Guia do Desenvolvedor

AWS HealthLake

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

AWS HealthLake: Guia do Desenvolvedor

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

As marcas comerciais e imagens de marcas da Amazon não podem ser usadas no contexto de nenhum produto ou serviço que não seja da Amazon, nem de qualquer maneira que possa gerar confusão entre os clientes ou que deprecie ou desprestigie a Amazon. Todas as outras marcas comerciais que não pertencem à Amazon pertencem a seus respectivos proprietários, que podem ou não ser afiliados, patrocinados pela Amazon ou ter conexão com ela.

Table of Contents

O que é AWS HealthLake? 1
Aviso importante 2
Recursos 2
Serviços relacionados 3
Acesso 4
HIPAA 4
Preços 5
Introdução 6
Conceitos 6
estratégia de autorização 7
PNL integrada 7
Análise integrada 7
Configurar 7
Inscreva-se para um Conta da AWS 8
Configurar um usuário ou uma função do IAM 8
Adicionar um usuário ou uma função de administrador do Data Lake 10
Crie buckets do S3 12
Criar um datastore 12
Configurar permissões de importação 13
Configurar permissões de exportação 15
Instale o AWS CLI 19
Tutorial 20
Gerenciar datastores 22
Criar um datastore 22
Obter propriedades do datastore 30
Listar datastore 34
Marcando armazenamentos de dados 38
Marcar um datastore 38
Listando tags para um armazenamento de dados 42
Desmarcando um armazenamento de dados 45
Excluir um datastore 48
Gerenciando assinaturas do FHIR 52
Como funcionam as assinaturas do FHIR 52
Componentes principais 53

Tópicos de assinatura 53
Assinaturas 53
Canais de notificação 54
Cargas de notificação 54
Práticas recomendadas 55
Ciclo de vida da assinatura 55
Criando uma assinatura 57
Exemplo de cargas de assinatura 60
Exemplo de cargas úteis de notificação 65
Pesquisando assinaturas 69
Filtrando notificações 72
Importando dados do FHIR 76
Como iniciar trabalho de trabalho de trabalho de trabalho 78
Como obter propriedades do trabalho de importação 83
Listar trabalhos de importação 87
Gerenciando recursos do FHIR 93
Criar um recurso 95
Lendo um recurso 98
Lendo o histórico de recursos 100
Lendo o histórico específico da versão 103
Atualizar um recurso 105
Atualização condicional 108
Configurando o nível de validação para atualizações de recursos 109
Modificando um recurso 110
Formatos PATCH suportados 110
Usage 110
Formato de patch JSON 111
FHIRPath Formato do patch 114
Cabeçalhos de solicitação 116
Resposta da amostra 116
Comportamento 117
Tratamento de erros 117
Resumo dos recursos 117
Limitações 118
Recursos adicionais 118
Agrupando recursos 118

Agrupe como entidades independentes 123
Condicional PUTs 127
Agrupe como uma entidade única 130
Configurando o nível de validação para pacotes 133
Suporte limitado para “mensagem” do tipo Bundle 135
Transações assíncronas 137
Excluir um recurso 146
Exclusão condicional para FHIR 148
Idempotência e concorrência 150
Chaves de idempotência 150
ETag na AWS HealthLake 152
Pesquisando recursos do FHIR 154
Pesquisando com GET 154
Exemplos de pesquisa GET 157
Pesquisando com POST 158
Exemplos de pesquisa POST 161
Níveis de consistência de pesquisa 163
Níveis de consistência 164
Exemplo de uso 164
Práticas recomendadas 165
Exportação de dados FHIR 166
Iniciando um trabalho de exportação 166
Obtendo propriedades de trabalho de exportação 171
Listando trabalhos de exportação 175
Exemplos de código 181
Conceitos básicos 181
Ações 182
Integrar 229
Processamento de linguagem natural 229
Bibliotecas de PNL 230
Usando o FHIR APIs 232
Parâmetros de pesquisa 232
Exemplo de solicitações 235
Índice e consulta SQL 252
Introdução 252
Consultando com SQL 256

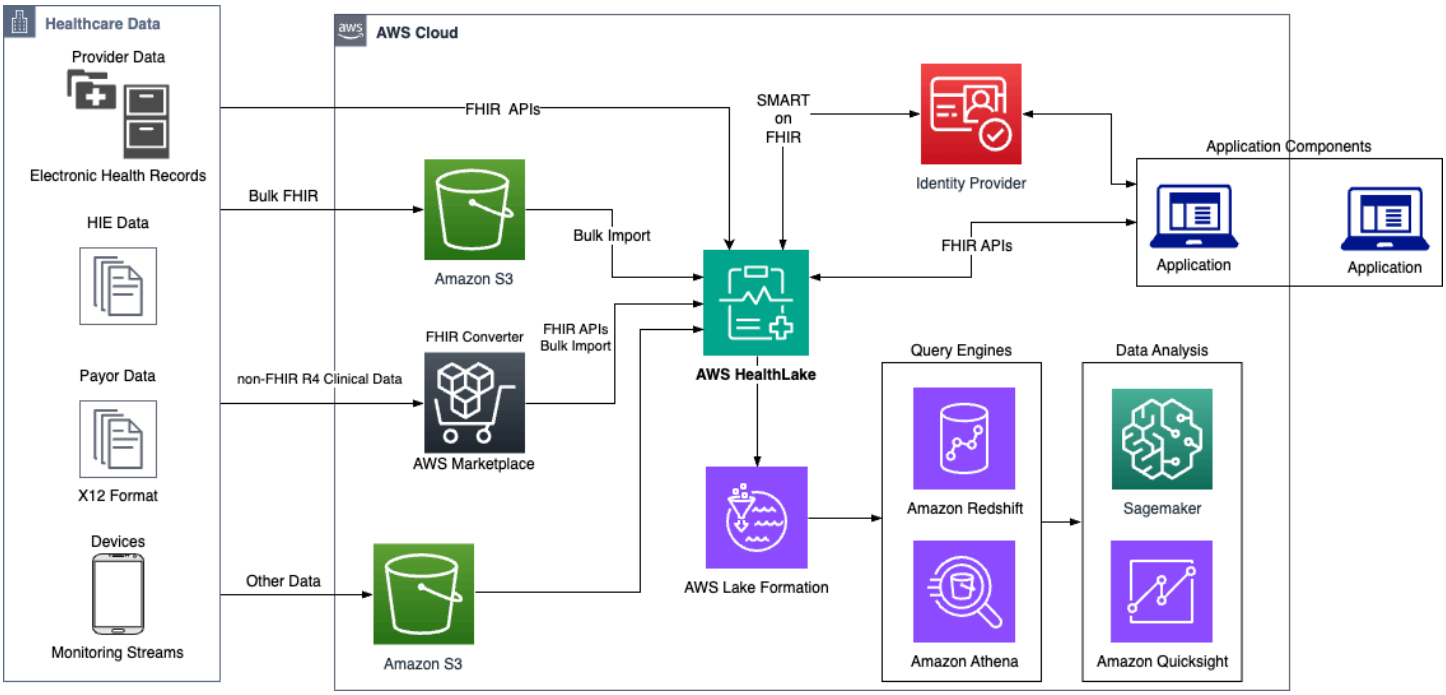
Consultas de exemplo 263
Monitoramento 270
CloudTrail (Chamadas de API) 270
AWS HealthLake Informações em CloudTrail 271
Entendendo as entradas do arquivo de AWS HealthLake log 272
CloudWatch (Métricas) 274
Visualizando HealthLake métricas 277
Criar um alarme 277
EventBridge (Eventos) 278
HealthLake eventos enviados para EventBridge 278
HealthLake estrutura do evento 280
Segurança 294
Proteção de dados 295
Criptografia em repouso 296
Chave KMS de propriedade da AWS 296
Chaves do KMS gerenciadas pelo cliente 296
Criar uma chave gerenciada pelo cliente 297
Permissões do IAM obrigatórias para usar uma chave KMS gerenciada pelo cliente 299
Criptografia em trânsito 306
Gerenciamento de identidade e acesso 306
Público 306
Autenticação com identidades 307
Gerenciar o acesso usando políticas 308
Como AWS HealthLake funciona com o IAM 310
Exemplos de políticas baseadas em identidade 315
AWS políticas gerenciadas 319
Solução de problemas 323
Validação de conformidade 325
Segurança da infraestrutura 326
Infraestrutura como código 326
HealthLake e CloudFormation modelos 326
Saiba mais sobre CloudFormation 327
Endpoints da VPC 327
Considerações sobre HealthLake VPC endpoints 327
Criação de uma interface VPC endpoint para; HealthLake 328
Criação de uma política de VPC endpoint para HealthLake 328

Práticas recomendadas 329
Resiliência 329
Referência 331
SMART em FHIR 331
Introdução 332
Autenticação 335
OAuth escopos 2.0 337
Validação de token 340
Autorização refinada 352
Documento de descoberta 353
Exemplo de solicitação 355
PARA R4 356
Declaração de capacidade 356
Validações de perfil 357
Resource types 364
Parâmetros de pesquisa 366
$Operações 380
Compliance 531
CMS 532
HealthLake 537
Endpoints e cotas 538
Tipos de dados pré-carregados 550
Projetos de amostra 551
Solução de problemas 551
Trabalhando com AWS SDKs 559
Versões 561
..... dlxxxviii

O que é AWS HealthLake?

AWS HealthLake é um serviço qualificado pela HIPAA para armazenar, analisar e compartilhar dados de saúde na nuvem usando a especificação R4 da Fast Healthcare Interoperability Resources (FHIR). HealthLake Os casos de uso incluem:

- Dados de saúde corporativos — gerencie e compartilhe dados de saúde do FHIR R4 diretamente, Nuvem AWS preservando o alto desempenho e a disponibilidade.
- Interoperabilidade na área de saúde — Apoie a conformidade do cliente com o 21st Century Cures Act para acesso de pacientes por meio de um armazenamento de dados FHIR totalmente gerenciado.
- Processamento de linguagem natural (PNL) — Utilize modelos integrados de PNL para extrair informações médicas significativas de dados de saúde não estruturados.
- Análise multimodal — combine HealthLake dados com dados e AWS HealthImaging AWS HealthOmics dados para fornecer informações sobre medicina de precisão.



Tópicos

- Aviso importante
- Características do AWS HealthLake

- AWS Serviços relacionados
- Acessando AWS HealthLake
- Elegibilidade e segurança de dados da HIPAA
- Preços

Aviso importante

AWS HealthLake não substitui o aconselhamento, diagnóstico ou tratamento médico profissional e não se destina a curar, tratar, mitigar, prevenir ou diagnosticar qualquer doença ou condição de saúde. Você é responsável por instituir a avaliação humana como parte de qualquer uso de AWS HealthLake, inclusive em associação com, qualquer produto de terceiros destinado a informar a tomada de decisões clínicas. AWS HealthLake só deve ser usado no atendimento ao paciente ou em cenários clínicos após análise por profissionais médicos treinados, aplicando um bom julgamento médico.

Características do AWS HealthLake

AWS HealthLake fornece os seguintes recursos.

Importar dados de saúde do FHIR R4

> Com a ação de importação HealthLake nativa, você pode migrar facilmente seus dados FHIR de um bucket do Amazon S3 para um armazenamento de HealthLake dados, incluindo notas clínicas, relatórios de laboratório, reclamações de seguros e muito mais. HealthLake suporta a especificação FHIR R4 para troca de dados de assistência médica. Se necessário, você pode trabalhar com um AWS HealthLake parceiro para converter seus dados de saúde para o formato FHIR R4.

Armazene dados de saúde de forma segura, compatível e auditável

> Um armazenamento HealthLake de dados ajuda a indexar dados de saúde para que possam ser consultados. O armazenamento de dados cria uma visão completa do histórico médico de cada paciente em ordem cronológica e facilita a troca de informações usando a especificação FHIR R4. E está sempre funcionando para manter seu índice atualizado, oferecendo a capacidade de pesquisar as informações a qualquer momento usando interações padrão do FHIR R4 com armazenamento primário durável e escalabilidade de índice.

Aproveite o servidor FHIR transacional

Aproveite o FHIR APIs para validação padrão de recursos, a autorização SMART na FHIR e os recursos de exportação da API FHIR de dados em massa para apoiar a unificação e análise de seus dados para reduzir os custos operacionais e melhorar a tomada de decisões. HealthLake oferece suporte à conformidade do cliente com os mais recentes padrões regulatórios de ONC e CMS, incluindo: HL7 FHIR R4 APIs, FHIR Bulk Data Access, US Core IG STU, HL7 SMART App Launch Framework IG, 2.0 e OpenID Connect. OAuth

Transforme dados médicos não estruturados usando a PNL

O processamento integrado de linguagem natural médica (NLP) transforma todos os dados brutos de texto médico em um armazenamento de HealthLake dados para entender e extrair informações significativas de dados de saúde não estruturados. Com a PNL médica integrada, você pode extrair automaticamente entidades, relacionamentos de entidades, características de entidades e informações de saúde protegidas (PHI) do seu texto médico. As entidades extraídas do NLP são armazenadas como recursos nativos do FHIR R4 em um armazenamento de HealthLake dados e podem ser acessadas por meio do FHIR R4 ou do Amazon Athena (SQL). APIs

AWS Serviços relacionados

AWS HealthLake apresenta uma forte integração com outros AWS serviços. O conhecimento dos serviços a seguir é útil para aproveitar totalmente HealthLake.

- AWS Identity and Access Management— Use o IAM para gerenciar com segurança as identidades e o acesso aos recursos. HealthLake
- Amazon Simple Storage Service — Use o Amazon S3 como uma área de armazenamento para importar dados DICOM. HealthLake
- AWS CloudTrail— Use CloudTrail para rastrear a atividade HealthLake do usuário e o uso da API.
- Amazon CloudWatch — Use CloudWatch para observar e monitorar HealthLake recursos.
- AWS CloudFormation— Use CloudFormation para implementar modelos de infraestrutura como código (IaC) para criar recursos em HealthLake.
- AWS PrivateLink— Use a Amazon VPC para estabelecer conectividade entre HealthLake a Amazon Virtual Private Cloud sem expor dados à Internet.
- Amazon EventBridge — Use EventBridge para criar aplicativos escaláveis e orientados por eventos criando regras que direcionam HealthLake eventos para destinos.

- AWS Lake Formation— Use o Lake Formation para governar, proteger e compartilhar HealthLake dados de forma centralizada para análise e aprendizado de máquina.
- Amazon Athena — Use o Athena para consultar HealthLake dados com SQL para permitir uma análise mais profunda.

Acessando AWS HealthLake

Você pode acessar AWS HealthLake usando o Console de gerenciamento da AWS, AWS Command Line Interface AWS SDKs e. Este guia fornece instruções processuais para o Console de gerenciamento da AWS e exemplos de código para e. AWS CLI AWS SDKs

AWS Command Line Interface (AWS CLI)

O AWS CLI fornece comandos para um amplo conjunto de AWS produtos e é compatível com Windows, Mac e Linux. Para obter mais informações, consulte o Guia do usuário do AWS Command Line Interface .

AWS SDKs

AWS SDKs forneça bibliotecas, exemplos de código e outros recursos para desenvolvedores de software. Essas bibliotecas fornecem funções básicas que automatizam tarefas como a assinatura criptografada das solicitações, novas tentativas de solicitações e tratamento das respostas de erro. Para obter mais informações, consulte Ferramentas para desenvolver AWS.

Console de gerenciamento da AWS

O Console de gerenciamento da AWS fornece uma interface de usuário baseada na web para gerenciamento HealthLake e seus recursos associados. Se você se inscreveu em uma AWS conta, pode entrar no HealthLake console.

Elegibilidade e segurança de dados da HIPAA

Este é um serviço qualificado da HIPAA. Para obter mais informações sobre AWS a Lei de Portabilidade e Responsabilidade de Seguros de Saúde dos EUA de 1996 (HIPAA) e o uso de AWS serviços para processar, armazenar e transmitir informações de saúde protegidas (PHI), consulte Visão geral da HIPAA.

As conexões que HealthLake contêm PHI e informações de identificação pessoal (PII) devem ser criptografadas. Por padrão, todas as conexões devem HealthLake usar HTTPS sobre TLS.

HealthLake armazena conteúdo criptografado do cliente e opera de acordo com o Modelo de Responsabilidade AWS Compartilhada.

Preços

Para obter informações sobre HealthLake preços, consulte AWS HealthLake preços. Para estimar os custos, use a calculadora HealthLake de preços.

Começando com AWS HealthLake

Para começar a usar AWS HealthLake, configure uma AWS conta e crie um AWS Identity and Access Management usuário. Para usar o AWS CLIou o AWS SDKs, você deve instalá-los e configurá-los.

> **Note**
>
> O Referência capítulo deste guia fornece conteúdo de suporte para SMART em FHIR, FHIR R4 e. AWS HealthLake Por exemplo, você pode encontrar informações sobre SMART na configuração do FHIR, validações de perfil FHIR suportadas e endpoints. HealthLake

Depois de aprender sobre HealthLake conceitos e configuração, um breve tutorial com exemplos de código está disponível para ajudar você a começar.

Tópicos

- AWS HealthLake conceitos
- Configurar AWS HealthLake
- AWS HealthLake tutorial

AWS HealthLake conceitos

A terminologia e os conceitos a seguir são fundamentais para sua compreensão e uso do AWS HealthLake.

Conceitos

- Estratégia de autorização do armazenamento de dados
- PNL integrada
- Análise integrada

Estratégia de autorização do armazenamento de dados

Um armazenamento HealthLake de dados é um repositório de dados de saúde do FHIR R4 que reside em um único. Região da AWS HealthLake suporta as seguintes estratégias de autorização do armazenamento de dados.

- Autorização SigV4 — HealthLake autoriza chamadas da API FHIR usando a autorização AWS Signature Version 4 (SigV4).
- SMART na autorização do FHIR — HealthLake autoriza chamadas de API do FHIR usando aplicativos médicos substituíveis e tecnologias reutilizáveis (SMART) na autorização do FHIR.

Para obter mais informações, consulte Criando um armazenamento HealthLake de dados.

PNL integrada

AWS HealthLake integra-se às bibliotecas de processamento de linguagem natural (PNL) qualificadas pela HIPAA para extrair dados de saúde significativos de textos médicos não estruturados. As bibliotecas da PNL identificam entidades médicas como condições, medicamentos, dosagens, testes, tratamentos e procedimentos. Eles reconhecem as relações entre as entidades e as vinculam a bibliotecas de ontologia médica, como ICD-10-CM e. RxNorm Para obter mais informações, consulte Processamento de linguagem natural (NLP) integrado para HealthLake.

Análise integrada

AWS HealthLake vai além do `bundle` APIs FHIR `search` e fornece análises integradas para consultar e analisar grandes volumes de dados de saúde. Durante a importação, gera HealthLake automaticamente tabelas para índice e consulta SQL. Isso permite que você obtenha insights acionáveis a partir de dados complexos de saúde sem exigir um extenso trabalho de engenharia de dados. Para obter mais informações, consulte Consultando HealthLake dados com o Amazon Athena e AWS HealthLake projetos de amostra.

Configurar AWS HealthLake

Neste capítulo, você usa o Console de gerenciamento da AWS para configurar as permissões necessárias para começar a usar AWS HealthLake e criar um armazenamento de dados. Para configurar permissões para criar um armazenamento de dados, você cria um usuário ou uma função do IAM que é administrador e HealthLake administrador do data lake. Você torna esse usuário

um administrador de data lake no AWS Lake Formation. O administrador do data lake concede ao Lake Formation acesso aos recursos necessários para usar o Amazon Athena para consultar um armazenamento de dados. Depois de criar um armazenamento de HealthLake dados, você pode configurar permissões para importar e exportar arquivos.

Tópicos

- Inscreva-se para um Conta da AWS
- Configurar um usuário ou uma função do IAM para usar HealthLake (administrador do IAM)
- Adicionar um usuário ou função como administrador do Data Lake em Lake Formation (administrador do IAM)
- Crie buckets do S3
- Criar um datastore
- Configurando permissões para trabalhos de importação
- Configurando permissões para trabalhos de exportação
- Instale o AWS CLI

Inscreva-se para um Conta da AWS

Para começar AWS, você precisa de um Conta da AWS. Para obter informações sobre como criar um Conta da AWS, consulte Introdução a um Conta da AWS no Guia de AWS Gerenciamento de contas referência.

Configurar um usuário ou uma função do IAM para usar HealthLake (administrador do IAM)

> Persona: administrador do IAM
>
> Um usuário que pode criar usuários e funções do IAM e adicionar administradores de data lake.

Essas etapas neste tópico devem ser executadas por um administrador do IAM.

Para conectar seu armazenamento de HealthLake dados ao Athena, você precisa criar um usuário ou uma função do IAM que seja administrador e administrador do data lake. HealthLake Esse novo

usuário ou função concede acesso aos recursos encontrados em um armazenamento de dados por meio do AWS Lake Formation e tem a política `AmazonHealthLakeFullAccess` AWS gerenciada adicionada ao usuário ou função.

> **Important**
>
> Um usuário ou função do IAM que seja administrador de data lake não pode criar novos administradores de data lake. Para adicionar mais administrador de data lake, você deve usar um usuário ou uma função do IAM que tenha recebido `AdministratorAccess` acesso.

Para criar um administrador

1. Adicione a política AWS gerenciada **`AmazonHealthlakeFullAccess`** do IAM a um usuário ou função na sua organização.

   Se você não estiver familiarizado com a criação de um usuário do IAM, consulte Criação de um usuário do IAM e Visão geral das políticas AWS do IAM no Guia do usuário do IAM.
2. Conceda ao usuário ou à função do IAM acesso ao AWS Lake Formation.

   - Adicione a seguinte política AWS gerenciada do IAM a um usuário ou função na sua organização: **`AWSLakeFormationDataAdmin`**

   > **Note**
   >
   > A `AWSLakeFormationDataAdmin` política concede acesso a todos os recursos AWS do Lake Formation. Recomendamos que você sempre use as permissões mínimas necessárias para realizar sua tarefa. Para obter mais informações, consulte Práticas recomendadas de segurança no IAM no Guia do usuário do IAM.
3. Adicione a seguinte política embutida ao usuário ou à função. Para obter mais informações, consulte Políticas em linha no Guia do usuário do 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": "*"
        }
    ]
}
```

Para obter mais informações sobre a `AWSLakeFormationDataAdmin` política, consulte Lake Formation Personas e Referência de permissões do IAM no Guia do desenvolvedor do AWS Lake Formation.

Adicionar um usuário ou função como administrador do Data Lake em Lake Formation (administrador do IAM)

> **Note**
>
> Essa etapa é necessária se você estiver integrando. Índice e consulta SQL

Em seguida, o administrador do IAM deve adicionar o usuário ou a função criada na etapa anterior como administrador do data lake no Lake Formation.

Para adicionar um usuário ou uma função do IAM como administrador do data lake

1. Abra o console do AWS Lake Formation: https://console.aws.amazon.com/lakeformation/

> **Note**
>
> Se esta é a primeira vez que você visita Lake Formation, uma caixa de diálogo Welcome to Lake Formation é exibida solicitando que você defina um administrador do Lake Formation.

2. Atribua ao novo usuário ou função um administrador de data lake do AWS Lake Formation.
   - Opção 1: Se você recebeu a caixa de diálogo Welcome to Lake Formation.
     1. Escolha Adicionar outros AWS usuários ou funções.
     2. Escolha a seta para baixo (▼).
     3. Escolha o HealthLake administrador que você gostaria que também fosse administrador do Lake Formation.
     4. Escolha Começar.
   - Opção 2: Use o painel de navegação (☰).
     1. Escolha o painel de navegação (☰).
     2. Em Permissões, escolha Funções e tarefas administrativas.
     3. Na seção Administradores do Data Lake, selecione Escolher administradores.
     4. Na caixa de diálogo Gerenciar administradores do data lake, escolha a seta para baixo (▼).
     5. Em seguida, selecione ou pesquise os HealthLake administradores, usuários ou funções que você também deseja que sejam administradores do Lake Formation.
     6. Escolha Salvar.
3. Altere as configurações de segurança padrão a serem gerenciadas pelo Lake Formation.
   Os recursos do armazenamento de HealthLake dados precisam ser gerenciados pelo Lake Formation, não pelo IAM. Para atualizar, consulte Alterar o modelo de permissão padrão no AWS Lake Formation Developer Guide.

Crie buckets do S3

Para importar dados do FHIR R4 AWS HealthLake, dois buckets Amazon S3 são recomendados. O bucket de entrada do Amazon S3 contém os dados FHIR a serem importados e lidos desse bucket HealthLake . O bucket de saída do Amazon S3 armazena os resultados de processamento do trabalho de importação e HealthLake grava (registros) nesse bucket.

> **Note**
>
> Devido à política AWS Identity and Access Management (IAM), seus nomes de bucket do Amazon S3 devem ser exclusivos. Para obter mais informações, consulte as Regras para nomear buckets no Guia do usuário do Amazon Simple Storage Service.

Para fins deste guia, especificamos os seguintes buckets de entrada e saída do Amazon S3 ao configurar as permissões de importação posteriormente nesta seção.

- Bucket de entrada: `arn:aws:s3:::amzn-s3-demo-source-bucket`
- Bucket de saída: `arn:aws:s3:::amzn-s3-demo-logging-bucket`

Para obter informações adicionais, consulte Criar um bucket no Guia do usuário do Amazon S3.

Criar um datastore

Um armazenamento HealthLake de dados é um repositório de dados do FHIR R4 que reside em uma única região. AWS Uma AWS conta pode ter zero ou muitos armazenamentos de dados. HealthLake suporta duas estratégias de autorização de armazenamento de dados.

> **Importante**
>
> Antes de criar um armazenamento de HealthLake dados, revise as políticas de controle de serviços (SCPs) em sua AWS organização que podem restringir a criação ou o gerenciamento de HealthLake recursos. Os SCPs podem impedir a criação bem-sucedida de armazenamentos de HealthLake dados, mesmo que suas permissões do IAM estejam configuradas corretamente.

A `datastoreID` é gerado quando você cria um armazenamento HealthLake de dados. Você deve usar o `datastoreID` ao configurar as permissões de importação posteriormente nesta seção.

Para criar um armazenamento HealthLake de dados, consulteCriando um armazenamento HealthLake de dados.

Configurando permissões para trabalhos de importação

Antes de importar arquivos para um armazenamento de dados, você deve conceder HealthLake permissão para acessar seus buckets de entrada e saída no Amazon S3. Para conceder HealthLake acesso, você cria uma função de serviço do IAM para HealthLake, adiciona uma política de confiança à função para conceder permissões de HealthLake assumir função e anexa uma política de permissões à função que concede acesso aos seus buckets do Amazon S3.

Ao criar um trabalho de importação, você especifica o Amazon Resource Name (ARN) dessa função para o. `DataAccessRoleArn` Para obter mais informações sobre funções e políticas de confiança do IAM, consulte Funções do IAM.

Depois de configurar a permissão, você estará pronto para importar arquivos para o seu armazenamento de dados com uma tarefa de importação. Para obter mais informações, consulte Iniciando um trabalho de importação de FHIR.

Para configurar permissões de importação

1. Caso ainda não tenha feito isso, crie um bucket Amazon S3 de destino para os arquivos de log de saída. O bucket do Amazon S3 deve estar na mesma AWS região do serviço, e o Block Public Access deve estar ativado para todas as opções. Para saber mais, consulte Como usar o Amazon S3 para bloquear o acesso público. Uma chave KMS Amazon-owned ou de propriedade do cliente também deve ser usada para criptografia. Para saber mais sobre o uso de chaves KMS, consulte Amazon Key Management Service.
2. Crie uma função de serviço de acesso a dados HealthLake e dê permissão ao HealthLake serviço para assumi-la com a seguinte política de confiança. HealthLake usa isso para gravar o bucket de saída do Amazon S3.

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. Adicione uma política de permissões à função de acesso a dados que permita que ela acesse o bucket do Amazon S3. `amzn-s3-demo-bucket`Substitua pelo nome do seu bucket.

   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"
    }]
}
```

Configurando permissões para trabalhos de exportação

Antes de exportar arquivos de um armazenamento de dados, você deve conceder HealthLake permissão para acessar seu bucket de saída no Amazon S3. Para conceder HealthLake acesso, você cria uma função de serviço do IAM paraHealthLake, adiciona uma política de confiança à função para conceder permissões de HealthLake assumir função e anexa uma política de permissões à função que concede acesso ao seu bucket do Amazon S3.

