aws

Guide du développeur

AWS HealthLake

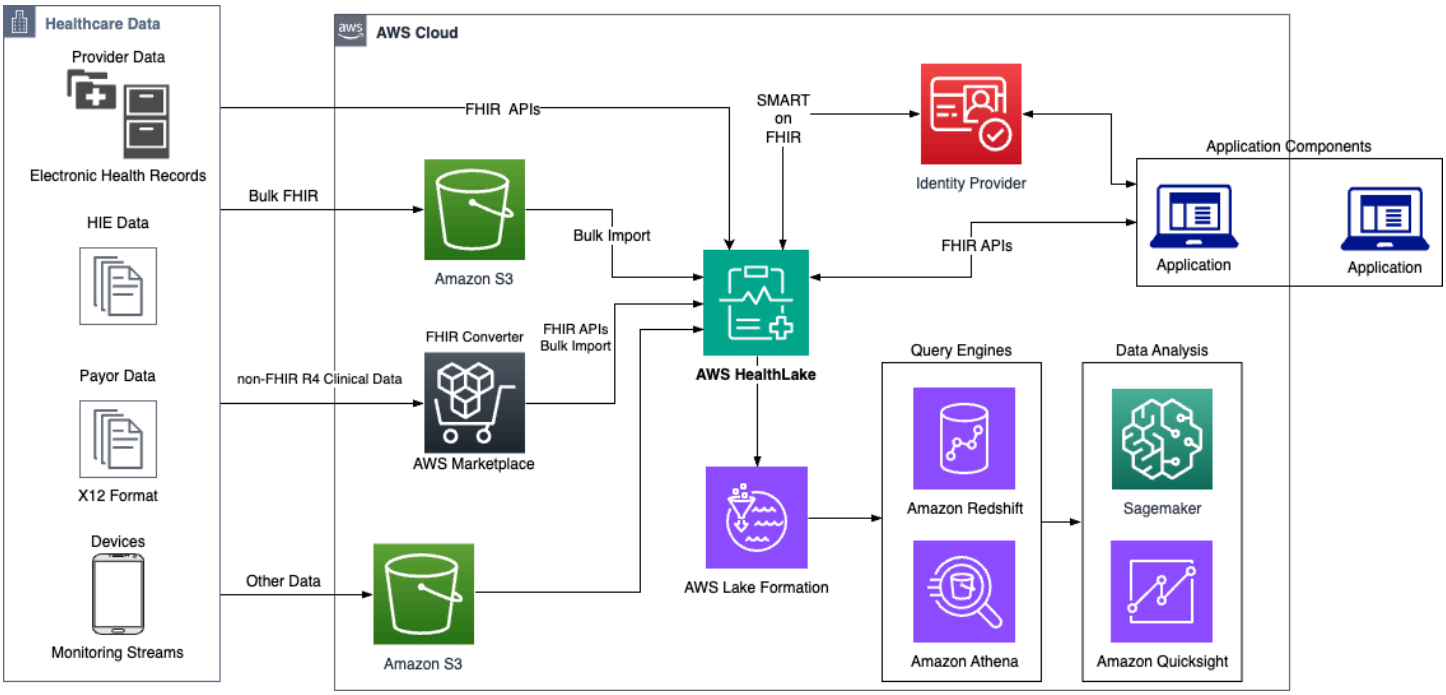
AWS HealthLake: Guide du développeur

Table of Contents

Qu'est-ce que c'est AWS HealthLake ?

AWS HealthLake est un service éligible à la loi HIPAA pour le stockage, l'analyse et le partage de données de santé dans le cloud à l'aide de la spécification Fast Healthcare Interoperability Resources (FHIR) R4. HealthLake les cas d'utilisation incluent :

- Données de santé d'entreprise — Gérez et partagez les données de santé du FHIR R4 directement à partir de celles-ci AWS Cloud tout en préservant des performances et une disponibilité élevées.
- Interopérabilité des soins de santé — Support à la conformité des clients avec la 21st Century Cures Act en matière d'accès des patients via un magasin de données FHIR entièrement géré.
- Traitement du langage naturel (NLP) — Utilisez des modèles NLP intégrés pour extraire des informations médicales pertinentes à partir de données de santé non structurées.
- Analyse multimodale : combinez les HealthLake données avec les AWS HealthImaging données et les AWS HealthOmics données pour fournir des informations utiles à la médecine de précision.



Rubriques

- Avis important
- Caractéristiques de AWS HealthLake
- AWS Services connexes

- Accès AWS HealthLake
- Éligibilité à la loi HIPAA et sécurité des données
- Tarification

Avis important

AWS HealthLake ne remplace pas un avis médical, un diagnostic ou un traitement professionnel et n'est pas destiné à guérir, traiter, atténuer, prévenir ou diagnostiquer une maladie ou un problème de santé. Il vous incombe de procéder à un examen humain dans le cadre de toute utilisation de tout produit tiers destiné à éclairer la prise de décisions cliniques AWS HealthLake, y compris en association avec celui-ci. AWS HealthLake ne doit être utilisé dans le cadre de soins aux patients ou dans des scénarios cliniques qu'après examen par des professionnels de la santé qualifiés faisant preuve de bon jugement médical.

Caractéristiques de AWS HealthLake

AWS HealthLake fournit les fonctionnalités suivantes.

Importer les données de santé du FHIR R4

Grâce à l'action d'importation HealthLake native, vous pouvez facilement migrer vos données FHIR d'un compartiment Amazon S3 vers un magasin de HealthLake données, y compris les notes cliniques, les rapports de laboratoire, les réclamations d'assurance, etc. HealthLake prend en charge la spécification FHIR R4 pour l'échange de données sur les soins de santé. Si nécessaire, vous pouvez travailler avec un AWS HealthLake partenaire pour convertir vos données de santé au format FHIR R4.

Stockez les données de santé de manière sécurisée, conforme et auditable

Un magasin HealthLake de données permet d'indexer les données de santé afin qu'elles puissent être consultées. Le magasin de données crée une vue complète des antécédents médicaux de chaque patient par ordre chronologique et facilite l'échange d'informations en utilisant la spécification FHIR R4. Et il fonctionne en permanence pour maintenir votre index à jour, en vous offrant la possibilité de rechercher des informations à tout moment à l'aide d'interactions FHIR R4 standard, d'un stockage principal durable et d'une mise à l'échelle de l'index.

Tirez parti du serveur FHIR transactionnel

Tirez parti du FHIR APIs pour la validation standard des ressources, de l'autorisation SMART sur FHIR et des fonctionnalités d'exportation de données en masse via l'API FHIR pour unifier et analyser vos données afin de réduire les coûts opérationnels et d'améliorer la prise de décision. HealthLake permet aux clients de se conformer aux dernières normes réglementaires ONC et CMS, notamment : HL7 FHIR R4 APIs, FHIR Bulk Data Access, US Core IG STU, HL7 SMART App Launch Framework IG, 2.0 OAuth et OpenID Connect.

Transformez les données médicales non structurées à l'aide du NLP

Le traitement médical intégré du langage naturel (NLP) transforme toutes les données textuelles médicales brutes dans un magasin de données afin de comprendre et d'extraire HealthLake des informations pertinentes à partir de données de santé non structurées. Grâce à la PNL médicale intégrée, vous pouvez extraire automatiquement les entités, les relations entre entités, les traits des entités et les informations de santé protégées (PHI) de votre texte médical. Les entités extraites du NLP sont stockées en tant que ressources FHIR R4 natives dans un magasin de HealthLake données et sont accessibles via FHIR R4 ou APIs Amazon Athena (SQL).

AWS Services connexes

AWS HealthLake offre une intégration étroite avec les autres AWS services. Il est utile de connaître les services suivants pour en tirer pleinement parti HealthLake.

- Gestion des identités et des accès AWS— Utilisez IAM pour gérer en toute sécurité les identités et l'accès aux HealthLake ressources.
- Amazon Simple Storage Service : utilisez Amazon S3 comme zone intermédiaire pour importer des données DICOM dans HealthLake.
- AWS CloudTrail— CloudTrail À utiliser pour suivre HealthLake l'activité des utilisateurs et l'utilisation de l'API.
- Amazon CloudWatch — CloudWatch À utiliser pour observer et surveiller les HealthLake ressources.
- AWS CloudFormation— CloudFormation À utiliser pour implémenter des modèles d'infrastructure sous forme de code (IaC) dans lesquels créer des ressources HealthLake.
- AWS PrivateLink— Utilisez Amazon VPC pour établir une connectivité entre Amazon Virtual Private Cloud HealthLake et Amazon sans exposer les données à Internet.

- Amazon EventBridge — EventBridge À utiliser pour créer des applications évolutives axées sur les événements en créant des règles qui acheminent les HealthLake événements vers des cibles.
- AWS Lake Formation— Utilisez Lake Formation pour gérer, sécuriser et partager des HealthLake données de manière centralisée à des fins d'analyse et d'apprentissage automatique.
- Amazon Athena : utilisez Athena pour interroger des HealthLake données avec SQL afin de permettre une analyse plus approfondie.

Accès AWS HealthLake

Vous pouvez y accéder AWS HealthLake en utilisant le AWS Management Console, AWS Command Line Interface et le AWS SDKs. Ce guide fournit des instructions procédurales pour le AWS Management Console et des exemples de code pour le AWS CLI et AWS SDKs.

AWS Command Line Interface (AWS CLI)

AWS CLI Il fournit des commandes pour un large éventail de AWS produits et est pris en charge sous Windows, Mac et Linux. Pour plus d'informations, consultez le Guide de l'utilisateur AWS Command Line Interface .

AWS SDKs

AWS SDKs fournir des bibliothèques, des exemples de code et d'autres ressources aux développeurs de logiciels. Ces bibliothèques fournissent des fonctions de base qui automatisent des tâches telles que la signature cryptographique de vos demandes, les nouvelles tentatives et la gestion des réponses aux erreurs. Pour plus d'informations, consultez la section Outils sur lesquels vous pouvez vous appuyer AWS.

AWS Management Console

AWS Management Console fournit une interface utilisateur Web pour la gestion HealthLake et les ressources associées. Si vous avez créé un AWS compte, vous pouvez vous connecter à la HealthLake console.

Éligibilité à la loi HIPAA et sécurité des données

Il s'agit d'un service admissible en vertu de la loi HIPAA. Pour plus d'informations sur AWS le Health Insurance Portability and Accountability Act des États-Unis de 1996 (HIPAA) et sur l'utilisation de AWS services pour traiter, stocker et transmettre des informations de santé protégées (PHI), consultez la section Présentation de la HIPAA.

Les connexions HealthLake contenant des PHI et des informations personnelles identifiables (PII) doivent être cryptées. Par défaut, toutes les connexions doivent HealthLake utiliser le protocole HTTPS sur TLS. HealthLake stocke le contenu crypté des clients et fonctionne selon le modèle de responsabilitéAWS partagée.

Tarification

Pour plus d'informations sur les HealthLake prix, consultez la section AWS HealthLake sur les prix. Pour estimer les coûts, utilisez le calculateur de HealthLake prix.

Commencer avec AWS HealthLake

Pour commencer à l'utiliser AWS HealthLake, configurez un AWS compte et créez un Gestion des identités et des accès AWS utilisateur. Pour utiliser le AWS CLIou le AWS SDKs, vous devez les installer et les configurer.

> **Note**
>
> Le Référence chapitre de ce guide fournit du contenu de support pour SMART sur FHIR, FHIR R4 et. AWS HealthLake Par exemple, vous pouvez trouver des informations sur SMART sur la configuration FHIR, les validations de profil FHIR prises en charge et les points de terminaison. HealthLake

Après avoir appris HealthLake les concepts et la configuration, un court didacticiel avec des exemples de code est disponible pour vous aider à démarrer.

Rubriques

- AWS HealthLake concepts
- Configuration AWS HealthLake
- AWS HealthLake didacticiel

AWS HealthLake concepts

La terminologie et les concepts suivants sont essentiels à votre compréhension et à votre utilisation de AWS HealthLake.

Concepts

- Stratégie d'autorisation du magasin de données
- PNL intégré
- Analyses intégrées

Stratégie d'autorisation du magasin de données

Un magasin de HealthLake données est un référentiel de données de santé FHIR R4 qui se trouvent dans un seul. Région AWS HealthLake prend en charge les stratégies d'autorisation de stockage de données suivantes.

- Autorisation SigV4 : HealthLake autorise les appels d'API FHIR à l'aide de l'autorisation AWS Signature Version 4 (SigV4).
- Autorisation SMART sur autorisation FHIR : HealthLake autorise les appels API FHIR à l'aide d'applications médicales substituables et de technologies réutilisables (SMART) sur autorisation FHIR.

Pour de plus amples informations, veuillez consulter Création d'un magasin HealthLake de données.

PNL intégré

AWS HealthLake s'intègre aux bibliothèques de traitement du langage naturel (NLP) éligibles à la loi HIPAA pour extraire des données de santé significatives à partir de textes médicaux non structurés. Les bibliothèques de PNL identifient les entités médicales telles que les affections, les médicaments, les dosages, les tests, les traitements et les procédures. Ils reconnaissent les relations entre les entités et les relient à des bibliothèques d'ontologie médicale telles que ICD-10-CM et. RxNorm Pour de plus amples informations, veuillez consulter Traitement du langage naturel (NLP) intégré pour HealthLake.

Analyses intégrées

AWS HealthLake va au-delà du FHIR `search` et `bundle` APIs fournit des analyses intégrées pour interroger et analyser de grands volumes de données de santé. Lors de l'importation, génère HealthLake automatiquement des tables pour l'index SQL et la requête. Cela vous permet d'obtenir des informations exploitables à partir de données de santé complexes sans avoir à effectuer de gros travaux d'ingénierie des données. Pour plus d'informations, consultez Interrogation de HealthLake données avec Amazon Athena et AWS HealthLake exemples de projets.

Configuration AWS HealthLake

Dans ce chapitre, vous allez utiliser le AWS Management Console pour configurer les autorisations requises pour commencer à utiliser AWS HealthLake et à créer un magasin de données. Pour

configurer les autorisations nécessaires à la création d'un magasin de données, vous devez créer un utilisateur ou un rôle IAM qui est un administrateur et HealthLake un administrateur de lac de données. Vous faites de cet utilisateur un administrateur de lac de données dans AWS Lake Formation. L'administrateur du lac de données accorde à Lake Formation l'accès aux ressources nécessaires pour utiliser Amazon Athena afin d'interroger un magasin de données. Après avoir créé un magasin de HealthLake données, vous pouvez configurer les autorisations d'importation et d'exportation de fichiers.

Rubriques

Inscrivez-vous pour un Compte AWS

Pour commencer AWS, vous avez besoin d'un Compte AWS. Pour plus d'informations sur la création d'un Compte AWS, voir Getting started with an Compte AWS dans le Guide de Gestion de compte AWS référence.

Configuration d'un utilisateur ou d'un rôle IAM à utiliser HealthLake (administrateur IAM)

> Persona : administrateur IAM
>
> Utilisateur capable de créer des utilisateurs et des rôles IAM et d'ajouter des administrateurs de data lake.

Les étapes décrites dans cette rubrique doivent être effectuées par un administrateur IAM.

Pour connecter votre banque de HealthLake données à Athena, vous devez créer un utilisateur ou un rôle IAM à la fois administrateur de lac de données et administrateur. HealthLake Ce nouvel utilisateur ou rôle accorde l'accès aux ressources trouvées dans un magasin de données via AWS Lake Formation, et la politique `AmazonHealthLakeFullAccess` AWS gérée est ajoutée à son utilisateur ou à son rôle.

> **Important**
>
> Un utilisateur ou un rôle IAM administrateur de lac de données ne peut pas créer de nouveaux administrateurs de lacs de données. Pour ajouter un administrateur de lac de données supplémentaire, vous devez utiliser un utilisateur ou un rôle IAM auquel l'`AdministratorAccess`accès a été accordé.

Pour créer un administrateur

1. Ajoutez la politique AWS gérée par **`AmazonHealthlakeFullAccess`** IAM à un utilisateur ou à un rôle au sein de votre organisation.

   Si vous n'êtes pas habitué à créer un utilisateur IAM, consultez les sections Création d'un utilisateur IAM et Présentation des politiques AWS IAM dans le guide de l'utilisateur IAM.
2. Accordez à l'utilisateur ou au rôle IAM l'accès à AWS Lake Formation.

   - Ajoutez la politique AWS gérée par IAM suivante à un utilisateur ou à un rôle au sein de votre organisation : **`AWSLakeFormationDataAdmin`**

     > **Note**
     >
     > La `AWSLakeFormationDataAdmin` politique donne accès à toutes les ressources AWS du Lake Formation. Nous vous recommandons de toujours utiliser les autorisations minimales requises pour accomplir votre tâche. Pour plus d'informations, consultez Bonnes pratiques IAM dans le Guide de l'utilisateur IAM.
3. Ajoutez la politique intégrée suivante à l'utilisateur ou au rôle. Pour plus d'informations, consultez la section Politiques intégrées dans le guide de l'utilisateur IAM.

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": "*"
        }
    ]
}
```

Pour plus d'informations sur cette `AWSLakeFormationDataAdmin` politique, consultez la section Lake Formation Personas and IAM Permissions Reference dans le AWS Lake Formation Developer Guide.

Ajouter un utilisateur ou un rôle en tant qu'administrateur du lac de données dans Lake Formation (administrateur IAM)

> **Note**
>
> Cette étape est obligatoire si vous procédez à une intégrationIndex et requête SQL.

Ensuite, l'administrateur IAM doit ajouter l'utilisateur ou le rôle créé à l'étape précédente en tant qu'administrateur de lac de données dans Lake Formation.

Pour ajouter un utilisateur ou un rôle IAM en tant qu'administrateur de lac de données

1. Ouvrez la console AWS Lake Formation : https://console.aws.amazon.com/lakeformation/

   > **Note**
   >
   > Si c'est la première fois que vous visitez Lake Formation, une boîte de dialogue Welcome to Lake Formation apparaît, vous demandant de définir un administrateur de Lake Formation.

2. Attribuez au nouvel utilisateur ou au nouveau rôle un administrateur AWS du lac de données de Lake Formation.
   - Option 1 : Si vous avez reçu la boîte de dialogue Welcome to Lake Formation.
     1. Choisissez Ajouter d'autres AWS utilisateurs ou rôles.
     2. Cliquez sur la flèche vers le bas (▼).
     3. Choisissez l' HealthLake administrateur que vous souhaitez également voir administrateur de Lake Formation.
     4. Choisissez Démarrer.
   - Option 2 : utilisez le volet de navigation (☰).
     1. Choisissez le volet de navigation (☰).
     2. Sous Autorisations, sélectionnez Rôles et tâches administratifs.
     3. Dans la section Administrateurs du lac de données, sélectionnez Choisir les administrateurs.

4. Dans la boîte de dialogue Gérer les administrateurs des lacs de données, cliquez sur la flèche vers le bas (▼).
5. Ensuite, sélectionnez ou recherchez les HealthLake administrateurs, utilisateurs ou rôles que vous souhaitez également voir devenir administrateurs de Lake Formation.
6. Choisissez Enregistrer.

3. Modifiez les paramètres de sécurité par défaut à gérer par Lake Formation. Les ressources du magasin de HealthLake données doivent être gérées par Lake Formation et non par IAM. Pour effectuer une mise à jour, voir Modifier le modèle d'autorisation par défaut dans le guide du développeur de AWS Lake Formation.

Création de compartiments S3

Pour importer des données FHIR R4 dans AWS HealthLake, deux compartiments Amazon S3 sont recommandés. Le compartiment d'entrée Amazon S3 contient les données FHIR à importer et HealthLake lit à partir de ce compartiment. Le compartiment de sortie Amazon S3 stocke les résultats du traitement de la tâche d'importation et HealthLake écrit (journaux) dans ce compartiment.

> **Note**
>
> En raison de la politique Gestion des identités et des accès AWS (IAM), les noms de vos compartiments Amazon S3 doivent être uniques. Pour plus d'informations, veuillez consulter la section Règles de dénomination de compartiment dans le Guide de l'utilisateur Amazon Simple Storage Service.

Dans le cadre de ce guide, nous indiquons les compartiments d'entrée et de sortie Amazon S3 suivants lors de la configuration des autorisations d'importation plus loin dans cette section.

- Seau d'entrée : `arn:aws:s3:::amzn-s3-demo-source-bucket`
- Seau de sortie : `arn:aws:s3:::amzn-s3-demo-logging-bucket`

Pour plus d'informations, consultez la section Création d'un compartiment dans le guide de l'utilisateur Amazon S3.

Création d'un magasin de données

Un magasin de HealthLake données est un référentiel de données FHIR R4 résidant dans une seule AWS région. Un AWS compte peut avoir zéro ou plusieurs banques de données. HealthLake prend en charge deux stratégies d'autorisation de stockage de données.

> **Important**
>
> Avant de créer un magasin de HealthLake données, consultez les politiques de contrôle des services (SCP) de votre AWS organisation susceptibles de restreindre la création ou la gestion des HealthLake ressources. Les SCP peuvent empêcher la création réussie de magasins de HealthLake données, même si vos autorisations IAM sont correctement configurées.
>
> A `datastoreID` est généré lorsque vous créez un magasin HealthLake de données. Vous devez utiliser le `datastoreID` lors de la configuration des autorisations d'importation plus loin dans cette section.

Pour créer un magasin HealthLake de données, voirCréation d'un magasin HealthLake de données.

Configuration des autorisations pour les tâches d'importation

Avant d'importer des fichiers dans un magasin de données, vous devez HealthLake autoriser l'accès à vos compartiments d'entrée et de sortie dans Amazon S3. Pour accorder HealthLake l'accès, vous créez un rôle de service IAM pour HealthLake, vous ajoutez une politique de confiance au rôle pour accorder les autorisations d' HealthLake assumer le rôle, et vous attachez une politique d'autorisation au rôle qui lui accorde l'accès à vos compartiments Amazon S3.

Lorsque vous créez une tâche d'importation, vous spécifiez le nom de ressource Amazon (ARN) de ce rôle pour le`DataAccessRoleArn`. Pour plus d'informations sur les rôles IAM et les politiques de confiance, consultez la section Rôles IAM.

Après avoir configuré l'autorisation, vous êtes prêt à importer des fichiers dans votre banque de données à l'aide d'une tâche d'importation. Pour de plus amples informations, veuillez consulter Démarrage d'une tâche d'importation FHIR.

Pour configurer les autorisations d'importation

1. Si ce n'est pas déjà fait, créez un compartiment Amazon S3 de destination pour les fichiers journaux de sortie. Le compartiment Amazon S3 doit se trouver dans la même AWS région que le service, et le blocage de l'accès public doit être activé pour toutes les options. Pour en savoir plus, consultez Utiliser Amazon S3 pour bloquer l'accès public. Une clé KMS Amazon-owned ou une clé KMS appartenant au client doit également être utilisée pour le chiffrement. Pour en savoir plus sur l'utilisation des clés KMS, consultez Amazon Key Management Service.
2. Créez un rôle de service d'accès aux données pour HealthLake et autorisez le HealthLake service à l'assumer conformément à la politique de confiance suivante. HealthLake l'utilise pour écrire le bucket Amazon S3 de sortie.

   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. Ajoutez une politique d'autorisation au rôle d'accès aux données qui lui permet d'accéder au compartiment Amazon S3. `amzn-s3-demo-bucket`Remplacez-le par le nom de votre compartiment.

   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"
    }]
}
```

Configuration des autorisations pour les tâches d'exportation

Avant d'exporter des fichiers depuis un magasin de données, vous devez HealthLake autoriser l'accès à votre compartiment de sortie dans Amazon S3. Pour accorder HealthLake l'accès, vous créez un rôle de service IAM pourHealthLake, vous ajoutez une politique de confiance au rôle pour accorder les autorisations d' HealthLake assumer le rôle, et vous attachez une politique d'autorisation au rôle qui lui accorde l'accès à votre compartiment Amazon S3.

Si vous avez déjà créé un rôle pour HealthLake, vous pouvez le réutiliser et lui accorder les autorisations supplémentaires pour votre compartiment Amazon S3 d'exportation répertoriées dans cette rubrique. Pour en savoir plus sur les rôles IAM et les politiques de confiance, consultez Politiques et autorisations IAM.

Important

HealthLake prend en charge à la fois les demandes d'exportation du SDK natif et l'opération FHIR `$export` R4. Des actions IAM distinctes doivent être fournies en fonction de l'API d'exportation que vous décidez d'utiliser. Cela vous permet de gérer `allow` et `deny` d'autoriser séparément. Si vous souhaitez restreindre les exportations du HealthLake SDK et de l'API REST FHIR, vous devez appliquer des autorisations de refus aux différentes actions IAM. Les modifications des autorisations des utilisateurs IAM ne sont pas nécessaires si vous accordez aux utilisateurs un accès complet à HealthLake.

Utilisation AWS CLI and AWS SDK :

Les HealthLake actions natives suivantes sont disponibles pour exporter des données depuis un magasin de données à l'aide AWS CLI des AWS SDK et :

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

À l'aide des API FHIR :

Les actions IAM suivantes sont disponibles pour exporter des données depuis un magasin de HealthLake données et pour annuler (supprimer) une tâche d'exportation à l'aide de l'opération `$export` FHIR :
`POST`:

- StartFHIRExportJobWithPost

GET:

- StartFHIRExportJobWithGet
- DescribeFHIRExportJobWithGet
- GetExportedFile

DELETE:

- CancelFHIRExportJobWithDelete

L'utilisateur ou le rôle qui définit les autorisations doit être autorisé à créer des rôles, à créer des politiques et à associer des politiques aux rôles. La politique IAM suivante accorde ces autorisations.

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"
                }
            }
```

```
        }
    ]
}
```

Pour configurer les autorisations d'exportation

1. Si ce n'est pas déjà fait, créez un compartiment Amazon S3 de destination pour les données que vous allez exporter depuis votre magasin de données. Le compartiment Amazon S3 doit se trouver dans la même région AWS que le service, et le blocage de l'accès public doit être activé pour toutes les options. Pour en savoir plus, consultez Utiliser Amazon S3 pour bloquer l'accès public. Une clé KMS Amazon-owned ou une clé KMS appartenant au client doit également être utilisée pour le chiffrement. Pour en savoir plus sur l'utilisation des clés KMS, consultez Amazon Key Management Service.
2. Si ce n'est pas déjà fait, créez un rôle de service d'accès aux données HealthLake et autorisez le HealthLake service à l'assumer conformément à la politique de confiance suivante. HealthLakel'utilise pour écrire le bucket Amazon S3 de sortie. Si vous en avez déjà créé unConfiguration des autorisations pour les tâches d'importation, vous pouvez le réutiliser et lui accorder des autorisations pour votre compartiment Amazon S3 à l'étape suivante.

   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. Ajoutez une politique d'autorisation au rôle d'accès aux données qui lui permet d'accéder à votre compartiment Amazon S3 de sortie. `amzn-s3-demo-bucket`Remplacez-le par le nom de votre compartiment.

   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"
    }]
}
```

Installer la AWS CLI

Le AWS CLI est requis pour décrire et répertorier les propriétés des tâches d' HealthLake importation et d'exportation. Vous pouvez également demander ces informations à l'aide HealthLake des SDK.

Pour configurer le AWS CLI

1. Téléchargez et configurez l'interface AWS CLI. Pour obtenir des instructions, consultez les rubriques suivantes dans le Guide de l'utilisateur de l'AWS Command Line Interface :

   - Installation ou mise à jour de la dernière version du AWS CLI
   - Commencer à utiliser le AWS CLI

2. Dans le AWS CLI `config` fichier, ajoutez un profil nommé pour l'administrateur. Vous utilisez ce profil lors de l'exécution des AWS CLI commandes. Conformément au principe de sécurité du moindre privilège, nous vous recommandons de créer un rôle IAM distinct doté de privilèges spécifiques aux tâches effectuées. Pour plus d'informations sur les profils nommés, consultez la section Configuration et paramètres des fichiers d'identification dans le Guide de l'AWS Command Line Interface utilisateur.

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

3. Vérifiez la configuration à l'aide de la `help` commande suivante.

   ```
   aws healthlake help
   ```

   Si la configuration AWS CLI est correcte, une brève description AWS HealthLake et une liste des commandes disponibles s'affichent.

AWS HealthLake didacticiel

Objectif

Dans ce didacticiel, vous allez importer des données FHIR R4 dans un HealthLake magasin de données à l'aide d'actions natives HealthLake. Ensuite, vous allez gérer (créer, lire, mettre à jour, supprimer) une ressource FHIR à l'aide de FHIR. RESTful APIs Pour terminer le didacticiel, vous allez exporter les données FHIR à l'aide d' HealthLakeactions natives.

Conditions préalables

Toutes les procédures répertoriées dans le présent document Configuration sont nécessaires pour terminer ce didacticiel.

Étapes du didacticiel

1. Démarrer la tâche d'importation FHIR
2. Obtenir les propriétés des tâches d'importation FHIR
3. Créer une ressource FHIR
4. Lire la ressource FHIR
5. Mettre à jour la ressource FHIR
6. Supprimer la ressource FHIR
7. Exporter les données FHIR
8. Supprimer le magasin de données

Gestion des banques de données avec AWS HealthLake

Avec AWS HealthLake, vous créez et gérez des magasins de données pour les ressources FHIR R4. Lorsque vous créez un magasin de HealthLake données, un référentiel de données FHIR est mis à disposition via un point de terminaison d'API RESTful. Vous pouvez choisir d'importer (précharger) les données de santé open source FHIR R4 de Synthea dans votre banque de données lorsque vous les créez. Pour de plus amples informations, veuillez consulter Types de données préchargés.

> **Important**
>
> HealthLake prend en charge deux types de stratégies d'autorisation de stockage de données FHIR, AWS SIGv4 ou SMART sur FHIR. Vous devez choisir l'une des stratégies d'autorisation avant de créer un magasin de données HealthLake FHIR. Pour de plus amples informations, veuillez consulter Stratégie d'autorisation du magasin de données.

Pour connaître les FHIR-related capacités (comportements) d'un magasin de HealthLake données actif, récupérez sa déclaration de capacité.

Les rubriques suivantes décrivent comment utiliser les actions natives du HealthLake cloud pour créer, décrire, répertorier, mettre à jour, étiqueter et supprimer des banques de données FHIR à l' AWS CLI aide AWS des SDK et. AWS Management Console

Rubriques

- Création d'un magasin HealthLake de données
- Obtenir HealthLake les propriétés du magasin de données
- Lister HealthLake les magasins de données
- Mettre à jour un magasin HealthLake de données
- Marquage des magasins de HealthLake données
- Supprimer un magasin HealthLake de données

Création d'un magasin HealthLake de données

`CreateFHIRDatastore`À utiliser pour créer un magasin de AWS HealthLake données conforme à la spécification FHIR R4. HealthLake les magasins de données sont utilisés pour importer, gérer, rechercher et exporter des données FHIR. Vous pouvez choisir d'importer (précharger) les données

de santé open source FHIR R4 de Synthea dans votre banque de données lorsque vous les créez. Pour de plus amples informations, veuillez consulter Types de données préchargés.

> **Important**
>
> HealthLake prend en charge deux types de stratégies d'autorisation de stockage de données FHIR, AWS SIGv4 ou SMART sur FHIR. Vous devez choisir l'une des stratégies d'autorisation avant de créer un magasin de données HealthLake FHIR. Pour de plus amples informations, veuillez consulter Stratégie d'autorisation du magasin de données.

Lorsque vous créez un magasin de HealthLake données, un référentiel de données FHIR est mis à disposition via un point de terminaison d'API RESTful. Après avoir créé votre banque de HealthLake données, vous pouvez demander sa déclaration de capacité pour trouver toutes les FHIR-related fonctionnalités (comportements) associées.

Après avoir créé un magasin de données, vous pouvez modifier son nom, les profils de validation FHIR par défaut, la configuration NLP, la configuration d'analyse et la configuration du fournisseur d'identité. La configuration de chiffrement ne peut pas être modifiée. Pour de plus amples informations, veuillez consulter Mettre à jour un magasin de données.

Les menus suivants fournissent des exemples pour AWS les SDK AWS CLI et une procédure pour. AWS Management Console Pour plus d'informations, consultez `CreateFHIRDatastore` dans la Référence d'API AWS HealthLake .

Pour créer un magasin HealthLake de données

Choisissez un menu en fonction de vos préférences d'accès à AWS HealthLake.

AWS CLI et SDK

CLI

AWS CLI

Exemple 1 : création d'un magasin SigV4-enabled HealthLake de données

L'`create-fhir-datastore`exemple suivant montre comment créer un nouveau magasin de données dans AWS HealthLake.

```
aws healthlake create-fhir-datastore \
```

```
    --datastore-type-version R4 \
    --datastore-name "FhirTestDatastore"
```

Sortie :

```
{
    "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)"
}
```

Exemple 2 : créer un magasin de FHIR-enabled HealthLake données SMART on

L'`create-fhir-datastore`exemple suivant montre comment créer un nouveau magasin de FHIR-enabled données SMART on dans AWS HealthLake.

```
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
```

Contenu de `identity_provider_configuration.json` :

```
{
    "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\"]}"
}
```

Sortie :

```
{
    "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)"
}
```

- Pour plus de détails sur l'API, consultez CreateFHIRDatastore dans la Référence des commandes de l'AWS CLI .

Python

Kit 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
```

Le code suivant montre un exemple de paramètres pour un magasin de FHIR-enabled HealthLake données SMART on.

```
            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
            )
```

- Pour plus de détails sur l'API, consultez CreateFHIRDatastore dans la Référence des API du kit AWS SDK pour Python (Boto3).

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    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.
```

- Pour plus de détails sur l'API, voir CreateFhirDataStore dans le AWS SDK pour la référence de l'API SAP ABAP.

> **Exemple de disponibilité**
>
> Vous n’avez pas trouvé ce dont vous avez besoin ? Demandez un exemple de code à l'aide du lien Fournir des commentaires dans la barre latérale droite de cette page.

AWS Console

> **Remarque**
>
> La procédure suivante crée un magasin de HealthLake données avec l'autorisation AWS SigV4. La HealthLake console ne prend pas en charge la création d'un magasin de données SMART sur FHIR.

Pour créer un magasin HealthLake de données avec AWS Autorisation SigV4

1. Connectez-vous à la page Créer un magasin de données sur la HealthLake console.
2. Choisissez Create Data Store.
3. Dans la section des paramètres du magasin de données, pour le nom du magasin de données, spécifiez un nom.
4. (Facultatif) Dans la section des paramètres du magasin de données, pour Précharger les échantillons de données, cochez la case pour précharger les données Synthea. Synthea Data est un exemple de jeu de données open source. Pour de plus amples informations, veuillez consulter Types de données préchargés Synthea pour HealthLake.

5. Dans la section Chiffrement du magasin de données, choisissez Utiliser une clé appartenant à AWS (par défaut) ou Choisir une autre clé AWS KMS (avancée).
6. Dans la section Balises - optionnelle, vous pouvez ajouter des balises à votre banque de données. Pour en savoir plus sur le balisage de votre banque de données, consultezMarquage des magasins de HealthLake données.
7. Choisissez Create Data Store.

   L'état de votre magasin de données est disponible sur la page des magasins de données.

Obtenir HealthLake les propriétés du magasin de données

Permet `DescribeFHIRDatastore` d'obtenir les propriétés d'un magasin AWS HealthLake de données. Les menus suivants fournissent une procédure AWS Management Console et des exemples de code pour le AWS CLI et AWS SDKs. Pour plus d'informations, consultez `DescribeFHIRDatastore` dans la Référence d'API AWS HealthLake .

Pour obtenir les propriétés d'un magasin HealthLake de données

Choisissez un menu en fonction de vos préférences d'accès à AWS HealthLake.

AWS CLI et SDKs

CLI

AWS CLI

Pour décrire un magasin de données FHIR

L'`describe-fhir-datastore`exemple suivant montre comment rechercher les propriétés d'un magasin de données dans AWS HealthLake.

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

Sortie :

```
{
    "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
        }
    }
}
```

- Pour plus de détails sur l'API, voir Description FHIRDatastore dans le manuel de référence des AWS CLI commandes.

Python

Kit 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
```

- Pour plus de détails sur l'API, voir AWS Describe FHIRDatastore in SDK for Python (Boto3) API Reference.

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    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.
```

- Pour plus de détails sur l'API, consultez la section Description FHIRDatastore dans le AWS SDK pour la référence de l'API SAP ABAP.

> **Exemple de disponibilité**
>
> Vous n’avez pas trouvé ce dont vous avez besoin ? Demandez un exemple de code à l'aide du lien Fournir des commentaires dans la barre latérale droite de cette page.

AWS Console

1. Connectez-vous à la page Data stores de la HealthLake console.
2. Choisissez un magasin de données.

La page de détails du magasin de données s'ouvre et toutes les propriétés du magasin de HealthLake données sont disponibles.

Lister HealthLake les magasins de données

Permet `ListFHIRDatastores` de répertorier tous les magasins de HealthLake données du compte d'un utilisateur, quel que soit le statut du magasin de données. Les menus suivants fournissent une procédure AWS Management Console et des exemples de code pour le AWS CLI et AWS SDKs. Pour plus d'informations, consultez `ListFHIRDatastores` dans la Référence d'API AWS HealthLake .

Pour répertorier tous les magasins HealthLake de données

Choisissez un menu en fonction de vos préférences d'accès à AWS HealthLake.

AWS CLI et SDKs

CLI

AWS CLI

Pour répertorier des magasins de données FHIR

L'`list-fhir-datastores`exemple suivant montre comment utiliser la commande et comment les utilisateurs peuvent filtrer les résultats en fonction de l'état du magasin de données dans AWS HealthLake.

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

Sortie :

```
{
    "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
        }
    }
    ]
}
```

- Pour plus de détails sur l'API, consultez la section Liste FHIRDatastores dans AWS CLI la référence des commandes.

Python

Kit 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
```

- Pour plus de détails sur l'API, voir Liste FHIRDatastores dans le AWS manuel de référence de l'API SDK for Python (Boto3).

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    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.
```

- Pour plus de détails sur les API, consultez la section Liste FHIRDatastores dans le AWS SDK pour la référence des API SAP ABAP.

> **Exemple de disponibilité**
>
> Vous n'avez pas trouvé ce dont vous avez besoin ? Demandez un exemple de code à l'aide du lien Fournir des commentaires dans la barre latérale droite de cette page.

AWS Console

- Connectez-vous à la page Data stores de la HealthLake console.

Tous les magasins de HealthLake données sont répertoriés dans la section Magasins de données.

Mettre à jour un magasin HealthLake de données

`UpdateFHIRDatastore`À utiliser pour mettre à jour la configuration d'un magasin de AWS HealthLake données existant. Vous pouvez mettre à jour le nom du magasin de données, la configuration du traitement du langage naturel (NLP), la configuration des analyses, les profils de validation FHIR par défaut et la configuration du fournisseur d'identité. La configuration de chiffrement que vous avez choisie lors de la création du magasin de données ne peut pas être modifiée.

> **Note**
>
> La mise à jour de la configuration du fournisseur d'identité la remplace dans son intégralité : incluez tous les champs que vous souhaitez conserver. Tout champ que vous omettez est effacé.

Le menu suivant fournit des exemples pour les AWS SDK AWS CLI et. Pour plus d'informations, consultez `UpdateFHIRDatastore` dans la Référence d'API AWS HealthLake .

> **Important**
>
> Les modifications du NLP et des analyses sont appliquées via un flux de travail asynchrone : le statut du magasin de données change `UPDATING` et revient à une fois la mise à `ACTIVE` jour terminée, ou indique `UPDATE_FAILED` si ce n'est pas le cas. Les modifications du nom du magasin de données, du profil de validation FHIR et du fournisseur d'identité prennent effet immédiatement et ne modifient pas le statut. Une seule mise à jour peut être en cours pour un magasin de données à la fois ; une seconde mise à jour soumise pendant qu'une banque de données est en cours d'exécution revient`ConflictException`. `DescribeFHIRDatastore`À utiliser pour suivre l'état d'une mise à jour.

Pour mettre à jour un magasin HealthLake de données

Choisissez un menu en fonction de vos préférences d'accès à AWS HealthLake.

AWS CLI et SDK

AWS CLI

Exemple 1 : renommer un magasin de données

```
aws healthlake update-fhir-datastore \
  --datastore-id "datastore-id" \
  --datastore-name "RenamedFhirDatastore"
```

Exemple 2 : activer le NLP

```
aws healthlake update-fhir-datastore \
  --datastore-id "datastore-id" \
  --nlp-configuration '{ "Status": "ENABLED" }'
```

Exemple 3 : suspendre les analyses

```
aws healthlake update-fhir-datastore \
  --datastore-id "datastore-id" \
  --analytics-configuration '{ "Status": "PAUSED" }'
```

Exemple 4 : Mettre à jour les profils de validation FHIR par défaut

```
aws healthlake update-fhir-datastore \
  --datastore-id "datastore-id" \
  --profile-configuration '{ "DefaultProfiles": ["us-core-3.1.1", "carin-
bb-2.0.0"] }'
```

La réponse renvoie la totalité, `DatastoreProperties` c'est-à-dire la même forme renvoyée par`DescribeFHIRDatastore`.

```
{
    "DatastoreProperties": {
        "DatastoreId": "datastore-id",
        "DatastoreArn": "arn:aws:healthlake:us-east-1:account-
id:datastore/datastore-id",
        "DatastoreName": "RenamedFhirDatastore",
        "DatastoreStatus": "UPDATING",
```

```
        "DatastoreTypeVersion": "R4",
        "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/
datastore/datastore-id/r4/",
        "NlpConfiguration": { "Status": "ENABLING" },
        "AnalyticsConfiguration": { "Status": "PAUSING" },
        "ProfileConfiguration": { "DefaultProfiles": [ "us-core-3.1.1", "carin-
bb-2.0.0" ] },
        "IdentityProviderConfiguration": {
            "AuthorizationStrategy": "SMART_ON_FHIR_V1",
            "FineGrainedAuthorizationEnabled": true
        }
    }
}
```

Pour plus de détails sur l'API, consultez update-fhir-datastore dans le manuel de référence des commandes de la CLI.AWS

Python

SDK pour Python (Boto3)

```
def update_fhir_datastore(
    self,
    datastore_id: str,
    **kwargs,
) -> dict[str, any]:
    """
    Updates the configuration of an existing HealthLake data store.

    Pass any of DatastoreName, NlpConfiguration, AnalyticsConfiguration,
    ProfileConfiguration, or IdentityProviderConfiguration as keyword
    arguments. Omitted fields are left unchanged.

    :param datastore_id: The ID of the data store to update.
    :return: The response, including the full DatastoreProperties.
    """
    try:
        return self.health_lake_client.update_fhir_datastore(
            DatastoreId=datastore_id, **kwargs
        )
    except ClientError as err:
        logger.exception(
            "Couldn't update data store %s. Here's why: %s",
            datastore_id,
```

```
            err.response["Error"]["Message"],
        )
        raise
```

Pour plus de détails sur l'API, consultez UpdateFhirDataStore dans le manuel de référence de l'API AWS SDK pour Python (Boto3).

> **Exemple de disponibilité**
>
> Vous n'avez pas trouvé ce dont vous avez besoin ? Demandez un exemple de code à l'aide du lien Fournir des commentaires dans la barre latérale droite de cette page.

Marquage des magasins de HealthLake données

Vous pouvez attribuer des métadonnées aux HealthLake banques de données sous forme de balises. Chaque balise est une étiquette composée d'une clé définie par l'utilisateur et d'une valeur. Les balises vous aident à gérer, identifier, organiser, rechercher et filtrer les banques de données.

> **Important**
>
> Ne stockez pas d'informations de santé protégées (PHI), d'informations personnelles identifiables (PII) ou d'autres informations confidentielles ou sensibles dans des balises. Les étiquettes ne sont pas destinées à être utilisées pour des données privées ou sensibles.

Les rubriques suivantes décrivent comment utiliser les opérations de HealthLake balisage à l'aide des AWS Management Console touches AWS CLI, et AWS SDKs. Pour plus d'informations, consultez la section Marquage de vos AWS ressources dans le Références générales AWS guide.

Rubriques

- Marquage d'un magasin de HealthLake données
- Répertorier les balises d'un magasin HealthLake de données
- Débalisage d'un magasin de données HealthLake

Marquage d'un magasin de HealthLake données

`TagResource`À utiliser pour étiqueter un magasin de HealthLake données. Les menus suivants fournissent une procédure AWS Management Console et des exemples de code pour le AWS CLI et AWS SDKs. Pour plus d'informations, consultez `TagResource` dans la Référence d'API AWS HealthLake .

Pour étiqueter un magasin HealthLake de données

Choisissez un menu en fonction de vos préférences d'accès à AWS HealthLake.

AWS CLI et SDKs

CLI

AWS CLI

Pour ajouter une balise au magasin de données

L'exemple `tag-resource` suivant montre comment ajouter une balise au magasin de données.

```
aws healthlake tag-resource \
    --resource-arn "arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/0725c83f4307f263e16fd56b6d8ebdbe" \
    --tags '[{"Key": "key1", "Value": "value1"}]'
```

Cette commande ne produit aucune sortie.

.

- Pour plus de détails sur l'API, reportez-vous TagResourceà la section Référence des AWS CLI commandes.

Python

Kit 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
```

- Pour plus de détails sur l'API, consultez TagResourcele AWS manuel de référence de l'API SDK for Python (Boto3).

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    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.
```

- Pour plus de détails sur l'API, reportez-vous TagResourceà la section de référence du AWS SDK pour l'API SAP ABAP.

> **Exemple de disponibilité**
>
> Vous n'avez pas trouvé ce dont vous avez besoin ? Demandez un exemple de code à l'aide du lien Fournir des commentaires dans la barre latérale droite de cette page.

AWS Console

1. Connectez-vous à la page Data stores de la HealthLake console.
2. Choisissez un magasin de données.

   La page de détails du magasin de données s'ouvre.
3. Dans la section Balises, choisissez Gérer les balises.

   La page Gérer les balises s'ouvre.
4. Sélectionnez Ajouter une nouvelle balise.
5. Entrez une clé et une valeur (facultatif).
6. Choisissez Enregistrer.

Répertorier les balises d'un magasin HealthLake de données

`ListTagsForResource`À utiliser pour répertorier les balises d'un magasin de HealthLake données. Les menus suivants fournissent une procédure AWS Management Console et des exemples de code pour le AWS CLI et AWS SDKs. Pour plus d'informations, consultez `ListTagsForResource` dans la Référence d'API AWS HealthLake .

Pour répertorier les balises d'un magasin HealthLake de données

Choisissez un menu en fonction de vos préférences d'accès à AWS HealthLake.

AWS CLI et SDKs

CLI

AWS CLI

Pour répertorier les balises d'un magasin de données

L'exemple `list-tags-for-resource` suivant répertorie les balises associées au magasin de données spécifié.

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

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous ListTagsForResourceà la section Référence des AWS CLI commandes.

Python

Kit 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
```

- Pour plus de détails sur l'API, consultez ListTagsForResourcele AWS manuel de référence de l'API SDK for Python (Boto3).

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    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.
```

- Pour plus de détails sur l'API, reportez-vous ListTagsForResourceà la section de référence du AWS SDK pour l'API SAP ABAP.

> **Exemple de disponibilité**
>
> Vous n’avez pas trouvé ce dont vous avez besoin ? Demandez un exemple de code à l'aide du lien Fournir des commentaires dans la barre latérale droite de cette page.

AWS Console

1. Connectez-vous à la page Data stores de la HealthLake console.
2. Choisissez un magasin de données.

   La page de détails du magasin de données s'ouvre. Dans la section Balises, toutes les balises du magasin de données sont répertoriées.

Débalisage d'un magasin de données HealthLake

`UntagResource`À utiliser pour supprimer une étiquette d'un magasin de HealthLake données. Les menus suivants fournissent une procédure AWS Management Console et des exemples de code pour le AWS CLI et AWS SDKs. Pour plus d’informations, consultez `UntagResource` dans la Référence d’API AWS HealthLake .

Pour supprimer le balisage d'un magasin HealthLake de données

Choisissez un menu en fonction de vos préférences d'accès à AWS HealthLake.

AWS CLI et SDKs

CLI

AWS CLI

Pour supprimer des balises d'un magasin de données.

L'exemple `untag-resource` suivant montre comment supprimer des balises d'un magasin de données.

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

Cette commande ne produit aucune sortie.

- Pour plus de détails sur l'API, reportez-vous UntagResourceà la section Référence des AWS CLI commandes.

Python

Kit 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
```

- Pour plus de détails sur l'API, consultez UntagResourcele AWS manuel de référence de l'API SDK for Python (Boto3).

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    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.
```

- Pour plus de détails sur l'API, reportez-vous UntagResourceà la section de référence du AWS SDK pour l'API SAP ABAP.

> **Exemple de disponibilité**
>
> Vous n’avez pas trouvé ce dont vous avez besoin ? Demandez un exemple de code à l'aide du lien Fournir des commentaires dans la barre latérale droite de cette page.

AWS Console

1. Connectez-vous à la page Data stores de la HealthLake console.
2. Choisissez un magasin de données.

   La page de détails du magasin de données s'ouvre.
3. Dans la section Balises, choisissez Gérer les balises.

   La page Gérer les balises s'ouvre.
4. Choisissez Supprimer à côté du tag que vous souhaitez supprimer.
5. Choisissez Enregistrer.

Supprimer un magasin HealthLake de données

`DeleteFHIRDatastore`À utiliser pour supprimer un magasin HealthLake de données. Les menus suivants fournissent une procédure AWS Management Console et des exemples de code pour le AWS CLI et AWS SDKs. Pour plus d'informations, consultez `DeleteFHIRDatastore` dans la Référence d'API AWS HealthLake .

Pour supprimer un magasin HealthLake de données

Choisissez un menu en fonction de vos préférences d'accès à AWS HealthLake.

AWS CLI et SDKs

CLI

AWS CLI

Pour supprimer un magasin de données FHIR

L'`delete-fhir-datastore`exemple suivant montre comment supprimer un magasin de données et tout son contenu dans AWS HealthLake.

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

Sortie :

```
{
    "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)"
}
```

- Pour plus de détails sur l'API, voir Supprimer FHIRDatastore dans le manuel de référence des AWS CLI commandes.

Python

Kit 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
```

- Pour plus de détails sur l'API, consultez le manuel de référence de l'API Delete FHIRDatastore in AWS SDK for Python (Boto3).

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    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.
```

- Pour plus de détails sur l'API, voir Supprimer FHIRDatastore dans le AWS SDK pour la référence de l'API SAP ABAP.

Exemple de disponibilité

Vous n’avez pas trouvé ce dont vous avez besoin ? Demandez un exemple de code à l'aide du lien Fournir des commentaires dans la barre latérale droite de cette page.

AWS Console

1. Connectez-vous à la page Data stores de la HealthLake console.
2. Choisissez un magasin de données.

   La page de détails du magasin de données s'ouvre.
3. Sélectionnez Delete (Supprimer).

   La page Supprimer le magasin de données s'ouvre.
4. Pour confirmer la suppression du magasin de données, entrez le nom du magasin de données dans le champ de saisie de texte.
5. Sélectionnez Supprimer.

Gestion des abonnements FHIR dans AWS HealthLake

AWS HealthLake prend en charge les abonnements FHIR, vous permettant de recevoir des notifications en temps réel lorsque des modifications spécifiques des données de santé se produisent. Cette fonctionnalité met en œuvre le modèle d'abonnement thématique FHIR R5 Backport, offrant une évolutivité et une flexibilité améliorées par rapport au modèle d'abonnement FHIR R4 traditionnel.

Avec les abonnements FHIR, vous pouvez créer des applications de santé axées sur les événements qui répondent immédiatement aux modifications des données cliniques, permettant des interventions rapides, des flux de travail automatisés et une meilleure coordination des soins.

Rubriques

Comment fonctionnent les abonnements FHIR

Les abonnements FHIR HealthLake fonctionnent selon un modèle thématique dans lequel :

1. Création de sujets pour définir des événements : créez des sujets d'abonnement qui spécifient les événements susceptibles de déclencher des notifications
2. Vous vous abonnez : créez des abonnements à ces sujets avec des critères de filtrage spécifiques
3. HealthLake moniteurs : Le service surveille en permanence les événements correspondant à vos critères
4. Notifications envoyées : des événements CWhen correspondants se produisent, HealthLake envoie des notifications via le canal de votre choix

Composants clés

Les abonnements FHIR comprennent les éléments suivants.

Sujets d'abonnement

Les sujets d'abonnement constituent la base du système de notification et définissent :

- Événements déclencheurs : quels changements déclenchent des notifications (par exemple : création de ressources, mises à jour, suppressions)
- Filtres disponibles : Quelles sont les options de filtrage disponibles pour les abonnés
- Contenu des notifications : quelles données sont incluses dans les notifications

Le tableau suivant répertorie les types de sujets courants.

Type d’événement	Description	Cas d’utilisation courants
Création de ressources	Déclenché lors de la création de ressources	Enregistrement d'un nouveau patient, enregistrement d'une nouvelle observation
Mises à jour de ressource	Déclenché lorsque les ressources sont modifiées	Changements de statut, mises à jour cliniques
Suppression de ressources	Déclenché lorsque des ressources sont supprimées	Audit et suivi de la conformité

Abonnements

Un abonnement est votre demande de recevoir des notifications pour des événements spécifiques définis par un sujet d'abonnement. Chaque abonnement inclut :

- Référence de rubrique : Spécifie la rubrique d'abonnement à laquelle vous vous abonnez
- Filtres : critères permettant de sélectionner les événements qui génèrent des notifications
- Configuration du canal : où et comment les notifications doivent être envoyées

- Préférences de charge utile : quel niveau de détail doit être inclus dans les notifications

Canaux de notification

HealthLake prend en charge les canaux de notification suivants :

Type de canal	Cas d’utilisation	
EventBridge	Intégrations d'entreprise, flux de travail sans serveur, orchestration multiservices AWS	
REST Hook	Notifications directes des terminaux, intégration de systèmes tiers	

Charges utiles de notification

Choisissez le type de charge utile approprié en fonction de vos besoins :

Type de charge utile	Description	Considérations sur la sécurité
Identifiant uniquement	Contient uniquement des identificateurs de ressources	Exposition minimale aux PHI
Ressource complète	Contient le contenu complet des ressources d'une taille maximale de 256 Ko. Si la taille est supérieure à 256 Ko, elle reviendra à l'ID uniquement	Contient des données PHI ; vérifiez le traitement sécurisé

Bonnes pratiques

Optimisation des performances

- Utilisez des filtres ciblés : affinez vos critères pour ne recevoir que les notifications essentielles
- Choisissez les types de charge utile appropriés : utilisez des charges utiles uniquement par identifiant lorsque cela est possible pour de meilleures performances
- Mettez en œuvre des récepteurs efficaces : assurez-vous que les destinataires des notifications traitent les messages rapidement

Considérations sur la sécurité

- Points de terminaison sécurisés : implémentez une authentification appropriée pour les points de terminaison REST Hook
- Protection des PHI : soyez prudent avec les charges utiles contenant des ressources complètes, car elles contiennent des PHI
- Contrôle d'accès : restreindre la création d'abonnements aux utilisateurs autorisés uniquement

Excellence opérationnelle

- Définissez des dates de fin appropriées : utilisez des dates de fin pour les abonnements temporaires
- Surveiller l'état de vos abonnements : vérifiez régulièrement l'état de vos abonnements
- Implémentation de la gestion des erreurs : concevez vos applications pour gérer les échecs de livraison des notifications

Le cycle de vie de l'abonnement FHIR avec AWS HealthLake

Suivez ces étapes pour comprendre le cycle de vie de l'abonnement FHIR :

1. Créer une `SubscriptionTopic`
   - Créer un `SubscriptionTopic` avec statut `"unknown"`
2. Créer une `Subscription`
   - Créer un `Subscription` avec statut `"requested"`
   - HealthLake valide la configuration `Subscription`
   - `Subscription`doit faire référence à un sujet déjà existant (le sujet doit avoir le statut`unknown`,`draft`,`active`).
3. Activation

- S'il est valide, HealthLake met à jour le statut `Subscription` de `"active"`
- Lors de la création d'un`Subscription`, si le sujet indiqué était en statut`"unknown"`, HealthLake met à jour le statut `"active"` une fois que l'abonnement est également actif
- Les abonnements prennent généralement 5 à 10 minutes pour être créés avec succès
- Si la création `Subscription` échoue, le statut passera à l'`error`endroit où vous devez effectuer une opération DELETE, puis réessayer de créer un abonnement. Vous pouvez consulter le `"error"` champ dans la ressource Abonnement pour voir pourquoi l'abonnement n'a pas été créé correctement.

4. Ingestion pendant l'abonnement `active`
5. Alors que `Subscription` c'est `active`
   - HealthLake moniteurs pour les événements correspondant à vos critères
   - Des notifications sont envoyées au point de terminaison configuré lorsque des correspondances se produisent
6. Gestion des erreurs
   - HealthLake tente de réessayer pendant 14 jours, puis arrête de réessayer pour ces événements
7. Désactivation
   - A `Subscription` peut être désactivé par :

     Définition d'une date de fin (désactivation automatique)

```
{
  "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"
        }
      ]
    }
  }
}
```

Supprimer la `Subscription` ressource

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

Création d'un abonnement FHIR avec AWS HealthLake

Le guide suivant vous montre comment créer un abonnement FHIR à l'aide AWS HealthLake de.

Pour créer un abonnement FHIR

1. Créez un `SubscriptionTopic`.

   Exemple de ressource thématique d'abonnement :

```
{
  "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. Préparez votre point de terminaison de notification (canal personnalisé). Les étapes suivantes sont obligatoires pour garantir que le terminal recevra des notifications

   Lors de l'utilisation de REST Hook

   - Faites confiance `events.amazonaws.com` à votre politique de clé KMS si vous utilisez la banque de données CM_CMK.
   - Si vous utilisez une banque de données CM_CMK, vous devez ajouter la `EventBridgeApiDestinations` balise à votre clé KMS avec la valeur de `true`
   - HealthLake utilise OAuth pour authentifier votre point de terminaison REST Hook. Par conséquent, lors de la création d'un abonnement à un hook REST, vous devez transmettre un identifiant client, un secret client et oAuth-endpoint-url le canal. _type.extension [*].

   Exemple de politique de clé KMS en cas d'utilisation de la banque de données CM_CMK :

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

   Lors de l'utilisation EventBridge

   - Faites confiance `events.amazonaws.com` à votre politique de clé KMS si vous utilisez la banque de données CM_CMK.

- Vérifiez que votre politique de EventBridge ressources fait confiance `healthlake.amazonaws.com` au principal du service.
- Lorsque vous utilisez CM_CMK en tant que point de terminaison, vérifiez que vous chiffrez votre EventBridge bus avec la même clé KMS que la clé KMS de la banque de données. EventBridge
- Vérifiez que votre EventBridge bus possède au moins une règle correspondant aux événements générés par HealthLake.

Exemple de politique de ressources pour le bus de EventBridge canaux :

```
{
  "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"
    }
  ]
  }
```

Exemple de modèle d'événement de EventBridge règle pour recevoir des événements depuis : HealthLake

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

> **Note**
>
> HealthLake prend en charge 2 sources :
>
> - “`healthlake`”: Uniquement pour les abonnements.
> - “`aws.healthlake`”: pour recevoir des événements HealthLake de service.

“healthlake”À utiliser comme source lors de la création d'une règle pour les bus d'événements FHIR Subscriptions.

3. Créez votre Subscription

   Soumettez une ressource d'abonnement avec :

   - État : "requested"
   - Référence à l'SubscriptionTopicidentifiant que vous avez choisi
   - Critères de filtrage. Pour plus d'informations, consultez la section Filtrage des notifications pour les filtres pris en charge.
   - Configuration de canal

Exemples de charges utiles d'abonnement

Les exemples de code suivants montrent comment créer des charges utiles d'abonnement.

EventBridge

Abonnement avec event-bridge chaîne et type id-only de charge utile.

```
{
  "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"
        }
      ]
    }
  }
}
```

Abonnement avec `event-bridge` point de terminaison et type `full-resource` de charge utile.

```
{
  "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"
        }
      ]
    }
  }
}
```

Crochet de repos

Abonnement avec `rest-hook` point de terminaison et type `id-only` de charge utile.

```
{
  "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"
        }
      ]
    }
  }
}
```

Abonnement avec `rest-hook` chaîne et type `full-resource` de charge utile.

```
{
  "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"
        }
      ]
    }
  }
}
```

Exemples de charges utiles de notification

Lors de la création d'un abonnement, vérifiez que HealthLake la configuration de l'abonnement est réussie en envoyant un bundle Handshake à la chaîne que vous avez configurée. La charge utile suivante est un exemple de bundle de handshake.

```
{
  "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>"
          }
        }
      ]
    }
```

```
  }
}
```

Exemple de bundle de notifications avec identifiant uniquement.

```
{
  "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"
              }
            ]
          }
        }
      ]
    }
  }
}
```

Exemple de bundle de notifications contenant toutes les ressources.

```
{
  "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"
          }
        }
      ]
    }
```

```
  }
}
```

Versionnage des événements

HealthLake prend en charge l'historique FHIR par défaut.

Pour connaître la version de la ressource que vous avez reçue dans votre bundle de notifications, procédez comme suit :

- ressource complète : étant donné que les ensembles de ressources complets incluent l'intégralité de la ressource, la version sera incluse dans le `entry[*]` pour chaque ressource du bundle.
- id only : les bundles n'incluront aucune information sur les ressources. HealthLakeinclut la version correspondante et incluse dans le bundle via le `entry[0].notificationEvent[*].additionalContext` champ. Ce champ est au format`<ResourceType>/<ResourceId>/_history/<Version Id>`. Pour plus d'informations, consultez le champ AdditionalContext dans l'exemple de charge utile contenant uniquement un identifiant.

Détection de la duplication d'événements

HealthLakeLa fonction d'abonnement FHIR garantit au moins une livraison. Cela signifie que vous pouvez recevoir le même événement plusieurs fois, soit dans le même pack, soit dans un autre pack. Pour identifier les doublons, fournissez HealthLake un identifiant unique pour chaque événement du bundle de notifications dans. `entry[0].notificationEvent[*].id`

Cet identifiant est propre à la version spécifique de l'événement qui a été mise en correspondance et diffusée. Par exemple, si la même rencontre est mise à jour deux fois et que les deux mises à jour correspondent aux critères du filtre, vous recevrez deux événements distincts avec la même référence de rencontre. Ils auront le mêmenotificationEvent[*].focus, mais ils auront un uniquenotificationEvent[*].id. En outre, ces événements peuvent être envoyés dans des ensembles distincts ou dans le même ensemble de notifications.

Recherche d'abonnements FHIR avec AWS HealthLake

`Subscription`et les `SubscriptionTopic` ressources sont également consultables. HealthLake prend en charge tous les paramètres de recherche courants pour les abonnements et SubscriptionTopic les ressources.

En outre, nous prenons en charge des fonctionnalités de recherche supplémentaires grâce aux paramètres suivants :

Abonnement

Paramètre de recherche	Description	Exemple
contact	Effectuez une recherche dans le champ Subscription.Contact dans la spécification de base du R4	`Subscription?contact=phone`
criteria	Effectuez une recherche dans le champ SUBSCRIPTION.CRITERIA dans la spécification de base R4	`Subscription?criteria=[baseUrl]/datastore/[datastoreId]/r4/SubscriptionTopic/[topicId]`
payload	Effectuez une recherche dans le champ subscription.channel.payload dans la spécification de base R4	`Subscription?payload=application/fhir+json`
status	Effectuez une recherche dans le champ SUBSCRIPTION.STATUS dans la spécification de base R4	`Subscription?status=error`
type	Effectuez une recherche dans le champ subscription.channel.type dans la spécification de base R4	`Subscription?topic=event-bridge`
url	Effectuez une recherche dans le champ subscription.channel.Endpoint dans la spécification de base R4	`Subscription?url=[Subscription.channel.endpoint]`

Paramètre de recherche	Description	Exemple
critères-filtres	Effectuez une recherche dans le champ d'extension des critères avec l'url « http://hl 7 ». org/fhir/uv/subscriptions-backport/StructureDefinition/backport-filter-criteria » sous forme de chaîne	`Subscription?filter-criteria=Encounter?`
canal personnalisé	Recherchez dans le champ d'extension de type de canal personnalisé avec l'url « http:// hl7 ». org/fhir/uv/subscriptions-backport/StructureDefinition /backport-channel-type » en tant que codage Exemple de charge utile d'abonnement	`Subscription?custom-channel=[System]|[code]`
type de charge utile	Recherchez dans le champ d'extension du type de charge utile où vous spécifiez le format de la charge utile (soit) `id-only` `full-resource`	`Subscription?payload-type=full-resource`
topic	Effectuez une recherche dans le champ SUBSCRIPTION.CRITERIA, où le sujet est ajouté	`Subscription?topic=[topicId]`

SubscriptionTopic

Paramètre de recherche	Description	Exemple
date	Effectuez une recherche dans le champ de date de la SubscriptionTopic ressource	`SubscriptionTopic?date=[SubscriptionTopic.date]`
derived-or-self	Effectuez une recherche dans `derivedFrom` les champs `url` ou dans la SubscriptionTopic ressource	`SubscriptionTopic?derived-or-self=[SubscriptionTopic.url | SubscriptionTopic.derivedFrom]`
identifiant	Effectuez une recherche dans le SubscriptoinTopic champ .identifier	`SubscriptionTopic?identifier=[SubscriptionTopic.identifier]`
ressource	Recherchez dans le champ SubscriptionTopic .Resource Trigger.Resource	`SubscriptionTopic?resource=Encounter`
status	Effectuez une recherche sur le statut du SubscriptionTopic	`SubscriptionTopic?status=unknown`
title	Recherchez sur le titre un SubscriptionTopic	`SubscriptionTopic?title=admission`
description du déclencheur	Rechercher sur SubscriptionTopic .ResourceTrigger.Description	`SubscriptionTopic.trigger-description=resource moving to state 'in-progress'`
url	Recherchez sur l'URL du SubscriptionTopic	`SubscriptionTopic?url=[SubscriptionTopic.url]`

Paramètre de recherche	Description	Exemple
version	Recherchez sur la version du SubscriptionTopic	`SubscriptionTopic?version=1`

Filtrer les notifications avec AWS HealthLake

Affinez vos notifications en utilisant les paramètres de recherche FHIR standard de vos `Subscription` critères.

Filtres pris en charge

Le tableau suivant présente la liste des critères de filtre pris HealthLake en charge pour les abonnements.

Types de paramètres de recherche	Type de données FHIR	Modificateurs	Préfixes pris en charge
String	chaîne HumanName Adresse	exact, contient des éléments manquants	
Jeton	boolean code chaîne Codage CodeableConcept Identifiant ContactPoint id	non, texte, manquant	

Types de paramètres de recherche	Type de données FHIR	Modificateurs	Préfixes pris en charge
Number	entier decimal Int positif Int non signé	manquant	« eq », « ne », « gt », « lt », « ge », « le », « sa », « eb », « ap »
Date	date dateTime instantané Period Timing (Durée)		« eq », « ne », « gt », « lt », « ge », « le », « sa », « eb », « ap »
Quantity	Quantity Argent Range SimpleQuantity		« eq », « ne », « gt », « lt », « ge », « le », « sa », « eb », « ap »
Référence	Référence	manquant, identifiant, type	
URI	uri url canonial uid oid	manquant	

Exemples de filtres pris en charge

Le tableau suivant présente des exemples de critères de filtre pris en HealthLake charge pour les abonnements :

Objectif	Critères de filtrage	Description
Observations spécifiques au patient	`Observation?patient=Patient/[id]&status=final`	Recevez des notifications lorsque les observations pour un patient spécifique sont finalisées
Observations spécifiques au patient	`Patient?birthdate=gt2021`	Recevez des notifications lorsque des patients nés après 2021 sont enregistrés ou mis à jour
Observations spécifiques au patient	`Condition?code=http://snomed.info/sct|39065001`	Recevez des notifications pour des conditions avec des codes SNOMED spécifiques
Observations spécifiques au patient	`Observation?code=http://loinc.org|8480-6&value-quantity=gt160`	Recevez des notifications pour les mesures d'hypertension artérielle systolique

Importation de données FHIR avec AWS HealthLake

Après avoir créé un magasin de HealthLake données, l'étape suivante consiste à importer des fichiers depuis un compartiment Amazon Simple Storage Service (S3). Vous pouvez démarrer une tâche d'importation FHIR en utilisant le AWS Management Console AWS CLI, ou AWS SDKs. Utilisez AWS HealthLake des actions natives pour démarrer, décrire et répertorier les tâches d'importation FHIR.

> **Important**
>
> HealthLake prend en charge la spécification FHIR R4 pour l'échange de données sur les soins de santé. Si nécessaire, vous pouvez travailler avec un AWS HealthLake partenaire pour convertir vos données de santé au format FHIR R4 avant de les importer.

Lorsque vous démarrez une tâche d'importation FHIR, vous spécifiez un emplacement d'entrée de compartiment Amazon S3, un emplacement de sortie de compartiment Amazon S3 (pour les résultats du traitement de la tâche), un rôle IAM qui donne HealthLake accès à vos compartiments Amazon S3 et une clé détenue ou AWS détenue par le client. AWS Key Management Service Pour de plus amples informations, veuillez consulter Configuration des autorisations pour les tâches d'importation.

> **Note**
>
> Vous pouvez mettre en file d'attente les tâches d'importation. Les tâches d'importation asynchrones sont traitées selon le mode FIFO (First In First Out). Vous pouvez mettre les tâches en file d'attente de la même manière que vous commencez à importer des tâches. Si l'un d'entre eux est en cours, il sera simplement mis en file d'attente. Vous pouvez créer, lire, mettre à jour ou supprimer des ressources FHIR pendant qu'une tâche d'importation est en cours.

HealthLake génère un `manifest.json` fichier pour chaque tâche d'importation FHIR. Le fichier décrit à la fois les réussites et les échecs d'une tâche d'importation FHIR. HealthLake envoie le `manifest.json` fichier dans le compartiment Amazon S3 spécifié lors du démarrage d'une tâche d'importation FHIR. Les fichiers journaux sont organisés en deux dossiers, nommés `SUCCESS` et`FAILURE`. Utilisez le `manifest.json` fichier comme première étape pour résoudre les problèmes liés à l'échec d'une tâche d'importation, car il fournit des informations détaillées sur chaque fichier.

