには、コピーボタンをスクロールします。

   curl

```
curl --request GET \
  'https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/.well-known/smart-configuration \
  --aws-sigv4 'aws:amz:region:healthlake' \
  --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
  --header "x-amz-security-token:$AWS_SESSION_TOKEN" \
  --header 'Accept: application/json'
```

   HealthLake データストアの検出ドキュメントは JSON BLOB として を返します。ここでは、authorization_endpointと token_endpoint、およびデータストアの仕様と定義された機能を確認できます。

```
{
    "authorization_endpoint": "https://oidc.example.com/authorize",
    "token_endpoint": "https://oidc.example.com/oauth/token",
    "capabilities": [
        "launch-ehr",
        "client-public"
    ]
}
```

   クライアントアプリケーションを起動するには、 authorization_endpointと の両方token_endpointが必要です。

   - 認可エンドポイント — クライアントアプリケーションまたはユーザーを認可するために必要な URL。

- トークンエンドポイント — クライアントアプリケーションが通信に使用する認可サーバーのエンドポイント。

SMART 対応 HealthLake データストアでの FHIR REST API リクエストの実行

SMART on FHIR 対応の HealthLake データストアで FHIR REST API リクエストを行うことができます。次の例は、認可ヘッダーに JWT を含むクライアントアプリケーションからのリクエストと、Lambda がレスポンスをデコードする方法を示しています。クライアントアプリケーションリクエストが承認され、認証されたら、認可サーバーからベアラートークンを受信する必要があります。SMART on FHIR 対応 HealthLake データストアで FHIR REST API リクエストを送信するときは、認可ヘッダーのベアラートークンを使用します。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/[ID]
Authorization: Bearer auth-server-provided-bearer-token
```

ベアラートークンが認可ヘッダーで見つかり、IAM ID AWS が検出されなかったため、HealthLake は SMART on FHIR 対応 HealthLake データストアの作成時に指定された Lambda 関数を呼び出します。トークンが Lambda 関数によって正常にデコードされると、次のレスポンス例が HealthLake に送信されます。

```
{
  "authPayload": {
    "iss": "https://authorization-server-endpoint/oauth2/token", # The issuer
 identifier of the authorization server
    "aud": "https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/", #
 Required, data store endpoint
    "iat": 1677115637,  # Identifies the time at which the token was issued
    "nbf": 1677115637,  # Required, the earliest time the JWT would be valid
    "exp": 1997877061,  # Required, the time at which the JWT is no longer valid
    "isAuthorized": "true",  # Required, boolean indicating the request has been
 authorized
    "uid": "100101",  # Unique identifier returned by the auth server
    "scope": "system/*.*" # Required, the scope of the request
  },
  "iamRoleARN": "iam-role-arn" #Required, IAM role to complete the request
}
```

の FHIR R4 サポート AWS HealthLake

AWS HealthLake は、ヘルスデータ交換の FHIR R4 仕様をサポートしています。以下のセクションでは、HealthLake が FHIR R4 仕様を利用して、FHIR R4 RESTful APIs を使用して HealthLake データストア内の FHIR リソースを管理および検索する方法に関するサポート情報を提供します。

トピック

の FHIR R4 機能ステートメント AWS HealthLake

アクティブな HealthLake データストアの FHIR 関連の機能 (動作) を確認するには、その機能ステートメントを取得する必要があります。機能ステートメントは、実際のサーバー機能のステートメント、または必要または必要なサーバー実装のステートメントとして使用されます。FHIR `capabilities`インタラクションは、HealthLake データストア機能と、それがサポートする FHIR 仕様のどの部分に関する情報を取得します。HealthLake は、FHIR R4 リソースに従って FHIR `StructureDefinition`リソースタイプを検証します。

HealthLake データストアの Capability Statement を取得するには

1. HealthLake `region`と `datastoreId`の値を収集します。詳細については、「データストアのプロパティの取得」を参照してください。
2. HealthLake `region`と の収集された値を使用して、リクエストの URL を作成しますdatastoreId。URL に FHIR `metadata`要素も含めます。次の例の URL パス全体を表示するには、コピーボタンにスクロールします。

   ```
   https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/metadata
   ```

3. リクエストを送信します。FHIR `capabilities`インタラクションは、AWS 署名バージョン 4 の署名プロトコルを持つGETリクエストを使用します。次の のcurl例では、 で指定された HealthLake データストアの Capability Statement を取得しますdatastoreId。例全体を表示するには、コピーボタンをスクロールします。

curl

```
curl --request GET \
  'https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/metadata \
  --aws-sigv4 'aws:amz:region:healthlake' \
  --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
  --header "x-amz-security-token:$AWS_SESSION_TOKEN" \
  --header 'Accept: application/json'
```

HealthLake データストアの 200 HTTP レスポンスコードと機能ステートメントを受け取ります。詳細については、FHIR R4 ドキュメントCapabilityStatementの「」を参照してください。

HealthLake の FHIR プロファイル検証

AWS HealthLake は、基本 FHIR R4 仕様をサポートしています。基本 FHIR R4 仕様には FHIR プロファイルが含まれています。プロファイルは、FHIR リソースタイプで使用され、基本リソースタイプの制約や拡張を使用してより具体的なリソースタイプ定義を定義します。たとえば、FHIR プロファイルは、拡張機能や値セットなどの必須フィールドを識別できます。リソースは複数のプロファイルをサポートできます。すべての HealthLake データストアは、FHIR プロファイルの使用をサポートしています。

> **Note**
>
> HealthLake データストアにデータを追加するときは、FHIR プロファイルを追加する必要はありません。リソースが追加または更新されたときに FHIR プロファイルが指定されていない場合、リソースは基本 FHIR R4 スキーマに対してのみ検証されます。
> FHIR リソースが準拠する FHIR プロファイルは、HealthLake にインポートされる前にリソースに含まれます。したがって、FHIR プロファイルはインポート中に HealthLake によって検証されます。

FHIR プロファイルは実装ガイドで指定されています。FHIR 実装ガイド (IG) は、特定の目的で FHIR 標準を使用する方法を説明する一連の手順です。HealthLake は、以下の実装ガイドで定義されている FHIR プロファイルを検証します。

でサポートされている FHIR プロファイル AWS HealthLake

名前	バージョン	実装ガイド	機能	米国東部(オハイオ)	米国東部(バージニア北部)	米国西部(オレゴン)	アジアパシフィック(ムンバイ)	アジアパシフィック(シドニー	カナダ(中部)	欧州(アイルランド)	欧州(ロンドン)
米国コア	3.1	http://hl7.org/fhir/us/core/STU3.1.1/	デフォルト	X	X	X	X	X	X	X	X
米国コア	4.0	https://hl7.org/fhir/us/core/STU4/index.html	サポート	X	X	X	X	X	X	X	X
米国コア	5.0	https://hl7.org/fhir/us/core/STU5.0.1/index.html	サポート	X	X	X	X	X	X	X	X
米国コア	6.1	https://hl7.org/fhir/us/core/STU6.1/index.html	サポート	X	X	X	X	X	X	X	X
米国コア	7.0	https://hl7.org/fhir/us/core/STU7/	サポート	X	X	X	X	X	X	X	X
英国コア	2.0	https://simplifier.net/guide/uk-core-implementation-guide-stu2/Home/ProfilesandExtensions/ProfilesIndex?version=2.0.1	サポート	X	X	X	X			X	X

名前	バージョン	実装ガイド	機能	米国東部(オハイオ)	米国東部(バージニア北部)	米国西部(オレゴン)	アジアパシフィック(ムンバイ)	アジアパシフィック(シドニー	カナダ(中部)	欧州(アイルランド)	欧州(ロンドン)
CARIN ブルーボタン	1.1	http://hl7.org/fhir/us/carin-bb/STU1.1/	デフォルト	X	X	X	X	X	X	X	X
CARIN ブルーボタン	1.0.0、	https://hl7.org/fhir/us/carin-bb/history.html	サポート	X	X	X	X	X	X	X	X
Da Vinci Payer Data Exchange	1.0	https://hl7.org/fhir/us/davinci-pdex/	デフォルト	X	X	X	X	X	X	X	X
Da Vinci Payer Data Exchange	2.0.0	https://hl7.org/fhir/us/davinci-pdex/history.html	サポート	X	X	X	X	X	X	X	X
Da Vinci Health Record Exchange (HRex)	0.2	https://hl7.org/fhir/us/davinci-hrex/2020Sep/	デフォルト	X	X	X	X	X	X	X	X
DaVinci PDEX プランネット	1.1	https://hl7.org/fhir/us/davinci-pdex-plan-net/STU1.1/	デフォルト	X	X	X	X	X	X	X	X

名前	バージョン	実装ガイド	機能	米国東部(オハイオ)	米国東部(バージニア北部)	米国西部(オレゴン)	アジアパシフィック(ムンバイ)	アジアパシフィック(シドニー	カナダ(中部)	欧州(アイルランド)	欧州(ロンドン)
DaVinci PDEX プランネット	1.0	https://hl7.org/fhir/us/davinci-pdex-plan-net/STU1/	サポート	X	X	X	X	X	X	X	X
DaVinci Payer Data Exchange (PDex) 米国薬物フォーミュラリー	1.1	https://hl7.org/fhir/us/davinci-drug-formulary/STU1.1/	デフォルト	X	X	X	X	X	X	X	X
DaVinci Payer Data Exchange (PDex) 米国薬物フォーミュラリー	1.0.1、	https://hl7.org/fhir/us/davinci-drug-formulary/history.html	サポート	X	X	X	X	X	X	X	X
Da Vinci Clinical Data Exchange (CDex)	2.1	https://build.fhir.org/ig/HL7/davinci-ecdx/index.html	デフォルト	X	X	X	X	X	X	X	X

名前	バージョン	実装ガイド	機能	米国東部(オハイオ)	米国東部(バージニア北部)	米国西部(オレゴン)	アジアパシフィック(ムンバイ)	アジアパシフィック(シドニー	カナダ(中部)	欧州(アイルランド)	欧州(ロンドン)
Da Vinci 事前認可サポート (PAS) FHIR IG	2.1	https://hl7.org/fhir/us/davinci-pas/	デフォルト	X	X	X	X	X	X	X	X
NCQA HEDIS® 実装ガイド	0.3	https://www.ncqa.org/resources/hedis-ig-resource-page/	デフォルト	X	X	X	X			X	X
国際患者概要 (IPS)	2.0 ブート	https://hl7.org/fhir/uv/ips/2024Sep/	デフォルト	X	X	X	X	X	X	X	X
品質測定	5.0	https://registry.fhir.org/package/hl7.fhir.us.cqfmeasures%7C5.0.0	デフォルト	X	X	X	X			X	X
ゲノミクスレポート	3.0	https://build.fhir.org/ig/HL7/genomics-reporting/index.html	デフォルト	X	X	X	X			X	X

名前	バージョン	実装ガイド	機能	米国東部(オハイオ)	米国東部(バージニア北部)	米国西部(オレゴン)	アジアパシフィック(ムンバイ)	アジアパシフィック(シドニー	カナダ(中部)	欧州(アイルランド)	欧州(ロンドン)
米国保健局の Ayushman Bharat Digital Mission (ABDM)	2.0	https://www.nrces.in/ndhm/fhir/r4/index.html	デフォルト	X	X	X	X			X	X
CA Core+	1.1	https://simplifier.net/ca-core	サポート						X		
CA:eReC Pan-Canadian eReferral-eConsult	1.1	https://simplifier.net/CA-eReC/~introduction	サポート						X		
患者概要カナダ版 - (PS-CA)	2.1	https://simplifier.net/PS-CA-R1/~introduction	サポート						X		

名前	バージョン	実装ガイド	機能	米国東部(オハイオ)	米国東部(バージニア北部)	米国西部(オレゴン)	アジアパシフィック(ムンバイ)	アジアパシフィック(シドニー	カナダ(中部)	欧州(アイルランド)	欧州(ロンドン)
オンタリオデジタルヘルスドラッグリポジトリ	4.0	https://simplifier.net/ca-on-dhdr-r4/~introduction	サポート						X		
AU コア	1.0	https://hl7.org.au/fhir/core/	サポート					X			
マゼンタス練習管理	1.2	https://fhir-versions.dev.geniesolutions.io/1.2.16/downloads.html	サポート					X			

リソースで指定された FHIR プロファイルの検証

FHIR プロファイルを検証するには、実装ガイドで指定されたプロファイル URL を使用して、個々のリソースの `profile`要素に追加します。

FHIR プロファイルは、データストアに新しいリソースを追加するときに検証されます。新しいリソースを追加するには、 `StartFHIRImportJob` API オペレーションを使用するか、リクエストを実行して新しいリソース`POST`を追加するか、 `PUT` を実行して既存のリソースを更新します。

Example– リソースで参照されている FHIR プロファイルを確認するには

プロファイル URL は、"meta" : "profile"キーと値のペアの profile要素に追加されます。このリソースはわかりやすくするために切り捨てられました。

```
{
    "resourceType": "Patient",
    "id": "abcd1234efgh5678hijk9012",
    "meta": {
        "lastUpdated": "2023-05-30T00:48:07.8443764-07:00",
        "profile": [
            "http://hl7.org/fhir/us/core/StructureDefinition/us-core-patient"
        ]
    }
}
```

Example– サポートされているデフォルト以外の FHIR プロファイルを参照する方法

サポートされているデフォルト以外のプロファイルに対して検証するには、バージョニングされたプロファイル URL を meta.profile要素に追加します。バージョニングされた URL には、ベースプロファイル URL の後にパイプ文字 (|) とバージョン番号が続きます。このサンプルリソースはわかりやすくするために切り捨てられました。

```
{
    "resourceType": "ExplanationOfBenefit",
    "id": "sample-EOB",
    "meta": {
        "lastUpdated": "2024-02-02T05:56:09.4+00:00",
        "profile": [
            "http://hl7.org/fhir/us/carin-bb/StructureDefinition/C4BB-
ExplanationOfBenefit-Pharmacy|1.0.0"
        ]
    }
}
```

バージョニングされた URL とベースプロファイル URL の両方を含めることもできます。どちらの形式も有効です。

```
{
    "resourceType": "ExplanationOfBenefit",
    "id": "sample-EOB",
```

```
    "meta": {
        "lastUpdated": "2024-02-02T05:56:09.4+00:00",
        "profile": [
            "http://hl7.org/fhir/us/carin-bb/StructureDefinition/C4BB-
ExplanationOfBenefit-Pharmacy|1.0.0",
            "http://hl7.org/fhir/us/carin-bb/StructureDefinition/C4BB-
ExplanationOfBenefit-Pharmacy"
        ]
    }
}
```

HealthLake でサポートされている FHIR R4 リソースタイプ

次の表に、 でサポートされている FHIR R4 リソースタイプを示します AWS HealthLake。詳細については、FHIR R4 ドキュメントの「リソースインデックス」を参照してください。

HealthLake でサポートされている FHIR R4 リソースタイプ

アカウント	DetectedIssue	Invoice	プラクティショナー
ActivityDefinition	デバイス	Library	PractitionerRole
AdverseEvent	DeviceDefinition	リンク	手順
AllergyIntolerance	DeviceMetric	リスト	プロベナンス
予約	DeviceUseStatement	ロケーション	アンケート
AppointmentResponse	DeviceRequest	メジャー	QuestionnaireResponse
AuditEvent - 注を参照	DiagnosticReport	MeasureReport	RelatedPerson
バイナリ	DocumentManifest	メディア	RequestGroup
BodyStructure	DocumentReference	薬剤	ResearchStudy
バンドル - 注を参照	EffectEvidenceSynthesis	MedicationAdministration	ResearchSubject
CapabilityStatement	エンカウンター	MedicationDispense	RiskAssessment

CarePlan	Endpoint	MedicationKnowledge	RiskEvidenceSynthesis
CareTeam	EpisodeOfCare	MedicationRequest	スケジュール
ChargeItem	EnrollmentRequest	MedicationStatement	ServiceRequest
ChargeItemDefinition	EnrollmentResponse	MessageHeader	スロット
Claim	ExplanationOfBenefit	MolecularSequence	標本
ClaimResponse	FamilyMemberHistory	NutritionOrder	StructureDefinition
コミュニケーション	フラグ	監視結果	StructureMap
CommunicationRequest	目標	OperationOutcome - 注を参照	Substance
コンポジション	Group	組織	SupplyDelivery
ConceptMap	GuidanceResponse	OrganizationAffiliation	SupplyRequest
Condition	HealthcareService	パラメータ - 注を参照	タスク
同意	ImagingStudy	患者	ValueSet
Contract	イミュナイゼーション	PaymentNotice	VisionPrescription
カバレッジ	ImmunizationEvaluation	PaymentReconciliation	VerificationResult - 注を参照
CoverageEligibilityRequest	ImmunizationRecommendation	個人	
CoverageEligibilityResponse	InsurancePlan	PlanDefinition	

⚠ FHIR 仕様と HealthLake

- FHIR OperationOutcomeおよび Parametersリソースタイプを使用して GETまたは POSTリクエストを行うことはできません。
- AuditEvent — AuditEvent リソースを作成または読み取ることができますが、更新または削除することはできません。
- Bundle — HealthLake が Bundle リクエストを管理する方法は複数あります。詳細については、FHIR リソースのバンドルを参照してください。
- VerificationResult — このリソースタイプは、2023 年 12 月 9 日以降に作成されたデータストアでのみサポートされます。

HealthLake の FHIR R4 検索パラメータ

FHIR searchインタラクションを使用して、一部のフィルター条件に基づいて HealthLake データストア内の一連の FHIR リソースを検索します。search インタラクションは、 GETまたは POSTリクエストを使用して実行できます。個人を特定できる情報 (PII) または保護された医療情報 (PHI) を含む検索では、PII と PHI がPOSTリクエスト本文の一部として追加され、転送中に暗号化されるため、リクエストを使用することをお勧めします。

Note

この章で説明する FHIR searchインタラクションは、医療データ交換のための HL7 FHIR R4 標準に準拠して構築されています。HL7 FHIR サービスの表現であるため、 AWS CLI および AWS SDKs では提供されません。詳細については、FHIR R4 RESTful API ドキュメントsearchの「」を参照してください。 R4 RESTful
Amazon Athena を使用して SQL で HealthLake データストアをクエリすることもできます。詳細については、「統合」を参照してください。

HealthLake は、以下の FHIR R4 検索パラメータのサブセットをサポートしています。詳細については、「HealthLake の FHIR R4 検索パラメータ」を参照してください。

サポートされている検索パラメータタイプ

次の表は、HealthLake でサポートされている検索パラメータタイプを示しています。

サポートされている検索パラメータタイプ

検索パラメータ	説明
_id	リソース ID (完全な URL ではない)
_lastUpdated	最終更新日。サーバーには境界の精度に関する裁量があります。
_tag	リソースタグで検索します。
プロファイル	プロファイルでタグ付けされたすべてのリソースを検索します。
セキュリティ	このリソースに適用されたセキュリティラベルを検索します。
_ソース	リソースのソースを検索します。
テキスト	リソースの説明を検索します。
createdAt	カスタム拡張機能 createdAt で検索します。

> **Note**
>
> 次の検索パラメータは、2023 年 12 月 9 日以降に作成されたデータストアでのみサポートされています: _security、_source、_text、createdAt。

次の表は、特定のリソースタイプの指定されたデータ型に基づいてクエリ文字列を変更する方法の例を示しています。わかりやすくするために、例列の特殊文字はエンコードされていません。クエリを成功させるには、クエリ文字列が適切にエンコードされていることを確認します。

検索パラメータの例

検索パラメータタイプ	Details	例

Number	指定されたリソース内の数値を検索します。有効数字が観察されます。有効桁数は、先	`[parameter]=100`
		
		`[parameter]=1e2`

検索パラメータタイプ	Details	例
	頭の 0 を除く検索パラメータ値で に固有です。比較プレフィックスは許可されます。	`[parameter]=lt100`

日付/DateTime	特定の日付または時刻を検索します。想定される形式は `yyyy-mm-ddThh:mm:ss[Z|(+|-)hh:mm]` ですが、異なる場合があります。
	
	、`date`、`dateTime`、`instant` よび のデータ型を受け入れますTiming。検索でこれらのデータ型を使用する詳細については、FHIR R4 RESTful API ドキュメントの「日付」を参照してください。
	
	比較プレフィックスは許可されます。
		`[parameter]=eq2013-01-14`
		
		`[parameter]=gt2013-01-14T10:00`
		
		`[parameter]=ne2013-01-14`

String	大文字と小文字を区別して一連の文字を検索します。
	
	`HumanName` と の両方Addressのタイプをサポートします。詳細については、FHIR R4 ドキュメントの HumanName データ型エントリとAddressデータ型エントリを参照してください。
	
	詳細検索は`:text`、修飾子を使用してサポートされています。
		`[base]/Patient?given=eve`
		
		`[base]/Patient?given:contains=eve`

検索パラメータタイプ	Details	例
トークン	多くの場合、医療コード値のペアと比較して、文字列に対するclose-to-exact一致を検索します。 大文字と小文字の区別は、クエリの作成時に使用するコードシステムにリンクされます。サブsumption ベースのクエリは、大文字と小文字の区別に関連する問題を減らすのに役立ちます。わかりやすくするために、 |はエンコードされていません。	`[parameter]=[system]|[code]` ：ここでは、コーディングシステム`[system]`を参照し、その特定のシステム内で見つかったコード値`[code]`を参照します。 `[parameter]=[code]` ：ここで、入力はコードまたはシステムのいずれかに一致します。 `[parameter]=|[code]` ：ここで、入力はコードと一致し、システムプロパティには識別子がありません。

複合	修飾子`$`と `,`オペレーションを使用して、単一のリソースタイプ内の複数のパラメータを検索します。
	
	比較プレフィックスは許可されます。
		`/Patient?language=FR,NL&language=EN`
		
		`Observation?component-code-value-quantity=http://loinc.org|8480-6$lt60`
		
		`[base]/Group?characteristic-value=gender$mixed`

検索パラメータタイプ	Details	例

数量	数値、システム、コードを値として検索します。番号は必須ですが、システムとコードはオプションです。数量データ型に基づきます。詳細については、FHIR R4 ドキュメントの「数量」を参照してください。
	
	次の想定構文を使用します。
	`[parameter]=[prefix][number]|[system]|[code]`
		`[base]/Observation?value-quantity=5.4|http://unitsofmeasure.org|mg`
		
		`[base]/Observation?value-quantity=5.4|http://unitsofmeasure.org|mg`
		
		`[base]/Observation?value-quantity=5.4|http://unitsofmeasure.org|mg`
		
		`[base]/Observation?value-quantity=le5.4|http://unitsofmeasure.org|mg`

リファレンス	他のリソースへの参照を検索します。	`[base]/Observation?subject=Patient/23`
		
		`test`

[URI]	特定のリソースを明確に識別する文字列を検索します。	`[base]/ValueSet?url=http://acme.org/fhir/ValueSet/123`
スペシャル	統合された医療 NLP 拡張機能に基づいて検索します。	

HealthLake でサポートされている高度な検索パラメータ

HealthLake は、次の高度な検索パラメータをサポートしています。

名前	説明	例	機能
_includ	検索リクエストで追加のリソースが返されるようにリクエストするために使用されます。ターゲットリソースインスタンスによって参照されるリソースを返します。	Encounter?_include=Encounter:subject	
_revinc de	検索リクエストで追加のリソースが返されるようにリクエストするために使用されます。プライマリリソースインスタンスを参照するリソースを返します。	Patient?_id= **patient-identifier** &_revinclude=Encounter:patient	
_summar	概要は、リソースのサブセットをリクエストするために使用できます。	Patient?_summary=text	次の概要パラメータがサポートされています: _summary=true 、_summary=false 、_summary=text 、_summary=data 。
_elemen	検索結果のリソースの一部として返される特定の要素のセットをリクエストします。	Patient?_elements=identifier,active,link	
_total	検索パラメータに一致するリソースの数を返します。	Patient?_total=accurate	_total=accurate 、 をサポートします_total=none 。
_sort	カンマ区切りリストを使用して、返された検索結果のソート順を示します。- プレフィックスは、カンマ区切りリストの任意のソートルール	Observation?_sort=status,-date	タイプが のフィールドによるソートをサポートしますNumber, String, Quantity, Token, URI, Reference 。によるソートDateは、2023 年 12 月 9 日

名前	説明	例	機能
	で、降順を示すために使用できます。		以降に作成されたデータストアでのみサポートされています。最大 5 つのソートルールをサポートします。
_count	検索バンドルのページごとに返されるリソースの数を制御します。	Patient?_count=100	最大ページサイズは 100 です。
chainin	参照されるリソースの要素を検索します。. は、連鎖検索を、参照されるリソース内の要素に送信します。	DiagnosticReport?subject:Patient.name=peter	
reverse chainin (_has)	リソースを参照するリソースの要素に基づいてリソースを検索します。	Patient?_has:Observation:patient:code=1234-5	

_include

検索クエリ_includeで を使用すると、追加の指定された FHIR リソースも返されます。を使用して_include、前方にリンクされたリソースを含めます。

Example– を使用して_include、せきの診断を受けた患者または患者のグループを検索するには

診断コードを指定するConditionリソースタイプを検索し、その診断subjectの も返すように_include指定します。Condition リソースタイプでは、患者リソースタイプまたはグループリソースタイプのいずれかsubjectを指します。

わかりやすくするために、この例の特殊文字はエンコードされていません。クエリを成功させるには、クエリ文字列が適切にエンコードされていることを確認します。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
Condition?code=49727002&_include=Condition:subject
```

`_include:iterate` 修飾子

修飾子を使用すると、2 `_include:iterate` つのレベルで参照されるリソースを再帰的に含めることができます。例えば、

```
GET /ServiceRequest?
identifier=025C0931195&_include=ServiceRequest:requester&_include:iterate=PractitionerRole:prac
```

は、ServiceRequest、関連付けられた PractitionerRole (リクエスタリファレンス経由) を返し、その PractitionerRole によって参照される Practitioner を再帰的に含めます。この修飾子は、HealthLake のすべてのリソースタイプで使用できます。

`_include=*` 修飾子

`_include=*` 修飾子は、検索結果によって直接参照されるすべてのリソースを自動的に含むワイルドカードです。例えば、

```
GET /ServiceRequest?specimen.accession=12345&_include=*
```

は、各参照パスを個別に指定することなく、一致する ServiceRequest を、それが参照するすべてのリソース (患者、プラクティショナー、医療関係者など) とともに返します。この修飾子は、HealthLake のすべてのリソースタイプで使用できます。

`_revinclude`

検索クエリ`_revinclude`で を使用すると、指定された追加の FHIR リソースも返されます。を使用して`_revinclude`、後方にリンクされたリソースを含めます。

Example– を使用して_revinclude、特定の患者にリンクされた関連する Encounter および Observation リソースタイプを含めるには

この検索を行うには、まず`_id`検索パラメータで識別子を指定`Patient`して個人を定義します。次に、 構造`Encounter:patient`と を使用して追加の FHIR リソースを指定します`Observation:patient`。

わかりやすくするために、この例の特殊文字はエンコードされていません。クエリを成功させるには、クエリ文字列が適切にエンコードされていることを確認します。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
```

```
Patient?_id=patient-
identifier&_revinclude=Encounter:patient&_revinclude=Observation:patient
```

_summary

検索クエリ_summaryで を使用すると、ユーザーは FHIR リソースのサブセットをリクエストできます。次のいずれかの値を含めることができます: true, text, data, false。その他の値は無効として扱われます。返されたリソースは meta.tag 'SUBSETTED' で とマークされ、リソースが不完全であることを示します。

- true: リソースの基本定義で「概要」としてマークされているサポートされているすべての要素を返します (複数可) 。
- text: 「text」、「id」、「meta」要素のみ、および最上位の必須要素のみを返します。
- data: 「text」要素を除くすべてのパートを返します。
- false: リソースのすべての部分を返します (複数可)

単一の検索リクエストでは、 を _includeまたは _revinclude 検索パラメータと組み合わせる_summary=textことはできません。

Example– データストア内の患者リソースの「テキスト」要素を取得します。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?
_summary=text
```

_elements

検索クエリ_elementsで を使用すると、特定の FHIR リソース要素をリクエストできます。返されたリソースは meta.tag 'SUBSETTED' で とマークされ、リソースが不完全であることを示します。

_elements パラメータは、リソースのルートレベルで定義された要素など、基本要素名のカンマ区切りリストで構成されます。リストされている要素のみが返されます。_elements パラメータ値に無効な要素が含まれている場合、サーバーはそれらを無視し、必須要素と有効な要素を返します。

_elements は、含まれるリソース (検索モードが である返されたリソースinclude) には適用されません。

単一の検索リクエストでは、 を検索_summaryパラメータと組み合わせる_elementsことはできません。

Example– HealthLake データストアで患者リソースの「識別子」、「アクティブ」、「リンク」要素を取得します。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?
_elements=identifier,active,link
```

_total

検索クエリ_totalで を使用すると、リクエストされた検索パラメータに一致するリソースの数が返されます。HealthLake は、検索レスポンスBundle.totalの で一致したリソース (検索モードが である返されたリソースmatch) の総数を返します。

_total は accurate、noneパラメータ値をサポートしています。 _total=estimateはサポートされていません。その他の値は無効として扱われます。 _totalは、含まれるリソース (検索モードが である返されたリソースinclude) には適用されません。

Example– データストア内の患者リソースの総数を取得します。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?
_total=accurate
```

_sort

検索クエリ_sortで を使用すると、結果が特定の順序で配置されます。結果は、ソートルールのカンマ区切りリストに基づいて優先順位が付けられます。ソートルールは有効な検索パラメータである必要があります。その他の値は無効として扱われます。

1 つの検索リクエストで、最大 5 つのソート検索パラメータを使用できます。オプションで、-プレフィックスを使用して降順を指定できます。デフォルトでは、サーバーは昇順でソートされます。

サポートされているソート検索パラメータタイプは、 ですNumber, String, Date, Quantity, Token, URI, Reference。検索パラメータがネストされた要素を参照している場合、この検索パラメータはソートではサポートされていません。たとえば、リソースタイプの「名前」を検索する患者 は HumanName データ型の Patient.name 要素を参照し、ネストされたと見なされます。したがって、患者リソースを「名前」でソートすることはサポートされていません。

Example– データストアで患者リソースを取得し、生年月日で昇順にソートします。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?
_sort=birthdate
```

_count

パラメータ_countは、1 ページに返されるリソースの数に関するサーバーへの指示として定義されます。

最大ページサイズは 100 です。100 を超える値は無効です。 _count=0はサポートされていません。

Example– 患者リソースを検索し、検索ページサイズを 25 に設定します。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?_count=25
```

Chaining and Reverse Chaining(_has)

FHIR での連鎖と逆連鎖は、相互接続されたデータを取得するためのより効率的でコンパクトな方法を提供し、複数の個別のクエリの必要性を減らし、デベロッパーやユーザーにとってデータの取得をより便利にします。

いずれかのレベルの再帰が 100 を超える結果を返した場合、HealthLake はデータストアが過負荷になり、複数のページ分割が発生するのを防ぐために 4xx を返します。

Example– 連鎖 - 患者名がペターである患者を参照するすべての DiagnosticReport を取得します。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/DiagnosticReport?
subject:Patient.name=peter
```

Example– リバースチェイニング - 患者リソースを取得します。ここで、患者リソースは少なくとも 1 つのオブザベーションによって参照され、オブザベーションのコードは 1234 で、オブザベーションは患者検索パラメータ内の患者リソースを参照します。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?
_has:Observation:patient:code=1234
```

サポートされている検索修飾子

検索修飾子は文字列ベースのフィールドで使用されます。HealthLake のすべての検索修飾子は、ブールベースのロジックを使用します。たとえば、 :containsを指定して、検索結果に含めるために大きな文字列フィールドに小さな文字列を含めるように指定できます。

サポートされている検索修飾子

検索修飾子	タイプ
: 欠落	を除くすべてのパラメータ `Composite`
:exact	String
:contains	String
:not	トークン
:text	トークン
:identifier	リファレンス
:以下	[URI]

サポートされている検索コンパレータ

検索コンパレータを使用して、検索の一致の性質を制御できます。数値、日付、数量フィールドを検索するときにコンパレータを使用できます。次の表に、HealthLake でサポートされている検索コンパレータとその定義を示します。