Se você já criou uma função para HealthLake, você pode reutilizá-la e conceder a ela as permissões adicionais para seu bucket de exportação do Amazon S3 listado neste tópico. Para saber mais sobre as políticas de confiança e os perfis do IAM, consulte Políticas e permissões do IAM.

> **Importante**
>
> HealthLake oferece suporte às solicitações de exportação do SDK nativo e à operação FHIR R4. `$export` Ações do IAM separadas devem ser fornecidas dependendo da API

de exportação que você decidir usar. Isso permite que você `allow` gerencie `deny` as permissões separadamente. Se você quiser restringir as exportações da API REST do HealthLake SDK e do FHIR, aplique as permissões de negação às ações separadas do IAM. As alterações de permissão do usuário do IAM não são necessárias se você der aos usuários acesso total HealthLake a.

Utilizar AWS CLI and AWS SDKs:

As seguintes HealthLake ações nativas estão disponíveis para exportar dados de um armazenamento de dados usando os AWS SDKs AWS CLI e:

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

Usando APIs FHIR:

As seguintes ações do IAM estão disponíveis para exportar dados de um armazenamento de HealthLake dados e para cancelar (excluir) um trabalho de exportação usando a operação FHIR: `$export`
`POST`:

- `StartFHIRExportJobWithPost`

`GET`:

- `StartFHIRExportJobWithGet`
- `DescribeFHIRExportJobWithGet`
- `GetExportedFile`

`DELETE`:

- `CancelFHIRExportJobWithDelete`

O usuário ou função que configura as permissões deve ter permissão para criar funções, criar políticas e anexar políticas às funções. A política do IAM a seguir concede essas permissões.

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"
                }
            }
        }
    ]
}
```

Para configurar permissões de exportação

1. Caso ainda não tenha feito isso, crie um bucket Amazon S3 de destino para os dados que você exportará do seu armazenamento de dados. O bucket do Amazon S3 deve estar na mesma região da AWS do serviço, e o Block Public Access deve estar ativado para todas as opções. Para saber mais, consulte Como usar o Amazon S3 para bloquear o acesso público. Uma chave KMS Amazon-owned ou de propriedade do cliente também deve ser usada para criptografia. Para saber mais sobre o uso de chaves KMS, consulte Amazon Key Management Service.
2. Se você ainda não o fez, crie uma função de serviço de acesso a dados HealthLake e dê permissão ao HealthLake serviço para assumi-la com a seguinte política de confiança. HealthLakeusa isso para gravar o bucket de saída do Amazon S3. Se você já criou

umConfigurando permissões para trabalhos de importação, pode reutilizá-lo e conceder permissões para seu bucket do Amazon S3 na próxima etapa.

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. Adicione uma política de permissões à função de acesso a dados que permita que ela acesse seu bucket de saída do Amazon S3. `amzn-s3-demo-bucket`Substitua pelo nome do seu bucket.

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"
    }]
}
```

Instale o AWS CLI

O AWS CLI é necessário para descrever e listar as propriedades da tarefa de HealthLake importação e exportação. Você também pode solicitar essas informações usando HealthLake SDKs.

Para configurar o AWS CLI

1. Faça download e configure a AWS CLI. Para obter instruções, consulte os seguintes tópicos no Guia do usuário do AWS Command Line Interface .

   - Instalando ou atualizando a versão mais recente do AWS CLI
   - Começando com o AWS CLI

2. No AWS CLI `config` arquivo, adicione um perfil nomeado para o administrador. Você usa esse perfil ao executar os AWS CLI comandos. De acordo com o princípio de segurança do privilégio mínimo, recomendamos que você crie um perfil do IAM separado com privilégios específicos para as tarefas que estão sendo executadas. Para obter mais informações sobre perfis nomeados, consulte Configurações de arquivos de configuração e credenciais no Guia do usuário da AWS Command Line Interface .

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

3. Verifique a configuração usando o comando `help`.

```
aws healthlake help
```

Se o AWS CLI estiver configurado corretamente, você verá uma breve descrição AWS HealthLake e uma lista dos comandos disponíveis.

AWS HealthLake tutorial

Objetivo

Neste tutorial, você importará dados do FHIR R4 em um armazenamento de HealthLake dados usando ações nativas. HealthLake Em seguida, você gerenciará (criará, lerá, atualizará, excluirá) um recurso FHIR usando o FHIR. RESTful APIs Para concluir o tutorial, você exportará dados FHIR usando ações nativas HealthLake.

Pré-requisitos

Todos os procedimentos listados em Configurar são necessários para concluir este tutorial.

Etapas do tutorial

1. Inicie o trabalho de importação do FHIR
2. Obtenha propriedades de trabalho de importação do FHIR
3. Crie um recurso FHIR
4. Leia o recurso FHIR

5. Atualizar recurso FHIR
6. Excluir recurso FHIR
7. Exportar dados FHIR
8. Excluir datastore

Gerenciando armazenamentos de dados com AWS HealthLake

Com AWS HealthLake, você cria e gerencia armazenamentos de dados para recursos do FHIR R4. Quando você cria um armazenamento de HealthLake dados, um repositório de dados FHIR é disponibilizado por meio de um endpoint de API. RESTful Você pode optar por importar (pré-carregar) dados de saúde FHIR R4 de código aberto do Synthea para seu armazenamento de dados ao criá-los. Para obter mais informações, consulte Tipos de dados pré-carregados.

> **Importante**
>
> HealthLake suporta dois tipos de estratégias de autorização de armazenamento de dados FHIR, AWS SigV4 ou SMART no FHIR. Você deve escolher uma das estratégias de autorização antes de criar um armazenamento de dados HealthLake FHIR. Para obter mais informações, consulte Estratégia de autorização do armazenamento de dados.

Para encontrar os recursos (comportamentos) relacionados ao FHIR de um armazenamento de HealthLake dados ativo, recupere sua Declaração de Capacidade.

Os tópicos a seguir descrevem como usar ações nativas da HealthLake nuvem para criar, descrever, listar, marcar e excluir armazenamentos de dados FHIR usando o AWS CLI AWS SDKs, e. Console de gerenciamento da AWS

Tópicos

- Criando um armazenamento HealthLake de dados
- Obtendo propriedades do armazenamento de HealthLake dados
- Listando armazenamentos HealthLake de dados
- Marcando armazenamentos HealthLake de dados
- Excluindo um armazenamento de HealthLake dados

Criando um armazenamento HealthLake de dados

Use `CreateFHIRDatastore` para criar um armazenamento de AWS HealthLake dados em conformidade com a especificação FHIR R4. HealthLake os armazenamentos de dados são usados

para importar, gerenciar, pesquisar e exportar dados FHIR. Você pode optar por importar (pré-carregar) dados de saúde FHIR R4 de código aberto do Synthea para seu armazenamento de dados ao criá-los. Para obter mais informações, consulte Tipos de dados pré-carregados.

> **Importante**
>
> HealthLake suporta dois tipos de estratégias de autorização de armazenamento de dados FHIR, AWS SigV4 ou SMART no FHIR. Você deve escolher uma das estratégias de autorização antes de criar um armazenamento de dados HealthLake FHIR. Para obter mais informações, consulte Estratégia de autorização do armazenamento de dados.

Quando você cria um armazenamento de HealthLake dados, um repositório de dados FHIR é disponibilizado por meio de um RESTful endpoint de API. Depois de criar seu armazenamento de HealthLake dados, você pode solicitar sua Declaração de Capacidade para encontrar todos os recursos (comportamentos) relacionados ao FHIR associados.

Os menus a seguir fornecem exemplos para o AWS CLI AWS SDKs e e um procedimento para Console de gerenciamento da AWS o. Para obter mais informações, consulte `CreateFHIRDatastore` na Referência de APIs do AWS HealthLake .

Para criar um armazenamento HealthLake de dados

Escolha um menu com base na sua preferência de acesso AWS HealthLake a.

AWS CLI and SDKs

CLI

AWS CLI

Exemplo 1: Criar um armazenamento de dados habilitado para SIGv4 HealthLake

O `create-fhir-datastore` exemplo a seguir demonstra como criar um novo armazenamento de dados em AWS HealthLake.

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

Saída:

```
{
    "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)"
}
```

Exemplo 2: Crie um SMART no armazenamento de dados habilitado para FHIR HealthLake

O `create-fhir-datastore` exemplo a seguir demonstra como criar um novo SMART no armazenamento de dados habilitado para FHIR em. 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
```

Conteúdo 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\"]}"
}
```

Saída:

```
{
    "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)"
}
```

- Para obter detalhes da API, consulte Create FHIRDatastore in AWS CLI Command Reference.

Python

SDK para 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
```

O código a seguir mostra um exemplo de parâmetros para um SMART no armazenamento de dados habilitado para 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
            )
```

- Para obter detalhes da API, consulte Referência da API Create FHIRDatastore in AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte a referência da API Create FHIRDatastore in AWS SDK for SAP ABAP.

> **Exemplo de disponibilidade**
>
> Não consegue encontrar o que precisa? Solicite um exemplo de código usando o link Fornecer feedback na barra lateral direita desta página.

AWS Console

> **Observação**
>
> O procedimento a seguir cria um armazenamento HealthLake de dados com autorização AWS SigV4. O HealthLake console não suporta a criação de um SMART no armazenamento de dados FHIR.

Para criar um armazenamento HealthLake de dados com autorização AWS SigV4

1. Faça login na página Criar armazenamento de dados no HealthLake console.
2. Escolha Criar armazenamento de dados.
3. Na seção Configurações do Armazenamento de Dados, em Nome do Armazenamento de Dados, especifique um nome.
4. (Opcional) Na seção Configurações do Data Store, em Pré-carregar dados de amostra, marque a caixa de seleção para pré-carregar dados do Synthea. Os dados Synthea são um conjunto de dados de amostra de código aberto. Para obter mais informações, consulte Tipos de dados pré-carregados do Synthea para HealthLake.
5. Na seção Criptografia do armazenamento de dados, escolha Usar chave própria da AWS (padrão) ou Escolher uma chave diferente do AWS KMS (avançada).
6. Na seção Tags - opcional, você pode adicionar tags ao seu armazenamento de dados. Para saber mais sobre como marcar seu armazenamento de dados, consulteMarcando armazenamentos HealthLake de dados.

7. Escolha Criar armazenamento de dados.

   O status do seu armazenamento de dados está disponível na página Armazenamentos de dados.

Obtendo propriedades do armazenamento de HealthLake dados

Use `DescribeFHIRDatastore` para obter propriedades para um armazenamento AWS HealthLake de dados. Os menus a seguir fornecem um procedimento para o Console de gerenciamento da AWS e exemplos de código para AWS CLI AWS SDKs e. Para obter mais informações, consulte `DescribeFHIRDatastore` na Referência de APIs do AWS HealthLake .

Para obter propriedades para um armazenamento HealthLake de dados

Escolha um menu com base na sua preferência de acesso AWS HealthLake a.

AWS CLI and SDKs

CLI

AWS CLI

Como descrever um datastore FHIR

O `describe-fhir-datastore` exemplo a seguir demonstra como encontrar as propriedades de um armazenamento de dados em AWS HealthLake.

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

Saída:

```
{
    "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
        }
    }
}
```

- Para obter detalhes da API, consulte Descrever FHIRDatastore na Referência de AWS CLI Comandos.

Python

SDK para 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
```

- Para obter detalhes da API, consulte a Referência da API Descrever FHIRDatastore no AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte Descrição FHIRDatastore na referência da API ABAP do AWS SDK for SAP.

> **Exemplo de disponibilidade**
>
> Não consegue encontrar o que precisa? Solicite um exemplo de código usando o link Fornecer feedback na barra lateral direita desta página.

AWS Console

1. Faça login na página Armazenamentos de dados no HealthLake console.
2. Escolha um datastore.

   A página de detalhes do armazenamento de dados é aberta e todas as propriedades do armazenamento de HealthLake dados estão disponíveis.

Listando armazenamentos HealthLake de dados

Use `ListFHIRDatastores` para listar todos os armazenamentos de HealthLake dados na conta de um usuário, independentemente do status do armazenamento de dados. Os menus a seguir fornecem um procedimento para o Console de gerenciamento da AWS e exemplos de código para AWS CLI AWS SDKs e. Para obter mais informações, consulte `ListFHIRDatastores` na Referência de APIs do AWS HealthLake .

Para listar todos os armazenamentos HealthLake de dados

Escolha um menu com base na sua preferência de acesso AWS HealthLake a.

AWS CLI and SDKs

CLI

AWS CLI

Como listar datastores FHIR

O `list-fhir-datastores` exemplo a seguir mostra como usar o comando e como os usuários podem filtrar os resultados com base no status do armazenamento de dados em AWS HealthLake.

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

Saída:

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

- Para obter detalhes da API, consulte Lista FHIRDatastores na Referência de AWS CLI Comandos.

Python

SDK para 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
```

- Para obter detalhes da API, consulte Referência da API List FHIRDatastores in AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte Lista FHIRDatastores no AWS SDK para referência da API SAP ABAP.

> **Exemplo de disponibilidade**
>
> Não consegue encontrar o que precisa? Solicite um exemplo de código usando o link Fornecer feedback na barra lateral direita desta página.

AWS Console

- Faça login na página Armazenamentos de dados no HealthLake console.

Todos os armazenamentos de HealthLake dados estão listados na seção Armazenamentos de dados.

Marcando armazenamentos HealthLake de dados

Você pode atribuir metadados aos armazenamentos de HealthLake dados na forma de tags. Cada tag é um rótulo que consiste em um valor e uma chave definida pelo usuário. As tags ajudam você a gerenciar, identificar, organizar, pesquisar e filtrar armazenamentos de dados.

> **Importante**
>
> Não armazene informações de saúde protegidas (PHI), informações de identificação pessoal (PII) nem outras informações confidenciais ou sigilosas em tags. As tags não devem ser usadas para dados sensíveis ou privados.

Os tópicos a seguir descrevem como usar operações de HealthLake marcação usando o Console de gerenciamento da AWS AWS CLI, e. AWS SDKs Para obter mais informações, consulte Como marcar seus AWS recursos no Referência geral da AWS Guia.

Tópicos

- Marcando um armazenamento de HealthLake dados
- Listando tags para um armazenamento HealthLake de dados
- Desmarcando um armazenamento de dados HealthLake

Marcando um armazenamento de HealthLake dados

Use `TagResource` para marcar um armazenamento HealthLake de dados. Os menus a seguir fornecem um procedimento para o Console de gerenciamento da AWS e exemplos de código para AWS CLI AWS SDKs e. Para obter mais informações, consulte `TagResource` na Referência de APIs do AWS HealthLake .

Para marcar um armazenamento HealthLake de dados

Escolha um menu com base na sua preferência de acesso AWS HealthLake a.

AWS CLI and SDKs

CLI

AWS CLI

Como adicionar uma tag ao datastore

O exemplo de `tag-resource` a seguir mostra como adicionar uma tag a um datastore.

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

Este comando não produz saída.

.

- Para obter detalhes da API, consulte TagResourcena Referência de AWS CLI Comandos.

Python

SDK para 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
```

- Para obter detalhes da API, consulte a TagResourceReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte a TagResourcereferência da API AWS SDK for SAP ABAP.

> **Exemplo de disponibilidade**
>
> Não consegue encontrar o que precisa? Solicite um exemplo de código usando o link Fornecer feedback na barra lateral direita desta página.

AWS Console

1. Faça login na página Armazenamentos de dados no HealthLake console.
2. Escolha um datastore.

   A página Detalhes do datastore é aberta.
3. Na seção Tags, escolha Gerenciar tags.

   A página Gerenciar tags é aberta.
4. Selecione Adicionar nova tag.
5. Insira uma chave e um valor (opcional).
6. Escolha Salvar.

Listando tags para um armazenamento HealthLake de dados

Use `ListTagsForResource` para listar tags para um armazenamento HealthLake de dados. Os menus a seguir fornecem um procedimento para o Console de gerenciamento da AWS e exemplos de código para AWS CLI AWS SDKs e. Para obter mais informações, consulte `ListTagsForResource` na Referência de APIs do AWS HealthLake .

Para listar tags para um armazenamento HealthLake de dados

Escolha um menu com base na sua preferência de acesso AWS HealthLake a.

AWS CLI and SDKs

CLI

AWS CLI

Como listar tags para um datastore

O exemplo de `list-tags-for-resource` a seguir lista as tags associadas ao datastore especificado.

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

Saída:

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

- Para obter detalhes da API, consulte ListTagsForResourcena Referência de AWS CLI Comandos.

Python

SDK para 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
```

- Para obter detalhes da API, consulte a ListTagsForResourceReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte a ListTagsForResourcereferência da API AWS SDK for SAP ABAP.

> **Exemplo de disponibilidade**
>
> Não consegue encontrar o que precisa? Solicite um exemplo de código usando o link Fornecer feedback na barra lateral direita desta página.

AWS Console

1. Faça login na página Armazenamentos de dados no HealthLake console.
2. Escolha um datastore.

   A página Detalhes do datastore é aberta. Na seção Tags, todas as tags do datastore são listadas.

Desmarcando um armazenamento de dados HealthLake

Use `UntagResource` para remover uma tag de um armazenamento HealthLake de dados. Os menus a seguir fornecem um procedimento para o Console de gerenciamento da AWS e exemplos de código para AWS CLI AWS SDKs e. Para obter mais informações, consulte `UntagResource` na Referência de APIs do AWS HealthLake .

Para desmarcar um armazenamento de HealthLake dados

Escolha um menu com base na sua preferência de acesso AWS HealthLake a.

AWS CLI and SDKs

CLI

AWS CLI

Como remover tags de um datastore.

O exemplo de `untag-resource` a seguir mostra como remover tags de um datastore.

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

Este comando não produz saída.

- Para obter detalhes da API, consulte UntagResourcena Referência de AWS CLI Comandos.

Python

SDK para 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
```

- Para obter detalhes da API, consulte a UntagResourceReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte a UntagResourcereferência da API AWS SDK for SAP ABAP.

> **Exemplo de disponibilidade**
>
> Não consegue encontrar o que precisa? Solicite um exemplo de código usando o link Fornecer feedback na barra lateral direita desta página.

AWS Console

1. Faça login na página Armazenamentos de dados no HealthLake console.
2. Escolha um datastore.

   A página Detalhes do datastore é aberta.
3. Na seção Tags, escolha Gerenciar tags.

   A página Gerenciar tags é aberta.
4. Escolha Remover ao lado da tag que você quer removida.
5. Escolha Salvar.

Excluindo um armazenamento de HealthLake dados

Use `DeleteFHIRDatastore` para excluir um armazenamento HealthLake de dados. Os menus a seguir fornecem um procedimento para o Console de gerenciamento da AWS e exemplos de código para AWS CLI AWS SDKs e. Para obter mais informações, consulte `DeleteFHIRDatastore` na Referência de APIs do AWS HealthLake .

Para excluir um armazenamento HealthLake de dados

Escolha um menu com base na sua preferência de acesso AWS HealthLake a.

AWS CLI and SDKs

CLI

AWS CLI

Como excluir um datastore FHIR

O `delete-fhir-datastore` exemplo a seguir demonstra como excluir um armazenamento de dados e todo o seu conteúdo em AWS HealthLake.

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

Saída:

```
{
    "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)"
}
```

- Para obter detalhes da API, consulte Excluir FHIRDatastore na Referência de AWS CLI Comandos.

Python

SDK para 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
```

- Para obter detalhes da API, consulte a Referência da API Excluir FHIRDatastore no AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte Excluir FHIRDatastore no AWS SDK para referência da API SAP ABAP.

> **Exemplo de disponibilidade**
>
> Não consegue encontrar o que precisa? Solicite um exemplo de código usando o link Fornecer feedback na barra lateral direita desta página.

AWS Console

1. Faça login na página Armazenamentos de dados no HealthLake console.
2. Escolha um datastore.

   A página Detalhes do datastore é aberta.
3. Escolha Excluir.

   A página Excluir datastore é aberta.
4. Para confirmar a exclusão do datastore, insira o nome do datastore no campo de entrada de texto.
5. Escolha Excluir.

Gerenciando assinaturas FHIR em AWS HealthLake

AWS HealthLake suporta assinaturas FHIR, permitindo que você receba notificações em tempo real quando ocorrerem alterações específicas nos dados de saúde. Esse recurso implementa o modelo de assinatura baseado em tópicos do FHIR R5 Backport, fornecendo maior escalabilidade e flexibilidade em relação ao modelo de assinatura tradicional do FHIR R4.

Com as assinaturas do FHIR, você pode criar aplicativos de saúde orientados a eventos que respondem imediatamente às mudanças nos dados clínicos, permitindo intervenções oportunas, fluxos de trabalho automatizados e coordenação aprimorada do atendimento.

Tópicos

- Como funcionam as assinaturas do FHIR
- Componentes principais
- Práticas recomendadas
- O ciclo de vida da assinatura FHIR com AWS HealthLake
- Criando uma assinatura FHIR com AWS HealthLake
- Pesquisando assinaturas do FHIR com AWS HealthLake
- Filtrando notificações com AWS HealthLake

Como funcionam as assinaturas do FHIR

As assinaturas do FHIR HealthLake operam em um modelo baseado em tópicos, onde:

1. Crie tópicos para definir eventos: Crie tópicos de assinatura que especificam eventos que podem acionar notificações
2. Você se inscreve: Crie assinaturas para esses tópicos com critérios de filtragem específicos
3. HealthLake monitores: o serviço monitora continuamente os eventos que atendem aos seus critérios
4. Notificações entregues: eventos CWhen correspondentes ocorrem, HealthLake entrega notificações por meio do canal escolhido

Componentes principais

As assinaturas do FHIR consistem nos seguintes componentes.

Tópicos de assinatura

Os tópicos de assinatura são a base do sistema de notificação e definem:

- Eventos de gatilho: o que muda as notificações acionadas (por exemplo: criação de recursos, atualizações, exclusões)
- Filtros disponíveis: quais opções de filtragem estão disponíveis para os assinantes
- Conteúdo da notificação: quais dados estão incluídos nas notificações

A tabela a seguir lista os tipos de tópicos comuns.

Tipo de evento	Description	Casos de uso comuns
Criação de recursos	Acionado quando os recursos são criados	Novo registro de pacientes, nova observação registrada
Atualizações de recursos	Acionado quando os recursos são modificados	Mudanças de status, atualizaç ões clínicas
Exclusão de recursos	Acionado quando os recursos são excluídos	Rastreamento de auditoria e conformidade

Assinaturas

Uma Assinatura é sua solicitação para receber notificações de eventos específicos definidos por um Tópico de Assinatura. Cada assinatura inclui:

- Referência do tópico: especifica em qual tópico de assinatura você está se inscrevendo
- Filtros: critérios para selecionar quais eventos geram notificações
- Configuração do canal: onde e como as notificações devem ser entregues

- Preferências de carga útil: qual nível de detalhe deve ser incluído nas notificações

Canais de notificação

HealthLake suporta os seguintes canais de notificação:

Tipo de canal	Casos de uso	
EventBridge	Integrações corporativas, fluxos de trabalho sem servidor, orquestração entre serviços AWS	
REST Hook	Notificações diretas de endpoints, integração de sistemas de terceiros	

Cargas de notificação

Escolha o tipo de carga útil adequado com base em suas necessidades:

Tipo de carga	Description	Considerações sobre segurança
Somente ID	Contém somente identific adores de recursos	Exposição mínima ao PHI
Recurso completo	Contém conteúdo completo de recursos com um tamanho máximo de 256 KB. Se o tamanho for maior que 256 KB, ele voltará para somente ID	Contém PHI; verifique o manuseio seguro

Práticas recomendadas

Otimização de desempenho

- Use filtros focados: restrinja seus critérios para receber apenas notificações essenciais
- Escolha os tipos de carga útil apropriados: use cargas somente de ID quando possível para melhorar o desempenho
- Implemente receptores eficientes: garanta que os receptores de notificações processem mensagens rapidamente

Considerações sobre segurança

- Endpoints seguros: implemente a autenticação adequada para endpoints REST Hook
- Proteção de PHI: tenha cuidado com cargas de recursos completos, pois elas contêm PHI
- Controle de acesso: restrinja a criação de assinaturas somente a usuários autorizados

Excelência operacional

- Defina datas de término apropriadas: use datas de término para assinaturas temporárias
- Monitore o status da assinatura: verifique regularmente o status de suas assinaturas
- Implemente o tratamento de erros: projete seus aplicativos para lidar com falhas na entrega de notificações

O ciclo de vida da assinatura FHIR com AWS HealthLake

Siga estas etapas para entender o ciclo de vida da assinatura do FHIR:

1. Criar uma `SubscriptionTopic`
   - Crie um `SubscriptionTopic` com status `"unknown"`
2. Criar uma `Subscription`
   - Crie um `Subscription` com status `"requested"`
   - HealthLake valida a configuração `Subscription`
   - `Subscription`deve fazer referência a um tópico já existente (o tópico deve estar no status`unknown`,`draft`,`active`).