```
{
    "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
}
```

Configuration du niveau de validation pour les importations

Lorsque vous démarrez une tâche d'importation FHIR, vous pouvez éventuellement spécifier un `ValidationLevel` à appliquer à chaque ressource. AWS HealthLake prend actuellement en charge les niveaux de validation suivants :

- `strict`: Les ressources sont validées en fonction de l'élément de profil de la ressource ou de la spécification R4 si aucun profil n'est présent. Il s'agit du niveau de validation par défaut pour AWS HealthLake.
- `structure-only`: Les ressources sont validées par rapport à R4, en ignorant les profils référencés.

- `minimal`: Les ressources sont validées de manière minimale, sans tenir compte de certaines règles R4. Les ressources qui échouent aux vérifications de structure requises search/analytics seront mises à jour pour inclure un avertissement d'audit.

Lors de l'importation à l'aide du niveau de `minimal` validation, des fichiers journaux supplémentaires peuvent être générés dans un dossier nomméSUCCESS_WITH_SEARCH_VALIDATION_FAILURES. Les ressources contenues dans les fichiers journaux de ce dossier ont été ingérées dans votre banque de données malgré l'échec des contrôles de validation liés à la recherche. Cela implique que certains aspects de votre ressource FHIR n'étaient pas valides selon FHIR et que les champs mal formés peuvent ne pas être consultables. Ces ressources seront accompagnées d'`extension`une annexe décrivant ladite défaillance.

Rubriques

Démarrage d'une tâche d'importation FHIR

`StartFHIRImportJob`À utiliser pour démarrer une tâche d'importation FHIR dans un magasin de HealthLake données. Les menus suivants fournissent une procédure AWS Management Console et des exemples de code pour le AWS CLI et AWS SDKs. Pour plus d'informations, consultez `StartFHIRImportJob` dans la Référence d'API AWS HealthLake .

> **Important**
>
> HealthLake prend en charge la spécification FHIR R4 pour l'échange de données sur les soins de santé. Si nécessaire, vous pouvez travailler avec un AWS HealthLake partenaire pour convertir vos données de santé au format FHIR R4 avant de les importer.

Pour démarrer une tâche d'importation FHIR

Choisissez un menu en fonction de vos préférences d'accès à AWS HealthLake.

AWS CLI et SDKs

CLI

AWS CLI

Pour démarrer une tâche d'importation FHIR

L'`start-fhir-import-job`exemple suivant montre comment démarrer une tâche d'importation FHIR à l'aide AWS HealthLake de.

```
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)"
```

Sortie :

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

- Pour plus de détails sur l'API, consultez Start FHIRImport Job dans AWS CLI Command Reference.

Python

Kit 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
```

- Pour plus de détails sur l'API, consultez le manuel de référence de l'API Start FHIRImport Job in AWS SDK for Python (Boto3).

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    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.
```

- Pour plus de détails sur l'API, voir Start FHIRImport Job in AWS SDK for SAP ABAP API reference.

> **Exemple de disponibilité**
>
> Vous n’avez pas trouvé ce dont vous avez besoin ? Demandez un exemple de code à l'aide du lien Fournir des commentaires dans la barre latérale droite de cette page.

AWS Console

1. Connectez-vous à la page Data stores de la HealthLake console.
2. Choisissez un magasin de données.
3. Choisissez Importer.

   La page d'importation s'ouvre.

4. Dans la section Données d'entrée, entrez les informations suivantes :
   - Emplacement des données d'entrée dans Amazon S3
5. Dans la section Importer des fichiers de sortie, entrez les informations suivantes :
   - Emplacement des fichiers de sortie d'importation dans Amazon S3
   - Chiffrement des fichiers de sortie d'importation
6. Dans la section Autorisations d'accès, choisissez Utiliser un rôle de service IAM existant et sélectionnez le rôle dans le menu Nom du rôle de service ou choisissez Créer un rôle IAM.
7. Choisissez Importer les données.

> **Note**
>
> Lors de l'importation, choisissez Copier l'identifiant de la tâche sur la bannière en haut de la page. Vous pouvez utiliser le `JobID`pour demander les propriétés de la tâche d'importation à l'aide du AWS CLI. Pour de plus amples informations, veuillez consulter Obtenir les propriétés des tâches d'importation FHIR.

Obtenir les propriétés des tâches d'importation FHIR

`DescribeFHIRImportJob`À utiliser pour obtenir les propriétés de la tâche d'importation FHIR. Les menus suivants fournissent une procédure AWS Management Console et des exemples de code pour le AWS CLI et AWS SDKs. Pour plus d'informations, consultez `DescribeFHIRImportJob` dans la Référence d'API AWS HealthLake .

Pour obtenir les propriétés de la tâche d'importation FHIR

Choisissez un menu en fonction de vos préférences d'accès à AWS HealthLake.

AWS CLI et SDKs

CLI

AWS CLI

Pour décrire une tâche d'importation FHIR

L'`describe-fhir-import-job`exemple suivant montre comment apprendre les propriétés d'une tâche d'importation FHIR à l'aide AWS HealthLake de.

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

Sortie :

```
{
    "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)"
    }
}
```

- Pour plus de détails sur l'API, voir FHIRImportDescribe Job dans AWS CLI Command Reference.

Python

Kit 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
```

- Pour plus de détails sur l'API, consultez le manuel de référence de l'API Describe FHIRImport Job in AWS SDK for Python (Boto3).

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    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.
```

- Pour plus de détails sur l'API, consultez la section de référence de l'API Describe FHIRImport Job in AWS SDK for SAP ABAP.

Exemple de disponibilité

Vous n’avez pas trouvé ce dont vous avez besoin ? Demandez un exemple de code à l'aide du lien Fournir des commentaires dans la barre latérale droite de cette page.

AWS Console

Note

Les informations relatives aux tâches d'importation FHIR ne sont pas disponibles sur la HealthLake console. Utilisez plutôt le AWS CLI with `DescribeFHIRImportJob` pour demander des propriétés de tâche d'importation telles que `JobStatus`. Pour plus d'informations, reportez-vous à l' AWS CLI exemple présenté sur cette page.

Liste des tâches d'importation FHIR

Permet `ListFHIRImportJobs` de répertorier les tâches d'importation FHIR pour un magasin de HealthLake données actif. Les menus suivants fournissent une procédure AWS Management Console et des exemples de code pour le AWS CLI et AWS SDKs. Pour plus d’informations, consultez `ListFHIRImportJobs` dans la Référence d’API AWS HealthLake .

Pour répertorier les tâches d'importation FHIR

Choisissez un menu en fonction de vos préférences d'accès à AWS HealthLake.

AWS CLI et SDKs

CLI

AWS CLI

Pour répertorier toutes les tâches d’importation FHIR

L’exemple `list-fhir-import-jobs` suivant montre comment utiliser la commande pour afficher une liste de toutes les tâches d’importation associées à un compte.

```
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)
```

Sortie :

```
{
    "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)"
        }
    ]
}
```

- Pour plus de détails sur l'API, consultez la section FHIRImportRépertorier les tâches dans AWS CLI Command Reference.

Python

Kit 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
```

- Pour plus de détails sur l'API, consultez le manuel de référence de l'API List FHIRImport Jobs in AWS SDK for Python (Boto3).

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    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.
```

- Pour plus de détails sur l'API, consultez la section FHIRImportRépertorier les tâches dans le AWS SDK pour la référence de l'API SAP ABAP.

Exemple de disponibilité

Vous n’avez pas trouvé ce dont vous avez besoin ? Demandez un exemple de code à l'aide du lien Fournir des commentaires dans la barre latérale droite de cette page.

AWS Console

Note

Les informations relatives aux tâches d'importation FHIR ne sont pas disponibles sur la HealthLake console. Utilisez plutôt le AWS CLI with `ListFHIRImportJobs` pour répertorier toutes les tâches d'importation FHIR. Pour plus d'informations, reportez-vous à l' AWS CLI exemple présenté sur cette page.

Gestion des ressources FHIR dans AWS HealthLake

Utilisez les interactions de l' RESTful API FHIR R4 pour gérer les ressources FHIR dans un HealthLake magasin de données. Les sections suivantes décrivent toutes les interactions de l' RESTful API FHIR R4 HealthLake prises en charge et disponibles pour la gestion des ressources FHIR. Pour plus d'informations sur HealthLake les fonctionnalités du magasin de données et sur les parties de la spécification FHIR prises en charge, consultezDéclaration de capacité du FHIR R4 pour AWS HealthLake.

> **Note**
>
> Les interactions FHIR répertoriées dans ce chapitre sont conçues conformément à la norme HL7 FHIR R4 pour l'échange de données sur les soins de santé. Parce qu'ils sont des représentations des services HL7 FHIR, ils ne sont pas proposés par le biais de AWS CLI et AWS SDKs. Pour plus d'informations, consultez la section RESTful API dans la documentation de l' RESTful API FHIR R4.

Le tableau suivant répertorie les interactions FHIR R4 prises en charge par. AWS HealthLake Pour plus d'informations sur les types de ressources FHIR pris en charge par HealthLake, consultezTypes de ressources.

Interactions FHIR R4 prises en charge par AWS HealthLake

Interaction	Description
Interactions avec l'ensemble du système	
`capabilities`	Obtenez une déclaration de capacité pour le système. Consultez Déclaration de capacité du FHIR R4 pour AWS HealthLake.
`batch`	Mettez à jour, créez ou supprimez un ensemble de ressources en une seule interaction. Consultez Regroupement des ressources FHIR.
Interactions au niveau du type	
`create`	Créez une nouvelle ressource avec un ID attribué par le serveur. Consultez Création d'une ressource FHIR.

Interaction	Description
`search`	Recherchez un type de ressource en fonction de certains critères de filtrage. Consultez Recherche de ressources FHIR.
`history`	Récupérez l'historique des modifications pour un type de ressource spécifique. Consultez Lire l'historique des ressources du FHIR.
Interactions au niveau de l'instance	
`read`	Lisez l'état actuel d'une ressource. Consultez Lire une ressource FHIR.
`history`	Lisez l'historique des modifications pour une ressource donnée. Consultez Lire l'historique des ressources du FHIR.
`vread`	Lisez l'état d'une version spécifique de la ressource. Consultez Lire l'historique des ressources FHIR spécifiques à une version.
`update`	Mettez à jour une ressource par son identifiant (ou créez-la s'il s'agit d'une nouvelle ressource). Consultez Mettre à jour une ressource FHIR.
`delete`	Supprimez une ressource. Consultez Supprimer une ressource FHIR.

Rubriques

- Création d'une ressource FHIR
- Lire une ressource FHIR
- Lire l'historique des ressources du FHIR
- Mettre à jour une ressource FHIR
- Modification des ressources à l'aide de l'opération PATCH
- Regroupement des ressources FHIR
- Supprimer une ressource FHIR
- Idempotencie et simultanéité

Création d'une ressource FHIR

L'`create`interaction FHIR crée une nouvelle ressource FHIR dans un magasin de HealthLake données. Pour plus d'informations, consultez la `create`documentation de l' RESTful API FHIR R4.

Pour créer une ressource FHIR

1. Collectez HealthLake `region` et `datastoreId` valorisez. Pour de plus amples informations, veuillez consulter Obtenir les propriétés du magasin de données.
2. Déterminez le type de FHIR `Resource` à créer. Pour de plus amples informations, veuillez consulter Types de ressources.
3. Créez une URL pour la demande en utilisant les valeurs collectées pour HealthLake `region` et`datastoreId`. Incluez également le `Resource` type FHIR à créer. Pour afficher le chemin complet de l'URL dans l'exemple suivant, faites défiler le curseur sur le bouton Copier.

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

4. Construisez un corps JSON pour la demande, en spécifiant les données FHIR de la nouvelle ressource. Dans le cadre de cette procédure, nous utilisons une `Patient` ressource FHIR. Enregistrez donc le fichier sous`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. Envoyez la demande . L'`create`interaction FHIR utilise une `POST` demande avec AWS signature version 4 ou SMART sur autorisation FHIR. Les exemples suivants créent une `Patient` ressource FHIR en HealthLake utilisant curl ou la HealthLake console. Pour afficher un exemple complet, faites défiler la souris sur le bouton Copier.

   SigV4

   Autorisation SigV4

   ```
   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

   Exemple d'autorisation SMART sur FHIR pour le type de `IdentityProviderConfiguration`données.

   ```
   {
       "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\"]}"
   ```

```
}
```

L'appelant peut attribuer des autorisations dans le lambda d'autorisation. Pour de plus amples informations, veuillez consulter Étendue OAuth 2.0.

AWS Console

> **Note**
>
> La HealthLake console prend uniquement en charge l'autorisation AWS SigV4.

1. Connectez-vous à la page Exécuter une requête sur la HealthLake console.
2. Dans la section Paramètres de requête, effectuez les sélections suivantes.
   - ID du magasin de données : choisissez un identifiant de magasin de données pour générer une chaîne de requête.
   - Type de requête : choisissez`Create`.
   - Type de ressource : choisissez le type de ressource FHIR à créer.
   - Corps de la demande : créez un corps JSON pour la demande, en spécifiant les données FHIR de la nouvelle ressource.
3. Choisissez Exécuter la requête.

Configuration du niveau de validation pour la création de ressources

Lorsque vous créez une ressource FHIR, vous pouvez éventuellement spécifier un en-tête `x-amzn-healthlake-fhir-validation-level` HTTP pour configurer un niveau de validation pour la ressource. AWS HealthLake prend actuellement en charge les niveaux de validation suivants :

- `strict`: Les ressources sont validées en fonction de l'élément de profil de la ressource ou de la spécification R4 si aucun profil n'est présent. Il s'agit du niveau de validation par défaut pour AWS HealthLake.
- `structure-only`: Les ressources sont validées par rapport à R4, en ignorant les profils référencés.

- `minimal`: Les ressources sont validées de manière minimale, sans tenir compte de certaines règles R4. Les ressources qui échouent aux vérifications de structure requises search/analytics seront mises à jour pour inclure un avertissement d'audit.

Les ressources créées avec le niveau de validation minimal peuvent être ingérées dans une banque de données malgré l'échec de la validation requise pour l'indexation des recherches. Dans ce cas, les ressources seront mises à jour pour inclure une extension spécifique à Healthlake afin de documenter ces échecs :

```
{
    "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.\"}]}"
}
```

En outre, l'en-tête de réponse HTTP suivant sera inclus avec la valeur « true » :

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

> **Note**
>
> Les données ingérées qui sont mal formées conformément à la spécification R4 peuvent ne pas être consultables comme prévu si ces erreurs sont présentes.

Lire une ressource FHIR

L'`read`interaction FHIR lit l'état actuel d'une ressource dans un magasin de HealthLake données. Pour plus d'informations, consultez la `read`documentation de l' RESTful API FHIR R4.

Pour lire une ressource FHIR

1. Collectez HealthLake `region` et `datastoreId` valorisez. Pour de plus amples informations, veuillez consulter Obtenir les propriétés du magasin de données.
2. Déterminez le type de FHIR `Resource` à lire et collectez la `id` valeur associée. Pour de plus amples informations, veuillez consulter Types de ressources.

3. Construisez une URL pour la demande en utilisant les valeurs collectées pour HealthLake `region` et`datastoreId`. Incluez également le `Resource` type FHIR et son associé`id`. Pour afficher le chemin complet de l'URL dans l'exemple suivant, faites défiler le curseur sur le bouton Copier.

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

4. Envoyez la demande . L'`read`interaction FHIR utilise une `GET` demande avec AWS signature version 4 ou SMART sur autorisation FHIR. L'`curl`exemple suivant lit l'état actuel d'une `Patient` ressource FHIR dans HealthLake. Pour afficher l'exemple dans son intégralité, faites défiler la souris sur le bouton Copier.

   SigV4

   Autorisation SigV4

```
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

   Exemple d'autorisation SMART sur FHIR pour le type de `IdentityProviderConfiguration`données.

```
{
    "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\"]}"
}
```

L'appelant peut attribuer des autorisations dans le lambda d'autorisation. Pour de plus amples informations, veuillez consulter Étendue OAuth 2.0.

AWS Console

1. Connectez-vous à la page Exécuter une requête sur la HealthLake console.
2. Dans la section Paramètres de requête, effectuez les sélections suivantes.
   - ID du magasin de données : choisissez un identifiant de magasin de données pour générer une chaîne de requête.
   - Type de requête : choisissez`Read`.
   - Type de ressource : choisissez le type de ressource FHIR à lire.
   - ID de ressource — entrez l'ID de ressource FHIR.
3. Choisissez Exécuter la requête.

Lire l'historique des ressources du FHIR

L'`history`interaction FHIR permet de récupérer l'historique d'une ressource FHIR particulière dans un HealthLake magasin de données. Grâce à cette interaction, vous pouvez déterminer l'évolution du contenu d'une ressource FHIR au fil du temps. Il est également utile en coordination avec les journaux d'audit pour voir l'état d'une ressource avant et après modification. Les interactions `create` FHIR et `delete` aboutissent à une version historique de la ressource à enregistrer. `update` Pour plus d'informations, consultez la `history`documentation de l' RESTful API FHIR R4.

> **Note**
>
> Vous pouvez choisir de ne pas utiliser `history` des types de ressources FHIR spécifiques. Pour vous désinscrire, créez un dossier en utilisant AWS Support Center Console. Pour créer votre dossier, connectez-vous à votre dossier Compte AWS et choisissez Créer un dossier.

Pour lire l'historique des ressources du FHIR

1. Collectez HealthLake `region` et `datastoreId` valorisez. Pour de plus amples informations, veuillez consulter Obtenir les propriétés du magasin de données.
2. Déterminez le type de FHIR `Resource` à lire et collectez la `id` valeur associée. Pour de plus amples informations, veuillez consulter Types de ressources.
3. Créez une URL pour la demande en utilisant les valeurs collectées pour HealthLake `region` et`datastoreId`. Incluez également le `Resource` type FHIR, ses paramètres de recherche associés `id` et facultatifs. Pour afficher le chemin complet de l'URL dans l'exemple suivant, faites défiler le curseur sur le bouton Copier.

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

HealthLake paramètres de recherche pris en charge pour l'interaction FHIR **`history`**

Paramètre	Description
`_count : integer`	Le nombre maximum de résultats de recherche sur une page. Le serveur renverra le nombre demandé ou le nombre maximum de résultats de recherche autorisés par défaut pour le magasin de données, le plus faible des deux étant retenu.
`_since : instant`	N'incluez que les versions de ressources créées à l'instant donné ou après.
`_at : date(Time)`	N'incluez que les versions de ressources qui étaient à jour à un moment donné pendant

Paramètre	Description
	la période spécifiée dans la valeur de date et d'heure. Pour plus d'informations, consultez la documentation *date*de l' RESTfulAPI HL7 FHIR.

4. Envoyez la demande . L'`history`interaction FHIR utilise une `GET` demande avec AWS signature version 4 ou SMART sur autorisation FHIR. L'`curl`exemple suivant utilise le paramètre `_count` de recherche pour renvoyer 100 résultats de recherche historiques par page pour une `Patient` ressource FHIR dans HealthLake. Pour afficher l'exemple dans son intégralité, faites défiler la souris sur le bouton Copier.

   SigV4

   Autorisation SigV4

   ```
   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

   Exemple d'autorisation SMART sur FHIR pour le type de `IdentityProviderConfiguration`données.

   ```
   {
       "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\"]}"
}
```

L'appelant peut attribuer des autorisations dans le lambda d'autorisation. Pour de plus amples informations, veuillez consulter Étendue OAuth 2.0.

Le contenu renvoyé par une `history` interaction est contenu dans une ressource FHIR`Bundle`, dont le type est défini sur. `history` Il contient l'historique des versions spécifié, trié avec les versions les plus anciennes en dernier et inclut les ressources supprimées. Pour plus d'informations, consultez la `Resource Bundle`documentation du FHIR R4.

Lire l'historique des ressources FHIR spécifiques à une version

L'`vread`interaction FHIR effectue une lecture spécifique à la version d'une ressource dans un HealthLake magasin de données. Grâce à cette interaction, vous pouvez visualiser le contenu d'une ressource FHIR tel qu'il était à un moment donné dans le passé.

> **Note**
>
> Si vous utilisez l'`history`interaction FHIR sans`vread`, renvoie HealthLake toujours la dernière version des métadonnées de la ressource.

HealthLake déclare qu'il prend en charge le versionnement `CapabilityStatement.rest.resource.versioning`pour chaque ressource prise en charge. Tous les magasins de HealthLake données incluent `Resource.meta.versionId` (`vid`) sur toutes les ressources.

Lorsque `history` l'interaction FHIR est activée (par défaut pour les magasins de données créés après le 25/10/2024 ou par demande pour les anciens magasins de données), la `Bundle` réponse inclut `vid` le dans l'élément. `location` Dans l'exemple suivant, le `vid` s'affiche sous forme de numéro1. Pour voir l'exemple complet, voir Example bundle/bundle-response (JSON).

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

Pour lire l'historique des ressources FHIR spécifiques à une version

1. Collectez HealthLake `region` et `datastoreId` valorisez. Pour de plus amples informations, veuillez consulter Obtenir les propriétés du magasin de données.
2. Déterminez le `Resource` type de FHIR à lire et à collecter les `vid` valeurs `id` et associées. Pour de plus amples informations, veuillez consulter Types de ressources.
3. Construisez une URL pour la demande en utilisant les valeurs collectées pour HealthLake et FHIR. Pour afficher le chemin complet de l'URL dans l'exemple suivant, faites défiler le curseur sur le bouton Copier.

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

4. Envoyez la demande . L'`history`interaction FHIR utilise une `GET` demande avec AWS signature version 4 ou SMART sur autorisation FHIR. L'`vread`interaction suivante renvoie une seule instance avec le contenu spécifié pour la `Patient` ressource FHIR pour la version des métadonnées de ressource spécifiée par le`vid`. Pour afficher le chemin complet de l'URL dans l'exemple suivant, faites défiler le curseur sur le bouton Copier.

   SigV4

   Autorisation SigV4

   ```
   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

Exemple d'autorisation SMART sur FHIR pour le type de `IdentityProviderConfiguration`données.

```
{
    "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\"]}"
}
```

L'appelant peut attribuer des autorisations dans le lambda d'autorisation. Pour de plus amples informations, veuillez consulter Étendue OAuth 2.0.

Mettre à jour une ressource FHIR

L'`update`interaction FHIR crée une nouvelle version actuelle pour une ressource existante ou crée une version initiale si aucune ressource n'existe déjà pour la ressource donnée`id`. Pour plus d'informations, consultez la `update`documentation de l' RESTful API FHIR R4.

Pour mettre à jour une ressource FHIR

1. Collectez HealthLake `region` et `datastoreId` valorisez. Pour de plus amples informations, veuillez consulter Obtenir les propriétés du magasin de données.

2. Déterminez le type de FHIR `Resource` à mettre à jour et collectez la `id` valeur associée. Pour de plus amples informations, veuillez consulter Types de ressources.

3. Construisez une URL pour la demande en utilisant les valeurs collectées pour HealthLake `region` et`datastoreId`. Incluez également le `Resource` type FHIR et son associé`id`. Pour afficher le chemin complet de l'URL dans l'exemple suivant, faites défiler le curseur sur le bouton Copier.

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

4. Construisez un `JSON` corps pour la demande, en spécifiant les mises à jour des données FHIR à effectuer. Dans le cadre de cette procédure, enregistrez le fichier sous`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. Envoyez la demande . L'`update`interaction FHIR utilise une PUT demande avec AWS signature version 4 ou SMART sur autorisation FHIR. L'`curl`exemple suivant met à jour une `Patient` ressource dans HealthLake. Pour afficher l'exemple dans son intégralité, faites défiler la souris sur le bouton Copier.

SigV4

Autorisation SigV4

```
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
```

Votre demande renverra un code d'état `200` HTTP si une ressource existante est mise à jour ou un code d'état `201` HTTP si une nouvelle ressource est créée.

SMART on FHIR

Exemple d'autorisation SMART sur FHIR pour le type de `IdentityProviderConfiguration`données.

```
{
    "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\"]}"
}
```

L'appelant peut attribuer des autorisations dans le lambda d'autorisation. Pour de plus amples informations, veuillez consulter Étendue OAuth 2.0.

AWS Console

1. Connectez-vous à la page Exécuter une requête sur la HealthLake console.
2. Dans la section Paramètres de requête, effectuez les sélections suivantes.
   - ID du magasin de données : choisissez un identifiant de magasin de données pour générer une chaîne de requête.
   - Type de requête : choisissez`Update (PUT)`.
   - Type de ressource : choisissez le type de ressource FHIR à mettre à jour ou à créer.
   - Corps de la demande : créez un corps JSON pour la demande, en spécifiant les données FHIR avec lesquelles mettre à jour la ressource.
3. Choisissez Exécuter la requête.

Mise à jour des ressources du FHIR en fonction des conditions

La mise à jour conditionnelle vous permet de mettre à jour une ressource existante en fonction de certains critères de recherche d'identification, plutôt que selon un FHIR `id` logique. Lorsque le serveur traite la mise à jour, il effectue une recherche à l'aide de ses fonctionnalités de recherche standard pour le type de ressource, dans le but de résoudre une seule logique `id` pour la demande.

L'action entreprise par le serveur dépend du nombre de correspondances qu'il trouve :

- Aucune correspondance, aucune information **`id`** fournie dans le corps de la requête : le serveur crée la ressource FHIR.
- Aucune correspondance, **`id`** fournie et la ressource n'existe pas déjà avec le **`id`** : Le serveur traite l'interaction comme une interaction Mettre à jour en tant qu'interaction Créer.
- Aucune correspondance, **`id`** fournie et existante : le serveur rejette la mise à jour avec une `409 Conflict` erreur.
- Une correspondance, aucune ressource **`id`** fournie OU (ressource **`id`** fournie et elle correspond à la ressource trouvée) : Le serveur effectue la mise à jour par rapport à la ressource correspondante comme ci-dessus où, si la ressource a été mise à jour, le serveur DOIT renvoyer un`200 OK`.

- Une correspondance, ressource **`id`** fournie mais ne correspondant pas à la ressource trouvée : le serveur renvoie une `409 Conflict` erreur indiquant que la spécification de l'identifiant du client posait problème, de préférence avec un `OperationOutcome`
- Correspondances multiples : le serveur renvoie une `412 Precondition Failed` erreur indiquant que les critères du client n'étaient pas suffisamment sélectifs, de préférence avec un OperationOutcome

L'exemple suivant met à jour une `Patient` ressource dont le nom est Peter, la date de naissance est le 1er janvier 2000 et le numéro de téléphone est 1234567890.

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

Configuration du niveau de validation pour les mises à jour des ressources

Lors de la mise à jour d'une ressource FHIR, vous pouvez éventuellement spécifier un en-tête `x-amzn-healthlake-fhir-validation-level` HTTP pour configurer un niveau de validation pour la ressource. AWS HealthLake prend actuellement en charge les niveaux de validation suivants :

- `strict`: Les ressources sont validées en fonction de l'élément de profil de la ressource ou de la spécification R4 si aucun profil n'est présent. Il s'agit du niveau de validation par défaut pour AWS HealthLake.
- `structure-only`: Les ressources sont validées par rapport à R4, en ignorant les profils référencés.
- `minimal`: Les ressources sont validées de manière minimale, sans tenir compte de certaines règles R4. Les ressources qui échouent aux vérifications de structure requises search/analytics seront mises à jour pour inclure un avertissement d'audit.

Les ressources mises à jour avec le niveau de validation minimal peuvent être ingérées dans une banque de données malgré l'échec de la validation requise pour l'indexation des recherches. Dans ce cas, les ressources seront mises à jour pour inclure une extension spécifique à Healthlake afin de documenter ces échecs :

```
{
    "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.\"}]}"
}
```

En outre, l'en-tête de réponse HTTP suivant sera inclus avec la valeur « true » :

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

> **Note**
>
> Notez que les données ingérées qui sont mal formées conformément à la spécification R4 peuvent ne pas être consultables comme prévu si ces erreurs sont présentes.

Modification des ressources à l'aide de l'opération PATCH

AWS HealthLake prend en charge l'opération PATCH pour les ressources FHIR, vous permettant de modifier les ressources en ciblant des éléments spécifiques à ajouter, remplacer ou supprimer sans mettre à jour l'intégralité de la ressource. Cette opération est particulièrement utile lorsque vous devez :

- Procéder à des mises à jour ciblées à des ressources importantes
- Réduire l'utilisation de la bande passante du réseau
- Effectuer des modifications atomiques sur des éléments de ressources spécifiques
- Minimisez le risque de remplacer des modifications simultanées
- Mettre à jour les ressources dans le cadre des flux de travail par lots et par transactions

Formats PATCH pris en charge

AWS HealthLake prend en charge deux formats PATCH standard :

Correctif JSON (RFC 6902)

Utilise la syntaxe du pointeur JSON pour cibler les éléments en fonction de leur position dans la structure des ressources.

Type de contenu : `application/json-patch+json`

FHIRPath Patch (spécification FHIR R4)

Utilise des FHIRPath expressions pour cibler les éléments en fonction de leur contenu et de leurs relations, offrant ainsi une approche native du FHIR pour l'application de correctifs.

Type de contenu : `application/fhir+json`

Usage

Opérations PATCH directes

L'opération PATCH peut être invoquée directement sur les ressources FHIR à l'aide de la méthode HTTP PATCH :

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

PATCH en packs

Les opérations PATCH peuvent être incluses sous forme d'entrées dans des ensembles FHIR de type `batch` ou`transaction`, ce qui vous permet de combiner des opérations de correctif avec d'autres interactions FHIR (création, lecture, mise à jour, suppression) dans une seule demande.

- Packs de transactions : toutes les entrées réussissent ou échouent de manière atomique
- Batch bundles : chaque entrée est traitée indépendamment

Format de correctif JSON

Opérations prises en charge

Opération	Description
`add`	Ajouter une nouvelle valeur à la ressource
`remove`	Supprimer une valeur de la ressource
`replace`	Remplacer une valeur existante dans la ressource
`move`	Supprimer une valeur d'un emplacement et l'ajouter à un autre

Opération	Description
`copy`	Copier une valeur d'un emplacement à un autre
`test`	Vérifiez qu'une valeur à l'emplacement cible est égale à une valeur spécifiée

Syntaxe du chemin

Le patch JSON utilise la syntaxe du pointeur JSON (RFC 6901) :

Exemple de chemin	Description
`/name/0/family`	Élément de famille du prénom
`/telecom/-`	Ajouter au réseau de télécommunications
`/active`	Élément actif au niveau de la racine
`/address/0/line/1`	Deuxième ligne de la première adresse

Exemples

Demande de correctif JSON directe avec plusieurs opérations

```
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"
  }
]
```

Demande de correctif JSON directe avec une seule opération

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

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

Patch JSON dans le bundle

Utilisez une ressource binaire contenant la charge utile du patch JSON codé en base64 :

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

FHIRPath Format du correctif

Opérations prises en charge

Opération	Description
`add`	Ajouter un nouvel élément à une ressource
`insert`	Insérer un élément à un endroit précis dans une liste
`delete`	Supprimer un élément d'une ressource
`replace`	Remplacer la valeur d'un élément existant
`move`	Réorganiser les éléments d'une liste

Syntaxe du chemin

FHIRPath Le correctif utilise des FHIRPath expressions prenant en charge :

- Accès basé sur un index : `Patient.name[0]`
- Filtrer avec **`where()`** : `Patient.name.where(use = 'official')`

- Logique booléenne : `Patient.telecom.where(system = 'phone' and use = 'work')`
- Fonctions de sous-définition :`first()`, `last()`
- Contrôles d'existence :`exists()`, `count()`
- Navigation polymorphe : `0bservation.value`

Exemples

Demande de FHIRPath correctif directe

```
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" }
    ]
  }]
}
```

FHIRPath Patch dans le bundle

Utilisez une ressource Parameters comme ressource d'entrée avec `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"
    }
  }]
}
```

En-têtes de demande

En-tête	Obligatoire	Description
`Content-Type`	Oui	`application/json-patch+json` pour JSON Patch ou `application/fhir+json` pour FHIRPath Patch
`If-Match`	Non	Mise à jour conditionnelle spécifique à la version à l'aide de ETag

Exemple de réponse

L'opération renvoie la ressource mise à jour avec les nouvelles informations de version :

```
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"
  }
}
```

Comportement

L'opération PATCH :

- Valide la syntaxe du correctif conformément à la spécification appropriée (RFC 6902 pour le correctif JSON, FHIR R4 pour le correctif) FHIRPath
- Applique les opérations de manière atomique : toutes les opérations réussissent ou échouent
- Met à jour l'ID de version de la ressource et crée une nouvelle entrée d'historique
- Préserve la ressource d'origine dans l'historique avant d'appliquer les modifications
- Valide les contraintes de ressources FHIR après l'application des correctifs
- Supporte les mises à jour conditionnelles à l'aide de l'en-tête If-Match avec ETag

Gestion des erreurs

L'opération gère les conditions d'erreur suivantes :

- 400 Mauvaise demande : syntaxe de correctif non valide (demande non conforme ou document de correctif mal formé)
- 404 Introuvable : ressource introuvable (l'ID spécifié n'existe pas)
- 409 Conflit : conflit de version (mises à jour simultanées ou identifiant de version non actuel fourni)
- 422 Entité non traitable : les opérations de correctif ne peuvent pas être appliquées aux éléments de ressource spécifiés

Résumé des capacités

Capacité	Correctif JSON	FHIRPath Patch
Type de contenu	`application/json-patch+json`	`application/fhir+json`
Format du chemin	Pointeur JSON (RFC 6901)	FHIRPath expressions
API PATCH directe	Pris en charge	Pris en charge
Bundle Batch	Pris en charge (via le binaire)	Pris en charge (via les paramètres)
Transaction groupée	Pris en charge (via le binaire)	Pris en charge (via les paramètres)
Opérations	ajouter, supprimer, remplacer, déplacer, copier, tester	ajouter, insérer, supprimer, remplacer, déplacer

Limitations

- Les opérations PATCH conditionnelles utilisant des conditions de recherche ne sont pas prises en charge
- Les patchs JSON intégrés aux bundles doivent utiliser des ressources binaires avec un contenu codé en base64
- FHIRPath Les packs de patchs doivent utiliser les ressources de paramètres

Ressources supplémentaires

Pour plus d'informations sur les opérations PATCH, voir :

- Documentation du PATCH FHIR R4
- Spécification du patch FHIR R4 FHIRPath
- RFC 6902 - Correctif JSON
- RFC 6901 - Pointeur JSON

Regroupement des ressources FHIR

Un FHIR `Bundle` est un conteneur contenant une collection de ressources FHIR. AWS HealthLake AWS HealthLake prend en charge deux types de bundles avec des comportements de traitement différents.

`Batch`les bundles traitent chaque ressource indépendamment. Si l'une des ressources échoue, les ressources restantes peuvent toujours réussir. Chaque opération est traitée individuellement et le traitement se poursuit même en cas d'échec de certaines opérations. Utilisez des lots pour les opérations groupées où un succès partiel est acceptable, comme le téléchargement de plusieurs dossiers de patients indépendants.

`Transaction`les bundles traitent toutes les ressources de manière atomique comme une seule unité. Soit toutes les opérations sur les ressources réussissent, soit aucune d' AWS HealthLake entre elles n'est validée. Utilisez des ensembles de transactions lorsque vous avez besoin de garantir l'intégrité référentielle des ressources connexes, par exemple pour créer un patient présentant des observations et des affections connexes où toutes les données doivent être enregistrées ensemble.

Différences entre les lots et les ensembles de transactions

Fonctionnalité	Par lots	Transaction
Modèle de traitement	Chaque opération réussit ou échoue indépendamment.	Toutes les opérations réussissent ou échouent en tant qu'unité atomique unique.
Gestion des défaillances	Le traitement se poursuit même en cas d'échec des opérations individuelles.	L'ensemble échoue en cas d'échec d'une seule opération.
Ordre d'exécution	L'ordre d'exécution n'est pas garanti.	Les opérations sont traitées dans l'ordre indiqué.
Intégrité référentielle	Non appliqué dans toutes les opérations.	Appliqué pour les ressources référencées localement au sein du bundle.

Fonctionnalité	Par lots	Transaction
Utiliser en priorité pour	Opérations groupées pour lesquelles un succès partiel est acceptable.	Ressources connexes qui doivent être créées ou mises à jour ensemble.

Vous pouvez regrouper des ressources FHIR de types identiques ou différents, et elles peuvent inclure une combinaison d'opérations FHIR, telles que`create`,, `read` `update``delete`, et. `patch` Pour plus d'informations, consultez le pack de ressources dans la documentation du FHIR R4.

Voici des exemples de cas d'utilisation pour chaque type de bundle.

Packs Batch

- Téléchargez plusieurs dossiers de patients indépendants provenant de différents établissements lors de la synchronisation nocturne des données.
- Téléchargez en masse l'historique des médicaments lorsque certains dossiers peuvent présenter des problèmes de validation.
- Chargez des données de référence, telles que les organisations et les professionnels, où les défaillances individuelles n'affectent pas les autres entrées.

Packs de transactions

- Créez un patient présentant des observations et des affections connexes lors d'une admission au service des urgences, où toutes les données doivent être enregistrées ensemble.
- Mettez à jour la liste des médicaments d'un patient et les informations connexes sur les allergies qui doivent rester cohérentes.
- Enregistrez une rencontre complète avec le patient, les observations, les procédures et les informations de facturation dans une seule unité atomique.

> **⚠ Important**
>
> Les ensembles de lots et de transactions utilisent la même structure de `Bundle` ressources. La seule différence réside dans la valeur du `type` champ.

L'exemple suivant montre un ensemble de transactions comprenant plusieurs types de ressources et opérations.

```
{
  "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"
      }
    }
  ]
}
```

Regrouper les ressources du FHIR en tant qu'entités indépendantes

Regrouper les ressources FHIR en tant qu'entités indépendantes

1. Collectez HealthLake `region` et `datastoreId` valorisez. Pour de plus amples informations, veuillez consulter Obtenir les propriétés du magasin de données.
2. Créez une URL pour la demande en utilisant les valeurs collectées pour HealthLake `region` et`datastoreId`. Ne spécifiez pas de type de ressource FHIR dans l'URL. Pour afficher le chemin complet de l'URL dans l'exemple suivant, faites défiler le curseur sur le bouton Copier.

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

3. Construisez un corps JSON pour la demande, en spécifiant chaque verbe HTTP comme faisant partie des `method` éléments. L'exemple suivant utilise une interaction `batch` de type avec la `Bundle` ressource pour créer de nouvelles `Medication` ressources `Patient` et. Toutes les sections requises sont commentées en conséquence. Dans le cadre de cette procédure, enregistrez le fichier sous`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. Envoyez la demande . Le type de `Bundle` lot FHIR utilise une `POST` demande avec AWS signature version 4 ou SMART sur autorisation FHIR. L'exemple de code suivant utilise l'outil de ligne de `curl` commande à des fins de démonstration.

   SigV4

   Autorisation SigV4

   ```
   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

Exemple d'autorisation SMART sur FHIR pour le type de `IdentityProviderConfiguration`données.

```
{
    "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\"]}"
}
```

L'appelant peut attribuer des autorisations dans le lambda d'autorisation. Pour de plus amples informations, veuillez consulter Étendue OAuth 2.0.

Le serveur renvoie une réponse indiquant les `Medication` ressources `Patient` et créées à la suite de la demande de type de `Bundle` lot.

Conditionnel PUTs en lots

AWS HealthLake prend en charge les mises à jour conditionnelles au sein des bundles à l'aide des paramètres de requête suivants :

> **Note**
>
> PUTs Les options conditionnelles ne sont prises en charge que dans `batch` les offres groupées. `Transaction`les offres groupées ne sont pas compatibles avec le conditionnel PUTs.

- `_id`(autonome)
- `_id`en combinaison avec l'un des éléments suivants :
  - `_tag`
  - `_createdAt`
  - `_lastUpdated`

Lorsque vous utilisez le conditionnel PUTs dans les bundles, vous AWS HealthLake évaluez les paramètres de requête par rapport aux ressources existantes et prenez des mesures en fonction des résultats du match.

Comportement des mises à jour

Scénario	Statut HTTP	Mesures prises
Ressource sans identifiant fourni	201 créés	Crée toujours une nouvelle ressource.
Ressource avec un nouvel identifiant (aucune correspondance)	201 créés	Crée une nouvelle ressource avec l'ID spécifié.
Ressource avec identifiant existant (correspondance unique)	200 OK	Met à jour la ressource correspondante.
Ressource avec identifiant existant (conflit détecté)	409 – Conflit	Renvoie une erreur. Aucune modification n'est apportée.
Ressource avec ID existant (ID non concordant)	400 Requête erronée	Renvoie une erreur. Aucune modification n'est apportée.
Plusieurs ressources correspondent aux conditions	412 – Échec de condition préalable	Renvoie une erreur. Aucune modification n'est apportée.

Dans l'exemple de bundle suivant avec une mise à jour conditionnelle, la `Patient` ressource avec un identifiant FHIR est mise 476 à jour uniquement si la condition `_lastUpdated=lt2025-04-20` est remplie.

```
{
    "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"
            }
        }
    ]
}
```

Regroupement des ressources FHIR en une seule entité

Regrouper les ressources FHIR en une seule entité

1. Collectez HealthLake `region` et `datastoreId` valorisez. Pour de plus amples informations, veuillez consulter Obtenir les propriétés du magasin de données.
2. Créez une URL pour la demande en utilisant les valeurs collectées pour HealthLake `region` et`datastoreId`. Incluez le type `Bundle` de ressource FHIR dans l'URL. Pour afficher le chemin complet de l'URL dans l'exemple suivant, faites défiler le curseur sur le bouton Copier.

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

3. Construisez un corps JSON pour la demande, en spécifiant les ressources FHIR à regrouper. L'exemple suivant regroupe deux `Patient` ressources dans HealthLake. Dans le cadre de cette procédure, enregistrez le fichier sous`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. Envoyez la demande . Le type de `Bundle` document FHIR utilise une `POST` demande avec le protocole de AWS signature Signature Version 4. L'exemple de code suivant utilise l'outil de ligne de `curl` commande à des fins de démonstration.

   SigV4

   Autorisation SigV4

   ```
   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

   Exemple d'autorisation SMART sur FHIR pour le type de `IdentityProviderConfiguration`données.

   ```
   {
       "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\"]}"
}
```

L'appelant peut attribuer des autorisations dans le lambda d'autorisation. Pour de plus amples informations, veuillez consulter Étendue OAuth 2.0.

Le serveur renvoie une réponse indiquant deux `Patient` ressources créées à la suite de la demande de type de `Bundle` document.

Configuration du niveau de validation pour les bundles

Lorsque vous regroupez des ressources FHIR, vous pouvez éventuellement spécifier un en-tête `x-amzn-healthlake-fhir-validation-level` HTTP pour configurer un niveau de validation pour la ressource. Ce niveau de validation sera défini pour toutes les demandes de création et de mise à jour du bundle. AWS HealthLake prend actuellement en charge les niveaux de validation suivants :

- `strict`: Les ressources sont validées en fonction de l'élément de profil de la ressource ou de la spécification R4 si aucun profil n'est présent. Il s'agit du niveau de validation par défaut pour AWS HealthLake.
- `structure-only`: Les ressources sont validées par rapport à R4, en ignorant les profils référencés.
- `minimal`: Les ressources sont validées de manière minimale, sans tenir compte de certaines règles R4. Les ressources qui échouent aux vérifications de structure requises search/analytics seront mises à jour pour inclure un avertissement d'audit.

Les ressources groupées avec le niveau de validation minimal peuvent être ingérées dans une banque de données malgré l'échec de la validation requise pour l'indexation des recherches. Dans ce cas, les ressources seront mises à jour pour inclure une extension spécifique à Healthlake afin de documenter ces échecs, et les entrées de la réponse du bundle incluront les OperationOutcome ressources suivantes :

```
{
    "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."
                        }
                    ]
                }
            }
        }
    ]
}
```

En outre, l'en-tête de réponse HTTP suivant sera inclus avec la valeur « true » :

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

> **Note**
>
> Notez que les données ingérées qui sont mal formées conformément à la spécification R4 peuvent ne pas être consultables comme prévu si ces erreurs sont présentes.

Support limité pour le type « message » de type Bundle

HealthLake fournit un support limité pour le type de bundle FHIR `message` via un processus de conversion interne. Cette prise en charge est conçue pour les scénarios dans lesquels les ensembles

de messages ne peuvent pas être reformatés à la source, tels que l'ingestion de flux ADT (admission, sortie, transfert) provenant des anciens systèmes hospitaliers.

> **Warning**
>
> Cette fonctionnalité nécessite une liste explicite des AWS comptes autorisés et n'applique pas la sémantique des messages FHIR R4 ni l'intégrité référentielle. Contactez AWS le Support pour demander l'activation de votre compte avant d'utiliser des ensembles de messages.

Principales différences par rapport au traitement standard des messages

- Ensembles de messages (spécification FHIR) : La première entrée doit être une `MessageHeader` qui fait référence à d'autres ressources. Les ressources ne disposent pas `request` d'objets individuels, et l' MessageHeader événement détermine les actions de traitement.
- HealthLake Traitement : convertit les ensembles de messages en ensembles par lots en affectant automatiquement des opérations PUT à chaque entrée de ressource. Les ressources sont traitées indépendamment sans appliquer la sémantique des messages ou l'intégrité référentielle.

Limitations importantes

- FHIR R4 Les règles de traitement spécifiques aux messages ne sont pas appliquées
- Aucune intégrité transactionnelle entre les ressources
- Les références inter-ressources ne sont pas validées
- Nécessite une liste explicite des comptes autorisés

Exemple de structure de bundle de messages

```
{
              "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**
>
> Chaque ressource est stockée indépendamment, comme si elle était soumise via des opérations PUT individuelles. Si une sémantique complète de la messagerie FHIR ou une validation de l'intégrité référentielle sont requises, prétraitez les ensembles de messages ou implémentez une validation au niveau de l'application avant de les soumettre.

Transactions groupées asynchrones

AWS HealthLake prend en charge le `Bundle` type asynchrone `transaction` qui vous permet de soumettre des transactions avec un maximum de 500 ressources. Lorsque vous soumettez une transaction asynchrone, mettez-la en HealthLake file d'attente pour traitement et renvoyez immédiatement une URL d'interrogation. Vous pouvez utiliser cette URL pour vérifier le statut et récupérer la réponse. Cela suit le modèle de bundle asynchrone FHIR.

Quand utiliser les transactions asynchrones

- Vous devez soumettre plus de 100 ressources (limite synchrone) en une seule transaction.
- Vous voulez éviter de bloquer votre application en attendant la fin du traitement de la transaction.
- Vous devez traiter de gros volumes de ressources connexes avec un meilleur débit.

> **Important**
>
> Les résultats du sondage sont disponibles pendant 90 jours après la fin de la transaction. Après cette période de 90 jours, l'URL du sondage ne renvoie plus de résultats. Concevez votre intégration pour récupérer et stocker les résultats dans cette fenêtre.

> **Note**
>
> `transaction`Le `Bundle` type synchrone continue de prendre en charge jusqu'à 100 ressources et constitue le mode de traitement par défaut. Si vous soumettez un `Bundle` type `transaction` contenant plus de 100 ressources sans `Prefer: respond-async` en-tête, HealthLake renvoie une `422 Unprocessable Entity` erreur. Les ensembles avec type ne `batch` sont pas pris en charge pour le traitement asynchrone : seul le `Bundle` type `transaction` peut être soumis de manière asynchrone (avec un maximum de 500 opérations).

> **Note**
>
> `PATCH`les opérations et les conditionnelles ne PUTs sont pas prises en charge dans les transactions groupées asynchrones.

Soumission d'une transaction asynchrone

Pour soumettre une transaction asynchrone, envoyez une `POST` demande au point de terminaison du magasin de données avec l'`Prefer: respond-async`en-tête. Le bundle doit avoir un type`transaction`. Les ensembles avec type ne `batch` sont pas pris en charge pour le traitement asynchrone des ensembles.

HealthLake effectue les validations initiales du bundle au moment de la soumission. Si la validation aboutit, HealthLake renvoie HTTP 202 Accepted avec un en-tête de `content-location` réponse contenant l'URL d'interrogation.

Pour soumettre un type asynchrone **`Bundle transaction`**

1. Envoyez une `POST` demande au point de terminaison du magasin de HealthLake données.

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

2. Construisez un corps JSON pour la demande avec le type de bundle`transaction`. Dans le cadre de cette procédure, enregistrez le fichier sous`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. Envoyez la demande avec l'`Prefer: respond-async`en-tête. Le type de `Bundle` transaction FHIR utilise une `POST` demande avec AWS signature version 4 ou une autorisation SMART on FHIR. L'exemple de code suivant utilise l'outil de ligne de `curl` commande à des fins de démonstration.

   SigV4

   Autorisation SigV4

   ```
   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

   Exemple d'autorisation SMART sur FHIR pour le type de `IdentityProviderConfiguration`données.

   ```
   {
       "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\"]}"
}
```

L'appelant peut attribuer des autorisations dans le lambda d'autorisation. Pour de plus amples informations, veuillez consulter Étendue OAuth 2.0.

4. En cas de soumission réussie, le serveur renvoie HTTP 202 Accepted. L'en-tête de `content-location` réponse contient l'URL du sondage. L'organisme de réponse est une `OperationOutcome` ressource.

```
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"
            ]
        }
    ]
}
```

Sondage de l'état de la transaction

Après avoir soumis une transaction asynchrone, utilisez l'URL de sondage figurant dans l'en-tête de `content-location` réponse pour vérifier le statut de la transaction. Envoyez une `GET` demande à l'URL du sondage.

> **Note**
>
> Pour les magasins de données compatibles SMART sur FHIR, le jeton d'autorisation doit inclure `read` des autorisations sur le type de `Transaction` ressource pour demander l'état de la transaction. Pour plus d'informations sur SMART sur les oscilloscopes FHIR, consultez. SMART sur les oscilloscopes FHIR OAuth 2.0 pris en charge par HealthLake

Envoyez une `GET` demande à l'URL du sondage. L'exemple suivant utilise l'outil de ligne de `curl` commande.

SigV4

Autorisation SigV4

```
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

Autorisation SMART sur FHIR. Le jeton d'autorisation doit inclure `read` des autorisations sur le type de `Transaction` ressource.

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

Le tableau suivant décrit les réponses possibles.

Codes de réponse aux sondages

Statut HTTP	Signification	Corps de la réponse
202 Accepté	La transaction est en file d'attente	`OperationOutcome` avec diagnostic « SOUMIS »
202 Accepté	La transaction est en cours de traitement	`OperationOutcome` avec diagnostic « IN_PROGRESS »
200 OK	Transaction terminée avec succès	`Bundle`avec type `transaction-response`
4xx/5xx	Échec de la transaction	`OperationOutcome` avec les détails de l'erreur

Les exemples suivants montrent chaque type de réponse.

Transaction en file d'attente (202)

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

Traitement des transactions (202)

```
{
    "resourceType": "OperationOutcome",
    "id": "transactionId",
```

```
    "issue": [
        {
            "severity": "information",
            "code": "informational",
            "diagnostics": "IN_PROGRESS"
        }
    ]
}
```

Transaction terminée (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"
            }
        }
    ]
}
```

Échec de la transaction (4xx/5xx)

```
{
    "resourceType": "OperationOutcome",
    "issue": [
        {
            "severity": "error",
```

```
            "code": "exception",
            "diagnostics": "Transaction failed: conflict detected on resource Patient/
example-id"
        }
    ]
}
```

Commande en cours

Les ensembles de type asynchrone sont mis en file d'attente mais ne `transaction` sont pas traités dans un ordre de soumission strict. HealthLake optimise le traitement en fonction de la capacité disponible et de la charge du système.

> **Important**
>
> Ne vous fiez pas au traitement des transactions dans l'ordre dans lequel elles ont été soumises. Par exemple, si vous soumettez la transaction A à 10 h 00 et la transaction B à 10 h 01, la transaction B peut être terminée avant la transaction A. Concevez votre application pour :
>
> - Terminez out-of-order la manipulation.
> - Utilisez l'URL du sondage pour suivre chaque transaction indépendamment.
> - Mettez en œuvre le séquençage au niveau de l'application si la commande est importante pour votre cas d'utilisation.

Quotas et régulation

Les quotas et limites de taux suivants s'appliquent aux transactions asynchrones.

Quotas de transactions asynchrones

Quota	Value	Ajustable
Nombre maximum d'opérations par transaction asynchrone	500	Non
Nombre maximum de transactions en attente par magasin de données	500	Oui

- Les transactions asynchrones partagent les mêmes limites de débit d'API définies ci-dessous. Quotas de service
- L'interrogation du statut des transactions partage les mêmes limites de débit d'API que les opérations read (`GET`) sur les ressources FHIR.
- Si la limite de transactions en attente est atteinte, les soumissions suivantes renvoient une erreur jusqu'à ce que les transactions existantes soient terminées.

Gestion des erreurs

Pour un bundle « transactionnel », toutes les ressources FHIR contenues dans le bundle sont traitées comme une opération atomique. Toutes les ressources de l'opération doivent réussir, sinon aucune opération du bundle n'est traitée.

Les erreurs se répartissent en deux catégories : les erreurs de soumission HealthLake renvoyées de manière synchrone et les erreurs de traitement que vous détectez par le biais d'un sondage.

Erreurs de soumission

HealthLake valide le bundle au moment de l'envoi et renvoie les erreurs de manière synchrone avant que la transaction ne soit mise en file d'attente. Les erreurs de soumission incluent des erreurs de validation des ressources FHIR non valides, des types de ressources non pris en charge, le dépassement de la limite de 500 opérations et l'utilisation de l'`Prefer: respond-async`en-tête avec des ensembles de lots. Si la limite de transactions en attente pour le magasin de données a été atteinte, HealthLake renvoie `ThrottlingException` a. Lorsqu'une erreur de soumission se produit, la transaction ne sera pas mise en file d'attente.

Erreurs de traitement

Des erreurs de traitement se produisent une fois que la transaction a été mise en file d'attente et sont renvoyées via l'URL d'interrogation. Il s'agit notamment des conflits de transactions, lorsqu'une autre opération a modifié une ressource faisant partie de la transaction, et des erreurs de serveur pendant le traitement. Lorsqu'une erreur de traitement se produit, aucune mutation de ressource n'est effectuée pour les ressources de la transaction. L'URL du sondage renverra un `OperationOutcome` contenant les détails de l'erreur.

Supprimer une ressource FHIR

L'`delete`interaction FHIR supprime une ressource FHIR existante d'un magasin de HealthLake données. Pour plus d'informations, consultez la `delete`documentation de l' RESTful API FHIR R4.

Pour supprimer une ressource FHIR

1. Collectez HealthLake `region` et `datastoreId` valorisez. Pour de plus amples informations, veuillez consulter Obtenir les propriétés du magasin de données.
2. Déterminez le type de FHIR `Resource` à supprimer et collectez la `id` valeur associée. Pour de plus amples informations, veuillez consulter Types de ressources.
3. Construisez une URL pour la demande en utilisant les valeurs collectées pour HealthLake `region` et`datastoreId`. Incluez également le `Resource` type FHIR et son associé`id`. Pour afficher le chemin complet de l'URL dans l'exemple suivant, faites défiler le curseur sur le bouton Copier.

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

4. Envoyez la demande . L'`delete`interaction FHIR utilise une `DELETE` demande avec AWS signature version 4 ou SMART sur autorisation FHIR. L'`curl`exemple suivant supprime une `Patient` ressource FHIR existante d'un magasin de HealthLake données. Pour afficher l'exemple dans son intégralité, faites défiler la souris sur le bouton Copier.

   SigV4

   Autorisation SigV4

   ```
   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'
   ```

   Le serveur renvoie un code d'état `204` HTTP confirmant que la ressource a été supprimée du magasin de HealthLake données. Si une demande de suppression échoue, vous recevrez un code d'état HTTP en `400` série indiquant pourquoi la demande a échoué.

SMART on FHIR

Exemple d'autorisation SMART sur FHIR pour le type de `IdentityProviderConfiguration`données.

```
{
    "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\"]}"
}
```

L'appelant peut attribuer des autorisations dans le lambda d'autorisation. Pour de plus amples informations, veuillez consulter Étendue OAuth 2.0.

AWS Console

1. Connectez-vous à la page Exécuter une requête sur la HealthLake console.

2. Dans la section Paramètres de requête, effectuez les sélections suivantes.

- ID du magasin de données : choisissez un identifiant de magasin de données pour générer une chaîne de requête.
- Type de requête : choisissez`Delete`.
- Type de ressource : choisissez le type de ressource FHIR à supprimer.
- ID de ressource — entrez l'ID de ressource FHIR.

3. Choisissez Exécuter la requête.

Suppression de ressources FHIR en fonction des conditions

La suppression conditionnelle est particulièrement utile lorsque vous ne connaissez pas l'ID de ressource FHIR spécifique mais que vous disposez d'autres informations d'identification concernant la ressource que vous souhaitez supprimer.

La suppression conditionnelle vous permet de supprimer une ressource existante en fonction de critères de recherche plutôt que d'un identifiant FHIR logique. Lorsque le serveur traite la demande de suppression, il effectue une recherche à l'aide des fonctionnalités de recherche standard pour le type de ressource afin de résoudre un identifiant logique unique pour la demande.

Fonctionnement de la suppression conditionnelle

L'action du serveur dépend du nombre de correspondances qu'il trouve :

1. Aucune correspondance : le serveur tente une suppression ordinaire et répond de manière appropriée (404 introuvable pour une ressource inexistante, 204 Aucun contenu pour une ressource déjà supprimée)
2. Une correspondance : le serveur effectue une suppression ordinaire sur la ressource correspondante
3. Correspondances multiples : renvoie une erreur 412 Precondition Failed indiquant que les critères du client n'étaient pas suffisamment sélectifs

Scénarios de réponse

AWS HealthLake gère les opérations de suppression conditionnelle avec les modèles de réponse suivants :

Opérations réussies

- Lorsque vos critères de recherche identifient avec succès une seule ressource active, le système renvoie 204 No Content une fois la suppression terminée, comme pour les opérations de suppression standard.

Suppression conditionnelle basée sur un identifiant

Lorsque vous effectuez une suppression conditionnelle basée sur `id` des paramètres supplémentaires (`createdAt`,`tag`, ou`_lastUpdated`) :

- 204 Aucun contenu : la ressource a déjà été supprimée
- 404 Introuvable : la ressource n'existe pas
- 409 Conflit : l'identifiant correspond mais les autres paramètres ne correspondent pas

Non-ID-Based Suppression conditionnelle

Quand n'`id`est pas fourni ou lorsque vous utilisez des paramètres autres que `createdAttag`, ou `_lastUpdated` :

- 404 Introuvable : Aucune correspondance n'a été trouvée

Situations de conflit

Plusieurs scénarios entraînent l'échec de 412 réponses à la condition préalable :

- Plusieurs ressources correspondent à vos critères de recherche (critères trop peu spécifiques)
- Conflits de version lors de l'utilisation d' ETag en-têtes avec `If-Match`
- Mises à jour des ressources survenant entre les opérations de recherche et de suppression

Exemple de suppression conditionnelle réussie

L'exemple suivant supprime une ressource Patient en fonction de critères spécifiques :

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

Cette demande supprime une ressource Patient dans laquelle :

- Le nom est « Peter »
- La date de naissance est le 1er janvier 2000
- Le numéro de téléphone est 1234567890

Bonnes pratiques

1. Utilisez des critères de recherche spécifiques pour éviter les correspondances multiples et éviter 412 erreurs.
2. Envisagez ETag des en-têtes pour le contrôle de version lorsque cela est nécessaire pour gérer des modifications simultanées.
3. Gérez les réponses aux erreurs de manière appropriée :
   - Pour 404 : Affinez vos critères de recherche
   - Pour 4.1.2 : préciser les critères ou résoudre les conflits de versions
4. Préparez-vous à des conflits temporels dans des environnements où la concurrence est élevée, dans lesquels les ressources peuvent être modifiées entre les opérations de recherche et de suppression.

Idempotencie et simultanéité

Clés d'impuissance

AWS HealthLake prend en charge les clés d'idempuissance pour les `POST` opérations FHIR, fournissant un mécanisme robuste pour garantir l'intégrité des données lors de la création des ressources. En incluant un UUID unique comme clé d'idempuissance dans l'en-tête de demande, les applications de santé peuvent garantir que chaque ressource FHIR est créée exactement une fois, même dans les scénarios impliquant une instabilité du réseau ou des tentatives automatiques.

Cette fonctionnalité est particulièrement cruciale pour les systèmes de santé où les doublons de dossiers médicaux peuvent avoir de graves conséquences. Lorsqu'une demande est reçue avec la même clé d'idempuissance qu'une demande précédente, elle HealthLake renvoie la ressource d'origine au lieu de créer un doublon. Par exemple, cela peut se produire lors d'une boucle de nouvelles tentatives ou en raison de pipelines de requêtes redondants. L'utilisation de la clé d'idempotencie permet HealthLake de maintenir la cohérence des données tout en offrant une expérience fluide aux applications clientes traitant des problèmes de connectivité intermittents.

Mise en œuvre

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

```
}
```

Scénarios de réponse

Première demande (201 créés)

- Nouvelle ressource créée avec succès
- La réponse inclut l'ID de ressource

Demande dupliquée (conflit 409)

- Même clé d'impuissance détectée
- Ressource d'origine renvoyée
- Aucune nouvelle ressource n'a été créée

Demande non valide (400 demandes erronées)

- UUID mal formé
- Champs obligatoires manquants

Bonnes pratiques

- Générez un UUID unique pour chaque nouvelle création de ressource
- Stockez les clés d'impuissance pour la logique des nouvelles tentatives
- Utiliser un format de clé cohérent : UUID v4 recommandé
- Implémenter dans les applications clientes gérant la création de ressources

> **Note**
>
> Cette fonctionnalité est particulièrement utile pour les systèmes de santé qui exigent une précision stricte des données et évitent la duplication des dossiers médicaux.

ETag dans AWS HealthLake

AWS HealthLake utilise ETags pour un contrôle de simultanéité optimiste des ressources FHIR, fournissant un mécanisme fiable pour gérer les modifications simultanées et maintenir la cohérence des données. An ETag est un identifiant unique qui représente une version spécifique d'une ressource, fonctionnant comme un système de contrôle de version via des en-têtes HTTP. Lors

de la lecture ou de la modification des ressources, les applications peuvent les utiliser ETags pour empêcher les remplacements involontaires et garantir l'intégrité des données, en particulier dans les scénarios impliquant des mises à jour simultanées potentielles.

Exemple de mise en œuvre

```
// 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
```

Scénarios de réponse

Opération réussie (200 OK ou 204 sans contenu)

- ETag correspond à la version actuelle
- L'opération se déroule comme prévu

Conflit de version (échec de la condition préalable 412)

- ETag ne correspond pas à la version actuelle
- Mise à jour rejetée pour éviter la perte de données

Bonnes pratiques

- Inclure ETags dans toutes les opérations de mise à jour et de suppression
- Implémenter une logique de nouvelle tentative pour gérer les conflits de versions
- Utilisez If-None-Match :* pour les create-if-not-exists scénarios
- Vérifiez toujours la ETag fraîcheur avant de procéder à des modifications

Ce système de contrôle de simultanéité est essentiel pour maintenir l'intégrité des données de santé, en particulier dans les environnements où plusieurs utilisateurs ou systèmes accèdent aux mêmes ressources et les modifient.

Recherche de ressources FHIR dans AWS HealthLake

Utilisez l'`search`interaction FHIR pour rechercher un ensemble de ressources FHIR dans un magasin de HealthLake données en fonction de certains critères de filtrage. L'`search`interaction peut être effectuée à l'aide d'une `POST` demande `GET` ou. Pour les recherches impliquant des informations personnelles identifiables (PII) ou des informations de santé protégées (PHI), il est recommandé d'utiliser des `POST` demandes, car les informations personnelles et les PHI sont ajoutées dans le corps de la demande et sont cryptées pendant le transfert.

> **Note**
>
> L'`search`interaction FHIR décrite dans ce chapitre est construite conformément à la norme HL7 FHIR R4 pour l'échange de données sur les soins de santé. Comme il s'agit d'une représentation d'un service HL7 FHIR, il n'est pas proposé par le biais de AWS CLI et AWS SDKs. Pour plus d'informations, consultez la `search`documentation de l' RESTful API FHIR R4.

HealthLake prend en charge un sous-ensemble de paramètres de recherche FHIR R4. Pour de plus amples informations, veuillez consulter Paramètres de recherche FHIR R4 pour HealthLake.

Rubriques

- Recherche de ressources FHIR avec GET
- Recherche de ressources FHIR avec POST
- Niveaux de cohérence des recherches FHIR

Recherche de ressources FHIR avec GET

Vous pouvez utiliser des `GET` requêtes pour effectuer une recherche dans un magasin de HealthLake données. Lors de l'utilisation`GET`, HealthLake prend en charge la fourniture de paramètres de recherche dans le cadre de l'URL, mais pas dans le corps d'une requête. Pour de plus amples informations, veuillez consulter Paramètres de recherche FHIR R4 pour HealthLake.

> **Important**
>
> Pour les recherches impliquant des informations personnelles identifiables (PII) ou des informations de santé protégées (PHI), les meilleures pratiques en matière de sécurité nécessitent l'utilisation de `POST` demandes, car les informations personnelles et les PHI sont ajoutées dans le corps de la demande et sont cryptées pendant le transfert. Pour de plus amples informations, veuillez consulter Recherche de ressources FHIR avec POST.

La procédure suivante est suivie d'exemples utilisés `GET` pour effectuer des recherches dans un magasin de HealthLake données.

Pour effectuer une recherche dans un magasin de HealthLake données avec **`GET`**

1. Collectez HealthLake `region` et `datastoreId` valorisez. Pour de plus amples informations, veuillez consulter Obtenir les propriétés du magasin de données.
2. Déterminez le type de ressource FHIR à rechercher et collectez la `id` valeur associée. Pour de plus amples informations, veuillez consulter Types de ressources.
3. Construisez une URL pour la demande en utilisant les valeurs collectées pour HealthLake `region` et`datastoreId`. Incluez également le `Resource` type FHIR et les paramètres de recherche pris en charge. Pour afficher le chemin complet de l'URL dans l'exemple suivant, faites défiler le curseur sur le bouton Copier.

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

4. Envoyez la `GET` demande avec AWS Signature Version 4 ou SMART sur autorisation FHIR. L'`curl`exemple suivant renvoie le nombre total de `Patient` ressources dans un magasin de HealthLake données. Pour afficher l'exemple dans son intégralité, faites défiler la souris sur le bouton Copier.

   SigV4

   Autorisation SigV4

   ```
   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

Exemple d'autorisation SMART sur FHIR pour le type de `IdentityProviderConfiguration`données.

```
{
    "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\"]}"
}
```

L'appelant peut attribuer des autorisations dans le lambda d'autorisation. Pour de plus amples informations, veuillez consulter Étendue OAuth 2.0.

AWS Console

> **Note**
>
> La HealthLake console prend uniquement en charge l'autorisation SigV4. L'autorisation SMART on FHIR est prise en charge via AWS CLI et AWS SDKs.

1. Connectez-vous à la page Exécuter une requête sur la HealthLake console.

2. Dans la section Paramètres de requête, effectuez les sélections suivantes.

- ID du magasin de données : choisissez un identifiant de magasin de données pour générer une chaîne de requête.
- Type de requête : choisissez`Search with GET`.
- Type de ressource : choisissez le type de ressource FHIR sur lequel effectuer la recherche.
- Paramètres de recherche : sélectionnez un paramètre de recherche ou une combinaison de paramètres de recherche pour concentrer votre recherche sur des enregistrements spécifiques.

3. Choisissez Exécuter la requête.

Exemples : recherche avec GET

Les onglets suivants fournissent des exemples de recherche sur des types de ressources FHIR spécifiques avec`GET`. Les exemples montrent comment spécifier les paramètres de recherche dans la demande URLs.

> **Note**
>
> La HealthLake console prend uniquement en charge l'autorisation SigV4. L'autorisation SMART on FHIR est prise en charge via AWS CLI et AWS SDKs.
> HealthLake prend en charge un sous-ensemble de paramètres de recherche FHIR R4. Pour de plus amples informations, veuillez consulter Paramètres de recherche.

Patient (age)

Bien que l'âge ne soit pas un type de ressource défini dans le FHIR, il est capturé en tant qu'élément du type de `Patient`ressource. Utilisez l'exemple suivant pour effectuer une demande de recherche `GET` basée sur les types de `Patient`ressources à l'aide de l'élément BirthDate et du `eq` comparateur de recherche pour rechercher des personnes nées en 1997.

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

Condition

Utilisez l'exemple suivant pour effectuer une GET demande sur le type de Conditionressource. La recherche permet de trouver dans votre banque de HealthLake données les conditions qui contiennent le code médical SNOMED72892002, qui se traduit par. Normal pregnancy

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

DocumentationReference

L'exemple suivant montre comment créer une GET demande sur le type de DocumentReferenceressource pour Patient les personnes ayant reçu un diagnostic de streptocoque et à qui l'amoxicilline a également été prescrite.