サポートされている検索コンパレータ

検索コンパレータ	説明
eq	リソースの パラメータの値は、指定された値と等しくなります。
ne	リソースの パラメータの値は、指定された値と等しくありません。
gt	リソースの パラメータの値が、指定された値を超えています。
lt	リソースの パラメータの値が、指定された値未満です。

検索コンパレータ	説明
g	リソースの パラメータの値は、指定された値以上です。
le	リソースの パラメータの値は、指定された値以下です。
sa	リソースの パラメータの値は、指定された値の後に開始されます。
EBS	リソースの パラメータの値は、指定された値より前に終了します。

HealthLake でサポートされていない FHIR 検索パラメータ

HealthLake は、次の表に示すものを除き、すべての FHIR 検索パラメータをサポートします。FHIR 検索パラメータの完全なリストについては、FHIR 検索パラメータレジストリを参照してください。

サポートされていない検索パラメータ

バンドル構成	Location-near
バンドル識別子	Consent-source-reference
バンドルメッセージ	契約患者
バンドルタイプ	リソースの内容
バンドルタイムスタンプ	リソースクエリ

HealthLake **$operations**の FHIR R4

FHIR $ オペレーション (「ドルオペレーション」とも呼ばれます) は、FHIR 仕様の標準 CRUD (`Create`、、`ReadUpdate`、`Delete`) オペレーションを超えて拡張される特殊なサーバー側の関数です。これらのオペレーションは「$」プレフィックスで識別され、標準の REST API コールを使用して実行するのが困難または非効率的な複雑な処理、データ変換、一括オペレーションを可能にします。$ オペレーションは、システムレベル、リソースタイプレベル、または特定のリソースイン

スタンスで呼び出すことができ、FHIR データを操作する柔軟な方法を提供します。 は複数の FHIR AWS HealthLake をサポートします$operations。詳細については、以下の個々のページを参照してください。

トピック

HealthLake の FHIR R4 **$attribution-status**オペレーション

特定のメンバーの属性ステータスを取得し、患者に関連するすべての属性リソースを含むバンドルを返します。このオペレーションは、FHIR メンバー属性 (ATR) リスト IG 2.1.0 実装の一部です。

Endpoint

```
POST [base]/Group/[id]/$attribution-status
```

リクエストパラメーター

オペレーションでは、次のオプションパラメータを使用できます。

パラメータ	Type	説明
memberId	識別子	属性ステータスがリクエストされたメンバーのMemberId
patientReference	リファレンス	プロデューサーのシステム内の患者リソースへの参照

> **Note**
>
> 検証目的でpatientReference、 memberIdまたは のいずれか、または両方を指定できます。

リクエスト例

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "memberId",
      "valueIdentifier": {
        "system": "http://example.org",
        "value": "MBR123456789"
      }
    },
    {
      "name": "patientReference",
      "valueReference": {
        "reference": "Patient/patient-123",
        "display": "John Doe"
      }
    }
  ]
}
```

応答

患者に関連する属性リソースを含むバンドルを返します。

[リソース]	カーディナリティ	ロケーション
患者	1..1	Group.member.entity
カバレッジ	0..1	Group.member.extension:coverageReference
Organization/Practitioner/PractitionerRole	0..1	Group.member.extension:attributedProvider
任意のリソース	0..1	Group.member.extension:associatedData

レスポンス例

```
{
  "resourceType": "Bundle",
  "id": "bundle-response",
  "meta": {
    "lastUpdated": "2014-08-18T01:43:33Z"
  },
  "type": "collection",
  "entry": [
    {
      "fullUrl": "http://example.org/fhir/Patient/12423",
      "resource": {
        "resourceType": "Patient",
        "id": "12423",
        "meta": {
          "versionId": "1",
          "lastUpdated": "2014-08-18T01:43:31Z"
        },
        "active": true,
        "name": [
          {
            "use": "official",
            "family": "Chalmers",
            "given": ["Peter", "James"]
```

```
          }
        ],
        "gender": "male",
        "birthDate": "1974-12-25"
      }
    },
    {
      "fullUrl": "http://example.org/fhir/Coverage/123456",
      "resource": {
        "resourceType": "Coverage",
        "id": "1"
        // ... additional Coverage resource details
      }
    },
    {
      "fullUrl": "http://example.org/fhir/Organization/666666",
      "resource": {
        "resourceType": "Organization",
        "id": "2"
        // ... additional Organization resource details
      }
    }
  ]
}
```

エラー処理

オペレーションは、次のエラー条件を処理します。

エラー	HTTP ステータス	説明
無効なオペレーションリクエスト	400	非準拠のリクエストパラメータまたは構造
グループリソースが見つかりません	404	指定されたグループ ID は存在しません
患者リソースが見つかりません	404	指定された患者リファレンスは存在しません

認可とセキュリティ

SMART スコープの要件

クライアントには、グループリソースおよび関連する属性リソースを読み取るための適切な権限が必要です

標準の FHIR 認可メカニズムがすべてのオペレーションに適用されます

を使用したリソースタイプの削除 **$bulk-delete**

AWS HealthLake は $bulk-deleteオペレーションをサポートし、データストア内の特定のタイプのすべてのリソースを削除できます。このオペレーションは、以下が必要な場合に特に役立ちます。

- 季節的な監査とクリーンアップを実行する
- 大規模なデータライフサイクルの管理
- 特定のリソースタイプを削除する
- データ保持ポリシーに準拠する

使用方法

$bulk-delete オペレーションは POST メソッドを使用して呼び出すことができます。

```
POST [base]/[ResourceType]/$bulk-delete?isHardDelete=false&deleteAuditEvent=true
```

パラメータ

パラメータ	タイプ	[Required] (必須)	デフォルト	説明
isHardDelete	boolean	不可	false	true の場合、ストレージからリソースを完全に削除します
deleteAuditEvent	boolean	なし	true	true の場合、関連する監査イベントを削除します
_since	string	不可	データストアの作成時刻	入力すると、開始カットオフ時間を選択して、lastModified時間に基づいてリ

パラメータ	タイプ	[Required] (必須)	デフォルト	説明
				ソースを検索します。開始または終了では使用できません
start	string	不可	データストアの作成時刻	入力すると、カットオフ時間を選択して、lastModified時間に基づいてリソースを検索します。末尾で使用できます
end	string	不可	ジョブの送信時間	入力すると、終了カットオフ時間を選択して、lastModified時間に基づいてリソースを検索します。

例

リクエストの例

```
POST [base]/Observation/$bulk-delete?isHardDelete=false
```

レスポンスの例

```
{
      "jobId": "jobId",
      "jobStatus": "SUBMITTED"
    }
```

ジョブのステータス

一括削除ジョブのステータスを確認するには:

```
GET [base]/$bulk-delete/[jobId]
```

オペレーションはジョブステータス情報を返します。

```
{
      "datastoreId": "datastoreId",
```

```
    "jobId": "jobId",
    "status": "COMPLETED",
    "submittedTime": "2025-10-09T15:09:51.336Z"
}
```

行動

$bulk-delete オペレーション:

1. 大量のリソースを処理するために非同期的に処理する
2. データ整合性のために ACID トランザクションを維持します
3. リソース削除数を含むジョブステータスの追跡を提供します
4. ソフト削除モードとハード削除モードの両方をサポート
5. 削除アクティビティの包括的な監査ログ記録が含まれます
6. 履歴バージョンと監査イベントの選択的な削除を許可する

監査ログ記録

$bulk-delete オペレーションは、詳細なオペレーション情報とともに StartFHIRBulkDeleteJob および DescribeFHIRBulkDeleteJobとしてログに記録されます。

制限事項

- isHardDelete が true に設定されている場合、ハード削除されたリソースは検索結果や_historyクエリに表示されません。
- このオペレーションで削除されるリソースは、処理中に一時的にアクセスできない場合があります。
- ストレージ計測は履歴バージョンでのみ調整されます - deleteVersionHistory=false はデータストアストレージを調整しません

$bulk-member-match HealthLake の オペレーション

AWS HealthLake は、複数のメンバー一致リクエストを非同期的に処理する $bulk-member-matchオペレーションをサポートします。このオペレーションにより、医療組織は、属性とカバレッジ情報を 1 回の一括リクエストで使用して、さまざまな医療システム全体で何百人ものメンバーの一意の識別子を効率的に照合できます。この機能は、大規模なpayer-to-payerデータ交換、メンバー移行、CMS コンプライアンス要件に不可欠であり、FHIR 仕様に従います。

> **Note**
>
> $bulk-member-match オペレーションは、現在実験段階にあり、変更される可能性のある基盤となる FHIR 仕様に基づいています。仕様が進化するにつれて、この API の動作とインターフェイスはそれに応じて更新されます。開発者は、AWS HealthLake リリースノートと関連する FHIR 仕様の更新をモニタリングして、統合に影響を与える可能性のある変更を常に把握することをお勧めします。

このオペレーションは、以下が必要な場合に特に役立ちます。

- オープン登録期間中に大規模なメンバーマッチングを処理する
- 支払者間の一括メンバー移行を容易にする
- 大規模な CMS コンプライアンスデータ交換要件をサポート
- 医療ネットワーク全体でメンバーコホートを効率的にマッチングする
- API コールを最小限に抑え、大量のマッチングシナリオの運用効率を向上させる

使用方法

$bulk-member-match オペレーションは、POST メソッドを使用してグループリソースで呼び出される非同期オペレーションです。

```
POST [base]/Group/$bulk-member-match
```

一括一致リクエストを送信したら、以下を使用してジョブステータスをポーリングできます。

```
GET [base]/$bulk-member-match-status/{jobId}
```

サポートされているパラメータ

HealthLake は、次の FHIR $bulk-member-matchパラメータをサポートしています。

パラメータ	タイプ	必須	説明
MemberPatient	患者	はい	一致するメンバーの属性情報を含む患者リソース。

パラメータ	タイプ	必須	説明
CoverageToMatch	カバレッジ	はい	既存のレコードとの照合に使用されるカバレッジリソース。
CoverageToLink	カバレッジ	いいえ	一致するプロセス中にリンクされるカバレッジリソース。
Consent	同意	はい	認可のための同意リソースも保存されます。これは、同意を必要としない個々の$member-match オペレーションとは異なります。

一括メンバー一致ジョブを送信する POST リクエスト

次の例は、一括メンバー一致ジョブを送信する POST リクエストを示しています。各メンバーは、必要な MemberPatient、CoverageToMatch、および Consentリソースとオプションの を含むMemberBundleパラメータでラップされますCoverageToLink。

```
POST [base]/Group/$bulk-member-match
Content-Type: application/fhir+json
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "MemberBundle",
      "part": [
        {
          "name": "MemberPatient",
          "resource": {
            "resourceType": "Patient",
            "identifier": [
              {
                "system": "http://example.org/patient-id",
                "value": "patient-0"
              }
            ],
            "name": [
              {
                "family": "Smith",
                "given": ["James"]
              }
```

```
            ],
            "gender": "male",
            "birthDate": "1950-01-01"
          }
        },
        {
          "name": "CoverageToMatch",
          "resource": {
            "resourceType": "Coverage",
            "status": "active",
            "identifier": [
              {
                "system": "http://example.org/coverage-id",
                "value": "cov-0"
              }
            ],
            "subscriberId": "sub-0",
            "beneficiary": {
              "reference": "Patient/patient123"
            },
            "relationship": {
              "coding": [
                {
                  "system": "http://terminology.hl7.org/CodeSystem/subscriber-
relationship",
                  "code": "self"
                }
              ]
            },
            "payor": [
              {
                "reference": "Organization/org123"
              }
            ]
          }
        },
        {
          "name": "Consent",
          "resource": {
            "resourceType": "Consent",
            "status": "active",
            "scope": {
              "coding": [
                {
```

```
                "system": "http://terminology.hl7.org/CodeSystem/consentscope",
                "code": "patient-privacy"
              }
            ]
          },
          "category": [
            {
              "coding": [
                {
                  "system": "http://terminology.hl7.org/CodeSystem/v3-ActCode",
                  "code": "IDSCL"
                }
              ]
            }
          ],
          "patient": {
            "reference": "Patient/patient123"
          },
          "performer": [
            {
              "reference": "Patient/patient123"
            }
          ],
          "sourceReference": {
            "reference": "http://example.org/DocumentReference/consent-source"
          },
          "policy": [
            {
              "uri": "http://hl7.org/fhir/us/davinci-hrex/StructureDefinition-hrex-
consent.html#regular"
            }
          ],
          "provision": {
            "type": "permit",
            "period": {
              "start": "2024-01-01",
              "end": "2025-12-31"
            },
            "actor": [
              {
                "role": {
                  "coding": [
                    {
```

```
                        "system": "http://terminology.hl7.org/CodeSystem/provenance-
participant-type",
                        "code": "performer"
                      }
                    ]
                  },
                  "reference": {
                    "identifier": {
                      "system": "http://hl7.org/fhir/sid/us-npi",
                      "value": "9876543210"
                    },
                    "display": "Old Health Plan"
                  }
                },
                {
                  "role": {
                    "coding": [
                      {
                        "system": "http://terminology.hl7.org/CodeSystem/v3-
ParticipationType",
                        "code": "IRCP"
                      }
                    ]
                  },
                  "reference": {
                    "identifier": {
                      "system": "http://hl7.org/fhir/sid/us-npi",
                      "value": "0123456789"
                    },
                    "display": "New Health Plan"
                  }
                }
              ],
              "action": [
                {
                  "coding": [
                    {
                      "system": "http://terminology.hl7.org/CodeSystem/consentaction",
                      "code": "disclose"
                    }
                  ]
                }
              ]
            }
```

```
          }
        },
        {
          "name": "CoverageToLink",
          "resource": {
            "resourceType": "Coverage",
            "status": "active",
            "identifier": [
              {
                "system": "http://example.org/coverage-link-id",
                "value": "cov-link-0"
              }
            ],
            "subscriberId": "new-sub-0",
            "beneficiary": {
              "reference": "Patient/patient123"
            },
            "relationship": {
              "coding": [
                {
                  "system": "http://terminology.hl7.org/CodeSystem/subscriber-
relationship",
                  "code": "self"
                }
              ]
            },
            "payor": [
              {
                "identifier": {
                  "system": "http://hl7.org/fhir/sid/us-npi",
                  "value": "0123456789"
                },
                "display": "New Health Plan"
              }
            ]
          }
        }
      ]
    }
  ]
}
```

完了したジョブレスポンスと出力

ジョブが完了すると、レスポンスにはジョブメタデータと、一致結果を分類する 3 つのグループリソースを含む FHIR パラメータリソースが含まれます。


```
{
  "datastoreId": "datastoreId",
  "jobId": "jobId",
  "status": "COMPLETED",
  "submittedTime": "2026-03-20T18:45:26.321Z",
  "numberOfMembers": 3,
  "numberOfMembersProcessedSuccessfully": 3,
  "numberOfMembersWithCustomerError": 0,
  "numberOfMembersWithServerError": 0,
  "output": {
    "resourceType": "Parameters",
    "meta": {
      "profile": [
        "http://hl7.org/fhir/us/davinci-pdex/StructureDefinition/pdex-parameters-multi-
member-match-bundle-out"
      ]
    },
    "parameter": [
      {
        "name": "MatchedMembers",
        "resource": {
          "resourceType": "Group",
          "id": "group1",
          "text": {
            "status": "generated",
            "div": "<div xmlns=\"http://www.w3.org/1999/xhtml\">Matched members group</
div>"
          },
          "contained": [
            {
              "resourceType": "Patient",
              "id": "1",
              "identifier": [
                {
                  "system": "http://example.org/patient-id",
                  "value": "patient-0"
                }
              ],
              "name": [
```

```
                {
                  "family": "Smith",
                  "given": ["James"]
                }
              ],
              "gender": "male",
              "birthDate": "1950-01-01"
            }
          ],
          "type": "person",
          "actual": true,
          "code": {
            "coding": [
              {
                "system": "http://hl7.org/fhir/us/davinci-pdex/CodeSystem/
PdexMultiMemberMatchResultCS",
                "code": "match",
                "display": "Matched"
              }
            ]
          },
          "quantity": 1,
          "member": [
            {
              "entity": {
                "extension": [
                  {
                    "url": "http://hl7.org/fhir/us/davinci-pdex/StructureDefinition/
base-ext-match-parameters",
                    "valueReference": {
                      "reference": "#1"
                    }
                  }
                ],
                "reference": "Patient/patient123"
              }
            }
          ]
        }
      },
      {
        "name": "NonMatchedMembers",
        "resource": {
          "resourceType": "Group",
```

```
          "id": "Group2",
          "text": {
            "status": "generated",
            "div": "<div xmlns=\"http://www.w3.org/1999/xhtml\">Non-matched members
 group</div>"
          },
          "contained": [
            {
              "resourceType": "Patient",
              "id": "1",
              "identifier": [
                {
                  "system": "http://example.org/patient-id",
                  "value": "patient-501"
                }
              ],
              "name": [
                {
                  "family": "Carter",
                  "given": ["Emily"]
                }
              ],
              "gender": "female",
              "birthDate": "1985-06-15"
            }
          ],
          "type": "person",
          "actual": true,
          "code": {
            "coding": [
              {
                "system": "http://hl7.org/fhir/us/davinci-pdex/CodeSystem/
PdexMultiMemberMatchResultCS",
                "code": "nomatch",
                "display": "Not Matched"
              }
            ]
          },
          "quantity": 1,
          "member": [
            {
              "entity": {
                "extension": [
                  {
```

```
                    "url": "http://hl7.org/fhir/us/davinci-pdex/StructureDefinition/
base-ext-match-parameters",
                    "valueReference": {
                      "reference": "#1"
                    }
                  }
                ],
                "reference": "Patient/patient123"
              }
            }
          ]
        }
      },
      {
        "name": "ConsentConstrainedMembers",
        "resource": {
          "resourceType": "Group",
          "id": "group3",
          "text": {
            "status": "generated",
            "div": "<div xmlns=\"http://www.w3.org/1999/xhtml\">Consent constrained
 members group</div>"
          },
          "contained": [
            {
              "resourceType": "Patient",
              "id": "1",
              "identifier": [
                {
                  "system": "http://example.org/patient-id",
                  "value": "patient-502"
                }
              ],
              "name": [
                {
                  "family": "Nguyen",
                  "given": ["David"]
                }
              ],
              "gender": "male",
              "birthDate": "1972-11-22"
            }
          ],
          "type": "person",
```

```
          "actual": true,
          "code": {
            "coding": [
              {
                "system": "http://hl7.org/fhir/us/davinci-pdex/CodeSystem/
PdexMultiMemberMatchResultCS",
                "code": "consentconstraint",
                "display": "Consent Constraint"
              }
            ]
          },
          "quantity": 1,
          "member": [
            {
              "entity": {
                "extension": [
                  {
                    "url": "http://hl7.org/fhir/us/davinci-pdex/StructureDefinition/
base-ext-match-parameters",
                    "valueReference": {
                      "reference": "#1"
                    }
                  }
                ],
                "reference": "Patient/123"
              }
            }
          ]
        }
      }
    ]
  }
}
```

HealthLake がメンバーを出力グループに分類する方法

$bulk-member-match リクエストで送信されたすべてのメンバーは、シーケンシャルパイプラインを通じて評価されます。各ステップの結果によって、メンバーを配置する出力グループが決まります。

1. 構造検証 — MemberBundle は必要なプロファイルに準拠していますか? 失敗時: エラー (どのグループでもない)。

2. 患者のマッチング — HealthLake は、送信された属性に一致する患者を見つけることができますか? 失敗の場合: NonMatchedMembers。
3. カバレッジの確認 — HealthLake は、有効な CoverageToMatch を持つ 1 人の患者だけに絞り込むことができますか? 失敗の場合: NonMatchedMembers。
4. 同意の評価 — 送信された同意は、現在受け入れられますか? (ステータス = アクティブ、期間は現在の日付をカバー、パフォーマーは検証可能) 。失敗の場合: ConsentConstrainedMembers。
5. Success — すべてのチェックに合格します。データストアに保存された同意。MatchedMembers に配置されたメンバー。

主な原則: メンバーは 1 つの送信先にのみ表示できます。最初の失敗ステップによって配置が決まります。ステップ 2 または 3 に失敗したメンバーはConsentConstrainedMembersに配置されません。そのグループは、正常に一致したが、同意を尊重できないメンバー専用です。

同意評価の詳細 (ステップ 4):

- チェック 1 — 同意ステータス:「アクティブ」と`Consent.status`等しいか? そうでない場合 → ConsentConstrainedMembers。
- チェック 2 — プロビジョニング期間: 現在の日付を`provision.period`カバーしていますか? 現在の日付が前`period.start`または後の場合 `period.end` → ConsentConstrainedMembers。
- チェック 3 — パフォーマーの検証: `Consent.performer`リファレンスは検証できますか? 参照されたリソースがデータストアで見つからない場合、または一致した患者に関連付けられていない場合 → ConsentConstrainedMembers。

メンバーを MatchedMembers に配置し、同意を保存するには、すべてのチェックに合格する必要があります。

カバレッジマッチングの動作

メンバーのマッチング中、 `CoverageToMatch`は応答する支払者のデータストアに対してのみ検証されます。 は新規/リクエストする支払者に`CoverageToLink`属し、古い支払者のデータストアに対して検証されません。リクエスト`CoverageToLink`に を含めても、一致する結果には影響しません。

リクエストの各患者とカバレッジの組み合わせは個別に処理されます。同じ患者を異なるカバレッジプランで複数回送信でき、各エントリは特定のカバレッジに基づいて独自の結果を受け取ります。

同意パフォーマーのリファレンス処理

新しい支払者は、 で一時的またはローカルな患者参照を送信できます `Consent.performer` (たとえば、 で使用されているのと同じ参照`Consent.patient`)。HealthLake はこれらの参照を自動的に解決します。

- に と同じローカルリファレンス`Consent.performer`が含まれている場合`Consent.patient`、HealthLake はマッチングが成功した後、実際に一致した患者リファレンスに置き換えます。
- HealthLake は、タイプ Patient、RelatedPerson、Practitioner、PractitionerRole、および Organization (直接参照と論理識別子参照の両方) のパフォーマー参照をサポートしています。
- パフォーマーの検証が失敗した場合 (リソースが見つからないか、一致する患者に関連付けられていない場合) 、メンバーはエラーを返すのではなくConsentConstrainedMembers に配置されます。

出力グループのリソース

完了したジョブは、3 つのグループリソースを含む Parameters リソースを返します。

MatchedMembers グループ

リクエスト時に同意がアクティブで有効な、正常に一致したすべてのメンバーに対する患者リファレンスが含まれます。同意リソースは、一致するメンバーごとに作成され、データストアに保存されます。このグループはデータストアでインスタンス化され、 で直接使用できます`$davinci-data-export`。

NonMatchedMembers グループ

一意の一致が見つからなかったメンバーへの参照が含まれます。データストア内のどの患者も指定された属性と一致しない場合、一致する患者候補に有効なカバレッジが存在しない場合、または複数の患者が属性に一致し、複数の患者が有効なカバレッジ (あいまい) を持っている場合、メンバーはここに配置されます。

ConsentConstrainedMembers グループ

正常にマッチングされた (人口統計とカバレッジが確認された) が、リクエスト時に同意を尊重できないメンバーに対する患者リファレンスが含まれます。同意リソースは、同意に制約のあるメンバーには保存されません。一致するメンバー ID (MemberIdentifier と MemberId) は引き続き含まれているため、リクエスト元の支払者は制約されたユーザーを認識します。

Group.quantity フィールドには、それぞれのグループのメンバーの合計数が含まれます。

グループメンバーリファレンス:

- Group.member.entity.reference — MatchedMembers と ConsentConstrainedMembers の場合、応答する支払者のシステム内の一致したメンバーの患者 ID が含まれます。NonMatchedMembers の場合、 は含まれている入力患者を参照します。
- Group.member.entity.extension（base-ext-match-parameters）— 元の入力リクエストからの患者 ID (Patient.id、、Coverage.beneficiary.referenceまたは から派生した、リクエスト元の支払者によって送信された ID) が含まれますConsent.patient.reference。

同意と連鎖

> **Important**
>
> 保存されている同意リソースは、リクエスト元の支払者から送信されたとおりに、患者リファレンスを保持します。HealthLake は、受信データストア内の一致する患者を指すように、同意の患者フィールドを自動的に更新しません。

保存されている同意を一致する患者にリンクするには、ジョブ出力を使用します。MatchedMembers グループ内の各メンバーには、一致する患者member.entity.referenceを指す と、含まれている入力患者を指す member.entity.extension (base-ext-match-parameters) があります。これらを同意の患者フィールドと相互参照して、アプリケーションレイヤーにマッピングを構築します。

保存されるものと一時的なもの

次の表は、$bulk-member-match処理中に HealthLake がデータストアに保持するものと、ジョブレスポンスにのみ存在するものを示しています。

[リソース]	保存済み?	REST 経由でクエリ可能か?	注意事項
MemberPatient (入力)	いいえ	いいえ	マッチングにのみ使用され、永続化されません

[リソース]	保存済み?	REST 経由でクエリ可能か?	注意事項
CoverageToMatch (入力)	いいえ	いいえ	カバレッジの確認にのみ使用されます
CoverageToLink (入力)	いいえ	いいえ	データストアに対して検証されていません。新しい支払者に属しています
同意 (一致するメンバー)	あり	はい — GET [base]/Consent/{id}	支払者のリクエストから受け取ったとおりに保存されます
同意 (制約のあるメンバー)	いいえ	いいえ	保存されていません。メンバー ID は引き続きレスポンスに含まれます。
MatchedMembers グループ (出力)	あり	はい — GET [base]/Group/{id}	インスタンス化。$davinci-data-export で使用可能
NonMatchedMembers グループ	いいえ	いいえ	ジョブレスポンスのみ
ConsentConstrained Members グループ	いいえ	いいえ	ジョブレスポンスのみ

$davinci-data-export との統合

によって返される MatchedMembers Group リソース`$bulk-member-match`は、一括メンバーデータを取得する `$davinci-data-export`オペレーションで直接使用できます。

```
POST [base]/Group/{matched-group-id}/$davinci-data-export
GET [base]/Group/{matched-group-id}
```

この統合により、一致したメンバーを最初に一括で識別し、結果のグループリソースを使用して完全なヘルスレコードをエクスポートする効率的なワークフローが可能になります。

エクスポート前に $member-remove を使用する

マッチング後にエクスポートから特定のメンバーを除外する必要がある場合 (たとえば、メンバーがマッチングとエクスポートの間の同意を取り消す場合) 、MatchedMembers グループ`$member-remove`で を使用します。

> **Important**
>
> でメンバーを削除すると、そのメンバーはグループ内で非アクティブとして`$member-remove`マークされますが、グループがステータス「最終」に更新された後に`$davinci-data-export`のみ非アクティブなメンバーを除外します。デフォルトのステータスのままのグループ`$davinci-data-export`で を呼び出しても、削除されたメンバーはエクスポート結果に表示されることがあります。

ワークフロー:

1. `POST [base]/Group/{id}/$member-remove` — メンバーを非アクティブにする
2. `PUT [base]/Group/{id}` — グループのステータスを「最終」に更新する
3. `POST [base]/Group/{id}/$davinci-data-export` — エクスポートで削除されたメンバーを除外するようになりました

パフォーマンス特性

`$bulk-member-match` オペレーションは大量の処理用に設計されており、非同期的に実行されます。

- 同時実行数: データストアあたり最大 5 回の同時オペレーション。
- スケーラビリティ: リクエストごとに最大 500 人のメンバーを処理します (5 MB のペイロード制限) 。
- 並列オペレーション: インポート、エクスポート、一括削除の同時オペレーションと互換性があります。

Authorization

API は、以下の必要なスコープで SMART on FHIR 認可プロトコルを使用します。

- `system/Patient.read` — 患者リソースを検索して一致させるために必要です。

- system/Coverage.read — カバレッジ情報を検証するために必要です。
- system/Group.write — 結果グループリソースを作成するために必要です。
- system/Organization.read — 条件付き。カバレッジが組織を参照する場合は必須です。
- system/Practitioner.read — 条件付き、カバレッジが実務者を参照する場合は必須です。
- system/PractitionerRole.read — 条件付き。カバレッジが実務者ロールを参照する場合は必須です。
- system/Consent.write — 条件付き。同意リソースが指定されている場合は必須です。

オペレーションは、プログラムによるアクセスの AWS IAM 署名バージョン 4 (SigV4) 認可もサポートしています。

検証ルール

ステップ 1 では、各 MemberBundle に次の検証ルールが適用されます。検証に失敗したメンバーはエラーとして報告され、出力グループには表示されません。

MemberPatient

フィールド	HealthLake がそれを使用する方法	次の場合、検証は失敗します。
name.family	デモグラフィック検索	Missing (見つからない)
name.given	デモグラフィック検索	欠落 (少なくとも 1 つ必須)
birthDate	デモグラフィック検索	Missing (見つからない)
gender	人口統計検索。存在しない場合は、Birth Sex 拡張機能を使用	性別も性別も存在しない (hrex-pat-1)
identifier	存在する場合は検索に含まれ、信頼度が向上	失敗を引き起こさない (オプション)

CoverageToMatch / CoverageToLink

フィールド	HealthLake がそれを使用する方法	次の場合、検証は失敗します。
status	カバレッジが実行可能であることを確認する	Missing (見つからない)
beneficiary	患者候補にカバレッジをリンクします	Missing (見つからない)
payor	複数の候補が存在する場合のあいまいさの排除	見つからない、または複数の支払者
relationship	サブスクライバーと受取人の関係を確認します	Missing (見つからない)
identifier (MB) or subscriberId	主な曖昧さ解消キー	どちらでもない

同意

フィールド	HealthLake がそれを使用する方法	次の場合、検証は失敗します。
scope	同意範囲が患者プライバシーであることを確認する	患者/プライベートコードがないか、ない
category	開示分類を確認します	Missing (見つからない)
patient	同意サブジェクトを識別します	Missing (見つからない)
performer	同意するユーザーを特定します。	Missing (見つからない)
sourceReference	同意ソースを文書化します。	Missing (見つからない)
policy.uri	データ共有の範囲を決定します	#regular または #sensitive で終わる URI がないか、URI がありません

フィールド	HealthLake がそれを使用する方法	次の場合、検証は失敗します。
`provision.type`	HRex 同意プロファイルごとに「許可」である必要があります	「許可」がないかないか (「拒否」を含む)
`provision.period`	同意に制約のあるチェックのためにステップ 4 で評価	開始/終了がないか、ない
`status`	ステップ 4 で評価 (ステップ 1 ではない)	ステップ 1 の失敗を引き起こさない — HealthLake は有効なステータスを受け入れ、ステップ 4 で評価します

> **Note**
>
> HRex 同意プロファイルは、固定値が「アクティブ」のステータスを定義します。HealthLake は、非アクティブな同意が包括的な検証拒否ではなく意味のある分類 (ConsentConstrainedMembers) を受け取るように、この制約を意図的に緩和します。

マッチング動作

- 患者検索 (ステップ 2) — HealthLake は、`name.family`+ `name.given` (完全、大文字と小文字を区別しない)`birthDate`、 (完全) 、 `gender` (性別がない場合に使用される完全、性別を使用) 、および `identifier` (存在する場合、オプション) を使用して検索します。
- カバレッジのあいまいさの排除 (ステップ 3) — 複数の患者候補が見つかった場合、`CoverageToMatch` を使用して 1 つに絞り込みます。または `subscriberId` `identifier` (MB タイプ) と に一致するデータストアにアクティブなカバレッジリソースが存在する場合、カバレッジは「有効」になりますpayor。
- 同意評価 (ステップ 4) — 一意の一致が成功した後にのみ実行されます。上記の同意評価の詳細セクションを参照してください。

エラー処理

オペレーションは、次のエラー条件を処理します。

- 400 不正なリクエスト: 無効なリクエスト形式、必須パラメータの欠落、またはペイロードがサイズ制限 (500 メンバーまたは 5 MB) を超えています。
- 422 未処理のエンティティ: ジョブ実行中の処理エラー。
- 個々のメンバーエラー: 特定のメンバーが処理に失敗した場合、オペレーションは残りのメンバーを続行し、適切な理由コードを使用して NonMatchedMembers グループにエラーの詳細を含めます。たとえば、 `birthDate`パラメータが欠落`MemberBundle`している患者がいる は、次のエラーを返します。