3. Ativação
   - Se válido, HealthLake atualiza o status de `Subscription` para `"active"`
   - Ao criar uma`Subscription`, se o tópico fornecido estiver em status`"unknown"`, HealthLake atualiza o status para `"active"` quando a Assinatura também estiver ativa
   - As assinaturas geralmente levam de 5 a 10 minutos para serem criadas com sucesso
   - Se `Subscription` a criação não for bem-sucedida, o status mudará para `error` onde você deve realizar uma operação DELETE e, em seguida, tentar criar novamente uma Assinatura. Você pode ver o `"error"` campo no recurso de assinatura para ver por que a assinatura não foi criada com sucesso.
4. Ingerir enquanto a assinatura é `active`
5. Enquanto `Subscription` é `active`
   - HealthLake monitores para eventos que atendem aos seus critérios
   - As notificações são enviadas para o endpoint configurado quando ocorrem correspondências
6. Tratamento de erros
   - HealthLake tenta novas tentativas por 14 dias e, em seguida, para de tentar novamente para esses eventos
7. Desativação
   - A `Subscription` pode ser desativado por:

     Definir uma data de término (desativação automática)

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

Excluindo o recurso `Subscription`

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

Criando uma assinatura FHIR com AWS HealthLake

O guia a seguir mostra como criar uma assinatura FHIR usando. AWS HealthLake

Para criar uma assinatura FHIR

1. Crie um `SubscriptionTopic`.

   Exemplo de recurso de tópico de assinatura:

```
{
  "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. Prepare seu endpoint de notificação (canal personalizado). As etapas a seguir são necessárias para garantir que o endpoint receba notificações:

   Ao usar o REST Hook

   - Confie `events.amazonaws.com` na sua política de chaves do KMS se estiver usando o armazenamento de dados CM_CMK.
   - Se estiver usando um armazenamento de dados CM_CMK, você deverá adicionar a `EventBridgeApiDestinations` tag à sua chave KMS com o valor de `true`
   - HealthLake usa OAuth para autenticar seu endpoint REST Hook. Portanto, ao criar uma assinatura de REST hook, você deve passar um ID de cliente, segredo do cliente e oAuth-endpoint-url no canal. _type.extension [*].

   Exemplo de política de chaves do KMS se estiver usando o armazenamento de dados CM_CMK:

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

   Ao usar EventBridge

   - Confie `events.amazonaws.com` na sua política de chaves do KMS se estiver usando o armazenamento de dados CM_CMK.

- Verifique se sua política de EventBridge recursos confia `healthlake.amazonaws.com` como principal de serviço.
- Ao usar CM_CMK e EventBridge for o endpoint, verifique se você está criptografando seu EventBridge barramento com a mesma chave KMS da chave KMS do armazenamento de dados.
- Verifique se seu EventBridge ônibus tem pelo menos uma regra que corresponda aos eventos gerados por HealthLake.

Exemplo de política de recursos para barramento de EventBridge canais:

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

Exemplo de padrão de eventos de EventBridge regra para receber eventos de: HealthLake

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

> **Note**
>
> HealthLake suporta 2 fontes:
>
> - “`healthlake`”: Somente para assinaturas.
> - “`aws.healthlake`”: Para receber eventos HealthLake de serviço.

Use “`healthlake`” como fonte ao criar uma regra para barramentos de eventos do FHIR Subscriptions.

3. Crie seu `Subscription`

   Envie um recurso de assinatura com:

   - Status: `"requested"`
   - Referência ao seu `SubscriptionTopic` ID escolhido
   - Critérios de filtro. Para obter mais informações, consulte Notificações de filtragem para filtros compatíveis.
   - Configuração de canal

Exemplo de cargas de assinatura

Os exemplos de código a seguir mostram como criar cargas de assinatura.

EventBridge

Assinatura com `event-bridge` canal e tipo `id-only` de carga útil.

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

Assinatura com `event-bridge` tipo de endpoint e `full-resource` payload.

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

Gancho de descanso

Assinatura com `rest-hook` tipo de endpoint e `id-only` payload.

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

Assinatura com `rest-hook` canal e tipo `full-resource` de carga útil.

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

Exemplo de cargas úteis de notificação

Enquanto uma assinatura está sendo criada, HealthLake verifique se a configuração da assinatura foi bem-sucedida enviando um pacote de handshake para o canal configurado. A carga útil a seguir é um exemplo de um pacote 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>"
          }
        }
      ]
    }
```

```
  }
}
```

Exemplo de pacote de notificação somente com identificação.

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

Exemplo de pacote de notificação com recursos completos.

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

```
  }
}
```

Controle de versão de eventos

HealthLake suporta o histórico FHIR por padrão.

Para saber qual versão do recurso você recebeu em seu pacote de notificação:

- recurso completo: como os pacotes completos de recursos incluem o recurso inteiro, a versão será incluída no `entry[*]` para cada recurso que estiver no pacote.
- id-only: os pacotes não incluirão nenhuma informação sobre recursos. HealthLakeinclui a versão que foi combinada e incluída no pacote por meio do `entry[0].notificationEvent[*].additionalContext` campo. Esse campo está no formato`<ResourceType>/<ResourceId>/_history/<Version Id>`. Para obter mais informações, consulte o campo additionalContext no exemplo de payload somente de identificação.

Detecção de duplicação de eventos

HealthLakeO recurso de assinatura FHIR da garante pelo menos uma entrega. Isso significa que você pode receber o mesmo evento várias vezes, no mesmo pacote ou em um pacote diferente. Para identificar duplicatas, HealthLake fornece uma identificação exclusiva para cada evento no pacote de notificação em. `entry[0].notificationEvent[*].id`

Esse ID é exclusivo para a versão específica do evento que foi correspondida e entregue. Por exemplo, se o mesmo Encontro for atualizado duas vezes e ambas corresponderem aos critérios do filtro, você receberá dois eventos separados com a mesma referência de Encontro. Eles terão o mesmo`notificationEvent[*].focus`, mas terão um único`notificationEvent[*].id`. Além disso, esses eventos podem ser enviados em pacotes separados ou dentro do mesmo pacote de notificação.

Pesquisando assinaturas do FHIR com AWS HealthLake

`Subscription`e `SubscriptionTopic` os recursos também podem ser pesquisados. HealthLake suporta todos os parâmetros de pesquisa comuns para assinatura e SubscriptionTopic recursos.

Além disso, oferecemos suporte a recursos adicionais de pesquisa por meio dos seguintes parâmetros:

Assinatura

Parâmetro de pesquisa	Description	Exemplo
contact	Pesquise no campo Assinatura.Contact na especificação principal do R4	`Subscription?contact=phone`
criteria	Pesquise no campo Assinatura.Critérios na especificação principal do R4	`Subscription?criteria=[baseUrl]/datastore/[datastoreId]/r4/SubscriptionTopic/[topicId]`
payload	Pesquise no campo subscription.channel.payload na especificação principal do R4	`Subscription?payload=application/fhir+json`
status	Pesquise no campo Subscription.Status na especificação principal do R4	`Subscription?status=error`
type	Pesquise no campo Subscription.channel.type na especificação principal do R4	`Subscription?topic=event-bridge`
url	Pesquise no campo subscription.channel.endpoint na especificação principal do R4	`Subscription?url=[Subscription.channel.endpoint]`
critérios de filtro	Pesquise no campo de extensão de critérios com o url "http://hl7. org/fhir/uv/subscriptions-backport/StructureDefinition/backport-filter-criteria” como uma string	`Subscription?filter-criteria=Encounter?`

Parâmetro de pesquisa	Description	Exemplo
canal personalizado	Pesquise no campo de extensão de tipo de canal personalizado com o url "http://hl7. org/fhir/uv/subscriptions-backport/StructureDefinition/backport-channel-type" como codificação Exemplo de carga útil de assinatura	`Subscription?custom-channel=[System]|[code]`
tipo de carga útil	Pesquise no campo de extensão do tipo de carga em que você especifica como a carga é formatada (ou) `id-only` `full-resource`	`Subscription?payload-type=full-resource`
tópico	Pesquise no campo Assinatura.Critérios, onde o tópico é adicionado	`Subscription?topic=[topicId]`

SubscriptionTopic

Parâmetro de pesquisa	Description	Exemplo
date	Pesquise no campo de data no SubscriptionTopic recurso	`SubscriptionTopic?date=[SubscriptionTopic.date]`
derived-or-self	Pesquise `derivedFrom` nos campos `url` ou no SubscriptionTopic recurso	`SubscriptionTopic?derived-or-self=[SubscriptionTopic.url | SubscriptionTopic.derivedFrom]`

Parâmetro de pesquisa	Description	Exemplo
Identifier	Pesquise no SubscriptoinTopic campo.identifier	`SubscriptionTopic?identifier=[SubscriptionTopic.identifier]`
recurso	Pesquise no campo SubscriptionTopic .resourceTrigger.resource	`SubscriptionTopic?resource=Encounter`
status	Pesquise o status do SubscriptionTopic	`SubscriptionTopic?status=unknown`
título	Pesquise no título por um SubscriptionTopic	`SubscriptionTopic?title=admission`
descrição do gatilho	Pesquise em SubscriptionTopic .resourceTrigger.description	`SubscriptionTopic.trigger-description=resource moving to state 'in-progress'`
url	Pesquise na URL o SubscriptionTopic	`SubscriptionTopic?url=[SubscriptionTopic.url]`
version	Pesquise a versão do SubscriptionTopic	`SubscriptionTopic?version=1`

Filtrando notificações com AWS HealthLake

Refine suas notificações usando parâmetros de pesquisa FHIR padrão em seus critérios. `Subscription`

Filtros compatíveis

A tabela a seguir mostra uma lista de critérios de filtro compatíveis HealthLake com assinaturas.

Tipos de parâmetros de pesquisa	Tipo de dados FHIR	Modifiers (Modifica dores)	Prefixos suportados
String	string HumanName Endereço	exato, contém o que falta	
Token	booleano código string Codificação CodeableConcept Identificador ContactPoint id	não, texto, ausente	
Número	integer decimal Int positivo Int não assinado	desaparecido	“eq”, “ne”, “gt”, “lt”, “ge”, “le”, “sa”, “eb”, “ap”
Data	date dateTime Instant Período Prazo		“eq”, “ne”, “gt”, “lt”, “ge”, “le”, “sa”, “eb”, “ap”

Tipos de parâmetros de pesquisa	Tipo de dados FHIR	Modifiers (Modificadores)	Prefixos suportados
Quantidade	Quantidade Dinheiro Intervalo SimpleQuantity		“eq”, “ne”, “gt”, “lt”, “ge”, “le”, “sa”, “eb”, “ap”
Referência	Referência	falta, identificador, tipo	
URI	uri url canônico uid oid	desaparecido	

Exemplos de filtros compatíveis

A tabela a seguir mostra exemplos de critérios de filtro compatíveis HealthLake com assinaturas:

Finalidade	Critérios de filtro	Description
Observações específicas do paciente	`Observation?patient=Patient/[id]&status=final`	Receba notificações quando as observações de um paciente específico forem finalizadas
Observações específicas do paciente	`Patient?birthdate=gt2021`	Receba notificações quando pacientes nascidos após 2021 forem registrados ou atualizados

Finalidade	Critérios de filtro	Description
Observações específicas do paciente	`Condition?code=http://snomed.info/sct|39065001`	Receba notificações de condições com códigos SNOMED específicos
Observações específicas do paciente	`Observation?code=http://loinc.org|8480-6&value-quantity=gt160`	Receba notificações para leituras de pressão arterial sistólica elevada

Importando dados FHIR com AWS HealthLake

Depois de criar um armazenamento de HealthLake dados, a próxima etapa é importar arquivos de um bucket do Amazon Simple Storage Service (S3). Você pode iniciar um trabalho de importação do FHIR usando o Console de gerenciamento da AWS, AWS CLI, ou. AWS SDKs Use AWS HealthLake ações nativas para iniciar, descrever e listar trabalhos de importação do FHIR.

> **Importante**
>
> HealthLake suporta a especificação FHIR R4 para troca de dados de assistência médica. Se necessário, você pode trabalhar com um AWS HealthLake parceiro para converter seus dados de saúde para o formato FHIR R4 antes da importação.

Ao iniciar um trabalho de importação do FHIR, você especifica um local de entrada do bucket do Amazon S3, um local de saída do bucket do Amazon S3 (para resultados do processamento do trabalho), uma função do IAM que concede acesso HealthLake aos seus buckets do Amazon S3 e uma chave de propriedade ou propriedade do cliente. AWS AWS Key Management Service Para obter mais informações, consulte Configurando permissões para trabalhos de importação.

> **Note**
>
> Você pode colocar trabalhos de importação em fila. As tarefas de importação assíncronas são processadas de forma FIFO (First In First Out). Você pode colocar trabalhos em fila da mesma forma que inicia a importação de trabalhos. Se um estiver em andamento, ele simplesmente entrará na fila. Você pode criar, ler, atualizar ou excluir recursos do FHIR enquanto um trabalho de importação está em andamento.

HealthLake gera um `manifest.json` arquivo para cada trabalho de importação do FHIR. O arquivo descreve os sucessos e as falhas de um trabalho de importação do FHIR. HealthLake envia o `manifest.json` arquivo para o bucket do Amazon S3 especificado ao iniciar um trabalho de importação do FHIR. Os arquivos de log são organizados em duas pastas, chamadas `SUCCESS` `FAILURE` e. Use o `manifest.json` arquivo como a primeira etapa para solucionar uma falha na tarefa de importação, pois ele fornece detalhes sobre cada arquivo.

```
{
```

```
    "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
}
```

Configurando o nível de validação para importações

Ao iniciar um trabalho de importação do FHIR, você pode, opcionalmente, especificar um `ValidationLevel` para aplicar a cada recurso. AWS HealthLake atualmente suporta os seguintes níveis de validação:

- `strict`: os recursos são validados de acordo com o elemento de perfil do recurso ou com a especificação R4 se nenhum perfil estiver presente. Esse é o nível de validação padrão para AWS HealthLake.
- `structure-only`: os recursos são validados em relação ao R4, ignorando quaisquer perfis referenciados.

- `minimal`: os recursos são validados minimamente, ignorando certas regras do R4. Os recursos que falharem nas verificações de estrutura exigidas search/analytics serão atualizados para incluir um aviso para auditoria.

Ao importar usando o nível de `minimal` validação, arquivos de log adicionais podem ser gerados em uma pasta chamada`SUCCESS_WITH_SEARCH_VALIDATION_FAILURES`. Os recursos dos arquivos de log dessa pasta foram ingeridos em seu armazenamento de dados, apesar de falharem nas verificações de validação relacionadas à pesquisa. Isso significa que certos aspectos do seu recurso FHIR eram inválidos de acordo com o FHIR, e campos malformados podem não ser pesquisáveis. Esses recursos terão um `extension` anexo descrevendo a referida falha.

Tópicos

- Iniciando um trabalho de importação de FHIR
- Obtendo propriedades de trabalho de importação do FHIR
- Listando trabalhos de importação do FHIR

Iniciando um trabalho de importação de FHIR

Use `StartFHIRImportJob` para iniciar um trabalho de importação do FHIR em um armazenamento de HealthLake dados. Os menus a seguir fornecem um procedimento para o Console de gerenciamento da AWS e exemplos de código para AWS CLI AWS SDKs e. Para obter mais informações, consulte `StartFHIRImportJob` na Referência de APIs do AWS HealthLake .

> **Importante**
>
> HealthLake suporta a especificação FHIR R4 para troca de dados de assistência médica. Se necessário, você pode trabalhar com um AWS HealthLake parceiro para converter seus dados de saúde para o formato FHIR R4 antes da importação.

Iniciar um trabalho de importação do FHIR

Escolha um menu com base na sua preferência de acesso AWS HealthLake a.

AWS CLI and SDKs

CLI

AWS CLI

Iniciar um trabalho de importação do FHIR

O `start-fhir-import-job` exemplo a seguir mostra como iniciar um trabalho de importação do FHIR usando o. AWS HealthLake

```
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)"
```

Saída:

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

- Para obter detalhes da API, consulte Start FHIRImport Job in AWS CLI Command Reference.

Python

SDK para 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
```

- Para obter detalhes da API, consulte Referência da API Start FHIRImport Job in AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte a referência da API Start FHIRImport Job in AWS SDK for SAP ABAP.

> **Exemplo de disponibilidade**
>
> Não consegue encontrar o que precisa? Solicite um exemplo de código usando o link Fornecer feedback na barra lateral direita desta página.

AWS Console

1. Faça login na página Armazenamentos de dados no HealthLake console.
2. Escolha um datastore.
3. Escolha Importar.

   A página Importar é aberta.

4. Na seção Dados de entrada, insira as seguintes informações:
   - Localização dos dados de entrada no Amazon S3
5. Na seção Importar arquivos de saída, insira as seguintes informações:
   - Localização dos arquivos de saída de importação no Amazon S3
   - Importar criptografia de arquivos de saída
6. Na seção Permissões de acesso, escolha Usar uma função de serviço do IAM existente e selecione a função no menu Nome da função do serviço ou escolha Criar uma função do IAM.
7. Escolha Importar dados.

> **Note**
>
> Durante a importação, escolha Copiar ID do trabalho no banner na parte superior da página. Você pode usar o `JobID`para solicitar a importação de propriedades do trabalho usando AWS CLI o. Para obter mais informações, consulte Obtendo propriedades de trabalho de importação do FHIR.

Obtendo propriedades de trabalho de importação do FHIR

Use `DescribeFHIRImportJob` para obter propriedades de trabalho de importação do FHIR. Os menus a seguir fornecem um procedimento para o Console de gerenciamento da AWS e exemplos de código para AWS CLI AWS SDKs e. Para obter mais informações, consulte `DescribeFHIRImportJob` na Referência de APIs do AWS HealthLake .

Para obter propriedades de trabalho de importação do FHIR

Escolha um menu com base na sua preferência de acesso AWS HealthLake a.

AWS CLI and SDKs

CLI

AWS CLI

Descrever um serviço de importação do FHIR

O `describe-fhir-import-job` exemplo a seguir mostra como aprender as propriedades de uma tarefa de importação FHIR usando. AWS HealthLake

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

Saída:

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

- Para obter detalhes da API, consulte Descreve FHIRImport Job in AWS CLI Command Reference.

Python

SDK para 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
```

- Para obter detalhes da API, consulte Referência da API Descreve FHIRImport Job in AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte a referência da API Describe FHIRImport Job in AWS SDK for SAP ABAP.

Exemplo de disponibilidade

Não consegue encontrar o que precisa? Solicite um exemplo de código usando o link Fornecer feedback na barra lateral direita desta página.

AWS Console

Note

As informações do trabalho de importação do FHIR não estão disponíveis no HealthLake console. Em vez disso, use AWS CLI with `DescribeFHIRImportJob` para solicitar propriedades do trabalho de importação, como `JobStatus`. Para obter mais informações, consulte o AWS CLI exemplo nesta página.

Listando trabalhos de importação do FHIR

Use `ListFHIRImportJobs` para listar trabalhos de importação FHIR para um armazenamento de HealthLake dados ativo. Os menus a seguir fornecem um procedimento para o Console de gerenciamento da AWS e exemplos de código para AWS CLI AWS SDKs e. Para obter mais informações, consulte `ListFHIRImportJobs` na Referência de APIs do AWS HealthLake .

Para listar trabalhos de importação do FHIR

Escolha um menu com base na sua preferência de acesso AWS HealthLake a.

AWS CLI and SDKs

CLI

AWS CLI

Listar todos os serviços de importação do FHIR

O exemplo `list-fhir-import-jobs` a seguir mostra como usar o comando para visualizar uma lista de todos os serviços de importação associados a uma conta.

```
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)
```

Saída:

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

- Para obter detalhes da API, consulte Listar FHIRImport trabalhos na Referência de AWS CLI Comandos.

Python

SDK para 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
```

- Para obter detalhes da API, consulte Referência da API Listar FHIRImport trabalhos no AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte Listar FHIRImport trabalhos no AWS SDK para referência da API SAP ABAP.

Exemplo de disponibilidade

Não consegue encontrar o que precisa? Solicite um exemplo de código usando o link Fornecer feedback na barra lateral direita desta página.

AWS Console

Note

As informações do trabalho de importação do FHIR não estão disponíveis no HealthLake console. Em vez disso, use AWS CLI com `ListFHIRImportJobs` para listar todos os trabalhos de importação do FHIR. Para obter mais informações, consulte o AWS CLI exemplo nesta página.

Gerenciando recursos do FHIR em AWS HealthLake

Use as interações da RESTful API FHIR R4 para gerenciar recursos do FHIR em um armazenamento de dados. HealthLake As seções a seguir descrevem todas as interações da RESTful API FHIR R4 HealthLake suportadas disponíveis para o gerenciamento de recursos do FHIR. Para obter informações sobre HealthLake os recursos do armazenamento de dados e quais partes da especificação FHIR são compatíveis, consulte. Declaração de capacidade do FHIR R4 para AWS HealthLake

> **Note**
>
> As interações FHIR listadas neste capítulo são construídas em conformidade com o padrão HL7 FHIR R4 para troca de dados de assistência médica. Por serem representações dos serviços da HL7 FHIR, não são oferecidos por meio AWS CLI de e. AWS SDKs Para obter mais informações, consulte RESTful API na documentação da API FHIR R4 RESTful .

A tabela a seguir lista as interações FHIR R4 suportadas pelo. AWS HealthLake Para obter informações sobre os tipos de recursos do FHIR suportados pelo HealthLake, consulte. Resource types

Interações FHIR R4 suportadas por AWS HealthLake

Interação	Description
Interações de todo o sistema	
`capabilities`	Obtenha uma declaração de capacidade para o sistema. Consulte Declaração de capacidade do FHIR R4 para AWS HealthLake.
`batch`	Atualize, crie ou exclua um conjunto de recursos em uma única interação . Consulte Agrupando recursos do FHIR.
Interações em nível de tipo	
`create`	Crie um novo recurso com uma ID atribuída ao servidor. Consulte Criando um recurso FHIR.

Interação	Description
`search`	Pesquise um tipo de recurso com base em alguns critérios de filtro. Consulte Pesquisando recursos do FHIR.
`history`	Recupere o histórico de alterações de um determinado tipo de recurso. Consulte Lendo o histórico de recursos do FHIR.
Interações em nível de instância	
`read`	Leia o estado atual de um recurso. Consulte Lendo um recurso do FHIR.
`history`	Leia o histórico de alterações de um recurso específico. Consulte Lendo o histórico de recursos do FHIR.
`vread`	Leia o estado de uma versão específica do recurso. Consulte Lendo o histórico de recursos FHIR específico da versão.
`update`	Atualize um recurso pelo ID (ou crie-o se for novo). Consulte Atualizando um recurso FHIR.
`delete`	Exclua um recurso. Consulte Excluindo um recurso FHIR.

Tópicos

- Criando um recurso FHIR
- Lendo um recurso do FHIR
- Lendo o histórico de recursos do FHIR
- Atualizando um recurso FHIR
- Modificando recursos com a operação PATCH
- Agrupando recursos do FHIR
- Excluindo um recurso FHIR
- Idempotência e concorrência

Criando um recurso FHIR

A `create` interação FHIR cria um novo recurso FHIR em um armazenamento de dados. HealthLake Para obter informações adicionais, consulte a `create`documentação da API FHIR R4 RESTful .

Para criar um recurso FHIR

1. Colecione HealthLake `region` e `datastoreId` valorize. Para obter mais informações, consulte Obter propriedades do datastore.
2. Determine o tipo de FHIR `Resource` a ser criado. Para obter mais informações, consulte Resource types.
3. Crie uma URL para a solicitação usando os valores coletados para HealthLake `region` `datastoreId` e. Inclua também o `Resource` tipo FHIR a ser criado. Para ver todo o caminho do URL no exemplo a seguir, role até o botão Copiar.

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

4. Crie um corpo JSON para a solicitação, especificando os dados FHIR para o novo recurso. Para fins deste procedimento, estamos usando um `Patient` recurso FHIR, então salve o arquivo como. `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. Envie a solicitação . A `create` interação FHIR usa uma `POST` solicitação com AWS Signature Version 4 ou SMART na autorização FHIR. Os exemplos a seguir criam um `Patient` recurso FHIR HealthLake usando o curl ou o console. HealthLake Para ver um exemplo inteiro, role até o botão Copiar.

   SigV4

   Autorização 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

   SMART no exemplo de autorização FHIR para o tipo de `IdentityProviderConfiguration`dados.

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