```
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

Utilisez l'exemple suivant pour effectuer une GET demande sur le type de Locationressource. La recherche suivante permet de trouver les emplacements de votre banque de HealthLake données dont l'adresse contient le nom de ville Boston.

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

Observation

Utilisez l'exemple suivant pour effectuer une demande de recherche GET basée sur le type de Observationressource. Cette recherche utilise le paramètre value-concept de recherche pour rechercher le code médical266919005, qui se traduit parNever smoker.

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

Recherche de ressources FHIR avec POST

Vous pouvez utiliser l'`search`interaction FHIR avec les `POST` demandes de recherche dans un magasin de HealthLake données. Lors de l'utilisation`POST`, HealthLake prend en charge les paramètres de recherche dans l'URL ou dans le corps d'une requête, mais vous ne pouvez pas utiliser les deux dans une seule demande.

> **Important**
>
> Pour les recherches impliquant des informations personnelles identifiables (PII) ou des informations de santé protégées (PHI), les meilleures pratiques de sécurité nécessitent l'utilisation de `POST` demandes, car les informations personnelles et les PHI sont ajoutées dans le corps de la demande et sont cryptées pendant le transfert.

La procédure suivante est suivie d'exemples utilisant l'`search`interaction FHIR R4 avec `POST` pour rechercher un magasin de HealthLake données. Les exemples montrent comment spécifier les paramètres de recherche dans le corps de la requête JSON.

Pour effectuer une recherche dans un magasin de HealthLake données avec **`POST`**

1. Collectez HealthLake `region` et `datastoreId` valorisez. Pour de plus amples informations, veuillez consulter Obtenir les propriétés du magasin de données.
2. Déterminez le type de ressource FHIR à rechercher et collectez la `id` valeur associée. Pour de plus amples informations, veuillez consulter Types de ressources.
3. Créez une URL pour la demande en utilisant les valeurs collectées pour HealthLake `region` et`datastoreId`. Incluez également le `Resource` type de FHIR et `_search` l'interaction. Pour afficher le chemin complet de l'URL dans l'exemple suivant, faites défiler le curseur sur le bouton Copier.

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

4. Construisez un corps JSON pour la demande, en spécifiant les données FHIR à rechercher. Dans le cadre de cette procédure, vous allez rechercher `Observation` des ressources pour découvrir des patients qui n'ont jamais fumé. Pour spécifier le statut du code médical`Never`

smoker, définissez-le value-concept=266919005 dans le corps de la requête JSON. Enregistrez le fichier sous le nom search-observation.json.

```
value-concept=266919005
```

5. Envoyez la demande . L'searchinteraction FHIR utilise la GET demande avec AWS Signature Version 4 ou SMART sur autorisation FHIR.

> **Note**
>
> Lorsque vous faites une POST demande avec des paramètres de recherche dans le corps de la demande, Content-Type: application/x-www-form-urlencoded utilisez-les dans le cadre de l'en-tête.

L'curlexemple suivant génère une demande de recherche basée sur le post sur le type de Observation ressource. La demande utilise le paramètre value-conceptde recherche pour rechercher un code médical 266919005 indiquant une valeurNever smoker. Pour afficher l'exemple dans son intégralité, faites défiler la souris sur le bouton Copier.

SigV4

Autorisation SigV4

```
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

Exemple d'autorisation SMART sur FHIR pour le type de IdentityProviderConfigurationdonnées.

```
{
```

```
    "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\"]}"
}
```

L'appelant peut attribuer des autorisations dans le lambda d'autorisation. Pour de plus amples informations, veuillez consulter Étendue OAuth 2.0.

Exemples : recherche avec POST

Les onglets suivants fournissent des exemples de recherche sur des types de ressources FHIR spécifiques avecPOST. Les exemples montrent comment spécifier une demande dans le URLs.

> **Note**
>
> La HealthLake console prend uniquement en charge l'autorisation SigV4. L'autorisation SMART on FHIR est prise en charge via AWS CLI et AWS SDKs.
> HealthLake prend en charge un sous-ensemble de paramètres de recherche FHIR R4. Pour de plus amples informations, veuillez consulter Paramètres de recherche.

Patient (age)

Bien que l'âge ne soit pas un type de ressource défini dans le FHIR, il est capturé en tant qu'élément du type de `Patient`ressource. Utilisez l'exemple suivant pour effectuer une demande

de recherche POST basée sur le type de `Patient` ressource. L'exemple de recherche suivant utilise le comparateur eq de recherche pour rechercher des personnes nées en 1997.

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

Pour spécifier l'année 1997 dans la recherche, ajoutez l'élément suivant dans le corps de la demande.

```
birthdate=eq1997
```

Condition

Utilisez ce qui suit pour effectuer une POST demande sur le type de `Condition` ressource. Cette recherche permet de trouver les emplacements de votre banque de HealthLake données qui contiennent le code médical`72892002`.

Vous devez spécifier une URL de demande et un corps de demande. Voici un exemple d'URL de demande.

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

Pour spécifier le code médical que vous souhaitez rechercher, ajoutez l'élément JSON suivant dans le corps de la requête.

```
code=72892002
```

DocumentReference

Pour voir les résultats du traitement HealthLake du langage naturel (NLP) intégré lorsque vous faites une POST demande sur le type de `DocumentReference` ressource, formatez la demande comme suit.

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

Pour spécifier les paramètres `DocumentReference` de recherche à référencer, voirTypes de paramètres de recherche. La chaîne de requête suivante utilise plusieurs paramètres de

recherche pour effectuer une recherche sur les opérations de l'API Amazon Comprehend Medical utilisées pour générer les résultats NLP intégrés.

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

Location

Utilisez l'exemple suivant pour effectuer une POST demande sur le type de `Location` ressource. La recherche permet de trouver des emplacements dans votre banque de HealthLake données dont l'adresse contient le nom de la ville de Boston.

Vous devez spécifier une URL de demande et un corps de demande. Voici un exemple d'URL de demande.

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

Pour spécifier `Boston` dans la recherche, ajoutez l'élément suivant dans le corps de la requête :

```
address=Boston
```

Observation

Utilisez l'exemple suivant pour effectuer une demande de recherche POST basée sur le type de `Observation` ressource. La recherche utilise le paramètre `value-concept` de recherche pour rechercher le code médical, `266919005` qui indique`Never smoker`.

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

Pour spécifier le statut`Never smoker`, définissez-le `value-concept=266919005` dans le corps du JSON.

```
value-concept=266919005
```

Niveaux de cohérence des recherches FHIR

L'index HealthLake de recherche d'AWS fonctionne selon un modèle de cohérence éventuelle pour `GET` et `POST` avec les opérations de recherche. Pour des raisons de cohérence, si une mise à jour de l'index de recherche est en attente pour une ressource, les résultats de la recherche excluent la version N-1 de la ressource jusqu'à ce que la mise à jour de l'index soit terminée.

AWS inclut HealthLake désormais la possibilité de sélectionner le comportement du modèle de cohérence pour les ressources mises à jour. Les développeurs peuvent inclure soit la « cohérence éventuelle », le comportement par défaut décrit ci-dessus, soit la « cohérence forte ». Une forte cohérence permettra d'inclure dans les résultats de recherche la version N-1 de la ressource pour les ressources dont les mises à jour de l'index de recherche sont en attente. Cela peut être utilisé pour des scénarios d'utilisation où toutes les ressources sont requises dans le résultat même lorsque la mise à jour de l'index de recherche n'est pas encore terminée. Les clients peuvent contrôler ce comportement à l'aide de l'en-tête de `x-amz-fhir-history-consistency-level` demande.

Niveaux de cohérence

Forte cohérence

`x-amz-fhir-history-consistency-level: strong`Paramétré pour renvoyer tous les enregistrements correspondants, y compris ceux dont les mises à jour de l'index de recherche sont en attente. Utilisez cette option lorsque vous devez rechercher des ressources immédiatement après les mises à jour.

Cohérence à terme

Paramétré `x-amz-fhir-history-consistency-level: eventual` pour renvoyer uniquement les enregistrements dont les mises à jour de l'index de recherche sont terminées. Il s'agit du comportement par défaut si aucun niveau de cohérence n'est spécifié.

Exemple d’utilisation

1. Lors de la mise à jour d'une ressource :

```
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. Recherche ultérieure :

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

Bonnes pratiques

- Faites preuve d'une grande cohérence lorsque vous devez rechercher immédiatement des ressources récemment mises à jour
- Utilisez la cohérence éventuelle pour les requêtes générales pour lesquelles une visibilité immédiate n'est pas essentielle
- Tenez compte du compromis entre visibilité immédiate et impact potentiel sur les performances

> **Note**
>
> Le paramètre du niveau de cohérence influe sur la rapidité avec laquelle les ressources mises à jour apparaissent dans les résultats de recherche, mais n'a aucune incidence sur le stockage réel des ressources.

Le fait de définir l'`x-amz-fhir-history-consistency-level`en-tête facultatif sur « fort » double la consommation de capacité d'écriture par ressource.
Cette fonctionnalité s'applique uniquement aux magasins de données dont l'historique des versions est activé (tous les magasins de données créés après le 25 octobre 2024 l'ont activé par défaut).

Exportation de données FHIR avec AWS HealthLake

Utilisez AWS HealthLake des actions natives pour démarrer, décrire et répertorier les tâches d'exportation FHIR. Vous pouvez mettre en file d'attente les tâches d'exportation. Les tâches d'exportation asynchrones sont traitées selon le mode FIFO (First In First Out). Vous pouvez mettre les tâches en file d'attente de la même manière que vous commencez à exporter des tâches. Si l'un d'entre eux est en cours, il sera simplement mis en file d'attente. Vous pouvez créer, lire, mettre à jour ou supprimer des ressources FHIR pendant qu'une tâche d'exportation est en cours.

> **Note**
>
> Vous pouvez également exporter des données FHIR depuis un magasin de HealthLake données à l'aide de l'opération FHIR `$export` R4. Pour de plus amples informations, veuillez consulter Exporter HealthLake des données avec FHIR `$export`.

Rubriques

- Démarrage d'une tâche d'exportation FHIR
- Obtenir les propriétés des tâches d'exportation FHIR
- Liste des emplois d'exportation FHIR

Démarrage d'une tâche d'exportation FHIR

`StartFHIRExportJob`À utiliser pour démarrer une tâche d'exportation FHIR à partir d'un magasin de HealthLake données. Les menus suivants fournissent une procédure AWS Management Console et des exemples de code pour le AWS CLI et AWS SDKs. Pour plus d'informations, consultez `StartFHIRExportJob` dans la Référence d'API AWS HealthLake .

> **Remarque**
>
> HealthLake prend en charge la spécification FHIR R4 pour l'échange de données sur les soins de santé. Par conséquent, toutes les données de santé sont exportées au format FHIR R4.

Pour démarrer une tâche d'exportation FHIR

Choisissez un menu en fonction de vos préférences d'accès à AWS HealthLake.

AWS CLI et SDKs

CLI

AWS CLI

Pour démarrer une tâche d’exportation FHIR

L'`start-fhir-export-job`exemple suivant montre comment démarrer une tâche d'exportation FHIR à l'aide AWS HealthLake de.

```
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)
```

Sortie :

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

- Pour plus de détails sur l'API, consultez Start FHIRExport Job dans AWS CLI Command Reference.

Python

Kit 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
```

- Pour plus de détails sur l'API, consultez le manuel de référence de l'API Start FHIRExport Job in AWS SDK for Python (Boto3).

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    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.
```

- Pour plus de détails sur l'API, voir Start FHIRExport Job in AWS SDK for SAP ABAP API reference.

> **Exemple de disponibilité**
>
> Vous n’avez pas trouvé ce dont vous avez besoin ? Demandez un exemple de code à l'aide du lien Fournir des commentaires dans la barre latérale droite de cette page.

AWS Console

1. Connectez-vous à la page Data stores de la HealthLake console.
2. Choisissez un magasin de données.
3. Cliquez sur Exporter.

   La page d'exportation s'ouvre.
4. Dans la section Données de sortie, entrez les informations suivantes :

   - Emplacement des données de sortie dans Amazon S3
   - Chiffrement de sortie

5. Dans la section Autorisations d'accès, choisissez Utiliser un rôle de service IAM existant et sélectionnez le rôle dans le menu Nom du rôle ou choisissez Créer un rôle IAM.
6. Choisissez Export data (Exporter des données).

> **Note**
>
> Lors de l'exportation, choisissez Copier l'identifiant de la tâche sur la bannière en haut de la page. Vous pouvez utiliser le `JobID`pour demander les propriétés de la tâche d'exportation à l'aide du AWS CLI. Pour de plus amples informations, veuillez consulter Obtenir les propriétés des tâches d'exportation FHIR.

Obtenir les propriétés des tâches d'exportation FHIR

`DescribeFHIRExportJob`À utiliser pour obtenir les propriétés des tâches d'exportation à partir d'un magasin de HealthLake données. Les menus suivants fournissent une procédure AWS Management Console et des exemples de code pour le AWS CLI et AWS SDKs. Pour plus d'informations, consultez `DescribeFHIRExportJob` dans la Référence d'API AWS HealthLake .

> **Remarque**
>
> HealthLake prend en charge la spécification FHIR R4 pour l'échange de données sur les soins de santé. Par conséquent, toutes les données de santé sont exportées au format FHIR R4.

Pour décrire une tâche d'exportation FHIR

Choisissez un menu en fonction de vos préférences d'accès à AWS HealthLake.

AWS CLI et SDKs

CLI

AWS CLI

Pour décrire une tâche d'exportation FHIR

L'`describe-fhir-export-job`exemple suivant montre comment rechercher les propriétés d'une tâche d'exportation FHIR dans AWS HealthLake.

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

Sortie :

```
{
    "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)"
    }
}
```

- Pour plus de détails sur l'API, voir FHIRExportDescribe Job dans AWS CLI Command Reference.

Python

Kit 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
```

- Pour plus de détails sur l'API, consultez le manuel de référence de l'API Describe FHIRExport Job in AWS SDK for Python (Boto3).

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    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.
```

- Pour plus de détails sur l'API, consultez la section de référence de l'API Describe FHIRExport Job in AWS SDK for SAP ABAP.

Exemple de disponibilité

Vous n’avez pas trouvé ce dont vous avez besoin ? Demandez un exemple de code à l'aide du lien Fournir des commentaires dans la barre latérale droite de cette page.

AWS Console

Note

Les informations relatives aux tâches d'exportation FHIR ne sont pas disponibles sur la HealthLake console. Utilisez plutôt le AWS CLI with `DescribeFHIRExportJob` pour demander des propriétés de tâche d'exportation telles que `JobStatus`. Pour plus d'informations, reportez-vous à l' AWS CLI exemple présenté sur cette page.

Liste des emplois d'exportation FHIR

Permet `ListFHIRExportJobs` de répertorier les tâches d'exportation FHIR pour un magasin HealthLake de données. Les menus suivants fournissent une procédure AWS Management Console et des exemples de code pour le AWS CLI et AWS SDKs. Pour plus d’informations, consultez `ListFHIRExportJobs` dans la Référence d’API AWS HealthLake .

Remarque

HealthLake prend en charge la spécification FHIR R4 pour l'échange de données sur les soins de santé. Par conséquent, toutes les données de santé sont exportées au format FHIR R4.

Pour répertorier les emplois d'exportation FHIR

Choisissez un menu en fonction de vos préférences d'accès à AWS HealthLake.

AWS CLI et SDKs

CLI

AWS CLI

Pour répertorier tous les emplois d'exportation FHIR

L'exemple `list-fhir-export-jobs` suivant montre comment utiliser la commande pour afficher une liste des tâches d'exportation associées à un compte.

```
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)
```

Sortie :

```
{
    "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)"
            }
        }
    ]
```

```
}
```

- Pour plus de détails sur l'API, consultez la section FHIRExportRépertorier les tâches dans AWS CLI Command Reference.

Python

Kit 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
```

- Pour plus de détails sur l'API, consultez le manuel de référence de l'API List FHIRExport Jobs in AWS SDK for Python (Boto3).

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    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.
```

- Pour plus de détails sur l'API, consultez la section List FHIRExport Jobs in AWS SDK for SAP ABAP API reference.

Exemple de disponibilité

Vous n’avez pas trouvé ce dont vous avez besoin ? Demandez un exemple de code à l'aide du lien Fournir des commentaires dans la barre latérale droite de cette page.

AWS Console

Note

Les informations relatives aux tâches d'exportation FHIR ne sont pas disponibles sur la HealthLake console. Utilisez plutôt le AWS CLI with `ListFHIRExportJobs` pour répertorier toutes les tâches d'exportation FHIR. Pour plus d'informations, reportez-vous à l' AWS CLI exemple présenté sur cette page.

Exemples de code pour HealthLake l'utilisation AWS SDKs

Les exemples de code suivants montrent comment utiliser HealthLake un kit de développement AWS logiciel (SDK).

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation HealthLake avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit de développement logiciel (SDK).

Exemples de code

Exemples de base pour HealthLake l'utilisation AWS SDKs

Les exemples de code suivants montrent comment utiliser les principes de base de AWS HealthLake with AWS SDKs.

Exemples

- Actions d' HealthLake utilisation AWS SDKs
  - Utilisation CreateFHIRDatastore avec un AWS SDK ou une CLI
  - Utilisation DeleteFHIRDatastore avec un AWS SDK ou une CLI
  - Utilisation DescribeFHIRDatastore avec un AWS SDK ou une CLI
  - Utilisation DescribeFHIRExportJob avec un AWS SDK ou une CLI
  - Utilisation DescribeFHIRImportJob avec un AWS SDK ou une CLI
  - Utilisation ListFHIRDatastores avec un AWS SDK ou une CLI
  - Utilisation ListFHIRExportJobs avec un AWS SDK ou une CLI
  - Utilisation ListFHIRImportJobs avec un AWS SDK ou une CLI
  - Utilisation ListTagsForResource avec un AWS SDK ou une CLI
  - Utilisation StartFHIRExportJob avec un AWS SDK ou une CLI
  - Utilisation StartFHIRImportJob avec un AWS SDK ou une CLI
  - Utilisation TagResource avec un AWS SDK ou une CLI
  - Utilisation UntagResource avec un AWS SDK ou une CLI

Actions d' HealthLake utilisation AWS SDKs

Les exemples de code suivants montrent comment effectuer des HealthLake actions individuelles avec AWS SDKs. Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions pour configurer et exécuter le code.

Les exemples suivants incluent uniquement les actions les plus couramment utilisées. Pour obtenir la liste complète, consultez la Référence des API AWS HealthLake.

Exemples

- Utilisation CreateFHIRDatastore avec un AWS SDK ou une CLI
- Utilisation DeleteFHIRDatastore avec un AWS SDK ou une CLI
- Utilisation DescribeFHIRDatastore avec un AWS SDK ou une CLI

- Utilisation DescribeFHIRExportJob avec un AWS SDK ou une CLI
- Utilisation DescribeFHIRImportJob avec un AWS SDK ou une CLI
- Utilisation ListFHIRDatastores avec un AWS SDK ou une CLI
- Utilisation ListFHIRExportJobs avec un AWS SDK ou une CLI
- Utilisation ListFHIRImportJobs avec un AWS SDK ou une CLI
- Utilisation ListTagsForResource avec un AWS SDK ou une CLI
- Utilisation StartFHIRExportJob avec un AWS SDK ou une CLI
- Utilisation StartFHIRImportJob avec un AWS SDK ou une CLI
- Utilisation TagResource avec un AWS SDK ou une CLI
- Utilisation UntagResource avec un AWS SDK ou une CLI

Utilisation **`CreateFHIRDatastore`** avec un AWS SDK ou une CLI

Les exemples de code suivants illustrent comment utiliser `CreateFHIRDatastore`.

CLI

AWS CLI

Exemple 1 : création d'un magasin de données compatible SIGv4 HealthLake

L'`create-fhir-datastore`exemple suivant montre comment créer un nouveau magasin de données dans AWS HealthLake.

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

Sortie :

```
{
    "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)"
```

```
}
```

Exemple 2 : créer un magasin de données compatible SMART sur FHIR HealthLake

L'`create-fhir-datastore`exemple suivant montre comment créer un nouveau magasin de données compatible SMART on FHIR dans. AWS HealthLake

```
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
```

Contenu de `identity_provider_configuration.json` :

```
{
    "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\"]}"
}
```

Sortie :

```
{
```

```
    "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)"
}
```

- Pour plus de détails sur l'API, voir Create FHIRDatastore in AWS CLI Command Reference.

Python

Kit 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
```

Le code suivant montre un exemple de paramètres pour un magasin de données compatible SMART sur FHIR HealthLake .

```
            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
            )
```

- Pour plus de détails sur l'API, consultez le manuel de référence de l'API Create FHIRDatastore in AWS SDK for Python (Boto3).

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    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.
```

- Pour plus de détails sur l'API, consultez la section de référence sur l'API Create FHIRDatastore in AWS SDK for SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation HealthLake avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **`DeleteFHIRDatastore`** avec un AWS SDK ou une CLI

Les exemples de code suivants illustrent comment utiliser `DeleteFHIRDatastore`.

CLI

AWS CLI

Pour supprimer un magasin de données FHIR

L'`delete-fhir-datastore`exemple suivant montre comment supprimer un magasin de données et tout son contenu dans AWS HealthLake.

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

Sortie :

```
{
    "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)"
}
```

- Pour plus de détails sur l'API, voir Supprimer FHIRDatastore dans le manuel de référence des AWS CLI commandes.

Python

Kit 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
```

- Pour plus de détails sur l'API, consultez le manuel de référence de l'API Delete FHIRDatastore in AWS SDK for Python (Boto3).

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    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.
```

- Pour plus de détails sur l'API, voir Supprimer FHIRDatastore dans le AWS SDK pour la référence de l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation HealthLake avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **`DescribeFHIRDatastore`** avec un AWS SDK ou une CLI

Les exemples de code suivants illustrent comment utiliser `DescribeFHIRDatastore`.

CLI

AWS CLI

Pour décrire un magasin de données FHIR

L'`describe-fhir-datastore`exemple suivant montre comment rechercher les propriétés d'un magasin de données dans AWS HealthLake.

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

Sortie :

```
{
    "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
```

```
        }
    }
}
```

- Pour plus de détails sur l'API, voir Description FHIRDatastore dans le manuel de référence des AWS CLI commandes.

Python

Kit 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
```

- Pour plus de détails sur l'API, voir AWS Describe FHIRDatastore in SDK for Python (Boto3) API Reference.

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    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.
```

- Pour plus de détails sur l'API, consultez la section Description FHIRDatastore dans le AWS SDK pour la référence de l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation HealthLake avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **`DescribeFHIRExportJob`** avec un AWS SDK ou une CLI

Les exemples de code suivants illustrent comment utiliser `DescribeFHIRExportJob`.

CLI

AWS CLI

Pour décrire une tâche d’exportation FHIR

L'`describe-fhir-export-job`exemple suivant montre comment rechercher les propriétés d'une tâche d'exportation FHIR dans AWS HealthLake.

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

Sortie :

```
{
    "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)"
    }
}
```

- Pour plus de détails sur l'API, voir FHIRExportDescribe Job dans AWS CLI Command Reference.

Python

Kit 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
```

- Pour plus de détails sur l'API, consultez le manuel de référence de l'API Describe FHIRExport Job in AWS SDK for Python (Boto3).

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
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.
```

- Pour plus de détails sur l'API, consultez la section de référence de l'API Describe FHIRExport Job in AWS SDK for SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation HealthLake avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **`DescribeFHIRImportJob`** avec un AWS SDK ou une CLI

Les exemples de code suivants illustrent comment utiliser `DescribeFHIRImportJob`.

CLI

AWS CLI

Pour décrire une tâche d’importation FHIR

L'`describe-fhir-import-job`exemple suivant montre comment apprendre les propriétés d'une tâche d'importation FHIR à l'aide AWS HealthLake de.

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

Sortie :

```
{
```

```
    "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)"
    }
}
```

- Pour plus de détails sur l'API, voir FHIRImportDescribe Job dans AWS CLI Command Reference.

Python

Kit 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
```

- Pour plus de détails sur l'API, consultez le manuel de référence de l'API Describe FHIRImport Job in AWS SDK for Python (Boto3).

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    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.
```

- Pour plus de détails sur l'API, consultez la section de référence de l'API Describe FHIRImport Job in AWS SDK for SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation HealthLake avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **`ListFHIRDatastores`** avec un AWS SDK ou une CLI

Les exemples de code suivants illustrent comment utiliser `ListFHIRDatastores`.

CLI

AWS CLI

Pour répertorier des magasins de données FHIR

L'`list-fhir-datastores`exemple suivant montre comment utiliser la commande et comment les utilisateurs peuvent filtrer les résultats en fonction de l'état du magasin de données dans AWS HealthLake.

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

Sortie :

```
{
    "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
        }
    }
    ]
}
```

- Pour plus de détails sur l'API, consultez la section Liste FHIRDatastores dans AWS CLI la référence des commandes.

Python

Kit 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
```

- Pour plus de détails sur l'API, voir Liste FHIRDatastores dans le AWS manuel de référence de l'API SDK for Python (Boto3).

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    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.
```

- Pour plus de détails sur les API, consultez la section Liste FHIRDatastores dans le AWS SDK pour la référence des API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation HealthLake avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **`ListFHIRExportJobs`** avec un AWS SDK ou une CLI

Les exemples de code suivants illustrent comment utiliser `ListFHIRExportJobs`.

CLI

AWS CLI

Pour répertorier tous les emplois d'exportation FHIR

L'exemple `list-fhir-export-jobs` suivant montre comment utiliser la commande pour afficher une liste des tâches d'exportation associées à un compte.

```
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)
```

Sortie :

```
{
    "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)"
            }
```

```
        }
    ]
}
```

- Pour plus de détails sur l'API, consultez la section FHIRExportRépertorier les tâches dans AWS CLI Command Reference.

Python

Kit 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
```

- Pour plus de détails sur l'API, consultez le manuel de référence de l'API List FHIRExport Jobs in AWS SDK for Python (Boto3).

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l’exemple complet et découvrez comment le configurer et l’exécuter dans le référentiel d’exemples de code AWS.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    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.
```

- Pour plus de détails sur l'API, consultez la section FHIRExportRépertorier les tâches dans le AWS SDK pour la référence de l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation HealthLake avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **`ListFHIRImportJobs`** avec un AWS SDK ou une CLI

Les exemples de code suivants illustrent comment utiliser `ListFHIRImportJobs`.

CLI

AWS CLI

Pour répertorier toutes les tâches d'importation FHIR

L'exemple `list-fhir-import-jobs` suivant montre comment utiliser la commande pour afficher une liste de toutes les tâches d'importation associées à un compte.

```
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)
```

Sortie :

```
{
    "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)"
        }
    ]
}
```

- Pour plus de détails sur l'API, consultez la section FHIRImportRépertorier les tâches dans AWS CLI Command Reference.

Python

Kit 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
```

- Pour plus de détails sur l'API, consultez le manuel de référence de l'API List FHIRImport Jobs in AWS SDK for Python (Boto3).

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    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.
```

- Pour plus de détails sur l'API, consultez la section FHIRImportRépertorier les tâches dans le AWS SDK pour la référence de l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation HealthLake avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **`ListTagsForResource`** avec un AWS SDK ou une CLI

Les exemples de code suivants illustrent comment utiliser `ListTagsForResource`.

CLI

AWS CLI

Pour répertorier les balises d'un magasin de données

L'exemple `list-tags-for-resource` suivant répertorie les balises associées au magasin de données spécifié.

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

Sortie :

```
{
    "tags": {
        "key": "value",
```

```
        "key1": "value1"
    }
}
```

- Pour plus de détails sur l'API, reportez-vous ListTagsForResourceà la section Référence des AWS CLI commandes.

Python

Kit 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
```

- Pour plus de détails sur l'API, consultez ListTagsForResourcele AWS manuel de référence de l'API SDK for Python (Boto3).

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    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.
```

- Pour plus de détails sur l'API, reportez-vous ListTagsForResourceà la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation HealthLake avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **`StartFHIRExportJob`** avec un AWS SDK ou une CLI

Les exemples de code suivants illustrent comment utiliser `StartFHIRExportJob`.

CLI

AWS CLI

Pour démarrer une tâche d'exportation FHIR

L'`start-fhir-export-job`exemple suivant montre comment démarrer une tâche d'exportation FHIR à l'aide AWS HealthLake de.

```
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)
```

Sortie :

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

- Pour plus de détails sur l'API, consultez Start FHIRExport Job dans AWS CLI Command Reference.

Python

Kit 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
```

- Pour plus de détails sur l'API, consultez le manuel de référence de l'API Start FHIRExport Job in AWS SDK for Python (Boto3).

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    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.
```

- Pour plus de détails sur l'API, voir Start FHIRExport Job in AWS SDK for SAP ABAP API reference.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation HealthLake avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **`StartFHIRImportJob`** avec un AWS SDK ou une CLI

Les exemples de code suivants illustrent comment utiliser `StartFHIRImportJob`.

CLI

AWS CLI

Pour démarrer une tâche d'importation FHIR

L'`start-fhir-import-job`exemple suivant montre comment démarrer une tâche d'importation FHIR à l'aide AWS HealthLake de.

```
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)"
```

Sortie :

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

- Pour plus de détails sur l'API, consultez Start FHIRImport Job dans AWS CLI Command Reference.

Python

Kit 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
```

- Pour plus de détails sur l'API, consultez le manuel de référence de l'API Start FHIRImport Job in AWS SDK for Python (Boto3).

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    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.
```

- Pour plus de détails sur l'API, voir Start FHIRImport Job in AWS SDK for SAP ABAP API reference.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation HealthLake avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **`TagResource`** avec un AWS SDK ou une CLI

Les exemples de code suivants illustrent comment utiliser `TagResource`.

CLI

AWS CLI

Pour ajouter une balise au magasin de données

L'exemple `tag-resource` suivant montre comment ajouter une balise au magasin de données.

```
aws healthlake tag-resource \
```

```
    --resource-arn "arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/0725c83f4307f263e16fd56b6d8ebdbe" \
    --tags '[{"Key": "key1", "Value": "value1"}]'
```

Cette commande ne produit aucune sortie.

.

- Pour plus de détails sur l'API, reportez-vous TagResourceà la section Référence des AWS CLI commandes.

Python

Kit 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
```

- Pour plus de détails sur l'API, consultez TagResourcele AWS manuel de référence de l'API SDK for Python (Boto3).

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    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.
```

- Pour plus de détails sur l'API, reportez-vous TagResourceà la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation HealthLake avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **`UntagResource`** avec un AWS SDK ou une CLI

Les exemples de code suivants illustrent comment utiliser `UntagResource`.

CLI

AWS CLI

Pour supprimer des balises d'un magasin de données.

L'exemple `untag-resource` suivant montre comment supprimer des balises d'un magasin de données.

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

Cette commande ne produit aucune sortie.

- Pour plus de détails sur l'API, reportez-vous UntagResourceà la section Référence des AWS CLI commandes.

Python

Kit 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
```

- Pour plus de détails sur l'API, consultez UntagResourcele AWS manuel de référence de l'API SDK for Python (Boto3).

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    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.
```

- Pour plus de détails sur l'API, reportez-vous UntagResourceà la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation HealthLake avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Intégrer AWS HealthLake

Les AWS services suivants s'intègrent directement AWS HealthLake pour permettre le traitement intégré du langage naturel, les requêtes SQL et l'entreposage de données.

- Amazon Comprehend Medical est un service de traitement du langage naturel (NLP) éligible à la loi HIPAA qui utilise des bibliothèques d'apprentissage automatique pour extraire des données de santé significatives à partir de textes médicaux non structurés. HealthLake Pour plus d'informations, consultez le guide du développeur Amazon Comprehend Medical.
- Amazon Athena est un service de requête interactif qui vous permet d'analyser HealthLake des données directement dans des compartiments Amazon Simple Storage Service (Amazon S3) à l'aide du SQL standard. Pour plus d'informations, consultez le guide du développeur Amazon Athena.

Rubriques

Traitement du langage naturel (NLP) intégré pour HealthLake

AWS HealthLake fournit des bibliothèques intégrées de traitement du langage naturel (NLP) pour analyser, identifier et cartographier les données non structurées stockées dans les types de ressources `DocumentReference`FHIR.

> **Important**
>
> La fonction NLP intégrée HealthLake est désactivée par défaut. Pour l'activer, soumettez un dossier d'assistance en utilisant AWS Support Center Console. Pour créer votre dossier, connectez-vous à votre dossier Compte AWS et choisissez Créer un dossier.

HealthLake Le NLP intégré fonctionne en appelant les actions Amazon Comprehend Medical `DetectEntities-V2` et `InferICD10-CM` `InferRxNorm` API. Les actions ajoutent leurs résultats sous forme d'extension à la `DocumentReference` ressource. Lorsque les actions de l'API Amazon Comprehend Medical détectent des traits qui `SIGN` sont `SYMPTOM` `DIAGNOSIS` et/ou génèrent une

ressource `Linkage`FHIR. Les nouvelles `Condition` `Observation` ressources sont créées à partir d'entités identifiées par les traits de`SIGN`, ou`SYMPTOM`, `DIAGNOSIS` et elles sont liées au document source à l'aide de la `Linkage` ressource.

> **Note**
>
> Bien que les `GET` demandes soient prises en charge pour les ressources FHIR générées par le NLP HealthLake intégré, la `search` fonctionnalité de l'API FHIR ne l'est pas. Pour en savoir plus sur la recherche d'extensions NLP à l'aide HealthLake de l'intégration avec Athena, consultez. Index et requête SQL

Table des matières

HealthLake Bibliothèques NLP intégrées

HealthLake déduit les données trouvées dans le type de `DocumentReference` ressource à l'aide des bibliothèques Amazon Comprehend Medical. L'API Amazon Comprehend Medical fonctionne `InferRxNorm` et `DetectEntities-V2` détecte `InferICD10-CM` les problèmes de santé en tant que caractéristiques. Chaque opération fournit des informations différentes.

> **Prise en charge des langages**
>
> Les opérations de l'API Amazon Comprehend Medical détectent uniquement les entités médicales dans les textes en anglais.

- DetectEntities-V2 : inspecte le texte clinique de diverses entités médicales et renvoie des informations spécifiques les concernant, telles que la catégorie d'entité, l'emplacement et le score de confiance.
- Déduire ICD1 0-CM : détecte les affections médicales figurant dans le dossier d'un patient sous forme d'entités, et relie ces entités aux identificateurs conceptuels normalisés de la base de

connaissances ICD-10-CM du National Center for Health Statistics des CDC, avec l'autorisation de l'Organisation mondiale de la santé (OMS).

- InferRxNorm: Détecte les médicaments en tant qu'entités répertoriées dans le dossier d'un patient et les lie aux identificateurs conceptuels normalisés de la RxNorm base de données de la National Library of Medicine.

Les caractéristiques prises en charge pour chaque opération d'API sont `SIGNSYMPTOM`, et`DIAGNOSIS`. Si des traits sont détectés, ils sont ajoutés en tant qu'extensions conformes à la norme FHIR à différents emplacements de votre HealthLake banque de données.

Emplacements où les extensions sont ajoutées.

- `DocumentReference`: Les résultats des opérations de l'API Amazon Comprehend Medical sont ajoutés sous forme `extension` de fichier à chaque document trouvé dans `DocumentReference` le type de ressource. Les résultats de l'extension sont divisés en deux groupes. Vous pouvez les trouver dans les résultats en fonction de leurs`URL`.
  - `http://healthlake.amazonaws.com/system-generated-resources/`
    - Il s'agit de types de ressources qui ont été créés ou ajoutés par HealthLake.
  - `http://healthlake.amazonaws.com/aws-cm/`
    - Où le résultat brut des opérations de l'API Amazon Comprehend Medical est ajouté à HealthLake votre magasin de données.
- `Linkage`: Ce type de ressource est soit ajouté, soit créé grâce au NLP intégré. Une `GET` requête portant sur un sujet spécifique `Linkage` renvoie une liste de ressources liées. Pour savoir si a `Linkage` été ajouté par HealthLake, recherchez la paire `"tag": [{"display": "SYSTEM_GENERATED"}]` clé-valeur ajoutée. Pour en savoir plus sur les spécifications FHIR pour Linkage, consultez `Linkage`la documentation FHIR R4.
- Types de ressources FHIR générés à la suite des opérations d'Amazon Comprehend Medical.
  - `Observation`: inclut les résultats des `DetectEntities-V2` actions de l'API Amazon Comprehend Medical `InferICD10-CM` et indique quand les caractéristiques `SIGN` sont `SYMPTOM` ou.
  - `Condition`: inclut les résultats des `DetectEntities-V2` actions de l'API Amazon Comprehend Medical `InferICD10-CM` et indique quand le trait `DIAGNOSIS` est.
  - `MedicationStatement`: inclut les résultats des actions de l'API `InferRxNorm` Amazon Comprehend Medical.

Utilisation des interactions avec l'API REST FHIR

Par défaut, les caractéristiques détectées par les opérations de l'API Amazon Comprehend Medical ne sont pas renvoyées lors `GET` d'une demande. Pour voir les résultats des opérations NLP intégrées, vous devez spécifier une ressource connue `ID` pour les types de ressources FHIR suivants.

- `Linkage`
- `Observation`
- `Condition`
- `MedicationStatement`

Les résultats des actions NLP HealthLake intégrées en dehors du type de `DocumentReference` ressource sont disponibles à l'aide d'une `GET` requête dont la valeur spécifiée `ID` est connue pour contenir les résultats des opérations de l'API Amazon Comprehend Medical.

Paramètres de recherche pour le HealthLake NLP intégré

Le tableau suivant répertorie les attributs consultables pour le NLP HealthLake intégré.

Paramètres de recherche pour le HealthLake NLP

Paramètres de recherche	Trouve des correspondances pour
Détecter ntities-entity-category	Catégorie d'entité au sein de la DetectEntities sous-extension de l'extension AWS CM
Détecter ntities-entity-text	Texte de l'entité dans la DetectEntities sous-extension de l'extension AWS CM
Détecter ntities-entity-type	Type d'entité dans la DetectEntities sous-extension de l'extension AWS CM
Détecter ntities-entity-score	Entity Score au sein de la DetectEntities sous-extension de l'extension AWS CM
infer-icd10 cm-entity-text	Texte d'entité dans la sous-extension Infer ICD1 0CM au sein de l'extension CM AWS

Paramètres de recherche	Trouve des correspondances pour
infer-icd10 cm-entity-score	Score d'entité dans la sous-extension Infer ICD1 0CM au sein de l'extension CM AWS
infer-icd10 cm-entity-concept-code	Code de concept d'entité dans la sous-extension Infer ICD1 0CM au sein de l'extension CM AWS
infer-icd10 cm-entity-concept-description	Description du concept d'entité dans la sous-extension ICD1 Infer 0CM de l'extension CM AWS
infer-icd10 cm-entity-concept-score	Score du concept d'entité dans la sous-extension Infer ICD1 0CM au sein de l'extension CM AWS
infer-rxnorm-entity-score	Entity Score au sein de la InferRxNorm sous-extension de l'extension AWS CM
infer-rxnorm-entity-text	Texte de l'entité dans la InferRxNorm sous-extension de l'extension AWS CM
infer-rxnorm-entity-concept-code	Code de concept d'entité dans la InferRxNorm sous-extension de l'extension AWS CM
infer-rxnorm-entity-concept-description	Description du concept d'entité dans la InferRxNorm sous-extension de l'extension AWS CM
infer-rxnorm-entity-concept-score	Entity Concept Score au sein de la InferRxNorm sous-extension de l'extension AWS CM

HealthLake fournit une recherche spéciale pour répondre aux critères lorsque `EntityText` et `EntityCategory` font partie de la même entité. Le tableau suivant décrit les paramètres de recherche spéciaux pris en charge par HealthLake.

Paramètres de recherche

Paramètres de recherche	Allumettes retournées
Détecter ntities-entity-text-category	S'il existe au moins une entité dans la DetectEntities sous-extension qui correspond à la fois aux entités EntityText et EntityCategory.
Détecter ntities-entity-type-score	S'il existe au moins une entité dans la DetectEntities sous-extension qui correspond à la fois à EntityType et à EntityScore.
Détecter ntities-entity-text-score	S'il existe au moins une entité dans la DetectEntities sous-extension qui correspond à la fois à EntityText et à EntityScore.
Détecter ntities-entity-text-type	S'il existe au moins une entité dans la DetectEntities sous-extension qui correspond à la fois aux entités EntityText et EntityType.
Détecter ntities-entity-category-score	S'il existe au moins une entité qui correspond à la fois à EntityCategory et à EntityScore.
infer-icd10 -code cm-entity-text-concept	S'il existe au moins une entité dans la sous-extension Infer ICD1 0CM qui correspond à l'EntityText et qu'il existe au moins un ConceptCode pour cette entité qui correspond au code.
score de infer-icd10 cm-entity-text-con cept	S'il existe au moins une entité dans la sous-extension Infer ICD1 0CM qui correspond au EntityText et qu'il existe au moins un ConceptScore pour cette entité qui correspond au score.
infer-icd10 - score conceptuel cm-entity-concept-description	S'il existe au moins un concept dans l'entité de la sous-extension Infer ICD1 0CM qui correspond à la description du concept et au ConceptScore.
infer-rxnorm-entity-text-code conceptuel	S'il existe au moins une entité dans la InferRxNorm sous-extension qui correspond à l'EntityText et qu'au

Paramètres de recherche	Allumettes retournées
	moins un ConceptCode pour cette entité correspond au code.
infer-rxnorm-entity-text-score conceptuel	S'il existe au moins une entité dans la InferRxNorm sous-extension qui correspond au EntityText et qu'au moins un ConceptScore pour cette entité correspond au score.
infer-rxnorm-entity-concept-description-concept-score	S'il existe au moins un concept dans l'entité de la InferRxNorm sous-extension qui correspond à la description du concept et au ConceptScore.

HealthLake exemples de demandes NLP intégrées

Exemple 1 : **`Patient`** enregistrement ingéré dans un magasin de HealthLake données

Voici un exemple de note clinique basée sur une `Patient` rencontre avec un professionnel de la santé.

> **Données synthétiques**
>
> Le texte de l'exemple suivant est un contenu synthétique et ne contient pas d'informations de santé protégées (PHI).

```
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)
```

Pour rappel, ces informations sont encodées au format base64 dans la `DocumentReference` ressource. Lorsque ce document est intégré HealthLake et que les opérations de l'API Amazon Comprehend Medical sont terminées, pour voir les résultats, vous pouvez commencer par la `GET` demande sur le `DocumentReference` type de ressource.

```
GET https://https://healthlake.region.amazonaws.com/datastore/datastoreId/
r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/DocumentReference
```

Lorsque les opérations de l'API Amazon Comprehend Medical sont réussies, recherchez ces paires clé-valeur dans `extension` le lien suivant `"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."
 }
```

Les onglets suivants indiquent comment le dossier médical ingéré est enregistré dans votre banque de HealthLake données en fonction du type de ressource.

DocumentReference

Pour voir les résultats pour un seul type de `DocumentReference` ressource, faites une `GET` demande dans laquelle le `id` nom d'une ressource spécifique est fourni.

```
GET https://https://healthlake.region.amazonaws.com/datastore/datastoreId/
r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/DocumentReference/0e938f03-da7f-4178-acd8-
eea9586c46ed
```

En cas de succès, vous obtenez un code de réponse `200` HTTP et la réponse JSON suivante (qui a été tronquée pour plus de clarté).

Voici la `http://healthlake.amazonaws.com/system-generated-resources/` portion. Vous pouvez voir qu'un nouveau `Linkage/e366d29f-2c22-4c19-866e-09603937935a` a été ajouté. Vous pouvez également voir où des résultats basés sur des inférences ont HealthLake été ajoutés à des types `Observation` de `Condition` ressources spécifiques.

Pour voir comment ces types de ressources ont été modifiés, sélectionnez les onglets correspondants.

```
{
  "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

Pour voir les résultats pour un seul type de `Linkage` ressource, faites une `GET` demande dans laquelle le `ID` nom d'une ressource spécifique est fourni.

```
GET https://https://healthlake.region.amazonaws.com/datastore/datastoreId/
r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/Linkage/e366d29f-2c22-4c19-866e-09603937935a
```

En cas de succès, vous obtenez un code de réponse `200` HTTP et la réponse JSON tronquée suivante.

La réponse contient l'`item`élément. La paire clé-valeur y `"type": "source"` indique l'`DocumentReference`entrée spécifique utilisée pour modifier la `Condition` et `Observations` répertoriée sous la paire `"type": "alternate"` clé-valeur.

Vous voyez également l'*`meta`*élément, ainsi que la paire clé-valeur correspondante*`"tag": [{"display": "SYSTEM_GENERATED"}]`*, indiquant que ces ressources ont été créées par. HealthLake

```
{
  "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

Pour voir les résultats pour un seul type de `Observation` ressource, faites une `GET` demande dans laquelle le `ID` nom d'une ressource spécifique est fourni.

```
GET https://https://healthlake.region.amazonaws.com/
datastore/datastoreId/r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/
Observation/e366d29f-2c22-4c19-866e-09603937935a
```

Les résultats des opérations de l'API Amazon Comprehend Medical sont modifiés selon les éléments suivants `code` :`meta`, `modifierExtension` et.

`code`

Un élément de type`CodeableConcept`. Pour en savoir plus, consultez la `CodeableConcept`documentation du FHIR R4.

HealthLake ajoute les trois paires clé-valeur suivantes.

- `"system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/"`: où l'URL fait référence à une opération spécifique de l'API Amazon Comprehend Medical. Dans ce cas, inférez ICD1 0 cm.
- `"code": "A52.06"`: Où se `A52.06` trouve le code ICD-10-CM qui identifie le concept trouvé dans la base de connaissances des Centers for Disease Control.
- `"display": "Other syphilitic heart involvement"`: Où se `"Other syphilitic heart involvement"` trouve la description détaillée du code ICD-10-CM dans l'ontologie.

La réponse JSON tronquée suivante contient uniquement l'`code`élément.

```
"code": {
  "coding":
  [
    {
   "system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/",
   "code": "A52.06",
   "display": "Other syphilitic heart involvement"
    }
  ],
  "text": "Other syphilitic heart involvement"
 }
```

Pour comprendre dans quelle mesure le modèle est sûr que le code ICD-10-CM attribué est correct, utilisez l'élément. `modifierExtension`

`meta`

L'`meta`élément contient des métadonnées qui indiquent s'il contient des informations ajoutées par les opérations de l'API Amazon Comprehend Medical. `code`

La réponse JSON tronquée suivante contient uniquement l'`meta`élément.

```
"meta": {
  "lastUpdated": "2022-10-21T19:38:30.879Z",
  "tag": [{
   "display": "SYSTEM_GENERATED"
  }]
```

```
 }
```

modifierExtension

L'`modifierExtension`élément contient plus de détails sur le niveau de confiance des codes assignés trouvés dans l'`code`élément. Il comporte également des paires clé-valeur qui fournissent un lien vers l'original DocumentReference utilisé pour générer les résultats et le type de ressource de liaison associé.

Pour chaque `coding` élément ajouté, vous verrez un `entity-score` et un `entity-Concept-Score` ajoutés à la ModifierExtension. Pour chaque valeur de la paire clé-valeur, vous voyez un score. En effet`entity-score`, ce score correspond au niveau de confiance d'Amazon Comprehend Medical quant à la précision de la détection. En effet`entity-Concept-Score`, ce score est le niveau de confiance d'Amazon Comprehend Medical quant à la précision du lien entre l'entité et un concept ICD-10-CM.

La réponse JSON tronquée suivante contient uniquement l'`modifierExtension`élément.

```
"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"
   }
  }
 ]
```

Réponse JSON complète


```
{
  "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

Pour voir les résultats pour un seul type de `Condition` ressource, faites une `GET` demande dans laquelle le `ID` nom d'une ressource spécifique est fourni.

```
GET https://https://healthlake.region.amazonaws.com/datastore/datastoreId/
r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/Condition/b06d343d-
ddb8-4f36-82cb-853fcd434dfd
```

Les résultats des opérations de l'API Amazon Comprehend Medical sont modifiés selon les éléments suivants `code` :`meta`, `modifierExtension` et.

`code`

Un élément de type`CodeableConcept`. Pour en savoir plus, consultez la `CodeableConcept`documentation du FHIR R4.

HealthLake ajoute les trois paires clé-valeur suivantes.

- `"system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/"`: où l'URL fait référence à une opération spécifique de l'API Amazon Comprehend Medical. Dans ce cas, inférez ICD1 0 cm.
- `"code": "I70.0"`: Où se `A52.06` trouve le code ICD-10-CM qui identifie le concept trouvé dans la base de connaissances des Centers for Disease Control.
- `"display": "Atherosclerosis of aorta"`: Où se `"Other syphilitic heart involvement"` trouve la description détaillée du code ICD-10-CM dans l'ontologie.

La réponse JSON tronquée suivante contient uniquement l'`code`élément.

```
"code": {
  "coding":
  [
    {
   "system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/",
   "code": "I70.0",
   "display": "Atherosclerosis of aorta"
```

```
    }
  ],
  "text": "Atherosclerosis of aorta"
 }
```

Pour comprendre dans quelle mesure le modèle est sûr que le code ICD-10-CM attribué est correct, utilisez l'élément. `modifierExtension`

`meta`

L'`meta`élément contient des métadonnées qui indiquent s'il contient des informations ajoutées par les opérations de l'API Amazon Comprehend Medical. `code`

La réponse JSON tronquée suivante contient uniquement l'`meta`élément.

```
"meta": {
  "lastUpdated": "2022-10-21T19:38:30.877Z",
  "tag": [{
   "display": "SYSTEM_GENERATED"
  }]
 }
```

`modifierExtension`

L'`modifierExtension`élément contient plus de détails sur le niveau de confiance des codes assignés trouvés dans l'`code`élément. Il comporte également des paires clé-valeur qui fournissent un lien vers l'original DocumentReference utilisé pour générer les résultats et le type de ressource de liaison associé.

Pour chaque `coding` élément ajouté, vous verrez un `entity-score` et un `entity-Concept-Score` ajoutés à la ModifierExtension. Pour chaque valeur de la paire clé-valeur, vous voyez un score. En effet`entity-score`, ce score correspond au niveau de confiance d'Amazon Comprehend Medical quant à la précision de la détection. En effet`entity-Concept-Score`, ce score est le niveau de confiance d'Amazon Comprehend Medical quant à la précision du lien entre l'entité et un concept ICD-10-CM.

La réponse JSON tronquée suivante contient uniquement l'`modifierExtension`élément.

```
"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"
   }
  }
 ]
```

Réponse JSON complète

```
{
  "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"
 }
```


Exemple 2 : A **`DocumentReference`** contenant le type de **`MedicationStatement`** ressource

Voici un exemple de note clinique basée sur la rencontre d'un patient avec un professionnel de santé.

> ⚠ Données synthétiques
>
> Le texte de cet exemple est un contenu synthétique et ne contient pas d'informations de santé protégées (PHI).

```
Tom is not prescribed Advil
```

Les onglets suivants montrent comment le dossier médical ingéré est enregistré dans votre banque de HealthLake données en fonction du type de ressource.

DocumentReference

Pour voir les résultats pour un seul type de `DocumentReference` ressource, faites une `GET` demande dans laquelle le `ID` nom d'une ressource spécifique est fourni.

```
GET https://https://healthlake.region.amazonaws.com/datastore/datastoreId/
r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/DocumentReference/c549125d-a218-421f-
b8bf-23614c5e796c
```

En cas de succès, vous obtenez un code de réponse `200` HTTP et la réponse JSON tronquée suivante.

La paire clé-valeur indique que les `"url": "http://healthlake.amazonaws.com/system-generated-resources/"` types de ressources qu'elle contient `extension` ont été ajoutés par les opérations de l'API Amazon Comprehend Medical. Vous pouvez voir le nouveau type de `Linkage` ressource et plusieurs `MedicationStatement` ressources.

```
"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

Pour voir les résultats pour un seul type de `Linkage` ressource, faites une `GET` demande dans laquelle le `ID` nom d'une ressource spécifique est fourni.

```
GET https://https://healthlake.region.amazonaws.com/datastore/datastoreId/
r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/Linkage/394bb244-177b-4409-8657-26b20ed56dd7
```

En cas de succès, vous obtenez un code de réponse `200` HTTP et la réponse JSON suivante.

La réponse contient l'`item`élément. La paire clé-valeur y `"type": "source"` indique l'`DocumentReference`entrée spécifique utilisée pour modifier les types de `MedicationStatement` ressources.

Vous pouvez également voir l'`meta`élément et une paire clé-valeur correspondante`"tag": [{"display": "SYSTEM_GENERATED"}]`, indiquant que ces ressources ont été créées par. HealthLake

```
{
  "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

Pour voir les résultats pour un seul type de `MedicationStatement` ressource, faites une `GET` demande dans laquelle le `ID` nom d'une ressource spécifique est fourni.

```
GET https://https://healthlake.region.amazonaws.com/
datastore/datastoreId/r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/
MedicationStatement/9a89b0d3-6681-45ca-9926-27951edce5c7
```

Le type de `MedicationStatement` ressource correspond à l'endroit où se trouvent les résultats de l'opération d'API Amazon Comprehend InferRxNorm Medical. Les résultats sont modifiés selon les éléments suivants : `medicationCodeableConceptmeta`, et`modifierExtension`.

medicationCodeableConcept

Un élément de type`CodeableConcept`. Pour en savoir plus, consultez la `CodeableConcept`documentation du FHIR R4.

HealthLake ajoute les trois paires clé-valeur suivantes.

- `"system": ""http://healthlake.amazonaws.com/aws-cm/infer-rxnorm/`: où l'URL fait référence à une opération spécifique de l'API Amazon Comprehend Medical. Dans ce cas, InferRxNorm.
- `"code": "731533"`: Où se `731533` trouve un identifiant de RxNorm concept, également connu sous le nom de RxCUI.
- `"display": "ibuprofen 200 MG Oral Capsule [Advil]"`: Où se `ibuprofen 200 MG Oral Capsule [Advil]` trouve la description du RxNorm concept.

La réponse JSON tronquée suivante contient uniquement l'`MedicationStatement`élément.

```
"medicationCodeableConcept": {
  "coding": [
   {
    "system": "http://healthlake.amazonaws.com/aws-cm/infer-rxnorm/",
    "code": "731533",
    "display": "ibuprofen 200 MG Oral Capsule [Advil]"
```

```
   }
  ]
 }
```

meta

L'`meta`élément contient des métadonnées qui indiquent s'il contient des informations ajoutées par les opérations de l'API Amazon Comprehend Medical. `code`

La réponse JSON tronquée suivante contient uniquement l'`meta`élément.

```
"meta": {
  "lastUpdated": "2022-10-24T20:05:02.800Z",
  "tag": [
   {
    "display": "SYSTEM_GENERATED"
   }
  ]
 }
```

modifierExtension

L'`modifierExtension`élément contient des paires clé-valeur qui fournissent un lien vers l'original `DocumentReference` utilisé pour générer les résultats et le type de ressource de liaison associé.

```
"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"
    }
   }
 ]
```

Interrogation de HealthLake données avec Amazon Athena

Au cours d'une tâche d' HealthLake importation, les données JSON FHIR imbriquées sont soumises à un processus ETL et sont stockées dans le format de table ouverte Apache Iceberg, où chaque type de ressource FHIR est représenté sous forme de table individuelle dans Athena. Cela permet aux utilisateurs d'interroger les données FHIR à l'aide de SQL, mais sans avoir à les exporter au préalable. Cela est utile, car cela permet aux cliniciens et aux scientifiques d'interroger les données du FHIR pour valider leurs décisions ou faire avancer leurs recherches. Pour plus d'informations sur le fonctionnement des tables Apache Iceberg dans Athena, voir Interroger les tables Apache Iceberg dans le guide de l'utilisateur d'Athena.

> **Note**
>
> HealthLake prend en charge l'`read`interaction FHIR R4 sur vos HealthLake données dans Athena. Pour de plus amples informations, veuillez consulter Lire une ressource FHIR.

Les rubriques de cette section décrivent comment connecter votre banque de HealthLake données à Athena, comment l'interroger à l'aide de SQL et comment connecter les résultats à d'autres AWS services pour une analyse plus approfondie.

Rubriques

- Commencer à utiliser Amazon Athena
- Interrogation de HealthLake données avec SQL
- Exemples de requêtes SQL avec filtrage complexe

Commencer à utiliser Amazon Athena

Pour intégrer HealthLake Amazon Athena, vous devez configurer des autorisations. Pour ce faire, vous allez créer un utilisateur, un groupe ou un rôle Athena et lui accorder l'accès aux ressources FHIR situées dans un HealthLake magasin de données.

- Accorder à un utilisateur, un groupe ou un rôle l'accès à un magasin de HealthLake données (AWS Lake Formation Console)
- Création d'un compte Athena

Accorder à un utilisateur, un groupe ou un rôle l'accès à un magasin de HealthLake données (AWS Lake Formation Console)

> Persona : administrator HealthLake
>
> Le personnage d' HealthLake administrateur est un administrateur de lac de données dans AWS Lake Formation. Ils donnent accès aux magasins de HealthLake données de Lake Formation.

Pour chaque magasin de données créé, deux entrées sont visibles dans la console AWS Lake Formation. L'une des entrées est un lien vers une ressource. Les noms des liens vers les ressources sont toujours affichés en italique. Chaque lien de ressource est affiché avec le nom et le propriétaire de la ressource partagée associée. Pour tous les magasins de HealthLake données, le propriétaire de la ressource partagée est le compte HealthLake de service. L'autre entrée est le magasin HealthLake de données dans le compte HealthLake de service. Les étapes de cette procédure utilisent le magasin de données qui est le lien de ressource.

Pour en savoir plus sur les liens vers des ressources, voir Comment fonctionnent les liens vers des ressources dans Lake Formation in the AWS Lake Formation Developer Guide.

Pour qu'un utilisateur, un groupe ou un rôle puisse interroger des données dans Athena, vous devez accorder l'autorisation Describe sur la base de données de ressources. Ensuite, vous devez autoriser Select et Describe sur les tables.

ÉTAPE 1 : Pour accorder des autorisations DESCRIBE sur une base de HealthLake données de liens de ressources

1. Ouvrez la console AWS Lake Formation : https://console.aws.amazon.com/lakeformation/
2. Dans la barre de navigation principale, sélectionnez Bases de données.
3. Sur la page Bases de données, cliquez sur le bouton radio à côté du nom du magasin de données en italique.
4. Choisissez Actions (▾).
5. Choisissez Accorder.
6. Sur la page Accorder les autorisations relatives aux données, sous Principaux, sélectionnez Utilisateurs ou rôles IAM.

7. Sous Utilisateurs ou rôles IAM, utilisez la flèche vers le bas (▼) ou recherchez l'utilisateur, le rôle ou le groupe IAM sur lequel vous souhaitez pouvoir effectuer des requêtes dans Athena.
8. Sous Balises LF ou carte de ressources de catalogue, choisissez l'option Ressources de catalogue de données nommées.
9. Sous Bases de données, utilisez la flèche vers le bas (▼) pour choisir la HealthLake base de données à laquelle vous souhaitez partager l'accès.
10. Dans la fiche d'autorisation des liens vers les ressources, sous Autorisations des liens vers les ressources, sélectionnez Décrire.

Lorsque l'autorisation est accordée, la bannière de réussite de l'autorisation apparaît. Pour consulter l'autorisation que vous venez d'accorder, choisissez Data lake permissions. Recherchez l'utilisateur, le groupe et le rôle dans le tableau. Sous la colonne Autorisations, vous verrez la liste Décrire.

Vous devez maintenant utiliser Grant on target pour autoriser Select et Describe sur toutes les tables de la base de données.

ÉTAPE 2 : Accorder l'accès à toutes les tables d'un lien de ressource d'un magasin de HealthLake données

1. Ouvrez la console AWS Lake Formation : https://console.aws.amazon.com/lakeformation/
2. Dans la barre de navigation principale, sélectionnez Bases de données.
3. Sur la page Bases de données, cliquez sur le bouton radio à côté du nom du magasin de données en italique.
4. Choisissez Actions (▼).
5. Choisissez Grant on target.
6. Sur la page Accorder les autorisations relatives aux données, sous Principaux, sélectionnez Utilisateurs ou rôles IAM.
7. Sous Utilisateurs ou rôles IAM, utilisez la flèche vers le bas (▼) ou recherchez l'utilisateur, le groupe ou le rôle IAM sur lequel vous souhaitez pouvoir effectuer des requêtes dans Athena.
8. Sous Balises LF ou carte de ressources de catalogue, choisissez l'option Ressources de catalogue de données nommées.
9. Sous Bases de données, utilisez la flèche vers le bas (▼) pour choisir la HealthLake base de données à laquelle vous souhaitez accorder l'accès.
10. Sous Tables, choisissez Toutes les tables pour partager toutes les tables avec un HealthLake utilisateur.

11. Dans la fiche Autorisations du tableau, sous Autorisations du tableau, choisissez Décrire et sélectionner.
12. Choisissez Accorder.

Après avoir choisi l'octroi, une bannière de réussite de l'octroi des autorisations apparaît. L'utilisateur spécifié peut désormais effectuer des requêtes sur un magasin de HealthLake données dans Athena.

Commencer à utiliser Athena

> **HealthLake utilisateur**
>
> L' HealthLake utilisateur utilisera la console Athena ou AWS SDKs pour interroger un magasin de HealthLake données partagé avec lui par l' HealthLake administrateur. AWS CLI

Pour interroger un magasin de données à l'aide d'Athena, vous devez effectuer les trois opérations suivantes.

- Accordez à l'utilisateur ou au rôle IAM l'accès au magasin de HealthLake données via Lake Formation. Pour en savoir plus, veuillez consulter la section Accorder à un utilisateur, un groupe ou un rôle l'accès à un magasin de HealthLake données (AWS Lake Formation Console).
- Créez un groupe de travail pour votre banque HealthLake de données.
- Désignez un compartiment Amazon S3 pour stocker les résultats de vos requêtes.

Pour commencer à utiliser Athena, ajoutez les politiques FullAccess AWS gérées AmazonAthenaFullAccesset AmazonS3 à votre utilisateur, groupe ou rôle. L'utilisation d'une politique AWS gérée est un excellent moyen de commencer à utiliser un nouveau service. N'oubliez pas que les politiques gérées par AWS peuvent ne pas accorder d'autorisations de moindre privilège pour vos cas d'utilisation spécifiques, car elles sont disponibles pour tous les clients AWS. Lorsque vous définissez des autorisations avec des stratégies IAM, accordez uniquement les autorisations nécessaires à l'exécution d'une seule tâche. Pour en savoir plus sur IAM et l'application du moindre privilège, consultez la section Appliquer les autorisations du moindre privilège dans le Guide de l'utilisateur d'IAM.

> **Important**
>
> Pour interroger un magasin de HealthLake données dans Athena, vous devez utiliser le moteur Athena version 3.

Les groupes de travail sont des ressources. Vous pouvez donc utiliser des politiques basées sur l'IAM pour contrôler l'accès à des groupes de travail spécifiques. Pour en savoir plus, consultez la section Utilisation de groupes de travail pour contrôler l'accès aux requêtes et les coûts dans le Guide de l'utilisateur d'Athena.

Pour en savoir plus sur la configuration des groupes de travail, consultez le guide https://docs.aws.amazon.com/athena/latest/ug/workgroups-procedure.html de l'utilisateur d'Athena.

> **Note**
>
> La région dans laquelle se trouve votre compartiment Amazon S3 et la console Athena doivent correspondre.

Pour pouvoir exécuter une requête, vous devez spécifier un emplacement de compartiment de résultats de requête dans Simple Storage Service (Amazon S3) ou utiliser un groupe de travail qui a spécifié un compartiment et dont la configuration remplace les paramètres du client. Les fichiers de sortie sont enregistrés automatiquement pour chaque requête qui s'exécute.

Pour plus de détails sur la spécification de l'emplacement des résultats de requête dans la console Athena, consultez la section Spécification d'un emplacement de résultat de requête à l'aide de la console Athena dans le guide de l'utilisateur d'Amazon Athena.

Pour voir des exemples illustrant la manière d'interroger votre HealthLake banque de données dans Athena, reportez-vous à. Interrogation de HealthLake données avec SQL

Interrogation de HealthLake données avec SQL

Lorsque vous importez vos données FHIR dans le magasin de HealthLake données, les données FHIR JSON imbriquées sont simultanément soumises à un processus ETL et sont stockées au format de table ouverte Apache Iceberg dans Amazon S3. Chaque type de ressource FHIR de votre banque de HealthLake données est converti en table, dans laquelle il peut être interrogé à l'aide

d'Amazon Athena. Les tables peuvent être interrogées individuellement ou en groupe à l'aide de requêtes SQL. En raison de la structure des magasins de données, vos données sont importées dans Athena sous la forme de plusieurs types de données différents. Pour en savoir plus sur la création de requêtes SQL pouvant accéder à ces types de données, consultez la section Tableaux de requêtes dotés de types complexes et de structures imbriquées dans le guide de l'utilisateur d'Amazon Athena.

> **Note**
>
> Tous les exemples présentés dans cette rubrique utilisent des données fictives créées à l'aide de Synthea. Pour en savoir plus sur la création d'un magasin de données préchargé avec des données Synthea, consultez. Création d'un magasin HealthLake de données

Pour chaque élément d'un type de ressource, la spécification FHIR définit une cardinalité. La cardinalité d'un élément définit les limites inférieure et supérieure du nombre de fois que cet élément peut apparaître. Lorsque vous créez une requête SQL, vous devez en tenir compte. Par exemple, examinons certains éléments du champ Type de ressource : Patient.

- Elément : Nom La spécification FHIR définit la cardinalité comme. `0..*`

  L'élément est capturé sous forme de tableau.

```
[{
 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
}]
```

Dans Athena, pour savoir comment un type de ressource a été ingéré, recherchez-le sous Tables et vues. Pour accéder aux éléments de ce tableau, vous pouvez utiliser la notation par points. Voici un exemple simple qui permettrait d'accéder aux valeurs de `given` et`family`.

```
SELECT
    name[1].given as FirstName,
    name[1].family as LastName
FROM Patient
```

- Élément : MaritalStatus La spécification FHIR définit la cardinalité comme. `0..1`

  Cet élément est capturé au format JSON.

```
{
 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
}
```

  Dans Athena, pour savoir comment un type de ressource a été ingéré, recherchez-le sous Tables et vues. Pour accéder aux paires clé-valeur dans le JSON, vous pouvez utiliser la notation par points. Comme il ne s'agit pas d'un tableau, aucun index de tableau n'est requis. Voici un exemple simple qui permettrait d'accéder à la valeur d`etext`.

```
SELECT
    maritalstatus.text as MaritalStatus
FROM Patient
```

Pour en savoir plus sur l'accès au JSON et la recherche dans celui-ci, consultez la section Interrogation de JSON dans le guide de l'utilisateur d'Athena.

Les instructions de requête DML (Athena Data Manipulation Language) sont basées sur Trino. Athena ne prend pas en charge toutes les fonctionnalités de Trino, et il existe des différences importantes. Pour en savoir plus, consultez les requêtes, fonctions et opérateurs DML dans le guide de l'utilisateur d'Amazon Athena.

En outre, Athena prend en charge plusieurs types de données que vous pouvez rencontrer lors de la création de requêtes dans votre banque de HealthLake données. Pour en savoir plus sur les types de données dans Athena, consultez la section Types de données dans Amazon Athena dans le guide de l'utilisateur d'Amazon Athena.

Pour en savoir plus sur le fonctionnement des requêtes SQL dans Athena, consultez la référence SQL pour Amazon Athena dans le guide de l'utilisateur d'Amazon Athena.

Chaque onglet présente des exemples de recherche sur les types de ressources spécifiés et les éléments associés à l'aide d'Athena.

Element: Extension

L'élément `extension` est utilisé pour créer des champs personnalisés dans un magasin de données.

Cet exemple montre comment accéder aux fonctionnalités de l'`extension`élément présent dans le type de `Patient` ressource.

Lorsque votre banque de HealthLake données est importée dans Athena, les éléments d'un type de ressource sont analysés différemment. Étant donné que la structure de la variable `element` is ne peut pas être entièrement spécifiée dans le schéma. Pour gérer cette variabilité, les éléments du tableau sont transmis sous forme de chaînes.

Dans la description du tableau de`Patient`, vous pouvez voir l'élément `extension` décrit comme`array<string>`, ce qui signifie que vous pouvez accéder aux éléments du tableau

en utilisant une valeur d'index. Pour accéder aux éléments de la chaîne, vous devez toutefois utiliser`json_extract`.

Voici une seule entrée de l'`extension`élément trouvé dans le tableau des patients.

```
[{
  "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"
 }
]
```

Même s'il s'agit d'un JSON valide, Athena le traite comme une chaîne.