```
"errors": [
  {
    "memberIndex": 1,
    "jsonBlob": {
      "resourceType": "OperationOutcome",
      "issue": [
        {
          "severity": "error",
          "code": "invalid",
          "diagnostics": "MemberPatient.birthDate is required"
        }
      ],
      "statusCode": 400
    }
  }
]
```

エラーの詳細は、ステータスポーリングエンドポイントを通じて利用でき、以下が含まれます。

- `numberOfMembersWithCustomerError`: 検証エラーまたは入力エラーがあるメンバーの数。
- `numberOfMembersWithServerError`: サーバー側の処理エラーがあるメンバーの数。

関連の オペレーション

- the section called “$member-match” — 個々のメンバーマッチングオペレーション。
- the section called “$davinci-data-export” — グループリソースを使用した一括データエクスポート。
- the section called “$オペレーション” — サポートされているオペレーションの完全なリスト。

HealthLake の FHIR R4 **$confirm-attribution-list**オペレーション

コンシューマーが属性リストに加える変更がないことをプロデューサーに示します。このオペレーションは、リストから非アクティブなメンバーを削除し、ステータスを「最終」に変更して、オペレーションを承認することで、属性リストを確定します。このオペレーションは、FHIR メンバー属性 (ATR) リスト IG 2.1.0 実装の一部です。

Endpoint

```
POST [base]/Group/[id]/$confirm-attribution-list
```

リクエスト

パラメータは必要ありません。

```
POST [base]/Group/[id]/$confirm-attribution-list
```

応答

確認メッセージを含む HTTP 201 を返します。

レスポンス例

```
HTTP Status Code: 201

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "message",
      "valueString": "Accepted."
    }
  ]
}
```

確認後のグループステータス

確認に成功すると、グループリソースの属性リストのステータスが「最終」に設定されます。

```
{
```

```
  "resourceType": "Group",
  "id": "fullexample",
  "extension": [
    {
      "url": "http://hl7.org/fhir/us/davinci-atr/StructureDefinition/ext-
attributionListStatus",
      "valueCode": "final"
    }
  ]
  // ... other Group properties
}
```

エラー処理

オペレーションは、次のエラー条件を処理します。

エラー	HTTP ステータス	説明
無効なオペレーションリクエスト	400	非準拠のリクエストパラメータまたは構造
グループリソースが見つかりません	404	指定されたグループ ID は存在しません

認可とセキュリティ

SMART スコープの要件

クライアントには、グループリソースおよび関連する属性リソースを読み取るための適切な権限が必要です

の場合`$confirm-attribution-list`、クライアントにはグループリソースを変更するための書き込み権限も必要です。

標準の FHIR 認可メカニズムがすべてのオペレーションに適用されます

HealthLake の FHIR R4 **`$davinci-data-export`**オペレーション

`$davinci-data-export` オペレーションは、医療データのエクスポートに使用できる非同期 FHIR オペレーションです AWS HealthLake。このオペレーションは、メンバー属性 (ATR)、PDex プロ

バイダーアクセス、Payer-to-Payer、メンバーアクセス APIs など、複数のエクスポートタイプをサポートしています。これは、DaVinci 実装ガイドの要件を満たすように設計された、標準の FHIR `$export`オペレーションの特殊なバージョンです。

主な機能

- 非同期処理: 標準の FHIR 非同期リクエストパターンに従います
- グループレベルのエクスポート: 特定のグループリソース内のメンバーのデータをエクスポートします。
- 複数のエクスポートタイプ: ATR (メンバー属性) 、PDex プロバイダーアクセス、Payer-to-Payer、およびメンバーアクセス APIsをサポート
- 包括的なプロファイルのサポート: US Core、CARIN Blue ボタン、PDex プロファイルが含まれます
- 柔軟なフィルタリング: 患者、リソースタイプ、時間範囲によるフィルタリングをサポート
- NDJSON 出力: データを改行区切りの JSON 形式で提供します

オペレーションエンドポイント

```
GET [base]/Group/[id]/$davinci-data-export
POST [base]/Group/[id]/$davinci-data-export
```

リクエストパラメーター

パラメータ	カーディナリティ	説明
`patient`	0..*	エクスポートするデータを持つ特定のメンバー。省略すると、グループ内のすべてのメンバーがエクスポートされます。
`_type`	0..1	エクスポートする FHIR リソースタイプのカンマ区切りリスト。省略すると、指定されたエクスポートタイプでサポートされているすべてのリソースタイプが含まれます。ATR エクスポートの場合、デフォルトで 8 つの属性リソースタイプになります。PDex エクスポートの場合、これには、すべての属性リソースタイプに加えて、US Core、CARIN Blue Button、およ

パラメータ	カーディナリティ	説明
		び PDex プロファイルからの臨床リソースタイプとクレームリソースタイプが含まれます。
_since	0..1	この日時以降に更新されたリソースのみを含めます。
_until	0..1	この日時より前に更新されたリソースのみを含めます。
exportType	0..1	実行するエクスポートのタイプ。有効な値は、hl7.fhir.us.davinci-atr、hl7.fhir.us.davinci-pdex、hl7.fhir.us.davinci-pdex.p2p 、hl7.fhir.us.davinci-pdex.member です。デフォルト: hl7.fhir.us.davinci-atr 。
_includeEOB2xWoFinancial	0..1	財務データが削除された CARIN BB 2.x ExplanationOfBenefit リソースを含めるかどうかを指定します。デフォルト: false。
_security	0..*	エクスポートされたリソースをmeta.security コーディング値でフィルタリングします。system|code 形式を使用します (パイプ文字は として URL エンコードされている必要があります%7C)。複数の値を指定する場合、リソースはすべての値 (AND セマンティクス) と一致する必要があります。system| (証跡パイプ、コードなし) を使用して、特定のシステムのコードと照合します。

パラメータ	カーディナリティ	説明
_tag	0..*	エクスポートされたリソースをmeta.tagコーディング値でフィルタリングします。と同じsystem|code 形式と AND セマンティクスを使用します_security 。_security と の両方を指定する場合、リソース_tagは両方のフィルターと一致する必要があります。

サポートされているリソースタイプ

サポートされているリソースタイプは、指定したエクスポートタイプによって異なります。ATR エクスポートでは、次のリソースタイプがサポートされています。

- Group
- Patient
- Coverage
- RelatedPerson
- Practitioner
- PractitionerRole
- Organization
- Location

PDex エクスポート (プロバイダーアクセス、Payer-to-Payer、メンバーアクセス) では、前述のタイプに加えて、すべての臨床リソースタイプとクレームリソースタイプがサポートされています。サポートされているリソースタイプの詳細なリストについては、US Core Implementation Guide (STU 6.1)、CARIN Blue Button Implementation Guide、Da Vinci Prior Authorization Support Implementation Guide を参照してください。

エクスポートタイプ

$davinci-data-export オペレーションでは、次のエクスポートタイプがサポートされています。エクスポートタイプを指定するには、 exportTypeパラメータを使用します。

エクスポートタイプ	目的	データスコープ	一時的な制限
`hl7.fhir.us.davinci-atr`	メンバー属性リスト	属性関連のリソース	なし
`hl7.fhir.us.davinci-pdex`	プロバイダーアクセス API	属性患者の臨床データとクレームデータ	5 年
`hl7.fhir.us.davinci-pdex.p2p`	Payer-to-Payer Exchange	保険移行の履歴メンバーデータ	5 年
`hl7.fhir.us.davinci-pdex.member`	メンバーアクセス API	メンバー自身のヘルスデータ	5 年

> **Note**
>
> PDex エクスポートの場合、5 年間の時間制限は ATR リソースタイプ (`Group`、`Patient`、、`Coverage`、、、`RelatedPerson``PractitionerRole`、`Practitioner`、`Organization`、`Location`には適用されません。これらのリソースは、年齢に関係なく常に含まれます。

ATR (hl7.fhir.us.davinci-atr)

ATR エクスポートタイプを使用すると、メンバー属性リストデータをエクスポートできます。このエクスポートタイプを使用して、グループ内のメンバーの属性関連のリソースを取得します。詳細については、「Da Vinci ATR Export Operation」を参照してください。

サポートされているリソースタイプ

`Group`, `Patient`, `Coverage`, `RelatedPerson`, `Practitioner`, `PractitionerRole`, `Organization`, `Location`

一時フィルタリング

一時的なフィルタリングは適用されません。一致するすべてのリソースは、日付に関係なくエクスポートされます。

PDex エクスポートタイプ

すべての PDex エクスポートタイプは、サポートされている同じプロファイルとフィルタリングロジックを共有します。詳細については、「Da Vinci PDex Provider Access API」を参照してください。次のプロファイルがサポートされています。

- 米国 Core 3.1.1、6.1.0、および 7.0.0
- PDex 事前認可 (メンバーアクセスではサポートされていません)
- CARIN BB 2.x Basis プロファイル: 入院患者用、外来患者用、専門のNonClinician、オーラル、薬剤性

PDex エクスポートの場合、 グループの患者ごとに臨床リソースとクレームリソースが自動的に検出されます。グループリソースでこれらのリソースを明示的に参照する必要はありません。オペレーションは`Observation`、属性化された患者に属するすべての患者セグメントリソース (`Condition`、`Coverage`、`RelatedPerson`、`MedicationRequest`、、 など`ExplanationOfBenefit`) を検索します。`Patient`、`Group`、およびnon-patient-compartment ATR タイプ (`Practitioner`、`PractitionerRole`、`Organization`、`Location`) のみが、 グループで明示的な参照を必要とします。

プロバイダーアクセス (`hl7.fhir.us.davinci-pdex`)

ネットワーク内プロバイダーが属性患者の患者データを取得できるようにします。

Payer-to-Payer (`hl7.fhir.us.davinci-pdex.p2p`)

患者が保険を変更するときに、支払者間のデータ交換を有効にします。

メンバーアクセス (`hl7.fhir.us.davinci-pdex.member`)

メンバーが自分のヘルスデータにアクセスできるようにします。このエクスポートタイプには、クレームリソースに財務データが含まれる場合があります。

プロファイルのサポートと包含ロジック

PDex エクスポートの場合、 `$davinci-data-export`オペレーションは `meta.profile`要素のプロファイル宣言を使用して、エクスポートに含めるリソースを決定します。

ExplanationOfBenefit リソース処理

`ExplanationOfBenefit` (EOB) リソースは、`meta.profile`宣言に基づいて PDex エクスポートに含まれるか除外されます。

- CARIN BB 1.x プロファイルを持つ ExplanationOfBenefit リソースはエクスポートから除外されます。
- `meta.profile` 設定されていない ExplanationOfBenefit リソースはエクスポートから除外されます。
- CARIN BB 2.x Basis プロファイルを持つ ExplanationOfBenefit リソースは常に含まれます。
- 財務データを含む CARIN BB 2.x プロファイルを持つ ExplanationOfBenefit リソースは、デフォルトで除外されます。`_includeEOB2xWoFinancial=true` が設定されている場合、それらは削除された財務データに含まれ、リソースは対応する Basis プロファイルに変換されます。
- PDex 事前認可プロファイルを持つ ExplanationOfBenefit リソースは常に含まれます。

財務データ変換

を設定すると`_includeEOB2xWoFinancial=true`、オペレーションは財務データを削除して CARIN BB 2.x ExplanationOfBenefit リソースを対応する Basis プロファイルに変換します。たとえば、`C4BB ExplanationOfBenefit Oral`リソースは に変換され`C4BB ExplanationOfBenefit Oral Basis`、FHIR 仕様に従ってレコードから財務データが削除されます。

変換中は、次の財務データ要素が削除されます。

- `total` 要素のすべてのスライス
- `amounttype` スライスを持つすべての`adjudication`要素
- 金額情報を含むすべての`item.adjudication`要素

このオペレーションは、変換中にプロファイルメタデータも更新します。

- `meta.profile` が Basis プロファイルの正規 URL に更新されました
- バージョンが CARIN BB 2.x Basis バージョンに更新されました
- データストア内の既存のリソースは変更されません
- エクスポートされたリソースはデータストアに保持されません

プロファイル検出ルール

オペレーションでは、次のルールを使用してプロファイルを検出および検証します。

- バージョン検出は`meta.profile`正規 URLsに基づいています

- 宣言されたプロファイルのいずれかがエクスポート基準に一致する場合、リソースが含まれます。
- プロファイルの検証はエクスポート処理中に行われます

PDex エクスポートの 5 年間の一時フィルタリング

すべての PDex エクスポートタイプについて、HealthLake はリソースが最後に更新された日時に基づいて 5 年間の時間フィルターを適用します。時間フィルターは、年齢に関係なく常にエクスポートされる以下のコア属性リソースタイプを除くすべてのリソースに適用されます。

- `Patient`
- `Coverage`
- `Organization`
- `Practitioner`
- `PractitionerRole`
- `RelatedPerson`
- `Location`
- `Group`

これらの管理リソースと属性リソースは、エクスポートされたデータに不可欠なコンテキストを提供するため、除外されます。ATR エクスポートは一時的なフィルタリングの対象ではありません。

リクエスト例

次の例は、さまざまなエクスポートタイプのエクスポートジョブを開始する方法を示しています。

ATR エクスポート

```
GET https://healthlake.{region}.amazonaws.com/datastore/
{datastoreId}/r4/Group/example-group/$davinci-data-export?
_type=Group,Patient,Coverage,Practitioner,Organization&exportType=hl7.fhir.us.davinci-
atr

POST https://healthlake.{region}.amazonaws.com/datastore/
{datastoreId}/r4/Group/example-group/$davinci-data-export?
_type=Group,Patient,Coverage,Practitioner,Organization&exportType=hl7.fhir.us.davinci-
atr
Content-Type: application/json
```

```
{
  "DataAccessRoleArn": "arn:aws:iam::444455556666:role/your-healthlake-service-role",
  "JobName": "attribution-export-job",
  "OutputDataConfig": {
    "S3Configuration": {
      "S3Uri": "s3://your-export-bucket/EXPORT-JOB",
      "KmsKeyId":
 "arn:aws:kms:region:444455556666:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    }
  }
}
```

ExplanationOfBenefit 財務データの削除によるプロバイダーアクセスのエクスポート

```
GET https://healthlake.{region}.amazonaws.com/datastore/
{datastoreId}/r4/Group/example-group/$davinci-data-export?
_type=Patient,Observation,Condition,MedicationRequest,ExplanationOfBenefit&exportType=hl7.fhir.
pdex&_includeEOB2xWoFinancial=true
```

Payer-to-Payer エクスポート

```
GET https://healthlake.{region}.amazonaws.com/datastore/
{datastoreId}/r4/Group/example-group/$davinci-data-export?
_type=Patient,Coverage,ExplanationOfBenefit,Condition,Procedure&exportType=hl7.fhir.us.davinci-
pdex.p2p&_includeEOB2xWoFinancial=true
```

特定の患者のメンバーアクセスエクスポート

```
GET https://healthlake.{region}.amazonaws.com/datastore/
{datastoreId}/r4/Group/example-group/$davinci-data-export?
_type=Patient,Observation,Condition,ExplanationOfBenefit,MedicationRequest&exportType=hl7.fhir.
pdex.member&patient=Patient/example-patient-id
```

レスポンス例

```
{
  "datastoreId": "eaee622d8406b41eb86c0f4741201ff9",
  "jobStatus": "SUBMITTED",
  "jobId": "48d7b91dae4a64d00d54b70862f33f61"
}
```

リソース関係

オペレーションは、メンバー属性リスト内の関係に基づいてリソースをエクスポートします。

```
Group (Attribution List)
### Patient (Members)
### Coverage # RelatedPerson (Subscribers)
### Practitioner (Attributed Providers)
### PractitionerRole # Location
### Organization (Attributed Providers)
```

> **Note**
>
> 前述のリソース関係図は、ATR エクスポートに適用されます。PDex エクスポートの場合、臨床リソースとクレームリソースは患者検索によって自動的に検出され、グループリソースで明示的な参照は必要ありません。

リソースソース

[リソース]	ソースの場所	説明
Patient	Group.member.entity	属性リストのメンバーである患者
Coverage	Group.member.extension:coverageReference	患者メンバーシップにつながるカバレッジ
Organization	Group.member.extension:attributedProvider	患者が属する組織
Practitioner	Group.member.extension:attributedProvider	患者が属する個々の実務者
PractitionerRole	Group.member.extension:attributedProvider	患者が属するプラクティショナーロール
RelatedPerson	Coverage.subscriber	カバレッジのサブスクライバー

[リソース]	ソースの場所	説明
Location	PractitionerRole.location	プラクティショナーロールに関連付けられた場所
Group	入力エンドポイント	属性リスト自体

ジョブ管理

ジョブのステータスを確認する

```
GET [base]/export/[job-id]
```

ジョブをキャンセルする

```
DELETE [base]/export/[job-id]
```

ジョブのライフサイクル

- SUBMITTED - ジョブが受信およびキューに入れられました
- IN_PROGRESS - ジョブはアクティブに処理中です
- COMPLETED - ジョブが正常に終了し、ダウンロード可能なファイル
- FAILED - ジョブでエラーが発生しました

[Output Format] (出力形式)

- ファイル形式: NDJSON (Newline Delimited JSON)
- File Organization: リソースタイプごとにファイルを区切ります。
- ファイル拡張子: .ndjson
- 場所: 指定された S3 バケットとパス

エラー処理

オペレーションは、以下の条件で HTTP 400 Bad Request with an OperationOutcome を返します。

認可エラー

で指定された IAM ロールには、エクスポートオペレーションを実行するための十分なアクセス許可DataAccessRoleArnがありません。必要な S3 および KMS アクセス許可の完全なリストについては、「エクスポートジョブのアクセス許可の設定」を参照してください。

パラメータ検証エラー

- patient パラメータは としてフォーマットされていません Patient/id,Patient/id,...
- 1 つ以上の患者参照が無効であるか、指定されたグループに属していません
- exportType パラメータ値がサポートされているエクスポートタイプではありません
- _type パラメータには、指定されたエクスポートタイプでサポートされていないリソースタイプが含まれています
- _type パラメータにhl7.fhir.us.davinci-atrエクスポートタイプに必要なリソースタイプ (Group、Patient、Coverage) がありません
- _includeEOB2xWoFinancial パラメータ値が有効なブール値ではありません

リソース検証エラー

- 指定されたグループリソースがデータストアに存在しません
- 指定されたグループリソースにはメンバーがありません
- 1 人以上のグループメンバーが、データストアに存在しない患者リソースを参照する

セキュリティと認可

- 標準の FHIR 認可メカニズムが適用されます
- データアクセスロールには、S3 および KMS オペレーションに必要な IAM アクセス許可が必要です。必要なアクセス許可の完全なリストについては、「エクスポートジョブのアクセス許可の設定」を参照してください。

ベストプラクティス

- リソースタイプの選択: エクスポートサイズと処理時間を最小限に抑えるために必要なリソースタイプのみをリクエストする
- 時間ベースのフィルタリング: 増分エクスポートに _sinceパラメータを使用する
- 患者のフィルタリング: 特定のメンバーのデータのみが必要な場合は、 patientパラメータを使用します。
- ジョブのモニタリング: 大規模なエクスポートのジョブステータスを定期的にチェックする

- エラー処理: 失敗したジョブに適切な再試行ロジックを実装する
- テンポラルフィルターの認識: PDex エクスポートの場合は、リソースタイプを選択するときに 5 年間のテンポラルフィルターを検討してください。
- 財務データの削除: 財務情報のないクレームデータ_includeEOB2xWoFinancial=trueが必要な場合に使用します。
- プロファイル管理: リソースに適切なプロファイル宣言があることを確認し、取り込み前にターゲットプロファイルと照合して検証し、プロファイルのバージョニングを使用してエクスポート動作を制御します。

制限事項

- patient パラメータでは最大 500 人の患者を指定できます
- エクスポートはグループレベルのオペレーションのみに制限されます
- は、エクスポートタイプごとに事前定義されたリソースタイプのセットのみをサポートします。
- 出力は常に NDJSON 形式です
- PDex のエクスポートは、5 年間の臨床データとクレームデータに制限されます。
- 財務データ変換は CARIN BB 2.x ExplanationOfBenefit プロファイルにのみ適用されます

その他のリソース

- Da Vinci メンバー属性リスト IG
- Da Vinci Payer Data Exchange IG
- CARIN コンシューマー向け支払者データ交換 IG
- US Core 実装ガイド
- FHIR バルクデータアクセス仕様

を使用した臨床ドキュメントの生成 **$document**

AWS HealthLake は Composition リソースの $documentオペレーションをサポートするようになりました。これにより、Composition と参照されるすべてのリソースを 1 つのまとまりのあるパッケージにバンドルすることで、完全な臨床ドキュメントを生成できます。このオペレーションは、以下を必要とするヘルスケアアプリケーションにとって不可欠です。

- 標準化された臨床文書を作成する

- 完全な患者レコードを交換する
- 包括的な臨床ドキュメントを保存する
- 関連するすべてのコンテキストを含むレポートを生成する

使用方法

$document オペレーションは、GET メソッドと POST メソッドの両方を使用して Composition リソースで呼び出すことができます。

サポートされているオペレーション

```
GET/POST [base]/Composition/[id]/$document
```

サポートされているパラメータ

HealthLake は、次の FHIR $documentパラメータをサポートしています。

パラメータ	タイプ	[Required] (必須)	デフォルト	説明
persist	boolean	不可	false	サーバーが生成されたドキュメントバンドルを保存するかどうかを示すブール値

例

GET リクエスト

```
GET [base]/Composition/180f219f-97a8-486d-99d9-ed631fe4fc57/$document?persist=true
```

パラメータを使用した POST リクエスト

```
POST [base]/Composition/180f219f-97a8-486d-99d9-ed631fe4fc57/$document
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
```

```
  "parameter": [
    {
      "name": "persist",
      "valueBoolean": true
    }
  ]
}
```

レスポンス例

オペレーションは、コンポジションと参照されるすべてのリソースを含む「ドキュメント」タイプの Bundle リソースを返します。

```
{
  "resourceType": "Bundle",
  "id": "180f219f-97a8-486d-99d9-ed631fe4fc57",
  "type": "document",
  "identifier": {
    "system": "urn:ietf:rfc:3986",
    "value": "urn:uuid:0c3151bd-1cbf-4d64-b04d-cd9187a4c6e0"
  },
  "timestamp": "2024-06-21T15:30:00Z",
  "entry": [
    {
      "fullUrl": "http://example.org/fhir/Composition/180f219f-97a8-486d-99d9-
ed631fe4fc57",
      "resource": {
        "resourceType": "Composition",
        "id": "180f219f-97a8-486d-99d9-ed631fe4fc57",
        "status": "final",
        "type": {
          "coding": [
            {
              "system": "http://loinc.org",
              "code": "34133-9",
              "display": "Summary of Episode Note"
            }
          ]
        },
        "subject": {
          "reference": "Patient/example"
        },
        "section": [
```

```
        {
          "title": "Allergies",
          "entry": [
            {
              "reference": "AllergyIntolerance/123"
            }
          ]
        }
      ]
    }
  },
  {
    "fullUrl": "http://example.org/fhir/Patient/example",
    "resource": {
      "resourceType": "Patient",
      "id": "example",
      "name": [
        {
          "family": "Smith",
          "given": ["John"]
        }
      ]
    }
  },
  {
    "fullUrl": "http://example.org/fhir/AllergyIntolerance/123",
    "resource": {
      "resourceType": "AllergyIntolerance",
      "id": "123",
      "patient": {
        "reference": "Patient/example"
      },
      "code": {
        "coding": [
          {
            "system": "http://snomed.info/sct",
            "code": "418689008",
            "display": "Allergy to penicillin"
          }
        ]
      }
    }
  }
]
```

```
}
```

行動

$document オペレーション:

1. 指定されたコンポジションリソースをドキュメントの基盤として使用します。
2. コンポジションによって直接参照されるすべてのリソースを識別して取得します
3. コンポジションと参照されるすべてのリソースを「document」タイプのバンドルにパッケージ化します
4. 永続パラメータが true に設定されている場合、生成されたドキュメントバンドルをデータストアに保存します。
5. 包括的なドキュメント生成のために コンポジションによって間接的に参照されるリソースを識別して取得します

$document オペレーションは現在、次の形式のリソースリファレンスの取得をサポートしています。

1. ```
   GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Resource/id
   ```
2. リソース/ID

コンポジションリソース内のサポートされていないリソース参照は、生成されたドキュメントから除外されます。

エラー処理

オペレーションは、次のエラー条件を処理します。

- 400 不正なリクエスト: 無効な$documentオペレーション (非準拠リクエスト) 、または永続化が true に設定されている場合に参照が除外されたために結果のドキュメントが FHIR 検証に失敗した場合
- 404 Not Found: コンポジションリソースが見つかりません

$document オペレーション仕様の詳細については、FHIR R4 コンポジション$documentドキュメントを参照してください。
   ```

を使用してリソースを完全に削除する $erase

AWS HealthLake は $eraseオペレーションをサポートし、特定のリソースとその履歴バージョンを完全に削除できます。このオペレーションは、以下が必要な場合に特に役立ちます。

- 個々のリソースを完全に削除する
- 特定のバージョン履歴を削除する
- 個々のリソースライフサイクルを管理する
- 特定のデータ削除要件に準拠する

使用方法

$erase オペレーションは 2 つのレベルで呼び出すことができます。

リソースインスタンスレベル

```
POST [base]/[ResourceType]/[ID]/$erase?deleteAuditEvent=true
```

バージョン固有のレベル

```
POST [base]/[ResourceType]/[ID]/_history/[VersionID]/$erase
```

パラメータ

パラメータ	タイプ	[Required] (必須)	デフォルト	説明
deleteAuditEvent	boolean	不可	false	true の場合、関連する監査イベントを削除します

例

リクエストの例

```
POST [base]/Patient/example-patient/$erase
```

レスポンスの例

```
{
      "jobId": "5df47e2f51ff3c731847678cb8cad48e",
      "jobStatus": "SUBMITTED"
    }
```

ジョブのステータス

消去ジョブのステータスを確認するには:

```
GET [base]/$erase/[jobId]
```

オペレーションはジョブステータス情報を返します。

```
{
      "datastoreId": "36622996b1fcecb7e12ee2ee085308d3",
      "jobId": "5df47e2f51ff3c731847678cb8cad48e",
      "status": "COMPLETED",
      "submittedTime": "2025-10-30T16:39:24.160Z"
    }
```

行動

$erase オペレーション:

1. 非同期的に処理してデータの整合性を確保する
2. ACID トランザクションを維持します
3. ジョブステータスの追跡を提供します
4. 指定されたリソースとそのバージョンを完全に削除します
5. 削除アクティビティの包括的な監査ログ記録が含まれます
6. 監査イベントの選択的な削除をサポート

監査ログ記録

$erase オペレーションは、ユーザー ID、タイムスタンプ、リソースの詳細とともに DeleteResource としてログに記録します。

制限事項

- $erased リソースは検索結果や_historyクエリに表示されません。
- 消去されるリソースは、処理中に一時的にアクセスできない場合があります。
- リソースが完全に削除されるとすぐにストレージ計測が調整されます

を使用した患者データの取得 **Patient/$everything**

Patient/$everything オペレーションは、FHIR Patientリソースとその に関連する他のリソースをクエリするために使用されますPatient。オペレーションを使用して、レコード全体へのアクセスを患者に提供したり、プロバイダーが患者に関連する一括データダウンロードを実行したりできます。HealthLake は、特定の患者の Patient/$everythingをサポートしますid。

Patient/$everything は、以下の例に示すように呼び出すことができる FHIR REST API オペレーションです。

GET request

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything
```

> **Note**
>
> 応答するリソースは、リソースタイプとリソース でソートされますid。
> レスポンスには常に が入力されますBundle.total。

Patient/$everything 個のパラメータ

HealthLake は、次のクエリパラメータをサポートしています。

パラメータ	Details
開始	指定された開始日より後にすべてのPatientデータを取得します。
end	指定された終了日より前にすべてのPatientデータを取得します。
since	指定された日付以降に更新されたすべてのPatientデータを取得します。

パラメータ	Details
_type	特定のリソースタイプのPatientデータを取得します。
_count	Patient データを取得し、ページサイズを指定します。

Example- 指定された開始日以降にすべての患者データを取得する

Patient/$everything は、startフィルターを使用して、特定の日付より後のデータのみをクエリできます。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?start=2024-03-15T00:00:00.000Z
```

Example- 指定された終了日より前にすべてのPatientデータを取得する

患者 $すべての は、endフィルターを使用して、特定の日付より前のデータのみをクエリできます。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?end=2024-03-15T00:00:00.000Z
```

Example- 指定された日付以降に更新されたすべてのPatientデータを取得する

Patient/$everything は、sinceフィルターを使用して、特定の日付以降に更新されたデータのみをクエリできます。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?since=2024-03-15T00:00:00.000Z
```

Example- 特定のリソースタイプのPatientデータを取得する

患者 $Everything は_typeフィルターを使用して、レスポンスに含める特定のリソースタイプを指定できます。複数のリソースタイプをカンマ区切りリストで指定できます。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?_type=Observation,Condition
```

Example- Patient データを取得し、ページサイズを指定する

患者 $すべての は、 _count を使用してページサイズを設定できます。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?_count=15
```

Patient/$everything start および end 属性

HealthLake は、 Patient/ $everythingstartおよび endクエリパラメータに対して次のリソース属性をサポートしています。

[リソース]	リソースエレメント
アカウント	Account.servicePeriod.start
AdverseEvent	AdverseEvent.date
AllergyIntolerance	AllergyIntolerance.recordedDate
予約	Appointment.start
AppointmentResponse	AppointmentResponse.start
AuditEvent	AuditEvent.period.start
Basic	Basic.created
BodyStructure	NO_DATE
CarePlan	CarePlan.period.start
CareTeam	CareTeam.period.start
ChargeItem	ChargeItem.occurrenceDateTime,ChargeItem.occurrencePeriod.start,ChargeItem.occurrenceTiming.event
Claim	Claim.billablePeriod.start

[リソース]	リソースエレメント
ClaimResponse	ClaimResponse.created
ClinicalImpression	ClinicalImpression.date
コミュニケーション	Communication.sent
CommunicationRequest	CommunicationRequest.occurrenceDateTime,CommunicationRequest.occurrencePeriod.start
コンポジション	Composition.date
Condition	Condition.recordedDate
同意	consent.dateTime
カバレッジ	Coverage.period.start
CoverageEligibilityRequest	CoverageEligibilityRequest.created
CoverageEligibilityResponse	CoverageEligibilityResponse.created
DetectedIssue	DetectedIssue.identified
DeviceRequest	DeviceRequest.authoredOn
DeviceUseStatement	DeviceUseStatement.recordedOn

[リソース]	リソースエレメント
DiagnosticReport	DiagnosticReport.effective
DocumentManifest	DocumentManifest.created
DocumentReference	DocumentReference.context.period.start
エンカウンター	Encounter.period.start
EnrollmentRequest	EnrollmentRequest.created
EpisodeOfCare	EpisodeOfCare.period.start
ExplanationOfBenefit	ExplanationOfBenefit.billablePeriod.start
FamilyMemberHistory	NO_DATE
フラグ	Flag.period.start
目標	Goal.statusDate
Group	NO_DATE
ImagingStudy	ImagingStudy.started
イミュナイゼーション	Immunization.recorded

[リソース]	リソースエレメント
ImmunizationEvaluation	ImmunizationEvaluation.date
ImmunizationRecommendation	ImmunizationRecommendation.date
Invoice	Invoice.date
リスト	List.date
MeasureReport	MeasureReport.period.start
メディア	Media.issued
MedicationAdministration	MedicationAdministration.effective
MedicationDispense	MedicationDispense.whenPrepared
MedicationRequest	MedicationRequest.authoredOn
MedicationStatement	MedicationStatement.dateAsserted
MolecularSequence	NO_DATE
NutritionOrder	NutritionOrder.dateTime
監視結果	観測効果

[リソース]	リソースエレメント
患者	NO_DATE
個人	NO_DATE
手順	Procedure.perform
プロベナンス	Provenance.occurredPeriod.start,Provenance.occurredDateTime
QuestionnaireResponse	QuestionnaireResponse.authored
RelatedPerson	NO_DATE
RequestGroup	RequestGroup.authoredOn
ResearchSubject	ResearchSubject.period
RiskAssessment	RiskAssessment.occurrenceDateTime, RiskAssessment.occurrencePeriod.start
スケジュール	Schedule.planningHorizon
ServiceRequest	ServiceRequest.authoredOn
標本	".receivedTime
SupplyDelivery	SupplyDelivery.occurrenceDateTime, SupplyDelivery.occurrencePeriod.start, SupplyDelivery.occurrenceTiming.event
SupplyRequest	SupplyRequest.authoredOn

[リソース]	リソースエレメント
VisionPrescription	VisionPrescription.dateWritten

を使用した ValueSet コードの取得 **$expand**

AWS HealthLake は、顧客として取り込まれた ValueSets の $expandオペレーションをサポートするようになりました。これにより、それらの ValueSet リソースに含まれるコードの完全なリストを取得できます (複数可) 。このオペレーションは、以下が必要な場合に特に役立ちます。

- 検証目的で可能なすべてのコードを取得する
- ユーザーインターフェイスで使用可能なオプションを表示する
- 特定の用語コンテキスト内で包括的なコード検索を実行する

使用方法

$expand オペレーションは、GET メソッドと POST メソッドの両方を使用して ValueSet リソースで呼び出すことができます。

サポートされているオペレーション