```
}
```

O chamador pode atribuir permissões na autorização lambda. Para obter mais informações, consulte OAuth escopos 2.0.

AWS Console

> **Note**
>
> O HealthLake console suporta somente a autorização AWS SigV4.

1. Faça login na página Executar consulta no HealthLake console.
2. Na seção Configurações da consulta, faça as seguintes seleções.
   - ID do armazenamento de dados — escolha um ID do armazenamento de dados para gerar uma sequência de caracteres de consulta.
   - Tipo de consulta — escolha`Create`.
   - Tipo de recurso — escolha o tipo de recurso FHIR a ser criado.
   - Corpo da solicitação — construa um corpo JSON para a solicitação, especificando os dados FHIR para o novo recurso.
3. Selecione Executar consulta.

Configurando o nível de validação para criação de recursos

Ao criar um recurso FHIR, você pode especificar opcionalmente um cabeçalho `x-amzn-healthlake-fhir-validation-level` HTTP para configurar um nível de validação para o recurso. AWS HealthLake atualmente suporta os seguintes níveis de validação:

- `strict`: os recursos são validados de acordo com o elemento de perfil do recurso ou com a especificação R4 se nenhum perfil estiver presente. Esse é o nível de validação padrão para AWS HealthLake.
- `structure-only`: os recursos são validados em relação ao R4, ignorando quaisquer perfis referenciados.

- `minimal`: os recursos são validados minimamente, ignorando certas regras do R4. Os recursos que falharem nas verificações de estrutura exigidas search/analytics serão atualizados para incluir um aviso para auditoria.

Os recursos criados com o nível mínimo de validação podem ser ingeridos em um Datastore, apesar da falha na validação necessária para a indexação de pesquisas. Nesse caso, os recursos serão atualizados para incluir uma extensão específica do Healthlake para documentar essas falhas:

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

Além disso, o seguinte cabeçalho de resposta HTTP será incluído com o valor “true”:

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

> **Note**
>
> Os dados ingeridos que estão malformados de acordo com a especificação R4 podem não ser pesquisáveis conforme o esperado se esses erros estiverem presentes.

Lendo um recurso do FHIR

A `read` interação FHIR lê o estado atual de um recurso em um armazenamento de HealthLake dados. Para obter informações adicionais, consulte a `read`documentação da API FHIR R4 RESTful .

Para ler um recurso do FHIR

1. Colecione HealthLake `region` e `datastoreId` valorize. Para obter mais informações, consulte Obter propriedades do datastore.
2. Determine o tipo de FHIR `Resource` para ler e coletar o valor associado`id`. Para obter mais informações, consulte Resource types.

3. Crie uma URL para a solicitação usando os valores coletados para HealthLake `region` `datastoreId` e. Inclua também o `Resource` tipo FHIR e seus associados. `id` Para ver todo o caminho do URL no exemplo a seguir, role até o botão Copiar.

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

4. Envie a solicitação . A `read` interação FHIR usa uma `GET` solicitação com AWS Signature Version 4 ou SMART na autorização FHIR. O `curl` exemplo a seguir lê o estado atual de um `Patient` recurso FHIR em. HealthLake Para ver o exemplo inteiro, role até o botão Copiar.

   SigV4

   Autorização 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

   SMART no exemplo de autorização FHIR para o tipo de `IdentityProviderConfiguration`dados.

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

O chamador pode atribuir permissões na autorização lambda. Para obter mais informações, consulte OAuth escopos 2.0.

AWS Console

1. Faça login na página Executar consulta no HealthLake console.
2. Na seção Configurações da consulta, faça as seguintes seleções.
   - ID do armazenamento de dados — escolha um ID do armazenamento de dados para gerar uma sequência de caracteres de consulta.
   - Tipo de consulta — escolha`Read`.
   - Tipo de recurso — escolha o tipo de recurso FHIR a ser lido.
   - ID do recurso — insira o ID do recurso FHIR.
3. Selecione Executar consulta.

Lendo o histórico de recursos do FHIR

A `history` interação FHIR recupera o histórico de um recurso FHIR específico em um armazenamento de dados. HealthLake Usando essa interação, você pode determinar como o conteúdo de um recurso FHIR mudou ao longo do tempo. Também é útil em coordenação com os registros de auditoria para ver o estado de um recurso antes e depois da modificação. As interações `create` do FHIR e `delete` resultam em uma versão histórica do recurso a ser salvo. `update` Para obter informações adicionais, consulte a `history`documentação da API FHIR R4 RESTful .

> **Note**
>
> Você pode optar por não receber tipos `history` específicos de recursos do FHIR. Para optar por não participar, crie um caso usando AWS Support Center Console. Para criar seu caso, faça login no seu Conta da AWS e escolha Criar caso.

Para ler o histórico de recursos do FHIR

1. Colecione HealthLake `region` e `datastoreId` valorize. Para obter mais informações, consulte Obter propriedades do datastore.
2. Determine o tipo de FHIR `Resource` para ler e coletar o valor associado`id`. Para obter mais informações, consulte Resource types.
3. Crie uma URL para a solicitação usando os valores coletados para HealthLake `region` `datastoreId` e. Inclua também o `Resource` tipo de FHIR, seus parâmetros de pesquisa associados `id` e opcionais. Para ver todo o caminho do URL no exemplo a seguir, role até o botão Copiar.

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

HealthLake parâmetros de pesquisa suportados para interação FHIR **`history`**

Parâmetro	Description
`_count : integer`	O número máximo de resultados de pesquisa em uma página. O servidor retornará o número solicitado ou o número máximo de resultados de pesquisa permitidos por padrão para o armazenamento de dados, o que for menor.
`_since : instant`	Inclua somente versões de recursos que foram criadas em ou após um determinado instante no tempo.
`_at : date(Time)`	Inclua somente versões de recursos que estavam atualizadas em algum momento durante o período especificado no valor de data e hora. Para obter mais informaçõ es, consulte a `date`documentação da RESTfulAPI HL7 FHIR.

4. Envie a solicitação . A `history` interação FHIR usa uma `GET` solicitação com AWS Signature Version 4 ou SMART na autorização FHIR. O `curl` exemplo a seguir usa o parâmetro de

`_count` pesquisa para retornar 100 resultados históricos de pesquisa por página para um `Patient` recurso FHIR em. HealthLake Para ver o exemplo inteiro, role até o botão Copiar.

SigV4

Autorização 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

SMART no exemplo de autorização FHIR para o tipo de `IdentityProviderConfiguration`dados.

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

O chamador pode atribuir permissões na autorização lambda. Para obter mais informações, consulte OAuth escopos 2.0.

O conteúdo de retorno de uma `history` interação está contido em um recurso FHIR`Bundle`, com o tipo definido como. `history` Ele contém o histórico de versões especificado, classificado com as versões mais antigas por último, e inclui recursos excluídos. Para obter mais informações, consulte a `Resource Bundle`documentação do FHIR R4.

Lendo o histórico de recursos FHIR específico da versão

A `vread` interação FHIR realiza uma leitura específica da versão de um recurso em um armazenamento de dados. HealthLake Usando essa interação, você pode visualizar o conteúdo de um recurso FHIR como era em um determinado momento no passado.

> **Note**
>
> Se você usar a `history` interação FHIR sem`vread`, HealthLake sempre retornará a versão mais recente dos metadados do recurso.

HealthLake declara que oferece suporte ao controle de versão `CapabilityStatement.rest.resource.versioning`para cada recurso suportado. Todos os armazenamentos de HealthLake dados incluem `Resource.meta.versionId` (`vid`) em todos os recursos.

Quando a `history` interação FHIR está ativada (por padrão para armazenamentos de dados criados após 25/10/2024 ou por solicitação para armazenamentos de dados mais antigos), a `Bundle` resposta inclui o como parte do `vid` elemento. `location` No exemplo a seguir, o é `vid` exibido como o número1. Para ver o exemplo completo, consulte Exemplo de bundle/bundle-response (JSON).

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

Para ler o histórico de recursos FHIR específico da versão

1. Colecione HealthLake `region` e `datastoreId` valorize. Para obter mais informações, consulte Obter propriedades do datastore.
2. Determine o `Resource` tipo de FHIR para ler e coletar valores `id` e `vid` associados. Para obter mais informações, consulte Resource types.
3. Crie uma URL para a solicitação usando os valores coletados para HealthLake e FHIR. Para ver todo o caminho do URL no exemplo a seguir, role até o botão Copiar.

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

4. Envie a solicitação . A `history` interação FHIR usa uma `GET` solicitação com AWS Signature Version 4 ou SMART na autorização FHIR. A `vread` interação a seguir retorna uma única instância com o conteúdo especificado para o `Patient` recurso FHIR para a versão dos metadados do recurso especificada pelo. `vid` Para ver todo o caminho do URL no exemplo a seguir, role até o botão Copiar.

   SigV4

   Autorização 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

   SMART no exemplo de autorização FHIR para o tipo de `IdentityProviderConfiguration`dados.

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

O chamador pode atribuir permissões na autorização lambda. Para obter mais informações, consulte OAuth escopos 2.0.

Atualizando um recurso FHIR

A `update` interação FHIR cria uma nova versão atual para um recurso existente ou cria uma versão inicial se nenhum recurso já existir para o determinado. `id` Para obter informações adicionais, consulte a `update`documentação da API FHIR R4 RESTful .

Para atualizar um recurso FHIR

1. Colecione HealthLake `region` e `datastoreId` valorize. Para obter mais informações, consulte Obter propriedades do datastore.
2. Determine o tipo de FHIR `Resource` para atualizar e coletar o valor associado`id`. Para obter mais informações, consulte Resource types.
3. Crie uma URL para a solicitação usando os valores coletados para HealthLake `region` `datastoreId` e. Inclua também o `Resource` tipo FHIR e seus associados. `id` Para ver todo o caminho do URL no exemplo a seguir, role até o botão Copiar.

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

4. Crie um `JSON` corpo para a solicitação, especificando as atualizações de dados do FHIR a serem feitas. Para fins desse procedimento, salve o arquivo como`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. Envie a solicitação . A `update` interação FHIR usa uma PUT solicitação com AWS Signature Version 4 ou SMART na autorização FHIR. O `curl` exemplo a seguir atualiza um `Patient` recurso em HealthLake. Para ver o exemplo inteiro, role até o botão Copiar.

   SigV4

   Autorização 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
   ```

   Sua solicitação retornará um código de status `200` HTTP se um recurso existente for atualizado ou um código de status `201` HTTP se um novo recurso for criado.

SMART on FHIR

SMART no exemplo de autorização FHIR para o tipo de `IdentityProviderConfiguration`dados.

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

O chamador pode atribuir permissões na autorização lambda. Para obter mais informações, consulte OAuth escopos 2.0.

AWS Console

1. Faça login na página Executar consulta no HealthLake console.

2. Na seção Configurações da consulta, faça as seguintes seleções.

- ID do armazenamento de dados — escolha um ID do armazenamento de dados para gerar uma sequência de caracteres de consulta.
- Tipo de consulta — escolha`Update (PUT)`.
- Tipo de recurso — escolha o tipo de recurso FHIR para atualizar ou criar.
- Corpo da solicitação — construa um corpo JSON para a solicitação, especificando os dados FHIR com os quais atualizar o recurso.

3. Selecione Executar consulta.

Atualização dos recursos do FHIR com base nas condições

A atualização condicional permite que você atualize um recurso existente com base em alguns critérios de pesquisa de identificação, em vez de usar o `id` FHIR lógico. Quando o servidor processa a atualização, ele realiza uma pesquisa usando seus recursos de pesquisa padrão para o tipo de recurso, com o objetivo de resolver uma única lógica `id` para a solicitação.

A ação que o servidor executa depende de quantas correspondências ele encontra:

- Sem correspondências, não **`id`** fornecido no corpo da solicitação: o servidor cria o recurso FHIR.
- Nenhuma correspondência, **`id`** fornecida e o recurso ainda não existe com o **`id`**: O servidor trata a interação como uma interação de atualização como criação.
- Nenhuma correspondência, **`id`** fornecida e já existe: o servidor rejeita a atualização com um `409 Conflict` erro.
- Uma correspondência, nenhum recurso **`id`** fornecido OU (recurso **`id`** fornecido e corresponde ao recurso encontrado): O servidor executa a atualização em relação ao recurso correspondente conforme descrito acima, onde, se o recurso foi atualizado, o servidor DEVE retornar `200 OK` a.
- One Match, recurso **`id`** fornecido, mas não corresponde ao recurso encontrado: o servidor retorna um `409 Conflict` erro indicando que a especificação do ID do cliente era um problema, preferencialmente com um `OperationOutcome`
- Várias correspondências: o servidor retorna um `412 Precondition Failed` erro indicando que os critérios do cliente não foram seletivos o suficiente, de preferência com um OperationOutcome

O exemplo a seguir atualiza um `Patient` recurso cujo nome é peter, data de nascimento é 1º de janeiro de 2000 e número de telefone 1234567890.

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

Configurando o nível de validação para atualizações de recursos

Ao atualizar um recurso FHIR, você pode especificar opcionalmente um cabeçalho `x-amzn-healthlake-fhir-validation-level` HTTP para configurar um nível de validação para o recurso. AWS HealthLake atualmente suporta os seguintes níveis de validação:

- `strict`: os recursos são validados de acordo com o elemento de perfil do recurso ou com a especificação R4 se nenhum perfil estiver presente. Esse é o nível de validação padrão para AWS HealthLake.
- `structure-only`: os recursos são validados em relação ao R4, ignorando quaisquer perfis referenciados.
- `minimal`: os recursos são validados minimamente, ignorando certas regras do R4. Os recursos que falharem nas verificações de estrutura exigidas search/analytics serão atualizados para incluir um aviso para auditoria.

Recursos atualizados com o nível mínimo de validação podem ser ingeridos em um Datastore, apesar da falha na validação necessária para a indexação de pesquisas. Nesse caso, os recursos serão atualizados para incluir uma extensão específica do Healthlake para documentar essas falhas:

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

Além disso, o seguinte cabeçalho de resposta HTTP será incluído com o valor “true”:

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

> **Note**
>
> Observe que os dados ingeridos que estão malformados de acordo com a especificação R4 podem não ser pesquisáveis conforme o esperado se esses erros estiverem presentes.

Modificando recursos com a operação PATCH

AWS HealthLake suporta a operação PATCH para recursos do FHIR, permitindo que você modifique recursos direcionando elementos específicos para adicionar, substituir ou excluir sem atualizar o recurso inteiro. Essa operação é particularmente útil quando você precisa:

- Faça atualizações direcionadas para grandes recursos
- Reduza o uso da largura de banda da rede
- Execute modificações atômicas em elementos de recursos específicos
- Minimize o risco de sobrescrever alterações simultâneas
- Atualize recursos como parte dos fluxos de trabalho em lotes e transações

Formatos PATCH suportados

AWS HealthLake suporta dois formatos PATCH padrão:

Correção JSON (RFC 6902)

Usa a sintaxe JSON Pointer para direcionar elementos de acordo com sua posição na estrutura do recurso.

Tipo de conteúdo: `application/json-patch+json`

FHIRPath Patch (especificação FHIR R4)

Usa FHIRPath expressões para direcionar elementos de acordo com seu conteúdo e relacionamentos, fornecendo uma abordagem nativa do FHIR para a aplicação de patches.

Tipo de conteúdo: `application/fhir+json`

Usage

Operações diretas de PATCH

A operação PATCH pode ser invocada diretamente nos recursos do FHIR usando o método PATCH HTTP:

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

PATCH em pacotes

As operações PATCH podem ser incluídas como entradas em pacotes FHIR do tipo `batch` ou `transaction`, permitindo que você combine operações de patch com outras interações FHIR (criar, ler, atualizar, excluir) em uma única solicitação.

- Pacotes de transações: todas as entradas são bem-sucedidas ou falham atomicamente
- Pacotes em lote: cada entrada é processada de forma independente

Formato de patch JSON

Operações com Suporte

Operation	Description
`add`	Adicione um novo valor ao recurso
`remove`	Remover um valor do recurso
`replace`	Substituir um valor existente no recurso
`move`	Remover um valor de um local e adicioná-lo a outro
`copy`	Copiar um valor de um local para outro
`test`	Teste se um valor no local de destino é igual a um valor especificado

Sintaxe do caminho

O JSON Patch usa a sintaxe JSON Pointer (RFC 6901):

Exemplo de caminho	Description
`/name/0/family`	Elemento familiar do primeiro nome
`/telecom/-`	Anexar à matriz de telecomunicações
`/active`	Elemento ativo no nível da raiz

Exemplo de caminho	Description
`/address/0/line/1`	Segunda linha do primeiro endereço

Exemplos

Solicitação direta de patch JSON com várias operações

```
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"
  }
]
```

Solicitação direta de patch JSON com operação única

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

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

Patch JSON no pacote

Use um recurso binário contendo a carga útil do patch JSON codificada em base64:

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

FHIRPath Formato do patch

Operações com Suporte

Operation	Description
`add`	Adicionar um novo elemento a um recurso
`insert`	Inserir um elemento em uma posição específica em uma lista
`delete`	Remover um elemento de um recurso
`replace`	Substituir o valor de um elemento existente
`move`	Reordenar elementos em uma lista

Sintaxe do caminho

FHIRPath O patch usa FHIRPath expressões, oferecendo suporte a:

- Acesso baseado em índices: `Patient.name[0]`
- Filtrando com **`where()`**: `Patient.name.where(use = 'official')`
- Lógica booleana: `Patient.telecom.where(system = 'phone' and use = 'work')`
- Funções de subconfiguração:`first()`, `last()`
- Verificações de existência:`exists()`, `count()`
- Navegação polimórfica: `0bservation.value`

Exemplos

Solicitação direta FHIRPath de patch

```
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 no pacote

Use um recurso de Parâmetros como recurso de entrada com `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"
    }
  }]
}
```

Cabeçalhos de solicitação

Cabeçalho	Obrigatório	Descrição
`Content-Type`	Sim	`application/json-patch+json` para JSON Patch ou `application/fhir+json` para FHIRPath Patch
`If-Match`	Não	Atualização condicional específica da versão usando ETag

Resposta da amostra

A operação retorna o recurso atualizado com as informações da nova versão:

```
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"
```

```
  }
}
```

Comportamento

A operação PATCH:

- Valida a sintaxe do patch de acordo com a especificação apropriada (RFC 6902 para JSON Patch, FHIR R4 para Patch) FHIRPath
- Aplica operações atomicamente - todas as operações são bem-sucedidas ou todas falham
- Atualiza o ID da versão do recurso e cria uma nova entrada no histórico
- Preserva o recurso original no histórico antes de aplicar as alterações
- Valida as restrições de recursos do FHIR após a aplicação de patches
- Suporta atualizações condicionais usando o cabeçalho If-Match com ETag

Tratamento de erros

A operação trata das seguintes condições de erro:

- 400 Solicitação incorreta: sintaxe de patch inválida (solicitação não conforme ou documento de patch malformado)
- 404 Não encontrado: recurso não encontrado (o ID especificado não existe)
- 409 Conflito: conflito de versão (atualizações simultâneas ou ID de versão não atual fornecida)
- 422 Entidade não processável: as operações de patch não podem ser aplicadas aos elementos de recursos especificados

Resumo dos recursos

Recurso	Patch JSON	FHIRPath Patch
Tipo de conteúdo	`application/json-patch+json`	`application/fhir+json`
Formato do caminho	Ponteiro JSON (RFC 6901)	FHIRPath expressões

Recurso	Patch JSON	FHIRPath Patch
API PATCH direta	Compatível	Compatível
Bundle Batch	Compatível (via binário)	Suportado (por meio de parâmetros)
Transação de pacote	Compatível (via binário)	Suportado (por meio de parâmetros)
Operações	adicionar, remover, substituir, mover, copiar, testar	adicionar, inserir, excluir, substituir, mover

Limitações

- Operações PATCH condicionais usando condições de pesquisa não são suportadas
- O patch JSON em pacotes deve usar recursos binários com conteúdo codificado em base64
- FHIRPath O patch em pacotes deve usar recursos de parâmetros

Recursos adicionais

Para obter mais informações sobre as operações PATCH, consulte:

- Documentação do FHIR R4 PATCH
- Especificação do patch FHIR R4 FHIRPath
- RFC 6902 - Correção JSON
- RFC 6901 - Ponteiro JSON

Agrupando recursos do FHIR

Um FHIR `Bundle` é um contêiner para uma coleção de recursos do FHIR em. AWS HealthLake AWS HealthLake suporta dois tipos de pacotes com diferentes comportamentos de processamento.

`Batch`os pacotes processam cada recurso de forma independente. Se um recurso falhar, os recursos restantes ainda poderão ser bem-sucedidos. Cada operação é processada individualmente e o processamento continua mesmo quando algumas operações falham. Use pacotes de lotes para

operações em massa em que o sucesso parcial seja aceitável, como o upload de vários registros de pacientes não relacionados.

`Transaction`os pacotes processam todos os recursos atomicamente como uma única unidade. Ou todas as operações de recursos são bem-sucedidas ou não AWS HealthLake comprometem nenhuma delas. Use pacotes de transações quando precisar de integridade referencial garantida em todos os recursos relacionados, como criar um paciente com observações e condições relacionadas em que todos os dados devem ser registrados juntos.

Diferenças entre pacotes de lotes e de transações

Recurso	Batch	TRANSACTION
Modelo de processamento	Cada operação é bem-sucedida ou falha de forma independente.	Todas as operações são bem-sucedidas ou falham como uma única unidade atômica.
Tratamento de falhas	O processamento continua mesmo se as operações individuais falharem.	O pacote inteiro falhará se uma única operação falhar.
Ordem de execução	A ordem de execução não é garantida.	As operações são processadas na ordem especificada.
Integridade referencial	Não aplicado em todas as operações.	Aplicado para recursos referenci ados localmente dentro do pacote.
Melhor usado para	Operações em massa em que o sucesso parcial é aceitável.	Recursos relacionados que devem ser criados ou atualizados juntos.

Você pode agrupar recursos de FHIR do mesmo tipo ou de tipos diferentes, e eles podem incluir uma combinação de operações de FHIR, como,`create`, e. `read` `update` `delete` `patch` Para obter informações adicionais, consulte Pacote de recursos na documentação do FHIR R4.

Veja a seguir exemplos de casos de uso para cada tipo de pacote.

Pacotes em lote

- Faça upload de vários registros de pacientes não relacionados de diferentes instalações durante a sincronização noturna de dados.

- Faça upload em massa de registros históricos de medicamentos em que alguns registros podem ter problemas de validação.
- Carregue dados de referência, como organizações e profissionais, em que falhas individuais não afetem outras entradas.

Pacotes de transações

- Crie um paciente com observações e condições relacionadas durante uma admissão no departamento de emergência, onde todos os dados devem ser registrados juntos.
- Atualize a lista de medicamentos do paciente e as informações relacionadas à alergia que devem permanecer consistentes.
- Registre um encontro completo com o paciente, observações, procedimentos e informações de cobrança como uma única unidade atômica.

> **Important**
>
> Os pacotes de lotes e transações usam a mesma estrutura `Bundle` de recursos. A única diferença é o valor do `type` campo.

O exemplo a seguir mostra um pacote de transações com vários tipos de recursos e operações.

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

Agrupando recursos do FHIR como entidades independentes

Para agrupar recursos do FHIR como entidades independentes

1. Colecione HealthLake `region` e `datastoreId` valorize. Para obter mais informações, consulte Obter propriedades do datastore.
2. Crie uma URL para a solicitação usando os valores coletados para HealthLake `region` `datastoreId` e. Não especifique um tipo de recurso FHIR no URL. Para ver todo o caminho do URL no exemplo a seguir, role até o botão Copiar.

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

3. Crie um corpo JSON para a solicitação, especificando cada verbo HTTP como parte dos elementos. `method` O exemplo a seguir usa uma interação `batch` de tipo com o `Bundle` recurso para criar novos `Patient` `Medication` recursos. Todas as seções obrigatórias são comentadas adequadamente. Para fins desse procedimento, salve o arquivo como`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. Envie a solicitação . O tipo de `Bundle` lote FHIR usa uma `POST` solicitação com AWS Signature Version 4 ou SMART na autorização FHIR. O exemplo de código a seguir usa a ferramenta de linha de `curl` comando para fins de demonstração.

   SigV4

   Autorização 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

   SMART no exemplo de autorização FHIR para o tipo de `IdentityProviderConfiguration`dados.

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

O chamador pode atribuir permissões na autorização lambda. Para obter mais informações, consulte OAuth escopos 2.0.

O servidor retorna uma resposta mostrando `Patient` os `Medication` recursos criados como resultado da solicitação do tipo de `Bundle` lote.

Condicional PUTs em pacotes

AWS HealthLake oferece suporte a atualizações condicionais em pacotes usando os seguintes parâmetros de consulta:

> **Note**
>
> PUTs As condicionais são suportadas somente em `batch` pacotes. `Transaction`pacotes não oferecem suporte a condicionais PUTs.

- `_id`(autônomo)
- `_id`em combinação com um dos seguintes:
  - `_tag`
  - `_createdAt`
  - `_lastUpdated`

Quando você usa condicional PUTs em pacotes, AWS HealthLake avalia os parâmetros de consulta em relação aos recursos existentes e age com base nos resultados da correspondência.

Comportamento de atualização condicional

Cenário	Status HTTP	Ação tomada
Recurso sem ID fornecido	201 Criado	Sempre cria um novo recurso.
Recurso com novo ID (sem correspondência)	201 Criado	Cria um novo recurso com o ID especificado.
Recurso com ID existente (correspondência única)	200 OK	Atualiza o recurso correspondente.
Recurso com ID existente (conflito detectado)	409 Conflito	Retorna um erro. Nenhuma alteração é feita.
Recurso com ID existente (incompatibilidade de ID)	400 solicitação inválida	Retorna um erro. Nenhuma alteração é feita.
Vários recursos correspondem às condições	412 Falha na pré-condição	Retorna um erro. Nenhuma alteração é feita.

No exemplo a seguir, pacote com uma atualização condicional, o `Patient` recurso com FHIR ID é `476` atualizado somente se a condição for atendida. `_lastUpdated=lt2025-04-20`

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

Agrupando recursos do FHIR como uma única entidade

Para agrupar recursos do FHIR como uma única entidade

1. Colecione HealthLake `region` e `datastoreId` valorize. Para obter mais informações, consulte Obter propriedades do datastore.
2. Crie uma URL para a solicitação usando os valores coletados para HealthLake `region` `datastoreId` e. Inclua o tipo de recurso FHIR `Bundle` como parte do URL. Para ver todo o caminho do URL no exemplo a seguir, role até o botão Copiar.

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

3. Crie um corpo JSON para a solicitação, especificando os recursos FHIR a serem agrupados. O exemplo a seguir agrupa dois `Patient` recursos em HealthLake. Para fins desse procedimento, salve o arquivo como`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. Envie a solicitação . O tipo de `Bundle` documento FHIR usa uma `POST` solicitação com o protocolo de AWS assinatura Signature Version 4. O exemplo de código a seguir usa a ferramenta de linha de `curl` comando para fins de demonstração.

   SigV4

   Autorização 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

SMART no exemplo de autorização FHIR para o tipo de `IdentityProviderConfiguration`dados.

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

O chamador pode atribuir permissões na autorização lambda. Para obter mais informações, consulte OAuth escopos 2.0.

O servidor retorna uma resposta mostrando dois `Patient` recursos criados como resultado da solicitação do tipo de `Bundle` documento.

Configurando o nível de validação para pacotes

Ao agrupar recursos do FHIR, você pode especificar opcionalmente um cabeçalho `x-amzn-healthlake-fhir-validation-level` HTTP para configurar um nível de validação para o recurso. Esse nível de validação será definido para todas as solicitações de criação e atualização dentro do pacote. AWS HealthLake atualmente suporta os seguintes níveis de validação:

- `strict`: os recursos são validados de acordo com o elemento de perfil do recurso ou com a especificação R4 se nenhum perfil estiver presente. Esse é o nível de validação padrão para AWS HealthLake.
- `structure-only`: os recursos são validados em relação ao R4, ignorando quaisquer perfis referenciados.
- `minimal`: os recursos são validados minimamente, ignorando certas regras do R4. Os recursos que falharem nas verificações de estrutura exigidas search/analytics serão atualizados para incluir um aviso para auditoria.

Recursos agrupados com o nível mínimo de validação podem ser ingeridos em um Datastore, apesar da falha na validação necessária para a indexação de pesquisas. Nesse caso, os recursos serão atualizados para incluir uma extensão específica do Healthlake para documentar essas falhas, e as entradas na resposta do pacote incluirão OperationOutcome os seguintes recursos:

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

```
            }
        }
    ]
}
```

Além disso, o seguinte cabeçalho de resposta HTTP será incluído com o valor “true”:

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

> **Note**
>
> Observe que os dados ingeridos que estão malformados de acordo com a especificação R4 podem não ser pesquisáveis conforme o esperado se esses erros estiverem presentes.

Suporte limitado para “mensagem” do tipo Bundle

HealthLake fornece suporte limitado para o tipo de pacote FHIR `message` por meio de um processo de conversão interno. Esse suporte foi projetado para cenários em que os pacotes de mensagens não podem ser reformatados na origem, como a ingestão de feeds ADT (Admissão, Descarga, Transferência) de sistemas hospitalares antigos.

> **Warning**
>
> Esse recurso requer uma lista explícita de permissões de AWS contas e não impõe a semântica da mensagem FHIR R4 ou a integridade referencial. Entre em contato com o AWS Support para solicitar a ativação da sua conta antes de usar pacotes de mensagens.

Principais diferenças em relação ao processamento padrão de mensagens

- Pacotes de mensagens (especificação FHIR): a primeira entrada deve ser uma `MessageHeader` que faça referência a outros recursos. Os recursos não têm `request` objetos individuais e o MessageHeader evento determina as ações de processamento.
- HealthLake Processamento: converte pacotes de mensagens em pacotes em lotes atribuindo automaticamente operações PUT a cada entrada de recurso. Os recursos são processados de forma independente, sem impor a semântica da mensagem ou a integridade referencial.

Limitações importantes

- As regras de processamento específicas de mensagens do FHIR R4 não são aplicadas
- Sem integridade transacional em todos os recursos
- As referências entre recursos não são validadas
- Requer uma lista explícita de permissões da conta

Exemplo de estrutura de pacote de mensagens

```
{
              "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**
>
> Cada recurso é armazenado de forma independente, como se fosse enviado por meio de operações PUT individuais. Se a semântica completa das mensagens FHIR ou a validação

da integridade referencial forem necessárias, pré-processe os pacotes de mensagens ou implemente a validação em nível de aplicativo antes do envio.

Transações assíncronas de pacotes

AWS HealthLake suporta o `Bundle` tipo assíncrono `transaction` que permite enviar transações com até 500 recursos. Quando você envia uma transação assíncrona, ela é colocada em HealthLake fila para processamento e retorna imediatamente uma URL de pesquisa. Você pode usar esse URL para verificar o status e recuperar a resposta. Isso segue o padrão de pacote assíncrono FHIR.

Quando usar transações assíncronas

- Você precisa enviar mais de 100 recursos (limite síncrono) em uma única transação.
- Você quer evitar o bloqueio do seu aplicativo enquanto aguarda a conclusão do processamento da transação.
- Você precisa processar grandes volumes de recursos relacionados com melhor rendimento.

Important

Os resultados da pesquisa estão disponíveis por 90 dias após a conclusão da transação. Após esse período de 90 dias, o URL da pesquisa não retorna mais os resultados. Projete sua integração para recuperar e armazenar resultados nessa janela.

Note

O `Bundle` tipo síncrono `transaction` continua suportando até 100 recursos e é o modo de processamento padrão. Se você enviar um `Bundle` tipo `transaction` com mais de 100 recursos sem o `Prefer: respond-async` cabeçalho, HealthLake retornará um `422 Unprocessable Entity` erro. Pacotes com tipo não `batch` são compatíveis com processamento assíncrono — somente `Bundle` tipos `transaction` podem ser enviados de forma assíncrona (com até 500 operações).

> **Note**
>
> PATCHoperações e condicionais não PUTs são suportadas em transações assíncronas de pacotes.

Enviando uma transação assíncrona

Para enviar uma transação assíncrona, envie uma POST solicitação ao endpoint do armazenamento de dados com o cabeçalho. `Prefer: respond-async` O pacote deve ter um tipo`transaction`. Pacotes com tipo não `batch` são suportados para processamento assíncrono de pacotes.

HealthLake faz as validações iniciais do pacote no momento do envio. Se a validação for bem-sucedida, HealthLake retornará HTTP 202 Accepted com um cabeçalho de `content-location` resposta que contém o URL da pesquisa.

Para enviar um tipo **`Bundle`** assíncrono **`transaction`**

1. Envie uma POST solicitação para o endpoint do armazenamento de HealthLake dados.

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

2. Construa um corpo JSON para a solicitação com o tipo de pacote. `transaction` Para fins desse procedimento, salve o arquivo como`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. Envie a solicitação com o `Prefer: respond-async` cabeçalho. O tipo de `Bundle` transação FHIR usa uma `POST` solicitação com AWS Signature Version 4 ou SMART na autorização FHIR. O exemplo de código a seguir usa a ferramenta de linha de `curl` comando para fins de demonstração.

   SigV4

   Autorização 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

SMART no exemplo de autorização FHIR para o tipo de `IdentityProviderConfiguration`dados.

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

O chamador pode atribuir permissões na autorização lambda. Para obter mais informações, consulte OAuth escopos 2.0.

4. Em caso de envio bem-sucedido, o servidor retorna HTTP 202 Accepted. O cabeçalho da `content-location` resposta contém o URL da pesquisa. O corpo da resposta é um `OperationOutcome` recurso.