Cet exemple de requête SQL montre comment créer une table contenant les `patient-birthPlace` éléments `patient-mothersMaidenName` et. Pour accéder à ces éléments, vous devez utiliser différents indices de tableau et `json_extract.`

```
SELECT
    extension[1],
    json_extract(extension[1], '$.valueString') AS MothersMaidenName,
    extension[2],
    json_extract(extension[2], '$.valueAddress.city') AS birthPlace
FROM patient
```

Pour en savoir plus sur les requêtes impliquant du JSON, consultez la section Extraction de données à partir de JSON dans le guide de l'utilisateur Amazon Athena.

Element: birthDate (Age)

L'âge n'est pas un élément du type de ressource patient dans le FHIR. Voici deux exemples de recherches filtrées en fonction de l'âge.

Comme l'âge n'est pas un élément, nous utilisons le `birthDate` pour les requêtes SQL. Pour voir comment un élément a été ingéré dans FHIR, recherchez le nom de la table sous Tables et vues. Vous pouvez voir qu'il est de type chaîne.

Exemple 1 : Calcul d'une valeur pour l'âge

Dans cet exemple de requête SQL, nous utilisons un outil SQL intégré `year` pour extraire ces composants. `current_date` Ensuite, nous les soustrayons pour renvoyer l'âge réel du patient sous la forme d'une colonne appeléeage.

```
SELECT
 (year(current_date) - year(date(birthdate))) as age
FROM patient
```

Exemple 2 : Filtrage pour les patients nés avant `2019-01-01` et ceux qui le sont`male`.

La requête SQL vous montre comment utiliser la `CAST` fonction pour convertir l'`birthDate`élément en type `DATE` et comment filtrer en fonction des deux critères de la `WHERE` clause. Étant donné que l'élément est ingéré en tant que chaîne de caractères par défaut, nous devons l'`CAST`ingérer en tant que type`DATE`. Ensuite, vous pouvez utiliser l'<opérateur pour le comparer à une autre date,`2019-01-01`. En utilisant`AND`, vous pouvez ajouter un deuxième critère à la `WHERE` clause.

```
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

Cet exemple montre les recherches de lieux dans le type de ressource Location dont le nom de ville est Attleboro.

```
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

La condition du type de ressource stocke les données de diagnostic relatives à des problèmes devenus préoccupants. HealthLakele traitement médical intégré du langage naturel (NLP) génère de nouvelles `Condition` ressources en fonction des détails trouvés dans le type de DocumentReference ressource. Lorsque de nouvelles ressources sont générées, HealthLake ajoute la balise `SYSTEM_GENERATED` à l'`meta`élément. Cet exemple de requête SQL montre comment effectuer une recherche dans la table des conditions et renvoyer des résultats lorsque les `SYSTEM_GENERATED` résultats ont été supprimés.

Pour en savoir plus sur HealthLake le traitement automatique du langage naturel (NLP) intégré, consultezTraitement du langage naturel (NLP) intégré pour HealthLake.

```
SELECT *
FROM condition
WHERE meta.tag[1] is NULL
```

Vous pouvez également effectuer une recherche dans un élément de chaîne spécifique pour filtrer davantage votre requête. L'`modifierextension`élément contient des détails sur la `DocumentReference` ressource qui a été utilisée pour générer un ensemble de conditions. Encore une fois, vous devez utiliser `json_extract` pour accéder aux éléments JSON imbriqués qui sont introduits dans Athena sous forme de chaîne.

Cet exemple de requête SQL montre comment vous pouvez rechercher tout `Condition` ce qui a été généré en fonction d'un élément spécifique`DocumentReference`. `CAST`À utiliser pour définir l'élément JSON sous forme de chaîne afin de pouvoir l'utiliser `LIKE` pour comparer.

```
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

Le type de ressource Observation stocke les mesures et les assertions simples faites à propos d'un patient, d'un appareil ou d'un autre sujet. HealthLakele traitement du langage naturel (NLP) intégré génère de nouvelles `Observation` ressources en fonction des détails trouvés dans une `DocumentReference` ressource. Cet exemple de requête SQL inclut des `WHERE meta.tag[1] is NULL` commentaires, ce qui signifie que les `SYSTEM_GENERATED` résultats sont inclus.

```
SELECT valueCodeableConcept.coding[1].code
FROM Observation
WHERE  valueCodeableConcept.coding[1].code = '266919005'
-- WHERE meta.tag[1] is NULL
```

Cette colonne a été importée en tant que `struct`. Par conséquent, vous pouvez accéder aux éléments qu'il contient en utilisant la notation par points.

Resource type: MedicationStatement

MedicationStatement est un type de ressource FHIR que vous pouvez utiliser pour stocker des informations sur les médicaments qu'un patient a pris, prend ou prendra à l'avenir. HealthLakele traitement médical intégré du langage naturel (NLP) génère de nouvelles MedicationStatement ressources sur la base des documents trouvés dans le type de DocumentReference ressource. Lorsque de nouvelles ressources sont générées, HealthLake ajoute la balise `SYSTEM_GENERATED` à l'`meta`élément. Cet exemple de requête SQL montre comment créer une requête filtrant en fonction d'un seul patient à l'aide de son identifiant et recherchant les ressources ajoutées par HealthLake le NLP intégré.

```
SELECT *
FROM medicationstatement
WHERE meta.tag[1].display = 'SYSTEM_GENERATED' AND subject.reference =
 'Patient/0679b7b7-937d-488a-b48d-6315b8e7003b';
```

Pour en savoir plus sur HealthLake le traitement automatique du langage naturel (NLP) intégré, consultezTraitement du langage naturel (NLP) intégré pour HealthLake.

Exemples de requêtes SQL avec filtrage complexe

Les exemples suivants montrent comment utiliser les requêtes SQL Amazon Athena avec un filtrage complexe pour localiser les données FHIR depuis un HealthLake magasin de données.

Example Créez des critères de filtrage basés sur les données démographiques

Il est important d'identifier les données démographiques correctes lors de la création d'une cohorte de patients. Cet exemple de requête montre comment utiliser la notation par points Trino et `json_extract` comment filtrer les données dans votre banque de HealthLake données.

```
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;
```

À l'aide de la console Athena, vous pouvez mieux trier et télécharger les résultats.

Example Créez des filtres pour un patient et ses affections associées

L'exemple de requête suivant montre comment vous pouvez rechercher et trier toutes les affections associées aux patients trouvés dans une banque de HealthLake données.

```
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;
```

Vous pouvez utiliser la console Athena pour trier davantage les résultats ou les télécharger pour une analyse plus approfondie.

Example Créez des filtres pour les patients et leurs observations associées

L'exemple de requête suivant montre comment rechercher et trier toutes les observations associées relatives aux patients trouvées dans un magasin de HealthLake données.

```
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 Créez des conditions de filtrage pour un patient et les procédures associées

Connecter les procédures aux patients est un aspect important des soins de santé. L'exemple de requête SQL suivant montre comment utiliser FHIR `Patient` et les types de `Procedure` ressources pour y parvenir. La requête SQL suivante renverra tous les patients et les procédures associées trouvés dans votre banque de HealthLake données.

```
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;
```

Vous pouvez utiliser la console Athena pour télécharger les résultats en vue d'une analyse plus approfondie ou pour les trier afin de mieux comprendre les résultats.

Example Créez des conditions de filtrage pour un patient et ses prescriptions associées

Il est important de consulter la liste à jour des médicaments que les patients prennent. À l'aide d'Athena, vous pouvez écrire une requête SQL qui utilise à la fois les types de `MedicationRequest` ressources `Patient` et les types de ressources présents dans votre banque de HealthLake données.

La requête SQL suivante joint les `MedicationRequest` tables `Patient` et importées dans Athena. Il organise également les prescriptions en entrées individuelles à l'aide de la notation par points.

```
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
```

Vous pouvez utiliser la console Athena pour trier les résultats ou les télécharger pour une analyse plus approfondie.

Example Voir les médicaments trouvés dans le type de **`MedicationStatement`** ressource

L'exemple de requête suivant vous montre comment organiser le JSON imbriqué importé dans Athena à l'aide de SQL. La requête utilise l'`meta`élément FHIR pour indiquer quand un médicament

a été ajouté par le traitement HealthLake du langage naturel (NLP) intégré. Il est également utilisé `json_extract` pour rechercher des données dans le tableau de chaînes JSON. Pour de plus amples informations, veuillez consulter Traitement du langage naturel.

```
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;
```

Vous pouvez utiliser la console Athena pour télécharger ces résultats ou les trier.

Example Filtre pour un type de maladie spécifique

L'exemple montre comment vous pouvez trouver un groupe de patients âgés de 18 à 75 ans chez qui on a diagnostiqué un diabète.

```
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%');
```

Vous pouvez désormais utiliser la console Athena pour trier les résultats ou les télécharger pour une analyse plus approfondie.

Surveillance AWS HealthLake

La surveillance et la journalisation sont des éléments importants du maintien de la sécurité, de la fiabilité, de la disponibilité et des performances de AWS HealthLake. AWS fournit les services suivants pour surveiller HealthLake, signaler un problème et prendre des mesures automatiques le cas échéant.

- AWS CloudTrailcapture les appels d'API et les événements associés effectués par ou pour le compte de votre AWS compte et envoie les fichiers journaux dans un compartiment Amazon S3 que vous spécifiez. Vous pouvez identifier les utilisateurs et les comptes appelés AWS, l'adresse IP source à partir de laquelle les appels ont été effectués et la date des appels. Pour plus d'informations, consultez le Guide de l'utilisateur AWS CloudTrail.
- Amazon CloudWatch surveille vos AWS ressources et les applications que vous utilisez AWS en temps réel. Vous pouvez collecter et suivre les métriques, créer des tableaux de bord personnalisés, et définir des alarmes qui vous informent ou prennent des mesures lorsqu'une métrique spécifique atteint un seuil que vous spécifiez. Par exemple, vous pouvez CloudWatch suivre l'utilisation du processeur ou d'autres indicateurs de vos EC2 instances Amazon et lancer automatiquement de nouvelles instances en cas de besoin. Pour plus d'informations, consultez le guide de CloudWatch l'utilisateur Amazon.
- Amazon EventBridge est un service de bus d'événements sans serveur qui permet de connecter facilement vos applications à des données provenant de diverses sources. EventBridge fournit un flux de données en temps réel à partir de vos propres applications, applications Software-as-a-Service (SaaS) et AWS services et achemine ces données vers des cibles telles que Lambda. Cela vous permet de surveiller les événements qui se produisent dans les services et de créer des architectures basées sur les événements. Pour plus d'informations, consultez le guide de EventBridge l'utilisateur Amazon.

Rubriques

Journalisation des appels HealthLake d'API à l'aide AWS CloudTrail

AWS HealthLake est intégré à AWS CloudTrail un service qui fournit un enregistrement des actions entreprises par un utilisateur, un rôle ou un AWS service dans HealthLake. CloudTrail capture tous les appels d'API HealthLake sous forme d'événements. Les appels capturés incluent des appels provenant de la HealthLake console et des appels de code vers les opérations de l' HealthLake API. Si vous créez un suivi, vous pouvez activer la diffusion continue d' CloudTrail événements vers un compartiment Amazon S3, y compris les événements pour HealthLake. Si vous ne configurez pas de suivi, vous pouvez toujours consulter les événements les plus récents dans la CloudTrail console dans Historique des événements. À l'aide des informations collectées par CloudTrail, vous pouvez déterminer la demande qui a été faite HealthLake, l'adresse IP à partir de laquelle la demande a été faite, qui a fait la demande, quand elle a été faite et des détails supplémentaires.

Pour en savoir plus CloudTrail, consultez le guide de AWS CloudTrail l'utilisateur.

AWS HealthLake Informations dans CloudTrail

CloudTrail est activé sur votre AWS compte lorsque vous le créez. Lorsqu'une activité se produit dans HealthLake, cette activité est enregistrée dans un CloudTrail événement avec d'autres événements de AWS service dans l'historique des événements. Vous pouvez afficher, rechercher et télécharger les événements récents dans votre compte AWS . Pour plus d'informations, consultez la section Affichage des événements à l'aide de l'historique des CloudTrail événements.

Pour un enregistrement continu des événements de votre AWS compte, y compris des événements pour HealthLake, créez un parcours. Un suivi permet CloudTrail de fournir des fichiers journaux à un compartiment Amazon S3. Par défaut, lorsque vous créez un journal d'activité dans la console, il s’applique à toutes les régions AWS. Le journal enregistre les événements de toutes les régions de la AWS partition et transmet les fichiers journaux au compartiment Amazon S3 que vous spécifiez. En outre, vous pouvez configurer d'autres AWS services pour analyser plus en détail les données d'événements collectées dans les CloudTrail journaux et agir en conséquence. Pour plus d’informations, consultez les ressources suivantes :

- Vue d’ensemble de la création d’un journal d’activité
- CloudTrail Services et intégrations pris en charge
- Configuration des notifications Amazon SNS pour CloudTrail

- Réception de fichiers CloudTrail journaux de plusieurs régions et réception de fichiers CloudTrail journaux de plusieurs comptes

Toutes les HealthLake actions sont enregistrées CloudTrail et documentées dans la référence de l'HealthLake API et dans ce guide du développeur pour les actions effectuées à l'aide de l'API REST FHIR. Par exemple, les appels aux actions suivantes génèrent des entrées dans les fichiers CloudTrail journaux :

- `DescribeFHIRImportJob`
- `DescribeFHIRExportJob`
- `StartFHIRImportJob`
- `ListFHIRImportJobs`
- `StartFHIRExportJob`
- `ListFHIRExportJobs`
- `CreateFHIRDatastore`
- `ListFHIRDatastores`
- `DeleteFHIRDatastore`
- `DescribeFHIRDatastore`
- `UpdateResource`
- `CreateResource`
- `DeleteResource`
- `ReadResource`
- `GetCapabilities`
- `SearchWithGet`
- `SearchWithPost`

Chaque événement ou entrée de journal contient des informations sur la personne ayant initié la demande. Les informations relatives à l'identité permettent de déterminer les éléments suivants :

- Si la demande a été faite avec les informations d'identification de l'utilisateur root ou Gestion des identités et des accès AWS (IAM).
- Si la demande a été effectuée avec les informations d'identification de sécurité temporaires d'un rôle ou d'un utilisateur fédéré.

- Si la demande a été faite par un autre AWS service.

Pour plus d'informations, consultez la section Élément userIdentity CloudTrail .

Comprendre les entrées du fichier AWS HealthLake journal

Un suivi est une configuration qui permet de transmettre des événements sous forme de fichiers journaux à un compartiment Amazon S3 que vous spécifiez. CloudTrail les fichiers journaux contiennent une ou plusieurs entrées de journal. Un événement représente une demande unique provenant de n'importe quelle source et inclut des informations sur l'action demandée, la date et l'heure de l'action, les paramètres de la demande, etc. CloudTrail les fichiers journaux ne constituent pas une trace ordonnée des appels d'API publics, ils n'apparaissent donc pas dans un ordre spécifique.

L'exemple suivant montre une entrée de CloudTrail journal illustrant l'`CreateFHIRDatastore`action.

```
{
    "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"
}
```


Surveillance des HealthLake métriques à l'aide d'Amazon CloudWatch

Vous pouvez effectuer un suivi HealthLake à l'aide d'Amazon CloudWatch, qui collecte les données brutes et les transforme en indicateurs lisibles en temps quasi réel. Ces statistiques sont conservées pendant 15 mois, afin que vous puissiez utiliser ces informations historiques et avoir une meilleure idée des performances de votre application ou service Web. Vous pouvez également définir des alarmes qui surveillent certains seuils et envoient des notifications ou prennent des mesures lorsque ces seuils sont atteints. Pour plus d'informations, consultez le guide de CloudWatch l'utilisateur Amazon.

> **Note**
>
> Les métriques sont signalées pour toutes les HealthLakeactions natives.

Les tableaux suivants répertorient HealthLake les mesures et les dimensions signalées à CloudWatch. Chacun est présenté sous forme de comptage de fréquences pour une plage de données spécifiée par l'utilisateur.

Les HealthLake mesures suivantes sont communiquées à CloudWatch.

HealthLake métriques communiquées à CloudWatch

Métrique	Description
Nombre d'appels	Le nombre d'appels vers APIs. Cela peut être signalé pour le compte ou pour un magasin de données spécifique. Unités : nombre Statistiques valides : somme, nombre Dimensions : fonctionnement, ID de banque de données, type de banque de données
Demandes réussies	Le nombre de demandes d'API réussies. Unités : nombre Statistiques valides : somme, moyenne Dimensions : fonctionnement, identifiant du magasin de données, type de magasin de données
Erreurs de l'utilisateur	Le nombre de demandes qui ont échoué en raison d'une erreur de l'utilisateur. Unités : nombre Statistiques valides : somme, moyenne

Métrique	Description
	Dimensions : fonctionnement, identifiant du magasin de données, type de magasin de données
Erreurs de serveur	Le nombre de demandes qui ont échoué en raison d'une erreur du serveur. Unités : nombre Statistiques valides : somme, moyenne Dimensions : fonctionnement, identifiant du magasin de données, type de magasin de données
Demandes restreintes	Le nombre de demandes qui ont été limitées. Cette métrique n'est pas incluse dans le nombre d'erreurs des utilisateurs ou des serveurs. Unités : nombre Statistiques valides : somme, moyenne Dimensions : fonctionnement, identifiant du magasin de données, type de magasin de données

Métrique	Description
Latence	Le temps nécessaire, en millisecondes, pour traiter la demande de l'utilisateur. Unité : millisecondes Statistiques valides : Minimum, Maximum, Average Dimensions : fonctionnement, identifiant du magasin de données, type de magasin de données

Les HealthLake dimensions suivantes sont indiquées à CloudWatch.

HealthLake Dimensions signalées à CloudWatch

Dimension	Description
Opération	L'opération d'API utilisée dans la demande
DataStoreID	L'identifiant du magasin de données utilisé dans la demande
DataStoreType	Type de magasin de données utilisé dans la demande

Vous pouvez obtenir des métriques à l' HealthLake aide de l'AWS Management Console, de AWS CLI, ou de l' CloudWatchAPI. Vous pouvez utiliser l' CloudWatch API via l'un des kits de développement logiciel Amazon AWS (SDKs) ou les outils CloudWatch d'API. La HealthLake console affiche des graphiques basés sur les données brutes de l' CloudWatch API.

Vous devez disposer des CloudWatch autorisations appropriées pour effectuer la surveillance HealthLake CloudWatch. Pour plus d'informations, consultez la section Gestion des identités et des accès pour Amazon CloudWatch dans le guide de CloudWatch l'utilisateur Amazon.

Afficher HealthLake les métriques

Pour consulter les métriques (CloudWatch console)

1. Connectez-vous à la CloudWatchconsole AWS Management Console et ouvrez-la.
2. Choisissez Metrics, choisissez All Metrics, puis AWS/ HealthLake.
3. Choisissez la dimension, le nom de la métrique, puis Ajouter au graphique.
4. Choisissez une valeur pour la plage de dates. Le décompte de la métrique pour la plage de dates sélectionnée est affiché dans le graphique.

Création d'une alarme à l'aide de CloudWatch

Une CloudWatch alarme surveille une seule métrique sur une période spécifiée et exécute une ou plusieurs actions : envoyer une notification à une rubrique Amazon Simple Notification Service (SNS) ou à une politique Auto Scaling. L'action ou les actions sont basées sur la valeur de la métrique par rapport à un seuil donné sur un certain nombre de périodes que vous spécifiez. CloudWatch peut également vous envoyer un message SNS lorsque l'alarme change d'état.

> **Note**
>
> CloudWatch les alarmes appellent des actions uniquement lorsque l'état change et persiste pendant la période que vous spécifiez.

Pour consulter les métriques (CloudWatch console)

1. Connectez-vous à la console CloudWatch.
2. Choisissez Alarmes, puis Créer une alarme.
3. Choisissez AWS/ HealthLake, puis choisissez une métrique.
4. Pour Période, choisissez un intervalle de temps à surveiller, puis cliquez sur Suivant.
5. Saisissez un Name (Nom) et une Description.
6. Pour Whenever, choisissez **>=**, puis saisissez une valeur maximale.
7. Si vous souhaitez CloudWatch envoyer un e-mail lorsque l'état d'alarme est atteint, dans la section Actions, pour Chaque fois que cette alarme est atteinte, choisissez State is ALARM. Pour Envoyer une notification à, choisissez une liste de diffusion ou choisissez Nouvelle liste et créez une nouvelle liste de diffusion.

8. Affichez un aperçu de l'alarme dans la section Aperçu de l'alarme. Si elle vous convient, choisissez Créer une alarme.

Surveillance des HealthLake événements à l'aide d'Amazon EventBridge

Amazon EventBridge est un service sans serveur qui utilise des événements pour connecter les composants de l'application entre eux, ce qui vous permet de créer plus facilement des applications évolutives pilotées par des événements. La base EventBridge est de créer des règles qui acheminent les événements vers des cibles. AWS HealthLake fournit une mise en œuvre durable des modifications apportées à l'État EventBridge. Pour plus d'informations, consultez Qu'est-ce qu'Amazon EventBridge ? dans le guide de EventBridge l'utilisateur Amazon.

> **Note**
>
> Pour savoir comment envoyer HealthLake des événements à Amazon EventBridge, consultez l'article sur EventBridge l'intégration d'Amazon AWS HealthLake dans le blog AWS for Industries.

Rubriques

- HealthLake événements envoyés à EventBridge
- HealthLake structure de l'événement

HealthLake événements envoyés à EventBridge

Le tableau suivant répertorie tous les HealthLake événements envoyés EventBridge pour traitement.

HealthLake type d'événement	State
Événements liés à la banque de données	
Création d'un magasin de données	`CREATING`
Stockage de données actif	`ACTIVE`

HealthLake type d'événement	State
Suppression du magasin de données	`DELETING`
Stockage de données supprimé	`DELETED`
Échec de la création du magasin de données	`CREATE_FAILED`
Pour plus d'informations, consultez `datastoreStatus` dans la Référence d'API AWS HealthLake .	
Importer des événements professionnels	
Importer un job soumis	`SUBMITTED`
Importer le Job en cours	`IN_PROGRESS`
Job d'importation terminé avec des erreurs	`COMPLETED_WITH_ERRORS`
Job d'importation terminé	`COMPLETED`
Échec de la tâche d'importation	`FAILED`
Pour plus d'informations, consultez `jobStatus` dans la Référence d'API AWS HealthLake .	
Exporter les événements professionnels	
Exporter le job soumis	`SUBMITTED`
Exporter le Job en cours	`IN_PROGRESS`
Job d'exportation terminé avec des erreurs	`COMPLETED_WITH_ERRORS`
Job d'exportation terminé	`COMPLETED`
Echec de la tâche d'exportation	`FAILED`
Pour plus d'informations, consultez `jobStatus` dans la Référence d'API AWS HealthLake .	

HealthLake structure de l'événement

HealthLake les événements sont des objets dotés d'une structure JSON qui contiennent également des détails sur les métadonnées. Vous pouvez utiliser les métadonnées comme entrée pour recréer un événement ou obtenir plus d'informations. Tous les champs de métadonnées associés sont répertoriés dans un tableau sous les exemples de code dans les menus suivants. Pour plus d'informations, consultez les métadonnées des événements de AWS service dans le guide de EventBridge l'utilisateur Amazon.

> **Note**
>
> Pour savoir comment envoyer HealthLake des événements à Amazon EventBridge, consultez l'article sur EventBridge l'intégration d'Amazon AWS HealthLake dans le blog AWS for Industries.

Événements liés à la banque de données

Data Store Creating

État - **`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

État - **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

État - **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

État - **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/"
    }
}
```

Événements du magasin de données : descriptions des métadonnées

Nom	Type	Description
`version`	chaîne	Version du schéma d' EventBridge événements.
`id`	chaîne	L'UUID version 4 généré pour chaque événement.
`detail-type`	chaîne	Type d'événement envoyé.
`source`	chaîne	Identifie le service qui a généré l'événement.
`account`	chaîne	L'ID de compte AWS à 12 chiffres du propriétaire du magasin de données.
`time`	chaîne	Heure à laquelle l'événement s'est produit.
`region`	chaîne	Identifie la AWS région du magasin de données.
`resources`	tableau (chaîne)	Un tableau JSON qui contient l'ARN du magasin de données.
`detail`	objet	Un objet JSON qui contient des informations sur l'événement.
`detail.datastoreId`	chaîne	ID de banque de données associé à l'événement de changement de statut.
`detail.datastoreName`	chaîne	Le nom du magasin de données.

Nom	Type	Description
`detail.datastoreTypeVersion`	chaîne	La version FHIR du magasin de données.
`detail.datastoreEndpoint`	chaîne	Le point de terminaison du magasin de données.

Importer des événements professionnels

Import Job Submitted

État - **`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

État - **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

État - **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

État - **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

État - **`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/"
        }
    }
}
```

Importer des événements professionnels : descriptions des métadonnées

Nom	Type	Description
`version`	chaîne	Version du schéma d' EventBridge événements.
`id`	chaîne	L'UUID version 4 généré pour chaque événement.
`detail-type`	chaîne	Type d'événement envoyé.

Nom	Type	Description
`source`	chaîne	Identifie le service qui a généré l'événement.
`account`	chaîne	L'ID de compte AWS à 12 chiffres du propriétaire du magasin de données.
`time`	chaîne	Heure à laquelle l'événement s'est produit.
`region`	chaîne	Identifie la AWS région du magasin de données.
`resources`	tableau (chaîne)	Un tableau JSON qui contient l'ARN du magasin de données.
`detail`	objet	Un objet JSON qui contient des informations sur l'événement.
`detail.jobId`	chaîne	ID de tâche d'importation associé à l'événement de changement de statut.
`detail.submitTime`	chaîne	Heure à laquelle la tâche d'importation a été soumise.
`detail.datastoreId`	chaîne	Le magasin de données qui a généré l'événement de changement de statut.
`detail.inputDataConfig`	chaîne	Le chemin du préfixe d'entrée pour le compartiment Amazon S3 qui contient les fichiers FHIR à importer.

Exporter les événements professionnels

Export Job Submitted

État - **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

État - **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

État - **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

État - **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

État - **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/"
        }
    }
}
```

Exporter les événements liés aux tâches : descriptions des métadonnées

Nom	Type	Description
`version`	chaîne	Version du schéma d' EventBridge événements.
`id`	chaîne	L'UUID version 4 généré pour chaque événement.
`detail-type`	chaîne	Type d'événement envoyé.
`source`	chaîne	Identifie le service qui a généré l'événement.
`account`	chaîne	L'ID de compte AWS à 12 chiffres du propriétaire du magasin de données.
`time`	chaîne	Heure à laquelle l'événement s'est produit.
`region`	chaîne	Identifie la AWS région du magasin de données.

Nom	Type	Description
`resources`	tableau (chaîne)	Un tableau JSON qui contient l'ARN du magasin de données.
`detail`	objet	Un objet JSON qui contient des informations sur l'événement.
`detail.jobId`	chaîne	ID de tâche d'exportation associé à l'événement de changement de statut.
`detail.submitTime`	chaîne	Heure à laquelle la tâche d'exportation a été soumise.
`detail.datastoreId`	chaîne	Le magasin de données qui a généré l'événement de changement de statut.
`detail.outputDataConfig`	chaîne	Le chemin du préfixe de sortie pour le compartiment Amazon S3 qui contient les fichiers FHIR à exporter.

Sécurité dans AWS HealthLake

La sécurité du cloud AWS est la priorité absolue. En tant que AWS client, vous bénéficiez d'un centre de données et d'une architecture réseau conçus pour répondre aux exigences des entreprises les plus sensibles en matière de sécurité.

La sécurité est une responsabilité partagée entre vous AWS et vous. Le modèle de responsabilité partagée décrit ceci comme la sécurité du cloud et la sécurité dans le cloud :

- La sécurité du cloud AWS est chargée de protéger l'infrastructure qui exécute AWS les services dans le AWS cloud. AWS vous fournit également des services que vous pouvez utiliser en toute sécurité. Third-party les auditeurs testent et vérifient régulièrement l'efficacité de notre sécurité dans le cadre des programmes de AWS conformité Programmes de de conformité. Pour en savoir plus sur les programmes de conformité applicables à HealthLake, consultez la section Services AWS concernés par programme de conformité .
- Sécurité dans le cloud — Votre responsabilité est déterminée par le AWS service que vous utilisez. Vous êtes également responsable d'autres facteurs, y compris de la sensibilité de vos données, des exigences de votre entreprise, ainsi que de la législation et de la réglementation applicables.

Cette documentation vous aide à comprendre comment appliquer le modèle de responsabilité partagée lors de son utilisation HealthLake. Les rubriques suivantes expliquent comment procéder à la configuration HealthLake pour atteindre vos objectifs de sécurité et de conformité. Vous apprendrez également à utiliser d'autres services AWS qui vous aident à surveiller et à sécuriser vos HealthLake ressources.

Rubriques

- Résilience dans AWS HealthLake

Protection des données dans AWS HealthLake

Le modèle de responsabilité AWS partagée Le modèle de s'applique à la protection des données dans AWS HealthLake. Comme décrit dans ce modèle, AWS est chargé de protéger l'infrastructure mondiale qui gère tous les AWS Cloud. La gestion du contrôle de votre contenu hébergé sur cette infrastructure relève de votre responsabilité. Vous êtes également responsable des tâches de configuration et de gestion de la sécurité des Services AWS que vous utilisez. Pour plus d'informations sur la confidentialité des données, consultez la FAQ sur la confidentialité des données et les . Pour plus d'informations sur la protection des données en Europe, consultez le Centre du règlement général sur la protection des données (RGPD).

À des fins de protection des données, nous vous recommandons de protéger les Compte AWS informations d'identification et de configurer les utilisateurs individuels avec AWS IAM Identity Center ou Gestion des identités et des accès AWS (IAM). Ainsi, chaque utilisateur se voit attribuer uniquement les autorisations nécessaires pour exécuter ses tâches. Nous vous recommandons également de sécuriser vos données comme indiqué ci-dessous :

- Utilisez l’authentification multifactorielle (MFA) avec chaque compte.
- SSL/TLS À utiliser pour communiquer avec AWS les ressources. Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Configurez l'API et la journalisation de l'activité des utilisateurs avec AWS CloudTrail. Pour plus d'informations sur l'utilisation des CloudTrail sentiers pour capturer AWS des activités, consultez la section Utilisation des CloudTrail sentiers dans le guide de AWS CloudTrail l'utilisateur.
- Utilisez des solutions de AWS chiffrement, ainsi que tous les contrôles de sécurité par défaut qu'ils contiennent Services AWS.
- Utilisez des services de sécurité gérés avancés tels qu’Amazon Macie, qui contribuent à la découverte et à la sécurisation des données sensibles stockées dans Amazon S3.
- Si vous avez besoin de modules cryptographiques validés par la norme FIPS 140-3 pour accéder AWS via une interface de ligne de commande ou une API, utilisez un point de terminaison FIPS. Pour plus d’informations sur les points de terminaison FIPS disponibles, consultez Norme FIPS (Federal Information Processing Standard) 140-3.

Nous vous recommandons fortement de ne jamais placer d’informations confidentielles ou sensibles, telles que les adresses e-mail de vos clients, dans des balises ou des champs de texte libre tels

que le champ Nom. Cela inclut lorsque vous travaillez avec HealthLake ou d'autres Services AWS utilisateurs de la console, de l'API ou AWS des SDK. AWS CLI Toutes les données que vous entrez dans des balises ou des champs de texte de forme libre utilisés pour les noms peuvent être utilisées à des fins de facturation ou dans les journaux de diagnostic. Si vous fournissez une adresse URL à un serveur externe, nous vous recommandons fortement de ne pas inclure d'informations d'identification dans l'adresse URL permettant de valider votre demande adressée à ce serveur.

Chiffrement chez REST pour AWS HealthLake

HealthLake fournit un chiffrement par défaut pour protéger les données sensibles des clients au repos en utilisant une clé AWS Key Management Service (AWS KMS) détenue par le service. Customer-managed Les clés KMS sont également prises en charge et sont requises pour importer et exporter des fichiers depuis un magasin de données. Pour en savoir plus sur Customer-managed KMS Key, consultez Amazon Key Management Service. Les clients peuvent choisir une clé KMS appartenant à AWS ou une clé Customer-managed KMS lors de la création d'un magasin de données. La configuration du chiffrement ne peut pas être modifiée une fois qu'un magasin de données a été créé. Si un magasin de données utilise une clé KMS appartenant à AWS, elle sera désignée comme telle AWS_OWNED_KMS_KEY et vous ne verrez pas la clé spécifique utilisée pour le chiffrement au repos.

Clé KMS détenue par AWS

HealthLake utilise ces clés par défaut pour chiffrer automatiquement les informations potentiellement sensibles telles que les données personnelles identifiables ou les données de santé privées (PHI) au repos. Les clés KMS détenues par AWS ne sont pas stockées dans votre compte. Elles font partie d'un ensemble de clés KMS qu'AWS possède et gère pour être utilisées dans plusieurs comptes AWS. Les services AWS peuvent utiliser les clés KMS détenues par AWS pour protéger vos données. Vous ne pouvez pas consulter, gérer, utiliser les clés KMS détenues par AWS, ni auditer leur utilisation. Cependant, vous n'avez pas besoin de travailler ou de modifier de programme pour protéger les clés qui chiffrent vos données.

Aucuns frais mensuels ni frais d'utilisation ne vous sont facturés si vous utilisez des clés KMS appartenant à AWS, et elles ne sont pas prises en compte dans les quotas AWS KMS de votre compte. Pour plus d'informations, consultez la section Clés détenues par AWS.

Clés KMS gérées par le client

HealthLake prend en charge l'utilisation d'une clé KMS symétrique gérée par le client que vous créez, détenez et gérez pour ajouter une deuxième couche de chiffrement au chiffrement existant détenu par AWS. Étant donné que vous avez le contrôle total de cette couche de chiffrement, vous pouvez effectuer les tâches suivantes :

- Établir et maintenir des politiques clés, des politiques IAM et des subventions
- Rotation des matériaux de chiffrement de clé
- Activation et désactivation des stratégies de clé
- Ajout de balises
- Création d’alias de clé
- Planification des clés pour la suppression

Vous pouvez également l' CloudTrail utiliser pour suivre les demandes HealthLake envoyées AWS KMS en votre nom. Des AWS KMS frais supplémentaires s'appliquent. Pour plus d'informations, consultez la section sur les clés détenues par le client.

Création d’une clé gérée par le client

Vous pouvez créer une clé symétrique gérée par le client à l'aide de l'AWS Management Console ou des AWS KMS API.

Suivez les étapes de création d'une clé symétrique gérée par le client dans le guide du développeur AWS Key Management Service.

Les stratégies de clés contrôlent l’accès à votre clé gérée par le client. Chaque clé gérée par le client doit avoir exactement une stratégie de clé, qui contient des instructions qui déterminent les personnes pouvant utiliser la clé et comment elles peuvent l’utiliser. Lorsque vous créez votre clé gérée par le client, vous pouvez spécifier une stratégie de clé. Pour plus d'informations, consultez la section Gestion de l'accès aux clés gérées par le client dans le guide du développeur AWS Key Management Service.

Pour utiliser votre clé gérée par le client avec vos HealthLake ressources, les CreateGrant opérations kms : doivent être autorisées dans la politique des clés. Cela ajoute une autorisation à une clé gérée par le client qui contrôle l'accès à une clé KMS spécifiée, ce qui donne à l'utilisateur l'accès aux opérations kms:grant requises. HealthLake Voir Utilisation des subventions pour plus d'informations.

Pour utiliser votre clé KMS gérée par le client avec vos HealthLake ressources, les opérations d'API suivantes doivent être autorisées dans la politique des clés :

- kms : CreateGrant ajoute des subventions à une clé KMS spécifique gérée par le client qui permet d'accéder aux opérations de subvention.
- kms : DescribeKey fournit les informations clés gérées par le client nécessaires à la validation de la clé. Cela est obligatoire pour toutes les opérations.
- kms : GenerateDataKey fournit un accès aux ressources de chiffrement au repos pour toutes les opérations d'écriture.
- KMS:Decrypt permet d'accéder aux opérations de lecture ou de recherche de ressources chiffrées.

Voici un exemple de déclaration de politique qui permet à un utilisateur de créer et d'interagir avec un magasin de AWS HealthLake données chiffré par cette clé :

```
"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"
            }
        }
    }
]
```

Autorisations IAM requises pour utiliser une clé KMS gérée par le client

Lors de la création d'un magasin de données dont le AWS KMS chiffrement est activé à l'aide d'une clé KMS gérée par le client, des autorisations sont requises pour la politique de clé et la politique IAM pour l'utilisateur ou le rôle qui crée le magasin de HealthLake données.

Vous pouvez utiliser la clé de ViaService condition kms : pour limiter l'utilisation de la clé KMS aux seules demandes provenant de HealthLake.

Pour plus d'informations sur les politiques clés, consultez la section Activation des politiques IAM dans le guide du développeur d'AWS Key Management Service.

L'utilisateur IAM, le rôle IAM ou le compte AWS qui crée vos référentiels doit disposer des autorisations kms :CreateGrant, kms : GenerateDataKey et kms : ainsi que DescribeKey des autorisations nécessaires. HealthLake

Comment HealthLake utilise les subventions dans AWS KMS

HealthLake nécessite une autorisation pour utiliser votre clé KMS gérée par le client. Lorsque vous créez un magasin de données chiffré à l'aide d'une clé KMS gérée par le client, vous HealthLake créez une subvention en votre nom en envoyant une CreateGrantdemande à AWS KMS. Les subventions dans AWS KMS sont utilisées pour donner HealthLake accès à une clé KMS dans un compte client.

Les subventions HealthLake créées en votre nom ne doivent pas être révoquées ou retirées. Si vous révoquez ou retirez l' HealthLake autorisation d'utiliser les clés AWS KMS de votre compte, vous HealthLake ne pouvez pas accéder à ces données, chiffrer les nouvelles ressources FHIR envoyées au magasin de données ou les déchiffrer lorsqu'elles sont extraites. Lorsque vous révoquez ou retirez une subvention pour HealthLake, le changement intervient immédiatement. Pour révoquer les droits d'accès, vous devez supprimer le magasin de données plutôt que de révoquer l'autorisation. Lorsqu'un magasin de données est supprimé, les HealthLake subventions sont annulées en votre nom.

Surveillance de vos clés de chiffrement pour HealthLake

Vous pouvez l'utiliser CloudTrail pour suivre les demandes HealthLake envoyées en votre AWS KMS nom lorsque vous utilisez une clé KMS gérée par le client. Les entrées du CloudTrail journal indiquent healthlake.amazonaws.com dans le champ UserAgent afin de distinguer clairement les demandes effectuées par. HealthLake

Les exemples suivants sont CloudTrail des événements pour CreateGrant GenerateDataKey, déchiffrer et surveiller les AWS KMS opérations appelées DescribeKey pour accéder HealthLake aux données chiffrées par votre clé gérée par le client.

Ce qui suit montre comment utiliser CreateGrant pour autoriser l'accès HealthLake à une clé KMS fournie par le client, ce qui HealthLake permet d'utiliser cette clé KMS pour chiffrer toutes les données client au repos.

Les utilisateurs ne sont pas tenus de créer leurs propres subventions. HealthLake crée une subvention en votre nom en envoyant une CreateGrant demande à AWS KMS. Les subventions AWS KMS sont utilisées pour donner HealthLake accès à une AWS KMS clé dans un compte client.

```
         {
"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"
}
```

Les exemples suivants montrent comment s' GenerateDataKey assurer que l'utilisateur dispose des autorisations nécessaires pour chiffrer les données avant de les stocker.

```
            {
    "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"
}
```

L'exemple suivant montre comment HealthLake appelle l'opération Decrypt pour utiliser la clé de données cryptée stockée afin d'accéder aux données cryptées.

```
         {
    "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"
}
```

L'exemple suivant montre comment HealthLake utiliser l' DescribeKey opération pour vérifier si la AWS KMS clé détenue par le AWS KMS client est dans un état utilisable et pour aider l'utilisateur à résoudre les problèmes si elle n'est pas fonctionnelle.

```
         {
    "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"
}
```


En savoir plus

Les ressources suivantes fournissent des informations supplémentaires sur le chiffrement des données au repos.

Pour plus d'informations sur les concepts de base d'AWS Key Management Service, consultez la AWS KMS documentation.

Pour plus d'informations sur les meilleures pratiques en matière de sécurité, AWS KMS consultez la documentation.

Chiffrement en transit pour AWS HealthLake

AWS HealthLake utilise le protocole TLS 1.2 pour chiffrer les données en transit via le point de terminaison public et via les services principaux.

Gestion des identités et des accès pour AWS HealthLake

Gestion des identités et des accès AWS (IAM) est un outil Service AWS qui permet à un administrateur de contrôler en toute sécurité l'accès aux AWS ressources. Les administrateurs IAM contrôlent qui peut être authentifié (connecté) et autorisé (autorisé) à utiliser HealthLake les ressources. IAM est un Service AWS outil que vous pouvez utiliser sans frais supplémentaires.

Rubriques

Public ciblé

La façon dont vous utilisez Gestion des identités et des accès AWS (IAM) varie en fonction de votre rôle :

- Utilisateur du service : demandez des autorisations à votre administrateur si vous ne pouvez pas accéder aux fonctionnalités (voir Résolution des problèmes AWS HealthLake d'identité et d'accès)
- Administrateur du service : déterminez l’accès des utilisateurs et soumettez les demandes d’autorisation (voir Comment AWS HealthLake fonctionne avec IAM)
- Administrateur IAM : rédigez des politiques pour gérer l’accès (voir Exemples de politiques basées sur l'identité pour AWS HealthLake)

Authentification par des identités

L'authentification est la façon dont vous vous connectez à AWS l'aide de vos informations d'identification. Vous devez être authentifié en tant qu'utilisateur IAM ou en assumant un rôle IAM. Utilisateur racine d'un compte AWS

Vous pouvez vous connecter en tant qu'identité fédérée à l'aide d'informations d'identification provenant d'une source d'identité telle que AWS IAM Identity Center (IAM Identity Center), d'une authentification unique ou d'informations d'identification. Google/Facebook Pour plus d'informations sur la connexion, consultez Connexion à votre Compte AWS dans le Guide de l'utilisateur Connexion à AWS .

Pour l'accès par programmation, AWS fournit un SDK et une CLI pour signer les demandes de manière cryptographique. Pour plus d'informations, consultez Signature AWS Version 4 pour les demandes d'API dans le Guide de l'utilisateur IAM.

Compte AWS utilisateur root

Lorsque vous créez un Compte AWS, vous commencez par une seule identité de connexion appelée utilisateur Compte AWS root qui dispose d'un accès complet à toutes Services AWS les ressources. Il est vivement déconseillé d'utiliser l'utilisateur racine pour vos tâches quotidiennes. Pour les tâches qui requièrent des informations d'identification de l'utilisateur racine, consultez Tâches qui requièrent les informations d'identification de l'utilisateur racine dans le Guide de l'utilisateur IAM.

Identité fédérée

Il est recommandé d'obliger les utilisateurs humains à utiliser la fédération avec un fournisseur d'identité pour accéder à Services AWS l'aide d'informations d'identification temporaires.

Une identité fédérée est un utilisateur provenant de l'annuaire de votre entreprise, de votre fournisseur d'identité Web ou Directory Service qui y accède à Services AWS l'aide d'informations d'identification provenant d'une source d'identité. Les identités fédérées assument des rôles qui fournissent des informations d'identification temporaires.

Pour une gestion des accès centralisée, nous vous recommandons d'utiliser AWS IAM Identity Center. Pour plus d'informations, consultez Qu'est-ce que IAM Identity Center ? dans le Guide de l'utilisateur AWS IAM Identity Center .

Utilisateurs et groupes IAM

Un utilisateur IAM est une identité qui dispose d'autorisations spécifiques pour une seule personne ou application. Nous vous recommandons d'utiliser ces informations d'identification temporaires au lieu des utilisateurs IAM avec des informations d'identification à long terme. Pour plus d'informations, voir Exiger des utilisateurs humains qu'ils utilisent la fédération avec un fournisseur d'identité pour accéder à AWS l'aide d'informations d'identification temporaires dans le guide de l'utilisateur IAM.

Les groupes IAM spécifient une collection d'utilisateurs IAM et permettent de gérer plus facilement les autorisations pour de grands ensembles d'utilisateurs. Pour plus d'informations, consultez Cas d'utilisation pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

Rôles IAM

Un rôle IAM est une identité dotée d'autorisations spécifiques qui fournit des informations d'identification temporaires. Vous pouvez assumer un rôle en passant d'un rôle d'utilisateur à un rôle IAM (console) ou en appelant une opération d' AWS API AWS CLI ou d'API. Pour plus d'informations, consultez Méthodes pour endosser un rôle dans le Guide de l'utilisateur IAM.

Les rôles IAM sont utiles pour l'accès des utilisateurs fédérés, les autorisations temporaires des utilisateurs IAM, les accès intercompte, les accès entre services et les applications exécutées sur Amazon EC2. Pour plus d'informations, consultez Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.

Gestion de l'accès à l'aide de politiques

Vous contrôlez l'accès en AWS créant des politiques et en les associant à AWS des identités ou à des ressources. Une politique définit les autorisations lorsqu'elles sont associées à une identité ou à une ressource. AWS évalue ces politiques lorsqu'un directeur fait une demande. La plupart des politiques sont stockées AWS sous forme de documents JSON. Pour plus d'informations les documents de politique JSON, consultez Vue d'ensemble des politiques JSON dans le Guide de l'utilisateur IAM.

À l'aide de politiques, les administrateurs précisent qui a accès à quoi en définissant quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

Par défaut, les utilisateurs et les rôles ne disposent d'aucune autorisation. Un administrateur IAM crée des politiques IAM et les ajoute aux rôles, que les utilisateurs peuvent ensuite assumer. Les politiques IAM définissent les autorisations quelle que soit la méthode que vous utilisez pour exécuter l'opération.

Politiques basées sur l’identité

Les stratégies basées sur l’identité sont des documents de stratégie d’autorisations JSON que vous attachez à une identité (utilisateur, groupe ou rôle). Ces politiques contrôlent les actions que peuvent exécuter ces identités, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l’identité, consultez Définition d’autorisations IAM personnalisées avec des politiques gérées par le client dans le Guide de l’utilisateur IAM.

Les politiques basées sur l’identité peuvent être des politiques intégrées (intégrées directement dans une seule identité) ou des politiques gérées (politiques autonomes associées à plusieurs identités). Pour découvrir comment choisir entre des politiques gérées et en ligne, consultez Choix entre les politiques gérées et les politiques en ligne dans le Guide de l’utilisateur IAM.

Politiques basées sur les ressources

Les politiques basées sur les ressources sont des documents de politique JSON que vous attachez à une ressource. Les exemples incluent les politiques de confiance de rôle IAM et les stratégies de compartiment Amazon S3. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l’accès à une ressource spécifique. Vous devez spécifier un principal dans une politique basée sur les ressources.

Les politiques basées sur les ressources sont des politiques en ligne situées dans ce service. Vous ne pouvez pas utiliser les politiques AWS gérées par IAM dans une stratégie basée sur les ressources.

Autres types de politique

AWS prend en charge des types de politiques supplémentaires qui peuvent définir les autorisations maximales accordées par les types de politiques les plus courants :

- Limites d’autorisations : une limite des autorisations définit le nombre maximum d’autorisations qu’une politique basée sur l’identité peut accorder à une entité IAM. Pour plus d’informations, consultez Limites d’autorisations pour des entités IAM dans le Guide de l’utilisateur IAM.
- Politiques de contrôle des services (SCPs) — Spécifiez les autorisations maximales pour une organisation ou une unité organisationnelle dans AWS Organizations. Pour plus d’informations, consultez Politiques de contrôle de service dans le Guide de l’utilisateur AWS Organizations .
- Politiques de contrôle des ressources (RCPs) : définissez le maximum d'autorisations disponibles pour les ressources de vos comptes. Pour plus d'informations, voir Politiques de contrôle des ressources (RCPs) dans le guide de AWS Organizations l'utilisateur.

- Politiques de session : politiques avancées que vous passez en tant que paramètre lorsque vous créez par programmation une session temporaire pour un rôle ou un utilisateur fédéré. Pour plus d'informations, consultez Politiques de session dans le Guide de l'utilisateur IAM.

Plusieurs types de politique

Lorsque plusieurs types de politiques s'appliquent à la requête, les autorisations en résultant sont plus compliquées à comprendre. Pour savoir comment AWS déterminer s'il faut autoriser une demande lorsque plusieurs types de politiques sont impliqués, consultez la section Logique d'évaluation des politiques dans le guide de l'utilisateur IAM.

Comment AWS HealthLake fonctionne avec IAM

Avant d'utiliser IAM pour gérer l'accès à HealthLake, découvrez les fonctionnalités IAM disponibles. HealthLake

Fonctionnalités IAM que vous pouvez utiliser avec AWS HealthLake

Fonctionnalité IAM	HealthLake soutien
Politiques basées sur l'identité	Oui
Politiques basées sur les ressources	Non
Actions de politique	Oui
Ressources de politique	Oui
Clés de condition de politique	Oui
ACLs	Non
ABAC (étiquettes dans les politiques)	Oui
Informations d'identification temporaires	Oui
Autorisations de principal	Oui
Rôles de service	Oui
Rôles liés à un service	Non

Pour obtenir une vue d'ensemble de la façon dont HealthLake les autres AWS services fonctionnent avec la plupart des fonctionnalités IAM, consultez la section AWS Services compatibles avec IAM dans le Guide de l'utilisateur IAM.

Politiques basées sur l'identité pour AWS HealthLake

Prend en charge les politiques basées sur l’identité : oui

Les politiques basées sur l’identité sont des documents de politique d’autorisations JSON que vous pouvez attacher à une identité telle qu’un utilisateur, un groupe d’utilisateurs ou un rôle IAM. Ces politiques contrôlent quel type d’actions des utilisateurs et des rôles peuvent exécuter, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l’identité, consultez Définition d’autorisations IAM personnalisées avec des politiques gérées par le client dans le Guide de l’utilisateur IAM.

Avec les politiques IAM basées sur l’identité, vous pouvez spécifier des actions et ressources autorisées ou refusées, ainsi que les conditions dans lesquelles les actions sont autorisées ou refusées. Pour découvrir tous les éléments que vous utilisez dans une politique JSON, consultez Références des éléments de politique JSON IAM dans le Guide de l’utilisateur IAM.

Exemples de politiques basées sur l'identité pour AWS HealthLake

Pour consulter des exemples de politiques HealthLake basées sur l'identité, consultez. Exemples de politiques basées sur l'identité pour AWS HealthLake

Politiques basées sur les ressources au sein de AWS HealthLake

Prend en charge les politiques basées sur les ressources : non

Les politiques basées sur les ressources sont des documents de politique JSON que vous attachez à une ressource. Par exemple, les politiques de confiance de rôle IAM et les politiques de compartiment Amazon S3 sont des politiques basées sur les ressources. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l’accès à une ressource spécifique. Pour la ressource dans laquelle se trouve la politique, cette dernière définit quel type d’actions un principal spécifié peut effectuer sur cette ressource et dans quelles conditions. Vous devez spécifier un principal dans une politique basée sur les ressources. Les principaux peuvent inclure des comptes, des utilisateurs, des rôles, des utilisateurs fédérés ou. Services AWS

Pour permettre un accès intercompte, vous pouvez spécifier un compte entier ou des entités IAM dans un autre compte en tant que principal dans une politique basée sur les ressources. Pour plus d'informations, consultez Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.

Actions politiques pour AWS HealthLake

Prend en charge les actions de politique : oui

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Action` d'une politique JSON décrit les actions que vous pouvez utiliser pour autoriser ou refuser l'accès à une politique. Intégration d'actions dans une politique afin d'accorder l'autorisation d'exécuter les opérations associées.

Pour consulter la liste des HealthLake actions, reportez-vous à la section Actions définies par AWS HealthLake dans la référence d'autorisation de service.

Les actions de politique en HealthLake cours utilisent le préfixe suivant avant l'action :

```
healthlake
```

Pour spécifier plusieurs actions dans une seule instruction, séparez-les par une virgule.

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

Pour consulter des exemples de politiques HealthLake basées sur l'identité, consultez. Exemples de politiques basées sur l'identité pour AWS HealthLake

Ressources politiques pour AWS HealthLake

Prend en charge les ressources de politique : oui

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément de politique JSON `Resource` indique le ou les objets auxquels l'action s'applique. Il est recommandé de définir une ressource à l'aide de son Amazon Resource Name (ARN). Pour les actions qui ne sont pas compatibles avec les autorisations de niveau ressource, utilisez un caractère générique (*) afin d'indiquer que l'instruction s'applique à toutes les ressources.

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

Pour consulter la liste des types de HealthLake ressources et leurs caractéristiques ARNs, consultez la section Ressources définies par AWS HealthLake dans la référence d'autorisation de service. Pour connaître les actions permettant de spécifier l'ARN de chaque ressource, consultez la section Actions définies par AWS HealthLake.

Pour consulter des exemples de politiques HealthLake basées sur l'identité, consultez. Exemples de politiques basées sur l'identité pour AWS HealthLake

Clés de conditions de politique pour AWS HealthLake

Prend en charge les clés de condition de politique spécifiques au service : oui

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Condition` indique à quel moment les instructions s'exécutent en fonction de critères définis. Vous pouvez créer des expressions conditionnelles qui utilisent des opérateurs de condition, tels que les signes égal ou inférieur à, pour faire correspondre la condition de la politique aux valeurs de la demande. Pour voir toutes les clés de condition AWS globales, voir les clés de contexte de condition AWS globales dans le guide de l'utilisateur IAM.

Pour consulter la liste des clés de HealthLake condition, reportez-vous à la section Clés de condition pour AWS HealthLake la référence d'autorisation de service. Pour connaître les actions et les ressources avec lesquelles vous pouvez utiliser une clé de condition, consultez la section Actions définies par AWS HealthLake.

Pour consulter des exemples de politiques HealthLake basées sur l'identité, consultez. Exemples de politiques basées sur l'identité pour AWS HealthLake

Listes de contrôle d'accès (ACLs) dans AWS HealthLake

Supports ACLs : Non

Les listes de contrôle d'accès (ACLs) contrôlent les principaux (membres du compte, utilisateurs ou rôles) autorisés à accéder à une ressource. ACLs sont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format de document de politique JSON.

Contrôle d'accès basé sur les attributs (ABAC) avec AWS HealthLake

Prise en charge d'ABAC (balises dans les politiques) : Oui

Le contrôle d'accès par attributs (ABAC) est une stratégie d'autorisation qui définit les autorisations en fonction des attributs appelés balises. Vous pouvez associer des balises aux entités et aux AWS ressources IAM, puis concevoir des politiques ABAC pour autoriser les opérations lorsque la balise du principal correspond à la balise de la ressource.

Pour contrôler l'accès basé sur des étiquettes, vous devez fournir les informations d'étiquette dans l'élément de condition d'une politique utilisant les clés de condition `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* ou `aws:TagKeys`.

Si un service prend en charge les trois clés de condition pour tous les types de ressources, alors la valeur pour ce service est Oui. Si un service prend en charge les trois clés de condition pour certains types de ressources uniquement, la valeur est Partielle.

Pour plus d'informations sur ABAC, consultez Définition d'autorisations avec l'autorisation ABAC dans le Guide de l'utilisateur IAM. Pour accéder à un didacticiel décrivant les étapes de configuration de l'ABAC, consultez Utilisation du contrôle d'accès par attributs (ABAC) dans le Guide de l'utilisateur IAM.

Utilisation d'informations d'identification temporaires avec AWS HealthLake

Prend en charge les informations d'identification temporaires : oui

Les informations d'identification temporaires fournissent un accès à court terme aux AWS ressources et sont automatiquement créées lorsque vous utilisez la fédération ou que vous changez de rôle. AWS recommande de générer dynamiquement des informations d'identification temporaires au

lieu d'utiliser des clés d'accès à long terme. Pour plus d'informations, consultez Informations d'identification de sécurité temporaires dans IAM et Services AWS compatibles avec IAM dans le Guide de l'utilisateur IAM.

Autorisations principales interservices pour AWS HealthLake

Prend en charge les sessions d'accès direct (FAS) : oui

Les sessions d'accès direct (FAS) utilisent les autorisations du principal appelant et Service AWS, combinées Service AWS à la demande d'envoi de demandes aux services en aval. Pour plus de détails sur la politique relative à la transmission de demandes FAS, consultez la section Sessions de transmission d'accès.

Rôles de service pour AWS HealthLake

Prend en charge les rôles de service : oui

Un rôle de service est un rôle IAM qu'un service endosse pour accomplir des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer un rôle de service à partir d'IAM. Pour plus d'informations, consultez Création d'un rôle pour la délégation d'autorisations à un Service AWS dans le Guide de l'utilisateur IAM.

Pour plus d'informations sur les rôles de service et la politique en ligne requise pour un accès complet à AWS HealthLake, consultezConfiguration AWS HealthLake.

> **Warning**
>
> La modification des autorisations associées à un rôle de service peut perturber HealthLake les fonctionnalités. Modifiez les rôles de service uniquement lorsque HealthLake vous recevez des instructions à cet effet.

Rôles liés à un service pour AWS HealthLake

Prend en charge les rôles liés à un service : non

Un rôle lié à un service est un type de rôle de service lié à un. Service AWS Le service peut endosser le rôle afin d'effectuer une action en votre nom. Les rôles liés à un service apparaissent dans votre Compte AWS répertoire et appartiennent au service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.

Pour plus d'informations sur la création ou la gestion des rôles liés à un service, consultez Services AWS qui fonctionnent avec IAM. Recherchez un service dans le tableau qui inclut un `Yes` dans la colonne Rôle lié à un service. Choisissez le lien Oui pour consulter la documentation du rôle lié à ce service.

Exemples de politiques basées sur l'identité pour AWS HealthLake

Par défaut, les utilisateurs et les rôles ne sont pas autorisés à créer ou modifier les ressources HealthLake. Pour octroyer aux utilisateurs des autorisations d'effectuer des actions sur les ressources dont ils ont besoin, un administrateur IAM peut créer des politiques IAM.

Pour apprendre à créer une politique basée sur l'identité IAM à l'aide de ces exemples de documents de politique JSON, consultez Création de politiques IAM (console) dans le Guide de l'utilisateur IAM.

Pour plus de détails sur les actions et les types de ressources définis par HealthLake, y compris le format de ARNs pour chacun des types de ressources, voir Actions, ressources et clés de condition AWS HealthLake dans la référence d'autorisation de service.

Rubriques

Bonnes pratiques en matière de politiques

Les politiques basées sur l'identité déterminent si quelqu'un peut créer, accéder ou supprimer HealthLake des ressources dans votre compte. Ces actions peuvent entraîner des frais pour votre Compte AWS. Lorsque vous créez ou modifiez des politiques basées sur l'identité, suivez ces instructions et recommandations :

- Commencez AWS par les politiques gérées et passez aux autorisations du moindre privilège : pour commencer à accorder des autorisations à vos utilisateurs et à vos charges de travail, utilisez les politiques AWS gérées qui accordent des autorisations pour de nombreux cas d'utilisation courants. Ils sont disponibles dans votre Compte AWS. Nous vous recommandons de réduire davantage les autorisations en définissant des politiques gérées par les AWS clients spécifiques à vos cas d'utilisation. Pour plus d'informations, consultez politiques gérées par AWS ou politiques gérées par AWS pour les activités professionnelles dans le Guide de l'utilisateur IAM.

- Accordez les autorisations de moindre privilège : lorsque vous définissez des autorisations avec des politiques IAM, accordez uniquement les autorisations nécessaires à l'exécution d'une seule tâche. Pour ce faire, vous définissez les actions qui peuvent être entreprises sur des ressources spécifiques dans des conditions spécifiques, également appelées autorisations de moindre privilège. Pour plus d'informations sur l'utilisation d'IAM pour appliquer des autorisations, consultez politiques et autorisations dans IAM dans le Guide de l'utilisateur IAM.
- Utilisez des conditions dans les politiques IAM pour restreindre davantage l'accès : vous pouvez ajouter une condition à vos politiques afin de limiter l'accès aux actions et aux ressources. Par exemple, vous pouvez écrire une condition de politique pour spécifier que toutes les demandes doivent être envoyées via SSL. Vous pouvez également utiliser des conditions pour accorder l'accès aux actions de service si elles sont utilisées par le biais d'un service spécifique Service AWS, tel que CloudFormation. Pour plus d'informations, consultez Conditions pour éléments de politique JSON IAM dans le Guide de l'utilisateur IAM.
- Utilisez l'Analyseur d'accès IAM pour valider vos politiques IAM afin de garantir des autorisations sécurisées et fonctionnelles : l'Analyseur d'accès IAM valide les politiques nouvelles et existantes de manière à ce que les politiques IAM respectent le langage de politique IAM (JSON) et les bonnes pratiques IAM. IAM Access Analyzer fournit plus de 100 vérifications de politiques et des recommandations exploitables pour vous aider à créer des politiques sécurisées et fonctionnelles. Pour plus d'informations, consultez Validation de politiques avec IAM Access Analyzer dans le Guide de l'utilisateur IAM.
- Exiger l'authentification multifactorielle (MFA) : si vous avez un scénario qui nécessite des utilisateurs IAM ou un utilisateur root, activez l'authentification MFA pour une sécurité accrue. Compte AWS Pour exiger la MFA lorsque des opérations d'API sont appelées, ajoutez des conditions MFA à vos politiques. Pour plus d'informations, consultez Sécurisation de l'accès aux API avec MFA dans le Guide de l'utilisateur IAM.

Pour plus d'informations sur les bonnes pratiques dans IAM, consultez Bonnes pratiques de sécurité dans IAM dans le Guide de l'utilisateur IAM.

Utilisation de la AWS HealthLake console

Pour accéder à la AWS HealthLake console, vous devez disposer d'un ensemble minimal d'autorisations. Ces autorisations doivent vous permettre de répertorier et d'afficher les détails HealthLake des ressources de votre Compte AWS. Si vous créez une politique basée sur l'identité qui est plus restrictive que l'ensemble minimum d'autorisations requis, la console ne fonctionnera pas comme prévu pour les entités (utilisateurs ou rôles) tributaires de cette politique.

Il n'est pas nécessaire d'accorder des autorisations de console minimales aux utilisateurs qui appellent uniquement l'API AWS CLI ou l' AWS API. Autorisez plutôt l'accès à uniquement aux actions qui correspondent à l'opération d'API qu'ils tentent d'effectuer.

Pour un accès complet à HealthLake, associez les politiques suivantes à un utilisateur ou à un rôle IAM : `AmazonHealthLakeFullAccess` et`AWSLakeFormationDataAdmin`. Vous devez également associer la politique HealthLake intégrée qui est un rôle de service. Un rôle de service est un rôle IAM qu'un service endosse pour accomplir des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer un rôle de service à partir d'IAM. Pour plus d'informations, consultez Création d'un rôle pour la délégation d'autorisations à un Service AWS dans le Guide de l'utilisateur IAM. Pour plus d'informations sur la politique intégrée qui crée le rôle de service requis, consultezConfiguration AWS HealthLake. Vous devez également utiliser la AWS Lake Formation console ou la CLI pour désigner votre HealthLake administrateur comme administrateur de AWS Lake Formation Data Lake. Pour de plus amples informations, veuillez consulter Configuration AWS HealthLake.

Accès à un magasin AWS HealthLake de données dans Amazon Athena

Si vous souhaitez permettre aux utilisateurs et aux rôles d'accéder aux magasins de HealthLake données Amazon Athena, associez les politiques IAM suivantes au rôle ou à l'utilisateur : `AmazonAthenaFullAccess` et`AmazonS3FullAccess`. `Select`et `Describe` des autorisations sont également requises sur les tables gérées par AWS Lake Formation. AWS Lake Formation les autorisations de table sont accordées par un AWS Lake Formation administrateur dans la AWS Lake Formation console ou via la CLI. Pour de plus amples informations, consultez Configuration AWS HealthLake.

Autoriser des utilisateurs à afficher leurs propres autorisations

Cet exemple montre comment créer une politique qui permet aux utilisateurs IAM d'afficher les politiques en ligne et gérées attachées à leur identité d'utilisateur. Cette politique inclut les autorisations permettant d'effectuer cette action sur la console ou par programmation à l'aide de l'API AWS CLI or AWS .

```
{
    "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 politiques gérées pour AWS HealthLake

Une politique AWS gérée est une politique autonome créée et administrée par AWS. AWS les politiques gérées sont conçues pour fournir des autorisations pour de nombreux cas d'utilisation courants afin que vous puissiez commencer à attribuer des autorisations aux utilisateurs, aux groupes et aux rôles.

N'oubliez pas que les politiques AWS gérées peuvent ne pas accorder d'autorisations de moindre privilège pour vos cas d'utilisation spécifiques, car elles sont accessibles à tous les AWS clients. Nous vous recommandons de réduire encore les autorisations en définissant des politiques gérées par le client qui sont propres à vos cas d'utilisation.

Vous ne pouvez pas modifier les autorisations définies dans les politiques AWS gérées. Si les autorisations définies dans une politique AWS gérée sont AWS mises à jour, la mise à jour affecte

toutes les identités principales (utilisateurs, groupes et rôles) auxquelles la politique est attachée. AWS est le plus susceptible de mettre à jour une politique AWS gérée lorsqu'une nouvelle Service AWS est lancée ou lorsque de nouvelles opérations d'API sont disponibles pour les services existants.

Pour plus d'informations, consultez Politiques gérées par AWS dans le Guide de l'utilisateur IAM.

AWS politique gérée : AmazonHealthLakeFullAccess

La `AmazonHealthLakeFullAccess` politique fournit un accès complet à HealthLake. Cette politique étant attachée à leur utilisateur ou à leur rôle, les utilisateurs peuvent l'utiliser HealthLake pour accéder aux données, les interroger, les importer et les exporter HealthLake. Pour effectuer de nombreuses actions courantes dans HealthLake, vous devez ajouter des politiques supplémentaires à l'utilisateur ou au rôle. Pour plus d'informations, consultez la section HealthLake Opérations Configuration AWS HealthLake et autorisations.

Vous pouvez associer la politique `AmazonHealthLakeFullAccess` à vos identités IAM.

Cette politique accorde des autorisations d'administration et de contribution qui permettent aux utilisateurs et aux rôles d'effectuer des requêtes, de rechercher HealthLake, d'importer et d'exporter, et elle permet également d' HealthLake effectuer des actions au nom des utilisateurs et des rôles dotés de ces autorisations.

Détails de l'autorisation

Cette politique inclut la déclaration suivante.

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 politique gérée : AmazonHealthLakeReadOnlyAccess

`AmazonHealthLakeReadOnlyAccess`la politique accorde un accès et des autorisations en lecture seule HealthLake aux ressources associées dans d'autres AWS services. Appliquez cette politique aux utilisateurs auxquels vous souhaitez accorder la possibilité d'interroger et de consulter le magasin de HealthLake données, mais pas la possibilité de le créer ou d'y apporter des modifications.

Vous pouvez associer la politique `AmazonHealthLakeReadOnlyAccess` à vos identités IAM.

Cette politique accorde des *`read-only`* autorisations qui permettent aux utilisateurs et aux rôles d'effectuer des requêtes HealthLake.

Détails de l’autorisation

Cette politique inclut la déclaration suivante.

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 opérations et autorisations

Le tableau suivant répertorie les opérations typiques HealthLake et les autorisations nécessaires pour les exécuter.

HealthLake opérations	Autorisations requises
Créez un magasin de données dans HealthLake	`AmazonHealthLakeFullAccess` `AmazonLakeFormationDataAdmin` , politique intégrée et autorisations d' AWS Lake Formation administrateur gérées par AWS Lake Formation
Supprimer un magasin de données dans HealthLake	`AmazonHealthLakeFullAccess` `AmazonLakeFormatio`

HealthLake opérations	Autorisations requises
	`nDataAdmin` , politique intégrée et autorisations d' AWS Lake Formation administrateur gérées par AWS Lake Formation
Répertorier, rechercher ou interroger un magasin de données dans HealthLake	`AmazonHealthLakeReadOnlyAccess`
Interrogez un magasin de données à l'aide de Amazon Athena	`AmazonAthenaFullAccess` `AmazonS3FullAccess` , AWS Lake Formation `Select` et les `Describe` autorisations sur les tables gérées par AWS Lake Formation
Importer des données depuis HealthLake	Consultez Configuration des autorisations pour les tâches d'importation.
Exporter des données depuis HealthLake	Consultez Configuration des autorisations pour les tâches d'exportation.

HealthLake mises à jour des politiques AWS gérées

Consultez les détails des mises à jour des politiques AWS gérées HealthLake depuis le moment où ce service a commencé à suivre ces modifications. Pour recevoir des alertes automatiques concernant les modifications apportées à cette page, abonnez-vous au flux RSS sur la page Historique du HealthLake document.

Modifier	Description	Date
AmazonHealthLakeFullAccess	`AmazonHealthLakeFullAccess` politique requise pour permettre un accès complet à HealthLake.	14 novembre 2022
AmazonHealthLakeReadOnlyAccess	`AmazonHealthLakeReadOnlyAccess` politique requise pour l'accès en lecture seule à. HealthLake	14 novembre 2022

Modifier	Description	Date
HealthLake a commencé à suivre les modifications	HealthLake a commencé à suivre les modifications apportées AWS à ses politiques gérées.	14 novembre 2022

Résolution des problèmes AWS HealthLake d'identité et d'accès

Utilisez les informations suivantes pour vous aider à diagnostiquer et à résoudre les problèmes courants que vous pouvez rencontrer lorsque vous travaillez avec HealthLake IAM.