```
GET/POST [base]/ValueSet/[id]/$expand
GET [base]/ValueSet/$expand?url=http://example.com
POST [base]/ValueSet/$expand
```

サポートされているパラメータ

HealthLake は、FHIR R4 $expandパラメータのサブセットをサポートしています。

パラメータ	タイプ	必須	説明
url	uri	不可	拡張する ValueSet の正規 URL
id	id	不可	拡張する ValueSet リソース ID (GET または POST オペレーションの場合)

パラメータ	タイプ	必須	説明
filter	string	不可	コード拡張結果をフィルタリングする
count	integer	不可	返されるコードの数
offset	integer	不可	戻る前にスキップする一致するコードの数。フィルタリング後、元の ValueSet の完全なフィルタリングされていないコンテンツではなく、一致するコードにのみ適用されます。

例

ID による GET リクエスト

```
GET [base]/ValueSet/example-valueset/$expand
```

フィルターを使用した URL による GET リクエスト

```
GET [base]/ValueSet/$expand?url=http://example.com/ValueSet/my-
valueset&filter=male&count=5
```

パラメータを含む POST リクエスト (ID 別)

```
POST [base]/ValueSet/example-valueset/$expand
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "count",
      "valueInteger": 10
    },
    {
      "name": "filter",
      "valueString": "admin"
    }
```

```
  ]
}
```

パラメータを含む POST リクエスト (URL による)

```
POST [base]/ValueSet/$expand
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "url",
      "valueUri": "http://hl7.org/fhir/ValueSet/administrative-gender"
    },
    {
      "name": "count",
      "valueInteger": 10
    }
  ]
}
```

レスポンス例

オペレーションは、展開されたコードを含む expansion要素を持つ ValueSet リソースを返します。

```
{
  "resourceType": "ValueSet",
  "id": "administrative-gender",
  "status": "active",
  "expansion": {
    "identifier": "urn:uuid:12345678-1234-1234-1234-123456789abc",
    "timestamp": "2024-01-15T10:30:00Z",
    "total": 4,
    "parameter": [
      {
        "name": "count",
        "valueInteger": 10
      }
    ],
    "contains": [
```

```
      {
        "system": "http://hl7.org/fhir/administrative-gender",
        "code": "male",
        "display": "Male"
      },
      {
        "system": "http://hl7.org/fhir/administrative-gender",
        "code": "female",
        "display": "Female"
      },
      {
        "system": "http://hl7.org/fhir/administrative-gender",
        "code": "other",
        "display": "Other"
      },
      {
        "system": "http://hl7.org/fhir/administrative-gender",
        "code": "unknown",
        "display": "Unknown"
      }
    ]
  }
}
```

レスポンスは以下のとおりです。

- expansion.total: 展開された ValueSet のコードの合計数
- expansion.contains: システム、コード、および表示値を含む拡張コードの配列
- expansion.parameter: 拡張リクエストで使用されるパラメータ

$expand オペレーション仕様の詳細については、FHIR R4 ValueSet $expand ドキュメントを参照してください。

FHIR を使用した HealthLake データのエクスポート **$export**

FHIR $export オペレーションを使用して、HealthLake データストアからデータを一括エクスポートできます。HealthLake は、 POSTおよび GETリクエスト$exportを使用した FHIR をサポートしています。でエクスポートリクエストを行うにはPOST、必要なアクセス許可を持つ IAM ユーザー、グループ、またはロールがあり、リクエスト$exportの一部として を指定し、リクエスト本文に必要なパラメータを含める必要があります。

> **Note**
>
> FHIR を使用して行われたすべての HealthLake エクスポートリクエスト`$export`は`ndjson`形式で返され、Amazon S3 バケットにエクスポートされます。各 Amazon S3 オブジェクトには 1 つの FHIR リソースタイプのみが含まれます。
> AWS アカウントサービスクォータごとにエクスポートリクエストをキューに入れることができます。詳細については、「Service Quotas」を参照してください。

HealthLake は、次の 3 種類の一括エクスポートエンドポイントリクエストをサポートしています。

HealthLake バルク**`$export`**タイプ

エクスポートタイプ	説明	構文
システム	HealthLake FHIR サーバーからすべてのデータをエクスポートします。	`POST https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/$export`

すべての患者	患者リソースタイプに関連付けられたリソースタイプを含む、すべての患者に関連するすべてのデータをエクスポートします。	`POST https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/Patient/$export`
		
		`GET https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/Patient/$export`

患者のグループ	グループ ID で指定された患者のグループに関連するすべてのデータをエクスポートします。	`POST https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/Group/ id/$export`
		
		`GET https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/Group/ id/$export`

[開始する前に]

HealthLake 用の FHIR REST API を使用してエクスポートリクエストを行うには、次の要件を満たします。

- エクスポートリクエストを行うために必要なアクセス許可を持つユーザー、グループ、またはロールを設定しておく必要があります。詳細については$export リクエストの承認を参照してください。
- データをエクスポートする Amazon S3 バケットへの HealthLake アクセスを許可するサービスロールを作成しておく必要があります。サービスロールは、サービスプリンシパルとして HealthLake も指定する必要があります。アクセス許可の設定の詳細については、「」を参照してくださいエクスポートジョブのアクセス許可の設定。

$export リクエストの承認

FHIR REST API を使用してエクスポートリクエストを成功させるには、IAM または OAuth2.0. サービスロールも必要です。

IAM を使用したリクエストの承認

$export リクエストを行うときは、ユーザー、グループ、またはロールに IAM アクションがポリシーに含まれている必要があります。詳細については、「エクスポートジョブのアクセス許可の設定」を参照してください。

SMART on FHIR (OAuth 2.0) を使用したリクエストの承認

SMART on FHIR 対応 HealthLake データストアで$exportリクエストを行うときは、適切なスコープを割り当てる必要があります。詳細については、「HealthLake の FHIR リソーススコープに関する SMART」を参照してください。

> **Note**
>
> GET リクエスト$exportを含む FHIR では、ファイルのエクスポートと取得をリクエストするために、同じ認証方法またはベアラートークン (SMART on FHIR の場合) が必要です。$export で FHIR を使用してエクスポートされたファイルはGET、48 時間ダウンロードできます。

$export リクエストを行う

このセクションでは、FHIR REST API を使用してエクスポートリクエストを行うときに必要な手順について説明します。

AWS アカウントで誤って課金されないように、$export構文を指定せずにリクエストを実行してPOSTリクエストをテストすることをお勧めします。

リクエストを行うには、以下を実行する必要があります。

1. サポートされているエンドポイント$exportのPOSTリクエスト URL で を指定します。
2. 必要なヘッダーパラメータを指定します。
3. 必要なパラメータを定義するリクエスト本文を指定します。

ステップ 1: サポートされているエンドポイント**$export**の**POST**リクエスト URL で を指定します。

HealthLake は、3 種類の一括エクスポートエンドポイントリクエストをサポートしています。一括エクスポートリクエストを行うには、サポートされている 3 つのエンドポイントのいずれかに対して POSTベースのリクエストを行う必要があります。次の例は、リクエスト URL $exportで を指定する場所を示しています。

- `POST https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/$export`
- `POST https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/$export`
- `POST https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Group/id/$export`

POST リクエスト文字列では、以下のサポートされている検索パラメータを使用できます。

サポートされている検索パラメータ

HealthLake は、一括エクスポートリクエストで次の検索修飾子をサポートしています。

次の例には、リクエストを送信する前にエンコードする必要がある特殊文字が含まれています。

名前	必須?	説明	例
_outputFormat	いいえ	リクエストされたバルクデータファイルを生成する形式。使用できる値は application/fhir+ndjson 、application/ndjson 、 ですndjson。	
_type	いいえ	エクスポートジョブに含めるカンマ区切りの FHIR リソースタイプの文字列。すべてのリソースをエクスポートするとコストがかかる_type可能性があるため、 を含めることをお勧めします。	&_type=MedicationStatement,Observation
_since	いいえ	日付タイムスタンプ以降に変更されたリソースタイプ。リソースタイプに最終更新時刻がない場合、レスポンスに含まれます。	&_since=2024-05-09T00%3A00%3A00Z
_until	いいえ	日付タイムスタンプ以前に変更されたリソースタイプ。と組み合わせて使用_sinceして、エクスポートする特定の	&_until=2024-12-31T23%3A59%3A59Z

名前	必須?	説明	例
		時間範囲を定義します。リソースタイプに最終更新時刻がない場合、レスポンスから除外されます。	
_security	いいえ	エクスポートされたリソースをmeta.security コーディング値でフィルタリングします。system|code 形式を使用します。複数の値を指定する場合、リソースはすべての値 (AND セマンティクス) と一致する必要があります。system| (証跡パイプ、コードなし) を使用して、特定のシステムのコードと照合します。	&_security=https://myorg.com/tenant%7Cclinic-A

名前	必須?	説明	例
_tag	いいえ	エクスポートされたリソースをmeta.tagコーディング値でフィルタリングします。と同じsystem|code 形式と AND セマンティクスを使用します_security 。_security との両方を指定する場合、リソース_tagは両方のフィルターと一致する必要があります。	&_tag=https://myorg.com/dept%7Ccardiology

ステップ 2: 必要なヘッダーパラメータを指定する

FHIR REST API を使用してエクスポートリクエストを行うには、次のヘッダーパラメータを指定する必要があります。

- Content-Type : application/fhir+json
- 優先: respond-async

次に、リクエスト本文で必要な要素を指定する必要があります。

ステップ 3: で必要なパラメータを定義するリクエスト本文を指定します。

エクスポートリクエストには、 JSON形式の本文も必要です。本文には、次のパラメータを含めることができます。

キー	必須?	説明	値
DataAccessRoleArn	はい	HealthLake サービスロールの ARN。	arn:aws:iam:: **444455556**

キー	必須?	説明	値
		使用するサービスロールは、サービスプリンシパルとして HealthLake を指定する必要があります。	**666** :role/**your-healthlake-service-role**
JobName	いいえ	エクスポートリクエストの名前。	**your-export-job-name**
S3Uri	はい	OutputDataConfig キーの一部。エクスポートされたデータがダウンロードされる送信先バケットの S3 URI。	s3://amzn-s3-demo-bucket/ **EXPORT-JOB** /
KmsKeyId	はい	OutputDataConfig キーの一部。Amazon S3 バケットの保護に使用される AWS KMS キーの ARN。	arn:aws:kms: **region-of-bucket**:**123456789012** :key/**1234abcd-12ab-34cd-56ef-1234567890ab**

Example FHIR REST API を使用して行われたエクスポートリクエストの本文

FHIR REST API を使用してエクスポートリクエストを行うには、次に示すように本文を指定する必要があります。

```
{
  "DataAccessRoleArn": "arn:aws:iam::444455556666:role/your-healthlake-service-role",
  "JobName": "your-export-job",
  "OutputDataConfig": {
    "S3Configuration": {
      "S3Uri": "s3://amzn-s3-demo-bucket/EXPORT-JOB",
      "KmsKeyId": "arn:aws:kms:region-of-
bucket:444455556666:key/1234abcd-12ab-34cd-56ef-1234567890ab"
```

```
    }
  }
}
```

リクエストが成功すると、次のレスポンスを受け取ります。

レスポンスヘッダー

```
content-location: https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
export/your-export-request-job-id
```

レスポンス本文

```
{
  "datastoreId": "your-data-store-id",
  "jobStatus": "SUBMITTED",
  "jobId": "your-export-request-job-id"
}
```

エクスポートリクエストの管理

エクスポートリクエストが成功したら、 を使用して現在のエクスポートリクエストのステータスを$export記述し、現在のエクスポートリクエストをキャンセル$exportして、リクエストを管理できます。

REST API を使用してエクスポートリクエストをキャンセルすると、キャンセルリクエストを送信した時点までにエクスポートされたデータの一部に対してのみ課金されます。

以下のトピックでは、現在のエクスポートリクエストのステータスを取得またはキャンセルする方法について説明します。

エクスポートリクエストのキャンセル

エクスポートリクエストをキャンセルするには、DELETEリクエストを行い、リクエスト URL にジョブ ID を指定します。

```
DELETE https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/export/your-
export-request-job-id
```

リクエストが成功すると、以下が表示されます。

```
{
```

```
  "exportJobProperties": {
    "jobId": "your-original-export-request-job-id",
    "jobStatus": "CANCEL_SUBMITTED",
    "datastoreId": "your-data-store-id"
  }
}
```

リクエストが成功しない場合、次の情報が表示されます。

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "not-supported",
      "diagnostics": "Interaction not supported."
    }
  ]
}
```

エクスポートリクエストの説明

エクスポートリクエストのステータスを取得するには、 exportと を使用してGETリクエストを行いますexport-request-job-id。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/export/your-export-request-id
```

JSON レスポンスには ExportJobProperties オブジェクトが含まれます。これには、次のキーと値のペアが含まれる場合があります。

名前	必須?	説明	値
DataAccessRoleArn	いいえ	HealthLake サービスロールの ARN。使用するサービスロールは、サービスプリンシパルとして HealthLake を指定する必要があります。	arn:aws:iam:: 444455556666 :role/your-healthlake-service-role

名前	必須?	説明	値
SubmitTime	いいえ	エクスポートジョブが送信された日付。	`Apr 21, 2023 5:58:02`
EndTime	いいえ	エクスポートジョブが完了した時刻。	`Apr 21, 2023 6:00:08 PM`
JobName	いいえ	エクスポートリクエストの名前。	**`your-export-job-name`**
JobStatus	いいえ		次の値を指定できます。 `SUBMITTED | IN_PROGRESS | COMPLETED_WITH_ERRORS | COMPLETED | FAILED`
S3Uri	はい	OutputDataConfig オブジェクトの一部。エクスポートされたデータがダウンロードされる送信先バケットの Amazon S3 URI。	`s3://amzn-s3-demo-bucket/` **`EXPORT-JOB /`**
KmsKeyId	はい	OutputDataConfig オブジェクトの一部。Amazon S3 バケットの保護に使用される AWS KMS キーの ARN。	`arn:aws:kms:` **`region-of-bucket:123456789012`** `:key/`**`1234abcd-12ab-34cd-56ef-1234567890ab`**

Example: FHIR REST API を使用して行われた describe エクスポートリクエストの本文

成功すると、次の JSON レスポンスが表示されます。

```
{
  "exportJobProperties": {
    "jobId": "your-export-request-id",
    "JobName": "your-export-job",
    "jobStatus": "SUBMITTED",
    "submitTime": "Apr 21, 2023 5:58:02 PM",
    "endTime": "Apr 21, 2023 6:00:08 PM",
    "datastoreId": "your-data-store-id",
    "outputDataConfig": {
      "s3Configuration": {
        "S3Uri": "s3://amzn-s3-demo-bucket/EXPORT-JOB",
        "KmsKeyId": "arn:aws:kms:region-of-
bucket:444455556666:key/1234abcd-12ab-34cd-56ef-1234567890ab""
      }
    },
    "DataAccessRoleArn": "arn:aws:iam::444455556666:role/your-healthlake-service-role",
  }
}
```

HealthLake の FHIR **$inquire**オペレーション

$inquire オペレーションを使用すると、以前に送信された事前承認リクエストのステータスを確認できます。このオペレーションでは、Da Vinci 事前認可サポート (PAS) 実装ガイドを実装し、現在の認可決定を取得するための標準化された FHIR ベースのワークフローを提供します。

仕組み

- 問い合わせの送信: 確認したいクレームとサポート情報を含む FHIR バンドルを送信する
- 検索: HealthLake は、データストア内の対応する ClaimResponse を検索します。
- 取得: 最新の認可ステータスが取得されます
- 応答: 現在の認可ステータス (キューに入れられた、承認された、拒否されたなど) の即時応答を受け取ります。

> **Note**
>
> `$inquire` は、既存の認可ステータスを取得する読み取り専用オペレーションです。データストア内のリソースは変更または更新されません。

API エンドポイント

```
POST /datastore/{datastoreId}/r4/Claim/$inquire
Content-Type: application/fhir+json
```

リクエスト構造

バンドルの要件

リクエストは、以下を含む FHIR Bundle リソースである必要があります。

- Bundle.type: である必要があります `"collection"`
- Bundle.entry: 以下を含むクレームリソースを 1 つだけ含める必要があります。
  - `use = "preauthorization"`
  - `status = "active"`
- 参照リソース: クレームによって参照されるすべてのリソースをバンドルに含める必要があります

> **Query-by-Example**
>
> 入力バンドルのリソースは検索テンプレートとして機能します。HealthLake は、提供された情報を使用して、対応する ClaimResponse を見つけます。

必要なリソース

[リソース]	カーディナリティ	プロファイル	説明
クレーム	1	PAS クレームの照会	照会する以前の認可
患者	1	PAS 受取人患者	患者の人口統計情報

[リソース]	カーディナリティ	プロファイル	説明
組織 (保険者)	1	PAS 保険会社組織	保険会社
組織 (プロバイダー)	1	PAS リクエスト組織	リクエストを送信した医療プロバイダー

重要な検索条件

HealthLake は、以下を使用して ClaimResponse を検索します。

- クレームからの患者リファレンス
- クレームからの保険者リファレンス
- クレームからのプロバイダーリファレンス
- クレームから作成日 (時間フィルターとして)

> **患者固有のクエリのみ**
>
> すべての問い合わせは、特定の患者に関連付ける必要があります。患者識別を使用しないシステム全体のクエリは許可されません。

リクエスト例

```
POST /datastore/example-datastore/r4/Claim/$inquire
Content-Type: application/fhir+json
Authorization: Bearer <your-token>

{
  "resourceType": "Bundle",
  "id": "PASClaimInquiryBundleExample",
  "meta": {
    "profile": ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-pas-
inquiry-request-bundle"]
  },
  "identifier": {
    "system": "http://example.org/SUBMITTER_TRANSACTION_IDENTIFIER",
```

```
    "value": "5269368"
  },
  "type": "collection",
  "timestamp": "2005-05-02T14:30:00+05:00",
  "entry": [
    {
      "fullUrl": "http://example.org/fhir/Claim/MedicalServicesAuthorizationExample",
      "resource": {
        "resourceType": "Claim",
        "id": "MedicalServicesAuthorizationExample",
        "meta": {
          "profile": ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
claim-inquiry"]
        },
        "status": "active",
        "type": {
          "coding": [{
            "system": "http://terminology.hl7.org/CodeSystem/claim-type",
            "code": "professional"
          }]
        },
        "use": "preauthorization",
        "patient": {
          "reference": "Patient/SubscriberExample"
        },
        "created": "2005-05-02T11:01:00+05:00",
        "insurer": {
          "reference": "Organization/InsurerExample"
        },
        "provider": {
          "reference": "Organization/UMOExample"
        }
      }
    },
    {
      "fullUrl": "http://example.org/fhir/Patient/SubscriberExample",
      "resource": {
        "resourceType": "Patient",
        "id": "SubscriberExample",
        "meta": {
          "profile": ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
beneficiary"]
        },
        "name": [{
```

```
          "family": "SMITH",
          "given": ["JOE"]
        }],
        "gender": "male"
      }
    },
    {
      "fullUrl": "http://example.org/fhir/Organization/UMOExample",
      "resource": {
        "resourceType": "Organization",
        "id": "UMOExample",
        "meta": {
          "profile": ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
requestor"]
        },
        "name": "Provider Organization"
      }
    },
    {
      "fullUrl": "http://example.org/fhir/Organization/InsurerExample",
      "resource": {
        "resourceType": "Organization",
        "id": "InsurerExample",
        "meta": {
          "profile": ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
insurer"]
        },
        "name": "Insurance Company"
      }
    }
  ]
}
```

レスポンスの形式

成功レスポンス (200 OK)

PAS Inquiry Response Bundle には、以下が含まれています。

- 現在の認可ステータスの ClaimResponse。検索条件に一致する場合は複数の ClaimResponse
- リクエストのすべての元のリソース (エコーバック)
- レスポンスがアセンブルされた時刻のタイムスタンプ

考えられる ClaimResponse の結果

結果	説明
queued	承認リクエストはまだレビュー保留中です
complete	認可の決定が行われました (承認/拒否disposition を確認してください)
error	処理中にエラーが発生しました
partial	部分的な認可が付与されました


```
{
  "resourceType": "Bundle",
  "identifier": {
        "system": "http://example.org/SUBMITTER_TRANSACTION_IDENTIFIER",
        "value": "5269367"
  },
  "type": "collection",
  "timestamp": "2005-05-02T14:30:15+05:00",
  "entry": [
    {
      "fullUrl": "http://example.org/fhir/ClaimResponse/InquiryResponseExample",
      "resource": {
        "resourceType": "ClaimResponse",
        "id": "InquiryResponseExample",
        "meta": {
          "profile": ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
claimresponse-inquiry"]
        },
        "status": "active",
        "type": {
          "coding": [{
            "system": "http://terminology.hl7.org/CodeSystem/claim-type",
            "code": "professional"
          }]
        },
        "use": "preauthorization",
        "patient": {
          "reference": "Patient/SubscriberExample"
```

```
        },
        "created": "2005-05-02T11:05:00+05:00",
        "insurer": {
          "reference": "Organization/InsurerExample"
        },
        "request": {
          "reference": "Claim/MedicalServicesAuthorizationExample"
        },
        "outcome": "complete",
        "disposition": "Approved",
        "preAuthRef": "AUTH12345"
      }
    },
    {
      "fullUrl": "http://example.org/fhir/Claim/MedicalServicesAuthorizationExample",
      "resource": {
        "resourceType": "Claim",
        "id": "MedicalServicesAuthorizationExample",
        "meta": {
          "profile": ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
claim-inquiry"]
        },
        "status": "active",
        "type": {
          "coding": [{
            "system": "http://terminology.hl7.org/CodeSystem/claim-type",
            "code": "professional"
          }]
        },
        "use": "preauthorization",
        "patient": {
          "reference": "Patient/SubscriberExample"
        },
        "created": "2005-05-02T11:01:00+05:00",
        "insurer": {
          "reference": "Organization/InsurerExample"
        },
        "provider": {
          "reference": "Organization/UMOExample"
        }
      }
    },
    {
      "fullUrl": "http://example.org/fhir/Patient/SubscriberExample",
```

```
      "resource": {
        "resourceType": "Patient",
        "id": "SubscriberExample",
        "meta": {
          "profile": ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
beneficiary"]
        },
        "name": [{
          "family": "SMITH",
          "given": ["JOE"]
        }],
        "gender": "male"
      }
    },
    {
      "fullUrl": "http://example.org/fhir/Organization/UMOExample",
      "resource": {
        "resourceType": "Organization",
        "id": "UMOExample",
        "meta": {
          "profile": ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
requestor"]
        },
        "name": "Provider Organization"
      }
    },
    {
      "fullUrl": "http://example.org/fhir/Organization/InsurerExample",
      "resource": {
        "resourceType": "Organization",
        "id": "InsurerExample",
        "meta": {
          "profile": ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
insurer"]
        },
        "name": "Insurance Company"
      }
    }
  ]
}
```

エラーレスポンス

400 Bad Request

リクエスト形式が無効であるか、検証が失敗した場合に返されます。

```
{
    "resourceType": "OperationOutcome",
    "issue": [
        {
            "severity": "error",
            "code": "required",
            "diagnostics": "Reference 'Patient/SubscriberExample' at path 'patient' for
 'CLAIM' resource not found(at Bundle.entry[0].resource)"
        }
    ]
}
```

401 Unauthorized

認証情報がないか無効である場合に返されます。

```
{
    "resourceType": "OperationOutcome",
    "issue": [
        {
            "severity": "error",
            "code": "forbidden",
            "diagnostics": "Invalid authorization header"
        }
    ]
}
```

403 Forbidden

認証されたユーザーに、リクエストされたリソースへのアクセス許可がない場合に返されます。

```
{
    "resourceType": "OperationOutcome",
    "issue": [
        {
            "severity": "error",
```

```
            "code": "exception",
            "diagnostics": "Insufficient SMART scope permissions."
        }
    ]
}
```

400 見つからない場合

問い合わせに一致する ClaimResponse が見つからない場合に返されます。

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "not-found",
    "diagnostics": "Resource not found. No ClaimResponse found from the input Claim
 that matches the specified Claim properties patient, insurer, provider, and created(at
 Bundle.entry[0].resource)"
  }]
}
```

415 サポートされていないメディアタイプ

Content-Type ヘッダーが application/fhir+json でない場合に返されます。

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "value",
    "diagnostics": "Incorrect MIME-type. Update request Content-Type header."
  }]
}
```

429 Too Many Requests

レート制限を超えたときに返されます。

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
```

```
    "code": "throttled",
    "diagnostics": "Rate limit exceeded. Please retry after some time."
  }]
}
```

検証ルール

HealthLake は、問い合わせに対して包括的な検証を実行します。

バンドルの検証

- PAS 照会リクエストバンドルプロファイルに準拠する必要があります
- Bundle.type は である必要があります "collection"
- クレームリソースを 1 つだけ含める必要があります
- 参照されるすべてのリソースをバンドルに含める必要があります

クレームの検証

- PAS クレーム照会プロファイルに準拠する必要があります
- Claim.use は である必要があります "preauthorization"
- Claim.status は である必要があります "active"
- 必須フィールド: patient、insurer、provider、 created

リソースの検証

- すべてのリソースは、それぞれの PAS 問い合わせプロファイルに準拠する必要があります
- 必要なサポートリソースが存在する必要があります (「」、「保険組織」、「プロバイダー組織」)
- クロスリファレンスは、バンドル内で有効で解決可能である必要があります

パフォーマンス仕様

メトリクス	の仕様
リソース数の制限	バンドルあたり 500 リソース
バンドルサイズ制限	最大 5 MB

必要なアクセス許可

$inquire オペレーションを使用するには、IAM ロールに次のものがあることを確認します。

- healthlake:InquirePreAuthClaim - オペレーションを呼び出すには

FHIR スコープでの SMART

最低限必要なスコープ:

- SMART v1: user/ClaimResponse.read
- SMART v2: user/ClaimResponse.s

重要な実装上の注意事項

検索動作

問い合わせを送信すると、HealthLake は以下を使用して ClaimResponse を検索します。

- 入力クレームからの患者リファレンス
- 入力クレームからの保険者リファレンス
- 入力クレームからのプロバイダーリファレンス
- 入力クレームから作成日 (タイムフィルターとして)

複数の一致: 複数の ClaimResponses が検索条件に一致する場合、HealthLake は一致するすべての結果を返します。最新のClaimResponse.createdステータスを特定するには、最新のタイムスタンプを使用する必要があります。

クレームの更新

同じ事前承認 (クレーム v1.1、v1.2、v1.3 など) に複数の更新を送信した場合、$inquireオペレーションは提供された検索条件に基づいて最新バージョンに関連付けられた ClaimResponse を取得します。

読み取り専用オペレーション

$inquire オペレーション:

- 既存の認可ステータスを取得します

- 最新の ClaimResponse を返します
- リソースを変更または更新しない
- 新しいリソースを作成しない
- 新しい認可処理をトリガーしない

ワークフローの例

一般的な事前認可照会ワークフロー

```
1. Provider submits PA request
   POST /Claim/$submit
   # Returns ClaimResponse with outcome="queued"

2. Payer reviews request (asynchronous)
   # Updates ClaimResponse status internally

3. Provider checks status
   POST /Claim/$inquire
   # Returns ClaimResponse with outcome="queued" (still pending)

4. Provider checks status again later
   POST /Claim/$inquire
   # Returns ClaimResponse with outcome="complete", disposition="Approved"
```

関連の オペレーション

- `Claim/$submit` - 新しい事前承認リクエストを送信するか、既存の承認リクエストを更新します。
- `Patient/$everything` - 事前認可コンテキストの包括的な患者データを取得する

を使用した概念の詳細の取得 **$lookup**

AWS HealthLake は CodeSystem リソースの `$lookup`オペレーションをサポートするようになりました。これにより、コードなどの識別情報を提供することで、コードシステム内の特定の概念に関する詳細を取得できます。このオペレーションは、以下が必要な場合に特に役立ちます。

- 特定の医療コードに関する詳細情報を取得する

- コードの意味とプロパティを検証する
- アクセス概念の定義と関係
- 正確な用語データで臨床上の意思決定をサポートする

使用方法

$lookup オペレーションは、GET メソッドと POST メソッドの両方を使用して CodeSystem リソースで呼び出すことができます。

サポートされているオペレーション

```
GET [base]/CodeSystem/$lookup?system=http://snomed.info/
sct&code=73211009&version=20230901
POST [base]/CodeSystem/$lookup
```

サポートされているパラメータ

HealthLake は、FHIR R4 $lookupパラメータのサブセットをサポートしています。

パラメータ	タイプ	必須	説明
code	コード	はい	検索する概念コード (SNOMED CT の「71620000」など)
system	uri	はい	コードシステムの正規 URL (「http://snomed.info/sct」など)
version	string	不可	コードシステムの特定のバージョン

例

GET リクエスト

```
GET [base]/CodeSystem/$lookup?system=http://snomed.info/
sct&code=71620000&version=2023-09
```

POST リクエスト

```
POST [base]/CodeSystem/$lookup
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "system",
      "valueUri": "http://snomed.info/sct"
    },
    {
      "name": "code",
      "valueCode": "71620000"
    },
    {
      "name": "version",
      "valueString": "2023-09"
    }
  ]
}
```

レスポンス例

オペレーションは、概念の詳細を含む Parameters リソースを返します。

```
{
    "resourceType": "Parameters",
    "parameter": [{
            "name": "name",
            "valueString": "SNOMED CT Fractures"
        },
        {
            "name": "version",
            "valueString": "2023-09"
        },
        {
            "name": "display",
            "valueString": "Fracture of femur"
        },
        {
```

```
            "name": "property",
            "part": [{
                    "name": "code",
                    "valueCode": "child"
                },
                {
                    "name": "value",
                    "valueCode": "263225007"
                },
                {
                    "name": "description",
                    "valueString": "Fracture of neck of femur"
                }
            ]
        },
        {
            "name": "property",
            "part": [{
                    "name": "code",
                    "valueCode": "child"
                },
                {
                    "name": "value",
                    "valueCode": "263227004"
                },
                {
                    "name": "description",
                    "valueString": "Fracture of shaft of femur"
                }
            ]
        }
    ]
}
```

レスポンスパラメータ

レスポンスには、使用可能な場合、次のパラメータが含まれます。

パラメータ	Type	説明
name	string	コードシステムの名前
version	string	コードシステムのバージョン

パラメータ	Type	説明
display	string	概念の表示名
designation	BackboneElement	この概念の追加表現。
property	BackboneElement	概念の追加プロパティ (定義、関係など)

行動

$lookup オペレーション:

1. 必要なパラメータを検証します (code および system)
2. データストアに保存されている指定されたコードシステム内の概念を検索します。
3. 表示名、指定、プロパティなど、詳細な概念情報を返します。
4. version パラメータが指定されている場合のバージョン固有のルックアップをサポート
5. HealthLake データストアに明示的に保存されているコードシステムでのみ動作します

エラー処理

オペレーションは、次のエラー条件を処理します。

- 400 不正なリクエスト: 無効な$lookupオペレーション (非準拠のリクエストまたは必須パラメータの欠落)
- 404 Not Found: コードシステムが見つからないか、指定されたコードシステムでコードが見つかりません

注意

このリリースでは、以下はサポートされていません。

- $lookup 外部用語サーバーを呼び出して オペレーションを実行する
- $lookup HealthLake によって管理されているが、データストアに明示的に保存されていない CodeSystems での オペレーション

$lookup オペレーション仕様の詳細については、FHIR R4 CodeSystem $lookup ドキュメントを参照してください。

$member-add HealthLake の オペレーション

FHIR $member-addオペレーションは、メンバー (患者) をグループリソース、特にメンバー属性リストに追加します。このオペレーションは DaVinci メンバー属性実装ガイドの一部であり、メンバー属性を管理するための調整プロセスをサポートしています。

オペレーションエンドポイント

```
POST [base]/datastore/{datastoreId}/r4/Group/{groupId}/$member-add
Content-Type: application/json
```

パラメータ

オペレーションは、次のパラメータの組み合わせを持つ FHIR Parameters リソースを受け入れます。

パラメータオプション

次のいずれかのパラメータの組み合わせを使用できます。

オプション 1: メンバー ID + プロバイダー NPI

- `memberId` + `providerNpi`
- `memberId` + `providerNpi` + `attributionPeriod`

オプション 2: 患者リファレンス + プロバイダーリファレンス

- `patientReference` + `providerReference`
- `patientReference` + `providerReference` + `attributionPeriod`

パラメータの詳細

memberId (オプション)

グループに追加するメンバーの識別子。

タイプ: Identifier

システム: 患者識別子システム

```
{
  "name": "memberId",
  "valueIdentifier": {
    "system": "http://example.org/patient-id",
    "value": "patient-new"
  }
}
```

providerNpi (オプション)

属性プロバイダーの国民プロバイダー識別子 (NPI)。

タイプ: Identifier

システム: http://terminology.hl7.org/CodeSystem/NPI

```
{
  "name": "providerNpi",
  "valueIdentifier": {
    "system": "http://terminology.hl7.org/CodeSystem/NPI",
    "value": "1234567890"
  }
}
```

patientReference (オプション)

追加する患者リソースへの直接参照。

タイプ: リファレンス

```
{
  "name": "patientReference",
  "valueReference": {
    "reference": "Patient/patient-123"
  }
}
```

providerReference (オプション)

プロバイダーリソースへの直接参照。

タイプ: リファレンス

```
{
  "name": "providerReference",
  "valueReference": {
    "reference": "Practitioner/provider-456"
  }
}
```

attributionPeriod (オプション)

患者がプロバイダーに属している期間。

タイプ: Period

```
{
  "name": "attributionPeriod",
  "valuePeriod": {
    "start": "2024-07-15",
    "end": "2025-07-14"
  }
}
```

リクエストの例

メンバー ID とプロバイダー NPI の使用

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "memberId",
      "valueIdentifier": {
        "system": "http://example.org/patient-id",
        "value": "patient-new"
      }
    },
    {
      "name": "providerNpi",
      "valueIdentifier": {
        "system": "http://terminology.hl7.org/CodeSystem/NPI",
        "value": "1234567890"
      }
    },
```

```
    {
      "name": "attributionPeriod",
      "valuePeriod": {
        "start": "2024-07-15",
        "end": "2025-07-14"
      }
    }
  ]
}
```

患者とプロバイダーのリファレンスの使用

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "patientReference",
      "valueReference": {
        "reference": "Patient/patient-123"
      }
    },
    {
      "name": "providerReference",
      "valueReference": {
        "reference": "Practitioner/provider-456"
      }
    },
    {
      "name": "attributionPeriod",
      "valuePeriod": {
        "start": "2024-07-15",
        "end": "2025-07-14"
      }
    }
  ]
}
```

レスポンスの形式

追加応答の成功

```
HTTP Status: 200 OK
Content-Type: application/fhir+json
```

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "success",
      "code": "informational",
      "details": {
        "text": "Member Patient/patient-new successfully added to the Member
 Attribution List."
      }
    }
  ]
}
```

エラーレスポンス

無効なリクエスト構文

HTTP ステータス: 400 Bad Request

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "invalid",
      "details": {
        "text": "Invalid parameter combination provided"
      }
    }
  ]
}
```

リソースが見つかりません

HTTP ステータス: 404 Not Found

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
```

```
      "code": "not-found",
      "details": {
        "text": "Resource not found."
      }
    }
  ]
}
```

バージョンの競合

HTTP ステータス: 409 競合

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "conflict",
      "details": {
        "text": "Resource version conflict detected"
      }
    }
  ]
}
```

無効な属性ステータス

HTTP ステータス: 422 Unprocessable Entity

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "business-rule",
      "details": {
        "text": "Cannot add member to Attribution List with status 'final'. Status
 must be 'draft' or 'open'."
      }
    }
  ]
}
```

ビジネスルール

属性ステータスの検証

オペレーションは、グループの属性ステータスが次の場合にのみ実行できます。

- `draft`
- `open`

ステータスが の場合、オペレーションは許可されませんfinal。

重複メンバーの防止

システムは、次の一意の組み合わせに基づいて重複したメンバーを追加することを防ぎます。

- メンバー識別子
- 支払者識別子
- カバレッジ識別子

カバレッジ期間の検証

attributionPeriod を指定する場合、メンバーのカバレッジ期間の範囲内にある必要があります。システムは以下を行います。

- メンバーのカバレッジリソースを検索する
- 複数の が存在する場合は、最新のカバレッジ (最高の versionId) を使用する
- 属性期間がカバレッジ期間を超えないことを確認する

リファレンスの検証

ID と参照の両方が同じリソース (患者またはプロバイダー) に提供されると、システムはそれらが同じリソースに対応していることを検証します。

ID フィールドと reference.identifier フィールドの両方が同じリソース (患者またはプロバイダー) に提供されると、エラーがスローされます。

認証と認可

オペレーションでは、以下に対して SMART on FHIR 認可が必要です。

- 読み取りアクセス許可 - 患者、プロバイダー、およびグループのリソースを検証するには
- 検索アクセス許可 - 識別子でリソースを検索するには
- アクセス許可を更新する - グループリソースを変更するには

運用上の動作

リソースの更新

- グループリソースバージョン ID を更新します。
- オペレーションの前に、元のリソース状態の履歴エントリを作成します。
- 以下を使用して Group.member 配列にメンバー情報を追加します。
  - entity.reference での患者リファレンス
  - 期間の属性期間
  - 拡張フィールドのカバレッジとプロバイダー情報

検証ステップ

- パラメータの検証 - 有効なパラメータの組み合わせを確認します
- リソースの存在 - 患者、プロバイダー、グループリソースが存在することを検証します
- 属性ステータス - グループステータスが変更を許可することを確認します
- 重複チェック - 既存のメンバーの追加を防止
- カバレッジ検証 - アトリビューション期間がカバレッジ範囲内であることを確認します

制限事項

- 参照されるすべてのリソースは、同じデータストア内に存在する必要があります
- オペレーションはメンバー属性リストグループのリソースでのみ機能します
- アトリビューション期間はカバレッジ範囲内である必要があります
- 「最終」ステータスのグループは変更できません

$member-match HealthLake の オペレーション

AWS HealthLake は、患者リソースの `$member-match`オペレーションをサポートするようになりました。これにより、医療組織は人口統計情報とカバレッジ情報を使用して、さまざまな医療システム全体でメンバーの一意の識別子を見つけることができます。このオペレーションは、CMS コンプライアンスを達成し、患者のプライバシーを維持しながらpayer-to-payerデータ交換を促進するために不可欠です。

このオペレーションは、以下が必要な場合に特に役立ちます。

- 組織間の安全な医療データ交換を有効にする

- さまざまなシステムにわたる患者のケアの継続性を維持する
- CMS コンプライアンス要件のサポート
- 医療ネットワーク全体で正確なメンバー識別を容易にする

使用方法

$member-match オペレーションは、POST メソッドを使用して患者リソースで呼び出すことができます。

```
POST [base]/Patient/$member-match
```

サポートされているパラメータ

HealthLake は、次の FHIR $member-matchパラメータをサポートしています。

パラメータ	タイプ	[Required] (必須)	デフォルト	説明
MemberPatient	患者	はい	—	一致するメンバーの属性情報を含む患者リソース
CoverageToMatch	カバレッジ	はい	—	既存のレコードとの照合に使用されるカバレッジリソース
CoverageToLink	カバレッジ	不可	—	マッチングプロセス中にリンクされるカバレッジリソース
同意	同意	不可	—	認可のための同意リソース

例

パラメータを使用した POST リクエスト

```
POST [base]/Patient/$member-match
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
  "parameter": [
```

```
    {
      "name": "MemberPatient",
      "resource": {
        "resourceType": "Patient",
        "name": [
          {
            "family": "Jones",
            "given": ["Sarah"]
          }
        ],
        "gender": "female",
        "birthDate": "1985-05-15"
      }
    },
    {
      "name": "CoverageToMatch",
      "resource": {
        "resourceType": "Coverage",
        "status": "active",
        "beneficiary": {
          "reference": "Patient/1"
        },
        "relationship": {
          "coding": [
            {
              "system": "http://terminology.hl7.org/CodeSystem/subscriber-
relationship",
              "code": "self",
              "display": "Self"
            }
          ]
        },
        "payor": [
          {
            "reference": "Organization/payer456"
          }
        ]
      }
    },
    {
      "name": "Consent",
      "resource": {
        "resourceType": "Consent",
        "status": "active",
```

```
        "scope": {
          "coding": [
            {
              "system": "http://terminology.hl7.org/CodeSystem/consentscope",
              "code": "patient-privacy"
            }
          ]
        },
        "category": [
          {
            "coding": [
              {
                "system": "http://terminology.hl7.org/CodeSystem/v3-ActCode",
                "code": "IDSCL"
              }
            ]
          }
        ],
        "patient": {
          "reference": "Patient/1"
        },
        "performer": [
          {
            "reference": "Patient/patient123"
          }
        ],
        "sourceReference": {
          "reference": "Document/someconsent"
        },
        "policy": [
          {
            "uri": "http://hl7.org/fhir/us/davinci-hrex/StructureDefinition-hrex-
consent.html#regular"
          }
        ]
      }
    }
  ]
}
```

レスポンス例

オペレーションは、一致する結果を含む Parameters リソースを返します。

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "MemberIdentifier",
      "valueIdentifier": {
        "system": "http://hospital.org/medical-record-number",
        "value": "MRN-123456"
      }
    },
    {
      "name": "MemberId",
      "valueReference": {
        "reference": "Patient/patient123"
      }
    },
    {
      "name": "matchAlgorithm",
      "valueString": "DEMOGRAPHIC_MATCH"
    },
    {
      "name": "matchDetails",
      "valueString": "Demographic match: DOB + Name"
    },
    {
      "name": "matchedFields",
      "valueString": "given,birthdate,gender,family"
    }
  ]
}
```

レスポンスパラメータ

一致が見つかった場合、レスポンスには次のパラメータが含まれます。

パラメータ	Type	説明
MemberIdentifier	識別子	一致したメンバーの一意の識別子
MemberId	リファレンス	患者リソースへの参照

パラメータ	Type	説明
matchAlgorithm	String	使用される一致アルゴリズムのタイプ (EXACT_MATCH、STRONG_MATCH、または DEMOGRAPHIC_MATCH)
matchDetails	String	マッチングプロセスに関する詳細情報
matchedFields	String	正常に一致した特定のフィールドのリスト

一致するアルゴリズム

$member-match API は、正確なメンバー識別を確保するために、多層マッチングアプローチを採用しています。

EXACT_MATCH

Coverage SubscriberId と組み合わせて患者識別子を使用します

メンバーマッチングの最高信頼度を提供します

STRONG_MATCH

最小カバレッジ情報で患者識別子を使用します

完全一致基準が満たされない場合に高い信頼性を提供します

DEMOGRAPHIC_MATCH

基本的な属性情報に依存する

識別子ベースのマッチングができない場合に使用されます。

行動

$member-match オペレーション:

- 患者属性、カバレッジの詳細、およびオプションの同意情報を入力として受け入れます
- 後続のインタラクションに使用できる一意のメンバー識別子を返します。
- 多層マッチング (完全、強力、属性) を実装し、さまざまな医療システム全体で正確なメンバー識別を確保します

- 将来の認可のために提供された同意情報を保存します
- 患者のプライバシーを維持しながら、安全なpayer-to-payerデータ交換をサポート
- 医療データ交換の CMS 要件に準拠

Authorization

API は、以下の必要なスコープで SMART on FHIR 認可プロトコルを使用します。

- `system/Patient.read`
- `system/Coverage.read`
- `system/Organization.read` (条件付き)
- `system/Practitioner.read` (条件付き)
- `system/PractitionerRole.read` (条件付き)
- `system/Consent.write` (条件付き)

エラー処理

オペレーションは、次のエラー条件を処理します。

- `400 Bad Request`: 無効な`$member-match`オペレーション (非準拠のリクエストまたは必須パラメータの欠落)
- `422 Unprocessable Entity`: 一致する または複数の一致する が見つかりません

`$member-remove` HealthLake の オペレーション

`$member-remove` オペレーションでは、FHIR メンバー属性リスト (グループリソース) からメンバーを削除できます AWS HealthLake。このオペレーションは DaVinci メンバー属性実装ガイドの一部であり、メンバー属性を管理するための調整プロセスをサポートしています。

前提条件

- AWS HealthLake FHIR データストア
- HealthLake オペレーションの適切な IAM アクセス許可
- ドラフトステータスまたはオープンステータスのメンバー属性リスト (グループリソース)

オペレーションの詳細

Endpoint

```
POST /Group/{id}/$member-remove
```

コンテンツタイプ

```
application/fhir+json
```

パラメータ

オペレーションは、以下のオプションパラメータを持つ FHIR Parameters リソースを受け入れます。

パラメータ	カーディナリティ	型	説明
memberId	0..1	識別子	削除するメンバーのビジネス識別子
providerNpi	0..1	識別子	属性プロバイダーの NPI
patientReference	0..1	リファレンス	患者リソースへの直接参照
providerReference	0..1	リファレンス	プロバイダーリソース (Practitioner、PractitionerRole、または Organization) への直接参照
coverageReference	0..1	リファレンス	カバレッジリソースへの参照

サポートされているパラメータの組み合わせ

以下のパラメータの組み合わせがサポートされています。

- `memberId` only - 指定されたメンバーのすべての属性を削除します
- `memberId` + `providerNpi` - 特定のメンバーとプロバイダーの組み合わせの属性を削除します
- `patientReference` のみ - 指定された患者のすべての属性を削除します
- `patientReference` + `providerReference` - 特定の患者とプロバイダーの組み合わせの属性を削除します

- patientReference + providerReference + coverageReference - 患者、プロバイダー、カバレッジに基づいて特定の属性を削除します

リクエストの例

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "patientReference",
      "valueReference": {
        "reference": "Patient/12345"
      }
    },
    {
      "name": "providerReference",
      "valueReference": {
        "reference": "Practitioner/67890"
      }
    }
  ]
}
```

応答

成功したレスポンス

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "result",
      "valueBoolean": true
    },
    {
      "name": "effectiveDate",
      "valueDate": "2024-06-30"
    },
    {
      "name": "status",
      "valueCode": "inactive"
    },
```

```
    {
      "name": "message",
      "valueString": "Member successfully removed from attribution list"
    }
  ]
}
```

行動

ステータス要件

オペレーションは、ステータスが draftまたは の属性リストでのみ機能します。 open

final ステータスが のリストは、422 エラーのオペレーションを拒否します。

メンバーの削除プロセス

ステータスリストのドラフト: メンバーは非アクティブ (inactive: true) としてマークされ、changeType拡張機能は に更新されます changed

オープンステータスリスト: ドラフトステータスと同様の動作

最終ステータスリスト: オペレーションが拒否されました

検証

リファレンスはHealthLake データストアに存在することを確認するために検証されます。

識別子と参照の両方が同じリソースタイプに提供される場合、それらは同じリソースに対応する必要があります

パラメータの組み合わせは、サポートされているパターンに従って検証されます。

エラー処理

一般的なエラーレスポンス

リソースが見つかりません (404)

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
```

```
      "code": "not-found",
      "details": {
        "text": "Patient with identifier 'http://example.org/fhir/identifiers|99999'
 not found in system"
      },
      "diagnostics": "Cannot remove member from attribution list. Verify patient
 identifier and try again.",
      "expression": ["memberId"]
    }
  ]
}
```

属性リストの最終ステータス (422)

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "business-rule",
      "details": {
        "coding": [
          {
            "system": "http://hl7.org/fhir/us/davinci-atr/CodeSystem/atr-error-
codes",
            "code": "list-final",
            "display": "Attribution list is final and cannot be modified"
          }
        ]
      },
      "diagnostics": "Cannot modify attribution list with status 'final'. List
 modifications are not permitted after finalization.",
      "expression": ["Group.status"]
    }
  ]
}
```

無効なオペレーション (400)

パラメータの組み合わせが無効または形式が正しくない場合に返されます。

複数の一致が見つかりました (412)

指定されたパラメータが属性リスト内の複数のメンバーと一致する場合に返されます。

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "processing",
      "diagnostics": "Multiple members found matching the criteria"
    }
  ]
}
```

ベストプラクティス

- 特定のパラメータを使用する: 可能な場合は、最も具体的なパラメータの組み合わせを使用して、意図しない削除を回避します。
- リストステータスのチェック: 削除を試みる前に属性リストのステータスを確認します
- エラーを適切に処理する: 考えられるすべてのエラー条件に対して適切なエラー処理を実装する
- リファレンスの検証: リクエストを行う前に、参照されるすべてのリソースが存在することを確認します。

を使用した患者部門リソースの削除 **$purge**

AWS HealthLake は $purgeオペレーションをサポートし、患者の区画内のすべてのリソースを完全に削除できます。このオペレーションは、以下が必要な場合に特に役立ちます。

- 患者に関連付けられているすべてのデータを削除する
- 患者データの削除リクエストへの準拠
- 患者データのライフサイクルを管理する
- 包括的な患者レコードのクリーンアップを実行する

使用方法

$purge オペレーションは、患者リソースで呼び出すことができます。

```
POST [base]/Patient/[ID]/$purge?deleteAuditEvent=true
```

パラメータ

パラメータ	タイプ	[Required] (必須)	デフォルト	説明
deleteAuditEvent	boolean	不可	false	true の場合、関連する監査イベントを削除します
_since	string	不可	データストアの作成時刻	入力すると、開始カットオフ時間を選択して、lastModified時間に基づいてリソースを検索します。開始または終了では使用できません
start	string	不可	データストアの作成時刻	入力すると、カットオフ時間を選択して、lastModified時間に基づいてリソースを検索します。末尾で使用できます
end	string	不可	ジョブの送信時間	入力すると、終了カットオフ時間を選択して、lastModified時間に基づいてリソースを検索します。

例

リクエストの例

```
POST [base]/Patient/example-patient/$purge?deleteAuditEvent=true
```

レスポンスの例

```
{
  "resourceType": "OperationOutcome",
  "id": "purge-job",
  "issue": [
    {
      "severity": "information",
      "code": "informational",
      "diagnostics": "Purge job started successfully. Job ID:
 12345678-1234-1234-1234-123456789012"
```

```
    }
  ]
}
```

ジョブのステータス

パージジョブのステータスを確認するには:

```
GET [base]/$purge/[jobId]
```

オペレーションはジョブステータス情報を返します。

```
{
      "datastoreId": "36622996b1fcecb7e12ee2ee085308d3",
      "jobId": "3dd1c7a5b6c0ef8c110f566eb87e2ef9",
      "status": "COMPLETED",
      "submittedTime": "2025-10-31T18:43:21.822Z"
    }
```

行動

$purge オペレーション:

1. 複数のリソースを処理するために非同期的に処理する
2. データ整合性のために ACID トランザクションを維持します
3. リソース削除数を含むジョブステータスの追跡を提供します
4. 患者区画内のすべてのリソースを完全に削除します
5. 削除アクティビティの包括的な監査ログ記録が含まれます
6. 監査イベントの選択的な削除をサポート

監査ログ記録

$purge オペレーションは、詳細なオペレーション情報を StartFHIRBulkDeleteJob および DescribeFHIRBulkDeleteJob としてログに記録します。

制限事項

- 消去されたリソースは検索レスポンスに表示されません
- 消去されるリソースは、処理中に一時的にアクセスできない場合があります

- 患者区画内のすべてのリソースが完全に削除されます

HealthLake の FHIR **$questionnaire-package**オペレーション

$questionnaire-package オペレーションは、FHIR アンケートとそのアンケートのレンダリングと処理に必要なすべての依存関係を含む包括的なバンドルを取得します。このオペレーションでは、Da Vinci ドキュメントテンプレートとルール (DTR) 実装ガイドを実装し、ヘルスケアワークフローのドキュメント要件の動的フォームレンダリングを有効にします。

仕組み

- リクエスト: カバレッジと注文コンテキストとともに、必要なアンケート (複数可) を識別するパラメータを送信します。
- 取得: HealthLake はアンケートとすべての依存関係 (ValueSets、CQL ライブラリなど) を収集します。
- パッケージ: すべてのリソースは標準化された形式でバンドルされます
- 応答: レンダリングとデータ収集の準備ができた完全なパッケージが届きます。

ユースケース

- 事前認可ドキュメント: 事前認可リクエストに必要な臨床情報を収集する
- カバレッジ要件: 支払者カバレッジ要件を満たすために必要なドキュメントを収集する
- 臨床データ交換: 支払者に送信するための臨床データを構築する
- 動的フォーム: 事前入力された患者データと条件付きロジックを使用してアンケートをレンダリングする

API エンドポイント

```
POST /datastore/{datastoreId}/r4/Questionnaire/$questionnaire-package
Content-Type: application/fhir+json
```

リクエストパラメータ

入力パラメータ

リクエスト本文には、次のパラメータを含む FHIR Parameters リソースが含まれている必要があります。

パラメータ	タイプ	カーディナリティ	説明
coverage	カバレッジ	1..* (必須)	ドキュメントのメンバーとカバレッジを確立するためのカバレッジリソース (複数可)
questionnaire	canonical	0..*	返す特定のアンケートの正規 URL (複数可) (バージョンを含む場合があります)
order	[リソース]	0..*	コンテキストを確立するためにリソース (DeviceRequest、ServiceRequest、MedicationRequest、Encounter、Appointment) を注文する
changedSince	dateTime	0..1	存在する場合、このタイムスタンプの後に変更されたリソースのみを返す

パラメータ検証ルール

次のいずれかを指定する必要があります (必須 に加えてcoverage)。

- 1 つ以上のquestionnaire正規 URLs
- 1 つ以上のorderリソース

有効なリクエストの組み合わせ:

- coverage + questionnaire
- coverage + order
- coverage + questionnaire + order

リクエスト例

```
POST /datastore/example-datastore/r4/Questionnaire/$questionnaire-package
Content-Type: application/fhir+json
Authorization: Bearer <your-token>

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "coverage",
      "resource": {
        "resourceType": "Coverage",
        "id": "example-coverage",
        "status": "active",
        "beneficiary": {
          "reference": "Patient/example-patient"
        },
        "payor": [{
          "reference": "Organization/example-payer"
        }],
        "class": [{
          "type": {
            "coding": [{
              "system": "http://terminology.hl7.org/CodeSystem/coverage-class",
              "code": "group"
            }]
          },
          "value": "12345"
        }]
      }
    },
    {
      "name": "questionnaire",
      "valueCanonical": "http://example.org/fhir/Questionnaire/home-oxygen-therapy|2.0"

    },
    {
      "name": "order",
      "resource": {
        "resourceType": "ServiceRequest",
        "id": "example-service-request",
        "status": "active",
        "intent": "order",
```

```
        "code": {
          "coding": [{
            "system": "http://www.ama-assn.org/go/cpt",
            "code": "94660",
            "display": "Continuous positive airway pressure ventilation (CPAP)"
          }]
        },
        "subject": {
          "reference": "Patient/example-patient"
        }
      }
    },
    {
      "name": "changedSince",
      "valueDateTime": "2024-01-01T00:00:00Z"
    }
  ]
}
```

レスポンスの形式

成功レスポンス (200 OK)

オペレーションは、1 つ以上のパッケージバンドルを含む FHIR Parameters リソースを返します。各パッケージバンドルには以下が含まれます。

エントリタイプ	カーディナリティ	説明
アンケート	1	レンダリングするアンケート
QuestionnaireResponse	0..1	事前入力済みまたは部分的に完了したレスポンス (該当する場合)
[Library] (ライブラリ)	0..*	事前入力と条件付きロジックを含む CQL ライブラリ
ValueSet	0..*	拡張 ValueSets (<40 拡張の回答選択の場合)

Exampleレスポンスの例

```
{
```

```
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "PackageBundle",
      "resource": {
        "resourceType": "Bundle",
        "id": "questionnaire-package-example",
        "meta": {
          "profile": ["http://hl7.org/fhir/us/davinci-dtr/StructureDefinition/DTR-
QPackageBundle"]
        },
        "type": "collection",
        "timestamp": "2024-03-15T10:30:00Z",
        "entry": [
          {
            "fullUrl": "http://example.org/fhir/Questionnaire/home-oxygen-therapy",
            "resource": {
              "resourceType": "Questionnaire",
              "id": "home-oxygen-therapy",
              "url": "http://example.org/fhir/Questionnaire/home-oxygen-therapy",
              "version": "2.0",
              "status": "active",
              "title": "Home Oxygen Therapy Documentation",
              "item": [
                {
                  "linkId": "1",
                  "text": "Patient diagnosis",
                  "type": "choice",
                  "answerValueSet": "http://example.org/fhir/ValueSet/oxygen-diagnoses"

                }
              ]
            }
          },
          {
            "fullUrl": "http://example.org/fhir/Library/oxygen-prepopulation",
            "resource": {
              "resourceType": "Library",
              "id": "oxygen-prepopulation",
              "url": "http://example.org/fhir/Library/oxygen-prepopulation",
              "version": "1.0",
              "type": {
                "coding": [{
                  "system": "http://terminology.hl7.org/CodeSystem/library-type",
```

```
                "code": "logic-library"
              }]
            },
            "content": [{
              "contentType": "text/cql",
              "data": "bGlicmFyeSBPeHlnZW5QcmVwb3B1bGF0aW9u..."
            }]
          }
        },
        {
          "fullUrl": "http://example.org/fhir/ValueSet/oxygen-diagnoses",
          "resource": {
            "resourceType": "ValueSet",
            "id": "oxygen-diagnoses",
            "url": "http://example.org/fhir/ValueSet/oxygen-diagnoses",
            "status": "active",
            "expansion": {
              "timestamp": "2024-03-15T10:30:00Z",
              "contains": [
                {
                  "system": "http://hl7.org/fhir/sid/icd-10",
                  "code": "J44.0",
                  "display": "COPD with acute lower respiratory infection"
                },
                {
                  "system": "http://hl7.org/fhir/sid/icd-10",
                  "code": "J96.01",
                  "display": "Acute respiratory failure with hypoxia"
                }
              ]
            }
          }
        },
        {
          "fullUrl": "http://example.org/fhir/QuestionnaireResponse/example-
prepopulated",
          "resource": {
            "resourceType": "QuestionnaireResponse",
            "id": "example-prepopulated",
            "questionnaire": "http://example.org/fhir/Questionnaire/home-oxygen-
therapy|2.0",
            "status": "in-progress",
            "subject": {
              "reference": "Patient/example-patient"
```

```
              },
              "basedOn": [{
                "reference": "ServiceRequest/example-service-request"
              }],
              "item": [
                {
                  "linkId": "1",
                  "text": "Patient diagnosis",
                  "answer": [{
                    "valueCoding": {
                      "system": "http://hl7.org/fhir/sid/icd-10",
                      "code": "J44.0",
                      "display": "COPD with acute lower respiratory infection"
                    }
                  }]
                }
              ]
            }
          }
        ]
      }
    },
    {
      "name": "Outcome",
      "resource": {
        "resourceType": "OperationOutcome",
        "issue": [{
          "severity": "information",
          "code": "informational",
          "details": {
            "text": "Successfully retrieved questionnaire package"
          }
        }]
      }
    }
  ]
}
```

オペレーションワークフロー

HealthLake がリクエストを処理する方法

を呼び出すと`$questionnaire-package`、HealthLake は次のステップを実行します。

1. 患者と支払い者を特定する: coverageパラメータから患者と保険組織を抽出します。
2. 適切なアンケートを見つけます。
   - パラメータの場合questionnaire: 指定した正規 URL を使用します
   - パラメータ付きorder: 注文コード (CPT/HCPCS/LOINC) と支払者を照合して、適切なアンケートを見つけます
3. 依存関係の収集: アンケートのレンダリングに必要なすべてを自動的に取得します。
   - CQL ライブラリ - 事前入力と条件付き質問のロジック
   - ValueSets - 選択肢への回答 (<40 オプションの場合は自動的に展開)
   - QuestionnaireResponse - 進行中の既存のレスポンスまたは完了したレスポンス
4. すべてをまとめてパッケージ化:
   - すべてのリソースをバンドルします (各リソースは 1 回のみ含まれます)
   - 指定した場合のchangedSinceタイムスタンプによるフィルタリング
   - Outcome リソースがない場合に警告を に追加します

結果: レンダリング可能な完全で自己完結型のパッケージ。

エラーレスポンス

400 Bad Request

リクエストの検証が失敗したときに返されます。

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "required",
    "details": {
      "text": "At least one of 'questionnaire' or 'order' must be provided along with
 'coverage'"
    }
  }]
}
```