```
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"
            ]
        }
    ]
}
```

Pesquisa sobre o status da transação

Depois de enviar uma transação assíncrona, use o URL de pesquisa do cabeçalho da `content-location` resposta para verificar o status da transação. Envie uma `GET` solicitação para o URL da pesquisa.

> **Note**
>
> Para SMART em armazenamentos de dados habilitados para FHIR, o token de autorização deve incluir `read` permissões sobre o tipo de `Transaction` recurso para pesquisar o status da transação. Para obter mais informações sobre SMART em escopos FHIR, consulte. SMART em escopos FHIR OAuth 2.0 suportados por HealthLake

Envie uma `GET` solicitação para o URL da pesquisa. O exemplo a seguir usa a ferramenta de linha de `curl` comando.

SigV4

Autorização 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

SMART na autorização do FHIR. O token de autorização deve incluir `read` permissões sobre o tipo de `Transaction` recurso.

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

A tabela a seguir descreve as respostas possíveis.

Códigos de resposta da pesquisa

Status HTTP	Significado	Corpo da resposta
202 aceito	A transação está na fila	`OperationOutcome` com diagnósticos “ENVIADOS”
202 aceito	A transação está sendo processada	`OperationOutcome` com diagnóstico “IN_PROGRESS”
200 OK	Transação concluída com sucesso	`Bundle`com tipo `transaction-response`
4xx/5xx	Falha na transação	`OperationOutcome` com detalhes do erro

Os exemplos a seguir mostram cada tipo de resposta.

Transação em fila (202)

```
{
```

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

Processamento de transações (202)

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

Transação concluída (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"
            }
        }
    ]
}
```

Falha na transação (4xx/5xx)

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

Pedido de processamento

Pacotes assíncronos do tipo `transaction` são colocados em fila, mas não são processados em uma ordem estrita de envio. HealthLake otimiza o processamento com base na capacidade disponível e na carga do sistema.

> **Important**
>
> Não dependa de transações sendo processadas na ordem em que foram enviadas. Por exemplo, se você enviar a Transação A às 10h e a Transação B às 10h01, a Transação B poderá ser concluída antes da Transação A. Crie sua inscrição para:
>
> - Lide com out-of-order a conclusão.
> - Use o URL da pesquisa para rastrear cada transação de forma independente.
> - Implemente o sequenciamento em nível de aplicativo se a ordem for importante para seu caso de uso.

Cotas e limitação

As cotas e limites de taxa a seguir se aplicam às transações assíncronas.

Cotas de transações assíncronas

Quota	Valor	Ajustável
Máximo de operações por transação assíncrona	500	Não
Máximo de transações pendentes por armazenamento de dados	500	Sim

- As transações assíncronas compartilham os mesmos limites de taxa de API definidos em. Cotas de serviço
- A pesquisa do status da transação compartilha os mesmos limites de taxa de API das operações read (`GET`) nos recursos do FHIR.
- Se o limite de transações pendentes for atingido, os envios subsequentes retornarão um erro até que as transações existentes sejam concluídas.

Tratamento de erros

Para um pacote de “transação”, todos os recursos FHIR contidos no pacote são processados como uma operação atômica. Todos os recursos da operação devem ser bem-sucedidos, ou nenhuma operação no pacote será processada.

Os erros se enquadram em duas categorias: erros de envio que HealthLake retornam de forma síncrona e erros de processamento que você recupera por meio de enquetes.

Erros de envio

HealthLake valida o pacote no momento do envio e retorna os erros de forma síncrona antes que a transação seja colocada na fila. Os erros de envio incluem erros inválidos de validação de recursos do FHIR, tipos de recursos não suportados, exceder o limite de 500 operações e usar o `Prefer: respond-async` cabeçalho com pacotes de lotes. Se o limite de transações pendentes para o armazenamento de dados tiver sido atingido, HealthLake retorna `ThrottlingException` a. Quando ocorre um erro de envio, a transação não será colocada na fila.

Erros de processamento

Os erros de processamento ocorrem depois que a transação é colocada na fila e retornada por meio do URL da pesquisa. Isso inclui conflitos de transação, em que outra operação modificou um recurso que faz parte da transação, e erros do servidor durante o processamento. Quando ocorre um erro de processamento, nenhuma mutação de recurso é feita para os recursos na transação. O URL da pesquisa retornará um `OperationOutcome` com os detalhes do erro.

Excluindo um recurso FHIR

A `delete` interação FHIR remove um recurso FHIR existente de um armazenamento de dados. HealthLake Para obter informações adicionais, consulte a `delete`documentação da API FHIR R4 RESTful .

Para excluir um recurso FHIR

1. Colecione HealthLake `region` e `datastoreId` valorize. Para obter mais informações, consulte Obter propriedades do datastore.
2. Determine o tipo de FHIR `Resource` para excluir e coletar o valor associado`id`. Para obter mais informações, consulte Resource types.
3. Crie uma URL para a solicitação usando os valores coletados para HealthLake `region` `datastoreId` e. Inclua também o `Resource` tipo FHIR e seus associados. `id` Para ver todo o caminho do URL no exemplo a seguir, role até o botão Copiar.

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

4. Envie a solicitação . A `delete` interação FHIR usa uma `DELETE` solicitação com AWS Signature Version 4 ou SMART na autorização FHIR. O `curl` exemplo a seguir remove um `Patient` recurso FHIR existente de um armazenamento de HealthLake dados. Para ver o exemplo inteiro, role até o botão Copiar.

   SigV4

   Autorização 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'
```

O servidor retorna um código de status 204 HTTP confirmando que o recurso foi removido do armazenamento de HealthLake dados. Se uma solicitação de exclusão falhar, você receberá um código de status HTTP em 400 série indicando por que a solicitação falhou.

SMART on FHIR

SMART no exemplo de autorização FHIR para o tipo de IdentityProviderConfigurationdados.

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

O chamador pode atribuir permissões na autorização lambda. Para obter mais informações, consulte OAuth escopos 2.0.

AWS Console

1. Faça login na página Executar consulta no HealthLake console.

2. Na seção Configurações da consulta, faça as seguintes seleções.

- ID do armazenamento de dados — escolha um ID do armazenamento de dados para gerar uma sequência de caracteres de consulta.
- Tipo de consulta — escolha`Delete`.
- Tipo de recurso — escolha o tipo de recurso FHIR a ser excluído.
- ID do recurso — insira o ID do recurso FHIR.

3. Selecione Executar consulta.

Excluindo recursos do FHIR com base nas condições

A exclusão condicional é particularmente útil quando você não sabe o ID específico do recurso FHIR, mas tem outras informações de identificação sobre o recurso que deseja excluir.

A exclusão condicional permite que você exclua um recurso existente com base nos critérios de pesquisa e não no ID lógico do FHIR. Quando o servidor processa a solicitação de exclusão, ele executa uma pesquisa usando recursos de pesquisa padrão para o tipo de recurso para resolver uma única ID lógica para a solicitação.

Como funciona a exclusão condicional

A ação do servidor depende de quantas correspondências ele encontra:

1. Sem resultados: o servidor tenta uma exclusão normal e responde adequadamente (404 Não encontrado para recurso inexistente, 204 Sem conteúdo para recurso já excluído)
2. Uma partida: o servidor executa uma exclusão normal no recurso correspondente
3. Várias correspondências: retorna um erro 412 Precondition Failed indicando que os critérios do cliente não foram seletivos o suficiente

Cenários de resposta

AWS HealthLake processa operações de exclusão condicional com os seguintes padrões de resposta:

Operações bem-sucedidas

- Quando seus critérios de pesquisa identificam com sucesso um único recurso ativo, o sistema retorna 204 Sem conteúdo após concluir a exclusão, assim como as operações de exclusão padrão.

Exclusão condicional baseada em ID

Ao realizar a exclusão condicional `id` com base em parâmetros adicionais (`createdAttag`, ou`_lastUpdated`):

- 204 Sem conteúdo: o recurso já foi excluído
- 404 Não encontrado: o recurso não existe
- 409 Conflito: o ID corresponde, mas outros parâmetros não coincidem

Non-ID-Based Exclusão condicional

Quando não `id` é fornecido ou ao usar parâmetros diferentes de `createdAttag`, ou`_lastUpdated`:

- 404 Não encontrado: Nenhuma correspondência encontrada

Situações de conflito

Vários cenários resultam em 412 respostas de falha na condição prévia:

- Vários recursos correspondem aos seus critérios de pesquisa (critérios não específicos o suficiente)
- Conflitos de versão ao usar ETag cabeçalhos com `If-Match`
- Atualizações de recursos que ocorrem entre as operações de pesquisa e exclusão

Exemplo de exclusão condicional bem-sucedida

O exemplo a seguir exclui um recurso do paciente com base em critérios específicos:

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

Essa solicitação exclui um recurso do paciente em que:

- O nome é “peter”
- A data de nascimento é 1º de janeiro de 2000
- O número de telefone é 1234567890

Práticas recomendadas

1. Use critérios de pesquisa específicos para evitar várias correspondências e evitar 412 erros.
2. Considere ETag cabeçalhos para controle de versão quando necessário para lidar com modificações simultâneas.
3. Lide com as respostas de erro de forma adequada:
   - Para 404: Refine seus critérios de pesquisa
   - Para 412: torne os critérios mais específicos ou resolva conflitos de versão
4. Prepare-se para conflitos temporais em ambientes de alta simultaneidade, nos quais os recursos podem ser modificados entre as operações de pesquisa e exclusão.

Idempotência e concorrência

Chaves de idempotência

AWS HealthLake suporta chaves de idempotência para `POST` operações de FHIR, fornecendo um mecanismo robusto para garantir a integridade dos dados durante a criação de recursos. Ao incluir um UUID exclusivo como chave de idempotência no cabeçalho da solicitação, os aplicativos de saúde podem garantir que cada recurso FHIR seja criado exatamente uma vez, mesmo em cenários que envolvam instabilidade de rede ou novas tentativas automáticas.

Esse recurso é particularmente crucial para sistemas de saúde em que registros médicos duplicados podem ter consequências graves. Quando uma solicitação é recebida com a mesma chave de idempotência de uma solicitação anterior, HealthLake retornará o recurso original em vez de criar uma duplicata. Por exemplo, isso pode ocorrer durante um ciclo de repetição ou devido a pipelines de solicitação redundantes. O uso da chave de idempotência permite manter HealthLake a consistência dos dados e, ao mesmo tempo, fornecer uma experiência perfeita para aplicativos clientes que lidam com problemas de conectividade intermitentes.

Implementação

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

Cenários de resposta

Primeira solicitação (201 criada)

- Novo recurso criado com sucesso
- A resposta inclui ID do recurso

Solicitação duplicada (409 Conflito)

- Mesma chave de idempotência detectada
- Recurso original devolvido
- Nenhum novo recurso criado

Solicitação inválida (400 solicitações inválidas)

- UUID malformado
- Campos obrigatórios ausentes

Práticas recomendadas

- Gere UUID exclusivo para cada criação de novo recurso
- Armazene as chaves de idempotência para a lógica de repetição
- Use um formato de chave consistente: recomendado o UUID v4
- Implemente em aplicativos clientes que lidam com a criação de recursos

> **Note**
>
> Esse recurso é particularmente valioso para sistemas de saúde que exigem uma precisão rigorosa dos dados e evitam registros médicos duplicados.

ETag na AWS HealthLake

AWS HealthLake usa ETags para controle otimista de simultaneidade em recursos do FHIR, fornecendo um mecanismo confiável para gerenciar modificações simultâneas e manter a consistência dos dados. An ETag é um identificador exclusivo que representa uma versão específica de um recurso, funcionando como um sistema de controle de versão por meio de cabeçalhos HTTP. Ao ler ou modificar recursos, os aplicativos podem ser usados ETags para evitar substituições não intencionais e garantir a integridade dos dados, especialmente em cenários com possíveis atualizações simultâneas.

Exemplo de implementação

```
// 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
```

Cenários de resposta

Operação bem-sucedida (200 OK ou 204 sem conteúdo)

- ETag corresponde à versão atual
- A operação prossegue conforme o planejado

Conflito de versão (falha na pré-condição 412)

- ETag não corresponde à versão atual
- Atualização rejeitada para evitar perda de dados

Práticas recomendadas

- Incluir ETags em todas as operações de atualização e exclusão
- Implemente a lógica de repetição para lidar com conflitos de versão
- Use If-None-Match: * para create-if-not-exists cenários
- Sempre verifique o ETag frescor antes das modificações

Esse sistema de controle de concorrência é essencial para manter a integridade dos dados de saúde, especialmente em ambientes com vários usuários ou sistemas acessando e modificando os mesmos recursos.

Pesquisando recursos do FHIR em AWS HealthLake

Use a `search`interação FHIR para pesquisar um conjunto de recursos FHIR em um armazenamento de HealthLake dados com base em alguns critérios de filtro. A `search` interação pode ser realizada usando uma `POST` solicitação `GET` ou. Para pesquisas que envolvam informações de identificação pessoal (PII) ou informações de saúde protegidas (PHI), é recomendável usar `POST` solicitações, pois PII e PHI são adicionadas como parte do corpo da solicitação e criptografadas em trânsito.

> **Note**
>
> A `search` interação FHIR descrita neste capítulo é construída em conformidade com o padrão HL7 FHIR R4 para troca de dados de assistência médica. Por ser uma representação de um serviço HL7 FHIR, não é oferecido por meio AWS CLI de e. AWS SDKs Para obter mais informações, consulte a `search`documentação da API FHIR R4 RESTful .

HealthLake suporta um subconjunto de parâmetros de pesquisa FHIR R4. Para obter mais informações, consulte Parâmetros de pesquisa FHIR R4 para HealthLake.

Tópicos

- Pesquisando recursos do FHIR com GET
- Pesquisando recursos do FHIR com POST
- Níveis de consistência de pesquisa do FHIR

Pesquisando recursos do FHIR com GET

Você pode usar `GET` solicitações para pesquisar um armazenamento HealthLake de dados. Ao usar`GET`, HealthLake suporta o fornecimento de parâmetros de pesquisa como parte do URL, mas não como parte do corpo de uma solicitação. Para obter mais informações, consulte Parâmetros de pesquisa FHIR R4 para HealthLake.

> **Important**
>
> Para pesquisas que envolvem informações de identificação pessoal (PII) ou informações de saúde protegidas (PHI), as melhores práticas de segurança exigem o uso de `POST`

solicitações, pois PII e PHI são adicionadas como parte do corpo da solicitação e criptografadas em trânsito. Para obter mais informações, consulte Pesquisando recursos do FHIR com POST.

O procedimento a seguir é seguido por exemplos usados `GET` para pesquisar um armazenamento HealthLake de dados.

Para pesquisar um armazenamento HealthLake de dados com **`GET`**

1. Colecione HealthLake `region` e `datastoreId` valorize. Para obter mais informações, consulte Obter propriedades do datastore.
2. Determine o tipo de recurso FHIR para pesquisar e coletar o valor associado`id`. Para obter mais informações, consulte Resource types.
3. Crie uma URL para a solicitação usando os valores coletados para HealthLake `region` `datastoreId` e. Inclua também o `Resource` tipo de FHIR e os parâmetros de pesquisa compatíveis. Para ver todo o caminho do URL no exemplo a seguir, role até o botão Copiar.

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

4. Envie a `GET` solicitação com AWS Signature Version 4 ou SMART na autorização FHIR. O `curl` exemplo a seguir retorna o número total de `Patient` recursos em um armazenamento HealthLake de dados. Para ver o exemplo inteiro, role até o botão Copiar.

   SigV4

   Autorização 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

SMART no exemplo de autorização FHIR para o tipo de `IdentityProviderConfiguration` dados.

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

O chamador pode atribuir permissões na autorização lambda. Para obter mais informações, consulte OAuth escopos 2.0.

AWS Console

> **Note**
>
> O HealthLake console suporta somente a autorização SigV4. A autorização SMART on FHIR é suportada por meio de e. AWS CLI AWS SDKs

1. Faça login na página Executar consulta no HealthLake console.

2. Na seção Configurações da consulta, faça as seguintes seleções.

- ID do armazenamento de dados — escolha um ID do armazenamento de dados para gerar uma sequência de caracteres de consulta.
- Tipo de consulta — escolha`Search with GET`.
- Tipo de recurso — escolha o tipo de recurso FHIR para pesquisar.
- Parâmetros de pesquisa — Selecione um parâmetro de pesquisa ou uma combinação de parâmetros de pesquisa para focar sua consulta em registros específicos.

3. Selecione Executar consulta.

Exemplos: pesquisar com GET

As guias a seguir fornecem exemplos para pesquisar tipos específicos de recursos do FHIR com. `GET` Os exemplos mostram como especificar parâmetros de pesquisa na solicitação URLs.

> **Note**
>
> O HealthLake console suporta somente a autorização SigV4. A autorização SMART on FHIR é suportada por meio de e. AWS CLI AWS SDKs
> HealthLake suporta um subconjunto de parâmetros de pesquisa FHIR R4. Para obter mais informações, consulte Parâmetros de pesquisa.

Patient (age)

Embora a idade não seja um tipo de recurso definido no FHIR, ela é capturada como um elemento no tipo de `Patient`recurso. Use o exemplo a seguir para fazer uma solicitação de pesquisa `GET` baseada em tipos de `Patient`recursos usando o elemento BirthDate e o `eq` comparador de pesquisa para pesquisar indivíduos nascidos no ano de 1997.

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

Condition

Use o exemplo a seguir para fazer uma `GET` solicitação sobre o tipo de `Condition`recurso. A pesquisa encontra condições em seu armazenamento de HealthLake dados que contêm o código médico SNOMED`72892002`, que se traduz em. `Normal pregnancy`

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

DocumentationReference

O exemplo a seguir mostra como criar uma GET solicitação sobre o tipo de DocumentReferencerecurso para Patient (s) com diagnóstico de estreptococos e para quem também foi prescrita amoxicilina.

```
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

Use o exemplo a seguir para fazer uma GET solicitação sobre o tipo de Locationrecurso. A pesquisa a seguir encontra locais em seu armazenamento de HealthLake dados que contêm o nome da cidade Boston como parte do endereço.

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

Observation

Use o exemplo a seguir para fazer uma solicitação de pesquisa GET baseada no tipo de Observationrecurso. Essa pesquisa usa o parâmetro de value-concept pesquisa para procurar o código médico266919005, que se traduz em. Never smoker

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

Pesquisando recursos do FHIR com POST

Você pode usar a searchinteração do FHIR com POST solicitações para pesquisar um armazenamento de HealthLake dados. Ao usarPOST, HealthLake oferece suporte a parâmetros de pesquisa no URL ou no corpo da solicitação, mas você não pode usar os dois em uma única solicitação.

> **Importante**
>
> Para pesquisas que envolvem informações de identificação pessoal (PII) ou informações de saúde protegidas (PHI), as melhores práticas de segurança exigem o uso de `POST` solicitações, pois PII e PHI são adicionadas como parte do corpo da solicitação e criptografadas em trânsito.

O procedimento a seguir é seguido por exemplos usando a `search` interação do FHIR R4 com `POST` para pesquisar um HealthLake armazenamento de dados. Os exemplos mostram como especificar parâmetros de pesquisa no corpo da solicitação JSON.

Para pesquisar um armazenamento HealthLake de dados com **`POST`**

1. Colecione HealthLake `region` e `datastoreId` valorize. Para obter mais informações, consulte Obter propriedades do datastore.
2. Determine o tipo de recurso FHIR para pesquisar e coletar o valor associado`id`. Para obter mais informações, consulte Resource types.
3. Crie uma URL para a solicitação usando os valores coletados para HealthLake `region` `datastoreId` e. Inclua também o `Resource` tipo e `_search` a interação do FHIR. Para ver todo o caminho do URL no exemplo a seguir, role até o botão Copiar.

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

4. Crie um corpo JSON para a solicitação, especificando os dados FHIR a serem pesquisados. Para fins deste procedimento, você buscará `Observation` recursos para descobrir pacientes que nunca fumaram. Para especificar o status do código médico`Never smoker`, defina `value-concept=266919005` no corpo da solicitação JSON. Salve o arquivo como `search-observation.json`.

   ```
   value-concept=266919005
   ```

5. Envie a solicitação . A `search` interação FHIR usa a `GET` solicitação com AWS Signature Version 4 ou SMART na autorização FHIR.

> **Note**
>
> Ao fazer uma POST solicitação com parâmetros de pesquisa no corpo da solicitação, use `Content-Type: application/x-www-form-urlencoded` como parte do cabeçalho.

O `curl` exemplo a seguir faz uma solicitação de pesquisa baseada em POST sobre o tipo `Observation` de recurso. A solicitação usa o parâmetro de `value-concept`pesquisa para procurar o código médico `266919005` que indica o valor`Never smoker`. Para ver o exemplo inteiro, role até o botão Copiar.

SigV4

Autorização 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

SMART no exemplo de autorização FHIR para o tipo de `IdentityProviderConfiguration`dados.

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

O chamador pode atribuir permissões na autorização lambda. Para obter mais informações, consulte OAuth escopos 2.0.

Exemplos: pesquisar com POST

As guias a seguir fornecem exemplos para pesquisar tipos específicos de recursos do FHIR com. `POST` Os exemplos mostram como especificar uma solicitação no URLs.

> **Note**
>
> O HealthLake console suporta somente a autorização SigV4. A autorização SMART on FHIR é suportada por meio de e. AWS CLI AWS SDKs
> HealthLake suporta um subconjunto de parâmetros de pesquisa FHIR R4. Para obter mais informações, consulte Parâmetros de pesquisa.

Patient (age)

Embora a idade não seja um tipo de recurso definido no FHIR, ela é capturada como um elemento no tipo de `Patient`recurso. Use o exemplo a seguir para fazer uma solicitação de pesquisa `POST` baseada no tipo de `Patient` recurso. O exemplo de pesquisa a seguir usa o `eq` comparador de pesquisa para pesquisar indivíduos nascidos em 1997.

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

Para especificar o ano de 1997 na pesquisa, adicione o seguinte elemento ao corpo da solicitação.

```
birthdate=eq1997
```

Condition

Use o seguinte para fazer uma POST solicitação sobre o tipo de `Condition` recurso. Essa pesquisa encontra locais em seu armazenamento HealthLake de dados que contêm o código médico72892002.

Você precisa especificar uma URL e um corpo da solicitação. Aqui está um exemplo de URL de solicitação.

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

Para especificar o código médico que você deseja pesquisar, adicione o seguinte elemento JSON ao corpo da solicitação.

```
code=72892002
```

DocumentReference

Para ver os resultados HealthLake do processamento integrado de linguagem natural (NLP) ao fazer uma POST solicitação sobre o tipo de `DocumentReference` recurso, formate uma solicitação da seguinte forma.

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

Para especificar os parâmetros `DocumentReference` de pesquisa a serem referenciados, consulteTipos de parâmetros de pesquisa. A sequência de caracteres de consulta a seguir usa vários parâmetros de pesquisa para pesquisar as operações da API Amazon Comprehend Medical usadas para gerar os resultados integrados de PNL.

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

Location

Use o exemplo a seguir para fazer uma `POST` solicitação sobre o tipo de `Location` recurso. A pesquisa encontra locais em seu armazenamento de HealthLake dados que contêm o nome da cidade Boston como parte do endereço.

Você deve especificar uma URL e um corpo da solicitação. Aqui está um exemplo de URL de solicitação.

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

Para especificar `Boston` na pesquisa, adicione o seguinte elemento ao corpo da solicitação:

```
address=Boston
```

Observation

Use o exemplo a seguir para fazer uma solicitação de pesquisa `POST` baseada no tipo de `Observation` recurso. A pesquisa usa o parâmetro de `value-concept` pesquisa para procurar o código médico, `266919005` isso indica`Never smoker`.

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

Para especificar o status`Never smoker`, defina `value-concept=266919005` no corpo do JSON.

```
value-concept=266919005
```

Níveis de consistência de pesquisa do FHIR

O índice HealthLake de pesquisa da AWS opera em um modelo de consistência eventual para `GET` e `POST` com operações de PESQUISA. Com consistência eventual, se houver uma atualização pendente do índice de pesquisa para um recurso, os resultados da pesquisa excluirão a versão N-1 do recurso até que a atualização do índice seja concluída.

A AWS HealthLake agora inclui a capacidade de selecionar como o modelo de consistência se comportará para recursos atualizados. Os desenvolvedores podem incluir “Consistência eventual”,

o comportamento padrão descrito acima, ou “Consistência forte”. O Strong Consistency permitirá que a versão N-1 do recurso para recursos com atualizações pendentes do índice de pesquisa seja incluída nos resultados da pesquisa. Isso pode ser usado para cenários de casos de uso em que todos os recursos são necessários no resultado, mesmo quando a atualização do índice de pesquisa ainda não foi concluída. Os clientes podem controlar esse comportamento usando o cabeçalho da `x-amz-fhir-history-consistency-level` solicitação.

Níveis de consistência

Consistência forte

Configure `x-amz-fhir-history-consistency-level: strong` para retornar todos os registros correspondentes, incluindo aqueles com atualizações pendentes do índice de pesquisa. Use essa opção quando precisar pesquisar recursos imediatamente após as atualizações.

Consistência eventual

`x-amz-fhir-history-consistency-level: eventual`Defina para retornar somente registros que tenham concluído as atualizações do índice de pesquisa. Esse é o comportamento padrão se nenhum nível de consistência for especificado.