Rubriques

- Je ne suis pas autorisé à effectuer une action dans AWS HealthLake
- Je ne suis pas autorisé à effectuer iam : PassRole
- Je souhaite autoriser des personnes extérieures à mon AWS compte à accéder à mes AWS HealthLake ressources

Je ne suis pas autorisé à effectuer une action dans AWS HealthLake

S'il vous AWS Management Console indique que vous n'êtes pas autorisé à effectuer une action, vous devez contacter votre administrateur pour obtenir de l'aide. Votre administrateur est la personne qui vous a fourni votre nom d'utilisateur et votre mot de passe.

L'exemple d'erreur suivant se produit quand l'utilisateur IAM `mateojackson` tente d'utiliser la console pour afficher des informations détaillées sur une ressource *`my-example-widget`* fictive, mais ne dispose pas des autorisations `healthlake:`*`GetWidget`* fictives.

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

Dans ce cas, Mateo demande à son administrateur de mettre à jour ses politiques pour lui permettre d'accéder à la ressource *`my-example-widget`* à l'aide de l'action `healthlake:`*`GetWidget`*.

Je ne suis pas autorisé à effectuer iam : PassRole

Si vous recevez une erreur selon laquelle vous n'êtes pas autorisé à exécuter `iam:PassRole` l'action, vos stratégies doivent être mises à jour afin de vous permettre de transmettre un rôle à HealthLake.

Certains vous Services AWS permettent de transmettre un rôle existant à ce service au lieu de créer un nouveau rôle de service ou un rôle lié à un service. Pour ce faire, vous devez disposer des autorisations nécessaires pour transmettre le rôle au service.

L'exemple d'erreur suivant se produit lorsqu'un utilisateur IAM nommé `marymajor` essaie d'utiliser la console pour exécuter une action dans HealthLake. Toutefois, l'action nécessite que le service ait des autorisations accordées par une fonction de service. Mary n'est pas autorisée à transmettre le rôle au service.

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

Dans ce cas, les politiques de Mary doivent être mises à jour pour lui permettre d'exécuter l'action `iam:PassRole`.

Si vous avez besoin d'aide, contactez votre AWS administrateur. Votre administrateur vous a fourni vos informations d'identification de connexion.

Je souhaite autoriser des personnes extérieures à mon AWS compte à accéder à mes AWS HealthLake ressources

Vous pouvez créer un rôle que les utilisateurs provenant d'autres comptes ou les personnes extérieures à votre organisation pourront utiliser pour accéder à vos ressources. Vous pouvez spécifier qui est autorisé à assumer le rôle. Pour les services qui prennent en charge les politiques basées sur les ressources ou les listes de contrôle d'accès (ACLs), vous pouvez utiliser ces politiques pour autoriser les utilisateurs à accéder à vos ressources.

Pour plus d'informations, consultez les éléments suivants :

- Pour savoir si ces fonctionnalités sont prises HealthLake en charge, consultezComment AWS HealthLake fonctionne avec IAM.
- Pour savoir comment fournir l'accès à vos ressources sur celles Comptes AWS que vous possédez, consultez la section Fournir l'accès à un utilisateur IAM dans un autre utilisateur Compte AWS que vous possédez dans le Guide de l'utilisateur IAM.

- Pour savoir comment fournir l'accès à vos ressources à des tiers Comptes AWS, consultez la section Fournir un accès à des ressources Comptes AWS détenues par des tiers dans le guide de l'utilisateur IAM.
- Pour savoir comment fournir un accès par le biais de la fédération d'identité, consultez Fournir un accès à des utilisateurs authentifiés en externe (fédération d'identité) dans le Guide de l'utilisateur IAM.
- Pour en savoir plus sur la différence entre l'utilisation des rôles et des politiques basées sur les ressources pour l'accès intercompte, consultez Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.

Validation de conformité pour AWS HealthLake

Third-party les auditeurs évaluent la sécurité et la conformité dans AWS HealthLake le cadre de multiples programmes de AWS conformité. Car HealthLake cela inclut la loi HIPAA.

Pour savoir si un programme Services AWS de conformité Service AWS s'inscrit dans le champ d'application de programmes de conformité spécifiques, consultez Services AWS la section de conformité et sélectionnez le programme de conformité qui vous intéresse. Pour des informations générales, voir Programmes de AWS conformité Programmes AWS de .

Vous pouvez télécharger des rapports d'audit tiers à l'aide de AWS Artifact. Pour plus d'informations, voir Téléchargement de rapports dans AWS Artifact .

Votre responsabilité en matière de conformité lors de l'utilisation Services AWS est déterminée par la sensibilité de vos données, les objectifs de conformité de votre entreprise et les lois et réglementations applicables. Pour plus d'informations sur votre responsabilité en matière de conformité lors de l'utilisation Services AWS, consultez AWS la documentation de sécurité.

Sécurité de l'infrastructure dans AWS HealthLake

En tant que service géré, AWS HealthLake il est protégé par les procédures de sécurité du réseau AWS mondial décrites dans le livre blanc Amazon Web Services : présentation des processus de sécurité.

Vous utilisez des appels d'API AWS publiés pour accéder HealthLake via le réseau. Les clients doivent supporter le protocole TLS (Sécurité de la couche transport) 1.0 ou une version ultérieure. Nous vous recommandons le certificat TLS 1.2 ou une version ultérieure. Les clients doivent

également prendre en charge les suites de chiffrement à parfaite confidentialité (PFS), telles que Ephemeral (DHE) ou Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Diffie-Hellman La plupart des systèmes modernes tels que Java 7 et les versions ultérieures prennent en charge ces modes.

En outre, les demandes doivent être signées à l'aide d'un ID de clé d'accès et d'une clé d'accès secrète associée à un principal IAM. Vous pouvez également utiliser AWS Security Token Service (AWS STS) pour générer des informations d'identification de sécurité temporaires et signer les demandes.

Création de AWS HealthLake ressources avec AWS CloudFormation

AWS HealthLake est intégré à AWS CloudFormation un service qui vous aide à modéliser et à configurer vos AWS ressources afin que vous puissiez passer moins de temps à créer et à gérer vos ressources et votre infrastructure. Vous créez un modèle qui décrit toutes les AWS ressources que vous souhaitez et qui CloudFormation fournit et configure ces ressources pour vous.

Lorsque vous l'utilisez CloudFormation, vous pouvez réutiliser votre modèle pour configurer vos HealthLake ressources de manière cohérente et répétée. Décrivez vos ressources une seule fois, puis distribuez les mêmes ressources encore et encore dans plusieurs Comptes AWS régions.

HealthLake et CloudFormation modèles

Pour fournir et configurer des ressources HealthLake et des services associés, vous devez comprendre les CloudFormation modèles. Les modèles sont des fichiers texte formatés en JSON ou YAML. Ces modèles décrivent les ressources que vous souhaitez mettre à disposition dans vos CloudFormation piles. Si vous n'êtes pas familiarisé avec JSON ou YAML, vous pouvez utiliser CloudFormation Designer pour vous aider à démarrer avec les CloudFormation modèles. Pour plus d'informations, consultez Qu'est-ce que CloudFormation Designer dans le Guide de l'utilisateur AWS CloudFormation .

> **Note**
>
> AWS HealthLake prend en charge la création de magasins de données avec CloudFormation. Pour plus d'informations, notamment des exemples de modèles JSON et YAML pour le provisionnement de magasins de HealthLake données, consultez la référence au type de AWS HealthLake ressource dans le guide de l'AWS CloudFormation utilisateur.

En savoir plus sur CloudFormation

Pour en savoir plus CloudFormation, consultez les ressources suivantes :

- AWS CloudFormation
- AWS CloudFormation Guide de l'utilisateur
- Référence de l’API CloudFormation
- AWS CloudFormation Guide de l'utilisateur de l'interface de ligne de commande

AWS HealthLake et points de terminaison VPC d'interface ()AWS PrivateLink

Vous pouvez établir une connexion privée entre votre VPC et créer un point de AWS HealthLake terminaison VPC d'interface. Les points de terminaison VPC d'interface sont AWS PrivateLinkalimentés par une technologie à laquelle vous pouvez accéder en privé HealthLake, APIs sans passerelle Internet, périphérique NAT, connexion VPN ou connexion. Direct Connect Les instances de votre VPC n'ont pas besoin d'adresses IP publiques pour communiquer avec ; HealthLake APIs Le trafic entre votre VPC et HealthLake ; ne quitte pas le réseau Amazon.

Chaque point de terminaison d’interface est représenté par une ou plusieurs interfaces réseau Elastic dans vos sous-réseaux.

Pour plus d'informations, consultez la section Interface VPC endpoints (AWS PrivateLink) dans le guide de l'utilisateur Amazon VPC.

Considérations relatives aux points de HealthLake terminaison VPC

Avant de configurer un point de terminaison VPC d'interface pour HealthLake, assurez-vous de consulter les propriétés et les limites du point de terminaison d'interface dans le guide de l'utilisateur Amazon VPC.

HealthLake permet d'appeler toutes ses actions d'API depuis votre VPC.

Création d'un point de terminaison VPC d'interface pour ; HealthLake

Vous pouvez créer un point de terminaison VPC pour le service HealthLake ; à l'aide de la console Amazon VPC ou du (). AWS Command Line Interface AWS CLI Pour plus d’informations, consultez Création d’un point de terminaison d’interface dans le Guide de l’utilisateur Amazon VPC.

Créez un point de terminaison VPC pour HealthLake ; en utilisant le nom de service suivant :

- com.amazonaws. *region*.healthlake

Si vous activez le DNS privé pour le point de terminaison, vous pouvez envoyer des demandes d'API HealthLake en utilisant son nom DNS par défaut pour la région. Par exemple, *healthlake.us-east-1.amazonaws.com*.

Pour plus d'informations, consultez Accès à un service via un point de terminaison d'interface dans le Guide de l'utilisateur Amazon VPC.

Création d'une politique de point de terminaison VPC pour HealthLake

Vous pouvez attacher une stratégie de point de terminaison à votre point de terminaison d'un VPC qui contrôle l'accès à HealthLake. La politique spécifie les informations suivantes :

- Le principal qui peut exécuter des actions.
- Les actions qui peuvent être effectuées.
- Les ressources sur lesquelles les actions peuvent être exécutées.

Pour plus d'informations, consultez Contrôle de l'accès aux services avec points de terminaison d'un VPC dans le Guide de l'utilisateur Amazon VPC.

Exemple : politique de point de terminaison VPC pour les actions HealthLake

Voici un exemple de politique de point de terminaison pour HealthLake. Lorsqu'elle est attachée à un point de terminaison, cette politique accorde l'accès à l' HealthLake`CreateFHIRDatastore`action à tous les principaux sur toutes les ressources.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "healthlake:create-fhir-datastore"
         ],
         "Resource":"*"
      }
   ]
```

```
}
```

Bonnes pratiques en matière de sécurité dans AWS HealthLake

AWS HealthLake fournit un certain nombre de fonctionnalités de sécurité à prendre en compte lors de l'élaboration et de la mise en œuvre de vos propres politiques de sécurité. Les bonnes pratiques suivantes doivent être considérées comme des instructions générales et ne représentent pas une solution de sécurité complète. Étant donné que ces bonnes pratiques peuvent ne pas être appropriées ou suffisantes pour votre environnement, considérez-les comme des remarques utiles plutôt que comme des recommandations.

- Mettez en œuvre l'accès avec le moindre privilège.
- Dans la mesure du possible, utilisez Customer-Managed-Keys (CMK) pour chiffrer vos données. Pour en savoir plus sur les CMK, consultez Amazon Key Management Service.
- Utilisez la recherche avec POST, et non la recherche avec GET lorsque vous recherchez des PHI ou des informations personnelles dans votre magasin de données.
- Limitez l'accès aux fonctions d'audit sensibles et importantes.
- Lorsque vous créez des ressources via les API de mise à jour ou d'importation en masse, n'utilisez pas de PHI ou de PII, y compris les noms des banques de données et des tâches, dans les champs visibles ou dans l'identifiant FHIR logique (LID).
- Lorsque vous envoyez des demandes de création, de lecture, de mise à jour, de suppression ou de recherche, n'utilisez pas PHI dans l'en-tête HTTP.
- Activez cette option AWS CloudTrail pour auditer AWS HealthLake l'utilisation et vous assurer qu'il n'y a aucune activité inattendue.
- Consultez les meilleures pratiques pour utiliser les compartiments Amazon S3 en toute sécurité. Pour en savoir plus, consultez les meilleures pratiques en matière de sécurité dans le guide de l'utilisateur d'Amazon S3.

Résilience dans AWS HealthLake

L'infrastructure AWS mondiale est construite autour des AWS régions et des zones de disponibilité. AWS Les régions fournissent plusieurs zones de disponibilité physiquement séparées et isolées, connectées par un réseau à faible latence, à haut débit et hautement redondant. Avec les zones de disponibilité, vous pouvez concevoir et exploiter des applications et des bases de données qui basculent automatiquement d’une zone à l’autre sans interruption. Les zones de disponibilité sont

davantage disponibles, tolérantes aux pannes et ont une plus grande capacité de mise à l'échelle que les infrastructures traditionnelles à un ou plusieurs centres de données.

Si vous avez besoin de répliquer vos données ou applications sur des distances géographiques plus importantes, utilisez des régions locales AWS . Une région AWS locale est un centre de données unique conçu pour compléter une AWS région existante. Comme toutes les AWS régions, les régions AWS locales sont complètement isolées des autres AWS régions.

Pour plus d'informations sur AWS les régions et les zones de disponibilité, consultez la section Infrastructure AWS mondiale.

AWS HealthLake référence

Les documents de référence complémentaires suivants sont disponibles pour SMART sur FHIR, FHIR et. AWS HealthLake

> **Note**
>
> Toutes les HealthLake actions natives et tous les types de données sont décrits dans une référence séparée. Pour plus d'informations, consultez la Référence des API AWS HealthLake .

Rubriques

SMART sur le support FHIR pour AWS HealthLake

Un magasin de HealthLake données compatible avec les applications médicales substituables et les technologies réutilisables (SMART) sur FHIR permet d'accéder aux applications compatibles SMART sur FHIR. HealthLake les données sont accessibles en authentifiant et en autorisant les demandes à l'aide d'un serveur d'autorisation tiers. Ainsi, au lieu de gérer les informations d'identification des utilisateurs via Gestion des identités et des accès AWS, vous le faites à l'aide d'un serveur d'autorisation compatible SMART on FHIR.

> **Note**
>
> HealthLake supporte SMART sur les versions 1.0 et 2.0 de FHIR. Pour en savoir plus sur ces frameworks, consultez SMART App Launch dans la documentation FHIR R4.
> HealthLake les magasins de données prennent en charge les cadres d'authentification et d'autorisation suivants pour les demandes SMART on FHIR :
>
> - OpenID (AuthN) : pour authentifier la personne ou l'application cliente, c'est la personne (ou ce que) elle prétend être.

- OAuth 2.0 (AuthZ) : pour autoriser les ressources FHIR de votre magasin de HealthLake données sur lesquelles une demande authentifiée peut lire ou écrire. Ceci est défini par les étendues configurées sur votre serveur d'autorisation.

Vous pouvez créer un magasin de données compatible SMART sur FHIR à l'aide des AWS SDK AWS CLI ou. Pour de plus amples informations, veuillez consulter Création d'un magasin HealthLake de données.

Après avoir créé un magasin de données compatible SMART on FHIR, vous pouvez mettre à jour la configuration de son fournisseur d'identité, y compris les métadonnées du serveur d'autorisation, la fonction Lambda de validation des jetons et la stratégie d'autorisation, en utilisant. `UpdateFHIRDatastore` Pour de plus amples informations, veuillez consulter Mettre à jour un magasin HealthLake de données.

Rubriques

Commencer à utiliser SMART sur FHIR

Les rubriques suivantes décrivent comment démarrer avec SMART sur l'autorisation FHIR pour AWS HealthLake. Ils incluent les ressources que vous devez fournir sur votre AWS compte, la création d'un magasin de HealthLake données compatible SMART on FHIR et un exemple de la façon dont une application client SMART on FHIR interagit avec un serveur d'autorisation et un HealthLake magasin de données.

Rubriques

- Configuration des ressources pour SMART sur FHIR
- Flux de travail des applications clientes pour SMART sur FHIR

Configuration des ressources pour SMART sur FHIR

Les étapes suivantes définissent la manière dont les demandes SMART on FHIR sont traitées HealthLake et les ressources nécessaires pour qu'elles aboutissent. Les éléments suivants fonctionnent ensemble dans un flux de travail pour effectuer une demande SMART on FHIR :

- L'utilisateur final : en général, un patient ou un clinicien utilisant une application SMART on FHIR tierce pour accéder aux données d'un HealthLake magasin de données.
- L'application SMART on FHIR (appelée application client) : application qui souhaite accéder aux données présentes dans le magasin de HealthLake données.
- Le serveur d'autorisation : un serveur compatible OpenID Connect capable d'authentifier les utilisateurs et d'émettre des jetons d'accès.
- Le magasin de HealthLake données : un magasin de HealthLake données compatible SMART on FHIR qui utilise une fonction Lambda pour répondre aux requêtes REST FHIR qui fournissent un jeton porteur.

Pour que ces éléments fonctionnent ensemble, vous devez créer les ressources suivantes.

> **Note**
>
> Nous vous recommandons de créer votre banque de HealthLake données compatible SMART on FHIR après avoir configuré le serveur d'autorisation, défini les étendues nécessaires et créé une AWS Lambda fonction pour gérer l'introspection des jetons.

1. Configuration d'un point de terminaison de serveur d'autorisation

Pour utiliser le framework SMART on FHIR, vous devez configurer un serveur d'autorisation tiers capable de valider les requêtes REST FHIR effectuées sur un magasin de données. Pour de plus amples informations, veuillez consulter HealthLake exigences d'authentification pour SMART sur FHIR.

2. Définissez des étendues sur votre serveur d'autorisation pour contrôler les niveaux d'accès au magasin de HealthLake données

Le framework SMART on FHIR utilise des étendues OAuth pour déterminer à quelles ressources FHIR une demande authentifiée a accès et dans quelle mesure. La définition des étendues est un moyen de concevoir avec le moindre privilège. Pour de plus amples informations, veuillez consulter SMART sur les oscilloscopes FHIR OAuth 2.0 pris en charge par HealthLake.

3. Configurez un AWS Lambda fonction capable d'effectuer une introspection symbolique

Une demande REST FHIR envoyée par l'application cliente sur un magasin de données compatible SMART on FHIR contient un jeton Web JSON (JWT). Pour plus d'informations, consultez la section Décodage d'un JWT.

4. Créez un magasin de HealthLake données compatible SMART on FHIR

Pour créer un magasin de HealthLake données SMART sur FHIR, vous devez fournir un`IdentityProviderConfiguration`. Pour de plus amples informations, veuillez consulter Création d'un magasin HealthLake de données.

> **Note**
>
> Une fois le magasin de données créé, vous pouvez mettre à jour la configuration du fournisseur d'identité, par exemple pour faire pivoter la fonction Lambda de validation des jetons, modifier les métadonnées du serveur d'autorisation ou modifier la stratégie d'autorisation, en utilisant. `UpdateFHIRDatastore` Pour de plus amples informations, veuillez consulter Mettre à jour un magasin HealthLake de données.

Flux de travail des applications clientes pour SMART sur FHIR

La section suivante explique comment lancer une application cliente et effectuer une requête REST FHIR réussie sur un magasin de HealthLake données dans le contexte de SMART on FHIR.

1. Envoyer une requête **`GET`** à Well-Known Uniform Resource Identifier à l'aide de l'application cliente

Une application cliente compatible SMART doit effectuer une `GET` demande pour trouver les points de terminaison d'autorisation de votre magasin de HealthLake données. Cela se fait via une demande d'identifiant de ressource Well-Known uniforme (URI). Pour de plus amples informations, veuillez consulter Récupération du document SMART sur FHIR Discovery.

2. Demande d'accès et champs d'application

L'application cliente utilise le point de terminaison d'autorisation du serveur d'autorisation afin que l'utilisateur puisse se connecter. Ce processus authentifie l'utilisateur. Les étendues sont utilisées pour définir les ressources FHIR de votre magasin de HealthLake données auxquelles une application cliente peut accéder. Pour de plus amples informations, veuillez consulter SMART sur les oscilloscopes FHIR OAuth 2.0 pris en charge par HealthLake.

3. Jetons d'accès

Maintenant que l'utilisateur est authentifié, une application cliente reçoit un jeton d'accès JWT du serveur d'autorisation. Ce jeton est fourni lorsque l'application cliente envoie une demande FHIR REST à HealthLake. Pour de plus amples informations, veuillez consulter Validation des jetons.

4. Effectuer une demande d'API REST FHIR sur SMART sur un magasin de données activé HealthLake par FHIR

L'application cliente peut désormais envoyer une demande d'API REST FHIR à un point de terminaison du magasin de HealthLake données à l'aide du jeton d'accès fourni par le serveur d'autorisation. Pour de plus amples informations, veuillez consulter Effectuer une demande d'API REST FHIR sur un magasin de données compatible SMART HealthLake.

5. Validez le jeton d'accès JWT

Pour valider le jeton d'accès envoyé dans la requête REST FHIR, utilisez une fonction Lambda. Pour de plus amples informations, veuillez consulter Validation des jetons en utilisant AWS Lambda.

HealthLake exigences d'authentification pour SMART sur FHIR

Pour accéder aux ressources FHIR dans un magasin de FHIR-enabled HealthLake données SMART on, une application cliente doit être autorisée par un serveur d'autorisation conforme à OAuth 2.0 et présenter un jeton OAuth Bearer dans le cadre d'une demande d'API REST FHIR. Pour trouver le point de terminaison du serveur d'autorisation, utilisez le document HealthLake SMART on FHIR Discovery via un identifiant de ressource `Well-Known` uniforme. Pour en savoir plus sur ce processus, consultez Récupération du document SMART sur FHIR Discovery.

Lorsque vous créez un magasin de HealthLake données SMART on FHIR, vous devez définir le point de terminaison du serveur d'autorisation et le point de terminaison du jeton dans l'`metadata`élément de la `CreateFHIRDatastore` demande. Pour en savoir plus sur la définition de l'`metadata`élément, voirCréation d'un magasin HealthLake de données.

Vous pouvez également mettre à jour l'`metadata`élément (y compris les points de terminaison d'autorisation et de jeton) sur un magasin de données existant en utilisant`UpdateFHIRDatastore`.

Pour de plus amples informations, veuillez consulter Mettre à jour un magasin HealthLake de données.

À l'aide des points de terminaison du serveur d'autorisation, l'application cliente authentifie un utilisateur auprès du service d'autorisation. Une fois autorisé et authentifié, un jeton Web JSON (JWT) est généré par le service d'autorisation et transmis à l'application cliente. Ce jeton contient des étendues de ressources FHIR que l'application cliente est autorisée à utiliser, ce qui limite les données auxquelles l'utilisateur peut accéder. Facultativement, si le périmètre de lancement a été fourni, la réponse contiendra ces détails. Pour en savoir plus sur les oscilloscopes SMART on FHIR pris en charge par HealthLake, voir. SMART sur les oscilloscopes FHIR OAuth 2.0 pris en charge par HealthLake

À l'aide du JWT accordé par le serveur d'autorisation, une application cliente effectue des appels d'API REST FHIR à un magasin de données compatible HealthLake SMART sur FHIR. Pour valider et décoder le JWT, vous devez créer une fonction Lambda. HealthLake invoque cette fonction Lambda en votre nom lorsqu'une demande d'API REST FHIR est reçue. Pour voir un exemple de fonction Lambda de démarrage, reportez-vous à. Validation des jetons en utilisant AWS Lambda

Éléments du serveur d'autorisation requis pour créer un magasin de HealthLake données compatible SMART on FHIR

Dans la `CreateFHIRDatastore` demande, vous devez fournir le point de terminaison d'autorisation et le point de terminaison du jeton dans le cadre de l'`metadata`élément de l'`IdentityProviderConfiguration`objet. Le point de terminaison d'autorisation et le point de terminaison jeton sont tous deux requis. Pour voir un exemple de la manière dont cela est spécifié dans la `CreateFHIRDatastore` demande, voirCréation d'un magasin HealthLake de données.

Réclamations requises pour compléter une demande d'API REST FHIR sur un magasin de données compatible HealthLake SMART on FHIR

Votre AWS Lambda fonction doit contenir les affirmations suivantes pour qu'il s'agisse d'une demande d'API REST FHIR valide sur un magasin de HealthLake données compatible SMART on FHIR.

- `nbf`: (Pas avant) Réclamation — La réclamation « nbf » (pas avant) indique le délai avant lequel le JWT NE DOIT PAS être accepté pour traitement. Le traitement de la réclamation « nbf » nécessite que le courant date/time DOIT être supérieur ou égal à la valeur non date/time indiquée dans la réclamation « nbf ». L'exemple de fonction Lambda que nous fournissons convertit la réponse `iat` du serveur en. `nbf`

- `exp`: (Date d'expiration) Réclamation — La demande « exp » (délai d'expiration) identifie le délai d'expiration à partir duquel ou après lequel le JWT ne doit pas être accepté pour traitement.
- `isAuthorized`: un booléen défini sur. `True` Indique que la demande a été autorisée sur le serveur d'autorisation.
- `aud`: Réclamation (audience) — La réclamation « aud » (audience) identifie les destinataires auxquels le JWT est destiné. Il doit s'agir d'un point de terminaison de banque de HealthLake données compatible SMART on FHIR.
- `scope`: Il doit s'agir d'au moins une étendue liée à une ressource FHIR. Cette étendue est définie sur votre serveur d'autorisation. Pour en savoir plus sur les champs d'application liés aux ressources FHIR acceptés par HealthLake, voir. SMART sur les champs de ressources FHIR pour HealthLake

SMART sur les oscilloscopes FHIR OAuth 2.0 pris en charge par HealthLake

HealthLake utilise OAuth 2.0 comme protocole d'autorisation. L'utilisation de ce protocole sur votre serveur d'autorisation vous permet de définir HealthLake des autorisations de stockage de données (création, lecture, mise à jour, suppression et recherche) pour les ressources FHIR auxquelles une application cliente a accès.

Le framework SMART on FHIR définit un ensemble de portées qui peuvent être demandées au serveur d'autorisation. Par exemple, une application client conçue uniquement pour permettre aux patients de consulter leurs résultats de laboratoire ou de consulter leurs coordonnées ne doit être autorisée qu'à demander `read` des scopes.

> **Note**
>
> HealthLake fournit un support pour SMART sur FHIR V1 et V2, comme décrit ci-dessous. Le SMART on FHIR `AuthorizationStrategy`est défini sur l'une des trois valeurs suivantes lors de la création de votre magasin de données :
>
> - `SMART_ON_FHIR_V1`— Support uniquement pour SMART sur FHIR V1, qui inclut les `read` autorisations (read/search) et `write` (create/update/delete).
> - `SMART_ON_FHIR`— Support de SMART sur FHIR V1 et V2, qui inclut`create`,`read`, `update`,`delete`, et `search` les autorisations.

- `AWS_AUTH`— La stratégie AWS HealthLake d'autorisation par défaut ; non affiliée à SMART on FHIR.

Périmètre de lancement autonome

HealthLake prend en charge le champ `launch/patient` d'application du mode de lancement autonome.

En mode de lancement autonome, une application client demande l'accès aux données cliniques du patient car l'utilisateur et le patient ne sont pas connus de l'application cliente. Ainsi, la demande d'autorisation de l'application cliente demande explicitement que le dossier du patient soit renvoyé. Une fois l'authentification réussie, le serveur d'autorisation émet un jeton d'accès contenant le champ d'application du patient de lancement demandé. Le contexte patient nécessaire est fourni avec le jeton d'accès dans la réponse du serveur d'autorisation.

Champs d'application du mode de lancement pris en charge

Scope	Description
`launch/patient`	Paramètre d'une demande d'autorisation OAuth 2.0 demandant que les données du patient soient renvoyées dans la réponse d'autorisation.

SMART sur les champs de ressources FHIR pour HealthLake

HealthLake définit trois niveaux de SMART sur les étendues de ressources FHIR.

- `patient`les scopes donnent accès à des données spécifiques concernant un seul patient.
- `user`les étendues donnent accès à des données spécifiques auxquelles un utilisateur peut accéder.
- `system`les étendues donnent accès à toutes les ressources FHIR présentes dans le magasin de HealthLake données.

Les sections suivantes répertorient la syntaxe permettant de créer des étendues de ressources FHIR à l'aide de SMART sur FHIR V1 ou SMART sur FHIR V2.

> **Note**
>
> La stratégie d'autorisation SMART on FHIR est définie lors de la création de votre magasin de données. Pour plus d'informations, consultez `AuthorizationStrategy` dans la Référence d'API AWS HealthLake .

SMART sur les oscilloscopes FHIR V1 pris en charge par HealthLake

Lorsque vous utilisez SMART sur FHIR V1, la syntaxe générale pour la construction des étendues de ressources FHIR est la suivante. HealthLake Pour afficher le chemin complet de l'URL dans l'exemple suivant, faites défiler le curseur sur le bouton Copier.

```
('patient' | 'user' | 'system') '/' (fhir-resource | '*') '.' ('read' | 'write' | '*')
```

Champs d'autorisation pris en charge par SMART on FHIR v1

Syntaxe Scope	Exemple de champ d'application	Résultat
`patient/(fhir-resource | '*').('read' | 'write' | '*')`	`patient/AllergyIntolerance.*`	L'application client du patient dispose d'un read/write accès au niveau de l'instance à toutes les allergies enregistrées.
`user/(fhir-resource | '*').('read' | 'write' | '*')`	`user/Observation.read`	L'application cliente utilisateur dispose d'un read/write accès au niveau de l'instance à toutes les observations enregistrées.
`system/('read' | 'write' | *)`	`system/*.*`	L'application cliente du système a read/write accès à toutes

Syntaxe Scope	Exemple de champ d'application	Résultat
		les données des ressources FHIR.

SMART sur les oscilloscopes FHIR V2 pris en charge par HealthLake

Lorsque vous utilisez SMART sur FHIR V2, la syntaxe générale pour la construction des étendues de ressources FHIR est la suivante. HealthLake Pour afficher le chemin complet de l'URL dans l'exemple suivant, faites défiler le curseur sur le bouton Copier.

```
('patient' | 'user' | 'system') '/' (fhir-resource | '*') '.' ('c' | 'r' | 'u' | 'd' |
 's')
```

> **Note**
>
> Pour utiliser SMART sur FHIR V2, vous devez transmettre la valeur `permission-v2`dans la `capabilities` chaîne de métadonnées, qui est membre du type de `IdentityProviderConfiguration`données.
> HealthLake prend en charge les oscilloscopes granulaires. Pour plus d'informations, voir les étendues granulaires prises en charge dans le guide de mise en œuvre de FHIR US Core.

Étendue d'autorisation compatible avec SMART on FHIR V2

Syntaxe Scope	Exemple de scope V1	Résultat
`patient/Observation.rs`	`user/Observation.read`	Autorisation de lire et de rechercher `Observation` une ressource pour le patient actuel.
`system/*.cruds`	`system/*.*`	L'application cliente du système dispose d'un accès complet create/read/update

Syntaxe Scope	Exemple de scope V1	Résultat
		/delete/search à toutes les données de ressources FHIR.

SMART sur les V2.2 oscilloscopes FHIR compatibles avec HealthLake

V2.2 étend le champ d'application de la V2 avec un filtrage basé sur les paramètres de recherche. Les serveurs d'autorisation peuvent désormais émettre des étendues qui limitent l'accès en fonction de caractéristiques de données spécifiques et pas uniquement en fonction du type de ressource et du fonctionnement du CRUDS.

Tout reste inchangé depuis la V2. V2.2 est purement additif :

- Les oscilloscopes V2 existants (sans filtres) continuent de fonctionner comme avant.
- La grammaire V2 est étendue avec une chaîne de `?param=value` requête facultative.
- Aucune modification des niveaux de portée (`patient`/`user`/`system`), des types de ressources ou des lettres CRUDS.

Conditions préalables

Avant d'activer SMART sur FHIR V2.2, assurez-vous de ce qui suit :

- Votre magasin de données a été créé avec `AuthorizationStrategy` set to `SMART_ON_FHIR` (supporte à la fois les versions V1 et V2). Les magasins de données utilisent `SMART_ON_FHIR_V1` ou ne `AWS_AUTH` sont pas éligibles.
- Votre magasin de données est déjà inclus `permission-v2` dans le `capabilities` tableau. V2.2 est additif car il étend la V2 et ne peut pas être utilisé seul.
- Votre IDP Lambda est configuré pour valider et transmettre le format de portée (étendues ? contenant V2.2 la syntaxe de requête).

Activant V2.2

Le chemin d'activation varie selon que vous créez un nouveau magasin de données ou que vous mettez à jour un magasin existant.

Nouveaux magasins de données

Lorsque vous créez un nouveau magasin de données, ajoutez-le `permission-v2.2` au `capabilities` tableau dans le `Metadata` champ de votre `IdentityProviderConfiguration`:

```
"capabilities": [
  "launch-ehr",
  "sso-openid-connect",
  "client-public",
  "permission-v2",
  "permission-v2.2"
]
```

Magasins de données existants

Pour activer SMART sur FHIR V2.2 sur un magasin de données existant, ajoutez-le `permission-v2.2` à la `capabilities` matrice dans le `Metadata` champ de votre `IdentityProviderConfiguration`et soumettez la modification avec`UpdateFHIRDatastore`. Pour de plus amples informations, veuillez consulter Mettre à jour un magasin HealthLake de données.

Prérequis:

- `permission-v2`doit rester dans le tableau. V2.2 étend la V2 et ne peut pas être utilisée seule.
- `AuthorizationStrategy`doit être `SMART_ON_FHIR` (pas `SMART_ON_FHIR_V1` ou`AWS_AUTH`).
- La mise à jour de la configuration du fournisseur d'identité la remplace dans son intégralité. Incluez donc tous les champs et fonctionnalités existants`permission-v2.2`.

La modification prend effet immédiatement, sans interruption de service. Pour vérifier, récupérez le Document de découverte (nécessite SigV4) :

```
GET {healthlake-endpoint}/r4/.well-known/smart-configuration
```

Si le `capabilities` tableau indiqué dans la réponse inclut`permission-v2.2`, SMART sur FHIR V2.2 est actif.

Syntaxe à portée étendue

La grammaire V2 est étendue avec une chaîne de requête facultative :

```
V2:   (patient|user|system) / resource . cruds
V2.2: (patient|user|system) / resource . cruds [? param=value [& param=value ...]]
```

V2.2 composants de chaîne de requête scope

Composant	Description
`?param=value`	Search-parameter filtre. Seules les ressources répondant à ce critère sont accessibles.
`&param=value`	Filtre supplémentaire. Les filtres multiples sont des filtres ANDED ; ils doivent tous correspondre.

Règles :

- Les filtres s'appliquent uniquement au type de ressource spécifié dans le champ d'application. Wildcard (*) avec filtres n'est pas pris en charge.
- Les paramètres doivent être valides pour le type de ressource conformément à la configuration de recherche de votre magasin de données (vérifiez via`GET /r4/metadata`).
- La chaîne de portée complète (par exemple,`patient/Observation.rs?category=laboratory`) est la valeur littérale qui apparaît dans le paramètre de portée OAuth 2.0 et dans la demande de jeton d'accès. `scp`
- URL-encode caractères spéciaux conformément à la RFC 6749 dans les demandes d'autorisation (par exemple, `|` →`%7C`).
- Pour les paramètres de date, de nombre et de quantité, V2.2 prend en charge les comparateurs de préfixes (par exemple,`?date=eq2023-01-01`). V2.2 prend également en charge les modificateurs de paramètres de recherche.

Exemples de scopes

V2.2 exemples de scopes

Scope	Accès accordé
`patient/DiagnosticReport.rs?category=LAB`	Seules les `DiagnosticReport` ressources là où elles `category` se trouvent`LAB`.

Scope	Accès accordé
`patient/Observation.rs?_security=http://terminology.hl7.org/CodeSystem/v3-Confidentiality|R`	Uniquement `Observation` les ressources dotées d'une étiquette de sécurité`Restricted` .
`patient/Observation.rs?date=ge2023-01-01`	Uniquement `Observation` les ressources datées du 1er janvier 2023 ou après cette date.
`patient/Observation.rs?category=laboratory&status=final`	Uniquement `Observation` les ressources qui sont de laboratoire ET finales.
`user/Condition.rs?clinical-status=active`	`Condition` Ressources actives uniquement.

Comportement d'application

Lorsqu'un jeton inclut V2.2 des étendues, HealthLake applique des filtres par opération :

V2.2 mise en œuvre par opération

Opération	Comportement
Lisez (`r`)	Réussit uniquement si la ressource correspond à tous les filtres de portée. Sinon, renvoie 403.
Rechercher (`s`)	Les filtres de portée sont intersectés avec la requête. Seules les ressources correspondantes sont renvoyées.
Create/Update (`c`/`u`)	La ressource doit satisfaire aux filtres de portée pour être écrite. Sinon, renvoie 403.
Supprimer (`d`)	La ressource cible doit correspondre aux filtres de portée. Sinon, renvoie 403.

Priorité du champ

- Plusieurs V2.2 étendues pour le même type de ressource sont unifiées (OU entre plusieurs étendues).
- Une portée V2 plus large sans filtres (par exemple,`patient/Observation.rs`) accorde un accès complet indépendamment des V2.2 étendues plus étroites dans le même jeton.
- V2.2 les étendues d'un magasin de données non `permission-v2.2` activées sont ignorées silencieusement.

Limitations

Les éléments suivants ne sont pas pris en charge dans les filtres V2.2 d'étendue :

- Paramètres de recherche composites (par exemple,`code-value-quantity`).
- Paramètres de recherche enchaînés (par exemple,`subject:Patient.name=Smith`).
- `_include`/paramètres `_revinclude` de recherche.
- `$export`/`$davinci-data-export`(Bulk Data) : V2.2 les filtres ne s'appliquent pas ; l'exportation en bloc utilise des étendues V2.
- Type de ressource générique combiné à des filtres (par exemple, non `patient/*.rs?category=LAB` valide). Vous devez spécifier un type de ressource explicite lorsque vous utilisez des filtres de paramètres de recherche (par exemple,`patient/Observation.rs?category=LAB`).

Résolution des problèmes

Symptôme	Cause	Corriger
La portée du jeton n'est pas reconnue	V2.2 non activé sur le magasin de données	Vérifiez `/.well-known/smart-configuration` et demandez l'activation via un ticket d'assistance.
403 sur une ressource qui existe	La ressource ne correspond pas au filtre de portée	Vérifiez les valeurs des ressources par rapport aux paramètres de portée.

Symptôme	Cause	Corriger
Résultats de recherche vides	Le filtre de portée est plus étroit que celui de la requête	Les résultats sont l'intersection des filtres de requête et de portée.
`InvalidScope` Erreur	Paramètre de recherche non valide dans le champ d'application	Confirmez le paramètre via `/metadata` CapabilityStatement.

End-to-end exemple

Scénario : une application destinée aux patients ne devrait afficher les résultats de laboratoire finalisés qu'à partir de 2023.

1. Le serveur d'autorisation émet un jeton dont la portée est la suivante :

```
patient/Observation.rs?category=laboratory&status=final&date=ge2023-01-01
```

2. Appels du client HealthLake :

```
GET {endpoint}/r4/Observation?patient=Patient/123
```

3. HealthLake applique les filtres de scope. La réponse contient uniquement `Observation` des ressources où `category=laboratory` ET `status=final` ET`date ≥ 2023-01-01`, même si le client a demandé toutes les observations.

Validation des jetons en utilisant AWS Lambda

Lorsque vous créez un magasin de données compatible HealthLake SMART sur FHIR, vous devez fournir l'ARN de la AWS Lambda fonction dans la `CreateFHIRDatastore` demande. L'ARN de la fonction Lambda est spécifié dans l'`IdentityProviderConfiguration`objet à l'aide du `IdpLambdaArn` paramètre.

Vous devez créer la fonction Lambda avant de créer votre magasin de données compatible SMART on FHIR. Une fois le magasin de données créé, vous pouvez pointer vers une autre fonction Lambda en la mettant à jour `IdpLambdaArn` avec. `UpdateFHIRDatastore` Pour voir l'ARN Lambda actuellement configuré pour le magasin de données, utilisez l'action `DescribeFHIRDatastore`

API. Pour de plus amples informations, veuillez consulter Mettre à jour un magasin HealthLake de données.

Pour qu'une demande REST FHIR aboutisse sur un magasin de données compatible SMART on FHIR, votre fonction Lambda doit effectuer les opérations suivantes :

- Renvoie une réponse en moins d'une seconde au point de terminaison du magasin de HealthLake données.
- Décodez le jeton d'accès fourni dans l'en-tête d'autorisation de la demande d'API REST envoyée par l'application cliente.
- Attribuez un rôle de service IAM doté d'autorisations suffisantes pour exécuter la demande d'API REST FHIR.
- Les demandes suivantes sont requises pour compléter une demande d'API REST FHIR. Pour en savoir plus, veuillez consulter la section Demandes requises.
  - `nbf`
  - `exp`
  - `isAuthorized`
  - `aud`
  - `scope`

Lorsque vous travaillez avec Lambda, vous devez créer un rôle d'exécution et une politique basée sur les ressources en plus de votre fonction Lambda. Le rôle d'exécution d'une fonction Lambda est un rôle IAM qui accorde à la fonction l'autorisation d'accéder aux services et aux ressources AWS nécessaires au moment de l'exécution. La politique basée sur les ressources que vous fournissez doit HealthLake permettre d'invoquer votre fonction en votre nom.

Les sections de cette rubrique décrivent un exemple de demande provenant d'une application cliente et une réponse décodée, les étapes nécessaires à la création d'une fonction AWS Lambda et la manière de créer une politique basée sur les ressources qui peut être assumée. HealthLake

- Partie 1 : Création d'une fonction Lambda
- Partie 2 : Création d'un rôle HealthLake de service utilisé par la fonction AWS Lambda
- Partie 3 : Mise à jour du rôle d'exécution de la fonction Lambda
- Partie 4 : Ajouter une politique de ressources à votre fonction Lambda
- Partie 5 : Configuration de la simultanéité pour votre fonction Lambda

Création d'un AWS Fonction Lambda

La fonction Lambda créée dans cette rubrique est déclenchée lors de la HealthLake réception d'une demande à un magasin de données compatible SMART on FHIR. La demande de l'application cliente contient un appel d'API REST et un en-tête d'autorisation contenant un jeton d'accès.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
Authorization: Bearer i8hweunweunweofiwweoijewiwe
```

L'exemple de fonction Lambda présenté dans cette rubrique permet de masquer AWS Secrets Manager les informations d'identification liées au serveur d'autorisation. Nous recommandons vivement de ne pas fournir les informations de connexion au serveur d'autorisation directement dans une fonction Lambda.

Example validation d'une requête REST FHIR contenant un jeton porteur d'autorisation

L'exemple de fonction Lambda vous montre comment valider une demande REST FHIR envoyée à un magasin de données compatible SMART on FHIR. Pour obtenir des instructions détaillées sur la façon d'implémenter cette fonction Lambda, consultez. Création d'une fonction Lambda à l'aide du AWS Management Console

Si la demande d'API REST FHIR ne contient pas de point de terminaison de stockage de données, de jeton d'accès et d'opération REST valides, la fonction Lambda échouera. Pour en savoir plus sur les éléments du serveur d'autorisation requis, consultezDemandes requises.

```
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()
```

Création d'une fonction Lambda à l'aide du AWS Management Console

La procédure suivante suppose que vous avez déjà créé le rôle de service que vous HealthLake souhaitez assumer lors du traitement d'une demande d'API REST FHIR sur un magasin de données compatible SMART sur FHIR. Si vous n'avez pas créé le rôle de service, vous pouvez toujours créer la fonction Lambda. Vous devez ajouter l'ARN du rôle de service pour que la fonction Lambda fonctionne. Pour en savoir plus sur la création d'un rôle de service et sa spécification dans la fonction Lambda, voir Création d'un rôle de HealthLake service à utiliser dans AWS Fonction Lambda utilisée pour décoder un JWT

Pour créer une fonction Lambda (AWS Management Console)

1. Ouvrez la page Functions (Fonctions) de la console Lambda.
2. Choisissez Créer une fonction.
3. Sélectionnez Créer à partir de zéro.
4. Dans Informations de base, entrez le nom de la fonction. Sous Runtime, choisissez un environnement d'exécution basé sur Python.
5. Pour Execution role (Rôle d'exécution), choisissez Create a new role with basic Lambda permissions (Créer un nouveau rôle avec les autorisations Lambda de base).

   Lambda crée un rôle d'exécution qui accorde à la fonction l'autorisation de télécharger des journaux sur Amazon. CloudWatch La fonction Lambda assume le rôle d'exécution lorsque vous appelez votre fonction et utilise le rôle d'exécution pour créer des informations d'identification pour le AWS SDK.
6. Choisissez l'onglet Code et ajoutez l'exemple de fonction Lambda.

   Si vous n'avez pas encore créé le rôle de service que la fonction Lambda doit utiliser, vous devez le créer pour que l'exemple de fonction Lambda fonctionne. Pour en savoir plus sur la création d'un rôle de service pour la fonction Lambda, consultez. Création d'un rôle de HealthLake service à utiliser dans AWS Fonction Lambda utilisée pour décoder un JWT

```
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()
```

Modifier le rôle d'exécution d'une fonction Lambda

Après avoir créé la fonction Lambda, vous devez mettre à jour le rôle d'exécution afin d'inclure les autorisations nécessaires pour appeler Secrets Manager. Dans Secrets Manager, chaque secret que vous créez possède un ARN. Pour appliquer le moindre privilège, le rôle d'exécution doit uniquement avoir accès aux ressources nécessaires à l'exécution de la fonction Lambda.

Vous pouvez modifier le rôle d'exécution d'une fonction Lambda en la recherchant dans la console IAM ou en choisissant Configuration dans la console Lambda. Pour en savoir plus sur la gestion de votre rôle d'exécution des fonctions Lambda, consultez. Rôle d’exécution Lambda

Example Rôle d'exécution de la fonction Lambda qui donne accès à **`GetSecretValue`**

L'ajout de l'action IAM `GetSecretValue` au rôle d'exécution accorde l'autorisation nécessaire au fonctionnement de l'exemple de fonction Lambda.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:secret-name-DKodTA"
        }
    ]
}
```

À ce stade, vous avez créé une fonction Lambda qui peut être utilisée pour valider le jeton d'accès fourni dans le cadre de la demande REST FHIR envoyée à votre banque de données compatible SMART on FHIR.

Création d'un rôle de HealthLake service à utiliser dans AWS Fonction Lambda utilisée pour décoder un JWT

> Persona : administrateur IAM
>
> Utilisateur qui peut ajouter ou supprimer des politiques IAM et créer de nouvelles identités IAM.
>
> Rôle de service
>
> Un rôle de service est un rôle IAM qu'un service endosse pour accomplir des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer un rôle de service à partir d'IAM. Pour plus d'informations, consultez Création d'un rôle pour la délégation d'autorisations à un Service AWS dans le Guide de l'utilisateur IAM.

Une fois le jeton Web JSON (JWT) décodé, l'autorisation dont Lambda a besoin doit également renvoyer un ARN de rôle IAM. Ce rôle doit disposer des autorisations nécessaires pour exécuter la demande d'API REST, sinon il échouera en raison d'autorisations insuffisantes.

Lorsque vous configurez une politique personnalisée à l'aide d'IAM, il est préférable d'accorder les autorisations minimales requises. Pour en savoir plus, consultez la section Appliquer les autorisations de moindre privilège dans le Guide de l'utilisateur IAM.

La création d'un rôle de HealthLake service à désigner dans la fonction Lambda d'autorisation nécessite deux étapes.

- Tout d'abord, vous devez créer une politique IAM. La politique doit spécifier l'accès aux ressources FHIR pour lesquelles vous avez fourni des étendues sur le serveur d'autorisation.
- Ensuite, vous devez créer le rôle de service. Lorsque vous créez le rôle, vous désignez une relation de confiance et associez la politique que vous avez créée à la première étape. La relation de confiance désigne HealthLake le principal du service. Vous devez spécifier un ARN de stockage de HealthLake données et un ID de AWS compte au cours de cette étape.

Création d'une nouvelle politique IAM

Les étendues que vous définissez dans votre serveur d'autorisation déterminent les ressources FHIR auxquelles un utilisateur authentifié a accès dans un HealthLake magasin de données.

La politique IAM que vous créez peut être adaptée aux étendues que vous avez définies.

Les actions suivantes peuvent être définies dans l'`Action`élément d'une déclaration de politique IAM. Pour chacun `Action` des éléments du tableau, vous pouvez définir un`Resource types`. Dans HealthLake un magasin de données, le seul type de ressource pris en charge peut être défini dans l'`Resource`élément d'une déclaration de politique d'autorisation IAM.

Les ressources FHIR individuelles ne sont pas des ressources que vous pouvez définir en tant qu'élément d'une politique d'autorisation IAM.

Actions définies par HealthLake

Actions	Description	Nivea d'acc	Type de ressource (obligatoire)
CreateRes ource	Accorde l'autorisation de créer une ressource	Écrire	ARN de la banque de données : arn:aws:h ealthlake : : :/**your-region 111122223 333** datastore/fhir**your-datastore-id**
DeleteRes ource	Accorde l'autorisation de supprimer une ressource	Écrire	ARN de la banque de données : arn:aws:h ealthlake : : :/**your-region 111122223 333** datastore/fhir**your-datastore-id**
ReadResou rce	Accorde l'autorisation de lire une ressource	Lectu	ARN de la banque de données : arn:aws:h ealthlake : : :/**your-region 111122223 333** datastore/fhir**your-datastore-id**
SearchWit hGet	Accorde l'autorisation de rechercher des ressource s avec la méthode GET	Lectu	ARN de la banque de données : arn:aws:h ealthlake : : :/**your-region 111122223 333** datastore/fhir**your-datastore-id**
SearchWit hPost	Accorde l'autorisation de rechercher des ressource s avec la méthode POST	Lectu	ARN de la banque de données : arn:aws:h ealthlake : : :/**your-region 111122223 333** datastore/fhir**your-datastore-id**

Actions	Description	Nivea d'acc	Type de ressource (obligatoire)
StartFHIR ExportJob WithPost	Donne l'autorisation de commencer un travail d'exportation FHIR avec GET	Écrire	ARN de la banque de données : arn:aws:healthlake : : :/**your-region 111122223333** datastore/fhir**your-datastore-id**
UpdateResource	Accorde l'autorisation de mettre à jour des ressources	Écrire	ARN de la banque de données : arn:aws:healthlake : : :/**your-region 111122223333** datastore/fhir**your-datastore-id**

Pour commencer, vous pouvez utiliser`AmazonHealthLakeFullAccess`. Cette politique autoriserait la lecture, l'écriture, la recherche et l'exportation de toutes les ressources FHIR présentes dans un magasin de données. Pour accorder des autorisations de lecture seule sur un magasin de données, utilisez. `AmazonHealthLakeReadOnlyAccess`

Pour en savoir plus sur la création d'une politique personnalisée à l'aide des SDK AWS Management Console AWS CLI, ou des SDK IAM, consultez la section Création de politiques IAM dans le guide de l'utilisateur IAM.

Création d'un rôle de service pour HealthLake (console IAM)

Utilisez cette procédure pour créer un rôle de service. Lorsque vous créez un service, vous devez également définir une politique IAM.

Pour créer le rôle de service pour HealthLake (console IAM)

1. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à https://console.aws.amazon.com/iam/l'adresse.
2. Dans le panneau de navigation de la console IAM, choisissez Rôles.
3. Puis, choisissez Create role (Créer un rôle).
4. Sur la page Sélectionner une entité de confiance, choisissez Politique de confiance personnalisée.
5. Ensuite, sous Politique de confiance personnalisée, mettez à jour l'exemple de politique comme suit. Remplacez-le **`your-account-id`** par votre numéro de compte et ajoutez l'ARN du magasin de données que vous souhaitez utiliser dans vos tâches d'importation ou d'exportation.

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. Ensuite, choisissez Suivant.
7. Sur la page Ajouter des autorisations, choisissez la politique que vous souhaitez que le HealthLake service adopte. Pour trouver votre politique, recherchez-la dans Politiques d'autorisations.
8. Choisissez ensuite Attach policy.
9. Ensuite, sur la page Nom, révision et création sous Nom du rôle, entrez un nom.
10. (Facultatif) Ensuite, sous Description, ajoutez une brève description de votre rôle.
11. Si possible, saisissez un nom de rôle ou le suffixe d'un nom de rôle vous permettant d'identifier l'objectif du rôle. Les noms de rôle doivent être uniques dans votre Compte AWS. Ils ne sont pas distingués au cas par cas. Par exemple, vous ne pouvez pas créer deux rôles nommés **`PRODROLE`** et **`prodrole`**. Différentes entités peuvent référencer le rôle et il n'est donc pas possible de modifier son nom après sa création.
12. Passez en revue les détails du rôle, puis choisissez Créer un rôle.

Pour savoir comment spécifier l'ARN du rôle dans l'exemple de fonction Lambda, consultez. Création d'un AWS Fonction Lambda

Rôle d’exécution Lambda

Le rôle d'exécution d'une fonction Lambda est un rôle IAM qui accorde à la fonction l'autorisation d'accéder aux AWS services et aux ressources. Cette page fournit des informations sur la façon de créer, d’afficher et de gérer le rôle d’exécution d’une fonction Lambda.

Par défaut, Lambda crée un rôle d'exécution avec des autorisations minimales lorsque vous créez une nouvelle fonction Lambda à l'aide du. AWS Management Console Pour gérer les autorisations accordées dans le rôle d'exécution, consultez la section Création d'un rôle d'exécution dans la console IAM du guide du développeur Lambda.

L'exemple de fonction Lambda fourni dans cette rubrique utilise Secrets Manager pour masquer les informations d'identification du serveur d'autorisation.

Comme pour tout rôle IAM que vous créez, il est important de suivre les meilleures pratiques du moindre privilège. Au cours de la phase de développement, vous pouvez parfois accorder des autorisations au-delà de ce qui est requis. Avant de publier votre fonction dans l’environnement de production, une bonne pratique consiste à ajuster la stratégie de manière à inclure uniquement les autorisations requises. Pour plus d'informations, consultez la section Appliquer le moindre privilège dans le guide de l'utilisateur IAM.

HealthLake Autoriser le déclenchement de votre fonction Lambda

HealthLake Vous pouvez donc invoquer la fonction Lambda en votre nom, vous devez procéder comme suit :

- Vous devez définir une `IdpLambdaArn` valeur égale à l'ARN de la fonction Lambda que vous HealthLake souhaitez invoquer dans la `CreateFHIRDatastore` demande.
- Vous avez besoin d'une politique basée sur les ressources permettant HealthLake d'appeler la fonction Lambda en votre nom.

Lorsqu'il HealthLake reçoit une demande d'API REST FHIR sur un magasin de données compatible SMART on FHIR, il a besoin d'autorisations pour appeler la fonction Lambda spécifiée lors de la création du magasin de données en votre nom. Pour accorder HealthLake l'accès, vous allez utiliser une politique basée sur les ressources. Pour en savoir plus sur la création d'une politique basée

sur les ressources pour une fonction Lambda, voir Autoriser un AWS service à appeler une fonction Lambda dans le Guide du développeur.AWS Lambda

Provisionnement de la simultanéité pour votre fonction Lambda

> **Important**
>
> HealthLake exige que la durée d'exécution maximale de votre fonction Lambda soit inférieure à une seconde (1 000 millisecondes).
> Si votre fonction Lambda dépasse la limite de temps d'exécution, vous obtenez une TimeOutexception.

Pour éviter cette exception, nous vous recommandons de configurer la simultanéité provisionnée. En allouant la simultanéité provisionnée avant une augmentation des appels, vous pouvez vous assurer que toutes les demandes sont servies par des instances initialisées avec une faible latence. Pour en savoir plus sur la configuration de la simultanéité provisionnée, voir Configuration de la simultanéité provisionnée dans le Guide du développeur Lambda

Pour connaître la durée d'exécution moyenne de votre fonction Lambda, utilisez actuellement la page de surveillance de votre fonction Lambda sur la console Lambda. Par défaut, la console Lambda fournit un graphique de durée qui indique le temps moyen, minimum et maximum consacré par votre code de fonction au traitement d'un événement. Pour en savoir plus sur la surveillance des fonctions Lambda, consultez la section Surveillance des fonctions dans la console Lambda dans le guide du développeur Lambda.

Si vous avez déjà configuré la simultanéité pour votre fonction Lambda et que vous souhaitez la surveiller, consultez la section Surveillance de la simultanéité dans le guide du développeur Lambda.

Utilisation d'une autorisation précise avec un magasin de données compatible SMART on FHIR HealthLake

Les champs d'application à eux seuls ne vous fournissent pas les précisions nécessaires quant aux données auxquelles un demandeur est autorisé à accéder dans un magasin de données. L'utilisation d'une autorisation précise permet d'obtenir un niveau de spécificité plus élevé lors de l'octroi de l'accès à un magasin de données compatible HealthLake SMART on FHIR. Pour utiliser une autorisation précise, définissez la `FineGrainedAuthorizationEnabled` valeur égale à `True` dans le `IdentityProviderConfiguration` paramètre de votre `CreateFHIRDatastore` demande.

Vous pouvez également activer ou désactiver l'autorisation précise sur un magasin de données existant en le mettant à jour `FineGrainedAuthorizationEnabled` avec. `UpdateFHIRDatastore` Pour de plus amples informations, veuillez consulter Mettre à jour un magasin HealthLake de données.

Si vous avez activé l'autorisation détaillée, votre serveur d'autorisation renvoie une `fhirUser` étendue `id_token` avec le jeton d'accès. Cela permet de récupérer des informations sur l'utilisateur par l'application cliente. L'application cliente doit traiter la `fhirUser` demande comme l'URI d'une ressource FHIR représentant l'utilisateur actuel. Cette valeur peut être `Patient`, `Practitioner` ou `RelatedPerson`. La réponse du serveur d'autorisation inclut également une `user/` étendue qui définit les données auxquelles l'utilisateur peut accéder. Cela utilise la syntaxe définie pour les portées liées aux portées spécifiques aux ressources FHIR :

```
user/(fhir-resource | '*').('read' | 'write' | '*')
```

Vous trouverez ci-dessous des exemples de la manière dont une autorisation précise peut être utilisée pour spécifier davantage les types de ressources FHIR liés à l'accès aux données.

- Quand `fhirUser` une `Practitioner` autorisation précise détermine-t-elle le nombre de patients auxquels l'utilisateur peut accéder. L'accès à n'`fhirUser`est autorisé qu'aux patients auxquels le patient fait référence en `fhirUser` tant que médecin généraliste.

  ```
  Patient.generalPractitioner : [{Reference(Practitioner)}]
  ```

- Quand `fhirUser` un `Patient` ou `RelatedPerson` le patient référencé dans la demande est différent de `fhirUser` l'autorisation précise détermine l'accès `fhirUser` du patient demandé. L'accès est autorisé lorsqu'une relation est spécifiée dans la `Patient` ressource demandée.

  ```
  Patient.link.other : {Reference(Patient|RelatedPerson)}
  ```

Récupération du document SMART sur FHIR Discovery

SMART définit un document de découverte qui permet aux clients de connaître le point de terminaison d'autorisation URLs et propose des fonctionnalités prises en charge par un magasin de HealthLake données. Ces informations aident les clients à diriger les demandes d'autorisation vers le point de terminaison approprié et à créer des demandes d'autorisation prises en charge par le magasin de HealthLake données.

Pour qu'une application cliente puisse envoyer une demande FHIR REST réussie HealthLake, elle doit rassembler les exigences d'autorisation définies par le magasin de HealthLake données. Aucun jeton porteur (autorisation) n'est requis pour que cette demande aboutisse.

Pour demander le document de découverte pour un magasin HealthLake de données

1. Collectez HealthLake `region` et `datastoreId` valorisez. Pour de plus amples informations, veuillez consulter Obtenir les propriétés du magasin de données.
2. Construisez une URL pour la demande en utilisant les valeurs collectées pour HealthLake `region` et`datastoreId`. Ajouter au point `/.well-known/smart-configuration` de terminaison de l'URL. Pour afficher le chemin complet de l'URL dans l'exemple suivant, faites défiler le curseur sur le bouton Copier.

   ```
   https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/.well-known/smart-configuration
   ```

3. Envoyez la demande en utilisant `GET` le protocole de AWS signature Signature Version 4. Pour afficher l'exemple dans son intégralité, faites défiler la souris sur le bouton Copier.

   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'
   ```

   Le document de découverte du magasin de HealthLake données est renvoyé sous la forme d'un blob JSON, dans lequel vous pouvez trouver le `authorization_endpoint` et le`token_endpoint`, ainsi que les spécifications et les fonctionnalités définies pour le magasin de données.

   ```
   {
       "authorization_endpoint": "https://oidc.example.com/authorize",
       "token_endpoint": "https://oidc.example.com/oauth/token",
       "capabilities": [
           "launch-ehr",
   ```

```
        "client-public"
    ]
}
```

Le `authorization_endpoint` et le `token_endpoint` sont tous deux nécessaires pour lancer une application client.

- Point de terminaison d'autorisation : URL nécessaire pour autoriser une application cliente ou un utilisateur.
- Point de terminaison du jeton : point de terminaison du serveur d'autorisation avec lequel l'application cliente communique.

Effectuer une demande d'API REST FHIR sur un magasin de données compatible SMART HealthLake

Vous pouvez effectuer des demandes d'API REST FHIR sur un magasin de données compatible SMART sur FHIR. HealthLake L'exemple suivant montre une demande d'une application cliente contenant un JWT dans l'en-tête d'autorisation et montre comment Lambda doit décoder la réponse. Une fois que la demande d'application cliente est autorisée et authentifiée, elle doit recevoir un jeton porteur du serveur d'autorisation. Utilisez le jeton porteur dans l'en-tête d'autorisation lorsque vous envoyez une demande d'API REST FHIR sur un magasin de données compatible SMART on HealthLake FHIR.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/[ID]
Authorization: Bearer auth-server-provided-bearer-token
```

Comme un jeton porteur a été trouvé dans l'en-tête d'autorisation et qu'aucune identité AWS IAM n'a été détectée, la fonction Lambda spécifiée lors de HealthLake la création du magasin de données compatible SMART on FHIR a été créé. HealthLake Lorsque le jeton est correctement décodé par votre fonction Lambda, l'exemple de réponse suivant est envoyé à. HealthLake

```
{
  "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
}
```

Support FHIR R4 pour AWS HealthLake

AWS HealthLake prend en charge la spécification FHIR R4 pour l'échange de données de santé. Les sections suivantes fournissent des informations complémentaires sur la façon dont la spécification FHIR R4 est HealthLake utilisée pour vous aider à gérer et à rechercher des ressources FHIR dans votre banque de HealthLake données à l'aide de FHIR R4. RESTful APIs

Rubriques

- Déclaration de capacité du FHIR R4 pour AWS HealthLake
- Validations du profil FHIR pour HealthLake
- Types de ressources pris en charge par FHIR R4 pour HealthLake
- Paramètres de recherche FHIR R4 pour HealthLake
- FHIR R4 pour $operations HealthLake

Déclaration de capacité du FHIR R4 pour AWS HealthLake

Pour connaître les fonctionnalités (comportements) liées au FHIR d'un magasin de HealthLake données actif, vous devez récupérer sa déclaration de capacité. La déclaration de capacité est utilisée comme déclaration de fonctionnalité réelle du serveur ou comme déclaration de mise en œuvre requise ou souhaitée du serveur. L'`capabilities`interaction FHIR récupère des informations sur les capacités du magasin de HealthLake données et les parties de la spécification FHIR prises en charge. HealthLake valide les types de ressources FHIR en fonction de la ressource FHIR R4. `StructureDefinition`

Pour obtenir la déclaration de capacité d'un magasin HealthLake de données

1. Collectez HealthLake `region` et `datastoreId` valorisez. Pour de plus amples informations, veuillez consulter Obtenir les propriétés du magasin de données.
2. Construisez une URL pour la demande en utilisant les valeurs collectées pour HealthLake `region` et`datastoreId`. Incluez également l'`metadata`élément FHIR dans l'URL. Pour afficher le chemin complet de l'URL dans l'exemple suivant, faites défiler le curseur sur le bouton Copier.

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

3. Envoyez la demande . L'`capabilities`interaction FHIR utilise une `GET` demande avec le protocole de AWS signature Signature Version 4. L'`curl`exemple suivant obtient la déclaration de capacité pour le magasin de HealthLake données spécifié par le`datastoreId`. Pour afficher l'exemple dans son intégralité, faites défiler la souris sur le bouton Copier.

   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'
   ```

   Vous recevrez un code de réponse `200` HTTP et la déclaration de capacité pour votre magasin de HealthLake données. Pour plus d'informations, consultez la `CapabilityStatement`documentation du FHIR R4.

Validations du profil FHIR pour HealthLake

AWS HealthLake prend en charge la spécification FHIR R4 de base. Les profils FHIR sont inclus dans la spécification FHIR R4 de base. Les profils sont utilisés sur un type de ressource FHIR pour définir une définition de type de ressource plus spécifique à l'aide d' and/or extensions de contraintes sur le type de ressource de base. Par exemple, un profil FHIR peut identifier les champs obligatoires tels que les extensions et les ensembles de valeurs. Une ressource peut prendre en charge plusieurs profils. Tous les magasins HealthLake de données prennent en charge l'utilisation des profils FHIR.

> **Note**
>
> L'ajout d'un profil FHIR n'est pas obligatoire lors de l'ajout de données dans un magasin de HealthLake données. Si aucun profil FHIR n'est spécifié lors de l'ajout ou de la mise à jour d'une ressource, la ressource est validée uniquement par rapport au schéma FHIR R4 de base.
>
> Les profils FHIR, auxquels les ressources FHIR sont conformes, sont inclus dans les ressources avant leur importation. HealthLake Par conséquent, les profils FHIR sont validés HealthLake lors de l'importation.

Les profils FHIR sont spécifiés dans un guide de mise en œuvre. Un guide de mise en œuvre FHIR (IG) est un ensemble d'instructions qui décrivent comment utiliser la norme FHIR dans un but spécifique. HealthLake valide les profils FHIR définis dans les guides de mise en œuvre suivants.