424 失敗した依存関係

依存リソースを取得できない場合に返されます。

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "warning",
    "code": "not-found",
    "details": {
      "text": "Referenced Library 'http://example.org/fhir/Library/missing-library'
 could not be retrieved"
    }
  }]
}
```

401 Unauthorized

認証情報がないか無効である場合に返されます。

403 Forbidden

認証されたユーザーに、リクエストされたリソースへのアクセス許可がない場合に返されます。

406 使用できません

リクエストされたコンテンツタイプを指定できない場合に返されます。

409 Conflict

バージョンまたは同時実行の競合がある場合に返されます。

410 終了

リクエストされたリソースが完全に削除されたときに返されます。

429 Too Many Requests

レート制限を超えたときに返されます。

500 Internal Server Error

予期しないサーバーエラーが発生した場合に返されます。

501 未実装

リクエストされたオペレーションがまだ実装されていない場合に返されます。

検証ルール

入力の検証

- coverage パラメータは必須です (1..* 基数)
- 少なくとも 1 つの questionnaireまたは を指定orderする必要があります
- すべてのカバレッジリソースは有効な FHIR リソースである必要があります
- すべての注文リソースは有効な FHIR リソースである必要があります
- 正規 URLs適切にフォーマットする必要があります
- changedSince は有効な ISO 8601 dateTime である必要があります

QuestionnaireResponse の検証

- status は適切である必要があります (in-progress、completed、amended)
- 構造は参照されるアンケートと一致する必要があります
- basedOn は有効な注文リソースを参照する必要があります
- subject は有効な患者リソースを参照する必要があります

リソース重複排除

- 各リソースはバンドルに 1 回のみ表示されます。
- 例外: 同じリソースの異なるバージョンの両方が含まれる場合があります
- リソースは正規 URL とバージョンによって識別されます

パフォーマンス仕様

メトリクス	の仕様
リソース数の制限	バンドルあたり 500 リソース
バンドルサイズ制限	最大 5 MB

必要なアクセス許可

$questionnaire-package オペレーションを使用するには、IAM ロールに次のものがあることを確認します。

- healthlake:QuestionnairePackage - オペレーションを呼び出すには
- healthlake:ReadResource - アンケートと依存リソースを取得するには
- healthlake:SearchWithPost - QuestionnaireResponse および関連リソースを検索するには

FHIR スコープでの SMART

最低限必要なスコープ:

- SMART v1: user/Questionnaire.read user/Library.read user/ValueSet.read user/QuestionnaireResponse.read
- SMART v2: user/Questionnaire.rs user/Library.rs user/ValueSet.rs user/QuestionnaireResponse.rs

重要な実装上の注意事項

リソース取得戦略

アンケート識別の優先度:

- 正規 URL (questionnaireパラメータが指定されている場合) - 最高優先度
- 注文分析 (order パラメータが指定されている場合):
  - 注文コード (CPT、HCPCS、LOINC) を支払い医療ポリシーと照合する
  - カバレッジ支払者を使用して支払者固有のアンケートをフィルタリングする
  - 追加のコンテキストの理由コードを検討する

依存関係の解決

CQL ライブラリ:

- アンケートリソースの cqf-library 拡張機能を介して取得
- 依存ライブラリを タイプLibrary.relatedArtifactで再帰的に取得する depends-on

- すべてのライブラリの依存関係がパッケージに含まれています

ValueSets:

- 40 未満の概念が含まれている場合、自動的に拡張されます
- より大きな ValueSets は拡張なしで含まれます
- アンケートリソースとライブラリリソースの両方で参照される ValueSets が含まれています

QuestionnaireResponse の事前入力

オペレーションは、次の場合に事前入力されたデータを含む QuestionnaireResponse を返す場合があります。

- 既存の進行中または完了したレスポンスが見つかった
- 関連するライブラリの CQL ロジックは、患者レコードからデータを抽出できます
- レスポンスが関連する注文とカバレッジにリンクされている

QuestionnaireResponse の検索条件:

検索パラメータ	FHIR パス	説明
based-on	QuestionnaireResponse.basedOn	ServiceRequest または CarePlan へのリンク
patient	QuestionnaireResponse.subject	対象である患者
questionnaire	QuestionnaireResponse.questionnaire	回答中のアンケート

変更されたリソースのフィルタリング

changedSince パラメータが指定されている場合:

- 指定されたタイムスタンプ後に変更されたリソースのみが含まれます

- リソースが変更されていない場合、 は空のパッケージ200 OKで を返します。
- 増分更新とキャッシュ戦略に役立ちます
- タイムスタンプ比較でリソースmeta.lastUpdatedフィールドを使用する

複数のパッケージバンドル

オペレーションは、次の場合に複数のパッケージバンドルを返す場合があります。

- 正規 URLs
- 複数の注文で異なるアンケートが必要
- 同じアンケートの異なるバージョンが適用される

各パッケージバンドルは、必要なすべての依存関係を含む自己完結型です。

一般的なユースケース

ユースケース 1: 事前承認ドキュメント

シナリオ: プロバイダーは、住宅用人工鼻の事前認可に関するドキュメントを収集する必要があります。

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "coverage",
      "resource": { /* Patient's insurance coverage */ }
    },
    {
      "name": "order",
      "resource": {
        "resourceType": "ServiceRequest",
        "code": {
          "coding": [{
            "system": "http://www.ama-assn.org/go/cpt",
            "code": "94660"
          }]
        }
      }
    }
```

```
  ]
}
```

結果: EHR から患者のバイタルと診断コードがあらかじめ入力されている、O2 度治療用アンケートを含むパッケージを返します。

ユースケース 2: 特定のアンケートバージョンを取得する

シナリオ: コンプライアンスのためにプロバイダーに特定のバージョンのアンケートが必要です。

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "coverage",
      "resource": { /* Coverage resource */ }
    },
    {
      "name": "questionnaire",
      "valueCanonical": "http://example.org/fhir/Questionnaire/dme-request|3.1.0"
    }
  ]
}
```

結果: すべての依存関係を持つ DME リクエストアンケートのバージョン 3.1.0 を正確に返します。

ユースケース 3: 更新を確認する

シナリオ: プロバイダーは、前回の取得以降に更新されたアンケートリソースがあるかどうかを確認したいと考えています。

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "coverage",
      "resource": { /* Coverage resource */ }
    },
    {
      "name": "questionnaire",
      "valueCanonical": "http://example.org/fhir/Questionnaire/medication-request"
    },
    {
```

```
      "name": "changedSince",
      "valueDateTime": "2024-03-01T00:00:00Z"
    }
  ]
}
```

結果: 2024 年 3 月 1 日以降に変更されたリソースのみ、または何も変更されていない場合は空のパッケージを返します。

ユースケース 4: 複数の注文

シナリオ: プロバイダーは、異なるアンケートを必要とする可能性のある複数のサービスリクエストを送信します。

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "coverage",
      "resource": { /* Coverage resource */ }
    },
    {
      "name": "order",
      "resource": { /* ServiceRequest for imaging */ }
    },
    {
      "name": "order",
      "resource": { /* ServiceRequest for DME */ }
    }
  ]
}
```

結果: 該当するアンケートごとに 1 つずつ、複数のパッケージバンドルを返します。

他の Da Vinci IGs との統合

カバレッジ要件検出 (CRD)

ワークフロー統合:

- プロバイダーが EHR でサービスを注文する
- CRD フックの起動、カバレッジ要件の確認

- 支払者がドキュメントが必要であることを示す応答
- プロバイダーが $questionnaire-packageを呼び出してドキュメントフォームを取得する
- プロバイダーがアンケートを完了する
- ドキュメントが PAS または CDex 経由で送信される

事前認可サポート (PAS)

ワークフロー統合:

- $questionnaire-package を使用してドキュメント要件を取得する
- 必要な臨床データを使用して QuestionnaireResponse を完了する
- 記入済みの QuestionnaireResponse Claim/$submit で を使用して事前認可を送信する
- を使用してステータスを確認する Claim/$inquire

臨床データ交換 (CDex)

ワークフロー統合:

- 支払者がクレームの追加ドキュメントをリクエストする
- プロバイダーが $questionnaire-packageを使用して構造化データ収集フォームを取得する
- プロバイダーが QuestionnaireResponse を完了する
- ドキュメントは CDex アタッチメントワークフローを介して支払者に送信されます

トラブルシューティングガイド

問題: アンケートが返されない

考えられる原因:

- 正規 URL がデータストア内のアンケートと一致しません
- 注文コードが支払者の医療ポリシーのアンケートにマッピングされない
- カバレッジ支払者にアンケートが関連付けられていない

解決方法:

- 正規 URL が正しく、アンケートが存在することを確認する
- 注文コード (CPT/HCPCS) が正しく指定されていることを確認します。
- 支払者の組織にアンケートが設定されていることを確認します。

問題: パッケージに依存関係がない

考えられる原因:

- 参照ライブラリまたは ValueSet がデータストアに存在しません
- ライブラリ参照が壊れているか正しくない
- ValueSet の拡張に失敗しました

解決方法:

- 不足しているリソースに関する警告については、 `Outcome`パラメータを確認してください。
- 参照されるすべてのリソースがデータストアに存在することを確認する
- ValueSet URLs が正しく解決可能であることを確認する

問題: changedSince のパッケージが空

考えられる原因:

- これは予想される動作です - 指定されたタイムスタンプ以降にリソースが変更されていません

解決方法:

- キャッシュされたバージョンのパッケージを使用する
- `changedSince` パラメータを削除して完全なパッケージを取得する

問題: QuestionnaireResponse が事前に入力されていません

考えられる原因:

- 既存の QuestionnaireResponse が見つかりません
- CQL ライブラリロジックが必要なデータを抽出できませんでした
- 患者データが欠落しているか不完全である

解決方法:

- これは想定されている場合があります。すべてのアンケートに事前入力ロジックがあるわけではありません。
- データストアに患者データが存在することを確認する
- データ抽出要件について CQL ライブラリロジックを確認する

ベストプラクティス

1. バージョンで正規 URLs を使用する

特定のアンケートをリクエストするときは、常にバージョン番号を指定します。

```
{
  "name": "questionnaire",
  "valueCanonical": "http://example.org/fhir/Questionnaire/dme|2.1.0"
}
```

理由: 一貫性を確保し、アンケートが更新されたときの予期しない変更を防ぎます。

2. changedSince for Performance を活用する

頻繁にアクセスchangedSinceされるアンケートの場合は、 を使用してデータ転送を最小限に抑えます。

```
{
  "name": "changedSince",
  "valueDateTime": "2024-03-10T15:30:00Z"
}
```

理由: 更新されたリソースのみを取得することで、レイテンシーと帯域幅の使用量を削減します。

3. 完全なカバレッジ情報を含める

アンケートを正しく選択できるように、包括的なカバレッジの詳細を提供します。

```
{
  "name": "coverage",
  "resource": {
    "resourceType": "Coverage",
    "beneficiary": { "reference": "Patient/123" },
    "payor": [{ "reference": "Organization/payer-abc" }],
    "class": [{ /* Group/plan information */ }]
  }
}
```

理由: HealthLake が支払者固有のアンケートと要件を特定するのに役立ちます。

関連の オペレーション

- `Claim/$submit` - 完成したドキュメントを使用して事前承認リクエストを送信する
- `Claim/$inquire` - 送信された事前認可のステータスを確認する
- `ValueSet/$expand` - ValueSets を展開して回答を選択する

HealthLake の FHIR **`$submit`**オペレーション

`$submit` オペレーションを使用すると、事前承認リクエストを支払者に送信して承認を受けることができます。このオペレーションでは、Da Vinci 事前認可サポート (PAS) 実装ガイドを実装し、事前認可送信用の標準化された FHIR ベースのワークフローを提供します。

仕組み

- 送信: 以前の承認リクエストとサポートする臨床データを含む FHIR バンドルを送信する
- 検証: HealthLake は PAS 要件に照らして送信を検証します
- 永続化: すべてのリソースが HealthLake データストアに保存されます。
- 応答: 「キューに入れられた」ステータスの即時応答を受け取ります。
- プロセス: 承認決定は支払者によって非同期的に処理されます

API エンドポイント

```
POST /datastore/{datastoreId}/r4/Claim/$submit
Content-Type: application/fhir+json
```

リクエスト構造

バンドルの要件

リクエストは、以下を含む FHIR Bundle リソースである必要があります。

- Bundle.type: である必要があります `"collection"`
- Bundle.entry: で 1 つのクレームリソースのみを含める必要があります `use = "preauthorization"`
- 参照リソース: クレームによって参照されるすべてのリソースをバンドルに含める必要があります

必要なリソース

[リソース]	カーディナリティ	プロファイル	説明
クレーム	1	PAS クレーム	以前の認可リクエスト
患者	1	PAS 患者	患者の人口統計情報
組織 (保険者)	1	PAS 保険会社	保険会社
組織 (プロバイダー)	1	PAS リクエスタ	リクエストを送信する医療プロバイダー
カバレッジ	1 つ以上	PAS カバレッジ	保険範囲の詳細

オプションのリソース

[リソース]	カーディナリティ	プロファイル	説明
プラクティショナー	0 以上	PAS プラクティショナー	医療関係者
PractitionerRole	0 以上	PAS PractitionerRole	プラクティショナーロール

[リソース]	カーディナリティ	プロファイル	説明
ServiceRequest	0 以上	PAS ServiceRequest	リクエストされた医療サービス
DeviceRequest	0 以上	PAS DeviceRequest	リクエストされた医療デバイス
MedicationRequest	0 以上	PAS MedicationRequest	リクエストされた薬剤
DocumentReference	0 以上	PAS DocumentReference	サポートされている臨床ドキュメント

リクエスト例

```
POST /datastore/example-datastore/r4/Claim/$submit
Content-Type: application/fhir+json
Authorization: Bearer <your-token>

{
  "resourceType" : "Bundle",
  "id" : "MedicalServicesAuthorizationBundleExample",
  "meta" : {
    "profile" : ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-pas-
request-bundle"]
  },
  "identifier" : {
    "system" : "http://example.org/SUBMITTER_TRANSACTION_IDENTIFIER",
    "value" : "5269367"
  },
  "type" : "collection",
  "timestamp" : "2005-05-02T11:01:00+05:00",
  "entry" : [{
    "fullUrl" : "http://example.org/fhir/Claim/MedicalServicesAuthorizationExample",
    "resource" : {
      "resourceType" : "Claim",
      "id" : "MedicalServicesAuthorizationExample",
      "meta" : {
        "profile" : ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
claim"]
```

```
      },
      "identifier" : [{
        "system" : "http://example.org/PATIENT_EVENT_TRACE_NUMBER",
        "value" : "111099"
      }],
      "status" : "active",
      "type" : {
        "coding" : [{
          "system" : "http://terminology.hl7.org/CodeSystem/claim-type",
          "code" : "professional"
        }]
      },
      "use" : "preauthorization",
      "patient" : {
        "reference" : "Patient/SubscriberExample"
      },
      "created" : "2005-05-02T11:01:00+05:00",
      "insurer" : {
        "reference" : "Organization/InsurerExample"
      },
      "provider" : {
        "reference" : "Organization/UMOExample"
      },
      "priority" : {
        "coding" : [{
          "system" : "http://terminology.hl7.org/CodeSystem/processpriority",
          "code" : "normal"
        }]
      },
      "insurance" : [{
        "sequence" : 1,
        "focal" : true,
        "coverage" : {
          "reference" : "Coverage/InsuranceExample"
        }
      }],
      "item" : [{
        "extension" : [{
          "url" : "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/extension-
serviceItemRequestType",
          "valueCodeableConcept" : {
            "coding" : [{
              "system" : "https://codesystem.x12.org/005010/1525",
              "code" : "IN",
```

```
              "display" : "Initial Medical Services Reservation"
            }]
          }
        },
        {
          "url" : "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/extension-
certificationType",
          "valueCodeableConcept" : {
            "coding" : [{
              "system" : "https://codesystem.x12.org/005010/1322",
              "code" : "I",
              "display" : "Initial"
            }]
          }
        },
        {
          "url" : "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/extension-
authorizationNumber",
          "valueString" : "1122344"
        },
        {
          "url" : "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/extension-
administrationReferenceNumber",
          "valueString" : "33441122"
        }],
        "sequence" : 1,
        "category" : {
          "coding" : [{
            "system" : "https://codesystem.x12.org/005010/1365",
            "code" : "1",
            "display" : "Medical Care"
          }]
        },
        "productOrService" : {
          "coding" : [{
            "system" : "http://www.cms.gov/Medicare/Coding/HCPCSReleaseCodeSets",
            "code" : "99212",
            "display" : "Established Office Visit"
          }]
        },
        "servicedDate" : "2005-05-10",
        "locationCodeableConcept" : {
          "coding" : [{
```

```
            "system" : "https://www.cms.gov/Medicare/Coding/place-of-service-codes/
Place_of_Service_Code_Set",
            "code" : "11"
          }]
        }
      }]
    }
  },
  {
    "fullUrl" : "http://example.org/fhir/Organization/UMOExample",
    "resource" : {
      "resourceType" : "Organization",
      "id" : "UMOExample",
      "meta" : {
        "profile" : ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
requestor"]
      },
      "identifier" : [{
        "system" : "http://hl7.org/fhir/sid/us-npi",
        "value" : "8189991234"
      }],
      "active" : true,
      "type" : [{
        "coding" : [{
          "system" : "https://codesystem.x12.org/005010/98",
          "code" : "X3"
        }]
      }],
      "name" : "DR. JOE SMITH CORPORATION",
      "address" : [{
        "line" : ["111 1ST STREET"],
        "city" : "SAN DIEGO",
        "state" : "CA",
        "postalCode" : "92101",
        "country" : "US"
      }]
    }
  },
  {
    "fullUrl" : "http://example.org/fhir/Organization/InsurerExample",
    "resource" : {
      "resourceType" : "Organization",
      "id" : "InsurerExample",
      "meta" : {
```

```
        "profile" : ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
insurer"]
      },
      "identifier" : [{
        "system" : "http://hl7.org/fhir/sid/us-npi",
        "value" : "1234567893"
      }],
      "active" : true,
      "type" : [{
        "coding" : [{
          "system" : "https://codesystem.x12.org/005010/98",
          "code" : "PR"
        }]
      }],
      "name" : "MARYLAND CAPITAL INSURANCE COMPANY"
    }
  },
  {
    "fullUrl" : "http://example.org/fhir/Coverage/InsuranceExample",
    "resource" : {
      "resourceType" : "Coverage",
      "id" : "InsuranceExample",
      "meta" : {
        "profile" : ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
coverage"]
      },
      "status" : "active",
      "subscriberId" : "1122334455",
      "beneficiary" : {
        "reference" : "Patient/SubscriberExample"
      },
      "relationship" : {
        "coding" : [{
          "system" : "http://terminology.hl7.org/CodeSystem/subscriber-relationship",
          "code" : "self"
        },
        {
          "system" : "https://codesystem.x12.org/005010/1069",
          "code" : "18"
        }]
      },
      "payor" : [{
        "reference" : "Organization/InsurerExample"
      }]
```

```
    }
  },
  {
    "fullUrl" : "http://example.org/fhir/Patient/SubscriberExample",
    "resource" : {
      "resourceType" : "Patient",
      "id" : "SubscriberExample",
      "meta" : {
        "profile" : ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
subscriber"]
      },
      "extension" : [{
        "url" : "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/extension-
militaryStatus",
        "valueCodeableConcept" : {
          "coding" : [{
            "system" : "https://codesystem.x12.org/005010/584",
            "code" : "RU"
          }]
        }
      }],
      "identifier" : [{
        "type" : {
          "coding" : [{
            "system" : "http://terminology.hl7.org/CodeSystem/v2-0203",
            "code" : "MB"
          }]
        },
        "system" : "http://example.org/MIN",
        "value" : "12345678901"
      }],
      "name" : [{
        "family" : "SMITH",
        "given" : ["JOE"]
      }],
      "gender" : "male"
    }
  }]
}
```

レスポンスの形式

成功レスポンス (200 OK)

以下を含む PAS レスポンスバンドルが届きます。

- outcome: "queued" と を使用した ClaimResponse status: "active"
- リクエストのすべての元のリソース
- 受信を確認するタイムスタンプ

```
{
  "resourceType" : "Bundle",
  "identifier": {
        "system": "http://example.org/SUBMITTER_TRANSACTION_IDENTIFIER",
        "value": "5269367"
  },
  "type" : "collection",
  "timestamp" : "2005-05-02T11:02:00+05:00",
  "entry" : [{
    "fullUrl" : "http://example.org/fhir/ClaimResponse/
PractitionerRequestorPendingResponseExample",
    "resource" : {
      "resourceType" : "ClaimResponse",
      "id" : "PractitionerRequestorPendingResponseExample",
      "meta" : {
        "profile" : ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
claimresponse"]
      },
      "identifier" : [{
        "system" : "http://example.org/PATIENT_EVENT_TRACE_NUMBER",
        "value" : "111099"
      }],
      "status" : "active",
      "type" : {
        "coding" : [{
          "system" : "http://terminology.hl7.org/CodeSystem/claim-type",
          "code" : "professional"
        }]
      },
      "use" : "preauthorization",
      "patient" : {
        "reference" : "Patient/SubscriberExample"
```

```
      },
      "created" : "2005-05-02T11:02:00+05:00",
      "insurer" : {
        "reference" : "Organization/InsurerExample"
      },
      "requestor" : {
        "reference" : "PractitionerRole/ReferralPractitionerRoleExample"
      },
      "request" : {
        "reference" : "Claim/MedicalServicesAuthorizationExample"
      },
      "outcome" : "queued"
    }
  },
  {
    "fullUrl" : "http://example.org/fhir/Claim/MedicalServicesAuthorizationExample",
    "resource" : {
      "resourceType" : "Claim",
      "id" : "MedicalServicesAuthorizationExample",
      "meta" : {
        "profile": [
            "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-claim",
            "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-claim|
2.1.0"
        ]
      },
      "identifier" : [{
        "system" : "http://example.org/PATIENT_EVENT_TRACE_NUMBER",
        "value" : "111099"
        }
      }],
      "status" : "active",
      "type" : {
        "coding" : [{
          "system" : "http://terminology.hl7.org/CodeSystem/claim-type",
          "code" : "professional"
        }]
      },
      "use" : "preauthorization",
      "patient" : {
        "reference" : "Patient/SubscriberExample"
      },
      "created" : "2005-05-02T11:01:00+05:00",
      "insurer" : {
```

```
        "reference" : "Organization/InsurerExample"
      },
      "provider" : {
        "reference" : "Organization/UMOExample"
      },
      "priority" : {
        "coding" : [{
          "system" : "http://terminology.hl7.org/CodeSystem/processpriority",
          "code" : "normal"
        }]
      },
      "insurance" : [{
        "sequence" : 1,
        "focal" : true,
        "coverage" : {
          "reference" : "Coverage/InsuranceExample"
        }
      }],
      "item" : [{
        "extension" : [{
          "url" : "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/extension-
serviceItemRequestType",
          "valueCodeableConcept" : {
            "coding" : [{
              "system" : "https://codesystem.x12.org/005010/1525",
              "code" : "IN",
              "display" : "Initial Medical Services Reservation"
            }]
          }
        },
        {
          "url" : "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/extension-
certificationType",
          "valueCodeableConcept" : {
            "coding" : [{
              "system" : "https://codesystem.x12.org/005010/1322",
              "code" : "I",
              "display" : "Initial"
            }]
          }
        },
        {
          "url" : "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/extension-
authorizationNumber",
```

```
          "valueString" : "1122344"
        },
        {
          "url" : "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/extension-
administrationReferenceNumber",
          "valueString" : "33441122"
        }],
        "sequence" : 1,
        "category" : {
          "coding" : [{
            "system" : "https://codesystem.x12.org/005010/1365",
            "code" : "1",
            "display" : "Medical Care"
          }]
        },
        "productOrService" : {
          "coding" : [{
            "system" : "http://www.cms.gov/Medicare/Coding/HCPCSReleaseCodeSets",
            "code" : "99212",
            "display" : "Established Office Visit"
          }]
        },
        "servicedDate" : "2005-05-10",
        "locationCodeableConcept" : {
          "coding" : [{
            "system" : "https://www.cms.gov/Medicare/Coding/place-of-service-codes/
Place_of_Service_Code_Set",
            "code" : "11"
          }]
        }
      }]
    }
  },
  {
    "fullUrl" : "http://example.org/fhir/Organization/UMOExample",
    "resource" : {
      "resourceType" : "Organization",
      "id" : "UMOExample",
      "meta" : {
        "profile": [
            "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-requestor",

            "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-requestor|
2.1.0"
```

```
        ]
      },
      "identifier" : [{
        "system" : "http://hl7.org/fhir/sid/us-npi",
        "value" : "8189991234"
      }],
      "active" : true,
      "type" : [{
        "coding" : [{
          "system" : "https://codesystem.x12.org/005010/98",
          "code" : "X3"
        }]
      }],
      "name" : "DR. JOE SMITH CORPORATION",
      "address" : [{
        "line" : ["111 1ST STREET"],
        "city" : "SAN DIEGO",
        "state" : "CA",
        "postalCode" : "92101",
        "country" : "US"
      }]
    }
  },
  {
    "fullUrl" : "http://example.org/fhir/Organization/InsurerExample",
    "resource" : {
      "resourceType" : "Organization",
      "id" : "InsurerExample",
      "meta" : {
           "profile": [
                "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
insurer",
                "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
insurer|2.1.0"
            ]
      },
      "identifier" : [{
        "system" : "http://hl7.org/fhir/sid/us-npi",
        "value" : "1234567893"
      }],
      "active" : true,
      "type" : [{
        "coding" : [{
          "system" : "https://codesystem.x12.org/005010/98",
```

```
          "code" : "PR"
        }]
      }],
      "name" : "MARYLAND CAPITAL INSURANCE COMPANY"
    }
  },
  {
    "fullUrl" : "http://example.org/fhir/Coverage/InsuranceExample",
    "resource" : {
      "resourceType" : "Coverage",
      "id" : "InsuranceExample",
      "meta": {
            "profile": [
                "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
coverage",
                "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
coverage|2.1.0"
            ]
        },
      "status" : "active",
      "subscriberId" : "1122334455",
      "beneficiary" : {
        "reference" : "Patient/SubscriberExample"
      },
      "relationship" : {
        "coding" : [{
          "system" : "http://terminology.hl7.org/CodeSystem/subscriber-relationship",
          "code" : "self"
        },
        {
          "system" : "https://codesystem.x12.org/005010/1069",
          "code" : "18"
        }]
      },
      "payor" : [{
        "reference" : "Organization/InsurerExample"
      }]
    }
  },
  {
    "fullUrl" : "http://example.org/fhir/Patient/SubscriberExample",
    "resource" : {
      "resourceType" : "Patient",
      "id" : "SubscriberExample",
```

```
      "meta": {
            "profile": [
                "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
subscriber",
                "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
beneficiary",
                "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
beneficiary|2.1.0"
            ]
        },
      "extension" : [{
        "url" : "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/extension-
militaryStatus",
        "valueCodeableConcept" : {
          "coding" : [{
            "system" : "https://codesystem.x12.org/005010/584",
            "code" : "RU"
          }]
        }
      }],
      "identifier" : [{
        "type" : {
          "coding" : [{
            "system" : "http://terminology.hl7.org/CodeSystem/v2-0203",
            "code" : "MB"
          }]
        },
        "system" : "http://example.org/MIN",
        "value" : "12345678901"
      }],
      "name" : [{
        "family" : "SMITH",
        "given" : ["JOE"]
      }],
      "gender" : "male"
    }
  }]
}
```

エラーレスポンス

400 Bad Request

リクエスト形式が無効または形式が間違っている場合に返されます。

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "invalid",
    "diagnostics": "The provided payload was invalid and could not be parsed
 correctly."
  }]
}
```

412 Precondition Failed

同じ以前の承認リクエストがすでに送信されている場合 (重複した送信が検出された場合) に返されます。

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "processing",
    "diagnostics": "PreAuth Claim already exists"
  }]
}
```

> **べき等性**
>
> $submit オペレーションはべき等です。同じリクエストを複数回送信しても、以前の承認リクエストが重複することはありません。代わりに、 $inquireを使用して元の送信のステータスを確認するように指示する 412 エラーが表示されます。

422 未処理のエンティティ

FHIR 検証が失敗したときに返されます。

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "required",
    "diagnostics": "Bundle contains more than one preauthorization claim"
```

```
  }]
}
```

検証ルール

HealthLake は、送信に対して包括的な検証を実行します。

バンドルの検証

- PAS リクエストバンドルプロファイルに準拠する必要があります
- `Bundle.type` は である必要があります `"collection"`
- 複数のクレームリソースを含めることができます
- ただし、 には、事前承認の使用を含むクレームリソースが 1 つだけ含まれている必要があります。
  - また、このクレームリソースはバンドルの最初のエントリである必要があります。
- 参照されるすべてのリソースをバンドルに含める必要があります

クレームの検証

- PAS クレームプロファイルに準拠する必要があります
- `Claim.use` は である必要があります `"preauthorization"`
- 必須フィールド: `patient`、`insurer`、`provider`、`created`、 `priority`
- ビジネス識別子が存在し、有効である必要があります

リソースの検証

- すべてのリソースは、それぞれの PAS プロファイルに準拠する必要があります
- 必要なサポートリソースが存在する必要があります (学生、カバレッジ、組織)
- クロスリファレンスは、バンドル内で有効で解決可能である必要があります

パフォーマンス仕様

メトリクス	の仕様
バンドルサイズ制限	最大 5 MB

メトリクス	の仕様
リソース数の制限	バンドルあたり 500 リソース

必要なアクセス許可

$submit オペレーションを使用するには、FHIR で AWS Sigv4 または SMART のいずれかを使用できます。

- IAM ロールに healthlake:SubmitPreAuthClaim- オペレーションを呼び出すには

FHIR スコープでの SMART

最低限必要なスコープ:

- SMART v1: user/Claim.write & <all_resourceTypes_in_Bundle>.write
- SMART v2: user/Claim.c & <all_resourceTypes_in_Bundle>.c or system/*.*

重要な実装上の注意事項

リソースの永続性

- すべてのバンドルエントリは、データストア内の個々の FHIR リソースとして保持されます。
- お客様が指定した IDs は、指定されたときに保持されます。
- バージョン履歴は監査目的で維持されます
- 重複検出によりリソースの競合を防止

処理動作

- 有効な送信ごとに、"queued"結果を含む 1 つの ClaimResponse のみが生成されます。
- 無効な送信は、詳細なエラー情報を含む 400 または 422 のステータスコードを返す
- システムエラーが適切な 5xx ステータスコードを返す
- すべての送信が成功すると、200 ステータスが返され、ClaimResponse が保留されます。

バンドルの要件

- Bundle.entry.fullUrl 値は REST URLs または"urn:uuid:[guid]"形式である必要があります
- すべての GUIDs は送信間で一意である必要があります (同じリソースインスタンスを除く)
- 参照されたリソースはバンドル内に存在するか、解決可能である必要があります

関連の オペレーション

- Claim/$inquire - 送信された以前の承認リクエストのステータスをクエリする
- Patient/$everything - 事前認可コンテキストの包括的な患者データを取得する

を使用した FHIR リソースの検証 **$validate**

AWS HealthLake は FHIR リソースの $validateオペレーションをサポートするようになりました。これにより、ストレージオペレーションを実行せずに、FHIR 仕様に照らしてリソースを検証し、指定されたプロファイルまたはベースリソース定義への準拠を確認できます。このオペレーションは、以下が必要な場合に特に役立ちます。

- FHIR CMS コンプライアンス要件を検証する
- 本番環境で使用する前にリソースをテストする
- ユーザーが臨床データを編集するときに、リアルタイムの検証フィードバックを提供する
- APIs を作成および更新するために無効なデータ送信を減らす

使用方法

$validate オペレーションは、POST メソッドを使用して FHIR リソースで呼び出すことができます。

サポートされているオペレーション

```
POST [base]/[type]/[id]/$validate
POST [base]/[type]/$validate
```

サポートされているペイロード

パラメータリソース

HealthLake は、次の FHIR $validateパラメータをサポートしています。

パラメータ	タイプ	必須	説明
resource	[リソース]	はい	検証するリソース
profile	canonical	不可	検証するプロファイルの正規 URL
mode	コード	不可	検証モード: create、または update

クエリパラメータを使用した直接リソース

パラメータ	タイプ	必須	説明
profile	canonical	不可	検証するプロファイルの正規 URL
mode	コード	不可	検証モード: create、または update

例

ID およびパラメータペイロードを持つリソースの POST リクエスト

```
POST [base]/Patient/example-patient/$validate
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "resource",
      "resource": {
        "resourceType": "Patient",
        "id": "example-patient",
        "name": [
          {
            "family": "Smith",
```

```
            "given": ["John"]
          }
        ],
        "gender": "male",
        "birthDate": "1990-01-01"
      }
    },
    {
      "name": "profile",
      "valueCanonical": "http://hl7.org/fhir/us/core/StructureDefinition/us-core-
patient"
    },
    {
      "name": "mode",
      "valueString": "create"
    }
  ]
}
```

リソースタイプとパラメータペイロードの POST リクエスト

```
POST [base]/Patient/$validate
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "resource",
      "resource": {
        "resourceType": "Patient",
        "name": [
          {
            "family": "Doe",
            "given": ["Jane"]
          }
        ],
        "gender": "female",
        "birthDate": "1985-05-15"
      }
    },
    {
      "name": "profile",
```

```
      "valueCanonical": "http://hl7.org/fhir/us/core/StructureDefinition/us-core-
patient"
    },
    {
      "name": "mode",
      "valueString": "update"
    }
  ]
}
```

ID と直接リソースペイロードを持つリソースの POST リクエスト

```
POST [base]/Patient/example-patient/$validate?profile=http://hl7.org/fhir/us/core/
StructureDefinition/us-core-patient&mode=create
Content-Type: application/fhir+json

{
    "resourceType": "Patient",
    "id": "example-patient",
    "name": [
        {
        "family": "Smith",
        "given": ["John"]
        }
    ],
    "gender": "male",
    "birthDate": "1990-01-01"
}
```

リソースタイプと直接リソースペイロードの POST リクエスト

```
POST [base]/Patient/$validate?profile=http://hl7.org/fhir/us/core/StructureDefinition/
us-core-patient&mode=create
Content-Type: application/fhir+json

{
    "resourceType": "Patient",
    "id": "example-patient",
    "name": [
        {
        "family": "Smith",
```

```
        "given": ["John"]
        }
    ],
    "gender": "male",
    "birthDate": "1990-01-01"
}
```