Exemplo de uso

1. Ao atualizar um recurso:

```
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. Pesquisa subseqüente:

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

Práticas recomendadas

- Use uma consistência forte quando precisar pesquisar imediatamente recursos atualizados recentemente
- Use a consistência eventual para consultas gerais em que a visibilidade imediata não é essencial
- Considere a compensação entre visibilidade imediata e impacto potencial no desempenho

> **Note**
>
> A configuração do nível de consistência afeta a rapidez com que os recursos atualizados aparecem nos resultados da pesquisa, mas não afeta o armazenamento real dos recursos. Definir o `x-amz-fhir-history-consistency-level` cabeçalho opcional como 'forte' dobra o consumo da capacidade de gravação por recurso.
> Esse recurso é aplicável somente para armazenamentos de dados com histórico de versões ativado (todos os armazenamentos de dados criados após 25 de outubro de 2024 o têm ativado por padrão).

Exportando dados FHIR com AWS HealthLake

Use AWS HealthLake ações nativas para iniciar, descrever e listar trabalhos de exportação do FHIR. Você pode colocar trabalhos de exportação em fila. As tarefas de exportação assíncrona são processadas de forma FIFO (First In First Out). Você pode colocar trabalhos em fila da mesma forma que inicia trabalhos de exportação. Se um estiver em andamento, ele simplesmente entrará na fila. Você pode criar, ler, atualizar ou excluir recursos do FHIR enquanto um trabalho de exportação está em andamento.

> **Note**
>
> Você também pode exportar dados FHIR de um armazenamento de HealthLake dados usando a operação FHIR R4. `$export` Para obter mais informações, consulte Exportação de HealthLake dados com o FHIR `$export`.

Tópicos

- Iniciando um trabalho de exportação do FHIR
- Obtendo propriedades de trabalho de exportação do FHIR
- Listando trabalhos de exportação do FHIR

Iniciando um trabalho de exportação do FHIR

Use `StartFHIRExportJob` para iniciar um trabalho de exportação FHIR a partir de um armazenamento de HealthLake dados. Os menus a seguir fornecem um procedimento para o Console de gerenciamento da AWS e exemplos de código para AWS CLI AWS SDKs e. Para obter mais informações, consulte `StartFHIRExportJob` na Referência de APIs do AWS HealthLake .

> **Observação**
>
> HealthLake suporta a especificação FHIR R4 para troca de dados de assistência médica. Portanto, todos os dados de saúde são exportados no formato FHIR R4.

Iniciar um serviço de exportação do FHIR

Escolha um menu com base na sua preferência de acesso AWS HealthLake a.

AWS CLI and SDKs

CLI

AWS CLI

Iniciar um serviço de exportação do FHIR

O `start-fhir-export-job` exemplo a seguir mostra como iniciar um trabalho de exportação do FHIR usando o. AWS HealthLake

```
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)
```

Saída:

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

- Para obter detalhes da API, consulte Start FHIRExport Job in AWS CLI Command Reference.

Python

SDK para 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
```

- Para obter detalhes da API, consulte Referência da API Start FHIRExport Job in AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte a referência da API Start FHIRExport Job in AWS SDK for SAP ABAP.

> **Exemplo de disponibilidade**
>
> Não consegue encontrar o que precisa? Solicite um exemplo de código usando o link Fornecer feedback na barra lateral direita desta página.

AWS Console

1. Faça login na página Armazenamentos de dados no HealthLake console.
2. Escolha um datastore.
3. Escolha Exportar.

   A página Exportar é aberta.
4. Na seção Dados de saída, insira as seguintes informações:

   - Localização dos dados de saída no Amazon S3
   - Criptografia de saída

5. Na seção Permissões de acesso, escolha Usar uma função de serviço do IAM existente e selecione a função no menu Nome da função ou escolha Criar uma função do IAM.
6. Escolha Exportar dados.

> **Note**
>
> Durante a exportação, escolha Copiar ID do trabalho no banner na parte superior da página. Você pode usar o `JobID`para solicitar propriedades do trabalho de exportação usando AWS CLI o. Para obter mais informações, consulte Obtendo propriedades de trabalho de exportação do FHIR.

Obtendo propriedades de trabalho de exportação do FHIR

Use `DescribeFHIRExportJob` para obter propriedades da tarefa de exportação de um armazenamento HealthLake de dados. Os menus a seguir fornecem um procedimento para o Console de gerenciamento da AWS e exemplos de código para AWS CLI AWS SDKs e. Para obter mais informações, consulte `DescribeFHIRExportJob` na Referência de APIs do AWS HealthLake .

> **Observação**
>
> HealthLake suporta a especificação FHIR R4 para troca de dados de assistência médica. Portanto, todos os dados de saúde são exportados no formato FHIR R4.

Descrever um serviço de exportação do FHIR

Escolha um menu com base na sua preferência de acesso AWS HealthLake a.

AWS CLI and SDKs

CLI

AWS CLI

Descrever um serviço de exportação do FHIR

O `describe-fhir-export-job` exemplo a seguir mostra como encontrar as propriedades de um trabalho de exportação FHIR em. AWS HealthLake

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

Saída:

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

- Para obter detalhes da API, consulte Descreve FHIRExport Job in AWS CLI Command Reference.

Python

SDK para 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
```

- Para obter detalhes da API, consulte Referência da API Descreve FHIRExport Job in AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte a referência da API Describe FHIRExport Job in AWS SDK for SAP ABAP.

> **Exemplo de disponibilidade**
>
> Não consegue encontrar o que precisa? Solicite um exemplo de código usando o link Fornecer feedback na barra lateral direita desta página.

AWS Console

> **Note**
>
> As informações do trabalho de exportação do FHIR não estão disponíveis no HealthLake console. Em vez disso, use AWS CLI with `DescribeFHIRExportJob` para solicitar propriedades do trabalho de exportação, como `JobStatus`. Para obter mais informações, consulte o AWS CLI exemplo nesta página.

Listando trabalhos de exportação do FHIR

Use `ListFHIRExportJobs` para listar trabalhos de exportação FHIR para um armazenamento de HealthLake dados. Os menus a seguir fornecem um procedimento para o Console de gerenciamento da AWS e exemplos de código para AWS CLI AWS SDKs e. Para obter mais informações, consulte `ListFHIRExportJobs` na Referência de APIs do AWS HealthLake .

> **Observação**
>
> HealthLake suporta a especificação FHIR R4 para troca de dados de assistência médica. Portanto, todos os dados de saúde são exportados no formato FHIR R4.

Para listar trabalhos de exportação do FHIR

Escolha um menu com base na sua preferência de acesso AWS HealthLake a.

AWS CLI and SDKs

CLI

AWS CLI

Listar todos os serviços de exportação do FHIR

O exemplo `list-fhir-export-jobs` a seguir mostra como usar o comando para exibir uma lista de serviços de exportação associados a uma conta.

```
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)
```

Saída:

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

```
}
```

- Para obter detalhes da API, consulte Listar FHIRExport trabalhos na Referência de AWS CLI Comandos.

Python

SDK para 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
```

- Para obter detalhes da API, consulte Referência da API Listar FHIRExport trabalhos no AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte Listar FHIRExport trabalhos no AWS SDK para referência da API SAP ABAP.

Exemplo de disponibilidade

Não consegue encontrar o que precisa? Solicite um exemplo de código usando o link Fornecer feedback na barra lateral direita desta página.

AWS Console

Note

As informações do trabalho de exportação do FHIR não estão disponíveis no HealthLake console. Em vez disso, use AWS CLI com `ListFHIRExportJobs` para listar todos os trabalhos de exportação do FHIR. Para obter mais informações, consulte o AWS CLI exemplo nesta página.

Exemplos de código para HealthLake usar AWS SDKs

Os exemplos de código a seguir mostram como usar HealthLake com um kit AWS de desenvolvimento de software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando HealthLake com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Exemplos de código

- Exemplos básicos de HealthLake uso AWS SDKs
  - Ações para HealthLake usar AWS SDKs
    - Use CreateFHIRDatastore com um AWS SDK ou CLI
    - Use DeleteFHIRDatastore com um AWS SDK ou CLI
    - Use DescribeFHIRDatastore com um AWS SDK ou CLI
    - Use DescribeFHIRExportJob com um AWS SDK ou CLI
    - Use DescribeFHIRImportJob com um AWS SDK ou CLI
    - Use ListFHIRDatastores com um AWS SDK ou CLI
    - Use ListFHIRExportJobs com um AWS SDK ou CLI
    - Use ListFHIRImportJobs com um AWS SDK ou CLI
    - Use ListTagsForResource com um AWS SDK ou CLI
    - Use StartFHIRExportJob com um AWS SDK ou CLI
    - Use StartFHIRImportJob com um AWS SDK ou CLI
    - Use TagResource com um AWS SDK ou CLI
    - Use UntagResource com um AWS SDK ou CLI

Exemplos básicos de HealthLake uso AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS HealthLake with AWS SDKs

Exemplos

- Ações para HealthLake usar AWS SDKs
  - Use CreateFHIRDatastore com um AWS SDK ou CLI
  - Use DeleteFHIRDatastore com um AWS SDK ou CLI
  - Use DescribeFHIRDatastore com um AWS SDK ou CLI
  - Use DescribeFHIRExportJob com um AWS SDK ou CLI
  - Use DescribeFHIRImportJob com um AWS SDK ou CLI
  - Use ListFHIRDatastores com um AWS SDK ou CLI
  - Use ListFHIRExportJobs com um AWS SDK ou CLI
  - Use ListFHIRImportJobs com um AWS SDK ou CLI
  - Use ListTagsForResource com um AWS SDK ou CLI
  - Use StartFHIRExportJob com um AWS SDK ou CLI
  - Use StartFHIRImportJob com um AWS SDK ou CLI
  - Use TagResource com um AWS SDK ou CLI
  - Use UntagResource com um AWS SDK ou CLI

Ações para HealthLake usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar HealthLake ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS HealthLake.

Exemplos

- Use CreateFHIRDatastore com um AWS SDK ou CLI
- Use DeleteFHIRDatastore com um AWS SDK ou CLI
- Use DescribeFHIRDatastore com um AWS SDK ou CLI
- Use DescribeFHIRExportJob com um AWS SDK ou CLI
- Use DescribeFHIRImportJob com um AWS SDK ou CLI
- Use ListFHIRDatastores com um AWS SDK ou CLI
- Use ListFHIRExportJobs com um AWS SDK ou CLI

- Use ListFHIRImportJobs com um AWS SDK ou CLI
- Use ListTagsForResource com um AWS SDK ou CLI
- Use StartFHIRExportJob com um AWS SDK ou CLI
- Use StartFHIRImportJob com um AWS SDK ou CLI
- Use TagResource com um AWS SDK ou CLI
- Use UntagResource com um AWS SDK ou CLI

Use **`CreateFHIRDatastore`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateFHIRDatastore`.

CLI

AWS CLI

Exemplo 1: Criar um armazenamento de dados habilitado para SIGv4 HealthLake

O `create-fhir-datastore` exemplo a seguir demonstra como criar um novo armazenamento de dados em AWS HealthLake.

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

Saída:

```
{
    "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)"
}
```

Exemplo 2: Crie um SMART no armazenamento de dados habilitado para FHIR HealthLake

O `create-fhir-datastore` exemplo a seguir demonstra como criar um novo SMART no armazenamento de dados habilitado para FHIR em. 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
```

Conteúdo 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\"]}"
}
```

Saída:

```
{
    "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)"
}
```

- Para obter detalhes da API, consulte Criar FHIRDatastore na Referência de AWS CLI Comandos.

Python

SDK para 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
```

O código a seguir mostra um exemplo de parâmetros para um SMART no armazenamento de dados habilitado para 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
            )
```

- Para obter detalhes da API, consulte Referência da API Create FHIRDatastore in AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte Criar FHIRDatastore no AWS SDK para referência da API SAP ABAP.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando HealthLake com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`DeleteFHIRDatastore`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteFHIRDatastore`.

CLI

AWS CLI

Como excluir um datastore FHIR

O `delete-fhir-datastore` exemplo a seguir demonstra como excluir um armazenamento de dados e todo o seu conteúdo em AWS HealthLake.

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

Saída:

```
{
    "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)"
}
```

- Para obter detalhes da API, consulte Excluir FHIRDatastore na Referência de AWS CLI Comandos.

Python

SDK para 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
```

- Para obter detalhes da API, consulte a Referência da API Excluir FHIRDatastore no AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte Excluir FHIRDatastore no AWS SDK para referência da API SAP ABAP.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando HealthLake com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`DescribeFHIRDatastore`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeFHIRDatastore`.

CLI

AWS CLI

Como descrever um datastore FHIR

O `describe-fhir-datastore` exemplo a seguir demonstra como encontrar as propriedades de um armazenamento de dados em AWS HealthLake.

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

Saída:

```
{
    "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
```

```
        }
    }
}
```

- Para obter detalhes da API, consulte Descrever FHIRDatastore na Referência de AWS CLI Comandos.

Python

SDK para 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
```

- Para obter detalhes da API, consulte a Referência da API Descrever FHIRDatastore no AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte Descrição FHIRDatastore na referência da API ABAP do AWS SDK for SAP.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando HealthLake com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`DescribeFHIRExportJob`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeFHIRExportJob`.

CLI

AWS CLI

Descrever um serviço de exportação do FHIR

O `describe-fhir-export-job` exemplo a seguir mostra como encontrar as propriedades de um trabalho de exportação FHIR em. AWS HealthLake

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

Saída:

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

- Para obter detalhes da API, consulte Descreve FHIRExport Job in AWS CLI Command Reference.

Python

SDK para 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
```

- Para obter detalhes da API, consulte Referência da API Descreve FHIRExport Job in AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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.
```

- Para obter detalhes da API, consulte a referência da API Describe FHIRExport Job in AWS SDK for SAP ABAP.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando HealthLake com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`DescribeFHIRImportJob`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeFHIRImportJob`.

CLI

AWS CLI

Descrever um serviço de importação do FHIR

O `describe-fhir-import-job` exemplo a seguir mostra como aprender as propriedades de uma tarefa de importação do FHIR usando. AWS HealthLake

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

Saída:

```
{
```

```
    "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)"
    }
}
```

- Para obter detalhes da API, consulte Descreve FHIRImport Job in AWS CLI Command Reference.

Python

SDK para 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
```

- Para obter detalhes da API, consulte Referência da API Descreve FHIRImport Job in AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte a referência da API Describe FHIRImport Job in AWS SDK for SAP ABAP.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando HealthLake com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`ListFHIRDatastores`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListFHIRDatastores`.

CLI

AWS CLI

Como listar datastores FHIR

O `list-fhir-datastores` exemplo a seguir mostra como usar o comando e como os usuários podem filtrar os resultados com base no status do armazenamento de dados em AWS HealthLake.

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

Saída:

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

- Para obter detalhes da API, consulte Lista FHIRDatastores na Referência de AWS CLI Comandos.

Python

SDK para 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
```

- Para obter detalhes da API, consulte Referência da API List FHIRDatastores in AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte Lista FHIRDatastores no AWS SDK para referência da API SAP ABAP.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando HealthLake com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`ListFHIRExportJobs`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListFHIRExportJobs`.

CLI

AWS CLI

Listar todos os serviços de exportação do FHIR

O exemplo `list-fhir-export-jobs` a seguir mostra como usar o comando para exibir uma lista de serviços de exportação associados a uma conta.

```
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)
```

Saída:

```
{
    "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)"
            }
```

```
        }
    ]
}
```

- Para obter detalhes da API, consulte Listar FHIRExport trabalhos na Referência de AWS CLI Comandos.

Python

SDK para 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
```

- Para obter detalhes da API, consulte Referência da API Listar FHIRExport trabalhos no AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte Listar FHIRExport trabalhos no AWS SDK para referência da API SAP ABAP.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando HealthLake com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`ListFHIRImportJobs`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListFHIRImportJobs`.

CLI

AWS CLI

Listar todos os serviços de importação do FHIR

O exemplo `list-fhir-import-jobs` a seguir mostra como usar o comando para visualizar uma lista de todos os serviços de importação associados a uma conta.

```
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)
```

Saída:

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

- Para obter detalhes da API, consulte Listar FHIRImport trabalhos na Referência de AWS CLI Comandos.

Python

SDK para 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
```

- Para obter detalhes da API, consulte Referência da API Listar FHIRImport trabalhos no AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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.
```

- Para obter detalhes da API, consulte Listar FHIRImport trabalhos no AWS SDK para referência da API SAP ABAP.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando HealthLake com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`ListTagsForResource`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListTagsForResource`.

CLI

AWS CLI

Como listar tags para um datastore

O exemplo de `list-tags-for-resource` a seguir lista as tags associadas ao datastore especificado.

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

Saída:

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

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

- Para obter detalhes da API, consulte ListTagsForResourceem Referência de AWS CLI Comandos.

Python

SDK para 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
```

- Para obter detalhes da API, consulte a ListTagsForResourceReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte a ListTagsForResourcereferência da API AWS SDK for SAP ABAP.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando HealthLake com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`StartFHIRExportJob`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `StartFHIRExportJob`.

CLI

AWS CLI

Iniciar um serviço de exportação do FHIR

O `start-fhir-export-job` exemplo a seguir mostra como iniciar um trabalho de exportação FHIR usando o. AWS HealthLake

```
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)
```

Saída:

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

- Para obter detalhes da API, consulte Start FHIRExport Job in AWS CLI Command Reference.

Python

SDK para 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
```

- Para obter detalhes da API, consulte Referência da API Start FHIRExport Job in AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte a referência da API Start FHIRExport Job in AWS SDK for SAP ABAP.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando HealthLake com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`StartFHIRImportJob`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `StartFHIRImportJob`.

CLI

AWS CLI

Iniciar um trabalho de importação do FHIR

O `start-fhir-import-job` exemplo a seguir mostra como iniciar um trabalho de importação FHIR usando o. AWS HealthLake

```
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)"
```

Saída:

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

- Para obter detalhes da API, consulte Start FHIRImport Job in AWS CLI Command Reference.

Python

SDK para 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
```

- Para obter detalhes da API, consulte Referência da API Start FHIRImport Job in AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte a referência da API Start FHIRImport Job in AWS SDK for SAP ABAP.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando HealthLake com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`TagResource`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `TagResource`.

CLI

AWS CLI

Como adicionar uma tag ao datastore

O exemplo de `tag-resource` a seguir mostra como adicionar uma tag a um datastore.

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

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

Este comando não produz saída.

.

- Para obter detalhes da API, consulte TagResourceem Referência de AWS CLI Comandos.

Python

SDK para 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
```

- Para obter detalhes da API, consulte a TagResourceReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte a TagResourcereferência da API AWS SDK for SAP ABAP.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando HealthLake com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`UntagResource`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UntagResource`.

CLI

AWS CLI

Como remover tags de um datastore.

O exemplo de `untag-resource` a seguir mostra como remover tags de um datastore.

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

Este comando não produz saída.

- Para obter detalhes da API, consulte UntagResourceem Referência de AWS CLI Comandos.

Python

SDK para 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
```

- Para obter detalhes da API, consulte a UntagResourceReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte a UntagResourcereferência da API AWS SDK for SAP ABAP.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando HealthLake com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Integrando AWS HealthLake

Os AWS serviços a seguir se integram diretamente AWS HealthLake para permitir processamento integrado de linguagem natural, consulta SQL e armazenamento de dados.

- O Amazon Comprehend Medical é um serviço de processamento de linguagem natural (NLP) qualificado pela HIPAA que usa bibliotecas de aprendizado de máquina para extrair dados de saúde significativos de textos médicos não estruturados. HealthLake Para obter mais informações, consulte o Guia do desenvolvedor do Amazon Comprehend Medical.
- O Amazon Athena é um serviço de consulta interativo que permite analisar HealthLake dados diretamente nos buckets do Amazon Simple Storage Service (Amazon S3) usando SQL padrão. Para obter mais informações, consulte o Guia do desenvolvedor do Amazon Athena.

Tópicos

- Processamento de linguagem natural (NLP) integrado para HealthLake
- Consultando HealthLake dados com o Amazon Athena

Processamento de linguagem natural (NLP) integrado para HealthLake

AWS HealthLake fornece bibliotecas integradas de processamento de linguagem natural (NLP) para analisar, identificar e mapear dados não estruturados armazenados em tipos de recursos `DocumentReference`FHIR.

> **Importante**
>
> A PNL integrada para HealthLake está desativada por padrão. Para ativá-lo, envie um caso de suporte usando AWS Support Center Console. Para criar seu caso, faça login no seu Conta da AWS e escolha Criar caso.

HealthLake A PNL integrada funciona chamando as ações Amazon Comprehend `DetectEntities-V2` Medical `InferICD10-CM` e API. `InferRxNorm` As ações anexam seus resultados como uma extensão ao `DocumentReference` recurso. Quando as ações da API Amazon Comprehend Medical detectam características que são`SIGN`, e/ou `SYMPTOMDIAGNOSIS`, elas geram

um recurso FHIR. `Linkage` `Observation`Recursos novos `Condition` e são criados a partir de entidades identificadas com as características de `SIGNSYMPTOM`,, ou `DIAGNOSIS` e são vinculados ao documento de origem usando o `Linkage` recurso.

> **Note**
>
> Embora as `GET` solicitações sejam suportadas para recursos FHIR gerados pela PNL HealthLake integrada, a funcionalidade da API FHIR não é. `search` Para saber mais sobre como pesquisar extensões de PNL usando a integração com HealthLake o Athena, consulte. Índice e consulta SQL

Sumário

- HealthLake Bibliotecas de PNL integradas
- Usando interações da API FHIR REST
- Parâmetros de pesquisa para HealthLake PNL integrada
- HealthLake exemplos de solicitações de PNL integradas

HealthLake Bibliotecas de PNL integradas

HealthLake infere dados encontrados no tipo de `DocumentReference` recurso usando as bibliotecas Amazon Comprehend Medical. A API Amazon Comprehend Medical opera e *`InferRxNorm`* detecta *`DetectEntities-V2`* condições *`InferICD10-CM`* médicas como características. Cada operação fornece informações diferentes.

> **Suporte a idiomas**
>
> As operações da API Amazon Comprehend Medical só detectam entidades médicas em textos em inglês.

- DetectEntities-V2: inspeciona o texto clínico de uma variedade de entidades médicas e retorna informações específicas sobre elas, como categoria da entidade, localização e pontuação de confiança.
- Inferir ICD1 0-CM: Detecta condições médicas em um prontuário de paciente como entidades e vincula essas entidades a identificadores de conceitos normalizados na base de conhecimento

CID-10-CM do National Center for Health Statistics do CDC, sob autorização da Organização Mundial da Saúde (OMS).

- InferRxNorm: Detecta medicamentos como entidades listadas em um prontuário de paciente e os vincula aos identificadores de conceitos normalizados no RxNorm banco de dados da National Library of Medicine.

As características suportadas para cada operação de API são `SIGNSYMPTOM`, `DIAGNOSIS` e. Se características forem detectadas, elas serão adicionadas como extensões compatíveis com FHIR em diferentes locais em seu armazenamento de dados. HealthLake

Locais onde as extensões são adicionadas.

- `DocumentReference`: Os resultados das operações da API Amazon Comprehend Medical são adicionados como um `extension` a cada documento encontrado dentro do tipo de recurso. `DocumentReference` Os resultados da extensão são divididos em dois grupos. Você pode encontrá-los nos resultados com base em seus`URL`.
  - `http://healthlake.amazonaws.com/system-generated-resources/`
    - Esses são tipos de recursos que foram criados ou adicionados por HealthLake.
  - `http://healthlake.amazonaws.com/aws-cm/`
    - Onde a saída bruta das operações da API Amazon Comprehend Medical é adicionada ao seu armazenamento de dados. HealthLake
- `Linkage`: esse tipo de recurso é adicionado ou criado como resultado da PNL integrada. Uma `GET` solicitação em um específico `Linkage` retorna uma lista de recursos vinculados. Para identificar se a `Linkage` foi adicionado por HealthLake, procure o par de `"tag": [{"display": "SYSTEM_GENERATED"}]` valores-chave adicionados. Para saber mais sobre as especificações do FHIR para o Linkage, consulte `Linkage`a documentação do FHIR R4.
- Tipos de recursos FHIR gerados como resultado das operações do Amazon Comprehend Medical.
  - `Observation`: inclui resultados das ações da API Amazon Comprehend Medical `DetectEntities-V2` e `InferICD10-CM` quando as características são ou. `SIGN` `SYMPTOM`
  - `Condition`: inclui resultados das ações da API Amazon Comprehend Medical `DetectEntities-V2` e `InferICD10-CM` quando a característica é. `DIAGNOSIS`
  - `MedicationStatement`: inclui resultados das ações da API Amazon Comprehend Medical. `InferRxNorm`

Usando interações da API FHIR REST

Por padrão, as características detectadas pelas operações da API Amazon Comprehend Medical não são retornadas ao fazer uma solicitação. `GET` Para ver os resultados das operações integradas de PNL, você deve especificar um conhecido `ID` para os seguintes tipos de recursos do FHIR.

- `Linkage`
- `Observation`
- `Condition`
- `MedicationStatement`

Os resultados das ações HealthLake integradas de PNL fora do tipo de `DocumentReference` recurso estão disponíveis por meio de uma `GET` solicitação em que `ID` se sabe que o especificado contém resultados das operações da API Amazon Comprehend Medical.

Parâmetros de pesquisa para HealthLake PNL integrada

A tabela a seguir lista os atributos pesquisáveis para a PNL HealthLake integrada.

Parâmetros de pesquisa para HealthLake PNL

Parâmetros de pesquisa	Encontra correspondências para
Detecte ntities-entity-category	Categoria de entidade dentro da DetectEntities subextensão dentro da extensão AWS CM
Detecte ntities-entity-text	Texto da entidade dentro da DetectEntities subextensão dentro da extensão AWS CM
Detecte ntities-entity-type	Tipo de entidade dentro da DetectEntities subextensão dentro da extensão AWS CM
Detecte ntities-entity-score	Pontuação da entidade dentro da DetectEntities subextensão dentro da extensão AWS CM
infer-icd10 cm-entity-text	Texto da entidade dentro da subextensão Infer ICD10CM dentro da extensão CM AWS

Parâmetros de pesquisa	Encontra correspondências para
infer-icd10 cm-entity-score	Pontuação da entidade dentro da subextensão Infer ICD1 0CM dentro da extensão CM AWS
infer-icd10 cm-entity-concept-code	Código de conceito de entidade dentro da subextensão Infer ICD1 0CM dentro da extensão CM AWS
infer-icd10 cm-entity-concept-description	Descrição do conceito de entidade dentro da subextensão Infer ICD1 0CM dentro da extensão CM AWS
infer-icd10 cm-entity-concept-score	Pontuação do conceito de entidade na subextensão Infer ICD1 0CM na extensão CM AWS
infer-rxnorm-entity-score	Pontuação da entidade dentro da InferRxNorm subextensão dentro da extensão AWS CM
infer-rxnorm-entity-text	Texto da entidade dentro da InferRxNorm subextensão dentro da extensão AWS CM
infer-rxnorm-entity-concept-código	Código de conceito de entidade dentro da InferRxNorm subextensão dentro da extensão AWS CM
infer-rxnorm-entity-concept-descrição	Descrição do conceito de entidade dentro da InferRxNorm subextensão dentro da extensão AWS CM
infer-rxnorm-entity-concept-pontuação	Pontuação do conceito de entidade dentro da InferRxNorm subextensão dentro da extensão AWS CM

HealthLake fornece uma pesquisa especial para corresponder aos critérios em que `EntityText` e `EntityCategory` fazem parte da mesma entidade. A tabela a seguir descreve os parâmetros de pesquisa especiais suportados pelo HealthLake.