Profils FHIR pris en charge par AWS HealthLake

Nom	Ve	Guide d'impléme ntation	Capacité	US Est (Ol	US Est (Vi du No	US Ou (Or	Asi Pa fiqu (Mu	Asi Pa fiqu (Sy	Ca (Ce	Eu (Irl	Europe (Londres)
Noyau américain	3.1	http://hl7.org/fhir/us/core/STU3.1.1/	Par défaut	X	X	X	X	X	X	X	X
Noyau américain	4.0	https://hl7.org/fhir/us/core/STU4/index.html	Pris en charge	X	X	X	X	X	X	X	X
Noyau américain	5.0	https://hl7.org/fhir/us/core/STU5.0.1/index.html	Pris en charge	X	X	X	X	X	X	X	X
Noyau américain	6.1	https://hl7.org/fhir/us/core/STU6.1/index.html	Pris en charge	X	X	X	X	X	X	X	X
Noyau américain	7.0	https://hl7.org/fhir/us/core/STU7/	Pris en charge	X	X	X	X	X	X	X	X

Nom	Ve	Guide d'impléme ntation	Capacité	US Est (Ol	US Est (Vi du No	US Ou (Or	Asi Pa fiqu (Mi	Asi Pa fiqu (Sy	Ca (Ce	Eu (Irl	Europe (Londres)
Noyau britannique	2.0	https://simplifier.net/guide/uk-core-implementation-guide-stu2/Home/ProfilesandExtensions/ProfilesIndex?version=2.0.1	Pris en charge	X	X	X	X			X	X
Bouton bleu CARIN	1.1	http://hl7.org/fhir/us/carin-bb/STU1.1/	Par défaut	X	X	X	X	X	X	X	X
Bouton bleu CARIN	1.0 2.0 2.1	https://hl7.org/fhir/us/carin-bb/history.html	Pris en charge	X	X	X	X	X	X	X	X
Da Vinci Payer Data Exchange	1.0	https://hl7.org/fhir/us/davinci-pdex/	Par défaut	X	X	X	X	X	X	X	X
Da Vinci Payer Data Exchange	2.0 2.1	https://hl7.org/fhir/us/davinci-pdex/history.html	Pris en charge	X	X	X	X	X	X	X	X
Échange de dossiers médicaux Da Vinci (HRex)	0.2	https://hl7.org/fhir/us/davinci-hrex/2020Sep/	Par défaut	X	X	X	X	X	X	X	X
DaVinci PDEX Plan Net	1.1	https://hl7.org/fhir/us/davinci-pdex-plan-net/STU1.1/	Par défaut	X	X	X	X	X	X	X	X

Nom	Ve	Guide d'impléme ntation	Capacité	US Est (Ol	US Est (Vi du No	US Ou (Or	Asi Pa fiqu (Mu	Asi Pa fiqu (Sy	Ca (Ce	Eu (Irl	Europe (Londres)
DaVinci PDEX Plan Net	1.0	https://hl7.org/fhir/us/davinci-pdex-plan-net/STU1/	Pris en charge	X	X	X	X	X	X	X	X
DaVinci Formulaire de médicaments américain Payer Data Exchange (PDex)	1.1	https://hl7.org/fhir/us/davinci-drug-formulary/STU1.1/	Par défaut	X	X	X	X	X	X	X	X
DaVinci Formulaire de médicaments américain Payer Data Exchange (PDex)	1,0 2.0 2.1	https://hl7.org/fhir/us/davinci-drug-formulary/history.html	Pris en charge	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	Par défaut	X	X	X	X	X	X	X	X

Nom	Ve	Guide d'impléme ntation	Capacité	US Est (Ol	US Est (Vi du No	US Ou (Or	Asi Pa fiqu (Mu	Asi Pa fiqu (Sy	Ca (Ce	Eu (Irl	Europe (Londres)
Support d'autoris ation préalable (PAS) Da Vinci FHIR IG	2.1	https://hl7.org/fhir/us/davinci-pas/	Par défaut	X	X	X	X	X	X	X	X
Guide de mise en œuvre du NCQA HEDIS®	0,3	https://www.ncqa.org/resources/hedis-ig-resource-page/	Par défaut	X	X	X	X			X	X
Résumé international des patients (IPS)	bul de vot 2.0	https://hl7.org/fhir/uv/ips/2024Sep/	Par défaut	X	X	X	X	X	X	X	X
Mesure de qualité	5.0	https://registry.fhir.org/package/hl7.fhir.us.cqfmeasures%7C5.0.0	Par défaut	X	X	X	X			X	X
Rapports sur la génomique	3.0	https://build.fhir.org/ig/HL7/genomics-reporting/index.html	Par défaut	X	X	X	X			X	X

Nom	Ve	Guide d'impléme ntation	Capacité	US Est (Ol	US Est (Vi du No	US Ou (Or	Asi Pa fiqu (Mu	Asi Pa fiqu (Sy	Ca (Ce	Eu (Irl	Europe (Londres)
Mission numérique Ayushman Bharat (ABDM) de la National Health Authority	2.0	https://www.nrces.in/ndhm/fhir/r4/index.html	Par défaut	X	X	X	X			X	X
CA Core+	1.1	https://simplifier.net/ca-core	Pris en charge						X		
CA:eReC Pan-Canadian e Referral-eConsult	1.1	https://simplifier.net/CA-eReC/~introduction	Pris en charge						X		
Résumé destiné aux patients, édition canadienne - (PS-CA)	2.1	https://simplifier.net/PS-CA-R1/~introduction	Pris en charge						X		
Répertoire des médicaments de santé numériques de l'Ontario	4.0	https://simplifier.net/ca-on-dhdr-r4/~introduction	Pris en charge						X		

Nom	Ve	Guide d'impléme ntation	Capacité	US Est (Ol	US Est (Vi du No	US Ou (Or	Asi Pa fiqu (Mi	Asi Pa fiqu (Sy	Ca (Ce	Eu (Irl	Europe (Londres)
Noyau UA	1.0	https://hl7.org.au/fhir/core/	Pris en charge					X			
Gestion du cabinet Magentus	1.2	https://fhir-versions.dev.geniesolutions.io/1.2.16/downloads.html	Pris en charge					X			

Validation des profils FHIR spécifiés dans une ressource

Pour qu'un profil FHIR soit validé, ajoutez-le à l'`profile`élément des ressources individuelles en utilisant l'URL du profil indiquée dans le guide de mise en œuvre.

Les profils FHIR sont validés lorsque vous ajoutez une nouvelle ressource à votre banque de données. Pour ajouter une nouvelle ressource, vous pouvez utiliser l'opération `StartFHIRImportJob` API, faire une `POST` demande pour ajouter une nouvelle ressource ou faire `PUT` pour mettre à jour une ressource existante.

Example— Pour voir quel profil FHIR est référencé dans une ressource

L'URL du profil est ajoutée à l'`profile`élément de la paire `"meta" : "profile"` clé-valeur. Cette ressource a été tronquée pour des raisons de clarté.

```
{
    "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— Comment référencer un profil FHIR non pris en charge par défaut

Pour valider par rapport à un profil autre que le profil par défaut pris en charge, ajoutez l'URL du profil versionné à l'`meta.profile`élément. L'URL versionnée inclut l'URL du profil de base suivie d'un tube (`|`) et du numéro de version. Cet exemple de ressource a été tronqué pour des raisons de clarté.

```
{
    "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"
        ]
    }
}
```

Vous pouvez également inclure à la fois l'URL versionnée et l'URL du profil de base. Les deux formats sont valides.

```
{
    "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"
        ]
    }
}
```

Types de ressources pris en charge par FHIR R4 pour HealthLake

Le tableau suivant répertorie les types de ressources FHIR R4 pris en charge par. AWS HealthLake Pour plus d'informations, consultez l'index des ressources dans la documentation du FHIR R4.

Types de ressources FHIR R4 pris en charge par HealthLake

Compte	DetectedIssue	Invoice	Praticien
ActivityDefinition	Appareil	Bibliothèque	PractitionerRole
AdverseEvent	DeviceDefinition	Liaison	Procédure
AllergyIntolerance	DeviceMetric	List	Provenance
Rendez-vous	DeviceUseStatement	Location	Questionnaire
AppointmentResponse	DeviceRequest	Mesure	QuestionnaireResponse
AuditEvent - Voir la note	DiagnosticReport	MeasureReport	RelatedPerson
Binaire	DocumentManifest	Multimédia	RequestGroup
BodyStructure	DocumentReference	Médicaments	ResearchStudy
Bundle - Voir note	EffectEvidenceSynthesis	MedicationAdministration	ResearchSubject
CapabilityStatement	Rencontre	MedicationDispense	RiskAssessment
CarePlan	Endpoint	MedicationKnowledge	RiskEvidenceSynthesis
CareTeam	EpisodeOfCare	MedicationRequest	Planning
ChargeItem	EnrollmentRequest	MedicationStatement	ServiceRequest
ChargeItemDefinition	EnrollmentResponse	MessageHeader	Slot
Demander	ExplanationOfBenefit	MolecularSequence	Spécimen
ClaimResponse	FamilyMemberHistory	NutritionOrder	StructureDefinition
Communication	Indicateur	Observation	StructureMap

CommunicationRequest	Objectif	OperationOutcome - Voir la note	Substance
Montage	Groupe	Organisation	SupplyDelivery
ConceptMap	GuidanceResponse	OrganizationAffiliation	SupplyRequest
Condition	HealthcareService	Paramètres - Voir note	Sous-tâche
Consentement	ImagingStudy	Patient	ValueSet
Contrat	Immunisation	PaymentNotice	VisionPrescription
Couverture	ImmunizationEvaluation	PaymentReconciliation	VerificationResult - Voir la note
CoverageEligibilityRequest	ImmunizationRecommendation	Personne	
CoverageEligibilityResponse	InsurancePlan	PlanDefinition	

⚠ spécifications FHIR et HealthLake

- Vous ne pouvez pas faire `GET` de `POST` demandes avec le FHIR `OperationOutcome` et les types de `Parameters` ressources.
- AuditEvent— Une AuditEvent ressource peut être créée ou lue, mais elle ne peut pas être mise à jour ou supprimée.
- Bundle — Il existe plusieurs façons de HealthLake gérer les demandes de bundle. Pour en savoir plus, consultez Regroupement des ressources FHIR.
- VerificationResult— Ce type de ressource n'est pris en charge que pour les magasins de données créés après le 9 décembre 2023.

Paramètres de recherche FHIR R4 pour HealthLake

Utilisez l'`search`interaction FHIR pour rechercher un ensemble de ressources FHIR dans un magasin de HealthLake données en fonction de certains critères de filtrage. L'`search`interaction peut être effectuée à l'aide d'une `POST` demande `GET` ou. Pour les recherches impliquant des informations personnelles identifiables (PII) ou des informations de santé protégées (PHI), il est recommandé d'utiliser des `POST` demandes, car les informations personnelles et les PHI sont ajoutées dans le corps de la demande et sont cryptées pendant le transfert.

> **Note**
>
> L'`search`interaction FHIR décrite dans ce chapitre est construite conformément à la norme HL7 FHIR R4 pour l'échange de données sur les soins de santé. Comme il s'agit d'une représentation d'un service HL7 FHIR, il n'est pas proposé par le biais de AWS CLI et AWS SDKs. Pour plus d'informations, consultez la `search`documentation de l' RESTful API FHIR R4.
>
> Vous pouvez également interroger les banques de HealthLake données avec SQL à l'aide d'Amazon Athena. Pour plus d'informations, consultez la section Intégration.

HealthLake prend en charge le sous-ensemble suivant de paramètres de recherche FHIR R4. Pour de plus amples informations, veuillez consulter Paramètres de recherche FHIR R4 pour HealthLake.

Types de paramètres de recherche pris en charge

Le tableau suivant indique les types de paramètres de recherche pris en charge dans HealthLake.

Types de paramètres de recherche pris en charge

Paramètre de recherche	Description
_identifiant	ID de ressource (il ne s'agit pas d'une URL complète)
_Dernière mise à jour	Date de dernière mise à jour. Le serveur a le pouvoir discrétionnaire de définir la précision des limites.

Paramètre de recherche	Description
_tag	Effectuez une recherche à l'aide d'une balise de ressource.
_profil	Recherchez toutes les ressources associées à un profil.
_sécurité	Effectuez une recherche sur les étiquettes de sécurité appliquées à cette ressource.
_source	Recherchez d'où vient la ressource.
_texte	Effectuez une recherche sur le récit de la ressource.
createdAt	Recherchez sur l'extension personnalisée CreatedAt.

> **Note**
>
> Les paramètres de recherche suivants ne sont pris en charge que pour les banques de données créées après le 9 décembre 2023 : _security, _source, _text, CreateDat.

Le tableau suivant présente des exemples de modification des chaînes de requête en fonction des types de données spécifiés pour un type de ressource donné. Pour des raisons de clarté, les caractères spéciaux de la colonne des exemples n'ont pas été codés. Pour que la requête soit réussie, assurez-vous que la chaîne de requête a été correctement codée.

Exemples de paramètres de recherche

Types de paramètres de recherche	Détails	Exemples
Number	Recherche une valeur numérique dans une ressource spécifiée. Des chiffres significatifs sont	`[parameter]=100` `[parameter]=1e2`

Types de paramètres de recherche	Détails	Exemples
	observés. Le nombre de chiffres significatifs est défini par la valeur du paramètre de recherche, à l'exception des zéros en tête. Les préfixes de comparaison sont autorisés.	`[parameter]=lt100`

Date/ DateTime	Recherche une date ou une heure spécifique. Le format attendu est `yyyy-mm-ddThh:mm:ss[Z|(+|-)hh:mm]` mais peut varier.
	
	Accepte les types de données suivants : `date`,`dateTime`,`instant`,`Period`,`Timing`. Pour plus de détails sur l'utilisation de ces types de données dans les recherches, consultez la date dans la documentation de l' RESTful API FHIR R4.
	
	Les préfixes de comparaison sont autorisés.
		`[parameter]=eq2013-01-14`
		
		`[parameter]=gt2013-01-14T10:00`
		
		`[parameter]=ne2013-01-14`

Types de paramètres de recherche	Détails	Exemples

String	Recherche une séquence de caractères en distinguant majuscules et minuscules.
	
	Supporte `HumanName` les deux `Address` types. Pour plus de détails, consultez la saisie du type de HumanName données et les entrées du type de `Address` données dans la documentation FHIR R4.
	
	La recherche avancée est prise en charge à l'aide de `:text` modificateurs.
		`[base]/Patient?given=eve`
		
		`[base]/Patient?given:contains=eve`

Jeton	Recherche une close-to-exact correspondance par rapport à une chaîne de caractères, souvent comparée à une paire de valeurs de code médical. La distinction majuscule s/minuscules est liée au système de code utilisé lors de la création d'une requête. Les requêtes basées sur les subsumptions peuvent aider à réduire les problèmes liés à la distinction majuscules/minuscules. Pour plus de clarté, `|` il n'a pas été encodé.	`[parameter]=[system]|[code]` : Ici, il `[system]` fait référence à un système de codage et `[code]` à une valeur de code trouvée dans ce système spécifique. `[parameter]=[code]` : Ici, votre saisie correspondra soit à un code, soit à un système. `[parameter]=|[code]` : Ici, votre entrée correspondra à un code, et la propriété du système n'a aucun identifiant.

Types de paramètres de recherche	Détails	Exemples

Composite	Recherche plusieurs paramètres au sein d'un même type de ressource à l'aide des modificateurs $ et de l'`,` opération.
	
	Les préfixes de comparaison sont autorisés.
		`/Patient?language=FR,NL&language=EN`
		
		`Observation?component-code-value-quantity=http://loinc.org|8480-6$lt60`
		
		`[base]/Group?characteristic-value=gender$mixed`

Quantity	Recherche un nombre, un système et un code sous forme de valeurs. Un numéro est requis, mais le système et le code sont facultatifs. Basé sur le type de données de quantité. Pour plus de détails, consultez la section Quantité dans la documentation du FHIR R4.
	
	Utilise la syntaxe supposée suivante `[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`

Types de paramètres de recherche	Détails	Exemples

Référence	Recherche des références à d'autres ressources.	`[base]/Observation?subject=Patient/23`
		
		`test`

URI	Recherche une chaîne de caractères identifiant sans ambiguïté une ressource particulière.	`[base]/ValueSet?url=http://acme.org/fhir/ValueSet/123`
Spécial	Recherches basées sur des extensions de PNL médicale intégrées.	

Paramètres de recherche avancés pris en charge par HealthLake

HealthLake prend en charge les paramètres de recherche avancée suivants.

Nom	Description	Exemple	Capacité
`_include`	Utilisé pour demander que des ressources supplémentaires soient renvoyées dans une demande de recherche. Elle renvoie les ressources référencées par l'instance de ressource cible.	`Encounter?_include=Encounter:subject`	
`_revinclude`	Utilisé pour demander que des ressources supplémentaires soient renvoyées dans une demande de recherche. Elle renvoie des ressources qui	`Patient?_id=` **`patient-identifier`** `&_revinclude=Encounter:patient`	

Nom	Description	Exemple	Capacité
	font référence à l'instance de ressource principale.		
`_summary`	Le résumé peut être utilisé pour demander un sous-ensemble de la ressource.	`Patient?_summary=text`	Les paramètres récapitulatifs suivants sont pris en charge : `_summary=true` `_summary=false` ,,`_summary=text` ,`_summary=data` .
`_elements`	Demandez qu'un ensemble spécifique d'éléments soit renvoyé dans le cadre d'une ressource dans les résultats de recherche.	`Patient?_elements=identifier,active,link`	
`_total`	Renvoie le nombre de ressources correspondant aux paramètres de recherche.	`Patient?_total=accurate`	Support`_total=accurate` ,`_total=none` .
`_sort`	Indiquez l'ordre de tri des résultats de recherche renvoyés à l'aide d'une liste séparée par des virgules. Le `-` préfixe peut être utilisé pour n'importe quelle règle de tri de la liste séparée par des virgules afin d'indiquer l'ordre décroissant.	`Observation?_sort=status,-date`	Support : tri par champs avec types`Number, String, Quantity, Token, URI, Reference` . Le tri par n'`Date`est pris en charge que pour les magasins de données créés après le 9 décembre 2023. Support jusqu'à 5 règles de tri.
`_count`	Contrôlez le nombre de ressources renvoyées par page du bundle de recherche.	`Patient?_count=100`	La taille de page maximale est de 100.

Nom	Description	Exemple	Capacité
`chainin`	Éléments de recherche des ressources référencées. `.`Dirige la recherche en chaîne vers l'élément de la ressource référencée.	`Diagnosti cReport?s ubject:Pa tient.nam e=peter`	
`reverse chainin (_has)`	Recherchez une ressource en fonction des éléments des ressources qui y font référence.	`Patient?_ has:Obser vation:pa tient:cod e=1234-5`	

`_include`

L'utilisation `_include` dans une requête de recherche permet de renvoyer également des ressources FHIR spécifiées supplémentaires. `_include`À utiliser pour inclure des ressources liées vers le bas.

Example— À utiliser _include pour retrouver les patients ou le groupe de patients chez lesquels une toux a été diagnostiquée

Vous devez effectuer une recherche sur le type de `Condition` ressource en spécifiant le code de diagnostic de la toux, puis en utilisant « `_include` Spécifier que vous souhaitez que le `subject` diagnostic soit également renvoyé ». Dans le type de `Condition` ressource, on `subject` entend soit le type de ressource du patient, soit le type de ressource du groupe.

Pour des raisons de clarté, les caractères spéciaux de l'exemple n'ont pas été codés. Pour réussir une requête, assurez-vous que la chaîne de requête a été correctement encodée.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
Condition?code=49727002&_include=Condition:subject
```

`_include:iterate`Modificateur

Le `_include:iterate` modificateur permet l'inclusion récursive de ressources référencées sur deux niveaux. Par exemple,

```
GET /ServiceRequest?
identifier=025C0931195&_include=ServiceRequest:requester&_include:iterate=PractitionerRole:prac
```

renverra le ServiceRequest, son associé PractitionerRole (via la référence du demandeur), puis inclura de manière récursive le praticien référencé par celui-ci. PractitionerRole Ce modificateur est disponible pour tous les types de ressources dans HealthLake.

`_include=*`Modificateur

Le `_include=*` modificateur est un caractère générique qui inclut automatiquement toutes les ressources directement référencées par les résultats de recherche. Par exemple,

```
GET /ServiceRequest?specimen.accession=12345&_include=*
```

renverra la correspondance ServiceRequest ainsi que toutes les ressources auxquelles elle fait référence (telles que le patient, le praticien, le spécimen, etc.) sans qu'il soit nécessaire de spécifier chaque chemin de référence individuellement. Ce modificateur est disponible pour tous les types de ressources dans HealthLake.

`_revinclude`

L'utilisation `_revinclude` dans une requête de recherche permet de renvoyer également des ressources FHIR spécifiées supplémentaires. `_revinclude`À utiliser pour inclure des ressources liées à l'envers.

Example— À utiliser _revinclude pour inclure des types de ressources de rencontre et d'observation connexes liés à un patient spécifique

Pour effectuer cette recherche, vous devez d'abord définir l'individu `Patient` en spécifiant son identifiant dans le paramètre `_id` de recherche. Ensuite, vous devez spécifier des ressources FHIR supplémentaires à l'aide de la structure `Encounter:patient` et`Observation:patient`.

Pour des raisons de clarté, les caractères spéciaux de l'exemple n'ont pas été codés. Pour réussir une requête, assurez-vous que la chaîne de requête a été correctement encodée.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
Patient?_id=patient-
identifier&_revinclude=Encounter:patient&_revinclude=Observation:patient
```

`_summary`

L'utilisation `_summary` dans une requête de recherche permet à l'utilisateur de demander un sous-ensemble de la ressource FHIR. Il peut contenir l'une des valeurs suivantes : `true, text, data, false` Toutes les autres valeurs seront considérées comme non valides. Les ressources renvoyées seront marquées `'SUBSETTED'` dans le méta.tag, pour indiquer que les ressources sont incomplètes.

- `true`: renvoie tous les éléments pris en charge marqués comme « résumé » dans la définition de base de la ou des ressources.
- `text`: Renvoie uniquement les éléments « text », « id », « méta » et uniquement les éléments obligatoires de haut niveau.
- `data`: Renvoie toutes les parties sauf l'élément « texte ».
- `false`: renvoie toutes les parties de la ou des ressources

Dans une seule demande de recherche, il `_summary=text` ne peut pas être combiné avec `_include` des paramètres `_revinclude` de recherche.

Example— Récupère l'élément « texte » des ressources destinées aux patients dans un magasin de données.

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

`_elements`

L'utilisation `_elements` dans une requête de recherche permet de demander des éléments de ressources FHIR spécifiques. Les ressources renvoyées seront marquées `'SUBSETTED'` dans le méta.tag, pour indiquer que les ressources sont incomplètes.

Le `_elements` paramètre consiste en une liste de noms d'éléments de base séparés par des virgules, tels que les éléments définis au niveau racine de la ressource. Seuls les éléments listés doivent être renvoyés. Si les valeurs des `_elements` paramètres contiennent des éléments non valides, le serveur les ignorera et renverra des éléments obligatoires et des éléments valides.

`_elements`ne sera pas applicable aux ressources incluses (ressources renvoyées dont le mode de recherche est`include`).

Dans une seule demande de recherche, il `_elements` ne peut pas être combiné avec les paramètres `_summary` de recherche.

Example— Obtenez les éléments « identifiant », « actif », « lien » des ressources destinées aux patients dans votre banque de HealthLake données.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?
_elements=identifier,active,link
```

`_total`

L'utilisation `_total` dans une requête de recherche renverra le nombre de ressources correspondant aux paramètres de recherche demandés. HealthLake renverra le nombre total de ressources correspondantes (ressources renvoyées dont le mode de recherche est`match`) dans la réponse `Bundle.total` de recherche.

`_total`prend en charge `accurate` les valeurs des `none` paramètres. `_total=estimate`n'est pas pris en charge. Toutes les autres valeurs seront considérées comme non valides. `_total`n'est pas applicable aux ressources incluses (ressources renvoyées dont le mode de recherche est`include`).

Example— Obtenez le nombre total de ressources pour les patients dans un magasin de données :

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

`_sort`

L'utilisation `_sort` dans la requête de recherche organise les résultats dans un ordre spécifique. Les résultats sont classés par ordre de priorité en fonction de la liste des règles de tri séparées par des virgules. Les règles de tri doivent être des paramètres de recherche valides. Toutes les autres valeurs seront considérées comme non valides.

Dans une seule demande de recherche, vous pouvez utiliser jusqu'à 5 paramètres de recherche de tri. Vous pouvez éventuellement utiliser un `-` préfixe pour indiquer l'ordre décroissant. Le serveur triera par ordre croissant par défaut.

Les types de paramètres de recherche de tri pris en charge sont les suivants :`Number, String, Date, Quantity, Token, URI, Reference`. Si un paramètre de recherche fait référence à un élément imbriqué, il n'est pas pris en charge pour le tri. Par exemple, une recherche sur le « nom » du type de ressource Patient fait référence à l'élément Patient.Name dont le type de HumanName données est considéré comme imbriqué. Par conséquent, le tri des ressources pour les patients par « nom » n'est pas pris en charge.

Example— Accédez aux ressources des patients dans un magasin de données et triez-les par date de naissance dans l'ordre croissant :

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

_count

Le paramètre `_count` est défini comme une instruction au serveur concernant le nombre de ressources à renvoyer sur une seule page.

La taille de page maximale est de 100. Toute valeur supérieure à 100 n'est pas valide. `_count=0`n'est pas pris en charge.

Example— Recherchez la ressource Patient et définissez la taille de la page de recherche sur 25 :

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?_count=25
```

Chaining and Reverse Chaining(_has)

Le chaînage et le chaînage inversé dans FHIR constituent un moyen plus efficace et plus compact d'obtenir des données interconnectées, réduisant ainsi le besoin de plusieurs requêtes distinctes et rendant la récupération de données plus pratique pour les développeurs et les utilisateurs.

Si un niveau de récursivité renvoie plus de 100 résultats, il HealthLake renverra 4xx pour éviter que le magasin de données ne soit surchargé et ne provoque des paginations multiples.

Example— Chainage - Obtient tout DiagnosticReport ce qui fait référence à un patient dont le nom est Peter.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/DiagnosticReport?
subject:Patient.name=peter
```

Example— Chainage inversé - Accédez aux ressources du patient, où la ressource du patient est référencée par au moins une observation dont le code de l'observation est 1234, et où l'observation fait référence à la ressource du patient dans le paramètre de recherche du patient.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?
_has:Observation:patient:code=1234
```

Modificateurs de recherche pris en charge

Les modificateurs de recherche sont utilisés avec les champs basés sur des chaînes. Tous les modificateurs de recherche HealthLake utilisés utilisent une logique booléenne. Par exemple, vous pouvez `:contains` spécifier qu'un champ de chaîne plus grand doit inclure une petite chaîne pour qu'il soit inclus dans les résultats de recherche.

Modificateurs de recherche pris en charge

Modificateur de recherche	Type
:manquant	Tous les paramètres sauf `Composite`
:exact	String
:contient	String
:non	Jeton
:texte	Jeton
:identifiant	Référence
:ci-dessous	URI

Comparateurs de recherche pris en charge

Vous pouvez utiliser des comparateurs de recherche pour contrôler la nature de la correspondance lors d'une recherche. Vous pouvez utiliser des comparateurs lorsque vous effectuez une recherche dans les champs de numéro, de date et de quantité. Le tableau suivant répertorie les comparateurs de recherche et leurs définitions pris en charge par HealthLake.

Comparateurs de recherche pris en charge

Comparateur de recherche	Description
eq	La valeur du paramètre dans la ressource est égale à la valeur fournie.

Comparateur de recherche	Description
ne	La valeur du paramètre dans la ressource n'est pas égale à la valeur fournie.
gt	La valeur du paramètre dans la ressource est supérieure à la valeur fournie.
lt	La valeur du paramètre dans la ressource est inférieure à la valeur fournie.
gm	La valeur du paramètre dans la ressource est supérieure ou égale à la valeur fournie.
le	La valeur du paramètre dans la ressource est inférieure ou égale à la valeur fournie.
sa	La valeur du paramètre dans la ressource commence après la valeur fournie.
eb	La valeur du paramètre dans la ressource se termine avant la valeur fournie.

Les paramètres de recherche FHIR ne sont pas pris en charge par HealthLake

HealthLake prend en charge tous les paramètres de recherche FHIR à l'exception de ceux répertoriés dans le tableau suivant. Pour une liste complète des paramètres de recherche FHIR, consultez le registre des paramètres de recherche FHIR.

Paramètres de recherche non pris en charge

Composition du bundle	Lieu : à proximité
Identifiant du bundle	C onsent-source-reference
Message groupé	Patient sous contrat
Type d'offre groupée	Contenu des ressources
Horodatage du bundle	Requête de ressources

FHIR R4 pour **`$operations`** HealthLake

Les opérations FHIR $ (également appelées « opérations dollar ») sont des fonctions spéciales côté serveur qui vont au-delà des opérations CRUD (`Create`,, `ReadUpdate`,`Delete`) standard de la spécification FHIR. Ces opérations sont identifiées par le préfixe « $ » et permettent un traitement complexe, une transformation de données et des opérations en masse qui seraient difficiles ou inefficaces à effectuer à l'aide d'appels d'API REST standard. Les opérations $ peuvent être invoquées au niveau du système, au niveau du type de ressource ou sur des instances de ressources spécifiques, offrant ainsi des moyens flexibles d'interagir avec les données FHIR. AWS HealthLake prend en charge plusieurs FHIR`$operations`. Veuillez vous référer à chacune des pages ci-dessous pour plus de détails.

Rubriques

Fonctionnement du FHIR R4 **`$attribution-status`** pour HealthLake

Récupère le statut d'attribution d'un membre spécifique, renvoyant un bundle contenant toutes les ressources d'attribution liées au patient. Cette opération fait partie de la mise en œuvre de la version 2.1.0 de la liste d'attribution des membres FHIR (ATR) List IG.

Endpoint

```
POST [base]/Group/[id]/$attribution-status
```

Paramètres de demande

L'opération accepte les paramètres facultatifs suivants :

Paramètre	Type	Description
memberId	Identifiant	Le MemberId membre pour lequel le statut d'attribution est demandé
Référence du patient	Référence	Référence à la ressource pour les patients dans le système du producteur

> **Note**
>
> L'un `memberId` ou l'autre `patientReference` peut être fourni, ou les deux à des fins de validation.

Exemple de demande

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "memberId",
      "valueIdentifier": {
        "system": "http://example.org",
        "value": "MBR123456789"
```

```
      }
    },
    {
      "name": "patientReference",
      "valueReference": {
        "reference": "Patient/patient-123",
        "display": "John Doe"
      }
    }
  ]
}
```

Réponse

Renvoie un bundle contenant des ressources d'attribution liées au patient :

Ressource	Cardinalité	Location
Patient	1..1	Groupe.membre/entité
Couverture	0,1	Group.Member.Extension : CoverageReference
Organization/Practitioner/PractitionerRole	0,1	group.member.extension : fournisseur attribué
N'importe quelle ressource	0,1	group.member.extension : données associées

Exemple de réponse

```
{
  "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
      }
    }
  ]
}
```

Gestion des erreurs

L'opération gère les conditions d'erreur suivantes :

Erreur	État du protocole HTTP	Description
Demande d'opération non valide	400	Paramètres ou structure de demande non conformes
Ressource de groupe introuvable	404	L'ID de groupe spécifié n'existe pas
Ressource pour les patients introuvable	404	La référence du patient spécifiée n'existe pas

Autorisation et sécurité

Exigences relatives à SMART Scope

Les clients doivent disposer des privilèges appropriés pour lire les ressources du groupe et les ressources d'attribution associées

Les mécanismes d'autorisation FHIR standard s'appliquent à toutes les opérations

Supprimer des types de ressources avec **`$bulk-delete`**

AWS HealthLake prend en charge l'`$bulk-delete`opération en permettant de supprimer toutes les ressources d'un type spécifique au sein d'une banque de données. Cette opération est particulièrement utile lorsque vous devez :

- Réaliser un audit saisonnier et un nettoyage
- Gérez le cycle de vie des données à grande échelle
- Supprimer des types de ressources spécifiques
- Respectez les politiques de conservation des données

Usage

L'`$bulk-delete`opération peut être invoquée à l'aide des méthodes POST :

```
POST [base]/[ResourceType]/$bulk-delete?isHardDelete=false&deleteAuditEvent=true
```

Parameters

Paramètre	Type	Obligatoire	Par défaut	Description
`isHardDelete`	booléen	Non	false	Lorsque c'est vrai, supprime définitivement les ressources du stockage
`deleteAuditEvent`	boolean	Non	true	Lorsque c'est vrai, supprime les événements d'audit associés
`_since`	chaîne	Non	Heure de création de la banque de données	Une fois saisie, sélectionne l'heure limite de début pour rechercher les ressources en fonction de leur heure de dernière modification. Ne peut pas être utilisé avec le début ou la fin
`start`	chaîne	Non	Heure de création de la banque de données	Une fois saisie, sélectionne l'heure limite pour rechercher les ressources en fonction de leur heure de dernière modification. Peut être utilisé avec extrémité
`end`	chaîne	Non	Heure de soumission des offres d'emploi	Une fois saisi, sélectionne l'heure limite de fin pour rechercher les ressources en fonction de leur heure de dernière modification

Exemples

Exemple de requête

```
POST [base]/Observation/$bulk-delete?isHardDelete=false
```

Exemple de réponse

```
{
      "jobId": "jobId",
```

```
      "jobStatus": "SUBMITTED"
    }
```

Statut de la tâche

Pour vérifier le statut d'une tâche de suppression en bloc, procédez comme suit :

```
GET [base]/$bulk-delete/[jobId]
```

L'opération renvoie les informations relatives à l'état de la tâche :

```
{
      "datastoreId": "datastoreId",
      "jobId": "jobId",
      "status": "COMPLETED",
      "submittedTime": "2025-10-09T15:09:51.336Z"
    }
```

Comportement

L'`$bulk-delete`opération :

1. Processus asynchrones pour gérer de gros volumes de ressources
2. Maintient les transactions ACID pour garantir l'intégrité des données
3. Fournit un suivi de l'état des tâches avec le nombre de ressources supprimées
4. Supporte les modes de suppression souple et rigide
5. Inclut un enregistrement d'audit complet des activités de suppression
6. Permet la suppression sélective des versions historiques et des événements d'audit

Journalisation des audits

L'`$bulk-delete`opération est enregistrée sous les noms Start FHIRBulk DeleteJob et Describe FHIRBulk DeleteJob avec des informations détaillées sur l'opération.

Limitations

- Lorsque `isHardDelete` ce paramètre est défini sur true, les ressources supprimées définitivement n'apparaissent pas dans les résultats de recherche ni dans les `_history` requêtes.

- Les ressources supprimées par cette opération peuvent être temporairement inaccessibles pendant le traitement
- La mesure du stockage est ajustée uniquement sur les versions historiques - deleteVersionHistory =false n'ajustera pas le stockage de la banque de données

`opération $bulk-member-match` pour HealthLake

AWS HealthLake prend en charge l'`$bulk-member-match`opération de traitement asynchrone des demandes de correspondance de plusieurs membres. Cette opération permet aux établissements de santé de faire correspondre efficacement les identifiants uniques de centaines de membres dans différents systèmes de santé en utilisant des informations démographiques et de couverture dans une seule demande groupée. Cette capacité est essentielle pour l'échange de données de payeur à payeur à grande échelle, les transitions de membres et les exigences de conformité du CMS et est conforme à la spécification FHIR.

> **Note**
>
> L'`$bulk-member-match`opération est basée sur une spécification FHIR sous-jacente qui est actuellement expérimentale et sujette à modification. Au fur et à mesure de l'évolution des spécifications, le comportement et l'interface de cette API seront mis à jour en conséquence. Il est conseillé aux développeurs de surveiller les notes de HealthLake publication d'AWS et les mises à jour pertinentes des spécifications FHIR afin de rester informés de toute modification susceptible d'avoir un impact sur leurs intégrations.

Cette opération est particulièrement utile lorsque vous devez :

- Procédez au jumelage des membres à grande échelle pendant les périodes d'inscription ouvertes
- Faciliter les transitions de membres en masse entre les payeurs
- Support des exigences d'échange de données de conformité des CMS à grande échelle
- Associez efficacement les cohortes de membres à travers les réseaux de santé
- Minimisez les appels d'API et améliorez l'efficacité opérationnelle pour les scénarios de mise en correspondance de volumes élevés

Usage

L'`$bulk-member-match`opération est une opération asynchrone invoquée sur les ressources du groupe à l'aide de la méthode POST :

```
POST [base]/Group/$bulk-member-match
```

Après avoir soumis une demande de correspondance groupée, vous pouvez consulter le statut du poste en utilisant :

```
GET [base]/$bulk-member-match-status/{jobId}
```

Paramètres pris en charge

HealthLake prend en charge les `$bulk-member-match` paramètres FHIR suivants :

Paramètre	Type	Obligatoire	Description
`MemberPatient`	Patient	Oui	Ressource destinée aux patients contenant des informations démographiques pour le membre à jumeler.
`CoverageToMatch`	Couverture	Oui	Ressource de couverture qui sera utilisée pour la comparaison avec les enregistrements existants.
`CoverageToLink`	Couverture	Non	Ressource de couverture à associer pendant le processus de mise en correspondance.
`Consent`	Consentement	Oui	La ressource de consentement à des fins d'autorisation est également stockée. Cela est différent de l'`$member-match` opération individuelle pour laquelle le consentement n'est pas requis.

Demande POST pour soumettre en masse un job correspondant à un nombre de membres

L'exemple suivant montre une demande POST visant à soumettre un job de mise en relation groupé de membres. Chaque membre est encapsulé dans un `MemberBundle` paramètre contenant

les Consent ressources requises MemberPatientCoverageToMatch, ainsi qu'un paramètre facultatifCoverageToLink.

```
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"
              }
            ]
          }
        }
      ]
    }
  ]
}
```

Réponse à la tâche terminée avec sortie

Lorsque la tâche est terminée, la réponse inclut les métadonnées de la tâche et une ressource de paramètres FHIR contenant trois ressources de groupe qui catégorisent les résultats du match.

```
{
  "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"
              }
            }
          ]
        }
      }
    ]
  }
}
```

Comment HealthLake classifie les membres en groupes de sortie

Chaque membre soumis dans une `$bulk-member-match` demande est évalué par le biais d'un pipeline séquentiel. Le résultat de chaque étape détermine le groupe de sortie dans lequel le membre est placé.

1. Validation structurale — MemberBundle Sont-ils conformes aux profils requis ? En cas d'échec : erreur (dans aucun groupe).
2. Correspondance des patients — Est-il possible de HealthLake trouver des patients correspondant aux données démographiques soumises ? En cas de panne : NonMatchedMembers.
3. Confirmation de couverture — Est-il possible de se HealthLake limiter à un seul patient valide CoverageToMatch ? En cas de panne : NonMatchedMembers.
4. Évaluation du consentement — Le consentement soumis est-il honorable en ce moment ? (statut = actif, la période couvre la date actuelle, l'interprète peut être validé). En cas de panne : ConsentConstrainedMembers.
5. Succès — Tous les chèques sont réussis. Consentement stocké dans une banque de données. Membre inscrit MatchedMembers.

Principe de base : un membre ne peut apparaître que dans une seule destination. La première étape défaillante détermine le placement. Les membres qui échouent aux étapes 2 ou 3 ne sont jamais inclus. Ce groupe est exclusivement réservé aux membres qui ont réussi le jumelage mais dont le consentement ne peut être honoré. ConsentConstrainedMembers

Détails de l'évaluation du consentement (étape 4) :

- Contrôle 1 — État du consentement : est-il `Consent.status` égal à « actif » ? Sinon → ConsentConstrainedMembers.
- Contrôle 2 — Période de mise à disposition : `provision.period` couvre la date actuelle ? Si la date actuelle est antérieure `period.start` ou postérieure `period.end` → ConsentConstrainedMembers.
- Contrôle 3 — Validation du performeur : la `Consent.performer` référence peut-elle être validée ? Si la ressource référencée ne se trouve pas dans la banque de données ou n'est pas associée au patient correspondant → ConsentConstrainedMembers.

Toutes les vérifications doivent être passées pour que le membre soit inscrit MatchedMembers et pour que le consentement soit conservé.

Comportement de correspondance de couverture

Lors de la mise en correspondance des membres, seule la validation `CoverageToMatch` est effectuée par rapport à la banque de données du payeur répondant. `CoverageToLink`appartient au new/requesting payeur et n'est pas validé par rapport à la banque de données de l'ancien payeur. L'inclusion `CoverageToLink` dans la demande n'affectera pas les résultats correspondants.

Chaque combinaison patient+couverture figurant dans la demande est traitée indépendamment. Le même patient peut être soumis plusieurs fois avec différents plans de couverture, et chaque inscription reçoit son propre résultat en fonction de sa couverture spécifique.

Gestion des références aux exécutants du consentement

Le nouveau payeur peut envoyer une référence de patient temporaire ou locale `Consent.performer` (par exemple, la même référence utilisée dans`Consent.patient`). HealthLake résout automatiquement ces références :

- S'il `Consent.performer` contient la même référence locale que`Consent.patient`, il la HealthLake remplace par la référence réelle du patient correspondant une fois l'appariement réussi.
- HealthLake prend en charge les références d'intervenants de type Patient RelatedPerson PractitionerRole, Practitioner et Organization (références directes et références d'identification logiques).

- Si la validation de l'intervenant échoue (ressource introuvable ou non associée au patient correspondant), le membre est intégré au ConsentConstrainedMembers lieu de renvoyer une erreur.

Ressources du groupe de sortie

La tâche terminée renvoie une ressource Parameters contenant trois ressources de groupe :

MatchedMembers Groupe

Contient les références des patients pour tous les membres correspondants dont le consentement est actif et valide au moment de la demande. La ressource Consent est créée et stockée dans la banque de données pour chaque membre correspondant. Ce groupe est instancié dans la banque de données et peut être utilisé directement avec. `$davinci-data-export`

NonMatchedMembers Groupe

Contient des références à des membres pour lesquels aucune correspondance unique n'a été trouvée. Un membre est placé ici lorsqu'aucun patient de la banque de données ne correspond aux données démographiques fournies, qu'il n'existe aucune couverture valide pour un patient candidat correspondant, ou lorsque plusieurs patients correspondent aux données démographiques et que plusieurs d'entre eux ont une couverture valide (ambigu).

ConsentConstrainedMembers Groupe

Contient des références de patients pour les membres qui ont été jumelés avec succès (données démographiques et couverture confirmées) mais dont le consentement ne peut être honoré au moment de la demande. La ressource de consentement n'est pas stockée pour les membres soumis à des contraintes de consentement. L'identité du membre correspondant (MemberIdentifier et MemberId) est toujours incluse afin que le payeur demandeur sache qui a été contraint.

Le `Group.quantity` champ contient le nombre total de membres dans chacun de leurs groupes respectifs.

Références des membres du groupe :

- `Group.member.entity.reference`— Pour MatchedMembers et ConsentConstrainedMembers, contient l'identifiant patient du membre correspondant dans le

système du payeur répondant. Pour NonMatchedMembers, fait référence à l'entrée Patient contenue.

- `Group.member.entity.extension (base-ext-match-parameters)`— Contient l'identifiant du patient issu de la demande d'entrée initiale (l'identifiant soumis par le payeur demandeur`Patient.id`, dérivé de`Coverage.beneficiary.reference`, ou`Consent.patient.reference`).

Consent-Patient lien

> **Important**
>
> La ressource de consentement stockée conserve la référence du patient exactement telle qu'elle a été soumise par le payeur demandeur. HealthLake ne met pas automatiquement à jour le champ patient du consentement pour qu'il pointe vers le patient correspondant dans la banque de données réceptrice.

Pour lier un consentement enregistré au patient correspondant, utilisez le résultat de la tâche : chaque membre du MatchedMembers groupe `member.entity.reference` pointe vers le patient correspondant et un `member.entity.extension` (`base-ext-match-parameters`) pointant vers le patient saisi contenu. Cross-reference ceux-ci avec le champ patient du consentement pour créer le mappage dans votre couche d'application.

Ce qui est stocké par rapport à ce qui est transitoire

Le tableau suivant décrit ce qui est HealthLake conservé dans la banque de données pendant le `$bulk-member-match` traitement et ce qui n'existe que dans la réponse à la tâche :

Ressource	Stocké ?	Interrogable via REST ?	Remarques
MemberPatient (entrée)	Non	Non	Utilisé uniquement pour la mise en correspondance ; non persistant
CoverageToMatch (entrée)	Non	Non	Utilisé uniquement pour confirmer la couverture

Ressource	Stocké ?	Interrogable via REST ?	Remarques
CoverageToLink (entrée)	Non	Non	Non validé par rapport à la banque de données ; appartient au nouveau payeur
Consentement (membres correspondants)	Oui	Oui — GET [base] / Consent/ {id}	Stocké tel qu'il a été reçu du payeur demandeur
Consentement (membres limités)	Non	Non	Non stocké. L'identité du membre est toujours incluse dans la réponse.
MatchedMembers Groupe (sortie)	Oui	Oui — GET [base] / Group/ {id}	Instancié ; utilisable avec $davinci-data-export
NonMatchedMembers Groupe	Non	Non	Réponse au job uniquement
ConsentConstrained Members Groupe	Non	Non	Réponse au job uniquement

Intégration avec $davinci-data-export

La ressource de MatchedMembers groupe renvoyée par `$bulk-member-match` peut être directement utilisée avec l'`$davinci-data-export`opération de récupération des données des membres en masse :

```
POST [base]/Group/{matched-group-id}/$davinci-data-export
GET [base]/Group/{matched-group-id}
```

Cette intégration permet des flux de travail efficaces dans lesquels vous identifiez d'abord les membres correspondants en masse, puis exportez leurs dossiers médicaux complets à l'aide de la ressource de groupe qui en résulte.

Utiliser $member-remove avant l'exportation

Si vous devez exclure des membres spécifiques de l'exportation après le rapprochement (par exemple, un membre révoque son consentement entre le rapprochement et l'exportation), utilisez-le `$member-remove` sur le MatchedMembers groupe.

> **Important**
>
> La suppression d'un membre `$member-remove` marque le membre comme inactif dans le groupe, mais exclut `$davinci-data-export` uniquement les membres inactifs une fois que le groupe est passé au statut « final ». Si vous faites appel `$davinci-data-export` à un groupe qui possède toujours le statut par défaut, les membres supprimés peuvent toujours apparaître dans les résultats d'exportation.

Flux de travail :

1. `POST [base]/Group/{id}/$member-remove`— marque les membres comme inactifs
2. `PUT [base]/Group/{id}`— mettre à jour le statut du groupe à « final »
3. `POST [base]/Group/{id}/$davinci-data-export`— l'export exclut désormais les membres supprimés

Caractéristiques de performance

L'`$bulk-member-match`opération est conçue pour le traitement de gros volumes et s'exécute de manière asynchrone :

- Simultanéité : maximum de 5 opérations simultanées par magasin de données.
- Évolutivité : gère jusqu'à 500 membres par demande (limite de charge utile de 5 Mo).
- Opérations parallèles : compatible avec les opérations simultanées d'importation, d'exportation ou de suppression en bloc.

Autorisation

L'API utilise le protocole d'autorisation SMART on FHIR avec les étendues requises suivantes :

- `system/Patient.read`— Nécessaire pour rechercher et associer les ressources destinées aux patients.
- `system/Coverage.read`— Nécessaire pour valider les informations de couverture.
- `system/Group.write`— Nécessaire pour créer des ressources de groupe de résultats.
- `system/Organization.read`— Conditionnel, obligatoire si la couverture fait référence à des organisations.
- `system/Practitioner.read`— Conditionnel, obligatoire si la couverture fait référence à des praticiens.
- `system/PractitionerRole.read`— Conditionnel, obligatoire si la couverture fait référence aux rôles des praticiens.
- `system/Consent.write`— Conditionnel, obligatoire si des ressources de consentement sont fournies.

L'opération prend également en charge AWS l'autorisation IAM Signature Version 4 (SigV4) pour l'accès programmatique.

Règles de validation

Les règles de validation suivantes sont appliquées à chacune MemberBundle à l'étape 1. Les membres dont la validation échoue sont signalés comme des erreurs et n'apparaissent dans aucun groupe de sortie.

MemberPatient

Champ	Comment HealthLake l'utilise	Échec de validation si...
`name.family`	Recherche démographique	Manquant
`name.given`	Recherche démographique	Manquant (au moins un élément requis)
`birthDate`	Recherche démographique	Manquant

Champ	Comment HealthLake l'utilise	Échec de validation si...
`gender`	Recherche démographique ; en cas d'absence, extension Birth Sex utilisée	Ni sexe ni sexe de naissance présents (hrex-pat-1)
`identifier`	Inclus dans la recherche lorsqu'il est présent ; améliore la confiance	Ne cause jamais de panne (facultatif)

CoverageToMatch / CoverageToLink

Champ	Comment HealthLake l'utilise	Échec de validation si...
`status`	Confirme que la couverture est exploitable	Manquant
`beneficiary`	Associe la couverture à un patient candidat	Manquant
`payor`	Homonymie lorsqu'il existe plusieurs candidats	Absent ou plus d'un payeur
`relationship`	Confirme la relation abonné-bénéficiaire	Manquant
`identifier (MB) or subscriberId`	Clé de désambiguïsation principale	Aucun des deux

Consentement

Champ	Comment HealthLake l'utilise	Échec de validation si...
`scope`	Confirme que la portée du consentement est axée sur la confidentialité des patients	Code de confidentialité des patients manquant ou inexistant

Champ	Comment HealthLake l'utilise	Échec de validation si...
`category`	Confirme la classification des informations	Manquant
`patient`	Identifie le sujet du consentement	Manquant
`performer`	Identifie qui est d'accord	Manquant
`sourceReference`	Documente la source du consentement	Manquant
`policy.uri`	Détermine l'étendue du partage de données	URI manquant ou ne se terminant pas par #regular ou #sensitive
`provision.type`	Doit être un « permis » selon le profil de consentement HRex	« Permis » manquant ou non (y compris « refus »)
`provision .period`	Évalué à l'étape 4 pour une vérification sous réserve de consentement	Manquant ou non start/end
`status`	Évalué à l'étape 4 (PAS à l'étape 1)	Ne cause jamais d'échec à l'étape 1 : HealthLake accepte tout statut valide et évalue à l'étape 4

> **Note**
>
> Le profil de consentement HRex définit le statut avec une valeur fixe de « actif ». HealthLake assouplit intentionnellement cette contrainte afin qu'un consentement non actif reçoive une classification significative (ConsentConstrainedMembers) plutôt qu'un rejet de validation général.

Comportement correspondant

- Recherche d'un patient (étape 2) : HealthLake recherche en utilisant `name.family` + `name.given` (exact, sans distinction `birthDate` majuscules/majuscules), `gender` (exact ; sexe de naissance utilisé en l'absence de sexe) et `identifier` (le cas échéant, facultatif).
- Désambiguïsation de la couverture (étape 3) — Lorsque plusieurs patients candidats sont trouvés, elle `CoverageToMatch` est utilisée pour se limiter à un seul. Une couverture est « valide » lorsqu'une ressource de couverture active existe dans la banque de données correspondant à `subscriberId` ou `identifier` (type MB) ETpayor.
- Évaluation du consentement (étape 4) — Effectuée uniquement après une correspondance unique réussie. Consultez la section sur les détails de l'évaluation du consentement ci-dessus.

Gestion des erreurs

L'opération gère les conditions d'erreur suivantes :

- 400 Mauvaise demande : format de demande non valide, paramètres requis manquants ou charge utile supérieure aux limites de taille (500 membres ou 5 Mo).
- 422 Entité intraitable : erreurs de traitement lors de l'exécution de la tâche.
- Erreurs relatives à un membre individuel : lorsqu'un membre en particulier échoue, l'opération se poursuit avec les membres restants et inclut les détails de l'erreur dans le NonMatchedMembers groupe avec les codes de motif appropriés. Par exemple, un `MemberBundle` Patient dont le `birthDate` paramètre est absent renverra le message d'erreur suivant :

```
"errors": [
  {
    "memberIndex": 1,
    "jsonBlob": {
      "resourceType": "OperationOutcome",
      "issue": [
        {
          "severity": "error",
          "code": "invalid",
          "diagnostics": "MemberPatient.birthDate is required"
        }
      ],
      "statusCode": 400
    }
  }
```

```
]
```

Les détails de l'erreur sont disponibles via le point de terminaison du sondage d'état et incluent :

- `numberOfMembersWithCustomerError`: Nombre de membres présentant des erreurs de validation ou de saisie.
- `numberOfMembersWithServerError`: nombre de membres présentant des erreurs de traitement côté serveur.

Opérations liées

- the section called “$member-match”— Opération de mise en correspondance de membres individuels.
- the section called “$davinci-data-export”— Exportation de données en masse à l'aide des ressources du groupe.
- the section called “$ Opérations”— Liste complète des opérations prises en charge.

Fonctionnement du FHIR R4 **`$confirm-attribution-list`** pour HealthLake

Indique au producteur que le consommateur n'a plus aucune modification à apporter à la liste d'attribution. Cette opération finalise la liste d'attribution en supprimant les membres inactifs de la liste, en modifiant le statut en « final » et en accusant réception de l'opération. Cette opération fait partie de la mise en œuvre de la version 2.1.0 de la liste d'attribution des membres FHIR (ATR) List IG.

Endpoint

```
POST [base]/Group/[id]/$confirm-attribution-list
```

Demande

Aucun paramètre n'est requis.

```
POST [base]/Group/[id]/$confirm-attribution-list
```

Réponse

Renvoie HTTP 201 avec un message de confirmation.

Exemple de réponse

```
HTTP Status Code: 201

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "message",
      "valueString": "Accepted."
    }
  ]
}
```

État du groupe après confirmation

Une fois la confirmation réussie, le statut de la liste d'attribution de la ressource du groupe sera défini sur « final » :

```
{
  "resourceType": "Group",
  "id": "fullexample",
  "extension": [
    {
      "url": "http://hl7.org/fhir/us/davinci-atr/StructureDefinition/ext-
attributionListStatus",
      "valueCode": "final"
    }
  ]
  // ... other Group properties
}
```

Gestion des erreurs

L'opération gère les conditions d'erreur suivantes :

Erreur	État du protocole HTTP	Description
Demande d'opération non valide	400	Paramètres ou structure de demande non conformes
Ressource de groupe introuvable	404	L'ID de groupe spécifié n'existe pas

Autorisation et sécurité

Exigences relatives à SMART Scope

Les clients doivent disposer des privilèges appropriés pour lire les ressources du groupe et les ressources d'attribution associées

En effet`$confirm-attribution-list`, les clients doivent également disposer de privilèges d'écriture pour modifier les ressources du groupe

Les mécanismes d'autorisation FHIR standard s'appliquent à toutes les opérations

Fonctionnement du FHIR R4 **`$davinci-data-export`** pour HealthLake

Il s'agit d'une `$davinci-data-export` opération FHIR asynchrone à partir de laquelle vous pouvez exporter des données de santé. AWS HealthLake Cette opération prend en charge plusieurs types d'exportation, notamment l'attribution de membres (ATR) Payer-to-Payer, l'accès au PDex fournisseur et l'accès aux membres APIs. Il s'agit d'une version spécialisée de l'`$export`opération FHIR standard, conçue pour répondre aux exigences des guides de mise DaVinci en œuvre.

Fonctionnalités principales

- Traitement asynchrone : suit le modèle de demande asynchrone FHIR standard
- Exportation au niveau du groupe : exporte les données pour les membres d'une ressource de groupe spécifique
- Plusieurs types d'exportation : prend en charge l'ATR (attribution de membres) Payer-to-Payer, l'accès aux PDex fournisseurs et l'accès aux membres APIs
- Support complet pour les profils : inclut US Core, CARIN Blue Button et PDex profils

- Filtrage flexible : prend en charge le filtrage par patients, types de ressources et plages de temps
- Sortie NDJSON : fournit des données au format JSON délimité par de nouvelles lignes

Opération Endpoint

```
GET [base]/Group/[id]/$davinci-data-export
POST [base]/Group/[id]/$davinci-data-export
```

Paramètres de demande

Paramètre	Cardinalité	Description
`patient`	0.. *	Membres spécifiques dont les données doivent être exportées. En cas d'omission, tous les membres du groupe sont exportés.
`_type`	0,1	Liste séparée par des virgules des types de ressources FHIR à exporter. En cas d'omissio n, tous les types de ressources pris en charge pour le type d'exportation spécifié sont inclus. Pour les exportations ATR, les 8 types de ressources d'attribution sont utilisés par défaut. Pour les PDex exportations, cela inclut tous les types de ressources d'attribution, ainsi que les types de ressources cliniques et de réclamati ons provenant du US Core, de CARIN Blue Button et PDex des profils.
`_since`	0,1	N'incluez que les ressources mises à jour après cette date et cette heure.
`_until`	0,1	N'incluez que les ressources mises à jour avant cette date et cette heure.
`exportType`	0,1	Type d'export à effectuer. Valeurs valides: `hl7.fhir.us.davinci-atr` , `hl7.fhir. us.davinci-pdex` , `hl7.fhir. us.davinci-pdex.p2p` , `hl7.fhir.`

Paramètre	Cardinalité	Description
		us.davinci-pdex.member . Valeur par défaut : hl7.fhir.us.davinci-atr .
_includeEOB2xWoFinancial	0,1	Spécifie s'il faut inclure les ExplanationOfBenefit ressources CARIN BB 2.x avec les données financières supprimées. Valeur par défaut : false.
_security	0.. *	Filtrez les ressources exportées par valeurs de meta.security codage. Utilisez le system|code format (le caractère en forme de tube doit être codé en URL sous %7C la forme). Lorsque plusieurs valeurs sont fournies, les ressources doivent correspondre à toutes (sémantique ET). Utilisez system| (tube final, pas de code) pour faire correspondre n'importe quel code d'un système donné.
_tag	0.. *	Filtrez les ressources exportées par valeurs de meta.tag codage. Utilise le même system|code format et la même sémantique AND que. _security Lorsque _security les deux filtres _tag sont spécifiés, les ressources doivent correspondre aux deux filtres.

Types de ressource pris en charge

Les types de ressources pris en charge dépendent du type d'exportation que vous spécifiez. Pour les exportations ATR, les types de ressources suivants sont pris en charge :

- Group
- Patient
- Coverage
- RelatedPerson
- Practitioner

- `PractitionerRole`
- `Organization`
- `Location`

Pour les PDex exportations (accès fournisseur et accès membre), tous les types de ressources cliniques et de réclamations sont pris en charge en plus des types précédents. Payer-to-Payer Pour une liste complète des types de ressources pris en charge, consultez le US Core Implementation Guide (STU 6.1), le CARIN Blue Button Implementation Guide et le Da Vinci Prior Authorization Support Implementation Guide.

Types d'exportation

L'`$davinci-data-export`opération prend en charge les types d'exportation suivants. Vous spécifiez le type d'exportation à l'aide du `exportType` paramètre.

Type d'exportation	Objectif	Étendue des données	Limite temporelle
`hl7.fhir.us.davinci-atr`	Liste d'attribution des membres	Ressources liées à l'attribution	Aucune
`hl7.fhir.us.davinci-pdex`	API d'accès aux fournisseurs	Données cliniques et relatives aux demandes de remboursement pour les patients concernés	5 ans
`hl7.fhir.us.davinci-pdex.p2p`	Payer-to-Payer Échange	Données historiques sur les membres pour les transitions d'assurance	5 ans
`hl7.fhir.us.davinci-pdex.member`	API d'accès aux membres	Données de santé propres au membre	5 ans

> **Note**
>
> Pour les PDex exportations, la limite temporelle de 5 ans ne s'applique pas aux types de ressources ATR

(`GroupPatient`,`Coverage`,`RelatedPerson`,`Practitioner`,`PractitionerRole`,`Organization`
Ces ressources sont toujours incluses quel que soit l'âge.

ATR (hl7.fhir.us.davinci-atr)

Avec le type d'exportation ATR, vous pouvez exporter les données de la liste d'attribution des membres. Utilisez ce type d'exportation pour récupérer les ressources liées à l'attribution pour les membres d'un groupe. Pour plus d'informations, consultez l'opération d'exportation de Da Vinci ATR.

Types de ressource pris en charge

`Group`, `Patient`, `Coverage`, `RelatedPerson`, `Practitioner`, `PractitionerRole`, `Organization`, `Location`

Filtrage temporel

Aucun filtrage temporel n'est appliqué. Toutes les ressources correspondantes sont exportées quelle que soit la date.

PDex Types d'exportation

Tous les types PDex d'exportation partagent les mêmes profils pris en charge et la même logique de filtrage. Pour plus d'informations, consultez l'API Da Vinci PDex Provider Access. Les profils suivants sont pris en charge :

- US Core 3.1.1, 6.1.0 et 7.0.0
- PDex Autorisation préalable (non prise en charge pour l'accès des membres)
- CARIN BB 2.x Profils de base : établissement hospitalier, établissement ambulatoire, professionnel, oral, pharmacie NonClinician

Pour les PDex exportations, les ressources cliniques et de gestion des sinistres sont automatiquement découvertes pour chaque patient du groupe. Il n'est pas nécessaire de faire explicitement référence à ces ressources dans la ressource Groupe. L'opération recherche toutes les ressources du compartiment patient (telles que `ObservationCondition`,`Coverage`,`RelatedPerson`,`MedicationRequest`, et`ExplanationOfBenefit`) appartenant aux patients attribués. Seuls `PatientGroup`, et les types d' non-patient-compartmentATR (`Practitioner`,`PractitionerRole`,`Organization`,`Location`) nécessitent des références explicites dans le Groupe.

Accès au fournisseur (`hl7.fhir.us.davinci-pdex`)

Permet aux fournisseurs du réseau de récupérer les données des patients pour les patients auxquels ils ont été attribués.

Payer-to-Payer (`hl7.fhir.us.davinci-pdex.p2p`)

Permet l'échange de données entre les payeurs lorsqu'un patient change d'assurance.

Accès aux membres (`hl7.fhir.us.davinci-pdex.member`)

Permet aux membres d'accéder à leurs propres données de santé. Ce type d'exportation peut inclure des données financières dans les ressources relatives aux sinistres.

Support du profil et logique d'inclusion

Pour les PDex exportations, l'`$davinci-data-export`opération utilise des déclarations de profil dans l'`meta.profile`élément pour déterminer les ressources à inclure dans l'exportation.

ExplanationOfBenefit Gestion des ressources

`ExplanationOfBenefit`Les ressources (EOB) sont incluses ou exclues des PDex exportations en fonction de leurs `meta.profile` déclarations :

- ExplanationOfBenefit les ressources dotées d'un profil CARIN BB 1.x sont exclues de l'exportation.
- ExplanationOfBenefit les ressources non `meta.profile` définies sont exclues de l'exportation.
- ExplanationOfBenefit les ressources avec un profil CARIN BB 2.x Basis sont toujours incluses.
- ExplanationOfBenefit les ressources dotées d'un profil CARIN BB 2.x contenant des données financières sont exclues par défaut. Lorsqu'il `_includeEOB2xWoFinancial=true` est défini, ils sont inclus dans les données financières supprimées et la ressource est transformée selon le profil de base correspondant.
- ExplanationOfBenefit les ressources avec un profil d'autorisation PDex préalable sont toujours incluses.

Transformation des données financières

Lorsque vous le définissez`_includeEOB2xWoFinancial=true`, l'opération transforme les ExplanationOfBenefit ressources CARIN BB 2.x en leurs profils de base correspondants en supprimant les données financières. Par exemple, une `C4BB ExplanationOfBenefit Oral`

ressource est transformée en`C4BB ExplanationOfBenefit Oral Basis`, ce qui supprime les données financières de l'enregistrement conformément à la spécification FHIR.

Les éléments de données financières suivants sont supprimés lors de la transformation :

- Tous les éléments sont tranchés `total`
- Tous les `adjudication` éléments avec `amounttype` tranche
- Tous les `item.adjudication` éléments avec des informations sur le montant

L'opération met également à jour les métadonnées du profil lors de la transformation :

- `meta.profile`est mis à jour vers l'URL canonique du profil Basis
- La version est mise à jour vers la version CARIN BB 2.x Basis
- Les ressources existantes dans le magasin de données ne sont pas modifiées
- Les ressources exportées ne sont pas renvoyées dans le magasin de données

Règles de détection des profils

L'opération utilise les règles suivantes pour détecter et valider les profils :

- La détection des versions est basée sur la norme `meta.profile` canonique URLs
- Une ressource est incluse si l'un de ses profils déclarés correspond aux critères d'exportation
- La validation du profil a lieu pendant le traitement de l'exportation

Filtrage temporel quinquennal pour les PDex exportations

Pour tous les types PDex d'exportation, HealthLake applique un filtre temporel de 5 ans basé sur la date de dernière mise à jour de la ressource. Le filtre temporel s'applique à toutes les ressources, à l'exception des types de ressources d'attribution de base suivants, qui sont toujours exportés quel que soit leur âge :

- `Patient`
- `Coverage`
- `Organization`
- `Practitioner`
- `PractitionerRole`

- `RelatedPerson`
- `Location`
- `Group`

Ces ressources administratives et démographiques sont exemptées car elles fournissent un contexte essentiel pour les données exportées. Les exportations ATR ne sont soumises à aucun filtrage temporel.

Exemple de demandes

Les exemples suivants montrent comment démarrer des tâches d'exportation pour différents types d'exportation.

Exportation ATR

```
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"
    }
  }
}
```

Exportation de l'accès au fournisseur avec suppression des données ExplanationOfBenefit financières

```
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 exportation

```
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
```

Exportation de l'accès aux membres pour un patient spécifique

```
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
```

Exemple de réponse

```
{
  "datastoreId": "eaee622d8406b41eb86c0f4741201ff9",
  "jobStatus": "SUBMITTED",
  "jobId": "48d7b91dae4a64d00d54b70862f33f61"
}
```

Relations avec les ressources

L'opération exporte les ressources en fonction de leurs relations au sein de la liste d'attribution des membres :

```
Group (Attribution List)
### Patient (Members)
### Coverage # RelatedPerson (Subscribers)
### Practitioner (Attributed Providers)
### PractitionerRole # Location
### Organization (Attributed Providers)
```

> **Note**
>
> Le diagramme des relations entre les ressources précédent s'applique aux exportations ATR. Pour les PDex exportations, les ressources cliniques et relatives aux réclamations sont automatiquement découvertes grâce à la recherche de patients et ne nécessitent pas de références explicites dans la ressource du groupe.

Sources de ressources

Ressource	Emplacement de la source	Description
`Patient`	`Group.member.entity`	Les patients membres de la liste d'attribution
`Coverage`	`Group.member.extension:coverageReference`	Couverture ayant abouti à l'adhésion du patient
`Organization`	`Group.member.extension:attributedProvider`	Organisations auxquelles les patients sont attribués
`Practitioner`	`Group.member.extension:attributedProvider`	Praticiens individuels auxquels les patients sont attribués
`PractitionerRole`	`Group.member.extension:attributedProvider`	Rôles de praticiens auxquels les patients sont attribués
`RelatedPerson`	`Coverage.subscriber`	Abonnés de la couverture
`Location`	`PractitionerRole.location`	Lieux associés aux rôles des praticiens
`Group`	Point d'entrée	La liste d'attribution elle-même

Gestion des emplois

Vérifier le statut du job

```
GET [base]/export/[job-id]
```

Annuler une tâche

```
DELETE [base]/export/[job-id]
```

Cycle de vie d'une tâche

- `SUBMITTED`- Le job a été reçu et mis en file d'attente
- `IN_PROGRESS`- Job en cours de traitement actif
- `COMPLETED`- Job terminé avec succès, fichiers disponibles au téléchargement
- `FAILED`- Job a rencontré une erreur

Format de sortie

- Format de fichier : NDJSON (JSON délimité par une nouvelle ligne)
- Organisation des fichiers : fichiers distincts pour chaque type de ressource
- Extension de fichier : .ndjson
- Emplacement : compartiment et chemin S3 spécifiés

Gestion des erreurs

L'opération renvoie une mauvaise demande HTTP 400 avec un OperationOutcome pour les conditions suivantes :

Erreurs d'autorisation

Le rôle IAM spécifié dans `DataAccessRoleArn` ne dispose pas des autorisations suffisantes pour effectuer l'opération d'exportation. Pour obtenir la liste complète des autorisations S3 et KMS requises, voir Configuration des autorisations pour les tâches d'exportation.

Erreurs de validation des paramètres

- Le `patient` paramètre n'est pas formaté comme `Patient/id,Patient/id,...`
- Une ou plusieurs références de patients ne sont pas valides ou n'appartiennent pas au groupe spécifié

- La valeur du `exportType` paramètre n'est pas un type d'exportation pris en charge
- Le `_type` paramètre contient les types de ressources qui ne sont pas pris en charge pour le type d'exportation spécifié
- Le `_type` paramètre ne contient pas les types de ressources requis (`Group`,`Patient`,`Coverage`) pour le type `hl7.fhir.us.davinci-atr` d'exportation
- La valeur du `_includeEOB2xWoFinancial` paramètre n'est pas un booléen valide

Erreurs de validation des ressources

- La ressource de groupe spécifiée n'existe pas dans le magasin de données
- La ressource de groupe spécifiée n'a aucun membre
- Un ou plusieurs membres du groupe font référence à des ressources pour patients qui n'existent pas dans le magasin de données

Sécurité et autorisation

- Les mécanismes d'autorisation FHIR standard s'appliquent
- Le rôle d'accès aux données doit disposer des autorisations IAM requises pour les opérations S3 et KMS. Pour obtenir la liste complète des autorisations requises, voir Configuration des autorisations pour les tâches d'exportation.

Bonnes pratiques

- Sélection du type de ressource : demandez uniquement les types de ressources dont vous avez besoin pour minimiser la taille des exportations et le temps de traitement
- Filtrage basé sur le temps : utilisez le `_since` paramètre pour les exportations incrémentielles
- Filtrage des patients : utilisez le `patient` paramètre lorsque vous n'avez besoin de données que pour des membres spécifiques
- Surveillance des tâches : vérifiez régulièrement l'état des tâches pour les exportations importantes
- Gestion des erreurs : implémentez une logique de nouvelle tentative appropriée pour les tâches ayant échoué
- Connaissance du filtre temporel : pour les PDex exportations, considérez le filtre temporel de 5 ans lorsque vous sélectionnez les types de ressources
- Suppression des données financières : à utiliser `_includeEOB2xWoFinancial=true` lorsque vous avez besoin de données relatives aux réclamations sans informations financières

- Gestion des profils : assurez-vous que les ressources disposent de déclarations de profil appropriées, validez par rapport aux profils cibles avant l'ingestion et utilisez le versionnement des profils pour contrôler le comportement d'exportation

Limitations

- Un maximum de 500 patients peut être spécifié dans le `patient` paramètre
- L'exportation est limitée aux opérations au niveau du groupe uniquement
- Supporte uniquement l'ensemble prédéfini de types de ressources pour chaque type d'exportation
- La sortie est toujours au format NDJSON
- PDex les exportations sont limitées à 5 ans de données cliniques et de réclamations
- La transformation des données financières ne s'applique qu'aux profils CARIN BB 2.x ExplanationOfBenefit

Ressources supplémentaires

- Liste d'attribution des membres de Da Vinci IG
- Da Vinci Payer Data Exchange IG
- CARIN Consumer Directed Payer Data Exchange IG
- Guide de mise en œuvre de base aux États-Unis
- Spécification d'accès aux données en masse FHIR

Génération de documents cliniques avec **`$document`**

AWS HealthLake prend désormais en charge le `$document` fonctionnement des ressources de composition, ce qui vous permet de générer un document clinique complet en regroupant la composition et toutes ses ressources référencées dans un seul package cohérent. Cette opération est essentielle pour les applications de santé qui doivent :

- Créez des documents cliniques standardisés
- Échangez les dossiers complets des patients
- Stockez une documentation clinique complète
- Générez des rapports qui incluent tous les contextes pertinents

Usage

L'`$document`opération peut être invoquée sur les ressources de composition à l'aide des méthodes GET et POST :

Opérations prises en charge

```
GET/POST [base]/Composition/[id]/$document
```

Paramètres pris en charge

HealthLake prend en charge le `$document` paramètre FHIR suivant :

Paramètre	Type	Obligatoire	Par défaut	Description
`persist`	booléen	Non	false	Booléen indiquant si le serveur doit stocker le paquet de documents généré

Exemples

Demande GET

```
GET [base]/Composition/180f219f-97a8-486d-99d9-ed631fe4fc57/$document?persist=true
```

Requête POST avec paramètres

```
POST [base]/Composition/180f219f-97a8-486d-99d9-ed631fe4fc57/$document
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "persist",
      "valueBoolean": true
    }
```

```
  ]
}
```

Exemple de réponse

L'opération renvoie une ressource Bundle de type « document » contenant la composition et toutes les ressources référencées :

```
{
  "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"
            }
          ]
        }
      }
    }
  ]
}
```

Comportement

L'$documentopération :

1. Utilise la ressource de composition spécifiée comme base du document
2. Identifie et récupère toutes les ressources directement référencées par la composition
3. Regroupe la composition et toutes les ressources référencées dans un ensemble de type « document »
4. Stocke le paquet de documents généré dans la banque de données lorsque le paramètre persist est défini sur true
5. Identifie et récupère les ressources référencées indirectement par la composition pour une génération complète de documents

L'`$document`opération prend actuellement en charge la récupération de références de ressources au format suivant :

1. ```
   GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Resource/id
   ```
2. Ressource/ID

Les références de ressources non prises en charge dans la ressource Composition seront éliminées du document généré.