レスポンス例

オペレーションは、検証結果を含む OperationOutcome リソースを返します。

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "information",
      "code": "informational",
      "diagnostics": "Validation successful"
    }
  ]
}
```

検証エラーによるレスポンスの例

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "required",
      "details": {
        "text": "Missing required element"
      },
      "diagnostics": "Patient.identifier is required by the US Core Patient profile",
      "location": [
        "Patient.identifier"
      ]
    },
    {
      "severity": "warning",
      "code": "code-invalid",
      "details": {
```

```
        "text": "Invalid code value"
      },
      "diagnostics": "The provided gender code is not from the required value set",
      "location": [
        "Patient.gender"
      ]
    }
  ]
}
```

行動

$validate オペレーション:

1. FHIR 仕様とベースリソース定義に照らしてリソースを検証します
2. profile パラメータが指定されている場合、指定されたプロファイルへの準拠をチェックします
3. 指定されたモード (create または update) に基づいて検証します。
4. エラー、警告、情報メッセージなど、詳細な検証結果を返します。
5. ストレージオペレーションを実行しない - 検証のみ
6. 検証の問題が見つかったかどうかに関係なく、検証を実行できるときに HTTP 200 OK を返します

検証モード

- create: 作成中であるかのようにリソースを検証します (新しいリソース)
- update: リソースが更新されているかのように検証します (既存のリソース)

エラー処理

オペレーションは以下を返します。

- 200 OK: 検証が正常に実行されました (検証結果に関係なく)
- 400 不正なリクエスト: 無効なリクエスト形式またはパラメータ
- 404 Not Found: リソースタイプまたはプロファイルが見つかりません

$validate オペレーション仕様の詳細については、FHIR R4 リソース$validateドキュメントを参照してください。

のコンプライアンスリファレンス AWS HealthLake

AWS HealthLake は、CMS (Centers for Medicare & Medicaid Services) の相互運用性要件に従って API の使用を追跡およびレポートするのに役立つように設計されています。これらの機能により、CMS が必須とするカテゴリ別に API トランザクションを分類し、コンプライアンスレポートのために使用状況メトリクスを自動的にキャプチャできます。

⚠ コンプライアンスの責任を理解する

AWS HealthLake とその CMS 相互運用性エンドポイントを使用することだけでは、CMS コンプライアンスを実現するには不十分です。以下の責任はユーザーにあります:

- 特定のユースケースと規制上の義務に基づいて、API ワークフローを適切な CMS カテゴリエンドポイントに正しくマッピングする
- CMS 要件を満たす適切な認証および認可コントロールの実装
- FHIR リソースとデータ交換が該当する CMS 規制と実装ガイドに準拠していることを確認する
- コンプライアンスレポートのニーズをサポートするために CloudWatch メトリクスを設定およびモニタリングする
- 組織に適用される CMS ルールを理解し、適切な技術的および運用上のコントロールを実装する

AWS HealthLake は、コンプライアンスの取り組みをサポートするインフラストラクチャとツールを提供しますが、特定の規制要件に基づいてこれらの機能を適切に使用する必要があります。これらのエンドポイントを介して API コールをルーティングするだけでは、アプリケーションが CMS 規制に自動的に準拠することはありません。

トピック

CMS コンプライアンス機能

AWS HealthLake には、CMS (Centers for Medicare & Medicaid Services) の相互運用性とコンプライアンス要件を満たすのに役立つ機能が用意されています。これらの機能により、CMS カテゴリ別の API 使用状況を追跡し、コンプライアンス上の使用状況メトリクスをレポートできます。

トピック

CMS 相互運用性エンドポイント

概要:

HealthLake には、CMS が義務付ける API カテゴリに対応する 4 つの CMS 相互運用性エンドポイントが用意されています。HealthLake データストアの基盤となるベース URL は変更されません。これらのエンドポイントは、CMS レポート目的で API コールを分類および追跡する方法を提供するだけです。

目的

これらの相互運用性エンドポイントの主な目的は、お客様が以下を実行できるようにすることです。

- CMS カテゴリ別に API トランザクションを簡単に追跡する
- CMS コンプライアンスの使用状況メトリクスを自動的にレポートする
- 最小限の変更で既存の FHIR ワークフローを維持する

すべての API コールは、相互運用性エンドポイントと標準 FHIR エンドポイントのどちらを使用する場合でも同じように機能します。唯一の違いは、トランザクションが CloudWatch メトリクスでどのように分類されるかです。

サポートされている CMS 相互運用性エンドポイント

CMS カテゴリ	相互運用性エンドポイント	使用例
患者アクセス	`/patientaccess/v2/r4`	`baseURL/patientaccess/v2/r4/Patient/123`
プロバイダーアクセス	`/provideraccess/v2/r4`	`baseURL/provideraccess/v2/r4/Observation?patient=123`
支払者から支払者へ	`/payertopayerdx/v2/r4`	`baseURL/payertopayerdx/v2/r4/Practitioner/456`

CMS カテゴリ	相互運用性エンドポイント	使用例
以前の認証サービス	/priorauthservice/v2/r4	baseURL/priorauthservice/v2/r4/ExplanationOfBenefit?patient=789

相互運用性エンドポイントの仕組み

標準 HealthLake API コール:

```
baseURL/resourceType/[id]
baseURL/resourceType?[parameters]
```

CMS 相互運用性エンドポイントの場合:

```
baseURL/interoperability-endpoint/resourceType/[id]
baseURL/interoperability-endpoint/resourceType?[parameters]
```

相互運用性エンドポイントパスは、ベース URL とリソースタイプの間に挿入されます。相互運用性エンドポイントパスの後はすべて、現在の API コールとまったく同じままです。

使用例

例 1: 患者アクセス API

現在の API コール (引き続き機能):

```
curl -X GET \
  https://healthlake.us-east-1.amazonaws.com/datastore/abc123/r4/Patient/123 \
  -H "Authorization: Bearer <token>" \
  -H "Content-Type: application/fhir+json"
```

患者アクセス相互運用性エンドポイント (CMS 追跡用) の場合:

```
curl -X GET \
  https://healthlake.us-east-1.amazonaws.com/datastore/abc123/patientaccess/v2/r4/
Patient/123 \
  -H "Authorization: Bearer <token>" \
  -H "Content-Type: application/fhir+json"
```

キーポイント:

- 基本 URL は残ります。 `https://healthlake.us-east-1.amazonaws.com/datastore/abc123`
- 相互運用性エンドポイントが挿入されました。 `/patientaccess/v2/r4`
- リソースパスが変更されていません。 `/Patient/123`
- 両方の呼び出しが同一のレスポンスを返す
- 相互運用性エンドポイント呼び出しは、CloudWatch `URIType=patient-access`の で自動的に追跡されます。
- POST、PUT、PATCH、DELETE オペレーションは同じように機能します。
- リクエスト本文は変更されません。

エンドポイント翻訳リファレンス

相互運用性エンドポイント	への翻訳	CMS カテゴリ
`baseURL/patientaccess/v2/r4/Patient`	`baseURL/r4/Patient`	患者アクセス
`baseURL/provideraccess/v2/r4/Observation`	`baseURL/r4/Observation`	プロバイダーアクセス
`baseURL/payertopayerdx/v2/r4/Practitioner/456`	`baseURL/r4/Practitioner/456`	Payer to Payer Data Exchange
`baseURL/priorauthservice/v2/r4/ExplanationOfBenefit?patient=789`	`baseURL/r4/ExplanationOfBenefit?patient=789`	事前認可

重要な注意事項

- 機能的な違いなし: 相互運用性エンドポイントと標準 FHIR エンドポイントは、同一のレスポンスを返し、同一のオペレーションをサポートします
- 基本 URL 変更なし: HealthLake データストアエンドポイントは変わりません

- シンプルな統合: 基本 URL とリソースタイプの間に相互運用性エンドポイントパスを挿入する
- 自動追跡: CloudWatch メトリクスは、使用される相互運用性エンドポイントによって呼び出しを自動的に分類します。
- 下位互換性: 相互運用性エンドポイントのない既存の API コールは引き続き正常に動作します。

CMS コンプライアンスの CloudWatch メトリクスの強化

概要:

CMS 相互運用性エンドポイントを使用すると、HealthLake は CMS レポート要件をサポートする追加のディメンションを含む拡張 CloudWatch メトリクスを自動的に出力します。これらのメトリクスは、追加の設定を必要とせずに、発信者 ID、アプリケーション、および CMS 固有の URI タイプごとに API の使用状況を追跡します。

仕組み

相互運用性エンドポイントを使用して API コールを行う場合:

```
# This call...
curl https://healthlake.us-east-1.amazonaws.com/datastore/abc123/patientaccess/v2/r4/
Patient/123

# Automatically generates metrics with:
# - URIType: "patient-access"
# - Sub: extracted from your bearer token (SMART on FHIR datastores only)
# - ClientId: extracted from your bearer token (SMART on FHIR datastores only)
# - Plus all standard dimensions (DatastoreId, Operation, etc.)
```

追加のコードや設定は必要ありません。相互運用性エンドポイントを使用するだけで、拡張メトリクスが自動的にキャプチャされます。

> **Note**
>
> FHIR データストアの非 SMART の場合、`URIType`ディメンションは引き続きキャプチャされるため、CMS カテゴリ別に API の使用状況を追跡できます。`Sub` および `ClientId`ディメンションは、これらのクレームを含むベアラートークンで FHIR 認証で SMART を使用する場合にのみ使用できます。

新しいメトリクスディメンション

既存のディメンション (DatastoreId、DatastoreType、Operation) に加えて、相互運用性エンドポイントを使用すると、次のディメンションが自動的に追加されます。

ディメンション	説明	値の例	ソース
URIType	CMS コンプライアンスカテゴリ	`patient-access` , `provider-access` , `payer-to-payer` , `prior-authorization`	相互運用性エンドポイントパスから自動的に決定される
サブ	発信者 ID	ユーザー/エンティティ識別子	ベアラートークンsubクレームから抽出
ClientId	アプリケーション識別子	`portal_app` , `ehr_system`	ベアラートークンclient_id クレームから抽出

使用可能なメトリクス

相互運用性エンドポイントを使用する場合、既存のすべての HealthLake メトリクスに追加のディメンションが含まれるようになりました。

- CallCount - API コールの合計数
- レイテンシー - ミリ秒単位の API 応答時間
- UserErrors - 4xx クライアントエラーの数
- SystemErrors - 5xx サーバーエラーの数
- スロットリング - スロットリングされたリクエストの数
- SuccessfulRequests - 成功した API コールの数

CloudWatch でのメトリクスのクエリ

CloudWatch Insights クエリの例

アプリケーションごとにすべての Patient Access API コールをクエリします。

```
SELECT SUM(CallCount)
FROM "AWS/HealthLake"
WHERE DatastoreId = '75c1cf9b0d71cd38fec8f7fb317c4c1a'
    AND URIType = 'patient-access'
GROUP BY ClientId
```

のサポートリファレンス AWS HealthLake

以下のサポートリファレンス資料が利用可能です AWS HealthLake。

> **Note**
>
> すべてのネイティブ HealthLake アクションとデータ型については、別のリファレンスで説明しています。詳細については、「AWS HealthLake APIリファレンス」を参照してください。

トピック

- AWS HealthLake エンドポイントとクォータ
- HealthLake の Synthea プリロードされたデータ型
- AWS HealthLake サンプルプロジェクト
- トラブルシューティング AWS HealthLake
- AWS SDK での HealthLake の使用

AWS HealthLake エンドポイントとクォータ

以下のトピックでは、 AWS HealthLake 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、より高速な接続を可能にするために、特定のリージョンのエンドポイントを提供します。次の表に、 のサービスエンドポイントを示します AWS HealthLake。

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 (オハイオ)	us-east-2	healthlake.us-east-2.amazonaws.com healthlake-fips.us-east-2.amazonaws.com	HTTPS HTTPS
米国東部 (バージニア北部)	us-east-1	healthlake.us-east-1.amazonaws.com healthlake-fips.us-east-1.amazonaws.com	HTTPS HTTPS
米国西部 (オレゴン)	us-west-2	healthlake.us-west-2.amazonaws.com healthlake-fips.us-west-2.amazonaws.com	HTTPS HTTPS
アジアパシフィック (ムンバイ)	ap-south-1	healthlake.ap-south-1.amazonaws.com	HTTPS
アジアパシフィック (シドニー)	ap-southeast-2	healthlake.ap-southeast-2.amazonaws.com	HTTPS
カナダ (中部)	ca-central-1	healthlake.ca-central-1.amazonaws.com	HTTPS
欧州 (アイルランド)	eu-west-1	healthlake.eu-west-1.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 (ロンドン)	eu-west-2	healthlake.eu-west-2.amazonaws.com	HTTPS	

Service Quotas

サービスクォータは、 AWS 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。

> **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「クォータの引き上げのリクエスト」を参照してください。
> Sync Write API レートはペイロードサイズに応じて比例的に増加し、1KB の増分ごとに追加の容量が消費されます (例えば、4KB のペイロードは 4 倍の書き込み容量を使用します)。オプションの `x-amz-fhir-history-consistency-level` ヘッダーを に設定すると、リソースあたりの書き込みキャパシティ消費量が 2 `strong`倍になります。
> バンドル内のリソースは、1 KB のペイロードサイズに基づく標準の読み取り/書き込み制限に従います。バンドルタイプのトランザクションは、タイプバッチと比較して書き込み容量を 2 倍消費します。つまり、バッチバンドルはトランザクションバンドルの 1 秒あたり 2 倍のリソースを処理できます。

次の表に、 のデフォルトのクォータを示します AWS HealthLake。

名前	デフォルト	引き上げ可能	説明
アカウントあたりのアクティブなサブスクリプションの数	サポートされている各リージョン: 100	可能	アカウントあたりのアクティブなサブスクリプションリソースの最大数。
データストアあたりのアクティブなサブスクリプションリソースの数	サポートされている各リージョン: 50	可能	データストアあたりのアクティブなサブスクリプションリソースの最大数。
アカウントあたりのアクティブな SubscriptionTopic の数	サポートされている各リージョン: 100	可能	アカウントあたりのアクティブな SubscriptionTopic リソースの最大数。
データストアあたりのアクティブな SubscriptionTopic リソースの数	サポートされている各リージョン: 50	可能	データストアあたりのアクティブな SubscriptionTopic リソースの最大数。
医療メモの文字数	サポートされている各リージョン: 10,000	いいえ	DocumentReference リソースタイプ (POST/PUT リクエスト) 内の個々の医療メモの最大文字数 。
StartFHIRExportJob 同時ジョブの数	サポートされている各リージョン: 1	[いいえ]	StartFHIRExportJob 同時ジョブの最大数。

名前	デフォルト	引き上げ可能	説明
StartFHIRImportJob 同時ジョブの数	サポートされている各リージョン: 1	[いいえ]	StartFHIRImportJob 同時ジョブの最大数。
アカウントごとのデータストアの数	サポートされている各リージョン: 10	あり	アカウントごとの、アクティブなデータストアのデフォルトの最大数。
StartFHIRImportJob 内のファイル数	サポートされている各リージョン: 1,000,000	あり	StartFHIRImportJob 内の最大ファイル数。
バンドルあたりのリソース数	サポートされている各リージョン: 500	いいえ	バンドルリクエストにおけるリソースの最大数。
アカウントごとの、DELETE を使用した CancelFHIRExportJob リクエストのレート	サポートされている各リージョン: 1	[いいえ]	アカウントごとに 1 秒あたりに実行できる、DELETE を使用した CancelFHIRExportJob リクエストの最大数。
アカウントごとの CreateFHIRDatastore リクエストのレート	サポートされている各リージョン: 1	[いいえ]	アカウントごとに 1 分あたりに実行できる CreateFHIRDatastore リクエストの最大数。
アカウントごとの DeleteFHIRDatastore リクエストのレート	サポートされている各リージョン: 1	[いいえ]	アカウントごとに 1 分あたりに実行できる DeleteFHIRDatastore リクエストの最大数。

名前	デフォルト	引き上げ可能	説明
アカウントあたりの DescribeFHIRBulkDeleteJob リクエストのレート	サポートされている各リージョン: 10	あり	アカウントごとに 1 秒あたりに実行できる DescribeFHIRBulkDeleteJob リクエストの最大数。
データストアあたりの DescribeFHIRBulkDeleteJob リクエストのレート	サポートされている各リージョン: 10	あり	データストアごとに 1 秒あたりに実行できる DescribeFHIRBulkDeleteJob リクエストの最大数。
アカウントごとの DescribeFHIRDatastoreリクエストのレート	サポートされている各リージョン: 10	いいえ	アカウントごとに 1 秒あたりに実行できる DescribeFHIRDatastore リクエストの最大数。
アカウントごとの DescribeFHIRExportJob リクエストのレート	サポートされている各リージョン: 10	いいえ	アカウントごとに 1 秒あたりに実行できる DescribeFHIRExportJob リクエストの最大数。
アカウントごとの、GET を使用した DescribeFHIRExportJob リクエストのレート	サポートされている各リージョン: 10	いいえ	アカウントごとに 1 秒あたりに実行できる、GET を使用した DescribeFHIRExportJob リクエストの最大数。

名前	デフォルト	引き上げ可能	説明
アカウントごとの DescribeFHIRImportJob リクエストのレート	サポートされている各リージョン: 10	いいえ	アカウントごとに 1 秒あたりに実行できる DescribeFHIRImportJob リクエストの最大数。
アカウントごとの Discovery リクエストのレート	サポートされている各リージョン: 10	いいえ	アカウントごとに 1 分あたりに実行できる Discovery リクエストの最大数。
アカウントごとの GET リクエストのレート	サポートされている各リージョン: 6,000	あり	アカウントごとに 1 秒あたりに実行できる GET リクエストの最大数。
データストアごとの GET リクエストのレート	サポートされている各リージョン: 3,000	あり	データストアごとに 1 秒あたりに実行できる GET リクエストの最大数。2023 年 8 月 21 日より前に作成されたデータストアの場合、1 秒あたり 100 リクエストに制限されます。
アカウントごとの GetCapabilities リクエストのレート	サポートされている各リージョン: 10	いいえ	アカウントごとに 1 秒あたりに実行できる GetCapabilities リクエストの最大数。

名前	デフォルト	引き上げ可能	説明
データストアごとの GetExportedFile リクエストのレート	サポートされている各リージョン: 10	いいえ	データストアごとに 1 秒あたりに実行できる GetExportedFile リクエストの最大数。
アカウントごとの ListFHIRDatastores リクエストのレート	サポートされている各リージョン: 10	いいえ	アカウントごとに 1 秒あたりに実行できる ListFHIRDatastores リクエストの最大数。
アカウントごとの ListFHIRExportJobs リクエストのレート	サポートされている各リージョン: 10	いいえ	アカウントごとに 1 秒あたりに実行できる ListFHIRExportJobs リクエストの最大数。
アカウントごとの ListFHIRImportJobs リクエストのレート	サポートされている各リージョン: 10	いいえ	アカウントごとに 1 秒あたりに実行できる ListFHIRImportJobs リクエストの最大数。
アカウントごとの ListTagsforResource リクエストのレート	サポートされている各リージョン: 10	いいえ	アカウントごとに 1 秒あたりに実行できる ListTagsforResource リクエストの最大数。
アカウントあたりの SearchEverything リクエストのレート	サポートされている各リージョン: 10	あり	アカウントごとに 1 秒あたりに実行できる SearchEverything リクエストの最大数。

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 SearchEverything リクエストのレート	サポートされている各リージョン: 10	あり	データストアごとに 1 秒あたりに実行できる SearchEverything リクエストの最大数。
アカウントあたりの StartFHIRBulkDeleteJob リクエストのレート	サポートされている各リージョン: 10	あり	アカウントごとに 1 秒あたりに実行できる StartFHIRBulkDeleteJob リクエストの最大数。
データストアあたりの StartFHIRBulkDeleteJob リクエストのレート	サポートされている各リージョン: 10	あり	データストアごとに 1 秒あたりに実行できる StartFHIRBulkDeleteJob リクエストの最大数。
アカウントあたりの StartFHIRBulkMemberMatchJob リクエストのレート	サポートされている各リージョン: 10	あり	アカウントごとに 1 秒あたりに実行できる StartFHIRBulkMemberMatchJob リクエストの最大数。
データストアあたりの StartFHIRBulkMemberMatchJob リクエストのレート	サポートされている各リージョン: 10	あり	データストアごとに 1 秒あたりに実行できる StartFHIRBulkMemberMatchJob リクエストの最大数。
アカウントごとの StartFHIRExportJob リクエストのレート	サポートされている各リージョン: 1	[いいえ]	アカウントごとに 1 秒あたりに実行できる StartFHIRExportJob リクエストの最大数。

名前	デフォルト	引き上げ可能	説明
アカウントごとの、GET を使用した StartFHIRExportJob リクエストのレート	サポートされている各リージョン: 1	[いいえ]	アカウントごとに 1 秒あたりに実行できる、GET を使用した StartFHIRExportJob リクエストの最大数。
アカウントごとの、POST を使用した StartFHIRExportJob リクエストのレート	サポートされている各リージョン: 1	[いいえ]	アカウントごとに 1 秒あたりに実行できる、POST を使用した StartFHIRExportJob リクエストの最大数。
アカウントごとの StartFHIRImportJob リクエストのレート	サポートされている各リージョン: 25	いいえ	アカウントごとに 1 秒あたりに実行できる StartFHIRImportJob リクエストの最大数。
アカウントごとの TagResource リクエストのレート	サポートされている各リージョン: 10	いいえ	アカウントごとに 1 秒あたりに実行できる TagResource リクエストの最大数。
アカウントごとの UntagResource リクエストのレート	サポートされている各リージョン: 10	いいえ	アカウントごとに 1 秒あたりに実行できる UntagResource リクエストの最大数。

名前	デフォルト	引き上げ可能	説明
アカウントあたりの ValidateResource リクエストのレート	サポートされている各リージョン: 2,000	あり	アカウントごとに 1 秒あたりに実行できる ValidateResource リクエストの最大数。2023 年 8 月 21 日より前に作成されたデータストアの場合、1 秒あたり 300 リクエストに制限されます。
データストアあたりの ValidateResource リクエストのレート	サポートされている各リージョン: 1,000	あり	データストアごとに 1 秒あたりに実行できる ValidateResource リクエストの最大数。2023 年 8 月 21 日より前に作成されたデータストアの場合、1 秒あたり 300 リクエストに制限されます。
アカウントごとの WRITE リクエストのレート	サポートされている各リージョン: 6,000	あり	アカウントごとに 1 秒あたりに実行できる CREATE|UPDATE|DELETE リクエストの最大数。

名前	デフォルト	引き上げ可能	説明
データストアごとの WRITE リクエストのレート	サポートされている各リージョン: 3,000	あり	データストアごとに 1 秒あたりに実行できる CREATE|UPDATE|DELETE リクエストの最大数。2023 年 8 月 21 日より前に作成されたデータストアの場合、1 秒あたり 300 リクエストに制限されます。
アカウントごとの、GET を使用した検索リクエストのレート	サポートされている各リージョン: 200	あり	アカウントごとの、1 秒あたりに実行できる GET を使用した検索リクエストの最大数。
データストアごとの、GET を使用した検索リクエストのレート	サポートされている各リージョン: 100	可能	データストアごとの、1 秒あたりに実行できる GET を使用した検索リクエストの最大数。
アカウントごとの、POST を使用した検索リクエストのレート	サポートされている各リージョン: 200	あり	1アカウントごとに 1 秒あたりに実行できる、POST を使用した検索リクエストの最大数。
データストアごとの、POST を使用した検索リクエストのレート	サポートされている各リージョン: 100	可能	データストアごとに 1 秒あたりに実行できる、POST を使用した検索リクエストの最大数。

名前	デフォルト	引き上げ可能	説明
インポートされた個々のファイルサイズ	サポートされている各リージョン: 50 GB	いいえ	StartFHIRImportJob に含まれる個々のファイルの最大サイズ (GB) 。
データストアあたりのキューに入れられた非同期バンドルトランザクションの合計	サポートされている各リージョン: 500	あり	任意の時点でのデータストアあたりのキューに入れられた非同期バンドルトランザクションの最大数。
データストアあたりのキューに入れられる一括エクスポートジョブの合計	サポートされている各リージョン: 25	可能	任意の時点でのデータストアあたりのキューに入れられる一括エクスポートジョブの最大数。
データストアあたりのキューに入れられる一括インポートジョブの合計	サポートされている各リージョン: 25	可能	任意の時点でのデータストアあたりのキューに入れられる一括インポートジョブの最大数。
インポートジョブの総サイズ	サポートされている各リージョン: 5,000 GB	あり	インポートジョブに含まれるファイルの最大サイズ (GB)。

HealthLake の Synthea プリロードされたデータ型

HealthLake は、事前ロードされたデータ型として SYNTHEA のみをサポートします。Synthea は、`Patient`医療履歴をモデル化する合成患者ジェネレーターです。これは、ユーザーが実際の患者データを使用せずにモデルをテスト`Bundle`できるように、HealthLake が FHIR R4-compliantのリソースを生成できるようにするオープンソースの Git リポジトリでホストされています。

次のリソースタイプは、プリロードされた HealthLake データストアで使用できます。Synthea データを使用した HealthLake データストアの事前ロードの詳細については、「」を参照してくださいHealthLake データストアの作成。

> **Note**
>
> HealthLake がサポートする FHIR R4 リソースの完全なリストを表示するには、「」を参照してくださいHealthLake でサポートされている FHIR R4 リソースタイプ。

HealthLake でサポートされている Synthea FHIR リソースタイプ

AllergyIntolerance	ロケーション
CarePlan	MedicationAdministration
CareTeam	MedicationRequest
Claim	監視結果
Condition	組織
デバイス	患者
DiagnosticReport	プラクティショナー
エンカウンター	PractitionerRole
ExplanationofBenefit	手順
ImagingStudy	プロベナンス
イミュナイゼーション	

AWS HealthLake サンプルプロジェクト

分析をさらに進めるには、次のブログ記事の例に示すように、HealthLake を他の AWS のサービスで使用できます。

HealthLake 統合分析

- を使用した母集団ヘルスアプリケーション AWS HealthLake – パート 1: Amazon Quick を使用した分析とモニタリング。
- AWS HealthLake 正規化されたデータで Amazon SageMaker AI を使用して予測疾患モデルを構築する。
- AWS AI サービスを使用して、認知検索とヘルスナレッジグラフを構築します。

HealthLake イベントモニタリング

- Amazon EventBridge と の統合 AWS HealthLake。

トラブルシューティング AWS HealthLake

以下のトピックでは、、 AWS SDKs AWS CLI、または HealthLake コンソールの使用時に発生する可能性のあるエラーや問題のトラブルシューティングに関するアドバイスを提供します。このセクションに記載されていない問題が見つかった場合は、このページの右側のサイドバーにあるフィードバックの提供ボタンを使用して報告します。

トピック

- データストアアクション
- インポートアクション
- FHIR APIs
- NLP 統合
- SQL 統合

データストアアクション

問題: HealthLake データストアを作成しようとすると、次のエラーが表示されます。

```
AccessDeniedException: Insufficient Lake Formation permission(s): Required Database on
 Catalog
```

2022 年 11 月 14 日、HealthLake は新しいデータストアを作成するために必要な IAM アクセス許可を更新しました。詳細については、「HealthLake を使用するように IAM ユーザーまたはロールを設定する (IAM 管理者)」を参照してください。

問題: AWS SDKs を使用して HealthLake データストアを作成すると、データストアの作成ステータスは例外または不明なステータスを返します。

DescribeFHIRDatastore または ListFHIRDatastores API 呼び出しが例外または不明なデータストアステータスを返す場合は、 AWS SDK を最新バージョンに更新します。

インポートアクション

問題: データが FHIR R4 形式でない場合でも HealthLake を使用できますか?

HealthLake データストアにインポートできるのは、FHIR R4 形式のデータのみです。既存のヘルスデータを FHIR R4 形式に変換するのに役立つパートナーのリストについては、AWS HealthLake「 パートナー」を参照してください。

問題: FHIR インポートジョブが失敗したのはなぜですか?

インポートジョブが成功すると、 .ndjson形式の結果 (出力ログ) を含むフォルダが生成されますが、個々のレコードはインポートに失敗する可能性があります。この場合、インポートに失敗したレコードのマニフェストを含む 2 番目のFAILUREフォルダが生成されます。詳細については、「を使用した FHIR データのインポート AWS HealthLake」を参照してください。

インポートジョブが失敗した理由を分析するには、 DescribeFHIRImportJob API を使用して JobProperties を分析します。以下が推奨されます。

- ステータスが FAILEDで、メッセージが存在する場合、失敗は入力データサイズや HealthLake クォータを超える入力ファイルの数などのジョブパラメータに関連しています。
- インポートジョブのステータスが の場合はCOMPLETED_WITH_ERRORS、マニフェストファイルをチェックしてmanifest.json、正常にインポートされなかったファイルに関する情報を確認してください。
- インポートジョブのステータスが FAILEDで、メッセージが存在しない場合は、ジョブの出力場所に移動してマニフェストファイル にアクセスしますmanifest.json。

入力ファイルごとに、インポートに失敗したリソースの入力ファイル名を持つ失敗出力ファイルがあります。レスポンスには、入力データの場所に対応する行番号 (lineId)、FHIR レスポンスオブジェクト (UpdateResourceResponse)、およびレスポンスのステータスコード (statusCode) が含まれます。

サンプル出力ファイルは次のようになります。

{"lineId":3, UpdateResourceResponse:{"jsonBlob":
{"resourceType":"OperationOutcome","issue":
[{"severity":"error","code":"processing","diagnostics":"1 validation error detected:
 Value 'Patient123' at 'resourceType' failed to satisfy constraint: Member must satisfy
 regular expression pattern: [A-Za-z]{1,256}"}]}, "statusCode":400}
{"lineId":5, UpdateResourceResponse:{"jsonBlob":
{"resourceType":"OperationOutcome","issue":
[{"severity":"error","code":"processing","diagnostics":"This property must be an
 simple value, not a com.google.gson.JsonArray","location":["/EffectEvidenceSynthesis/
name"]},{"severity":"error","code":"processing","diagnostics":"Unrecognised
 property '@telecom'","location":["/EffectEvidenceSynthesis"]},
{"severity":"error","code":"processing","diagnostics":"Unrecognised
 property '@gender'","location":["/EffectEvidenceSynthesis"]},
{"severity":"error","code":"processing","diagnostics":"Unrecognised
 property '@birthDate'","location":["/EffectEvidenceSynthesis"]},
{"severity":"error","code":"processing","diagnostics":"Unrecognised
 property '@address'","location":["/EffectEvidenceSynthesis"]},
{"severity":"error","code":"processing","diagnostics":"Unrecognised
 property '@maritalStatus'","location":["/EffectEvidenceSynthesis"]},
{"severity":"error","code":"processing","diagnostics":"Unrecognised
 property '@multipleBirthBoolean'","location":["/EffectEvidenceSynthesis"]},
{"severity":"error","code":"processing","diagnostics":"Unrecognised
 property '@communication'","location":["/EffectEvidenceSynthesis"]},
{"severity":"warning","code":"processing","diagnostics":"Name should be usable as an
 identifier for the module by machine processing applications such as code generation
 [name.matches('[A-Z]([A-Za-z0-9_]){0,254}')]","location":["EffectEvidenceSynthesis"]},
{"severity":"error","code":"processing","diagnostics":"Profile http://hl7.org/fhir/
StructureDefinition/EffectEvidenceSynthesis, Element 'EffectEvidenceSynthesis.status':
 minimum required = 1, but only found 0","location":["EffectEvidenceSynthesis"]},
{"severity":"error","code":"processing","diagnostics":"Profile
 http://hl7.org/fhir/StructureDefinition/EffectEvidenceSynthesis,
 Element 'EffectEvidenceSynthesis.population': minimum required
 = 1, but only found 0","location":["EffectEvidenceSynthesis"]},
{"severity":"error","code":"processing","diagnostics":"Profile
 http://hl7.org/fhir/StructureDefinition/EffectEvidenceSynthesis,
 Element 'EffectEvidenceSynthesis.exposure': minimum required =
 1, but only found 0","location":["EffectEvidenceSynthesis"]},
{"severity":"error","code":"processing","diagnostics":"Profile http://
hl7.org/fhir/StructureDefinition/EffectEvidenceSynthesis, Element
 'EffectEvidenceSynthesis.exposureAlternative': minimum required
 = 1, but only found 0","location":["EffectEvidenceSynthesis"]},
{"severity":"error","code":"processing","diagnostics":"Profile http://hl7.org/fhir/
StructureDefinition/EffectEvidenceSynthesis, Element 'EffectEvidenceSynthesis.outcome':

```
 minimum required = 1, but only found 0","location":["EffectEvidenceSynthesis"]},
{"severity":"information","code":"processing","diagnostics":"Unknown
 extension http://synthetichealth.github.io/synthea/disability-adjusted-
life-years","location":["EffectEvidenceSynthesis.extension[3]"]},
{"severity":"information","code":"processing","diagnostics":"Unknown extension
 http://synthetichealth.github.io/synthea/quality-adjusted-life-years","location":
["EffectEvidenceSynthesis.extension[4]"]}]}, "statusCode":400}
{"lineId":7, UpdateResourceResponse:{"jsonBlob":
{"resourceType":"OperationOutcome","issue":
[{"severity":"error","code":"processing","diagnostics":"2 validation errors detected:
 Value at 'resourceId' failed to satisfy constraint: Member must satisfy regular
 expression pattern: [A-Za-z0-9-.]{1,64}; Value at 'resourceId' failed to satisfy
 constraint: Member must have length greater than or equal to 1"}]}, "statusCode":400}
{"lineId":9, UpdateResourceResponse:{"jsonBlob":
{"resourceType":"OperationOutcome","issue":
[{"severity":"error","code":"processing","diagnostics":"Missing required id field in
 resource json"}]}, "statusCode":400}
{"lineId":15, UpdateResourceResponse:{"jsonBlob":
{"resourceType":"OperationOutcome","issue":
[{"severity":"error","code":"processing","diagnostics":"Invalid JSON found in input
 file"}]}, "statusCode":400}
```


上記の例は、入力ファイルから対応する入力行の 3、4、7、9、15 行で障害が発生したことを示しています。これらの各行の説明は次のとおりです。

- 行 3 で、入力ファイルの行 3 でresourceType指定された が有効でないことがレスポンスで説明されています。
- 5 行目には、入力ファイルの 5 行目に FHIR 検証エラーがあることがレスポンスで説明されています。
- 7 行目のレスポンスでは、入力としてresourceId提供された の検証の問題があることが説明されています。
- 9 行目のレスポンスでは、入力ファイルには有効なリソース ID が含まれている必要があることが説明されています。
- 15 行目で、入力ファイルのレスポンスは、ファイルが有効な JSON 形式ではないことです。

FHIR APIs

問題: FHIR RESTful APIs の認可を実装するにはどうすればよいですか?