Parâmetros de pesquisa

Parâmetros de pesquisa	Partidas devolvidas
Detecte ntities-entity-text-category	Se houver pelo menos uma entidade na DetectEnt ities subextensão que corresponda tanto à EntityText quanto à EntityCategory.
Detecte ntities-entity-type-score	Se houver pelo menos uma entidade na DetectEnt ities subextensão que corresponda ao entityType e ao entityScore.
Detecte ntities-entity-text-score	Se houver pelo menos uma entidade na DetectEnt ities subextensão que corresponda tanto ao EntityTex t quanto ao EntityScore.
Detecte ntities-entity-text-type	Se houver pelo menos uma entidade na DetectEnt ities subextensão que corresponda tanto ao EntityTex t quanto ao EntityType.
Detecte ntities-entity-category-score	Se houver pelo menos uma entidade que correspon da tanto à EntityCategory quanto à EntityScore.
código cm-entity-text-concept infer-icd10	Se houver pelo menos uma entidade na subextens ão Infer ICD1 0CM que corresponda ao EntityTex t e houver pelo menos um ConceptCode para essa entidade que corresponda ao código.
infer-icd10 - pontuação cm-entity-text-con cept	Se houver pelo menos uma entidade na subextens ão Infer ICD1 0CM que corresponda ao EntityText e houver pelo menos um ConceptScore para essa entidade que corresponda à pontuação.
cm-entity-concept-descriptioninfer-icd10 - pontuação conceitual	Se houver pelo menos um conceito dentro da entidade na subextensão Infer ICD1 0CM que corresponda à descrição do conceito e ao ConceptSc ore.
infer-rxnorm-entity-text-código conceitual	Se houver pelo menos uma entidade na InferRxNorm subextensão que corresponda ao EntityText e houver

Parâmetros de pesquisa	Partidas devolvidas
	pelo menos um ConceptCode para essa entidade que corresponda ao código.
infer-rxnorm-entity-text-pontuação conceitual	Se houver pelo menos uma entidade na InferRxNorm subextensão que corresponda ao EntityText e houver pelo menos um ConceptScore para essa entidade que corresponda à pontuação.
infer-rxnorm-entity-concept-description-concept-score	Se houver pelo menos um conceito dentro da entidade na InferRxNorm subextensão que correspon da à descrição do conceito e ao ConceptScore.

HealthLake exemplos de solicitações de PNL integradas

Exemplo 1: **`Patient`** registro ingerido em um armazenamento de HealthLake dados

A seguir está um exemplo de uma nota clínica baseada em um `Patient` encontro com um profissional de saúde.

> **Dados sintéticos**
>
> O texto no exemplo a seguir é conteúdo sintético e não contém informações de saúde protegidas (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)
```

Como lembrete, essas informações são codificadas no formato base64 no recurso. `DocumentReference` Quando esse documento é inserido HealthLake e as operações da API Amazon Comprehend Medical são concluídas, para ver os resultados, você pode começar com a solicitação sobre `GET` o tipo de recurso. `DocumentReference`

```
GET https://https://healthlake.region.amazonaws.com/datastore/datastoreId/
r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/DocumentReference
```

Quando as operações da API Amazon Comprehend Medical forem bem-sucedidas, procure esses pares de valores-chave nos links a seguir `extension` `"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."
 }
```

As guias a seguir mostram como o prontuário médico ingerido é relatado em seu armazenamento de HealthLake dados com base no tipo de recurso.

DocumentReference

Para ver os resultados de um único tipo de `DocumentReference` recurso, faça uma `GET` solicitação em que o `id` de um recurso específico seja fornecido.

```
GET https://https://healthlake.region.amazonaws.com/datastore/datastoreId/
r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/DocumentReference/0e938f03-da7f-4178-acd8-
eea9586c46ed
```

Quando bem-sucedido, você recebe um código de resposta `200` HTTP e a seguinte resposta JSON (que foi truncada para maior clareza).

Aqui está a `http://healthlake.amazonaws.com/system-generated-resources/` porção. Você pode ver que um novo `Linkage/e366d29f-2c22-4c19-866e-09603937935a` foi adicionado. Você também pode ver onde HealthLake foram adicionadas descobertas baseadas em inferências a tipos específicos `Observation` e de `Condition` recursos.

Para ver como esses tipos de recursos foram alterados, escolha as guias relacionadas.

```
{
  "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

Para ver os resultados de um único tipo de `Linkage` recurso, faça uma `GET` solicitação em que o `ID` de um recurso específico seja fornecido.

```
GET https://https://healthlake.region.amazonaws.com/datastore/datastoreId/
r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/Linkage/e366d29f-2c22-4c19-866e-09603937935a
```

Quando bem-sucedido, você recebe um código de resposta `200` HTTP e a seguinte resposta JSON truncada.

A resposta contém o `item` elemento. Nele, o par de valores-chave `"type": "source"` indica a `DocumentReference` entrada específica usada para modificar `Condition` e `Observations` listada sob o par de `"type": "alternate"` valores-chave.

Você também vê o *`meta`* elemento e um par de valores-chave correspondente*`"tag": [{"display": "SYSTEM_GENERATED"}]`*, indicando que esses recursos foram criados por. 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

Para ver os resultados de um único tipo de `Observation` recurso, faça uma `GET` solicitação em que o `ID` de um recurso específico seja fornecido.

```
GET https://https://healthlake.region.amazonaws.com/
datastore/datastoreId/r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/
Observation/e366d29f-2c22-4c19-866e-09603937935a
```

Os resultados das operações da API Amazon Comprehend Medical são alterados para os seguintes elementos:`code`,, e. `meta` `modifierExtension`

`code`

Um elemento do tipo`CodeableConcept`. Para saber mais, consulte a `CodeableConcept`documentação do FHIR R4.

HealthLake acrescenta os três pares de valores-chave a seguir.

- `"system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/"`: onde o URL se refere a uma operação específica da API Amazon Comprehend Medical. Nesse caso, inferir ICD1 0CM.
- `"code": "A52.06"`: Onde `A52.06` está o código ICD-10-CM que identifica o conceito encontrado na base de conhecimento dos Centros de Controle de Doenças.
- `"display": "Other syphilitic heart involvement"`: Onde `"Other syphilitic heart involvement"` está a descrição longa do código ICD-10-CM na ontologia.

A resposta JSON truncada a seguir contém somente o elemento. `code`

```
"code": {
  "coding":
  [
    {
   "system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/",
   "code": "A52.06",
   "display": "Other syphilitic heart involvement"
    }
  ],
  "text": "Other syphilitic heart involvement"
 }
```

Para entender a confiança do modelo de que o código ICD-10-CM atribuído está correto, use o elemento. `modifierExtension`

`meta`

O `meta` elemento contém metadados que indicam se o `code` elemento contém detalhes que foram adicionados pelas operações da API Amazon Comprehend Medical.

A resposta JSON truncada a seguir contém somente o elemento. `meta`

```
"meta": {
  "lastUpdated": "2022-10-21T19:38:30.879Z",
  "tag": [{
   "display": "SYSTEM_GENERATED"
  }]
```

```
 }
```

modifierExtension

O `modifierExtension` elemento contém mais detalhes sobre o nível de confiança dos códigos atribuídos encontrados no `code` elemento. Ele também tem pares de valores-chave que fornecem um link para o original DocumentReference usado para gerar os resultados e o tipo de recurso Linkage relacionado.

Para cada `coding` elemento adicionado, você verá um `entity-score` e um `entity-Concept-Score` adicionados à ModifierExtension. Para cada valor no par de valores-chave, você vê uma pontuação. Pois`entity-score`, essa pontuação é o nível de confiança que o Amazon Comprehend Medical tem na precisão da detecção. Pois`entity-Concept-Score`, essa pontuação é o nível de confiança que o Amazon Comprehend Medical tem de que a entidade está vinculada com precisão a um conceito do ICD-10-CM.

A resposta JSON truncada a seguir contém somente o elemento. `modifierExtension`

```
"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"
   }
  }
 ]
```

Resposta JSON completa


```
{
  "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

Para ver os resultados de um único tipo de `Condition` recurso, faça uma `GET` solicitação em que o `ID` de um recurso específico seja fornecido.

```
GET https://https://healthlake.region.amazonaws.com/datastore/datastoreId/
r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/Condition/b06d343d-
ddb8-4f36-82cb-853fcd434dfd
```

Os resultados das operações da API Amazon Comprehend Medical são alterados para os seguintes elementos:`code`,, e. `meta` `modifierExtension`

`code`

Um elemento do tipo`CodeableConcept`. Para saber mais, consulte a `CodeableConcept`documentação do FHIR R4.

HealthLake acrescenta os três pares de valores-chave a seguir.

- `"system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/"`: onde o URL se refere a uma operação específica da API Amazon Comprehend Medical. Nesse caso, inferir ICD1 0CM.
- `"code": "I70.0"`: Onde `A52.06` está o código ICD-10-CM que identifica o conceito encontrado na base de conhecimento dos Centros de Controle de Doenças.
- `"display": "Atherosclerosis of aorta"`: Onde `"Other syphilitic heart involvement"` está a descrição longa do código ICD-10-CM na ontologia.

A resposta JSON truncada a seguir contém somente o elemento. `code`

```
"code": {
  "coding":
  [
    {
   "system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/",
   "code": "I70.0",
   "display": "Atherosclerosis of aorta"
```

```
    }
  ],
  "text": "Atherosclerosis of aorta"
 }
```

Para entender a confiança do modelo de que o código ICD-10-CM atribuído está correto, use o elemento. `modifierExtension`

meta

O `meta` elemento contém metadados que indicam se o `code` elemento contém detalhes que foram adicionados pelas operações da API Amazon Comprehend Medical.

A resposta JSON truncada a seguir contém somente o elemento. `meta`

```
"meta": {
  "lastUpdated": "2022-10-21T19:38:30.877Z",
  "tag": [{
   "display": "SYSTEM_GENERATED"
  }]
 }
```

modifierExtension

O `modifierExtension` elemento contém mais detalhes sobre o nível de confiança dos códigos atribuídos encontrados no `code` elemento. Ele também tem pares de valores-chave que fornecem um link para o original DocumentReference usado para gerar os resultados e o tipo de recurso Linkage relacionado.

Para cada `coding` elemento adicionado, você verá um `entity-score` e um `entity-Concept-Score` adicionados à ModifierExtension. Para cada valor no par de valores-chave, você vê uma pontuação. Pois`entity-score`, essa pontuação é o nível de confiança que o Amazon Comprehend Medical tem na precisão da detecção. Pois`entity-Concept-Score`, essa pontuação é o nível de confiança que o Amazon Comprehend Medical tem de que a entidade está vinculada com precisão a um conceito do ICD-10-CM.

A resposta JSON truncada a seguir contém somente o elemento. `modifierExtension`

```
"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"
   }
  }
 ]
```

Resposta JSON completa

```
{
  "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"
 }
```

Exemplo 2: A **`DocumentReference`** que contém o tipo **`MedicationStatement`** de recurso

Aqui está um exemplo de uma nota clínica baseada no encontro de um paciente com um profissional médico.

> ⚠ Dados sintéticos
>
> O texto neste exemplo é conteúdo sintético e não contém informações de saúde protegidas (PHI).

```
Tom is not prescribed Advil
```

As guias a seguir mostram como o prontuário médico ingerido é relatado em seu armazenamento de HealthLake dados com base no tipo de recurso.

DocumentReference

Para ver os resultados de um único tipo de `DocumentReference` recurso, faça uma `GET` solicitação em que o `ID` de um recurso específico seja fornecido.

```
GET https://https://healthlake.region.amazonaws.com/datastore/datastoreId/
r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/DocumentReference/c549125d-a218-421f-
b8bf-23614c5e796c
```

Quando bem-sucedido, você recebe um código de resposta `200` HTTP e a seguinte resposta JSON truncada.

O par de valores-chave,`"url": "http://healthlake.amazonaws.com/system-generated-resources/"`, indica que os tipos de recursos dentro dele `extension` foram adicionados pelas operações da API Amazon Comprehend Medical. Você pode ver o novo tipo de `Linkage` recurso e vários `MedicationStatement` recursos.

```
"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

Para ver os resultados de um único tipo de `Linkage` recurso, faça uma `GET` solicitação em que o `ID` de um recurso específico seja fornecido.

```
GET https://https://healthlake.region.amazonaws.com/datastore/datastoreId/
r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/Linkage/394bb244-177b-4409-8657-26b20ed56dd7
```

Em caso de sucesso, você recebe um código de resposta `200` HTTP e a seguinte resposta JSON.

A resposta contém o `item` elemento. Nele, o par de valores-chave `"type": "source"` indica a `DocumentReference` entrada específica usada para modificar os tipos de `MedicationStatement` recursos.

Você também pode ver o `meta` elemento e um par de valores-chave correspondente`"tag": [{"display": "SYSTEM_GENERATED"}]`, indicando que esses recursos foram criados por. 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

Para ver os resultados de um único tipo de `MedicationStatement` recurso, faça uma `GET` solicitação em que o `ID` de um recurso específico seja fornecido.

```
GET https://https://healthlake.region.amazonaws.com/
datastore/datastoreId/r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/
MedicationStatement/9a89b0d3-6681-45ca-9926-27951edce5c7
```

O tipo de `MedicationStatement` recurso é onde os resultados da operação da InferRxNorm API Amazon Comprehend Medical são encontrados. Os resultados são alterados para os seguintes elementos: `medicationCodeableConceptmeta`, `modifierExtension` e.

medicationCodeableConcept

Um elemento do tipo`CodeableConcept`. Para saber mais, consulte a `CodeableConcept`documentação do FHIR R4.

HealthLake acrescenta os três pares de valores-chave a seguir.

- `"system": ""http://healthlake.amazonaws.com/aws-cm/infer-rxnorm/`: onde o URL se refere a uma operação específica da API Amazon Comprehend Medical. Nesse caso, InferRxNorm.
- `"code": "731533"`: Onde `731533` está um ID de RxNorm conceito, também conhecido como RxCUI.
- `"display": "ibuprofen 200 MG Oral Capsule [Advil]"`: Onde `ibuprofen 200 MG Oral Capsule [Advil]` está a descrição do RxNorm conceito.

A resposta JSON truncada a seguir contém somente o elemento. `MedicationStatement`

```
"medicationCodeableConcept": {
  "coding": [
   {
    "system": "http://healthlake.amazonaws.com/aws-cm/infer-rxnorm/",
    "code": "731533",
```

```
    "display": "ibuprofen 200 MG Oral Capsule [Advil]"
   }
  ]
 }
```

meta

O `meta` elemento contém metadados que indicam se o `code` elemento contém detalhes que foram adicionados pelas operações da API Amazon Comprehend Medical.

A resposta JSON truncada a seguir contém somente o elemento. `meta`

```
"meta": {
  "lastUpdated": "2022-10-24T20:05:02.800Z",
  "tag": [
   {
    "display": "SYSTEM_GENERATED"
   }
  ]
 }
```

modifierExtension

O `modifierExtension` elemento contém pares de valores-chave que fornecem um link para o original `DocumentReference` usado para gerar os resultados e o tipo de recurso Linkage relacionado.

```
"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"
    }
   }
 ]
```

Consultando HealthLake dados com o Amazon Athena

Durante um trabalho de HealthLake importação, os dados JSON FHIR aninhados passam por um processo ETL e são armazenados no formato de tabela aberta Apache Iceberg, em que cada tipo de recurso FHIR é representado como uma tabela individual no Athena. Isso permite que os usuários consultem os dados do FHIR usando SQL, mas sem precisar exportá-los primeiro. Isso é valioso, pois capacita médicos e cientistas a consultar dados do FHIR para validar suas decisões ou avançar em suas pesquisas. Para obter mais informações sobre como as tabelas do Apache Iceberg funcionam no Athena, consulte Consultar tabelas do Apache Iceberg no Guia do usuário do Athena.

> **Note**
>
> HealthLake suporta a `read` interação FHIR R4 em seus dados no Athena HealthLake . Para obter mais informações, consulte Lendo um recurso do FHIR.

Os tópicos desta seção descrevem como conectar seu armazenamento de HealthLake dados ao Athena, como consultá-lo usando SQL e como conectar resultados a outros AWS serviços para análise posterior.

Tópicos

- Comece a usar o Amazon Athena
- Consultando HealthLake dados com SQL
- Exemplos de consultas SQL com filtragem complexa

Comece a usar o Amazon Athena

Para fazer a integração HealthLake com o Amazon Athena, você deve configurar permissões. Para fazer isso, você criará um usuário, grupo ou função do Athena e concederá a eles acesso aos recursos do FHIR localizados em um armazenamento de dados. HealthLake

- Conceder a um usuário, grupo ou função acesso a um armazenamento de HealthLake dados (AWS Lake Formation Console)
- Configurando uma conta Athena

Conceder a um usuário, grupo ou função acesso a um armazenamento de HealthLake dados (AWS Lake Formation Console)

> Pessoa: administrador HealthLake
>
> A pessoa HealthLake administradora é uma administradora de data lake em AWS Lake Formation. Eles concedem acesso aos armazenamentos de HealthLake dados em Lake Formation.

Para cada armazenamento de dados criado, há duas entradas visíveis no console do AWS Lake Formation. Uma entrada é um link de recurso. Os nomes dos links de recursos são sempre exibidos em itálico. Cada link de recurso é exibido com o nome e o proprietário do recurso compartilhado vinculado. Para todos os armazenamentos de HealthLake dados, o proprietário do recurso compartilhado é a conta HealthLake de serviço. A outra entrada é o armazenamento HealthLake de dados na conta HealthLake de serviço. As etapas desse procedimento usam o armazenamento de dados que é o link do recurso.

Para saber mais sobre links de recursos, consulte Como os links de recursos funcionam no Lake Formation no AWS Lake Formation Developer Guide.

Para que um usuário, grupo ou função possa consultar dados no Athena, você deve conceder a permissão Descreve no banco de dados de recursos. Em seguida, você deve conceder Selecionar e Descrever nas tabelas.

ETAPA 1: Para conceder permissões DESCRIBE em um banco de HealthLake dados de links de recursos do armazenamento de dados

1. Abra o console do AWS Lake Formation: https://console.aws.amazon.com/lakeformation/
2. Na barra de navegação principal, escolha Bancos de dados.
3. Na página Bancos de dados, escolha o botão de rádio ao lado do nome do armazenamento de dados que está em itálico.
4. Escolha Ações (▾).
5. Selecione Conceder.
6. Na página Conceder permissões de dados, em Diretores, escolha usuários ou funções do IAM.
7. Em Usuários ou funções do IAM, use a seta para baixo (▾) ou pesquise o usuário, a função ou o grupo do IAM sobre o qual você deseja poder fazer consultas no Athena.

8. Em LF-Tags ou cartão de recursos do catálogo, escolha a opção Recursos do catálogo de dados nomeados.
9. Em Bancos de dados, use a seta para baixo (▼) para escolher o banco de HealthLake dados do armazenamento de dados ao qual você deseja compartilhar o acesso.
10. No cartão Permissões do link do recurso, em Permissões do link do recurso, escolha Descrever.

Quando a concessão é bem-sucedida, o banner Conceder permissão de sucesso é exibido. Para ver a permissão que você acabou de conceder, escolha Permissões do Data lake. Encontre o usuário, o grupo e a função na tabela. Na coluna Permissões, você verá a lista Descrever.

Agora você deve usar Grant on target para conceder Selecionar e Descrever em todas as tabelas no banco de dados.

ETAPA 2: Conceder acesso a todas as tabelas em um link de recurso do armazenamento de HealthLake dados

1. Abra o console do AWS Lake Formation: https://console.aws.amazon.com/lakeformation/
2. Na barra de navegação principal, escolha Bancos de dados.
3. Na página Bancos de dados, escolha o botão de rádio ao lado do nome do armazenamento de dados que está em itálico.
4. Escolha Ações (▼).
5. Escolha Grant no alvo.
6. Na página Conceder permissões de dados, em Diretores, escolha usuários ou funções do IAM.
7. Em Usuários ou funções do IAM, use a seta para baixo (▼) ou pesquise o usuário, grupo ou função do IAM sobre o qual você deseja poder fazer consultas no Athena.
8. Em LF-Tags ou cartão de recursos do catálogo, escolha a opção Recursos do catálogo de dados nomeados.
9. Em Bancos de dados, use a seta para baixo (▼) para escolher o banco de HealthLake dados do armazenamento de dados ao qual você deseja conceder acesso.
10. Em Tabelas, escolha Todas as tabelas para compartilhar todas as tabelas com um HealthLake usuário.
11. No cartão Permissões da tabela, em Permissões da tabela, escolha Descrever e selecionar.
12. Selecione Conceder.

Depois de escolher conceder, um banner de sucesso de permissões de concessão é exibido. O usuário especificado agora pode fazer consultas em um armazenamento de HealthLake dados no Athena.

Começando com Athena

> **HealthLake usuário**
>
> O HealthLake usuário usará o console do Athena ou AWS SDKs para consultar um armazenamento de HealthLake dados compartilhado com ele pelo HealthLake administrador. AWS CLI

Para consultar um armazenamento de dados usando o Athena, você deve fazer as três coisas a seguir.

- Conceda ao usuário ou à função do IAM acesso ao armazenamento de HealthLake dados por meio do Lake Formation. Para saber mais, consulte Conceder a um usuário, grupo ou função acesso a um armazenamento de HealthLake dados (AWS Lake Formation Console).
- Crie um grupo de trabalho para seu armazenamento HealthLake de dados.
- Designe um bucket do Amazon S3 para armazenar os resultados da consulta.

Para começar a usar o Athena, adicione as políticas FullAccess AWS gerenciadas AmazonAthenaFullAccesse o AmazonS3 ao seu usuário, grupo ou função. Usar uma política AWS gerenciada é uma ótima maneira de começar a usar um novo serviço. Lembre-se de que as políticas gerenciadas pela AWS podem não conceder permissões de privilégio mínimo para seus casos de uso específicos porque estão disponíveis para uso por todos os clientes da AWS. Ao definir permissões com as políticas do IAM, conceda apenas as permissões necessárias para executar uma tarefa. Para saber mais sobre o IAM e a aplicação de privilégios mínimos, consulte Aplicar permissões de privilégios mínimos no Guia do usuário do IAM.

> **Important**
>
> Para consultar um armazenamento HealthLake de dados no Athena, você deve usar o mecanismo Athena versão 3.

Grupos de trabalho são recursos e, portanto, você pode usar políticas baseadas em IAM para controlar o acesso a grupos de trabalho específicos. Para saber mais, consulte Como usar grupos de trabalho para controlar o acesso e os custos das consultas no Guia do usuário do Athena.

Para saber mais sobre como configurar grupos de trabalho, consulte o Guia do https://docs.aws.amazon.com/athena/latest/ug/workgroups-procedure.html usuário do Athena.

> **Note**
>
> A região em que seu bucket do Amazon S3 está e o console do Athena devem corresponder.

Antes de executar uma consulta, um local de bucket de resultados de consultas do Amazon S3 precisa ser especificado, ou você deve usar um grupo de trabalho que especificou um bucket e com uma configuração que substitui as configurações do cliente. Os arquivos de saída são salvos automaticamente para cada consulta executada.

Para obter mais detalhes sobre a especificação dos locais dos resultados da consulta no console do Athena, consulte Especificação de um local do resultado da consulta usando o console do Athena no Guia do usuário do Amazon Athena.

Para ver exemplos de como consultar seu armazenamento de HealthLake dados no Athena, consulte. Consultando HealthLake dados com SQL

Consultando HealthLake dados com SQL

Quando você importa seus dados FHIR para o armazenamento de dados, HealthLake os dados JSON FHIR aninhados passam simultaneamente por um processo de ETL e são armazenados no formato de tabela aberta Apache Iceberg no Amazon S3. Cada tipo de recurso FHIR do seu armazenamento de HealthLake dados é convertido em uma tabela, onde pode ser consultado usando o Amazon Athena. As tabelas podem ser consultadas individualmente ou em grupo usando consultas baseadas em SQL. Devido à estrutura dos armazenamentos de dados, seus dados são importados para o Athena como vários tipos de dados diferentes. Para saber mais sobre a criação de consultas SQL que podem acessar esses tipos de dados, consulte Matrizes de consultas com tipos complexos e estruturas aninhadas no Guia do usuário do Amazon Athena.

> **Note**
>
> Todos os exemplos neste tópico usam dados fictícios criados usando o Synthea. Para saber mais sobre como criar um armazenamento de dados pré-carregado com dados Synthea, consulte. Criando um armazenamento HealthLake de dados

Para cada elemento em um tipo de recurso, a especificação FHIR define uma cardinalidade. A cardinalidade de um elemento define os limites inferior e superior de quantas vezes esse elemento pode aparecer. Ao criar uma consulta SQL, você deve levar isso em consideração. Por exemplo, vamos ver alguns elementos em Tipo de recurso: Paciente.

- Elemento: Nome A especificação FHIR define a cardinalidade como. `0..*`

  O elemento é capturado como uma matriz.

```
[{
 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
}]
```

  No Athena, para ver como um tipo de recurso foi ingerido, procure-o em Tabelas e visualizações. Para acessar elementos nessa matriz, você pode usar a notação de pontos. Aqui está um exemplo simples que acessaria os valores de `given` `family` e.

```
SELECT
    name[1].given as FirstName,
```

```
    name[1].family as LastName
FROM Patient
```

- Elemento: MaritalStatus A especificação FHIR define a cardinalidade como. `0..1`

  Esse elemento é capturado como 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
}
```

  No Athena, para ver como um tipo de recurso foi ingerido, procure-o em Tabelas e visualizações. Para acessar pares de valores-chave no JSON, você pode usar a notação de pontos. Como não é uma matriz, nenhum índice de matriz é necessário. Aqui está um exemplo simples que acessaria o valor `detext`.

```
SELECT
    maritalstatus.text as MaritalStatus
FROM Patient
```

Para saber mais sobre como acessar e pesquisar JSON, consulte Consultar JSON no Guia do usuário do Athena.

As instruções de consulta da Athena Data Manipulation Language (DML) são baseadas no Trino. O Athena não suporta todos os recursos do Trino e há diferenças significativas. Para saber mais, consulte consultas, funções e operadores de DML no Guia do usuário do Amazon Athena.

Além disso, o Athena oferece suporte a vários tipos de dados que você pode encontrar ao criar consultas em seu HealthLake armazenamento de dados. Para saber mais sobre os tipos de dados no Athena, consulte Tipos de dados no Amazon Athena no Guia do usuário do Amazon Athena.

Para saber mais sobre como as consultas SQL funcionam no Athena, consulte a referência de SQL para o Amazon Athena no Guia do usuário do Amazon Athena.

Cada guia mostra exemplos de como pesquisar os tipos de recursos especificados e os elementos associados usando o Athena.

Element: Extension

O elemento `extension` é usado para criar campos personalizados em um armazenamento de dados.

Este exemplo mostra como acessar os recursos do `extension` elemento encontrado no tipo de `Patient` recurso.

Quando seu armazenamento HealthLake de dados é importado para o Athena, os elementos de um tipo de recurso são analisados de forma diferente. Como a estrutura do `element` é variável, ela não pode ser totalmente especificada no esquema. Para lidar com essa variabilidade, os elementos dentro da matriz são passados como strings.

Na descrição da tabela de`Patient`, você pode ver o elemento `extension` descrito como`array<string>`, o que significa que você pode acessar os elementos da matriz usando um valor de índice. Para acessar os elementos da string, no entanto, você deve usar`json_extract`.

Aqui está uma única entrada do `extension` elemento encontrado na tabela de pacientes.

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

Embora seja um JSON válido, Athena o trata como uma string.

Este exemplo de consulta SQL demonstra como você pode criar uma tabela que contém os `patient-birthPlace` elementos `patient-mothersMaidenName` e. Para acessar esses elementos, você precisa usar diferentes índices de matriz e `json_extract.`

```
SELECT
    extension[1],
    json_extract(extension[1], '$.valueString') AS MothersMaidenName,
    extension[2],
    json_extract(extension[2], '$.valueAddress.city') AS birthPlace
FROM patient
```

Para saber mais sobre consultas que envolvem JSON, consulte Extração de dados do JSON no Guia do usuário do Amazon Athena.