Gestion des erreurs

L'opération gère les conditions d'erreur suivantes :

- 400 Mauvaise demande : `$document` opération non valide (demande non conforme) ou si le document obtenu échoue à la validation FHIR en raison du filtrage des références lorsque persist est défini sur true
- 404 Introuvable : ressource de composition introuvable

Pour plus d'informations sur les spécifications de `$document` fonctionnement, consultez la documentation de composition `$document` du FHIR R4.

Suppression permanente de ressources avec **`$erase`**

AWS HealthLake prend en charge l'`$erase`opération, permettant la suppression permanente d'une ressource spécifique et de ses versions historiques. Cette opération est particulièrement utile lorsque vous devez :
   ```

- Supprimer définitivement des ressources individuelles
- Supprimer des historiques de versions spécifiques
- Gérez le cycle de vie des ressources individuelles
- Respectez les exigences spécifiques en matière de suppression des données

Usage

L'`$erase`opération peut être invoquée à deux niveaux :

Niveau de l'instance de ressource

```
POST [base]/[ResourceType]/[ID]/$erase?deleteAuditEvent=true
```

Niveau spécifique à la version

```
POST [base]/[ResourceType]/[ID]/_history/[VersionID]/$erase
```

Parameters

Paramètre	Type	Obligatoire	Par défaut	Description
`deleteAuditEvent`	booléen	Non	false	Lorsque c'est vrai, supprime les événements d'audit associés

Exemples

Exemple de requête

```
POST [base]/Patient/example-patient/$erase
```

Exemple de réponse

```
{
      "jobId": "5df47e2f51ff3c731847678cb8cad48e",
```

```
      "jobStatus": "SUBMITTED"
    }
```

Statut de la tâche

Pour vérifier le statut d'une tâche d'effacement :

```
GET [base]/$erase/[jobId]
```

L'opération renvoie les informations relatives à l'état de la tâche :

```
{
      "datastoreId": "36622996b1fcecb7e12ee2ee085308d3",
      "jobId": "5df47e2f51ff3c731847678cb8cad48e",
      "status": "COMPLETED",
      "submittedTime": "2025-10-30T16:39:24.160Z"
    }
```

Comportement

L'`$erase`opération :

1. Traite de manière asynchrone pour garantir l'intégrité des données
2. Maintient les transactions ACID
3. Assure le suivi de l'état des tâches
4. Supprime définitivement la ressource spécifiée et ses versions
5. Inclut un enregistrement d'audit complet des activités de suppression
6. Supporte la suppression sélective des événements d'audit

Journalisation des audits

L'`$erase`opération est enregistrée sous forme DeleteResource d'ID utilisateur, d'horodatage et de détails sur les ressources.

Limitations

- `$erased`la ressource n'apparaîtra pas dans les résultats de recherche ou `_history` les requêtes.
- Les ressources en cours d'effacement peuvent être temporairement inaccessibles pendant le traitement

- La mesure du stockage est ajustée immédiatement lorsque les ressources sont définitivement supprimées

Obtenir les données des patients avec **`Patient/$everything`**

L'`Patient/$everything`opération est utilisée pour interroger une `Patient` ressource FHIR, ainsi que toute autre ressource associée à celle-ci`Patient`. L'opération peut être utilisée pour permettre à un patient d'accéder à l'intégralité de son dossier ou pour qu'un fournisseur effectue un téléchargement massif de données relatives à un patient. HealthLakesoutiens `Patient/$everything` pour un patient spécifique`id`.

`Patient/$everything`est une opération d'API REST FHIR qui peut être invoquée comme indiqué dans les exemples ci-dessous.

GET request

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/$everything
```

> **Note**
>
> Les ressources en réponse sont triées par type de ressource et par ressource`id`.
> La réponse est toujours renseignée avec`Bundle.total`.

Paramètres dans l'**`Patient/$everything`**

HealthLake prend en charge les paramètres de requête suivants

Paramètre	Détails
démarrer	Obtenez toutes les `Patient` données après une date de début spécifiée.
end	Obtenez toutes les `Patient` données avant une date de fin spécifiée.
since	Mettez toutes les `Patient` données à jour après une date spécifiée.
_type	Obtenez `Patient` des données pour des types de ressources spécifiques.

Paramètre	Détails
_compter	Obtenez `Patient` des données et spécifiez le format de page.

Example- Obtenez toutes les données du patient après une date de début spécifiée

`Patient/$everything`peut utiliser le `start` filtre pour interroger uniquement les données après une date précise.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?start=2024-03-15T00:00:00.000Z
```

Example- Obtenez toutes les Patient données avant une date de fin spécifiée

Patient $everything peut utiliser le end filtre pour n'interroger que des données antérieures à une date précise.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?end=2024-03-15T00:00:00.000Z
```

Example- Mettre à jour toutes les Patient données après une date spécifiée

`Patient/$everything`peut utiliser le `since` filtre pour n'interroger que les données mises à jour après une date précise.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?since=2024-03-15T00:00:00.000Z
```

Example- Obtenir Patient des données pour des types de ressources spécifiques

Le patient $everything peut utiliser le `_type` filtre pour spécifier des types de ressources spécifiques à inclure dans la réponse. Plusieurs types de ressources peuvent être spécifiés dans une liste séparée par des virgules.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?_type=Observation,Condition
```

Example- Obtenez Patient des données et spécifiez le format de page

Le patient $everything peut utiliser le `_count` pour définir le format de page.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?_count=15
```

Patient/$everythingstart et **end** attributs

HealthLake prend en charge les attributs de ressource suivants pour les paramètres de end requête `Patient/ $everything` `start` et.

Ressource	Élément de ressource
Compte	Account.ServicePeriod.Start
AdverseEvent	AdverseEvent.date
AllergyIntolerance	AllergyIntolerance. Date d'enregistrement
Rendez-vous	Rendez-vous.Start
AppointmentResponse	AppointmentResponse.démarrer
AuditEvent	AuditEvent.period.start
Base	Basic. Créé
BodyStructure	AUCUNE DATE
CarePlan	CarePlan.period.start
CareTeam	CareTeam.period.start
ChargeItem	ChargeItem. occurrenceDateTime, ChargeItem .occurrencePeriod.start, .occurrenceTiming.Event ChargeItem
Demander	Réclamer. Période facturable. Début

Ressource	Élément de ressource
ClaimResponse	ClaimResponse.créé
ClinicalImpression	ClinicalImpression.date
Communication	Communication. Envoyée
CommunicationRequest	CommunicationRequest. occurrenceDateTime, CommunicationRequest .occurrencePeriod.start
Montage	Date de composition
Condition	État. Date d'enregistrement
Consentement	Consent.Date/Heure
Couverture	Couverture.Period.Start
CoverageEligibilityRequest	CoverageEligibilityRequest.créé
CoverageEligibilityResponse	CoverageEligibilityResponse.créé
DetectedIssue	DetectedIssue.identifié
DeviceRequest	DeviceRequest. Rédigé le
DeviceUseStatement	DeviceUseStatement. Enregistré le

Ressource	Élément de ressource
DiagnosticReport	DiagnosticReport.efficace
DocumentManifest	DocumentManifest.créé
DocumentReference	DocumentReference.context .period .start
Rencontre	Encounter.Period.Start
EnrollmentRequest	EnrollmentRequest.créé
EpisodeOfCare	EpisodeOfCare.period.start
ExplanationOfBenefit	ExplanationOfBenefit. Période facturable. Début
FamilyMemberHistory	AUCUNE DATE
Indicateur	Flag.Period.Start
Objectif	Objectif. Date d'état
Groupe	AUCUNE DATE
ImagingStudy	ImagingStudy.démarré
Immunisation	Vaccination enregistrée

Ressource	Élément de ressource
ImmunizationEvaluation	ImmunizationEvaluation.date
ImmunizationRecommendation	ImmunizationRecommendation.date
Invoice	Date de facturation
List	Date de la liste
MeasureReport	MeasureReport.period.start
Multimédia	Publié dans les médias
MedicationAdministration	MedicationAdministration.efficace
MedicationDispense	MedicationDispense. Une fois préparé
MedicationRequest	MedicationRequest. Rédigé le
MedicationStatement	MedicationStatement. Date affirmée
Molecular Sequence	AUCUNE DATE
Nutrition Order	NutritionOrder.Date/Heure
Observation	Observation. Efficace

Ressource	Élément de ressource
Patient	AUCUNE DATE
Personne	AUCUNE DATE
Procédure	Procédure. Exécutée
Provenance	Provenance. Période survenue. Début, provenance. occurredDateTime
QuestionnaireResponse	QuestionnaireResponse.écrit
RelatedPerson	AUCUNE DATE
RequestGroup	RequestGroup. Rédigé le
ResearchSubject	ResearchSubject.période
RiskAssessment	RiskAssessment. occurrenceDateTime, RiskAssessment .occurrencePeriod.start
Planning	Calendrier. Horizon de planification
ServiceRequest	ServiceRequest. Rédigé le
Spécimen	Spécimen. Heure de réception
SupplyDelivery	SupplyDelivery. occurrenceDateTime, SupplyDelivery .occurrencePeriod. start, .occurrenceTiming.Event SupplyDelivery
SupplyRequest	SupplyRequest. Rédigé le

Ressource	Élément de ressource
VisionPrescription	VisionPrescription. Date de rédaction

Récupération de ValueSet codes avec **$expand**

AWS HealthLake prend désormais en charge les `$expand` opérations ValueSets que vous avez ingérées en tant que client, ce qui vous permet de récupérer la liste complète des codes contenus dans ces ValueSet ressources. Cette opération est particulièrement utile lorsque vous devez :

- Récupérez tous les codes possibles à des fins de validation
- Afficher les options disponibles dans les interfaces utilisateur
- Effectuez des recherches de code complètes dans un contexte terminologique spécifique

Usage

L'`$expand`opération peut être invoquée sur les ValueSet ressources à l'aide des méthodes GET et POST :

Opérations prises en charge

```
GET/POST [base]/ValueSet/[id]/$expand
GET [base]/ValueSet/$expand?url=http://example.com
POST [base]/ValueSet/$expand
```

Paramètres pris en charge

HealthLake prend en charge un sous-ensemble de paramètres FHIR `$expand` R4 :

Paramètre	Type	Obligatoire	Description
`url`	uri	Non	URL canonique du ValueSet à développer
`id`	id	Non	ValueSet identifiant de ressource à étendre (pour les opérations GET ou POST)

Paramètre	Type	Obligatoire	Description
`filter`	chaîne	Non	Filtrer le résultat de l'extension du code
`count`	entier	Non	Nombre de codes à retourner
`offset`	entier	Non	Nombre de codes correspondants à ignorer avant de retourner. S'applique après le filtrage et uniquement aux codes correspondants, et non à l'intégralité du contenu non filtré de l'original ValueSet

Exemples

Demande GET par identifiant

```
GET [base]/ValueSet/example-valueset/$expand
```

Requête GET par URL avec filtre

```
GET [base]/ValueSet/$expand?url=http://example.com/ValueSet/my-
valueset&filter=male&count=5
```

Requête POST avec paramètres (par ID)

```
POST [base]/ValueSet/example-valueset/$expand
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "count",
      "valueInteger": 10
    },
    {
      "name": "filter",
```

```
      "valueString": "admin"
    }
  ]
}
```

Requête POST avec paramètres (par URL)

```
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
    }
  ]
}
```

Exemple de réponse

L'opération renvoie une ValueSet ressource avec un `expansion` élément contenant les codes développés :

```
{
  "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"
      }
    ]
  }
}
```

La réponse inclut :

- expansion.total : nombre total de codes dans l'extension ValueSet
- expansion.contains : tableau de codes étendus avec leur système, leur code et leurs valeurs d'affichage
- expansion.parameter : paramètres utilisés dans la demande d'extension

Pour plus d'informations sur les spécifications de `$expand` fonctionnement, consultez la documentation du FHIR R4 ValueSet `$expand`.

Exporter HealthLake des données avec FHIR **`$export`**

Vous pouvez exporter des données en masse depuis votre magasin de HealthLake données à l'aide de l'opération FHIR $export. HealthLake prend en charge l'`$export`utilisation `POST` et les `GET` demandes de FHIR. Pour effectuer une demande d'exportation avec`POST`, vous devez disposer

d'un utilisateur, d'un groupe ou d'un rôle IAM doté des autorisations requises, le spécifier dans le `$export` cadre de la demande et inclure les paramètres souhaités dans le corps de la demande.

> **Note**
>
> Toutes les demandes HealthLake d'exportation effectuées à l'aide de FHIR `$export` sont renvoyées au `ndjson` format et exportées vers un compartiment Amazon S3, où chaque objet Amazon S3 ne contient qu'un seul type de ressource FHIR.
> Vous pouvez mettre en file d'attente les demandes d'exportation conformément aux quotas de service du AWS compte. Pour de plus amples informations, veuillez consulter Quotas de service.

HealthLake prend en charge les trois types suivants de demandes de terminaux d'exportation en masse.

HealthLake **`$export`**types en vrac

Type d'exportation	Description	Syntaxe
Système	Exportez toutes les données depuis le serveur HealthLake FHIR.	`POST https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/$export`

Tous les patients	Exportez toutes les données relatives à tous les patients, y compris les types de ressources associés au type de ressource Patient.	`POST https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/$export`
		
		`GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/$export`

Groupe de patients	Exportez toutes les données relatives à un groupe de patients spécifié par un identifiant de groupe.	`POST https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Group/id/$export`

Type d'exportation	Description	Syntaxe
		`GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Group/id/$export`

Avant de commencer

Répondez aux exigences suivantes pour effectuer une demande d'exportation à l'aide de l'API REST FHIR pour HealthLake.

- Vous devez avoir configuré un utilisateur, un groupe ou un rôle disposant des autorisations nécessaires pour effectuer la demande d'exportation. Pour en savoir plus, veuillez consulter la section Autoriser une demande `$export`.
- Vous devez avoir créé un rôle de service qui accorde l' HealthLake accès au compartiment Amazon S3 vers lequel vous souhaitez que vos données soient exportées. Le rôle de service doit également être spécifié HealthLake comme principal de service. Pour plus d'informations sur la configuration des autorisations, consultezConfiguration des autorisations pour les tâches d'exportation.

Autoriser une demande **`$export`**

Pour effectuer une demande d'exportation réussie à l'aide de l'API REST FHIR, autorisez votre utilisateur, votre groupe ou votre rôle à l'aide d'IAM ou OAuth2 .0. Vous devez également avoir un rôle de service.

Autoriser une demande à l'aide d'IAM

Lorsque vous faites une `$export` demande, les actions IAM de l'utilisateur, du groupe ou du rôle doivent être incluses dans la politique. Pour de plus amples informations, veuillez consulter Configuration des autorisations pour les tâches d'exportation.

Autoriser une demande à l'aide de SMART sur FHIR (2.0) OAuth

Lorsque vous faites une `$export` demande sur un magasin de HealthLake données compatible SMART sur FHIR, les étendues appropriées doivent vous être attribuées. Pour de plus amples informations, veuillez consulter SMART sur les champs de ressources FHIR pour HealthLake.

> **Note**
>
> Le FHIR `$export` avec `GET` demandes nécessite la même méthode d'authentification ou le même jeton porteur (dans le cas de SMART sur FHIR) pour demander l'exportation et la récupération de fichiers. Les fichiers exportés à l'aide de FHIR `$export` avec `GET` peuvent être téléchargés pendant 48 heures.

Faire une **`$export`** demande

Cette section décrit les étapes obligatoires que vous devez suivre lorsque vous effectuez une demande d'exportation à l'aide de l'API REST FHIR.

Pour éviter des frais accidentels sur votre AWS compte, nous vous recommandons de tester vos demandes en effectuant une `POST` demande sans fournir de `$export` syntaxe.

Pour faire la demande, vous devez effectuer les opérations suivantes :

1. Spécifiez `$export` dans l'URL de `POST` demande un point de terminaison pris en charge.
2. Spécifiez les paramètres d'en-tête requis.
3. Spécifiez un corps de demande qui définit les paramètres requis.

Étape 1 : Spécifiez **`$export`** dans l'URL de **`POST`** demande un point de terminaison pris en charge.

HealthLake prend en charge trois types de demandes d'exportation groupées pour les terminaux. Pour effectuer une demande d'exportation groupée, vous devez effectuer une demande `POST` basée sur l'un des trois points de terminaison pris en charge. Les exemples suivants montrent où spécifier `$export` dans l'URL de la demande.

- `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`

Vous pouvez utiliser les paramètres de recherche pris en charge suivants dans la chaîne de `POST` requête.

Paramètres de recherche pris en charge

HealthLake prend en charge les modificateurs de recherche suivants dans les demandes d'exportation groupées.

Les exemples suivants incluent des caractères spéciaux qui doivent être codés avant de soumettre votre demande.

Nom	Obligatoire ?	Description	Exemple
`_outputFormat`	Non	Format des fichiers de données en masse demandés à générer. Les valeurs acceptées sont `applicati on/fhir+n djson applicati on/ndjson` ,`ndjson`.	
`_type`	Non	Chaîne de types de ressources FHIR séparés par des virgules que vous souhaitez inclure dans votre tâche d'exporta tion. Nous vous recommandons de l'inclure, `_type` car cela peut avoir une incidence financièr e lorsque toutes les ressources sont exportées.	`&_type=Me dicationS tatement, Observation`
`_since`	Non	Types de ressource s modifiés le jour ou après l'horodatage. Si	`&_since=2 024-05-09 T00%3A00%3A00Z`

Nom	Obligatoire ?	Description	Exemple
		un type de ressource n'a pas d'heure de dernière mise à jour, il sera inclus dans votre réponse.	
`_until`	Non	Types de ressources modifiés le jour ou avant l'horodatage. Utilisé en combinaison avec `_since` pour définir une plage de temps spécifique pour l'exportation. Si un type de ressource n'a pas d'heure de dernière mise à jour, il sera exclu de votre réponse.	`&_until=2024-12-31T23%3A59%3A59Z`

Nom	Obligatoire ?	Description	Exemple
`_security`	Non	Filtrez les ressources exportées par valeurs de `meta.security` codage. Utilisez le `system|code` format. Lorsque plusieurs valeurs sont fournies, les ressources doivent correspondre à toutes (sémantiq ue ET). Utilisez `system|` (tube final, pas de code) pour faire correspondre n'importe quel code d'un système donné.	`&_securit y=https:/ /myorg.co m/tenant% 7Cclinic-A`
`_tag`	Non	Filtrez les ressource s exportées par valeurs de `meta.tag` codage. Utilise le même `system| code` format et la même sémantiqu e AND que. `_security` Lorsque `_security` les deux filtres `_tag` sont spécifiés, les ressources doivent correspondre aux deux filtres.	`&_tag=https:// myorg.com/dept %7Ccardiology`

Étape 2 : Spécifier les paramètres d'en-tête requis

Pour effectuer une demande d'exportation à l'aide de l'API REST FHIR, vous devez spécifier les paramètres d'en-tête suivants.

- Type de contenu : `application/fhir+json`
- Préférez : `respond-async`

Ensuite, vous devez spécifier les éléments requis dans le corps de la demande.

Étape 3 : Spécifiez un corps de demande qui définit les paramètres requis.

La demande d'exportation nécessite également un corps au `JSON` format. Le corps peut inclure les paramètres suivants.

Clé	Obligatoire ?	Description	Value
DataAccessRoleArn	Oui	L'ARN d'un rôle HealthLake de service. Le rôle de service utilisé doit être spécifié HealthLake comme principal de service.	`arn:aws:iam:: 444455556666 :role/your-healthlake-service-role`
JobName	Non	Nom de la demande d'exportation.	**`your-export-job-name`**
S3Uri	Oui	C'est une partie d'une OutputDataConfig clé. L'URI S3 du compartiment de destination dans lequel vos données exportées seront téléchargées.	`s3://amzn-s3-demo-bucket/ EXPORT-JOB /`
KmsKeyId	Oui	C'est une partie d'une OutputDat	`arn:aws:kms: region-of-`

Clé	Obligatoire ?	Description	Value
		aConfig clé. L'ARN de la AWS KMS clé utilisée pour sécuriser le compartiment Amazon S3.	**bucket:123456789012** :key/**1234abcd-12ab-34cd-56ef-1234567890ab**

Example Corps d'une demande d'exportation effectuée à l'aide de l'API REST FHIR

Pour effectuer une demande d'exportation à l'aide de l'API REST FHIR, vous devez spécifier un corps, comme indiqué ci-dessous.

```
{
  "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"
    }
  }
}
```

Lorsque votre demande sera acceptée, vous recevrez la réponse suivante.

En-tête de réponse

```
content-location: https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
export/your-export-request-job-id
```

Organisme de réponse

```
{
  "datastoreId": "your-data-store-id",
  "jobStatus": "SUBMITTED",
  "jobId": "your-export-request-job-id"
}
```

Gestion de votre demande d'exportation

Après avoir effectué une demande d'exportation réussie, vous pouvez la gérer en `$export` décrivant le statut d'une demande d'exportation en cours et `$export` en annulant une demande d'exportation en cours.

Lorsque vous annulez une demande d'exportation à l'aide de l'API REST, vous n'êtes facturée que pour la partie des données exportées jusqu'au moment où vous avez soumis la demande d'annulation.

Les rubriques suivantes décrivent comment vous pouvez obtenir le statut d'une demande d'exportation en cours ou l'annuler.

Annulation d'une demande d'exportation

Pour annuler une demande d'exportation, faites une `DELETE` demande et indiquez l'ID de tâche dans l'URL de la demande.

```
DELETE https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/export/your-export-request-job-id
```

Lorsque votre demande est acceptée, vous recevez ce qui suit.

```
{
  "exportJobProperties": {
    "jobId": "your-original-export-request-job-id",
    "jobStatus": "CANCEL_SUBMITTED",
    "datastoreId": "your-data-store-id"
  }
}
```

Lorsque votre demande n'aboutit pas, vous recevez ce qui suit.

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "not-supported",
      "diagnostics": "Interaction not supported."
    }
  ]
```

```
}
```

Décrire une demande d'exportation

Pour connaître le statut d'une demande d'exportation, faites une `GET` demande en utilisant `export` et votre**`export-request-job-id`**.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/export/your-export-request-id
```

La réponse JSON contiendra un `ExportJobProperties` objet. Il peut contenir les paires clé:valeur suivantes.

Nom	Obligatoire ?	Description	Value
DataAccessRoleArn	Non	L'ARN d'un rôle HealthLake de service. Le rôle de service utilisé doit être spécifié HealthLake comme principal de service.	`arn:aws:iam:: 444455556666 :role/your-healthlake-service-role`
SubmitTime	Non	Date à laquelle une tâche d'exportation a été soumise.	`Apr 21, 2023 5:58:02`
EndTime	Non	Heure à laquelle une tâche d'exportation a été terminée.	`Apr 21, 2023 6:00:08 PM`
JobName	Non	Nom de la demande d'exportation.	**`your-export-job-name`**
JobStatus	Non		Les valeurs valides sont : `SUBMITTED | IN_PROGRESS`

Nom	Obligatoire ?	Description	Value
			`| COMPLETED _WITH_ERRORS | COMPLETED | FAILED`
S3Uri	Oui	Partie d'un OutputDataConfigobjet. L'URI Amazon S3 du compartiment de destination dans lequel vos données exportées seront téléchargées.	`s3://amzn-s3-demo-bucket/` **`EXPORT-JOB /`**
KmsKeyId	Oui	Partie d'un OutputDataConfigobjet. L'ARN de la AWS KMS clé utilisée pour sécuriser le compartiment Amazon S3.	`arn:aws:kms:` **`region-of-bucket:123456789012`** `:key/`**`1234abcd-12ab-34cd-56ef-1234567890ab`**

Example: corps d'une demande d'exportation de description effectuée à l'aide de l'API REST FHIR

En cas de succès, vous obtiendrez la réponse JSON suivante.

```
{
  "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",
  }
}
```

$inquire Opération FHIR pour HealthLake

L'`$inquire`opération vous permet de vérifier le statut d'une demande d'autorisation préalable soumise précédemment. Cette opération met en œuvre le guide de mise en œuvre du Da Vinci Prior Authorization Support (PAS), fournissant un flux de travail standardisé basé sur le FHIR pour récupérer la décision d'autorisation actuelle.

Comment ça marche

- Soumettre une demande : vous envoyez un bundle FHIR contenant la réclamation que vous souhaitez vérifier et les informations justificatives
- Rechercher : HealthLake recherche le correspondant ClaimResponse dans votre magasin de données
- Récupérer : le statut d'autorisation le plus récent est récupéré
- Répondre : vous recevez une réponse immédiate avec le statut d'autorisation actuel (en file d'attente, approuvé, refusé, etc.)

> **Note**
>
> `$inquire`est une opération en lecture seule qui permet de récupérer le statut d'autorisation existant. Il ne modifie ni ne met à jour aucune ressource de votre banque de données.

Point de terminaison d’API

```
POST /datastore/{datastoreId}/r4/Claim/$inquire
Content-Type: application/fhir+json
```

Structure de la demande

Exigences relatives au bundle

Votre demande doit être une ressource FHIR Bundle contenant :

- Bundle.type : Doit être `"collection"`
- Bundle.entry : Doit contenir exactement une ressource Claim avec :
  - `use = "preauthorization"`
  - `status = "active"`
- Ressources référencées : Toutes les ressources référencées par la réclamation doivent être incluses dans le bundle

> **Requête par exemple**
>
> Les ressources de votre bundle d'entrée servent de modèle de recherche. HealthLake utilise les informations fournies pour localiser le correspondant ClaimResponse.

Ressources requises

Ressource	Cardinalité	Profil	Description
Réclamation	1	Demande de réclamation PAS	L'autorisation préalable que vous demandez
Patient	1	Patient bénéficiaire du PAS	Informations démographiques sur les patients
Organisation (assureur)	1	Organisation d'assurance PAS	Compagnie d'assurance
Organisation (fournisseur)	1	Organisation demandeuse du PAS	Prestataire de santé qui a soumis la demande

Critères de recherche importants

HealthLake recherche l' ClaimResponse utilisation de :

- Référence du patient tirée de la réclamation
- Référence de l'assureur tirée de la réclamation
- Référence du fournisseur figurant dans la réclamation
- Date de création à partir de la réclamation (en tant que filtre temporel)

> **Demandes spécifiques au patient uniquement**
>
> Toutes les demandes doivent être liées à un patient en particulier. Les requêtes à l'échelle du système sans identification du patient ne sont pas autorisées.

Exemple de demande

```
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"
      }
    }
  ]
}
```

Format de la réponse

Réponse positive (200 OK)

Vous recevrez un ensemble de réponses aux demandes de renseignements PAS contenant :

- ClaimResponseavec le statut d'autorisation actuel ; multiple ClaimResponses'il correspond aux critères de recherche
- Toutes les ressources originales de votre demande (renvoyées)
- Horodatage du moment où la réponse a été assemblée

ClaimResponse Résultats possibles

Outcome	Description
`queued`	La demande d'autorisation est toujours en attente d'examen
`complete`	La décision d'autorisation a été prise (vérifier si elle est `disposition` approuvée/refusée)
`error`	Une erreur s'est produite lors du traitement

Outcome	Description
`partial`	Autorisation partielle accordée


```
{
  "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"
      }
    }
  ]
}
```

Réponses d'erreur

400 Requête erronée

Renvoyé lorsque le format de demande n'est pas valide ou que la validation échoue.

```
{
    "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 Accès non autorisé

Renvoyé lorsque les informations d'authentification sont manquantes ou non valides.

```
{
    "resourceType": "OperationOutcome",
    "issue": [
        {
            "severity": "error",
            "code": "forbidden",
            "diagnostics": "Invalid authorization header"
        }
    ]
}
```

403 Forbidden

Renvoyé lorsque l'utilisateur authentifié n'est pas autorisé à accéder à la ressource demandée.

```
{
    "resourceType": "OperationOutcome",
    "issue": [
        {
            "severity": "error",
            "code": "exception",
            "diagnostics": "Insufficient SMART scope permissions."
        }
    ]
}
```

400 Quand aucun n'est trouvé

Renvoyé lorsqu'aucune correspondance n' ClaimResponse est trouvée pour la demande.

```
{
  "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 Type de média non pris en charge

Renvoyé lorsque l'en-tête Content-Type n'est pas application/fhir+json.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "value",
    "diagnostics": "Incorrect MIME-type. Update request Content-Type header."
  }]
}
```

429 Trop de demandes

Renvoyé lorsque les limites de débit sont dépassées.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "throttled",
    "diagnostics": "Rate limit exceeded. Please retry after some time."
  }]
}
```

Règles de validation

HealthLake effectue une validation complète de votre demande :

Validation du bundle

- Doit être conforme au profil du bundle PAS Inquiry Request
- `Bundle.type`doit être `"collection"`

- Doit contenir exactement une ressource de réclamation
- Toutes les ressources référencées doivent être incluses dans le bundle

Validation des réclamations

- Doit être conforme au profil PAS Claim Inquiry
- `Claim.use`doit être `"preauthorization"`
- `Claim.status`doit être `"active"`
- Champs obligatoires : `patient`,`insurer`,`provider`, `created`

Validation des ressources

- Toutes les ressources doivent être conformes à leurs profils d'enquête PAS respectifs
- Les ressources de soutien requises doivent être présentes (patient, organisme assureur, organisme fournisseur)
- Les références croisées doivent être valides et résolvables dans le bundle

Spécifications de performance

Métrique	Spécification de
Limite du nombre de ressources	500 ressources par bundle
Limite de taille du bundle	5 Mo maximum

Autorisations requises

Pour utiliser l'`$inquire`opération, assurez-vous que votre rôle IAM possède les éléments suivants :

- `healthlake:InquirePreAuthClaim`- Pour appeler l'opération

SMART sur les oscilloscopes FHIR

Étendue minimale requise :

- SMART version 1 : `user/ClaimResponse.read`

- SMART v2 : `user/ClaimResponse.s`

Remarques de mise en œuvre importantes

Comportement de recherche

Lorsque vous soumettez une demande, recherchez l' HealthLake utilisateur à l' ClaimResponse aide de :

- Référence du patient tirée de la demande d'entrée
- Référence de l'assureur tirée de la demande d'entrée
- Référence du fournisseur à partir de la demande d'entrée
- Date de création à partir de la demande d'entrée (en tant que filtre temporel)

Correspondances multiples : si plusieurs ClaimResponses correspondent à vos critères de recherche, HealthLake renvoie tous les résultats correspondants. Vous devez utiliser l'`ClaimResponse.created`horodatage le plus récent pour identifier le dernier statut.

Demandes mises à jour

Si vous avez soumis plusieurs mises à jour pour la même autorisation préalable (par exemple, Claim v1.1, v1.2, v1.3), l'`$inquire`opération récupérera la version ClaimResponse associée à la version la plus récente en fonction des critères de recherche fournis.

Fonctionnement en lecture seule

L'`$inquire`opération :

- Récupère le statut d'autorisation existant
- Renvoie le plus récent ClaimResponse
- Ne modifie ni ne met à jour aucune ressource
- Ne crée pas de nouvelles ressources
- Ne déclenche pas de nouveau traitement d'autorisation

Exemple de flux de travail

Flux de travail typique de demande d'autorisation préalable

```
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"
```

Opérations liées

- `Claim/$submit`- Soumettre une nouvelle demande d'autorisation préalable ou mettre à jour une demande existante
- `Patient/$everything`- Récupérez les données complètes du patient pour le contexte de l'autorisation préalable

Récupération des détails du concept avec **`$lookup`**

AWS HealthLake prend désormais en charge les `$lookup` opérations relatives aux CodeSystem ressources, ce qui vous permet de récupérer les détails d'un concept spécifique dans un système de code en fournissant des informations d'identification telles que son code. Cette opération est particulièrement utile lorsque vous devez :

- Récupérez des informations détaillées sur des codes médicaux spécifiques
- Valider la signification et les propriétés du code
- Définitions et relations des concepts d'accès
- Support à la prise de décisions cliniques grâce à des données terminologiques précises

Usage

L'`$lookup`opération peut être invoquée sur les CodeSystem ressources à l'aide des méthodes GET et POST :

Opérations prises en charge

```
GET [base]/CodeSystem/$lookup?system=http://snomed.info/
sct&code=73211009&version=20230901
POST [base]/CodeSystem/$lookup
```

Paramètres pris en charge

HealthLake prend en charge un sous-ensemble de paramètres FHIR `$lookup` R4 :

Paramètre	Type	Obligatoire	Description
`code`	code	Oui	Le code conceptuel que vous recherchez (par exemple, « 71620000 » dans SNOMED CT)
`system`	uri	Oui	L'URL canonique du système de code (par exemple, "http://snomed.info/sct «)
`version`	chaîne	Non	Version spécifique du système de code

Exemples

Demande GET

```
GET [base]/CodeSystem/$lookup?system=http://snomed.info/
sct&code=71620000&version=2023-09
```

Demande POST

```
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"
    }
  ]
}
```

Exemple de réponse

L'opération renvoie une ressource Parameters contenant les détails du concept :

```
{
    "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"
                }
            ]
        }
    ]
}
```

Paramètres de réponse

La réponse inclut les paramètres suivants lorsqu'ils sont disponibles :

Paramètre	Type	Description
`name`	chaîne	Nom du système de code
`version`	chaîne	Version du système de code
`display`	chaîne	Afficher le nom du concept
`designation`	BackboneElement	Des représentations supplémentaires pour ce concept.
`property`	BackboneElement	Propriétés supplémentaires du concept (définition, relations, etc.)

Comportement

L'`$lookup`opération :

1. Valide les paramètres requis (`codeetsystem`)
2. Recherche le concept dans le système de code spécifié stocké dans la banque de données
3. Renvoie des informations détaillées sur le concept, notamment le nom d'affichage, les désignations et les propriétés.
4. Prend en charge les recherches spécifiques à la version lorsque le paramètre est fourni `version`
5. Fonctionne uniquement sur les systèmes de code explicitement stockés dans la HealthLake banque de données

Gestion des erreurs

L'opération gère les conditions d'erreur suivantes :

- 400 Mauvaise demande : `$lookup` opération non valide (demande non conforme ou paramètres requis manquants)
- 404 Introuvable : système de code introuvable ou code introuvable dans le système de code spécifié

Mises en garde

Dans cette version, les éléments suivants ne sont pas pris en charge :

- `$lookup`opération en appelant des serveurs terminologiques externes
- `$lookup`opération sur CodeSystems gérée par HealthLake mais non explicitement stockée dans la banque de données

Pour plus d'informations sur les spécifications de `$lookup` fonctionnement, consultez la documentation du FHIR R4 CodeSystem `$lookup`.

`$member-add`opération pour HealthLake

L'`$member-add`opération FHIR ajoute un membre (patient) à une ressource de groupe, en particulier à une liste d'attribution de membres. Cette opération fait partie du guide de mise en œuvre de l'attribution des DaVinci membres et soutient le processus de rapprochement pour la gestion des attributions des membres.

Opération Endpoint

```
POST [base]/datastore/{datastoreId}/r4/Group/{groupId}/$member-add
Content-Type: application/json
```

Parameters

L'opération accepte une ressource de paramètres FHIR avec les combinaisons de paramètres suivantes :

Options de paramètres

Vous pouvez utiliser l'une des combinaisons de paramètres suivantes :

Option 1 : ID de membre + NPI du fournisseur

- `memberId` + `providerNpi`
- `memberId` + `providerNpi` + `attributionPeriod`

Option 2 : référence du patient et référence du fournisseur

- `patientReference` + `providerReference`
- `patientReference` + `providerReference` + `attributionPeriod`

Détails des paramètres

ID de membre (facultatif)

Identifiant du membre à ajouter au groupe.

Type : Identifiant

Système : système d'identification du patient

```
{
  "name": "memberId",
  "valueIdentifier": {
    "system": "http://example.org/patient-id",
    "value": "patient-new"
  }
```

```
}
```

ProviderNPI (facultatif)

L'identifiant national du fournisseur (NPI) du fournisseur attribué.

Type : Identifiant

Système : http://terminology.hl7. org/CodeSystem/NPI

```
{
  "name": "providerNpi",
  "valueIdentifier": {
    "system": "http://terminology.hl7.org/CodeSystem/NPI",
    "value": "1234567890"
  }
}
```

Référence du patient (facultatif)

Référence directe à la ressource patient à ajouter.

Type : Référence

```
{
  "name": "patientReference",
  "valueReference": {
    "reference": "Patient/patient-123"
  }
}
```

Référence du fournisseur (facultatif)

Référence directe à la ressource du fournisseur.

Type : Référence

```
{
  "name": "providerReference",
  "valueReference": {
    "reference": "Practitioner/provider-456"
  }
}
```

Période d'attribution (facultatif)

Période pendant laquelle le patient est attribué au prestataire.

Type : Période

```
{
  "name": "attributionPeriod",
  "valuePeriod": {
    "start": "2024-07-15",
    "end": "2025-07-14"
  }
}
```

Exemples de demandes

Utilisation de l'ID de membre et du NPI du fournisseur

```
{
  "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"
      }
    }
```

```
  ]
}
```

Utilisation des références des patients et des prestataires

```
{
  "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"
      }
    }
  ]
}
```

Format de la réponse

Réponse d'ajout réussie

```
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."
      }
    }
  ]
}
```

Réponses d’erreur

Syntaxe de demande non valide

État HTTP : 400 demandes incorrectes

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "invalid",
      "details": {
        "text": "Invalid parameter combination provided"
      }
    }
  ]
}
```

Ressource introuvable

État HTTP : 404 introuvable

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "not-found",
      "details": {
        "text": "Resource not found."
      }
    }
  ]
}
```

Conflit de version

État HTTP : 409 Conflit

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "conflict",
      "details": {
        "text": "Resource version conflict detected"
      }
    }
  ]
}
```

État d'attribution non valide

État HTTP : 422 Entité non traitable

```
{
  "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'."
      }
    }
  ]
}
```

Règles commerciales

Validation du statut d'attribution

L'opération ne peut être effectuée que lorsque le statut d'attribution du groupe est :

- `draft`

- `open`

Les opérations ne sont pas autorisées lorsque le statut est défini`final`.

Prévention des doublons de membres

Le système empêche l'ajout de membres dupliqués sur la base de la combinaison unique de :

- Identifiant du membre
- Identifiant du payeur
- Identifiant de couverture

Validation des périodes de couverture

Lorsqu'une offre `attributionPeriod` est fournie, elle doit se situer dans les limites de la période de couverture du membre. Le système va :

- Recherchez la ressource de couverture du membre
- Utiliser la couverture la plus récente (versionID le plus élevé) s'il en existe plusieurs
- Validez que la période d'attribution ne dépasse pas la période de couverture

Validation des références

Lorsque l'identifiant et la référence sont fournis pour la même ressource (patient ou fournisseur), le système valide qu'ils correspondent à la même ressource.

Lorsque les champs ID et reference.identifier sont fournis pour la même ressource (patient ou fournisseur), une erreur est générée.

Authentification et autorisation

L'opération nécessite une autorisation SMART on FHIR pour :

- Permissions de lecture : pour valider les ressources du patient, du fournisseur et du groupe
- Autorisations de recherche : pour rechercher des ressources par identifiant
- Mettre à jour les autorisations : pour modifier la ressource du groupe

Comportement opérationnel

Mise à jour des ressources

- Met à jour l'ID de version de la ressource du groupe
- Crée une entrée d'historique avec l'état de la ressource d'origine avant l'opération

- Ajoute des informations sur les membres au tableau Group.member avec :
  - Référence du patient dans entity.reference
  - Période d'attribution dans la période
  - Couverture et informations sur les fournisseurs dans les domaines d'extension

Étapes de validation

- Validation des paramètres : garantit la validité des combinaisons de paramètres
- Existence des ressources : valide l'existence des ressources du patient, du fournisseur et du groupe
- État d'attribution : confirme que le statut du groupe autorise les modifications
- Duplicate Check - Empêche l'ajout de membres existants
- Validation de la couverture : garantit que la période d'attribution se situe dans les limites de couverture

Limitations

- Toutes les ressources référencées doivent exister dans la même banque de données
- L'opération ne fonctionne qu'avec les ressources du groupe de listes d'attribution de membres
- Les périodes d'attribution doivent se situer dans les limites de couverture
- Impossible de modifier les groupes ayant le statut « final »

`$member-match`opération pour HealthLake

AWS HealthLake soutient désormais le `$member-match` fonctionnement des ressources destinées aux patients, permettant aux établissements de santé de trouver l'identifiant unique d'un membre dans différents systèmes de santé à l'aide d'informations démographiques et de couverture. Cette opération est essentielle pour garantir la conformité au CMS et faciliter l'échange de payer-to-payer données sécurisé tout en préservant la confidentialité des patients.

Cette opération est particulièrement utile lorsque vous devez :

- Permettre un échange sécurisé de données de santé entre les organisations
- Assurer la continuité des soins aux patients dans les différents systèmes
- Support des exigences de conformité du CMS
- Faciliter l'identification précise des membres dans les réseaux de santé

Usage

L'`$member-match`opération peut être invoquée sur les ressources du patient à l'aide de la méthode POST :

```
POST [base]/Patient/$member-match
```

Paramètres pris en charge

HealthLake prend en charge les `$member-match` paramètres FHIR suivants :

Paramètre	Type	Obligatoire	Par défaut	Description
MemberPatient	Patient	Oui	—	Ressource destinée aux patients contenant des informations démographiques pour le membre à jumeler
CoverageToMatch	Couverture	Oui	—	Ressource de couverture qui sera utilisée pour la comparaison avec les enregistrements existants
CoverageToLink	Couverture	Non	—	Ressource de couverture à associer pendant le processus de jumelage
Consentement	Consentement	Non	—	Ressource de consentement à des fins d'autorisation

Exemples

Requête POST avec paramètres

```
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"
          }
        ]
      }
    }
  ]
}
```

Exemple de réponse

L'opération renvoie une ressource Parameters contenant les résultats correspondants :

```
{
  "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"
    }
  ]
}
```

Paramètres de réponse

La réponse inclut les paramètres suivants lorsqu'une correspondance est trouvée :

Paramètre	Type	Description
MemberIdentifier	Identifiant	L'identifiant unique du membre correspondant
MemberId	Référence	Référence à la ressource destinée aux patients

Paramètre	Type	Description
Algorithme de correspondance	String	Type d'algorithme de correspondance utilisé (EXACT_MATCH, STRONG_MATCH ou DEMOGRAPHIC_MATCH)
Détails du match	String	Informations détaillées sur le processus de mise en correspondance
Champs correspondants	String	Liste des champs spécifiques qui ont été correctement mis en correspondance

Algorithmes correspondants

L'`$member-match`API utilise une approche de correspondance à plusieurs niveaux pour garantir une identification précise des membres :

MATCH EXACT

- Utilise l'identifiant du patient combiné à la couverture SubscriberId
- Fournit le niveau de confiance le plus élevé pour le jumelage des membres

STRONG_MATCH

- Utilise l'identifiant du patient avec les informations de couverture minimales
- Offre un niveau de confiance élevé lorsque les critères de correspondance exacts ne sont pas respectés

MATCH DÉMOGRAPHIQUE

- S'appuie sur des informations démographiques de base
- Utilisé lorsque la correspondance basée sur un identifiant n'est pas possible

Comportement

L'`$member-match`opération :

- Accepte les données démographiques des patients, les détails de la couverture et les informations de consentement facultatives en entrée

- Renvoie un identifiant de membre unique qui peut être utilisé pour les interactions suivantes
- Met en œuvre un appariement à plusieurs niveaux (exact, solide, démographique) pour garantir l'identification précise des membres dans les différents systèmes de santé
- Enregistre toutes les informations de consentement fournies à des fins d'autorisation futures
- Permet un échange de payer-to-payer données sécurisé tout en préservant la confidentialité des patients
- Conforme aux exigences du CMS pour l'échange de données de santé

Autorisation

L'API utilise le protocole d'autorisation SMART on FHIR avec les étendues requises suivantes :

- `system/Patient.read`
- `system/Coverage.read`
- `system/Organization.read`(conditionnel)
- `system/Practitioner.read`(conditionnel)
- `system/PractitionerRole.read`(conditionnel)
- `system/Consent.write`(conditionnel)

Gestion des erreurs

L'opération gère les conditions d'erreur suivantes :

- `400 Bad Request`: `$member-match` opération non valide (demande non conforme ou paramètres obligatoires manquants)
- `422 Unprocessable Entity`: Aucune correspondance ou plusieurs correspondances trouvées

`$member-remove`opération pour HealthLake

L'`$member-remove`opération vous permet de supprimer des membres d'une liste d'attribution de membres FHIR (ressource de groupe) dans AWS HealthLake. Cette opération fait partie du guide de mise en œuvre de l'attribution des DaVinci membres et soutient le processus de rapprochement pour la gestion des attributions des membres.

Conditions préalables

- AWS HealthLake Banque de données FHIR
- Autorisations IAM appropriées pour les opérations HealthLake
- Liste d'attribution des membres (ressource du groupe) en version provisoire ou ouverte

Détails de l'opération

Endpoint

```
POST /Group/{id}/$member-remove
```

Type de contenu

```
application/fhir+json
```

Parameters

L'opération accepte une ressource de paramètres FHIR avec les paramètres facultatifs suivants :

Paramètre	Cardinalité	Type	Description
memberId	0,1	Identifiant	Identifiant professionnel du membre à supprimer
Fournisseur NPI	0,1	Identifiant	NPI du fournisseur attribué
Référence du patient	0,1	Référence	Référence directe à la ressource destinée aux patients
Référence du fournisseur	0,1	Référence	Référence directe à la ressource du fournisseur (praticien ou organisat ion) PractitionerRole
Référence de couverture	0,1	Référence	Référence à la ressource Coverage

Combinaisons de paramètres prises en charge

Les combinaisons de paramètres suivantes sont prises en charge :

- `memberId`uniquement - Supprime toutes les attributions pour le membre spécifié
- `memberId`+ `providerNpi` - Supprime les attributions pour la combinaison membre-fournisseur spécifique
- `patientReference`uniquement - Supprime toutes les attributions pour le patient spécifié
- `patientReference`+ `providerReference` - Supprime les attributions pour la combinaison patient-fournisseur spécifique
- `patientReference`+ `providerReference` + `coverageReference` - Supprime l'attribution spécifique en fonction du patient, du fournisseur et de la couverture

Exemple de requête

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "patientReference",
      "valueReference": {
        "reference": "Patient/12345"
      }
    },
    {
      "name": "providerReference",
      "valueReference": {
        "reference": "Practitioner/67890"
      }
    }
  ]
}
```

Réponse

Réponse réussie

```
{
  "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"
    }
  ]
}
```

Comportement

Exigences relatives au statut

- L'opération ne fonctionne que sur les listes d'attribution avec statut `draft` ou `open`
- Les listes avec un `final` statut rejetteront l'opération avec une erreur 422

Processus de suppression d'un membre

- Brouillons de listes de statut : les membres sont marqués comme inactifs (`inactive: true`) et leur `changeType` extension est mise à jour pour `changed`
- Listes de statut ouvertes : comportement similaire à celui du statut des brouillons
- Listes de statut final : l'opération est rejetée

Validation

- Les références sont validées pour garantir leur existence dans la HealthLake banque de données
- Si l'identifiant et la référence sont fournis pour le même type de ressource, ils doivent correspondre à la même ressource
- Les combinaisons de paramètres sont validées selon les modèles pris en charge

Gestion des erreurs

Réponses aux erreurs courantes

Ressource introuvable (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"]
    }
  ]
}
```

État final de la liste d'attribution (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"]
    }
  ]
}
```

Opération non valide (400)

Renvoyé lorsque les combinaisons de paramètres ne sont pas valides ou sont mal formées.

Plusieurs résultats trouvés (412)

Renvoyé lorsque les paramètres fournis correspondent à plusieurs membres de la liste d'attribution.

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "processing",
      "diagnostics": "Multiple members found matching the criteria"
    }
  ]
}
```

Bonnes pratiques

- Utiliser des paramètres spécifiques : Dans la mesure du possible, utilisez la combinaison de paramètres la plus spécifique pour éviter les suppressions involontaires
- État de la liste de vérification : vérifiez l'état de la liste d'attribution avant de tenter de le supprimer
- Gérez les erreurs avec élégance : implémentez une gestion des erreurs appropriée pour toutes les conditions d'erreur possibles
- Valider les références : assurez-vous que toutes les ressources référencées existent avant de faire la demande

Supprimer les ressources du compartiment réservé aux patients grâce à **$purge**

AWS HealthLake soutient l'`$purge`opération, permettant la suppression permanente de toutes les ressources présentes dans le compartiment du patient. Cette opération est particulièrement utile lorsque vous devez :

- Supprimer toutes les données associées à un patient
- Respectez les demandes de suppression des données des patients
- Gérer le cycle de vie des données des patients
- Effectuez un nettoyage complet des dossiers des patients

Usage

L'`$purge`opération peut être invoquée sur les ressources du patient :

```
POST [base]/Patient/[ID]/$purge?deleteAuditEvent=true
```

Parameters

Paramètre	Type	Obligatoire	Par défaut	Description
`deleteAuditEvent`	booléen	Non	false	Lorsque c'est vrai, supprime les événements d'audit associés
`_since`	chaîne	Non	Heure de création de la banque de données	Une fois saisie, sélectionne l'heure limite de début pour rechercher les ressources en fonction de leur heure de dernière modification. Ne peut pas être utilisé avec le début ou la fin
`start`	chaîne	Non	Heure de création de la banque de données	Une fois saisie, sélectionne l'heure limite pour rechercher les ressources en fonction de leur heure de dernière modification. Peut être utilisé avec extrémité
`end`	chaîne	Non	Heure de soumission des offres d'emploi	Une fois saisi, sélectionne l'heure limite de fin pour rechercher les ressources en fonction de leur heure de dernière modification

Exemples

Exemple de requête

```
POST [base]/Patient/example-patient/$purge?deleteAuditEvent=true
```

Exemple de réponse

```
{
  "resourceType": "OperationOutcome",
  "id": "purge-job",
  "issue": [
    {
      "severity": "information",
      "code": "informational",
      "diagnostics": "Purge job started successfully. Job ID:
 12345678-1234-1234-1234-123456789012"
    }
  ]
}
```

Statut de la tâche

Pour vérifier l'état d'une tâche de purge :

```
GET [base]/$purge/[jobId]
```

L'opération renvoie les informations relatives à l'état de la tâche :

```
{
      "datastoreId": "36622996b1fcecb7e12ee2ee085308d3",
      "jobId": "3dd1c7a5b6c0ef8c110f566eb87e2ef9",
      "status": "COMPLETED",
      "submittedTime": "2025-10-31T18:43:21.822Z"
    }
```

Comportement

L'`$purge`opération :

1. Processus asynchrones pour gérer plusieurs ressources
2. Maintient les transactions ACID pour garantir l'intégrité des données
3. Fournit un suivi de l'état des tâches avec le nombre de ressources supprimées
4. Supprime définitivement toutes les ressources du compartiment du patient
5. Inclut un enregistrement d'audit complet des activités de suppression
6. Supporte la suppression sélective des événements d'audit

Journalisation des audits

L'`$purge`opération est enregistrée sous les noms Start FHIRBulk DeleteJob et Describe FHIRBulk DeleteJob avec des informations détaillées sur l'opération.

Limitations

- Les ressources purgées n'apparaîtront pas dans les réponses de recherche
- Les ressources en cours de purge peuvent être temporairement inaccessibles pendant le traitement
- Toutes les ressources du compartiment patient sont définitivement supprimées

`$questionnaire-package`Opération FHIR pour HealthLake

L'`$questionnaire-package`opération récupère un ensemble complet contenant un questionnaire FHIR et toutes ses dépendances nécessaires au rendu et au traitement du questionnaire. Cette opération met en œuvre le guide de mise en œuvre des modèles et règles de documentation Da Vinci (DTR), permettant le rendu dynamique des formulaires pour les exigences en matière de documentation dans les flux de travail du secteur de la santé.

Comment ça marche

- Demande : vous envoyez des paramètres identifiant le ou les questionnaires nécessaires, ainsi que la couverture et le contexte de la commande
- Récupérer : HealthLake rassemble le questionnaire et toutes les dépendances (ValueSetsbibliothèques CQL, etc.)
- Package : Toutes les ressources sont regroupées dans un format standardisé
- Répondre : vous recevez un package complet prêt pour le rendu et la collecte de données

Cas d'utilisation

- Documentation d'autorisation préalable : Collectez les informations cliniques requises pour les demandes d'autorisation préalable
- Exigences de couverture : Rassemblez la documentation nécessaire pour satisfaire aux exigences de couverture du payeur
- Clinical Data Exchange : structurez les données cliniques pour les soumettre aux payeurs
- Formulaires dynamiques : créez des questionnaires avec des données préremplies sur les patients et une logique conditionnelle

Point de terminaison d'API

```
POST /datastore/{datastoreId}/r4/Questionnaire/$questionnaire-package
Content-Type: application/fhir+json
```

Paramètres de demande

Paramètres d'entrée

Le corps de la demande doit contenir une ressource de paramètres FHIR avec les paramètres suivants :

Paramètre	Type	Cardinalité	Description
`coverage`	Couverture	1.. * (Obligatoire)	Ressource (s) de couverture pour établir le membre et couverture pour la documentation
`questionnaire`	canonial	0.. *	URL (s) canoniques pour un ou plusieurs questionnaires spécifiques à renvoyer (peut inclure la version)
`order`	Ressource	0.. *	Commandez des ressources (DeviceRequest, ServiceRequest, MedicationRequest, Encounter, Appointment) pour établir le contexte

Paramètre	Type	Cardinalité	Description
`changedSince`	dateTime	0,1	Le cas échéant, ne renvoie que les ressources modifiées après cet horodatage

Règles de validation des paramètres

Au moins UN des éléments suivants doit être fourni (en plus de ce qui est obligatoire`coverage`) :

- Un ou plusieurs `questionnaire` canoniques URLs
- Une ou plusieurs `order` ressources

Combinaisons de demandes valides :

- `coverage` + `questionnaire`
- `coverage` + `order`
- `coverage` + `questionnaire` + `order`

Exemple de demande

```
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"
    }
  ]
}
```

Format de la réponse

Réponse positive (200 OK)

L'opération renvoie une ressource de paramètres FHIR contenant un ou plusieurs Package Bundles. Chaque Package Bundle inclut :

Type d'entrée	Cardinalité	Description
Questionnaire	1	Le questionnaire à rendre
QuestionnaireResponse	0,1	Réponse préremplie ou partiellement complétée (le cas échéant)
d'outils	0.. *	Bibliothèques CQL contenant une logique de pré-peuplement et une logique conditionnelle
ValueSet	0.. *	Étendu ValueSets (pour les choix de réponses avec moins de 40 extensions)

Example Exemple de réponse

```
{
  "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"
          }
        }]
      }
    }
  ]
}
```

Flux de travail des opérations

Comment HealthLake traite-t-on votre demande

Lorsque vous appelez`$questionnaire-package`, HealthLake exécute les étapes suivantes :

1. Identifier le patient et le payeur : extrait le patient et l'organisme d'assurance de vos `coverage` paramètres.
2. Trouvez le bon questionnaire :
   - Avec **`questionnaire`** paramètre : utilise l'URL canonique que vous avez fournie
   - Avec **`order`** paramètre : fait correspondre le code de commande (CPT/HCPCS/LOINC) et le payeur pour trouver le questionnaire approprié
3. Collecter les dépendances : récupère automatiquement tout ce qui est nécessaire pour afficher le questionnaire :
   - Bibliothèques CQL - Logique pour les questions de prépopulation et les questions conditionnelles
   - ValueSets- Choix de réponses (automatiquement étendus si <40 options)
   - QuestionnaireResponse- Toutes les réponses existantes en cours ou terminées
4. Emballez le tout ensemble :

- Regroupe toutes les ressources (chaque ressource n'est incluse qu'une seule fois)
- Filtre par `changedSince` horodatage s'il est fourni
- Ajoute des avertissements `Outcome` si des ressources sont manquantes

Résultat : un package complet et autonome prêt pour le rendu.

Réponses d'erreur

400 Requête erronée

Renvoyé en cas d'échec de la validation de la demande.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "required",
    "details": {
      "text": "At least one of 'questionnaire' or 'order' must be provided along with
 'coverage'"
    }
  }]
}
```

424 Dépendance défaillante

Renvoyé lorsqu'une ressource dépendante ne peut pas être récupérée.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "warning",
    "code": "not-found",
    "details": {
      "text": "Referenced Library 'http://example.org/fhir/Library/missing-library'
 could not be retrieved"
    }
  }]
}
```

401 Accès non autorisé

Renvoyé lorsque les informations d'authentification sont manquantes ou non valides.

403 Forbidden

Renvoyé lorsque l'utilisateur authentifié n'est pas autorisé à accéder aux ressources demandées.

406 Non acceptable

Renvoyé lorsque le type de contenu demandé ne peut pas être fourni.

409 – Conflit

Renvoyé en cas de conflit de version ou de simultanéité.

410 Disparus

Renvoyé lorsque la ressource demandée a été définitivement supprimée.

429 Trop de demandes

Renvoyé lorsque les limites de débit sont dépassées.

500 Erreur de serveur interne

Renvoyé lorsqu'une erreur de serveur inattendue se produit.

501 Non implémenté

Renvoyé lorsque l'opération demandée n'est pas encore implémentée.

Règles de validation

Validation des entrées

- `coverage`le paramètre est obligatoire (1.. * cardinalité)
- Au moins l'un `questionnaire` des éléments `order` suivants doit être fourni
- Toutes les ressources de couverture doivent être des ressources FHIR valides
- Toutes les ressources de la commande doivent être des ressources FHIR valides
- Canonical URLs doit être correctement formaté

- `changedSince`doit être un DateTime ISO 8601 valide

QuestionnaireResponse validation

- `status`doit être approprié (`in-progress`,`completed`,`amended`)
- La structure doit correspondre au questionnaire référencé
- `basedOn`doit faire référence à des ressources de commande valides
- `subject`doit faire référence à des ressources valides pour les patients

Déduplication des ressources

- Chaque ressource n'apparaît qu'une seule fois dans le bundle
- Exception : différentes versions de la même ressource peuvent toutes deux être incluses
- Les ressources sont identifiées par leur URL canonique et leur version

Spécifications de performance

Métrique	Spécification de
Limite du nombre de ressources	500 ressources par bundle
Limite de taille du bundle	5 Mo maximum

Autorisations requises

Pour utiliser l'`$questionnaire-package`opération, assurez-vous que votre rôle IAM possède les éléments suivants :

- `healthlake:QuestionnairePackage`- Pour appeler l'opération
- `healthlake:ReadResource`- Pour récupérer le questionnaire et les ressources dépendantes
- `healthlake:SearchWithPost`- Pour rechercher QuestionnaireResponse des ressources connexes

SMART sur les oscilloscopes FHIR

Étendue minimale requise :

- SMART version 1 : `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`

Remarques de mise en œuvre importantes

Stratégie de récupération des ressources

Priorité d'identification du questionnaire :

- URL canonique (si le `questionnaire` paramètre est fourni) - Priorité la plus élevée
- Analyse de la commande (si `order` le paramètre est fourni) :
  - Associez les codes de commande (CPT, HCPCS, LOINC) aux politiques médicales du payeur
  - Utilisez le payeur de couverture pour filtrer les questionnaires spécifiques au payeur
  - Tenez compte des codes de motif pour un contexte supplémentaire

Résolution des dépendances

Bibliothèques CQL :

- Récupéré via l'`cqf-library`extension sur les ressources du questionnaire
- Récupère récursivement les bibliothèques dépendantes via le type `Library.relatedArtifact` `depends-on`
- Toutes les dépendances de bibliothèque sont incluses dans le package

ValueSets:

- Développé automatiquement s'ils contiennent moins de 40 concepts
- ValueSets Les plus grands sont inclus sans extension
- ValueSets référencées à la fois dans le questionnaire et les ressources de la bibliothèque sont incluses

QuestionnaireResponse pré-population

L'opération peut renvoyer un fichier QuestionnaireResponse avec des données préremplies lorsque :

- Une réponse existante en cours ou terminée est trouvée
- La logique CQL des bibliothèques associées peut extraire des données des dossiers des patients
- La réponse est liée à la commande et à la couverture pertinentes

Critères de recherche pour QuestionnaireResponse :

Paramètre de recherche	Parcours FHIR	Description
`based-on`	`QuestionnaireResponse.basedOn`	Liens vers ServiceRequest ou CarePlan
`patient`	`QuestionnaireResponse.subject`	Le patient qui est le sujet
`questionnaire`	`QuestionnaireResponse.questionnaire`	Le questionnaire auquel il est répondu

Filtrage des ressources modifié

Lorsque le `changedSince` paramètre est fourni :

- Seules les ressources modifiées après l'horodatage spécifié sont incluses
- Si aucune ressource n'a changé, retourne `200 OK` avec un package vide
- Utile pour les mises à jour incrémentielles et les stratégies de mise en cache
- La comparaison des horodatages utilise le champ de ressources `meta.lastUpdated`

Forfaits groupés multiples

L'opération peut renvoyer plusieurs Package Bundles lorsque :

- Plusieurs questionnaires sont demandés via Canonical URLs

- Les commandes multiples nécessitent des questionnaires différents
- Différentes versions du même questionnaire sont applicables

Chaque Package Bundle est autonome avec toutes les dépendances nécessaires.

Cas d’utilisation courants

Cas d'utilisation 1 : documentation relative à l'autorisation préalable

Scénario : Un fournisseur doit recueillir de la documentation pour une autorisation préalable d'oxygénothérapie à domicile.

```
{
  "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"
          }]
        }
      }
    }
  ]
}
```

Résultat : renvoie un colis contenant le questionnaire d'oxygénothérapie, prérempli avec les données vitales du patient et les codes de diagnostic du dossier médical électronique.

Cas d'utilisation 2 : récupérer une version spécifique du questionnaire

Scénario : un fournisseur a besoin d'une version spécifique d'un questionnaire pour se conformer.

```
{
```

```
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "coverage",
      "resource": { /* Coverage resource */ }
    },
    {
      "name": "questionnaire",
      "valueCanonical": "http://example.org/fhir/Questionnaire/dme-request|3.1.0"
    }
  ]
}
```

Résultat : renvoie exactement la version 3.1.0 du questionnaire de demande DME avec toutes les dépendances.

Cas d'utilisation 3 : Vérifier les mises à jour

Scénario : un fournisseur souhaite vérifier si des ressources du questionnaire ont été mises à jour depuis la dernière extraction.

```
{
  "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"
    }
  ]
}
```

Résultat : renvoie uniquement les ressources qui ont été modifiées après le 1er mars 2024, ou un package vide si rien n'a changé.

Cas d'utilisation 4 : commandes multiples

Scénario : un fournisseur soumet plusieurs demandes de service qui peuvent nécessiter différents questionnaires.

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "coverage",
      "resource": { /* Coverage resource */ }
    },
    {
      "name": "order",
      "resource": { /* ServiceRequest for imaging */ }
    },
    {
      "name": "order",
      "resource": { /* ServiceRequest for DME */ }
    }
  ]
}
```

Résultat : renvoie plusieurs Packages groupés, un pour chaque questionnaire applicable.

Intégration avec d'autres Da Vinci IGs

Découverte des exigences de couverture (CRD)

Intégration du flux de travail :

- Le fournisseur commande un service dans son dossier électronique
- Feux à l'hameçon CRD, vérification des exigences de couverture
- Le payeur répond en indiquant que la documentation est nécessaire
- Le fournisseur appelle `$questionnaire-package` pour récupérer le formulaire de documentation
- Le fournisseur remplit le questionnaire
- La documentation est soumise via PAS ou CDex

Support d'autorisation préalable (PAS)

Intégration du flux de travail :

- `$questionnaire-package`À utiliser pour récupérer les exigences en matière de documentation
- Complétez le QuestionnaireResponse avec les données cliniques requises
- Soumettez l'autorisation préalable `Claim/$submit` en utilisant le QuestionnaireResponse
- Vérifiez le statut à l'aide de `Claim/$inquire`

Échange de données cliniques (CDex)

Intégration du flux de travail :

- Le payeur demande des documents supplémentaires pour une réclamation
- Le fournisseur utilise `$questionnaire-package` pour récupérer le formulaire de collecte de données structuré
- Le fournisseur complète le QuestionnaireResponse
- La documentation est soumise au payeur via un flux de travail de CDex pièces jointes

Guide de dépannage

Problème : aucun questionnaire n'a été renvoyé

Causes possibles :

- L'URL canonique ne correspond à aucun questionnaire du magasin de données
- Le code de commande ne correspond à aucun questionnaire de la police médicale du payeur
- Le payeur de couverture n'a pas de questionnaires associés

Solutions :

- Vérifiez que l'URL canonique est correcte et que le questionnaire existe
- Vérifiez que les codes de commande (CPT/HCPCS) sont correctement spécifiés

- Vérifiez que l'organisation payeuse a configuré des questionnaires

Problème : dépendances manquantes dans le package

Causes possibles :

- Bibliothèque référencée ou ValueSet n'existe pas dans le magasin de données
- Les références de bibliothèque sont cassées ou incorrectes
- ValueSet échec de l'extension

Solutions :

- Vérifiez le `Outcome` paramètre pour les avertissements concernant les ressources manquantes
- Vérifiez que toutes les ressources référencées existent dans votre magasin de données
- Assurez-vous qu' ValueSet URLs ils sont corrects et résolvables

Problème : Package vide avec ChangedSince

Causes possibles :

- Ce comportement est attendu : aucune ressource n'a été modifiée depuis l'horodatage spécifié

Solutions :

- Utiliser la version mise en cache du package
- Supprimer `changedSince` le paramètre pour récupérer le package complet

Problème : QuestionnaireResponse Non prérempli

Causes possibles :

- Aucun objet existant n' QuestionnaireResponse a été trouvé
- La logique de la bibliothèque CQL n'a pas pu extraire les données requises
- Les données du patient sont manquantes ou incomplètes

Solutions :

- On peut s'y attendre, car tous les questionnaires ne reposent pas sur une logique de pré-population
- Vérifiez que les données du patient existent dans le magasin de données
- Vérifiez la logique de la bibliothèque CQL pour connaître les exigences en matière d'extraction de données

Bonnes pratiques

1. Utiliser Canonical URLs avec les versions

Spécifiez toujours les numéros de version lorsque vous demandez des questionnaires spécifiques :

```
{
  "name": "questionnaire",
  "valueCanonical": "http://example.org/fhir/Questionnaire/dme|2.1.0"
}
```

Pourquoi : Garantit la cohérence et empêche les changements inattendus lors de la mise à jour des questionnaires.

2. Tirez parti de ChangedSince pour améliorer les performances

Pour les questionnaires fréquemment consultés, utilisez `changedSince` pour minimiser le transfert de données :

```
{
  "name": "changedSince",
  "valueDateTime": "2024-03-10T15:30:00Z"
}
```

Pourquoi : réduit la latence et l'utilisation de la bande passante en ne récupérant que les ressources mises à jour.

3. Incluez des informations complètes sur la couverture

Fournissez des informations complètes sur la couverture afin de garantir une sélection correcte du questionnaire :

```
{
  "name": "coverage",
  "resource": {
    "resourceType": "Coverage",
    "beneficiary": { "reference": "Patient/123" },
    "payor": [{ "reference": "Organization/payer-abc" }],
    "class": [{ /* Group/plan information */ }]
  }
}
```

Pourquoi : aide à HealthLake identifier les questionnaires et les exigences spécifiques au payeur.

Opérations liées

- `Claim/$submit`- Soumettre les demandes d'autorisation préalable avec la documentation complète
- `Claim/$inquire`- Vérifier l'état des autorisations antérieures soumises
- `ValueSet/$expand`- Élargir ValueSets pour les choix de réponses

`$submit`Opération FHIR pour HealthLake

L'`$submit`opération vous permet de soumettre électroniquement des demandes d'autorisation préalable aux payeurs pour approbation. Cette opération met en œuvre le guide de mise en œuvre du Da Vinci Prior Authorization Support (PAS), fournissant un flux de travail standardisé basé sur le FHIR pour les soumissions d'autorisations préalables.

Comment ça marche

- Soumettre : vous envoyez un bundle FHIR contenant votre demande d'autorisation préalable et les données cliniques à l'appui
- Valider : HealthLake valide la soumission par rapport aux exigences du PAS
- Persister : toutes les ressources sont stockées dans votre magasin HealthLake de données
- Répondre : vous recevez une réponse immédiate avec le statut « en file d'attente »
- Processus : La décision d'autorisation est traitée de manière asynchrone par le payeur

Point de terminaison d'API

```
POST /datastore/{datastoreId}/r4/Claim/$submit
```

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

Structure de la demande

Exigences relatives au bundle

Votre demande doit être une ressource FHIR Bundle contenant :

- Bundle.type : Doit être `"collection"`
- Bundle.entry : doit contenir exactement une ressource Claim avec `use = "preauthorization"`
- Ressources référencées : Toutes les ressources référencées par la réclamation doivent être incluses dans le bundle

Ressources requises

Ressource	Cardinalité	Profil	Description
Réclamation	1	Réclamation PAS	La demande d'autorisation préalable
Patient	1	Patient PAS	Informations démographiques sur les patients
Organisation (assureur)	1	Assureur PAS	Compagnie d'assurance
Organisation (fournisseur)	1	Demandeur PAS	Prestataire de santé soumettant la demande
Couverture	1 ou plus	Couverture PAS	Détails de la couverture d'assurance

Ressources facultatives

Ressource	Cardinalité	Profil	Description
Praticien	0 ou plus	Praticien PAS	Praticiens de santé

Ressource	Cardinalité	Profil	Description
PractitionerRole	0 ou plus	PAS PractitionerRole	Rôles des praticiens
ServiceRequest	0 ou plus	PAS ServiceRequest	Services médicaux demandés
DeviceRequest	0 ou plus	PAS DeviceRequest	Dispositifs médicaux demandés
MedicationRequest	0 ou plus	PAS MedicationRequest	Médicaments demandés
DocumentReference	0 ou plus	PAS DocumentReference	Documentation clinique à l'appui

Exemple de demande

```
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"
    }
  }]
}
```

Format de la réponse

Réponse positive (200 OK)

Vous recevrez un pack de réponses PAS contenant :

- ClaimResponseavec `outcome: "queued"` et `status: "active"`
- Toutes les ressources originales de votre demande
- Horodatage confirmant la réception

```
{
  "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"
    }
  }]
}
```

Réponses d'erreur

400 Requête erronée

Renvoyé lorsque le format de demande n'est pas valide ou est mal formé.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "invalid",
    "diagnostics": "The provided payload was invalid and could not be parsed
 correctly."
  }]
}
```

412 – Échec de condition préalable

Renvoyé lorsque la même demande d'autorisation préalable a déjà été soumise (envoi dupliqué détecté).