使用する データストア認可戦略を決定します。

を使用して SigV4 認可を作成するには AWS SDK for Python (Boto3)、次の例のようなスクリプトを作成します。

```
import boto3
import requests
import json
from requests_auth_aws_sigv4 import AWSSigV4

# Set the input arguments
data_store_endpoint = 'https://healthlake.us-east-1.amazonaws.com/datastore/<datastore
 id>/r4//'
resource_path = "Patient"
requestBody = {"resourceType": "Patient", "active": True, "name": [{"use":
 "official","family": "Dow","given": ["Jen"]},{"use": "usual","given":
 ["Jen"]}],"gender": "female","birthDate": "1966-09-01"}
region = 'us-east-1'

#Frame the resource endpoint
resource_endpoint = data_store_endpoint+resource_path
session = boto3.session.Session(region_name=region)
client = session.client("healthlake")

# Frame authorization
auth = AWSSigV4("healthlake", session=session)

# Call data store FHIR endpoint using SigV4 auth

r = requests.post(resource_endpoint, json=requestBody, auth=auth, )
print(r.json())
```

問題: 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に FHIR RESTful APIs を使用するときに*AccessDenied*エラーが表示されるのはなぜですか?

ユーザーまたはロールがデータストアにアクセスするには、カスタマーマネージドキーと IAM ポリシーの両方に対するアクセス許可が必要です。ユーザーは、カスタマーマネージドキーを使用するために必要な IAM アクセス許可を持っている必要があります。ユーザーがカスタマーマネージド KMS キーを使用するアクセス許可を HealthLake に付与する許可を取り消したり、廃止したりすると、HealthLake はAccessDeniedエラーを返します。

HealthLake には、顧客データにアクセスし、データストアにインポートされた新しい FHIR リソースを暗号化し、リクエストされたときに FHIR リソースを復号するためのアクセス許可が必要です。詳細については、「アクセス許可のトラブルシューティング AWS KMS」を参照してください。

問題: 10MB のドキュメントを使用して HealthLake への FHIR *POST* API オペレーションが*413 Request Entity Too Large*エラーを返しています。

AWS HealthLake の同期 Create and Update API 制限は 5MB で、レイテンシーとタイムアウトの増加を回避できます。一括インポート API を使用して、 `Binary`リソースタイプを使用して最大 164MB の大きなドキュメントを取り込むことができます。

NLP 統合

問題: HealthLake の統合自然言語処理機能を有効にするにはどうすればよいですか?

2022 年 11 月 14 日現在、HealthLake データストアのデフォルトの動作が変更されています。

現在のデータストア: 現在の HealthLake データストアはすべて、base64 でエンコードされた`DocumentReference`リソースでの自然言語処理 (NLP) の使用を停止します。つまり、新しい`DocumentReference`リソースは NLP を使用して分析されず、リソースタイプのテキストに基づいて新しい`DocumentReference`リソースは生成されません。既存の`DocumentReference`リソースの場合、NLP を介して生成されたデータとリソースは残りますが、2023 年 2 月 20 日以降は更新されません。

新しいデータストア: 2023 年 2 月 20 日以降に作成された HealthLake データストアは、base64 でエンコードされた`DocumentReference`リソースで自然言語処理 (NLP) を実行しません。

HealthLake NLP 統合を有効にするには、 を使用してサポートケースを作成しますAWS Support Center Console。ケースを作成するには、 にログインし AWS アカウント、ケースの作成を選択します。ケースとケース管理の作成の詳細については、「 サポート ユーザーガイド」の「サポートケースとケース管理の作成」を参照してください。

問題: >統合 NLP で処理できなかった*DocumentReference*リソースを見つけるにはどうすればよいですか?

`DocumentReference` リソースが有効でない場合、HealthLake は統合された医療 NLP 出力で提供するのではなく、検証エラーを示す拡張機能を提供します。NLP 処理中に検証エラーの原因となった`DocumentReference`リソースを見つけるには、HealthLake の FHIR `search`関数を検索キー cm-decoration-status および検索値 VALIDATION_ERROR とともに使用できます。この検索では、検証エラーの原因となったすべての`DocumentReference`リソースと、エラーの性質を説明するエラーメッセージが一覧表示されます。検証エラーがある`DocumentReference`リソースの拡張フィールドの構造は、次の例のようになります。

```
"extension": [
```

```
        {
            "extension": [
                {
                    "url": "http://healthlake.amazonaws.com/aws-cm/status/",
                    "valueString": "VALIDATION_ERROR"
                },
                {
                    "url": "http://healthlake.amazonaws.com/aws-cm/message/",
                    "valueString": "Resource led to too many nested objects after NLP
 operation processed the document. 10937 nested objects exceeds the limit of 10000."
                }
            ],
            "url": "http://healthlake.amazonaws.com/aws-cm/"
        }
    ]
```

> **Note**
>
> は、NLP デコレーションによって 10,000 個を超えるネストされたオブジェクトが作成された場合にも発生するVALIDATION_ERROR可能性があります。この場合、処理する前にドキュメントを小さなドキュメントに分割する必要があります。

SQL 統合

問題: 新しいデータレイク管理者を追加する*permissions error: lakeformation:PutDataLakeSettings*ときに Lake Formation を取得するのはなぜですか?

IAM ユーザーまたはロールに AWSLakeFormationDataAdmin AWS 管理ポリシーが含まれている場合、新しいデータレイク管理者を追加することはできません。以下を含むエラーが表示されます。

```
User arn:aws:sts::111122223333:assumed-role/lakeformation-admin-user is not authorized
 to perform: lakeformation:PutDataLakeSettings on resource: arn:aws:lakeformation:us-
east-2:111122223333:catalog:111122223333 with an explicit deny in an identity-based
 policy
```

Lake AWS Formation データレイク管理者として IAM ユーザーまたはロールを追加するには、 AWS 管理ポリシーAdministratorAccessが必要です。IAM ユーザーまたはロールにも アクションが含まれているAWSLakeFormationDataAdmin場合、アクションは失敗します。AWSLakeFormationDataAdmin AWS 管理ポリシー

には、 AWS Lake Formation API オペレーション の明示的な拒否が含まれていますPutDataLakeSetting。AdministratorAccess 管理ポリシー AWS を使用するためのフルアクセス権を持つ管理者でも、AWSLakeFormationDataAdminポリシーによって制限できます。

問題: Amazon Athena SQL 統合を使用するように既存の HealthLake データストアを移行するにはどうすればよいですか?

2022 年 11 月 14 日より前に作成された HealthLake データストアは機能しますが、SQL を使用して Athena でクエリすることはできません。Athena で既存のデータストアをクエリするには、まず新しいデータストアに移行する必要があります。

HealthLake データを新しいデータストアに移行するには

1. 新しいデータストアを作成します。
2. 既存の から Amazon S3 バケットにデータをエクスポートします。
3. Amazon S3 バケットから新しいデータストアにデータをインポートします。

> **Note**
>
> Amazon S3 バケットにデータをエクスポートすると、追加料金が発生します。追加料金は、エクスポートするデータのサイズによって異なります。

問題: SQL 統合用の新しい HealthLake データストアを作成する場合、データストアのステータスはから変更されませんCreating。

新しい HealthLake データストアを作成しようとしたときに、データストアのステータスが「作成中」から変更されない場合は、 を使用するように Athena を更新する必要があります AWS Glue Data Catalog。詳細については、「Amazon Athena ユーザーガイド」の「 AWS Glue データカタログへのstep-by-stepのアップグレード」を参照してください。 Amazon Athena

を正常にアップグレードしたら AWS Glue Data Catalog、HealthLake データストアを作成できます。

古い HealthLake データストアを削除するには、 を使用してサポートケースを作成しますAWS Support Center Console。ケースを作成するには、 にログインし AWS アカウント、ケースの作成を選択します。詳細については、「 サポート ユーザーガイド」の「サポートケースとケース管理の作成」を参照してください。

問題: 新しい HealthLake データストアにデータをインポートした後、Athena コンソールが機能しない

新しい HealthLake データストアにデータをインポートすると、そのデータはすぐに使用できない場合があります。これは、データが Apache Iceberg テーブルに取り込まれる時間を確保するためです。後でもう一度試してください。

問題: Athena の検索結果を他の AWS サービスに接続するにはどうすればよいですか?

Athena の検索結果を他の AWS サービスと共有する場合、SQL 検索クエリ`json_extract[1]`の一部として を使用すると問題が発生する可能性があります。この問題を修正するには、 を に更新する必要がありますCATVAR。

保存結果、テーブル (静的) 、またはビュー (動的) を作成しようとすると、この問題が発生する可能性があります。

AWS SDK での HealthLake の使用

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

SDK ドキュメント	コード例
AWS SDK for C++	AWS SDK for C++ コード例
AWS CLI	AWS CLI コード例
AWS SDK for Go	AWS SDK for Go コード例
AWS SDK for Java	AWS SDK for Java コード例
AWS SDK for JavaScript	AWS SDK for JavaScript コード例
AWS SDK for Kotlin	AWS SDK for Kotlin コード例
AWS SDK for .NET	AWS SDK for .NET コード例
AWS SDK for PHP	AWS SDK for PHP コード例
AWS Tools for PowerShell	AWS Tools for PowerShell コード例

SDK ドキュメント	コード例
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) コード例
AWS SDK for Ruby	AWS SDK for Ruby コード例
AWS SDK for Rust	AWS SDK for Rust コード例
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP コード例
AWS SDK for Swift	AWS SDK for Swift コード例

可用性の例

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

AWS HealthLake リリース

次の表は、機能と更新がいつリリースされたかを示しています AWS HealthLake。リリースの詳細については、リンクされたトピックを参照してください。

変更	説明	日付
$export と $davinci-data-export のセキュリティとタグのフィルタリング	$export および $davinci-data-export オペレーションは、エクスポートされたリソースを _security および meta.tag コーディング値でフィルタリングするための meta.security および _tag クエリパラメータをサポートするようになりました。これにより、標準の FHIR system|code 形式を使用したマルチテナントエクスポートときめ細かなアクセスコントロールが可能になります。	2026 年 4 月 30 日
$bulk-member-match オペレーション	AWS HealthLake は、複数のメンバー一致リクエストを非同期的に処理する $bulk-member-match オペレーションをサポートするようになりました。このオペレーションにより、医療組織は、属性とカバレッジ情報を 1 回の一括リクエストで使用して、さまざまな医療システム全体で何百人ものメンバーの一意の識別子を効率的に照合できます。 • データストアごとに最大 5 つの同時オペレーション	2026 年 4 月 1 日

	で、リクエストごとに最大 500 のメンバーを処理します • MatchedMembers、Non MatchedMembers、および ConsentConstrained Membersグループに分類された結果 • と統合してend-to-endのバルクデータワークフロー`$davinci-data-export` を実現 詳細については、「the section called “$bulk-member-match”」を参照してください。	
非同期バンドルトランザクション	AWS HealthLake は非同期`Bundle`タイプ をサポートするようになりました。これにより`transaction` 、最大 500 個のリソースを持つトランザクションを送信できます。HealthLake は処理のためにトランザクションをキューに入れ、すぐにポーリング URL を返し、ステータスをチェックして結果を取得します。詳細については、「非同期バンドルトランザクション」を参照してください。	2026 年 3 月 24 日

バンドルオペレーションでのパッチサポート	HealthLake が Bundle Patch をサポートするようになり、トランザクションバンドルとバッチバンドルの FHIRPath と JSON Patch の両方、および Patch API の FHIRPath が有効になりました。この機能を使用すると、完全なリソース置換、ペイロードサイズの縮小、統合ワークフローの簡素化、データ取り込みの高速化を必要とせずに、バンドルオペレーション内で直接リソースにパッチを適用できます。	2026 年 3 月 20 日
$davinci-data-export の DaVinci PDex エクスポートタイプ davinci-data-export	`$davinci-data-export` オペレーションは、プロバイダーアクセス、Payer-to-Payer、およびメンバーアクセス APIs の PDex エクスポートタイプをサポートするようになりました。 • ExplanationOfBenefit リソースのプロファイルベースの包含ロジック • `_includeEOB2xWoFinancial` パラメータを使用した財務データ変換 • 臨床データとクレームデータの 5 年間の時間フィルター	2026 年 3 月 20 日

$export & $davinci-data-export の _until パラメータ	エクスポートオペレーションの一時フィルタリングパラメータ	2026 年 2 月 26 日
_include 検索パラメータ	HealthLake が include:* と include:iterate をサポートするようになりました	2026 年 2 月 26 日
CMS 相互運用性エンドポイント	この機能を使用すると、CMS カテゴリ別の API 使用状況を追跡し、コンプライアンス上の使用状況メトリクスをレポートできます。	2026 年 2 月 26 日
バンドルメッセージタイプのサポート	メッセージタイプの FHIR Bundle リソースのサポートが制限されました	2026 年 2 月 26 日

新しい IGs のサポートを追加	AWS HealthLake は、CMS 0057F の FHIR 実装ガイド (IGs) サポートを拡張しました。 • CARIN Blue ボタン 2.0.0 および 2.1.0 のサポート • Da Vinci Payer Data Exchange 2.0.0 および 2.1.0 のサポート • DaVinci Payer Data Exchange (PDex) 米国薬物処方 2.0.1 および 2.1.0 のサポート • Da Vinci Clinical Data Exchange (CDex) 2.1.0 のサポート • Da Vinci 事前認可サポート (PAS) FHIR IG 2.1.0 のサポート	2026 年 2 月 26 日
$submit オペレーション	`$submit` オペレーションを使用すると、事前承認リクエストを支払者に送信して承認を受けることができます。	2026 年 2 月 26 日
$questionnaire-package オペレーション	`$questionnaire-package` オペレーションは、FHIR アンケートとそのアンケートのレンダリングと処理に必要なすべての依存関係を含む包括的なバンドルを取得します。	2026 年 2 月 26 日

$inquire オペレーション	$inquire オペレーションを使用すると、以前に送信された事前承認リクエストのステータスを確認できます。	2026 年 2 月 26 日
$member-remove オペレーション	$member-remove オペレーションでは、FHIR メンバー属性リスト (グループリソース) からメンバーを削除できます AWS HealthLake。	2025 年 11 月 12 日
$member-match オペレーション	AWS HealthLake は、患者リソースの $member-match オペレーションをサポートするようになりました。これにより、医療組織は人口統計情報とカバレッジ情報を使用して、さまざまな医療システム全体でメンバーの一意の識別子を見つけることができます。	2025 年 11 月 12 日
$member-add オペレーション	FHIR $member-add オペレーションは、メンバー (患者) をグループリソース、特にメンバー属性リストに追加します。	2025 年 11 月 12 日
$davinci-data-export オペレーション	$davinci-data-export オペレーションは、メンバー属性リストデータのエクスポートを可能にする非同期 FHIR オペレーションです AWS HealthLake。	2025 年 11 月 12 日

$confirm-attribution-list オペレーション	コンシューマーが属性リストに加える変更がなくなったことをプロデューサーに示し、非アクティブなメンバーを削除して属性リストを確定し、ステータスを「最終」に変更します。	2025 年 11 月 12 日
$attribution-status オペレーション	特定のメンバーの属性ステータスを取得し、患者に関連するすべての属性リソースを含むバンドルを返します。	2025 年 11 月 12 日
FHIR サブスクリプション	HealthLake は FHIR サブスクリプションをサポートしているため、特定の医療データの変更が発生したときにリアルタイムの通知を受信し、イベント駆動型のワークフローを構築できます。	2025 年 10 月 30 日
モントリオールカナダへのリージョン拡張	HealthLake は、カナダ (モントリオール) リージョンで利用できます。詳細については、「サービスエンドポイント」を参照してください。	2025 年 10 月 17 日

新しい IGs のサポートを追加	AWS HealthLake は、次のカナダ市場向けの FHIR 実装ガイド (IGs) サポートを拡張しました。 • カナダの医療システム間の相互運用性のためのコアデータ要素と制約を定義する CA Core+Canadian ベースライン FHIR プロファイル。 • CA:eReC Pan-Canadian eReferral-eConsultStandardized FHIR 仕様。 • 患者概要 カナダ版 (PS-CA) ケア設定間で重要な患者の健康情報を共有するための国際患者概要 (IPS) のカナダ適応。 • Ontario Digital Health Drug RepositoryOntario 固有の FHIR プロファイルは、州の医療システム内での標準化された薬剤および処方データ交換に使用されます。	2025 年 10 月 17 日
リージョン固有の IG サポート	HealthLake がリージョン固有の IGs をサポートするようになりました。詳細については、「プロファイルの検証」を参照してください。	2025 年 10 月 8 日

パッチオペレーション	HealthLake では、リソース全体を更新することなく、JSON Patch オペレーションを使用して FHIR リソースの特定の要素を変更できます。	2025 年 8 月 18 日
$validate オペレーション	HealthLake は、ストレージオペレーションを実行せずに仕様とプロファイルに対して FHIR リソースを検証し、詳細な検証結果を返します。	2025 年 8 月 18 日
$purge オペレーション	HealthLake には、データストアから患者の区画内のすべてのリソースを完全に削除する機能が含まれています。	2025 年 8 月 18 日
$lookup オペレーション	HealthLake は、コードとシステム識別子を提供することで、CodeSystem 内の特定の概念に関する詳細情報を取得する機能を提供します。	2025 年 8 月 18 日
$expand オペレーション	HealthLake ではValueSet リソースを拡張して、お客様が取り込んだ ValueSets に含まれるコードの完全なリストを取得できるようになりました。	2025 年 8 月 18 日
$erase オペレーション	HealthLake では、特定のリソースとそのすべての履歴バージョンをデータストアから完全に削除できるようになりました。	2025 年 8 月 18 日

$document オペレーション	HealthLake は、Composition リソースと参照されるすべてのリソースを 1 つのドキュメントバンドルにバンドルすることで、完全な臨床ドキュメントの生成をサポートします。	2025 年 8 月 18 日
:検索修飾子の下	HealthLake では、用語システム内の指定された URI より下位にある URI 値の検索が導入されています。	2025 年 8 月 8 日
条件付き削除	HealthLake は FHIR 条件付き削除をサポートするようになりました。これにより、医療組織は論理 FHIR ID ではなく検索条件に基づいて既存のリソースを削除できます。詳細については、「条件に基づく FHIR リソースの削除」を参照してください。	2025 年 7 月 7 日
新しい IGs のサポートを追加	AWS HealthLake は、以下の FHIR 実装ガイド (IGs) サポートを拡張しました。 • FHIR を使用して USCDI 4.0 標準を実装する方法を指定する US Core 7.0.0 • UK 全体の FHIR 実装ガイダンスを提供する UK Core 2.0.1 実装ガイド	2025 年 7 月 7 日

ダブリンアイルランドへのリージョン拡張	HealthLake は、欧州 (ダブリン) リージョンで利用できます。詳細については、「サービスエンドポイント」を参照してください。	2025 年 6 月 9 日
バンドルタイプのトランザクション	HealthLake は FHIR バンドルタイプの「トランザクション」をサポートするようになりました。これにより、医療組織は複数のリソースを単一のアトミックオペレーションとして送信できます。これにより、データ交換と統合のワークフローがより効率的になります。たとえば、医療プロバイダーは、患者レコード、薬剤リスト、予約を 1 回のトランザクションで更新できるため、複雑さと潜在的なエラーを軽減できます。詳細については、「FHIR リソースのバンドル」を参照してください。	2025 年 4 月 28 日

新しい IGs のサポートを追加	AWS HealthLake AWS HealthLake は、以下の FHIR 実装ガイド (IGs) サポートを拡張しました。 • NCQA HEDIS® 実装ガイド (0.3.1): ヘルスケアの有効性データおよび情報セット (HEDIS) の品質測定とレポートをサポートします。 • 国際患者概要 (IPS) (2.0.0): 重要な健康情報の交換を可能にし、患者のケアの継続性をサポートします。 • Quality Measure (5.0.0): 品質メジャーの定義とデータの表示と交換をサポートします。 • Genomics Reporting (3.0.0): ゲノムデータとレポートの交換を容易にします。	2025 年 4 月 28 日
べき等性キー	HealthLake は、FHIR POST オペレーションのべき等性キーをサポートするようになりました。これにより、リソースの作成中にデータの整合性を確保するための堅牢なメカニズムが提供されます。詳細については、「Idempotency and Concurrency」を参照してください。	2025 年 4 月 18 日

FHIR 履歴の整合性	HealthLake は、新しい`x-amz-fhir-history-consistency-level` ヘッダーを介して履歴対応データストアの強力な整合性をサポートするようになりました。「strong」に設定すると、FHIR 検索結果には、更新ステータスに関係なく、インデックスが作成されたすべてのレコードが含まれます。詳細については、「FHIR 検索整合性レベル」を参照してください。	2025 年 4 月 18 日
タグと「if-match」	HealthLake で eTag がサポートされるようになりました。これにより、クライアントは「If-Match」ヘッダーを使用してべき等な更新を確保できます。これにより、同時更新中の偶発的な上書きを防ぐことで、データの整合性を維持できます。これは、複数のシステムが同じレコードを同時に更新しようとする可能性のある大量の医療環境で特に重要です。詳細については、AWS HealthLake のETag」を参照してください。	2025 年 4 月 18 日

バンドルの条件付き PUTs	HealthLake は FHIR バンドルの条件付き更新をサポートするようになり、医療組織がデータをより柔軟に管理および更新できるようになりました。クライアントは、バンドルトランザクションの一部としてリソースを条件付きで作成、更新、または削除するための条件を指定できるようになりました。これにより、システム間のデータ同期プロセスが簡素化され、複雑なクライアント側のロジックの必要性が軽減されます。詳細については、PUTs」を参照してください。	2025 年 4 月 18 日

FHIR V2 スコープでの SMART	HealthLake は、FHIR リソースの作成、読み取り、更新、削除、検索のための SMART on FHIR V2 スコープをサポートしています。詳細については、HealthLake の FHIR リソーススコープの SMART」を参照してください。 • FHIR V2 スコープの SMART は、01/22/2025 以降に作成されたすべての HealthLake データストアで使用できます。データストアがこの日付より前に作成された場合は、サポートチケットを送信して SMART on FHIR V2 スコープを有効にできます。を使用してケースを作成しますAWS Support Center Console。ケースを作成するには、 にログイン AWS アカウントし、ケースの作成を選択します。	2025 年 1 月 22 日
FHIR US Core Profile、バージョン 6.1.0	HealthLake は、FHIR US Core Profile のバージョン 6.1.0 をサポートしています。詳細については、HealthLake の FHIR プロファイルの検証」を参照してください。	2025 年 1 月 22 日

GET を使用した FHIR $export	HealthLake は、 $exportで FHIR をサポートしています GET。詳細については、「FHIR を使用した HealthLake データのエクスポート$export」を参照してください。	2025 年 1 月 22 日
テスト済みのコード例を含むリファクタリングされたデベロッパーガイド	HealthLake では、ネイティブアクション AWS CLI と AWS SDK アクションのテスト済みコード例を含むリファクタリングされたデベロッパーガイドが導入されています。さらに、サポートされているすべての FHIR API インタラクションでプロシージャが利用可能になりました。詳細については、「コード例」と「FHIR リソースの管理」を参照してください。	2024 年 12 月 18 日

FHIR historyとvreadインタラクション	HealthLake は、特定のリソースの履歴を取得するためのFHIR historyインタラクションと、リソースのバージョン固有の読み取りを実行するためのvreadインタラクションをサポートしています。詳細については、「FHIR リソース履歴の読み取り」を参照してください。 • FHIR リソースhistoryは、10/25/2024 以降に作成されたすべてのHealthLake データストアに対してデフォルトで有効になっています。データストアがこの日付より前に作成された場合は、サポートチケットを送信して FHIR historyインタラクションを有効にできます。を使用してケースを作成しますAWS Support Center Console。ケースを作成するには、 AWS アカウントにログインし、ケースの作成を選択します。	2024 年 10 月 25 日

FHIR Patient/$everything オペレーション	HealthLake は、Patientリソースとそのすべての関連リソースを検索するための FHIR Patient/$everything オペレーションをサポートしています。このオペレーションを使用すると、患者のレコード全体にアクセスしたり、Patientデータを一括でダウンロードしたりできます。詳細については、「で患者データPatient/$everything を取得する」を参照してください。 • FHIR Patient/$everything は、02/27/2024 以降に作成されたすべての HealthLake データストアに対してデフォルトで有効になっています。データストアがこの日付より前に作成された場合は、サポートチケットを送信して Patient/$everything オペレーションを有効にできます。を使用してケースを作成しますAWS Support Center Console。ケースを作成するには、AWS アカウント にログインし、ケースの作成を選択します。	2024 年 2 月 27 日

FHIR VerificationResult リソース	HealthLake は、1 つ以上の要素の検証要件、ソース、ステータス、日付を記述するための FHIR VerificationResult リソースタイプをサポートしています。詳細については、HealthLake の FHIR R4 リソースタイプ」を参照してください。	2023 年 12 月 9 日

FHIR $exportオペレーション	HealthLake は、HealthLake データストアからヘルスデータを一括エクスポートするための FHIR HealthLake$exportオペレーションをサポートしています。詳細については、「FHIR を使用した HealthLake データのエクスポート$export」を参照してください。 • FHIR $exportは、2023 年 6 月 1 日以降に作成されたすべての HealthLake データストアに対してデフォルトで有効になっています。データストアがこの日付より前に作成された場合は、サポートチケットを送信して$exportオペレーションを有効にできます。を使用してケースを作成しますAWS Support Center Console。ケースを作成するには、 にログイン AWS アカウントし、ケースの作成を選択します。 • 06/01/23 より前に作成された HealthLake データストアは、システム全体のエクスポートの$exportジョブリクエストのみをサポートします。 • 06/01/23 より前に作成された HealthLake データストアは、データストアのエンド	2023 年 6 月 1 日

	ポイントに対するGETリクエスト$exportを使用したFHIR のステータスの取得をサポートしていません。	
SMART on FHIR のサポート	HealthLake は、SMART on FHIR 認可のサポートを追加します。詳細については、「SMART on FHIR support for AWS HealthLake」を参照してください。	2023 年 5 月 31 日
FHIR プロファイルの検証	HealthLake は、基本リソースタイプの制約や拡張を使用して特定のリソースタイプ定義を定義するための FHIR プロファイル検証をサポートしています。詳細については、「プロファイルの検証」を参照してください。	2023 年 5 月 31 日
アジアパシフィック (ムンバイ) リージョン	HealthLake は、アジアパシフィック (ムンバイ) リージョンで利用できます。詳細については、「サービスエンドポイント」を参照してください。	2023 年 4 月 4 日

自然言語処理がデフォルトでオフになっている	HealthLake は、2023 年 2 月 20 日の時点で、すべてのデータストアで統合自然言語処理 (NLP) をオフにしました。サポートチケットを送信して、統合された NLP 機能を有効にできます。を使用してケースを作成しますAWS Support Center Console。ケースを作成するには、 にログイン AWS アカウント し、ケースの作成を選択します。統合 NLP の詳細については、「NLP と HealthLake の統合」を参照してください。 •	2023 年 2 月 20 日

Amazon Athena での SQL インデックスとクエリ	HealthLake は、Amazon Athena を使用した SQL を使用した FHIR データのクエリをサポートしています。詳細については、Amazon Athena を使用した HealthLake データのクエリ」を参照してください。 • SQL クエリ機能は、11/14/2022 以降に作成されたすべての HealthLake データストアでデフォルトで有効になっています。データストアがこの日付より前に作成された場合は、サポートチケットを送信して SQL クエリ機能を有効にできます。を使用してケースを作成しますAWS Support Center Console。ケースを作成するには、 にログイン AWS アカウント し、ケースの作成を選択します。 • SQL クエリ機能では、HealthLake にアクセスするための IAM 設定を更新する必要があります。HealthLake データストアを作成し、Athena でデータストアへのアクセスを許可するには、IAM ユーザー、グループ、またはロールに `AWSLakeFormationDataAdmin` 管理ポリシーを追加する必要があります	2022 年 11 月 14 日

	。AWSLakeFormationDataAdmin ポリシーを使用してデータレイク管理者を作成し、Athena のデータストアへのアクセスを許可できます。詳細については、「IAM ユーザーまたはロールを設定する」を参照してください。	
インポートジョブの合計サイズの増加	HealthLake は、StartFHIRImportJob リクエストTotal import job sizeの を 500 GB に更新します。詳細については、Service Quotas を参照してください。	2022 年 10 月 3 日
FHIR Bundleリソース	HealthLake は、複数の FHIR Bundleリソースを同時に処理するための FHIR リソースタイプをサポートしています。詳細については、「FHIR リソースのバンドル」を参照してください。	2022 年 8 月 5 日
FHIR インタラクションのクォータの更新	HealthLake は、FHIR リソース管理インタラクションのクォータを更新します。詳細については、Service Quotas を参照してください。	2022 年 7 月 16 日

FHIR _include検索パラメータ	HealthLake は、searchリクエストで追加のリソースを返すための FHIR _include検索パラメータのサポートを追加します。詳細については、「高度な検索パラメータ」を参照してください。	2022 年 7 月 16 日
AWS HealthLake は一般利用可能です	HealthLake は、サポートされているすべてのリージョンで一般利用可能です。詳細については、「サービスエンドポイント」を参照してください。	2021 年 7 月 15 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。