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "processing",
    "diagnostics": "PreAuth Claim already exists"
  }]
}
```

Idempotence

L'`$submit`opération est idempotente. Le fait de soumettre la même demande plusieurs fois ne créera pas de demandes d'autorisation préalable dupliquées. Au lieu de cela, vous recevrez un message d'erreur 412 vous demandant de `$inquire` vérifier le statut de votre envoi initial.

422 Entité non traitable

Renvoyé en cas d'échec de la validation FHIR.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
```

```
    "code": "required",
    "diagnostics": "Bundle contains more than one preauthorization claim"
  }]
}
```

Règles de validation

HealthLake effectue une validation complète de votre soumission :

Validation du bundle

- Doit être conforme au profil du bundle de demandes PAS
- `Bundle.type`doit être `"collection"`
- Peut contenir plusieurs ressources de réclamation
- Cependant, il doit contenir exactement une ressource Claim avec une utilisation préautorisée
  - Et cette ressource Claim doit être la première entrée du bundle
- Toutes les ressources référencées doivent être incluses dans le bundle

Validation des réclamations

- Doit être conforme au profil de réclamation PAS
- `Claim.use`doit être `"preauthorization"`
- Champs obligatoires : `patient`,`insurer`,`provider`,`created`, `priority`
- Les identifiants commerciaux doivent être présents et valides

Validation des ressources

- Toutes les ressources doivent être conformes à leurs profils PAS respectifs
- Les ressources de soutien requises doivent être présentes (patient, couverture, organisation)
- Les références croisées doivent être valides et résolvables dans le bundle

Spécifications de performance

Métrique	Spécification de
Limite de taille du bundle	5 Mo maximum

Métrique	Spécification de
Limite du nombre de ressources	500 ressources par bundle

Autorisations requises

Pour utiliser l'`$submit`opération, on peut utiliser AWS Sigv4 ou SMART sur FHIR :

- Assurez-vous que votre rôle IAM comporte : `healthlake:SubmitPreAuthClaim` - Pour appeler l'opération

SMART sur les oscilloscopes FHIR

Étendue minimale requise :

- SMART version 1 : `user/Claim.write & <all_resourceTypes_in_Bundle>.write`
- SMART v2 : `user/Claim.c & <all_resourceTypes_in_Bundle>.c or system/*.*`

Remarques de mise en œuvre importantes

Persistance des ressources

- Toutes les entrées du bundle sont conservées en tant que ressources FHIR individuelles dans votre magasin de données
- Les produits fournis par le client IDs sont conservés lorsqu'ils sont fournis
- L'historique des versions est conservé à des fins d'audit
- La détection des doublons prévient les conflits de ressources

Comportement de traitement

- Chaque soumission valide donne lieu à exactement une ClaimResponse avec `"queued"` résultat
- Les soumissions non valides renvoient 400 ou 422 codes d'état avec des informations d'erreur détaillées
- Les erreurs du système renvoient les codes d'état 5xx appropriés
- Toutes les soumissions réussies renvoient le statut 200 avec un statut en attente ClaimResponse

Exigences relatives au bundle

- `Bundle.entry.fullUrl`les valeurs doivent être soit au format REST, URLs soit `"urn:uuid:[guid]"` au format
- Toutes les soumissions GUIDs doivent être uniques (sauf pour les mêmes instances de ressources)
- Les ressources référencées doivent exister dans le bundle ou être résolvables

Opérations liées

- `Claim/$inquire`- Vérifier le statut d'une demande d'autorisation préalable soumise
- `Patient/$everything`- Récupérez les données complètes du patient pour le contexte de l'autorisation préalable

Validation des ressources FHIR avec **`$validate`**

AWS HealthLake prend désormais en charge le `$validate` fonctionnement des ressources FHIR, ce qui vous permet de valider une ressource par rapport à la spécification FHIR et de vérifier sa conformité à un profil spécifié ou à une définition de ressource de base sans effectuer d'opérations de stockage. Cette opération est particulièrement utile lorsque vous devez :

- Valider les exigences de conformité du CMS FHIR
- Testez les ressources avant de les utiliser en production
- Fournir des commentaires de validation en temps réel lorsque les utilisateurs modifient les données cliniques
- Réduisez les soumissions de données non valides pour la création et la mise à jour APIs

Usage

L'`$validate`opération peut être invoquée sur les ressources FHIR à l'aide des méthodes POST :

Opérations prises en charge

```
POST [base]/[type]/[id]/$validate
POST [base]/[type]/$validate
```

Charges utiles prises en charge

Ressource de paramètres

HealthLake prend en charge les `$validate` paramètres FHIR suivants :

Paramètre	Type	Obligatoire	Description
`resource`	Ressource	Oui	La ressource à valider
`profile`	canonial	Non	URL canonique du profil par rapport auquel valider
`mode`	code	Non	Mode de validation :`create`, ou `update`

Ressource directe avec paramètres de requête

Paramètre	Type	Obligatoire	Description
`profile`	canonial	Non	URL canonique du profil par rapport auquel valider
`mode`	code	Non	Mode de validation :`create`, ou `update`

Exemples

Demande POST pour une ressource avec ID et paramètres (charge utile)

```
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"
    }
  ]
}
```

Demande POST pour le type de ressource et la charge utile des paramètres

```
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"
    }
  ]
}
```

Demande de ressource POST avec identifiant et charge utile directe de la ressource

```
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"
}
```

Demande POST pour le type de ressource et la charge utile directe de la ressource

```
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"
}
```

Exemple de réponse

L'opération renvoie une OperationOutcome ressource avec les résultats de validation :

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "information",
      "code": "informational",
      "diagnostics": "Validation successful"
    }
  ]
}
```

Exemple de réponse avec erreurs de validation

```
{
  "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"
      ]
    }
  ]
}
```

Comportement

L'`$validate`opération :

1. Valide la ressource par rapport à la spécification FHIR et à la définition de la ressource de base
2. Vérifie la conformité aux profils spécifiés lorsque le `profile` paramètre est fourni
3. Valide en fonction du mode spécifié (`create`ou`update`)
4. Renvoie les résultats de validation détaillés, y compris les erreurs, les avertissements et les messages d'information
5. N'effectue aucune opération de stockage - validation uniquement
6. Renvoie HTTP 200 OK lorsque la validation peut être effectuée, que des problèmes de validation soient détectés ou non

Modes de validation

- créer : valide la ressource comme si elle était en cours de création (nouvelle ressource)
- mise à jour : valide la ressource comme si elle était mise à jour (ressource existante)

Gestion des erreurs

L'opération renvoie :

- 200 OK : La validation a été effectuée avec succès (quel que soit le résultat de la validation)
- 400 Demande incorrecte : format ou paramètres de demande non valides
- 404 Introuvable : type de ressource ou profil introuvable

Pour plus d'informations sur les spécifications de `$validate` fonctionnement, consultez la documentation des ressources `$validate` FHIR R4.

Référence de conformité pour AWS HealthLake

AWS HealthLake fournit des fonctionnalités conçues pour vous aider à suivre et à signaler l'utilisation des API conformément aux exigences d'interopérabilité des CMS (Centers for Medicare & Medicaid Services). Ces fonctionnalités vous permettent de classer les transactions d'API par catégories mandatées par le CMS et de capturer automatiquement les statistiques d'utilisation à des fins de rapports de conformité.

⚠ Comprendre vos responsabilités en matière de conformité

L'utilisation AWS HealthLake de points de terminaison d'interopérabilité avec le CMS ne suffit pas à elle seule pour garantir la conformité avec le CMS. Vous êtes responsable de :

- Associer correctement vos flux de travail d'API aux points de terminaison de la catégorie CMS appropriée en fonction de vos cas d'utilisation spécifiques et de vos obligations réglementaires
- Mettre en œuvre des contrôles d'authentification et d'autorisation appropriés qui répondent aux exigences du CMS
- Veiller à ce que vos ressources FHIR et vos échanges de données soient conformes aux réglementations CMS applicables et aux guides de mise en œuvre
- Configuration et surveillance des CloudWatch indicateurs pour répondre à vos besoins en matière de rapports de conformité
- Comprendre quelles règles du CMS s'appliquent à votre organisation et mettre en œuvre les contrôles techniques et opérationnels appropriés

AWS HealthLake fournit l'infrastructure et les outils nécessaires pour soutenir vos efforts de mise en conformité, mais vous devez utiliser ces fonctionnalités de manière appropriée en fonction de vos exigences réglementaires spécifiques. Le simple fait de router les appels d'API via ces points de terminaison ne rend pas automatiquement votre application conforme aux réglementations du CMS.

Rubriques

Fonctionnalités de conformité du CMS

AWS HealthLake fournit des fonctionnalités pour vous aider à répondre aux exigences d'interopérabilité et de conformité des CMS (Centers for Medicare & Medicaid Services). Ces fonctionnalités vous permettent de suivre l'utilisation des API par catégorie de CMS et de signaler ensuite les mesures d'utilisation à des fins de conformité.

Rubriques

Points de terminaison d'interopérabilité CMS

Présentation de

HealthLake fournit quatre points de terminaison d'interopérabilité du CMS qui correspondent aux catégories d'API mandatées par le CMS. L'URL de base sous-jacente de votre HealthLake banque de données ne change pas. Ces points de terminaison fournissent simplement un moyen de classer et de suivre vos appels d'API à des fins de reporting CMS.

Objectif

L'objectif principal de ces points de terminaison d'interopérabilité est de permettre aux clients de :

- Suivez facilement les transactions d'API par catégorie de CMS
- Signalez automatiquement les mesures d'utilisation pour garantir la conformité au CMS
- Maintenir les flux de travail FHIR existants avec un minimum de modifications

Tous les appels d'API fonctionnent de la même manière, que vous utilisiez le point de terminaison d'interopérabilité ou le point de terminaison FHIR standard. La seule différence réside dans la manière dont les transactions sont classées dans les métriques. CloudWatch

Points de terminaison d'interopérabilité CMS pris en charge

Catégorie CMS	Endpoint d'interopérabilité	Exemple d'utilisation
Accès pour les patients	`/patientaccess/v2/r4`	`baseURL/patientaccess/v2/r4/Patient/123`

Catégorie CMS	Endpoint d'interopérabilité	Exemple d'utilisation
Accès du fournisseur	`/provideraccess/v2/r4`	`baseURL/provideraccess/v2/r4/Observation?patient=123`
Payeur à payeur	`/payertopayerdx/v2/r4`	`baseURL/payertopayerdx/v2/r4/Practitioner/456`
Service d'authentification préalable	`/priorauthservice/v2/r4`	`baseURL/priorauthservice/v2/r4/ExplanationOfBenefit?patient=789`

Comment fonctionnent les points de terminaison d'interopérabilité

Appel HealthLake d'API standard :

```
baseURL/resourceType/[id]
baseURL/resourceType?[parameters]
```

Avec CMS Interoperability Endpoint :

```
baseURL/interoperability-endpoint/resourceType/[id]
baseURL/interoperability-endpoint/resourceType?[parameters]
```

Le chemin du point de terminaison d'interopérabilité est simplement inséré entre votre URL de base et le type de ressource. Après le chemin du point de terminaison d'interopérabilité, tout reste exactement le même que celui de vos appels d'API actuels.

Exemples d’utilisation

Exemple 1 : API d'accès aux patients

Appel d'API en cours (fonctionne toujours) :

```
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"
```

Avec le point de terminaison d'interopérabilité Patient Access (pour le suivi du CMS) :

```
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"
```

Points clés :

- L'URL de base reste la suivante : `https://healthlake.us-east-1.amazonaws.com/datastore/abc123`
- Point de terminaison d'interopérabilité inséré : `/patientaccess/v2/r4`
- Le chemin de la ressource est inchangé : `/Patient/123`
- Les deux appels renvoient des réponses identiques
- L'appel du terminal d'interopérabilité est automatiquement suivi `URIType=patient-access` dans CloudWatch
- Les opérations POST, PUT, PATCH, DELETE fonctionnent de manière identique.
- Le corps de la demande reste inchangé.

Référence de traduction des terminaux

Endpoint d'interopérabilité	Traduit vers	Catégorie CMS
`baseURL/patientaccess/v2/r4/Patient`	`baseURL/r4/Patient`	Accès pour les patients
`baseURL/provideraccess/v2/r4/Observation`	`baseURL/r4/Observation`	Accès du fournisseur
`baseURL/payertopayerdx/v2/r4/Practitioner/456`	`baseURL/r4/Practitioner/456`	Data Exchange de payeur à payeur
`baseURL/priorauthservice/v2/r4/ExplanationOfBenefit?patient=789`	`baseURL/r4/ExplanationOfBenefit?patient=789`	Autorisation préalable

Remarques importantes

- Aucune différence fonctionnelle : les points de terminaison d'interopérabilité et les points de terminaison FHIR standard renvoient des réponses identiques et prennent en charge des opérations identiques
- URL de base inchangée : le point de terminaison de votre banque de HealthLake données reste le même
- Intégration simple : insérez le chemin du point de terminaison d'interopérabilité entre votre URL de base et le type de ressource
- Suivi automatique : CloudWatch les métriques classent automatiquement les appels par point de terminaison d'interopérabilité utilisé
- Rétrocompatible : les appels d'API existants sans points de terminaison d'interopérabilité continuent de fonctionner normalement

CloudWatch Mesures améliorées pour la conformité aux CMS

Présentation de

Lorsque vous utilisez les points de terminaison d'interopérabilité du CMS, émet HealthLake automatiquement des CloudWatch métriques améliorées avec des dimensions supplémentaires pour répondre aux exigences de reporting du CMS. Ces mesures permettent de suivre l'utilisation des API par identité de l'appelant, par application et par type d'URI spécifique au CMS, sans qu'aucune configuration supplémentaire ne soit requise.

Comment ça marche

Lorsque vous effectuez un appel d'API à l'aide d'un point de terminaison d'interopérabilité :

```
# 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.)
```

Aucun code ou configuration supplémentaire n'est nécessaire. Il suffit d'utiliser les points de terminaison d'interopérabilité pour que les métriques améliorées soient automatiquement capturées.

> **Note**
>
> Pour les magasins de données non intelligents sur FHIR, la `URIType` dimension est toujours capturée, ce qui vous permet de suivre l'utilisation des API par catégorie de CMS. Les `ClientId` dimensions `Sub` et ne sont disponibles que lors de l'utilisation de l'authentification SMART sur FHIR avec des jetons au porteur contenant ces allégations.

Nouvelles dimensions métriques

Outre les dimensions existantes (`DatastoreId`,,`Operation`)`DatastoreType`, les dimensions suivantes sont automatiquement ajoutées lors de l'utilisation de points de terminaison d'interopérabilité :

Dimension	Description	Exemple de valeurs	Source
URIType	Catégorie de conformité du CMS	`patient-access` , `provider-access` , `payer-to-payer` , `prior-authorization`	Déterminé automatiquement à partir du chemin du terminal d'interopérabilité
Sous	Identité de l'appelant	Identifiant de l'utilisateur/de l'entité	Extrait de la réclamation du jeton `sub` au porteur
ClientId	Identifiant de l’application	`portal_app` , `ehr_system`	Extrait de la réclamation du jeton `client_id` au porteur

Métriques disponibles

Toutes les HealthLake métriques existantes incluent désormais les dimensions supplémentaires lors de l'utilisation de points de terminaison d'interopérabilité :

- CallCount- Nombre total d'appels d'API

- Latence : temps de réponse de l'API en millisecondes
- UserErrors- Nombre de 4xx erreurs client
- SystemErrors- Nombre de 5xx erreurs de serveur
- Throttles : nombre de demandes limitées
- SuccessfulRequests- Nombre d'appels d'API réussis

Interrogation de métriques dans CloudWatch

CloudWatch Exemple de requête Insights

Interrogez tous les appels de l'API Patient Access par application :

```
SELECT SUM(CallCount)
FROM "AWS/HealthLake"
WHERE DatastoreId = '75c1cf9b0d71cd38fec8f7fb317c4c1a'
    AND URIType = 'patient-access'
GROUP BY ClientId
```

Support de référence pour AWS HealthLake

Les documents de référence complémentaires suivants sont disponibles pour AWS HealthLake.

> **Note**
>
> Toutes les HealthLake actions natives et tous les types de données sont décrits dans une référence séparée. Pour plus d'informations, consultez la Référence des API AWS HealthLake .

Rubriques

- AWS HealthLake points de terminaison et quotas
- Types de données préchargés Synthea pour HealthLake
- AWS HealthLake exemples de projets
- Résolution des problèmes AWS HealthLake
- Utilisation HealthLake avec un AWS SDK

AWS HealthLake points de terminaison et quotas

Les rubriques suivantes contiennent des informations sur les points de terminaison de AWS HealthLake service et les quotas.

Rubriques

Points de terminaison de service

Un point de terminaison de service est une URL qui identifie un hôte et un port comme point d'entrée d'un service Web. Chaque demande de service web contient un point de terminaison. La plupart AWS des services fournissent des points de terminaison pour des régions spécifiques afin de permettre une connectivité plus rapide. Le tableau suivant répertorie les points de terminaison de service pour AWS HealthLake.

Nom de la région	Région	Point de terminaison	Protocole
US East (Ohio)	us-east-2	healthlake.us-east-2.amazonaws.com	HTTPS
		healthlake-fips.us-east-2.amazonaws.com	HTTPS
USA Est (Virginie du Nord)	us-east-1	healthlake.us-east-1.amazonaws.com	HTTPS
		healthlake-fips.us-east-1.amazonaws.com	HTTPS
USA Ouest (Oregon)	us-west-2	healthlake.us-west-2.amazonaws.com	HTTPS
		healthlake-fips.us-west-2.amazonaws.com	HTTPS
Asia Pacific (Mumbai)	ap-south-1	healthlake.ap-south-1.amazonaws.com	HTTPS

Nom de la région	Région	Point de terminaison	Protocole	
Asie-Pacifique (Sydney)	ap-southeast-2	healthlake.ap-southeast-2.amazonaws.com	HTTPS	
Canada (Centre)	ca-central-1	healthlake.ca-central-1.amazonaws.com	HTTPS	
Europe (Irlande)	eu-west-1	healthlake.eu-west-1.amazonaws.com	HTTPS	
Europe (Londres)	eu-west-2	healthlake.eu-west-2.amazonaws.com	HTTPS	

Quotas de service

Les quotas de service sont définis comme la valeur maximale des ressources, des actions et des éléments de votre AWS compte.

> **Note**
>
> Pour les quotas ajustables, vous pouvez demander une augmentation de quota à l'aide de la console Service Quotas. Pour plus d'informations, consultez Demande d'augmentation de quota dans le Guide de l'utilisateur Service Quotas.
> Le débit de l'API Sync Write augmente proportionnellement à la taille de la charge utile, chaque incrément de 1 Ko consommant de la capacité supplémentaire (par exemple, une charge utile de 4 Ko utilise 4 fois la capacité d'écriture). Définissez l'`x-amz-fhir-history-consistency-level`en-tête facultatif pour `strong` doubler la consommation de capacité d'écriture par ressource.
> Les ressources contenues dans les bundles suivent les read/write limites standard basées sur une taille de charge utile de 1 Ko. Les transactions de type bundle consomment deux fois plus de capacité d'écriture que les transactions de type batch, ce qui signifie que les bundles par lots peuvent traiter deux fois plus de ressources par seconde que les bundles de transactions.

Le tableau suivant répertorie les quotas par défaut pour AWS HealthLake.

Nom	Par défaut	Ajusta	Description
Nombre d'abonnements actifs par compte	Chaque région prise en charge : 100	Oui	Le nombre maximum de ressources d'abonnement actives par compte.
Nombre de ressources d'abonnement actives par banque de données	Chaque Région prise en charge : 50	Oui	Le nombre maximum de ressources d'abonnement actives par banque de données.
Nombre d'actifs SubscriptionTopic par compte	Chaque région prise en charge : 100	Oui	Le nombre maximum de SubscriptionTopic ressources actives par compte.
Nombre de SubscriptionTopic ressources actives par banque de données	Chaque Région prise en charge : 50	Oui	Le nombre maximum de SubscriptionTopic ressources actives par banque de données.
Nombre de caractères d'une note médicale	Chaque région prise en charge : 10 000	Non	Le nombre maximum de caractères d'une note médicale individuelle au sein du type de DocumentReference ressource (POST/PUT demandes).
Nombre de StartFHIRExportJob tâches simultanées	Par région prise en charge : 1	Non	Le nombre maximum de StartFHIRExportJob tâches simultanées.
Nombre de StartFHIRImportJob tâches simultanées	Par région prise en charge : 1	Non	Le nombre maximum de StartFHIRImportJob tâches simultanées.

Nom	Par défaut	Ajusta	Description
Nombre de magasins de données par compte	Chaque région prise en charge : 20	Oui	Le nombre maximum par défaut de magasins de données actifs par compte.
Nombre de fichiers dans un StartFHIR ImportJob	Chaque Région prise en charge : 1 000 000	Oui	Le nombre maximum de fichiers dans un fichier StartFHIRImportJob.
Nombre de ressources par bundle	Chaque région prise en charge : 500	Non	Le nombre maximum de ressources autorisées dans une demande de bundle.
Taux de CancelFHIRExportJob demandes utilisant DELETE par compte	Par région prise en charge : 1	Non	Le nombre maximum de CancelFHIRExportJob demandes que vous pouvez effectuer à l'aide de DELETE par seconde et par compte.
Taux de demandes CreateFhirDataStore par compte	Par région prise en charge : 1	Non	Le nombre maximum de demandes CreateFhirDataStore que vous pouvez effectuer par minute et par compte.
Taux de demandes DeleteFhirDataStore par compte	Par région prise en charge : 1	Non	Le nombre maximum de demandes DeleteFhirDataStore que vous pouvez effectuer par minute et par compte.

Nom	Par défaut	Ajusta	Description
Taux de DescribeFHIRBulkDeleteJob demandes par compte	Par région prise en charge : 10	Oui	Le nombre maximum de DescribeFHIRBulkDeleteJob demandes que vous pouvez effectuer par seconde et par compte.
Taux de DescribeFHIRBulkDeleteJob demandes par magasin de données	Par région prise en charge : 10	Oui	Nombre maximal de DescribeFHIRBulkDeleteJob demandes que vous pouvez effectuer par seconde et par magasin de données.
Taux de demandes DescribefHirDataStore par compte	Chaque Région prise en charge : 10	Non	Le nombre maximum de demandes DescribefHirDataStore que vous pouvez effectuer par seconde et par compte.
Taux de DescribeFHIRExportJob demandes par compte	Chaque Région prise en charge : 10	Non	Le nombre maximum de DescribeFHIRExportJob demandes que vous pouvez effectuer par seconde et par compte.
Taux de DescribeFHIRExportJob demandes utilisant GET par compte	Chaque Région prise en charge : 10	Non	Le nombre maximum de DescribeFHIRExportJob requêtes que vous pouvez effectuer à l'aide de GET par seconde et par compte.

Nom	Par défaut	Ajusta	Description
Taux de DescribeFHIRImportJob demandes par compte	Chaque Région prise en charge : 10	Non	Le nombre maximum de DescribeFHIRImport Job demandes que vous pouvez effectuer par seconde et par compte.
Taux de demandes de découverte par compte	Chaque Région prise en charge : 10	Non	Le nombre maximum de demandes Discovery que vous pouvez effectuer par minute et par compte.
Taux de demandes GET par compte	Chaque région prise en charge : 6 000	Oui	Le nombre maximum de requêtes GET que vous pouvez effectuer par seconde et par compte.
Taux de requêtes GET par magasin de données	Chaque région prise en charge : 3 000	Oui	Nombre maximal de requêtes GET que vous pouvez effectuer par seconde et par magasin de données. Les banques de données créées 8/21 avant 2023 seront limitées à 100 demandes par seconde.
Taux de GetCapabilities demandes par compte	Chaque Région prise en charge : 10	Non	Le nombre maximum de GetCapabilities demandes que vous pouvez effectuer par seconde et par compte.

Nom	Par défaut	Ajusta	Description
Taux de GetExportedFile demandes par banque de données	Chaque Région prise en charge : 10	Non	Le nombre maximal de GetExportedFile demandes que vous pouvez effectuer par seconde et par banque de données.
Taux de demandes ListFhirDataStores par compte	Chaque Région prise en charge : 10	Non	Le nombre maximum de demandes ListFhirDataStores que vous pouvez effectuer par seconde et par compte.
Taux de ListFHIRExportJobs demandes par compte	Chaque Région prise en charge : 10	Non	Le nombre maximum de ListFHIRExportJobs demandes que vous pouvez effectuer par seconde et par compte.
Taux de ListFHIRImportJobs demandes par compte	Chaque Région prise en charge : 10	Non	Le nombre maximum de ListFHIRImportJobs demandes que vous pouvez effectuer par seconde et par compte.
Taux de ListTagsforResource demandes par compte	Chaque Région prise en charge : 10	Non	Le nombre maximum de ListTagsforResource demandes que vous pouvez effectuer par seconde et par compte.
Taux de SearchEverything demandes par compte	Par région prise en charge : 10	Oui	Le nombre maximum de SearchEverything demandes que vous pouvez effectuer par seconde et par compte.

Nom	Par défaut	Ajusta	Description
Taux de SearchEverything demandes par magasin de données	Par région prise en charge : 10	Oui	Nombre maximal de SearchEverything demandes que vous pouvez effectuer par seconde et par magasin de données.
Taux de StartFHIRBulkDeleteJob demandes par compte	Par région prise en charge : 10	Oui	Le nombre maximum de StartFHIRBulkDelet eJob demandes que vous pouvez effectuer par seconde et par compte.
Taux de StartFHIRBulkDeleteJob demandes par magasin de données	Par région prise en charge : 10	Oui	Nombre maximal de StartFHIRBulkDeleteJob demandes que vous pouvez effectuer par seconde et par magasin de données.
Taux de StartFHIRBulkMembe rMatchJob demandes par compte	Par région prise en charge : 10	Oui	Le nombre maximum de StartFHIRBulkMembe rMatchJob demandes que vous pouvez effectuer par seconde et par compte.
Taux de StartFHIRBulkMembe rMatchJob demandes par magasin de données	Par région prise en charge : 10	Oui	Nombre maximal de StartFHIRBulkMembe rMatchJob demandes que vous pouvez effectuer par seconde et par magasin de données.

Nom	Par défaut	Ajusta	Description
Taux de StartFHIRExportJob demandes par compte	Par région prise en charge : 1	Non	Le nombre maximum de StartFHIRExportJob demandes que vous pouvez effectuer par seconde et par compte.
Taux de StartFHIRExportJob demandes utilisant GET par compte	Par région prise en charge : 1	Non	Le nombre maximum de StartFHIRExportJob requêtes que vous pouvez effectuer à l'aide de GET par seconde et par compte.
Taux de StartFHIRExportJob demandes utilisant le POST par compte	Par région prise en charge : 1	Non	Le nombre maximum de StartFHIRExportJob requêtes que vous pouvez effectuer à l'aide de POST par seconde et par compte.
Taux de StartFHIRImportJob demandes par compte	Chaque région prise en charge : 25	Non	Le nombre maximum de StartFHIRImportJob demandes que vous pouvez effectuer par seconde et par compte.
Taux de TagResource demandes par compte	Chaque Région prise en charge : 10	Non	Le nombre maximum de TagResource demandes que vous pouvez effectuer par seconde et par compte.

Nom	Par défaut	Ajusta	Description
Taux de UntagResource demandes par compte	Chaque Région prise en charge : 10	Non	Le nombre maximum de UntagResource demandes que vous pouvez effectuer par seconde et par compte.
Taux de ValidateResource demandes par compte	Chaque Région prise en charge : 2 000	Oui	Le nombre maximum de ValidateResource demandes que vous pouvez effectuer par seconde et par compte. Les banques de données créées 8/21 avant 2023 seront limitées à 300 demandes par seconde.
Taux de ValidateResource demandes par magasin de données	Chaque Région prise en charge : 1 000	Oui	Nombre maximal de ValidateResource demandes que vous pouvez effectuer par seconde et par magasin de données. Les banques de données créées 8/21 avant 2023 seront limitées à 300 demandes par seconde.
Taux de demandes WRITE par compte	Chaque région prise en charge : 6 000	Oui	Le nombre maximum de demandes CREATE|UPDATE|DELETE que vous pouvez effectuer par seconde et par compte.

Nom	Par défaut	Ajusta	Description
Taux de demandes WRITE par magasin de données	Chaque région prise en charge : 3 000	Oui	Nombre maximal de demandes CREATE|UPDATE|DELETE que vous pouvez effectuer par seconde et par magasin de données. Les banques de données créées 8/21 avant 2023 seront limitées à 300 demandes par seconde.
Taux de demandes de recherche utilisant GET par compte	Chaque région prise en charge : 200	Oui	Le nombre maximum de demandes de recherche que vous pouvez effectuer à l'aide de GET par seconde et par compte.
Taux de demandes de recherche utilisant GET par magasin de données	Chaque région prise en charge : 100	Oui	Nombre maximal de demandes de recherche que vous pouvez effectuer à l'aide de GET par seconde et par magasin de données.
Taux de demandes de recherche utilisant POST par compte	Chaque région prise en charge : 200	Oui	Le nombre maximum de demandes de recherche que vous pouvez effectuer à l'aide de POST par seconde et par compte.

Nom	Par défaut	Ajusta	Description
Taux de demandes de recherche utilisant POST par magasin de données	Chaque région prise en charge : 100	Oui	Le nombre maximal de demandes de recherche que vous pouvez effectuer à l'aide de POST par seconde et par magasin de données.
Taille de chaque fichier importé	Chaque région prise en charge : 50 gigaoctets	Non	La taille maximale (en Go) d'un fichier individuel inclus dans un fichier StartFHIRImportJob.
Nombre total de transactions groupées asynchrones mises en file d'attente par banque de données	Chaque région prise en charge : 500	Oui	Nombre maximal de transactions groupées asynchrones en file d'attente par banque de données à un moment donné.
Nombre total de tâches d'exportation en bloc en file d'attente par banque de données	Chaque région prise en charge : 25	Oui	Nombre maximal de tâches d'exportation en bloc mises en file d'attente par banque de données à un moment donné.
Nombre total de tâches d'importation en bloc en file d'attente par banque de données	Chaque région prise en charge : 25	Oui	Nombre maximal de tâches d'importation en bloc mises en file d'attente par banque de données à un moment donné.

Nom	Par défaut	Ajusta	Description
Taille totale de la tâche d'importation	Chaque région prise en charge : 5 000 gigaoctets	Oui	Taille maximale (en Go) de tous les fichiers inclus dans la tâche d'importation.

Types de données préchargés Synthea pour HealthLake

HealthLake ne prend en charge que SYNTHEA en tant que type de données préchargé. Synthea est un générateur synthétique de patients qui modélise les antécédents `Patient` médicaux. Il est hébergé dans un référentiel Git open source qui permet de générer une ressource conforme HealthLake à la norme FHIR R4 `Bundle` afin que les utilisateurs puissent tester des modèles sans utiliser les données réelles des patients.

Les types de ressources suivants sont disponibles dans les magasins de HealthLake données préchargés. Pour plus d'informations sur le préchargement HealthLake des banques de données avec les données Synthea, consultez. Création d'un magasin HealthLake de données

> **Note**
>
> Pour consulter la liste complète des ressources FHIR R4 HealthLake prises en charge, voir. Types de ressources pris en charge par FHIR R4 pour HealthLake

Types de ressources Synthea FHIR pris en charge par HealthLake

AllergyIntolerance	Location
CarePlan	MedicationAdministration
CareTeam	MedicationRequest
Demander	Observation
Condition	Organisation
Appareil	Patient

DiagnosticReport	Praticien
Rencontre	PractitionerRole
ExplanationofBenefit	Procédure
ImagingStudy	Provenance
Immunisation	

AWS HealthLake exemples de projets

Pour approfondir votre analyse, vous pouvez utiliser HealthLake d'autres AWS services, comme le montrent les exemples d'articles de blog suivants.

HealthLake analyse intégrée

- Applications de santé de la population avec AWS HealthLake — Partie 1 : Analyse et surveillance à l'aide d'Amazon Quick.
- Création de modèles prédictifs de maladies à l'aide d'Amazon SageMaker AI avec des données AWS HealthLake normalisées.
- Créez une recherche cognitive et un graphe de connaissances sur la santé à l'aide de services d' AWS IA.

HealthLake surveillance des événements

- EventBridge Intégration d'Amazon avec AWS HealthLake.

Résolution des problèmes AWS HealthLake

Les rubriques suivantes fournissent des conseils de résolution des erreurs et des problèmes que vous pourriez rencontrer lors de l'utilisation de la HealthLake console AWS CLI AWS SDKs, ou. Si vous trouvez un problème qui n'est pas répertorié dans cette section, utilisez le bouton Envoyer des commentaires dans la barre latérale droite de cette page pour le signaler.

Rubriques

- Actions relatives au stockage des données

- Actions d'importation
- FHIR APIs
- Intégrations NLP
- Intégrations SQL

Actions relatives au stockage des données

Problème : Lorsque j'essaie de créer un magasin de HealthLake données, le message d'erreur suivant s'affiche :

```
AccessDeniedException: Insufficient Lake Formation permission(s): Required Database on
 Catalog
```

Le 14 novembre 2022, les autorisations IAM requises ont été HealthLake mises à jour pour créer un nouveau magasin de données. Pour de plus amples informations, veuillez consulter Configuration d'un utilisateur ou d'un rôle IAM à utiliser HealthLake (administrateur IAM).

Problème : lors de la création d'un magasin de HealthLake données à l'aide du AWS SDKs, le statut de création du magasin de données renvoie une exception ou un statut inconnu.

Mettez à jour votre AWS SDK vers la dernière version si vos appels `DescribeFHIRDatastore` ou ceux de `ListFHIRDatastores` l'API renvoient une exception ou un statut de banque de données inconnu.

Actions d'importation

Problème : Puis-je toujours l'utiliser HealthLake si mes données ne sont pas au format FHIR R4 ?

Seules les données au format FHIR R4 peuvent être importées dans un HealthLake magasin de données. Pour une liste des partenaires qui peuvent aider à transformer les données de santé existantes au format FHIR R4, voir AWS HealthLake Partenaires.

Problème : Pourquoi ma tâche d'importation FHIR a-t-elle échoué ?

Une tâche d'importation réussie générera un dossier avec les résultats (journal de sortie) au `.ndjson` format, mais l'importation d'enregistrements individuels peut échouer. Dans ce cas, un deuxième `FAILURE` dossier sera généré avec un manifeste des enregistrements dont l'importation n'a pas pu être effectuée. Pour de plus amples informations, veuillez consulter Importation de données FHIR avec AWS HealthLake.

Pour analyser pourquoi une tâche d'importation a échoué, utilisez l'`DescribeFHIRImportJob`API pour analyser le JobProperties. Il est recommandé de procéder comme suit :

- Si le statut est `FAILED` et qu'un message est présent, les échecs sont liés à des paramètres de tâche tels que la taille des données d'entrée ou le nombre de fichiers d'entrée dépassant les HealthLake quotas.
- Si le statut de la tâche d'importation est le suivant `COMPLETED_WITH_ERRORS`manifest.json, consultez le fichier manifeste pour savoir quels fichiers n'ont pas été importés correctement.
- Si le statut de la tâche d'importation est le même `FAILED` et qu'aucun message n'est présent, rendez-vous à l'emplacement de sortie de la tâche pour accéder au fichier manifeste,`manifest.json`.

Pour chaque fichier d'entrée, il existe un fichier de sortie d'échec avec le nom du fichier d'entrée pour toute ressource dont l'importation échoue. Les réponses contiennent le numéro de ligne (LineID) correspondant à l'emplacement des données d'entrée, l'objet de réponse FHIR (UpdateResourceResponse) et le code d'état (StatusCode) de la réponse.

Un exemple de fichier de sortie peut être similaire au suivant :


```
{"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}
```

L'exemple ci-dessus montre qu'il y a eu des défaillances sur les lignes 3, 4, 7, 9, 15 à partir des lignes d'entrée correspondantes du fichier d'entrée. Pour chacune de ces lignes, les explications sont les suivantes :

- Sur la ligne 3, la réponse explique que le contenu `resourceType` fourni dans la ligne 3 du fichier d'entrée n'est pas valide.
- Sur la ligne 5, la réponse explique qu'il y a une erreur de validation FHIR dans la ligne 5 du fichier d'entrée.
- À la ligne 7, la réponse explique qu'il existe un problème de validation avec la `resourceId` valeur fournie en entrée.
- Sur la ligne 9, la réponse explique que le fichier d'entrée doit contenir un identifiant de ressource valide.
- À la ligne 15, la réponse du fichier d'entrée est que le fichier n'est pas dans un format JSON valide.

FHIR APIs

Problème : Comment mettre en œuvre l'autorisation pour le FHIR RESTful APIs ?

Déterminez le Stratégie d'autorisation du magasin de données à utiliser.

Pour créer une autorisation SigV4 à l'aide de AWS SDK pour Python (Boto3), créez un script similaire à l'exemple suivant.

```
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())
```

Problème : Pourquoi est-ce que je reçois *`AccessDenied`* des erreurs lorsque j'utilise le FHIR RESTful APIs pour un magasin de données chiffré avec une clé KMS gérée par le client ?

Les autorisations relatives aux clés gérées par le client et aux politiques IAM sont requises pour qu'un utilisateur ou un rôle puisse accéder à un magasin de données. Un utilisateur doit disposer des autorisations IAM requises pour utiliser une clé gérée par le client. Si un utilisateur révoque ou retire une HealthLake autorisation autorisant l'utilisation de la clé KMS gérée par le client, un `AccessDenied` message d'erreur HealthLake sera renvoyé.

HealthLake doit avoir l'autorisation d'accéder aux données des clients, de chiffrer les nouvelles ressources FHIR importées dans un magasin de données et de déchiffrer les ressources FHIR lorsqu'elles sont demandées. Pour plus d'informations, consultez la section Résolution des problèmes liés AWS KMS aux autorisations.

Problème : Une opération de *`POST`* l'API FHIR HealthLake utilisant un document de 10 Mo renvoie l'*`413 Request Entity Too Large`*erreur.

AWS HealthLake dispose d'une limite d'API de création et de mise à jour synchrone de 5 Mo afin d'éviter des latences et des délais d'attente accrus. Vous pouvez ingérer des documents volumineux, jusqu'à 164 Mo, en utilisant le type de `Binary` ressource à l'aide de l'API d'importation en bloc.

Intégrations NLP

Problème : Comment activer la fonction intégrée HealthLake de traitement du langage naturel ?

Le 14 novembre 2022, le comportement par défaut des magasins de HealthLake données a changé.

Magasins de données actuels : tous les magasins de HealthLake données actuels cesseront d'utiliser le traitement du langage naturel (NLP) sur les ressources codées en base64`DocumentReference`. Cela signifie que les nouvelles `DocumentReference` ressources ne seront pas analysées à l'aide du NLP et qu'aucune nouvelle ressource ne sera générée à partir du texte du type de

`DocumentReference` ressource. Pour les `DocumentReference` ressources existantes, les données et les ressources générées via le NLP sont conservées, mais elles ne seront pas mises à jour après le 20 février 2023.

Nouveaux magasins de données : les magasins de HealthLake données créés après le 20 février 2023 n'effectueront pas de traitement du langage naturel (NLP) sur les ressources codées en base64`DocumentReference`.

Pour activer l'intégration du HealthLake NLP, créez un dossier d'assistance à l'aide AWS Support Center Consolede. Pour créer votre dossier, connectez-vous à votre dossier Compte AWS, puis choisissez Créer un dossier. Pour en savoir plus sur la création d'un dossier et sur la gestion des dossiers, voir Création de dossiers d'assistance et gestion de cas dans le Guide de Support l'utilisateur.

Problème : >Comment trouver les *`DocumentReference`* ressources qui n'ont pas pu être traitées par le NLP intégré ?

Si une `DocumentReference` ressource n'est pas valide, HealthLake fournit une extension indiquant une erreur de validation au lieu de la fournir dans la sortie PNL médicale intégrée. Pour rechercher les **`DocumentReference`** ressources qui ont entraîné une erreur de validation lors du traitement NLP, vous pouvez utiliser la **`search`** fonction FHIR avec la clé HealthLake de recherche cm-decoration-statuset la valeur de recherche VALIDATION_ERROR. Cette recherche répertorie toutes les `DocumentReference` ressources ayant entraîné des erreurs de validation, ainsi qu'un message d'erreur décrivant la nature de l'erreur. La structure du champ d'extension dans les `DocumentReference` ressources présentant des erreurs de validation ressemblera à l'exemple suivant.

```
"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**
>
> Un `VALIDATION_ERROR` peut également se produire si la décoration NLP crée plus de 10 000 objets imbriqués. Dans ce cas, le document doit être scindé en documents plus petits avant d'être traité.

Intégrations SQL

Problème : Pourquoi est-ce que j'obtiens un Lake Formation *`permissions error: lakeformation:PutDataLakeSettings`* lorsque j'ajoute un nouvel administrateur de lac de données ?

Si votre utilisateur ou rôle IAM contient la politique `AWSLakeFormationDataAdmin` AWS gérée, vous ne pouvez pas ajouter de nouveaux administrateurs de data lake. Vous recevrez un message d'erreur contenant les informations suivantes :

```
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
```

La politique AWS gérée `AdministratorAccess` est requise pour ajouter un utilisateur ou un rôle IAM en tant qu'administrateur AWS du lac de données de Lake Formation. Si votre utilisateur ou rôle IAM en contient également, `AWSLakeFormationDataAdmin` l'action échouera. La politique `AWSLakeFormationDataAdmin` AWS gérée contient un refus explicite de l'opération de l'API AWS Lake Formation,`PutDataLakeSetting`. Même les administrateurs disposant d'un accès complet à l' AWS utilisation de la politique `AdministratorAccess` gérée peuvent être limités par cette `AWSLakeFormationDataAdmin` dernière.

Problème : Comment migrer un magasin de HealthLake données existant pour utiliser l'intégration SQL d'Amazon Athena ?

HealthLake les magasins de données créés avant le 14 novembre 2022 sont fonctionnels, mais ne peuvent pas être interrogés dans Athena à l'aide de SQL. Pour interroger un magasin de données préexistant avec Athena, vous devez d'abord le migrer vers un nouveau magasin de données.

Pour migrer vos HealthLake données vers un nouveau magasin de données

1. Créez un nouveau magasin de données.
2. Exportez les données du compartiment préexistant vers un compartiment Amazon S3.
3. Importez les données dans le nouveau magasin de données depuis le compartiment Amazon S3.

> **Note**
>
> L'exportation de données vers un compartiment Amazon S3 entraîne des frais supplémentaires. Les frais supplémentaires dépendent de la taille des données que vous exportez.

Problème : lors de la création d'un nouveau magasin de HealthLake données pour l'intégration SQL, le statut du magasin de données ne change pas de*`Creating`*.

Si vous essayez de créer un nouveau magasin de HealthLake données et que le statut de votre magasin de données ne change pas depuis Création, vous devez mettre Athéna à jour pour qu'elle utilise le. AWS Glue Data Catalog Pour plus d'informations, consultez la section Mise à niveau vers le catalogue de données AWS Glue step-by-step dans le guide de l'utilisateur Amazon Athena.

Une fois la mise à niveau réussie AWS Glue Data Catalog, vous pouvez créer un magasin de HealthLake données.

Pour supprimer un ancien magasin de HealthLake données, créez un dossier d'assistance à l'aide de AWS Support Center Console. Pour créer votre dossier, connectez-vous à votre dossier Compte AWS, puis choisissez Créer un dossier. Pour en savoir plus, consultez les sections Création de demandes d'assistance et gestion de dossiers dans le Guide de Support l'utilisateur.

Problème : La console Athena ne fonctionne pas après l'importation de données dans un nouveau HealthLake magasin de données

Une fois que vous avez importé des données dans un nouveau magasin de HealthLake données, il est possible que celles-ci ne soient pas disponibles pour une utilisation immédiate. Cela permet de laisser le temps aux données d'être ingérées dans les tables Apache Iceberg. Réessayez ultérieurement.

Problème : Comment associer les résultats de recherche d'Athena à d'autres AWS services ?

Lorsque vous partagez les résultats de recherche d'Athena avec d'autres AWS services, des problèmes peuvent survenir lorsque vous les utilisez dans `json_extract[1]` le cadre d'une requête de recherche SQL. Pour résoudre ce problème, vous devez effectuer une mise à jour vers`CATVAR`.

Vous pouvez rencontrer ce problème lorsque vous essayez de créer des résultats de sauvegarde, une table (statique) ou une vue (dynamique).

Utilisation HealthLake avec un AWS SDK

AWS des kits de développement logiciel (SDKs) sont disponibles pour de nombreux langages de programmation courants. Chaque kit SDK fournit une API, des exemples de code et de la documentation qui facilitent la création d'applications par les développeurs dans leur langage préféré.

Documentation SDK	Exemples de code
AWS SDK pour C++	AWS SDK pour C++ exemples de code
AWS CLI	AWS CLI exemples de code
AWS SDK pour Go	AWS SDK pour Go exemples de code
AWS SDK pour Java	AWS SDK pour Java exemples de code
AWS SDK pour JavaScript	AWS SDK pour JavaScript exemples de code
AWS SDK pour Kotlin	AWS SDK pour Kotlin exemples de code
AWS SDK pour .NET	AWS SDK pour .NET exemples de code
AWS SDK pour PHP	AWS SDK pour PHP exemples de code
Outils AWS pour PowerShell	Outils AWS pour PowerShell exemples de code
AWS SDK pour Python (Boto3)	AWS SDK pour Python (Boto3) exemples de code
AWS SDK pour Ruby	AWS SDK pour Ruby exemples de code
AWS SDK pour Rust	AWS SDK pour Rust exemples de code

Documentation SDK	Exemples de code
AWS SDK pour SAP ABAP	AWS SDK pour SAP ABAP exemples de code
AWS SDK pour Swift	AWS SDK pour Swift exemples de code

Exemple de disponibilité

Vous n'avez pas trouvé ce dont vous avez besoin ? Demandez un exemple de code en utilisant le lien Provide feedback (Fournir un commentaire) en bas de cette page.

AWS HealthLake versions

Le tableau suivant indique la date de publication des fonctionnalités et des mises à jour pour AWS HealthLake. Pour plus d'informations sur une version, consultez la rubrique associée.

Modification	Description	Date
Sécurité et filtrage des balises pour $export et $ davinci-data-export	Les `$davinci-data-export` opérations `$export` et prennent désormais en charge `_security` et `_tag` interrogent les paramètres permettant de filtrer les ressources exportées par valeurs `meta.security` et de `meta.tag` codage. Cela permet l'exportation multi-locataires et le contrôle d'accès granulaire à l'aide du format `system|code` FHIR standard.	30 avril 2026
bulk-member-match opération $	AWS HealthLake prend désormais en charge l'`$bulk-member-match` opération de traitement asynchrone des demandes de correspondance de plusieurs membres. Cette opération permet aux établissements de santé de faire correspondre efficacement les identifiants uniques de centaines de membres dans différents systèmes de santé en utilisant des informations démographiques et de	1 avril 2026

couverture dans une seule demande groupée.

- Gère jusqu'à 500 membres par demande avec jusqu'à 5 opérations simultanées par magasin de données
- Résultats classés en MatchedMembers NonMatchedMembers, et ConsentConstrained Members groupes
- S'intègre aux flux `$davinci-data-export` de travail de données end-to-end en masse

Pour de plus amples informations, veuillez consulter the section called “$bulk-member-match”.

Transactions groupées asynchrones	AWS HealthLake supporte désormais le `Bundle` type asynchrone`transacti on` , ce qui vous permet de soumettre des transactions avec un maximum de 500 ressources. HealthLake met la transaction en file d'attente pour traitement et renvoie immédiatement une URL d'interrogation pour vérifier le statut et récupérer les résultats . Pour plus d'informations, consultez la section Transacti ons groupées asynchrones.	24 mars 2026
Support des correctifs dans le cadre des opérations groupées	HealthLake supporte désormais Bundle Patch, activant à la fois FHIRPath le patch JSON pour les bundles Transaction et Batch, ainsi que FHIRPath pour l'API Patch. Cette fonctionnalité permet aux clients de patcher les ressources directement dans le cadre des opérations du bundle sans avoir à remplacer complètement les ressource s, en réduisant la taille de la charge utile, en simplifiant les flux de travail d'intégration et en accélérant l'ingestion des données.	20 mars 2026

DaVinci PDex Types d'exportation pour $ davinci-data-export	L'`$davinci-data-export` opération prend désormais en charge les types PDex d'exportation pour l'accès fournisseur et l'accès membre APIs. Payer-to-Payer • Logique d'inclusion basée sur le profil pour les ressources ExplanationOfBenefit • Transformation des données financières avec le `_includeEOB2xWoFinancial` paramètre • Filtre temporel sur 5 ans pour les données cliniques et les données relatives aux demandes	20 mars 2026
Paramètre _until dans $export & $ davinci-data-export	Paramètre de filtrage temporel pour les opérations d'exportation	26 février 2026
_inclure le paramètre de recherche	HealthLake supporte désormais include : * et include:iterate	26 février 2026
Points de terminaison d'interopérabilité CMS	Cette fonctionnalité vous permet de suivre l'utilisation des API par catégorie de CMS et de signaler ensuite les mesures d'utilisation à des fins de conformité.	26 février 2026

Support par type de message groupé	Support limité pour les ressources du bundle FHIR avec type de message	26 février 2026
Ajout du support pour les nouveaux IGs	AWS HealthLake a étendu la prise en charge de ses guides de mise en œuvre FHIR (IGs) pour le CMS 0057F : • Support pour CARIN Blue Button 2.0.0 et 2.1.0 • Support pour Da Vinci Payer Data Exchange 2.0.0 et 2.1.0 • Support pour DaVinci Payer Data Exchange (PDex) US Drug Formulary 2.0.1 et 2.1.0 • Support pour Da Vinci Clinical Data Exchange (CDex) 2.1.0 • Support pour le support d'autorisation préalable (PAS) Da Vinci FHIR IG 2.1.0	26 février 2026
Opération $submit	L'`$submit`opération vous permet de soumettre électroni quement des demandes d'autorisation préalable aux payeurs pour approbation.	26 février 2026

Opération $questionnaire-package	L'`$questionnaire-package` opération récupère un ensemble complet contenant un questionnaire FHIR et toutes ses dépendances nécessaires au rendu et au traitement du questionnaire.	26 février 2026
Opération $inquire	L'`$inquire`opération vous permet de vérifier le statut d'une demande d'autorisation préalable soumise précédemment.	26 février 2026
Opération $member-remove	L'opération $member-remove vous permet de supprimer des membres d'une liste d'attribution de membres FHIR (ressource de groupe) dans. AWS HealthLake	12 novembre 2025
Opération $member-match	AWS HealthLake prend désormais en charge l'opération $member-match pour les ressources destinées aux patients, permettant aux établissements de santé de trouver l'identifiant unique d'un membre dans différents systèmes de santé à l'aide d'informations démographiques et de couverture.	12 novembre 2025

Opération $member-add	L'opération FHIR $member-add ajoute un membre (patient) à une ressource de groupe, en particulier à une liste d'attribution de membres.	12 novembre 2025
davinci-data-export Opération $	L'davinci-data-export opération $ est une opération FHIR asynchrone qui permet d'exporter les données de la liste d'attribution des membres depuis. AWS HealthLake	12 novembre 2025
confirm-attribution-list Opération $	Indique au producteur que le consommateur n'a plus aucune modification à apporter à la liste d'attribution, en finalisant la liste d'attribution en supprimant les membres inactifs et en modifiant le statut en « final ».	12 novembre 2025
Opération $attribution-status	Récupère le statut d'attribution d'un membre spécifique, renvoyant un bundle contenant toutes les ressources d'attribution liées au patient.	12 novembre 2025
Abonnements FHIR	HealthLake prend en charge les abonnements FHIR, vous permettant de recevoir des notifications en temps réel lorsque des modifications spécifiques des données de santé se produisent et de créer des flux de travail axés sur les événements.	30 octobre 2025

Expansion de la région à Montréal, Canada	HealthLake est disponible dans la région Canada (Montréal). Pour plus d'informations, consultez la section Points de terminaison du service.	17 octobre 2025

Ajout du support pour les nouveaux IGs	AWS HealthLake a étendu son support aux guides de mise en œuvre du FHIR (IGs) pour le marché canadien suivant : • CA Core+profil FHIR de référence canadien définissant les éléments de données de base et les contraintes d'interopérabilité entre les systèmes de santé canadiens. • CA:EREC Pan-Canadian eReferral- Spécification eConsultStandardized FHIR pour les recommandations électroniques et les consultations entre les fournisseurs de soins de santé à travers le Canada. • Résumé du patient Édition canadienne (PS-CA) Adaptation canadienne de l'International Patient Summary (IPS) pour partager les informations essentielles sur la santé des patients dans tous les établissements de soins. • Profil FHIR RepositoryOntario spécifique à Ontario Digital Health Drug pour l'échange de données normalisées sur les médicaments et les	17 octobre 2025

	ordonnances au sein du système de santé provincial.	
Support informatique spécifique à la région	HealthLake prend désormais en charge les spécificités régionales IGs. Pour plus d'informations, consultez la section Validations de profil.	8 octobre 2025
Fonctionnement du correctif	HealthLake permet de modifier des éléments spécifiques des ressources FHIR à l'aide des opérations de patch JSON sans mettre à jour l'intégralité de la ressource.	18 août 2025
Opération $validate	HealthLake permet de valider les ressources FHIR par rapport aux spécifications et aux profils sans effectuer d'opérations de stockage, renvoyant des résultats de validation détaillés.	18 août 2025
Opération $purge	HealthLake inclut une fonctionnalité permettant de supprimer définitivement de la banque de données toutes les ressources du compartiment d'un patient.	18 août 2025
Opération $lookup	HealthLake permet de récupérer des informations détaillées sur un concept spécifique dans un en CodeSystem fournissant le code et l'identifiant du système.	18 août 2025

Opération $expand	HealthLake permet désormais d'étendre ValueSet les ressources pour récupérer la liste complète des codes contenus dans les fichiers ingérés par le client ValueSets .	18 août 2025
Opération $erase	HealthLake permet désormais de supprimer définitivement une ressource spécifique et toutes ses versions historiques de la banque de données.	18 août 2025
Opération $document	HealthLake permet de générer des documents cliniques complets en regroupant une ressource de composition avec toutes ses ressource s référencées dans un seul ensemble de documents.	18 août 2025
:en dessous du modificateur de recherche	HealthLake introduit la recherche de valeurs d'URI hiérarchiquement inférieures à l'URI spécifié dans un système terminologique.	8 août 2025

Suppression conditionnelle	HealthLake prend désormais en charge la suppressi on conditionnelle FHIR, permettant aux établisse ments de santé de supprimer une ressource existante en fonction de critères de recherche plutôt que d'un identifiant FHIR logique. Voir Supprimer des ressources FHIR en fonction des condition s pour plus d'informations.	7 juillet 2025
Ajout du support pour les nouveaux IGs	AWS HealthLake a étendu la prise en charge de ses guides de mise en œuvre du FHIR (IGs) aux points suivants : • US Core 7.0.0 qui spécifie comment utiliser FHIR pour implémenter la norme USCDI 4.0 • Guide de mise en œuvre de UK Core 2.0.1 pour fournir des conseils de mise en œuvre du FHIR à l'échelle du Royaume-Uni	7 juillet 2025
Expansion de la région à Dublin, en Irlande	HealthLake est disponible dans la région UE (Dublin). Pour plus d'informations, consultez la section Points de terminaison du service.	9 juin 2025

Transactions de type groupé	HealthLake prend désormais en charge le type de bundle FHIR « Transaction », permettant aux établissements de santé de soumettre plusieurs ressources en une seule opération atomique. Cela permet un échange de données et des flux de travail d'intégration plus efficaces. Par exemple, un professionnel de santé peut désormais mettre à jour un dossier patient, une liste de médicaments et un rendez-vous en une seule transaction, ce qui réduit la complexité et les erreurs potentielles. Voir Regrouper les ressources FHIR pour plus d'informations.	28 avril 2025

Ajout du support pour les nouveaux IGs	AWS HealthLake AWS HealthLake a étendu la prise en charge de ses guides de mise en œuvre FHIR (IGs) aux éléments suivants : • Guide de mise en œuvre du NCQA HEDIS® (0.3.1) : prend en charge la mesure de la qualité et les rapports pour l'ensemble de données et d'informations sur l'efficacité des soins de santé (HEDIS). • Résumé international des patients (IPS) (2.0.0) : permet l'échange d'informations de santé essentielles pour assurer la continuité des soins aux patients. • Mesure de la qualité (5.0.0) : prend en charge la représentation et l'échange de définitions et de données de mesures de qualité. • Rapports génomiques (3.0.0) : Facilite l'échange de données et de rapports génomiques.	28 avril 2025

Clés d'impuissance	HealthLake prend désormais en charge les clés d'idempui ssance pour les opérations FHIR POST, fournissant ainsi un mécanisme robuste pour garantir l'intégrité des données lors de la création des ressources. Voir Idempoten cie et simultanéité pour plus d'informations.	18 avril 2025
Cohérence de l'historique FHIR	HealthLake prend désormais en charge une forte cohérence pour les magasins de données compatibles avec l'historique via le nouvel `x-amz-fhir-history-consistency-level` en-tête. Lorsqu'ils sont définis sur « fort », les résultats de recherche FHIR incluent tous les enregistr ements indexés, quel que soit le statut de mise à jour. Voir Niveaux de cohérence des recherches FHIR pour plus d'informations.	18 avril 2025

Etag et « if-match »	HealthLake fournit désormais le support eTag, permettant aux clients d'utiliser l'en-tête « If-Match » pour garantir des mises à jour idempotentes. Cela permet de préserver l'intégrité des données en empêchant les remplacements accidentels lors de mises à jour simultanées. Cela est particulièrement utile dans les environnements de soins de santé à volume élevé où plusieurs systèmes peuvent tenter de mettre à jour le même dossier simultanément. Consultez ETag AWS HealthLake pour plus d'informations.	18 avril 2025

Conditionnel PUTs dans les lots	HealthLake prend désormais en charge les mises à jour conditionnelles pour les offres groupées FHIR, offrant ainsi aux établissements de santé une plus grande flexibilité pour gérer et mettre à jour leurs données. Les clients peuvent désormais spécifier des critères pour créer, mettre à jour ou supprimer des ressources de manière conditionnelle dans le cadre d'une transaction Bundle. Cela simplifie les processus de synchronisation des données entre les systèmes et réduit le besoin d'une logique complexe côté client. Voir Conditionnel PUTs dans les offres groupées pour plus d'informations.	18 avril 2025

SMART sur les oscilloscopes FHIR V2	HealthLake prend en charge les scopes SMART sur FHIR V2 pour créer, lire, mettre à jour, supprimer et recherche r des ressources FHIR. Pour plus d'informations, consultez SMART sur les étendues de ressources FHIR pour. HealthLake • Les oscilloscopes SMART on FHIR V2 sont disponibl es pour tous les magasins de HealthLake données créés après le 22/01/2025. Si votre banque de données a été créée avant cette date, vous pouvez envoyer un ticket d'assistance pour activer les oscilloscopes SMART sur FHIR V2. Créez un dossier en utilisant AWS Support Center Console. Pour créer votre dossier, connectez-vous à votre dossier Compte AWS et choisissez Créer un dossier.	22 janvier 2025
Profil de base américain FHIR, version 6.1.0	HealthLake supporte la version 6.1.0 du FHIR US Core Profile. Pour plus d'informations, consultez la section Validations des profils FHIR pour. HealthLake	22 janvier 2025

FHIR $export avec GET	HealthLake prend en charge le FHIR `$export` avec`GET`. Pour plus d'informations, consultez Exporter HealthLake des données avec FHIR `$export`.	22 janvier 2025
Guide du développeur refactorisé avec des exemples de code testés	HealthLake présente un guide du développeur remanié avec des exemples de code testés pour les actions natives AWS CLI et AWS celles du SDK. En outre, des procédures sont désormais disponibles pour toutes les interactions d'API FHIR prises en charge. Pour plus d'informations, consultez Exemples de code et Gestion des ressources FHIR.	18 décembre 2024

FHIR `history` et interactions `vread`	HealthLake prend en charge l'`history`interaction FHIR pour récupérer l'historique d'une ressource particulière et l'`vread`interaction pour effectuer une lecture d'une ressource spécifique à la version. Pour plus d'informations, voir Lire l'historique des ressources FHIR. • La ressource FHIR `history` est activée par défaut pour tous les magasins de HealthLake données créés après le 25/10/2024. Si votre banque de données a été créée avant cette date, vous pouvez envoyer un ticket d'assistance pour activer `history` l'interaction FHIR. Créez un dossier en utilisant AWS Support Center Console. Pour créer votre dossier, connectez-vous à votre dossier Compte AWS et choisissez Créer un dossier.	25 octobre 2024

Opération FHIR `Patient/$everything`	HealthLake soutient l'`Patient/$everything` opération FHIR pour rechercher une `Patient` ressource et toutes les ressources associées. Grâce à cette opération, vous pouvez accéder à l'intégralité du dossier d'un patient ou télécharger `Patient` des données en masse. Pour plus d'informations, consultez la section Obtenir les données des patients avec `Patient/$everything`. • Le FHIR `Patient/$everything` est activé par défaut pour tous les magasins de HealthLake données créés après le 27/02/2024. Si votre banque de données a été créée avant cette date, vous pouvez envoyer un ticket d'assistance pour activer l'`Patient/$everything` opération. Créez un dossier en utilisant AWS Support Center Console. Pour créer votre dossier, connectez-vous à votre dossier Compte AWS et choisissez Créer un dossier.	27 février 2024

Ressource FHIR `VerificationResult`	HealthLake prend en charge le type de `VerificationResult` ressource FHIR pour décrire les exigences de validation, les sources, le statut et les dates d'un ou de plusieurs éléments. Pour plus d'informations, consultez les types de ressources FHIR R4 pour. HealthLake	9 décembre 2023

Opération FHIR `$export`	HealthLake prend en charge l'`$export`opération FHIR pour exporter des données de santé en masse à partir d'un magasin de HealthLake données. Pour plus d'informations, consultez Exporter HealthLake des données avec FHIR `$export`. • Le FHIR `$export` est activé par défaut pour tous les magasins de HealthLake données créés après le 1er juin 2023. Si votre banque de données a été créée avant cette date, vous pouvez envoyer un ticket d'assistance pour activer l'`$export`opération. Créez un dossier en utilisant AWS Support Center Console. Pour créer votre dossier, connectez-vous à votre dossier Compte AWS et choisissez Créer un dossier. • HealthLake les banques de données créées avant le 23/01/06 ne prennent en charge que les demandes de `$export` travail pour les exportations à l'échelle du système. • HealthLake les banques de données créées avant le 23/01/06 ne permettent pas	1er juin 2023

	d'obtenir le statut d'un FHIR à `$export` l'aide d'une `GET` requête sur le terminal de la banque de données.	
SMART sur support FHIR	HealthLake ajoute le support pour l'autorisation SMART on FHIR. Pour plus d'informations, consultez SMART sur le support FHIR pour AWS HealthLake.	31 mai 2023
Validations des profils FHIR	HealthLake prend en charge les validations de profils FHIR pour définir des définitions de types de ressources spécifiques à l'aide d' and/or extensions de contraintes sur les types de ressources de base. Pour plus d'informations, consultez la section Validations de profils.	31 mai 2023
Région Asie-Pacifique (Mumbai)	HealthLake est disponible dans la région Asie-Pacifique (Mumbai). Pour plus d'informations, consultez la section Points de terminaison du service.	4 avril 2023

Le traitement du langage naturel est désactivé par défaut	HealthLake a désactivé le traitement intégré du langage naturel (NLP) sur tous les magasins de données à compter du 20 février 2023. Vous pouvez envoyer un ticket d'assistance pour activer la fonctionnalité NLP intégrée. Créez un dossier en utilisant AWS Support Center Console. Pour créer votre dossier, connectez-vous à votre dossier Compte AWS et choisissez Créer un dossier. Pour en savoir plus sur le NLP intégré, voir Intégration du NLP avec. HealthLake •	20 février 2023

Index SQL et requête avec Amazon Athena	HealthLake prend en charge l'interrogation des données FHIR avec SQL à l'aide d'Amazon Athena. Pour plus d'informations, consultez Interrogation de HealthLake données avec Amazon Athena. • La fonctionnalité de requête SQL est activée par défaut pour tous les magasins de HealthLake données créés après le 14/11/2022. Si votre banque de données a été créée avant cette date, vous pouvez envoyer un ticket d'assistance pour activer la fonctionnalité de requête SQL. Créez un dossier en utilisant AWS Support Center Console. Pour créer votre dossier, connectez-vous à votre dossier Compte AWS et choisissez Créer un dossier. • Avec la fonctionnalité de requête SQL, les paramètres IAM d'accès HealthLake doivent être mis à jour. Pour créer des HealthLake banques de données et autoriser l'accès à celles-ci dans Athena, la politique `AWSLakeFormationDataAdmin` gérée doit être	14 novembre 2022

	ajoutée à votre utilisateur, groupe ou rôle IAM. Vous pouvez utiliser cette `AWSLakeFormationDataAdmin` politique pour créer des administrateurs de lacs de données et autoriser l'accès aux magasins de données d'Athena. Pour plus d'informations, consultez Configurer un utilisateur ou un rôle IAM.	
Augmentation de la taille totale des tâches d'importation	HealthLake met à jour le `Total import job size` pour une `StartFHIRImportJob` demande à 500 Go. Pour de plus amples informations, veuillez consulter Service Quotas.	3 octobre 2022
Ressource FHIR `Bundle`	HealthLake prend en charge le type de `Bundle` ressource FHIR pour traiter plusieurs ressources FHIR simultanément. Pour plus d'informations, voir Regrouper les ressources FHIR.	5 août 2022
Mises à jour des quotas pour les interactions FHIR	HealthLake met à jour les quotas pour les interactions de gestion des ressources du FHIR. Pour de plus amples informations, veuillez consulter Service Quotas.	16 juillet 2022

Paramètre de `_include` recherche FHIR	HealthLake ajoute la prise en charge du paramètre `_include` de recherche FHIR pour renvoyer des ressources supplémentaires dans une `search` demande. Pour plus d'informations, consultez la section Paramètres de recherche avancés.	16 juillet 2022
AWS HealthLake est généralement disponible	HealthLake est généralement disponible dans toutes les régions prises en charge. Pour plus d'informations, consultez la section Points de terminaison du service.	15 juillet 2021

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.