Element: birthDate (Age)

A idade não é um elemento do tipo de recurso do paciente no FHIR. Aqui estão dois exemplos de pesquisas que filtram com base na idade.

Como a idade não é um elemento, usamos o `birthDate` para as consultas SQL. Para ver como um elemento foi ingerido no FHIR, pesquise o nome da tabela em Tabelas e visualizações. Você pode ver que é do tipo string.

Exemplo 1: Calculando um valor para a idade

Neste exemplo de consulta SQL, usamos uma ferramenta SQL integrada `current_date` e `year` extraímos esses componentes. Em seguida, nós os subtraímos para retornar a idade real do paciente como uma coluna chamada`age`.

```
SELECT
 (year(current_date) - year(date(birthdate))) as age
FROM patient
```

Exemplo 2: Filtragem para pacientes que nasceram antes `2019-01-01` e são`male`.

A consulta SQL mostra como usar a `CAST` função para converter o `birthDate` elemento como tipo `DATE` e como filtrar com base em dois critérios na `WHERE` cláusula. Como o elemento é ingerido como string de tipo por padrão, devemos `CAST` usá-lo como tipo`DATE`. Em seguida, você pode usar o < operador para compará-lo com uma data diferente,`2019-01-01`. Ao usar`AND`, você pode adicionar um segundo critério à `WHERE` cláusula.

```
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

Este exemplo mostra pesquisas por locais dentro do tipo de recurso Localização em que o nome da cidade é 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

A condição do tipo de recurso armazena dados de diagnóstico relacionados a problemas que atingiram um nível de preocupação. HealthLakeO processamento médico de linguagem natural (PNL) integrado da gera novos `Condition` recursos com base nos detalhes encontrados no tipo de DocumentReference recurso. Quando um novo recurso é gerado, HealthLake anexa a tag `SYSTEM_GENERATED` ao `meta` elemento. Esse exemplo de consulta SQL demonstra como você pode pesquisar a tabela de condições e retornar os resultados em que os `SYSTEM_GENERATED` resultados foram removidos.

Para saber mais sobre o processamento integrado HealthLake de linguagem natural (NLP), consulteProcessamento de linguagem natural (NLP) integrado para HealthLake.

```
SELECT *
FROM condition
WHERE meta.tag[1] is NULL
```

Você também pode pesquisar em um elemento de string especificado para filtrar ainda mais sua consulta. O `modifierextension` elemento contém detalhes sobre qual `DocumentReference` recurso foi usado para gerar um conjunto de condições. Novamente, você deve usar `json_extract` para acessar os elementos JSON aninhados que são trazidos para o Athena como uma string.

Esse exemplo de consulta SQL demonstra como você pode pesquisar tudo o `Condition` que foi gerado com base em um específico`DocumentReference`. Use `CAST` para definir o elemento JSON como uma string para que você possa usá-lo `LIKE` para comparar.

```
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

O tipo de recurso, Observação, armazena medidas e afirmações simples feitas sobre um paciente, dispositivo ou outro assunto. HealthLakeO processamento integrado de linguagem

natural (PNL) da gera novos `Observation` recursos com base nos detalhes encontrados em um `DocumentReference` recurso. Esse exemplo de consulta SQL inclui `WHERE meta.tag[1] is NULL` comentários, o que significa que os `SYSTEM_GENERATED` resultados estão incluídos.

```
SELECT valueCodeableConcept.coding[1].code
FROM Observation
WHERE  valueCodeableConcept.coding[1].code = '266919005'
-- WHERE meta.tag[1] is NULL
```

Essa coluna foi importada como uma `struct`. Portanto, você pode acessar elementos dentro dele usando a notação de pontos.

Resource type: MedicationStatement

MedicationStatement é um tipo de recurso FHIR que você pode usar para armazenar detalhes sobre medicamentos que um paciente tomou, está tomando ou tomará no futuro. HealthLakeO processamento médico integrado de linguagem natural (PNL) gera novos MedicationStatement recursos com base em documentos encontrados no tipo de DocumentReference recurso. Quando novos recursos são gerados, HealthLake anexa a tag `SYSTEM_GENERATED` ao `meta` elemento. Este exemplo de consulta SQL demonstra como criar uma consulta que filtra com base em um único paciente usando seu identificador e encontra recursos que foram adicionados pela PNL integrada HealthLake da.

```
SELECT *
FROM medicationstatement
WHERE meta.tag[1].display = 'SYSTEM_GENERATED' AND subject.reference =
 'Patient/0679b7b7-937d-488a-b48d-6315b8e7003b';
```

Para saber mais sobre o processamento integrado HealthLake de linguagem natural (NLP), consulteProcessamento de linguagem natural (NLP) integrado para HealthLake.

Exemplos de consultas SQL com filtragem complexa

Os exemplos a seguir demonstram como usar consultas SQL do Amazon Athena com filtragem complexa para localizar dados FHIR de um armazenamento de dados. HealthLake

Example Crie critérios de filtragem com base em dados demográficos

Identificar os dados demográficos corretos do paciente é importante ao criar uma coorte de pacientes. Esse exemplo de consulta demonstra como você pode usar a notação de pontos Trino e `json_extract` filtrar dados em seu HealthLake armazenamento de dados.

```
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;
```

Usando o Athena Console, você pode classificar e baixar ainda mais os resultados.

Example Crie filtros para um paciente e suas condições relacionadas

O exemplo de consulta a seguir demonstra como você pode encontrar e classificar todas as condições relacionadas aos pacientes encontrados em um armazenamento de HealthLake dados.

```
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;
```

Você pode usar o console do Athena para classificar melhor os resultados ou baixá-los para análise posterior.

Example Crie filtros para pacientes e suas observações relacionadas

O exemplo de consulta a seguir demonstra como encontrar e classificar todas as observações relacionadas para pacientes encontradas em um armazenamento de HealthLake dados.

```
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 Crie condições de filtragem para um paciente e seus procedimentos relacionados

Conectar procedimentos aos pacientes é um aspecto importante da assistência médica. O exemplo de consulta SQL a seguir demonstra como usar o FHIR `Patient` e os tipos de `Procedure` recursos para fazer isso. A consulta SQL a seguir retornará todos os pacientes e seus procedimentos relacionados encontrados em seu armazenamento HealthLake de dados.

```
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;
```

Você pode usar o console do Athena para baixar os resultados para análise posterior ou classificá-los para entender melhor os resultados.

Example Crie condições de filtragem para um paciente e suas prescrições relacionadas

É importante ver uma lista atual de medicamentos que os pacientes estão tomando. Usando o Athena, você pode escrever uma consulta SQL que usa os tipos de `MedicationRequest` recursos `Patient` e os encontrados no seu armazenamento de HealthLake dados.

A consulta SQL a seguir une as `MedicationRequest` tabelas `Patient` e importadas para o Athena. Ele também organiza as prescrições em suas entradas individuais usando a notação de pontos.

```
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
```

Você pode usar o console do Athena para classificar os resultados ou baixá-los para análise posterior.

Example Veja os medicamentos encontrados no tipo **`MedicationStatement`** de recurso

O exemplo de consulta a seguir mostra como organizar o JSON aninhado importado para o Athena usando SQL. A consulta usa o `meta` elemento FHIR para indicar quando um medicamento foi adicionado pelo processamento integrado HealthLake de linguagem natural (NLP). Ele também é

usado `json_extract` para pesquisar dados dentro da matriz de strings JSON. Para obter mais informações, consulte Processamento de linguagem natural.

```
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;
```

Você pode usar o console do Athena para baixar esses resultados ou classificá-los.

Example Filtro para um tipo específico de doença

O exemplo mostra como você pode encontrar um grupo de pacientes, com idades entre 18 e 75 anos, que foram diagnosticados com diabetes.

```
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%');
```

Agora você pode usar o console do Athena para classificar os resultados ou baixá-los para análise posterior.

Monitoramento AWS HealthLake

O monitoramento e o registro são partes importantes da manutenção da segurança, confiabilidade, disponibilidade e desempenho do AWS HealthLake. AWS fornece os seguintes serviços para monitorar HealthLake, relatar quando algo está errado e realizar ações automáticas quando apropriado.

- AWS CloudTrailcaptura chamadas de API e eventos relacionados feitos por ou em nome de sua AWS conta e entrega os arquivos de log para um bucket do Amazon S3 que você especificar. Você pode identificar quais usuários e contas ligaram AWS, o endereço IP de origem a partir do qual as chamadas foram feitas e quando elas ocorreram. Para obter mais informações, consulte o Guia do usuário do AWS CloudTrail.
- A Amazon CloudWatch monitora seus AWS recursos e os aplicativos em que você executa AWS em tempo real. Você pode coletar e rastrear métricas, criar painéis personalizados e definir alarmes que o notificam ou que realizam ações quando uma métrica especificada atinge um limite definido. Por exemplo, você pode CloudWatch rastrear o uso da CPU ou outras métricas de suas EC2 instâncias da Amazon e iniciar automaticamente novas instâncias quando necessário. Para obter mais informações, consulte o Guia CloudWatch do usuário da Amazon.
- EventBridgeA Amazon é um serviço de ônibus de eventos sem servidor que facilita a conexão de seus aplicativos com dados de várias fontes. EventBridge fornece um fluxo de dados em tempo real de seus próprios aplicativos, aplicativos Software-as-a-Service (SaaS) e AWS serviços e encaminha esses dados para destinos como o Lambda. Isso permite monitorar eventos que ocorram em serviços e criem arquiteturas orientadas a eventos. Para obter mais informações, consulte o Guia EventBridge do usuário da Amazon.

Tópicos

- Registrando chamadas de HealthLake API usando AWS CloudTrail
- Monitoramento de HealthLake métricas usando a Amazon CloudWatch
- Monitorando HealthLake eventos usando a Amazon EventBridge

Registrando chamadas de HealthLake API usando AWS CloudTrail

AWS HealthLake é integrado com AWS CloudTrail, um serviço que fornece um registro das ações realizadas por um usuário, função ou AWS serviço em HealthLake. CloudTrail captura todas as

chamadas de API HealthLake como eventos. As chamadas capturadas incluem chamadas do HealthLake console e chamadas de código para as operações HealthLake da API. Se você criar uma trilha, poderá habilitar a entrega contínua de CloudTrail eventos para um bucket do Amazon S3, incluindo eventos para. HealthLake Se você não configurar uma trilha, ainda poderá ver os eventos mais recentes no CloudTrail console no Histórico de eventos. Usando as informações coletadas por CloudTrail, você pode determinar a solicitação que foi feita HealthLake, o endereço IP do qual a solicitação foi feita, quem fez a solicitação, quando ela foi feita e detalhes adicionais.

Para saber mais sobre isso CloudTrail, consulte o Guia AWS CloudTrail do usuário.

AWS HealthLake Informações em CloudTrail

CloudTrail é ativado em sua AWS conta quando você cria a conta. Quando a atividade ocorre em HealthLake, essa atividade é registrada em um CloudTrail evento junto com outros eventos AWS de serviço no histórico de eventos. É possível visualizar, pesquisar e baixar eventos recentes em sua AWS conta. Para obter mais informações, consulte Visualização de eventos com histórico de CloudTrail eventos.

Para um registro contínuo dos eventos em sua AWS conta, incluindo eventos para HealthLake, crie uma trilha. Uma trilha permite CloudTrail entregar arquivos de log para um bucket do Amazon S3. Por padrão, quando você cria uma trilha no console, ela é aplicada a todas as regiões da AWS. A trilha registra eventos de todas as regiões na AWS partição e entrega os arquivos de log ao bucket do Amazon S3 que você especificar. Além disso, você pode configurar outros AWS serviços para analisar e agir com base nos dados de eventos coletados nos CloudTrail registros. Para saber mais, consulte:

- Visão Geral para Criar uma Trilha
- CloudTrail Serviços e integrações compatíveis
- Configurando notificações do Amazon SNS para CloudTrail
- Recebendo arquivos de CloudTrail log de várias regiões e recebendo arquivos de CloudTrail log de várias contas

Todas as HealthLake ações são registradas CloudTrail e documentadas na Referência da HealthLake API e neste Guia do Desenvolvedor para ações realizadas usando a API FHIR REST. Por exemplo, chamadas para as ações a seguir geram entradas nos arquivos de CloudTrail log:

- `DescribeFHIRImportJob`
- `DescribeFHIRExportJob`

- `StartFHIRImportJob`
- `ListFHIRImportJobs`
- `StartFHIRExportJob`
- `ListFHIRExportJobs`
- `CreateFHIRDatastore`
- `ListFHIRDatastores`
- `DeleteFHIRDatastore`
- `DescribeFHIRDatastore`
- `UpdateResource`
- `CreateResource`
- `DeleteResource`
- `ReadResource`
- `GetCapabilities`
- `SearchWithGet`
- `SearchWithPost`

Cada entrada de log ou evento contém informações sobre quem gerou a solicitação. As informações de identidade ajudam a determinar o seguinte:

- Se a solicitação foi feita com credenciais de usuário root ou AWS Identity and Access Management (IAM).
- Se a solicitação foi feita com credenciais de segurança temporárias de um perfil ou de um usuário federado.
- Se a solicitação foi feita por outro AWS serviço.

Para obter mais informações, consulte Elemento userIdentity do CloudTrail .

Entendendo as entradas do arquivo de AWS HealthLake log

Uma trilha é uma configuração que permite a entrega de eventos como arquivos de log para um bucket do Amazon S3 que você especificar. CloudTrail os arquivos de log contêm uma ou mais entradas de log. Um evento representa uma única solicitação de qualquer fonte e inclui informações sobre a ação solicitada, a data e a hora da ação, os parâmetros da solicitação e assim por diante.

CloudTrail os arquivos de log não são um rastreamento de pilha ordenado das chamadas públicas de API, portanto, eles não aparecem em nenhuma ordem específica.

O exemplo a seguir mostra uma entrada de CloudTrail registro que demonstra a `CreateFHIRDatastore` ação.


```
{
    "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"
}
```


Monitoramento de HealthLake métricas usando a Amazon CloudWatch

Você pode monitorar HealthLake usando a Amazon CloudWatch, que coleta dados brutos e os processa em métricas legíveis, quase em tempo real. Essas estatísticas são mantidas por 15 meses, de maneira que você possa usar informações históricas e ter uma perspectiva melhor de como sua aplicação web ou o serviço está se saindo. Você também pode definir alarmes que observam determinados limites e enviam notificações ou realizam ações quando esses limites são atingidos. Para obter mais informações, consulte o Guia CloudWatch do usuário da Amazon.

> **Note**
>
> As métricas são relatadas para todas as HealthLakeações nativas.

As tabelas a seguir listam HealthLake as métricas e dimensões reportadas CloudWatch a. Cada uma é apresentada como uma contagem de frequência para um intervalo de datas especificado pelo usuário.

As HealthLake métricas a seguir são reportadas CloudWatch a.

HealthLake métricas reportadas para CloudWatch

Métrica	Description
Contagem de chamadas	O número de chamadas para APIs. Isso pode ser relatado para a conta ou para um armazenamento de dados especificado. Unidades: contagem Estatísticas válidas: Sum, Count Dimensões: operação, ID do armazenamento de dados, tipo de armazenamento de dados
Solicitações bem-sucedidas	O número de solicitações de API bem-suced idas. Unidades: contagem Estatísticas válidas: soma, média Dimensões: operação, ID do datastore, tipo de datastore
Erros do usuário	O número de solicitações que falharam devido a um erro do usuário. Unidades: contagem Estatísticas válidas: soma, média Dimensões: operação, ID do datastore, tipo de datastore
Erros de servidor	O número de solicitações que falharam devido a um erro do servidor. Unidades: contagem Estatísticas válidas: soma, média

Métrica	Description
	Dimensões: operação, ID do datastore, tipo de datastore
Solicitações limitadas	O número de solicitações que foram limitadas. Essa métrica não está incluída nas contagens de erros do usuário ou do servidor. Unidades: contagem Estatísticas válidas: soma, média Dimensões: operação, ID do datastore, tipo de datastore
Latência	O tempo gasto em milissegundos para processar a solicitação do usuário. Unidade: milissegundos Estatísticas válidas: mínimo, máximo, média Dimensões: operação, ID do datastore, tipo de datastore

As seguintes HealthLake dimensões são reportadas CloudWatch a.

HealthLake Dimensões relatadas a CloudWatch

Dimensão	Description
Operation	A operação de API usada na solicitação
DataStoreID	O ID do armazenamento de dados usado na solicitação
DataStoreType	O tipo de armazenamento de dados usado na solicitação

Você pode obter métricas HealthLake com o AWS Management Console AWS CLI, o ou a CloudWatch API. Você pode usar a CloudWatch API por meio de um dos kits de desenvolvimento de software da Amazon AWS (SDKs) ou das ferramentas de CloudWatch API. O HealthLake console exibe gráficos com base nos dados brutos da CloudWatch API.

Você deve ter as CloudWatch permissões apropriadas para monitorar HealthLake CloudWatch. Para obter mais informações, consulte Gerenciamento de identidade e acesso para a Amazon CloudWatch no Guia CloudWatch do usuário da Amazon.

Visualizando HealthLake métricas

Para visualizar métricas (CloudWatch console)

1. Faça login no Console de gerenciamento da AWS e abra o CloudWatchconsole.
2. Escolha Métricas, escolha Todas as métricas e, em seguida, escolha AWS/ HealthLake.
3. Escolha a dimensão, informe um nome de métrica e selecione Adicionar ao gráfico.
4. Escolha um valor para o intervalo de datas. A contagem da métrica para o intervalo de datas selecionado é exibida no gráfico.

Criando um alarme usando CloudWatch

Um CloudWatch alarme monitora uma única métrica durante um período de tempo especificado e executa uma ou mais ações: enviar uma notificação para um tópico do Amazon Simple Notification Service (SNS) ou política de Auto Scaling. A ação ou ações são baseadas no valor da métrica em relação a um determinado limite em vários períodos que você especifica. CloudWatch também pode enviar uma mensagem SNS quando o alarme muda de estado.

> **Note**
>
> CloudWatch os alarmes invocam ações somente quando o estado muda e persiste pelo período especificado.

Para visualizar métricas (CloudWatch console)

1. Faça login no console do CloudWatch.
2. Escolha Alarmes e, em seguida, Criar alarme.

3. Escolha AWS/ eHealthLake, em seguida, escolha uma métrica.
4. Para Intervalo de tempo, escolha um intervalo de tempo para monitorar e, em seguida, selecione Avançar.
5. Preencha os campos Nome e Descrição.
6. Em Whenever, escolha >= e digite um valor máximo.
7. Se você quiser CloudWatch enviar um e-mail quando o estado do alarme for atingido, na seção Ações, para Sempre que este alarme for atingido, escolha Estado é ALARME. Em Enviar notificação para, escolha uma lista de endereçamento ou escolha Nova lista e crie uma nova lista de endereçamento.
8. Visualize o alarme na seção Prévia do alarme. Se você estiver satisfeito com o alarme, selecione Criar alarme.

Monitorando HealthLake eventos usando a Amazon EventBridge

EventBridge A Amazon é um serviço sem servidor que usa eventos para conectar componentes do aplicativo, facilitando a criação de aplicativos escaláveis orientados por eventos. A base do EventBridge é criar regras que direcionem eventos para os alvos. AWS HealthLake fornece entrega durável de mudanças de estado para EventBridge. Para obter mais informações, consulte O que é a Amazon EventBridge? no Guia do EventBridge usuário da Amazon.

> **Note**
>
> Para saber como enviar HealthLake eventos para a Amazon EventBridge, consulte EventBridge Integração com a Amazon AWS HealthLake no blog AWS for Industries.

Tópicos

- HealthLake eventos enviados para EventBridge
- HealthLake estrutura do evento

HealthLake eventos enviados para EventBridge

A tabela a seguir lista todos os HealthLake eventos enviados EventBridge para processamento.

HealthLake tipo de evento	Estado
Eventos do armazenamento de dados	
Criação de armazenamento de dados	`CREATING`
Armazenamento de dados ativo	`ACTIVE`
Exclusão do armazenamento de dados	`DELETING`
Armazenamento de dados excluído	`DELETED`
Falha na criação do armazenamento de dados	`CREATE_FAILED`
Para obter mais informações, consulte `datastoreStatus` na Referência de APIs do AWS HealthLake .	
Importar eventos de trabalho	
Import Job enviado	`SUBMITTED`
Importar trabalho em andamento	`IN_PROGRESS`
Import Job concluído com erros	`COMPLETED_WITH_ERRORS`
Import Job concluído	`COMPLETED`
Falha no trabalho de importação	`FAILED`
Para obter mais informações, consulte `jobStatus` na Referência de APIs do AWS HealthLake .	
Exportar eventos de trabalho	
Export Job enviado	`SUBMITTED`
Exportar trabalho em andamento	`IN_PROGRESS`
Export Job concluído com erros	`COMPLETED_WITH_ERRORS`
Export Job concluído	`COMPLETED`

HealthLake tipo de evento	Estado
Falha no trabalho de exportação	`FAILED`

Para obter mais informações, consulte `jobStatus` na Referência de APIs do AWS HealthLake .

HealthLake estrutura do evento

HealthLake eventos são objetos com estrutura JSON que também contêm detalhes de metadados. Você pode usar os metadados como entrada para recriar um evento ou obter mais informações. Todos os campos de metadados associados estão listados em uma tabela abaixo dos exemplos de código nos menus a seguir. Para obter mais informações, consulte os metadados do evento de AWS serviço no Guia do EventBridge usuário da Amazon.

> **Note**
>
> Para saber como enviar HealthLake eventos para a Amazon EventBridge, consulte EventBridge Integração com a Amazon AWS HealthLake no blog AWS for Industries.

Eventos do armazenamento de dados

Data Store Creating

Estado - **`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

Estado - **`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

Estado - **`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

Estado - **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/"
```

```
    }
}
```

Eventos do armazenamento de dados - descrições de metadados

Nome	Tipo	Description
`version`	string	A versão EventBridge do esquema do evento.
`id`	string	O UUID da versão 4 gerado para cada evento.
`detail-type`	string	O tipo de evento que está sendo enviado.
`source`	string	Identifica o serviço que gerou o evento.
`account`	string	O ID de 12 dígitos da conta da AWS do proprietário do armazenamento de dados.
`time`	string	A hora em que o evento ocorreu.
`region`	string	Identifica a AWS região do armazenamento de dados.
`resources`	matriz (string)	Uma matriz JSON que contém o ARN do armazenamento de dados.
`detail`	objeto	Um objeto JSON contém informações sobre o evento.
`detail.datastoreId`	string	O ID do armazenamento de dados associado ao evento de alteração de status.

Nome	Tipo	Description
`detail.datastoreName`	string	O nome do armazenamento de dados.
`detail.datastoreTypeVersion`	string	A versão FHIR do armazenamento de dados.
`detail.datastoreEndpoint`	string	O endpoint do armazenamento de dados.

Importar eventos de trabalho

Import Job Submitted

Estado - **`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

Estado - **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

Estado - **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

Estado - **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

Estado - **`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/"
        }
    }
}
```

Importar eventos de trabalho - descrições de metadados

Nome	Tipo	Description
`version`	string	A versão EventBridge do esquema do evento.
`id`	string	O UUID da versão 4 gerado para cada evento.
`detail-type`	string	O tipo de evento que está sendo enviado.

Nome	Tipo	Description
`source`	string	Identifica o serviço que gerou o evento.
`account`	string	O ID de 12 dígitos da conta da AWS do proprietário do armazenamento de dados.
`time`	string	A hora em que o evento ocorreu.
`region`	string	Identifica a AWS região do armazenamento de dados.
`resources`	matriz (string)	Uma matriz JSON que contém o ARN do armazenamento de dados.
`detail`	objeto	Um objeto JSON contém informações sobre o evento.
`detail.jobId`	string	O ID do trabalho de importaçã o associado ao evento de mudança de status.
`detail.submitTime`	string	A hora em que o trabalho de importação foi enviado.
`detail.datastoreId`	string	O armazenamento de dados que gerou o evento de mudança de status.
`detail.inputDataCo nfig`	string	O caminho do prefixo de entrada para o bucket do Amazon S3 que contém os arquivos FHIR a serem importados.

Exportar eventos de trabalho

Export Job Submitted

Estado - **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

Estado - **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

Estado - **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

Estado - **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

Estado - **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/"
        }
    }
}
```

Exportar eventos de trabalho - descrições de metadados

Nome	Tipo	Description
`version`	string	A versão EventBridge do esquema do evento.
`id`	string	O UUID da versão 4 gerado para cada evento.
`detail-type`	string	O tipo de evento que está sendo enviado.
`source`	string	Identifica o serviço que gerou o evento.
`account`	string	O ID de 12 dígitos da conta da AWS do proprietário do armazenamento de dados.
`time`	string	A hora em que o evento ocorreu.

Nome	Tipo	Description
`region`	string	Identifica a AWS região do armazenamento de dados.
`resources`	matriz (string)	Uma matriz JSON que contém o ARN do armazenamento de dados.
`detail`	objeto	Um objeto JSON contém informações sobre o evento.
`detail.jobId`	string	O ID do trabalho de exportaçã o associado ao evento de alteração de status.
`detail.submitTime`	string	A hora em que o trabalho de exportação foi enviado.
`detail.datastoreId`	string	O armazenamento de dados que gerou o evento de mudança de status.
`detail.outputDataC onfig`	string	O caminho do prefixo de saída para o bucket do Amazon S3 que contém os arquivos FHIR a serem exportados.

Segurança em AWS HealthLake

A segurança na nuvem AWS é a maior prioridade. Como AWS cliente, você se beneficia de uma arquitetura de data center e rede criada para atender aos requisitos das organizações mais sensíveis à segurança.

A segurança é uma responsabilidade compartilhada entre você AWS e você. O modelo de responsabilidade compartilhada descreve isso como segurança da nuvem e segurança na nuvem:

- A segurança da nuvem AWS é responsável por proteger a infraestrutura que executa AWS os serviços na AWS nuvem. AWS também fornece serviços que você pode usar com segurança. Third-party auditores testam e verificam regularmente a eficácia de nossa segurança como parte dos Programas de Conformidade Programas de AWS de . Para saber mais sobre os programas de conformidade que se aplicam HealthLake, consulte Serviços da AWS no escopo do programa de conformidade .
- Segurança na nuvem — Sua responsabilidade é determinada pelo AWS serviço que você usa. Você também é responsável por outros fatores, incluindo a confidencialidade de seus dados, os requisitos da empresa e as leis e regulamentos aplicáveis.

Esta documentação ajuda você a entender como aplicar o modelo de responsabilidade compartilhada ao usar HealthLake. Os tópicos a seguir mostram como configurar para atender HealthLake aos seus objetivos de segurança e conformidade. Você também aprende a usar outros serviços da AWS que ajudam você a monitorar e proteger seus HealthLake recursos.

Tópicos

- Proteção de dados em AWS HealthLake
- Criptografia em REST para AWS HealthLake
- Criptografia em trânsito para AWS HealthLake
- Gerenciamento de identidade e acesso para AWS HealthLake
- Validação de conformidade AWS HealthLake
- Segurança da infraestrutura em AWS HealthLake
- Criando AWS HealthLake recursos com AWS CloudFormation
- AWS HealthLake e endpoints VPC de interface ()AWS PrivateLink
- Melhores práticas de segurança em AWS HealthLake

- Resiliência em AWS HealthLake

Proteção de dados em AWS HealthLake

O AWS modelo de responsabilidade compartilhada se aplica à proteção de dados no AWS HealthLake. Conforme descrito neste modelo, AWS é responsável por proteger a infraestrutura global que executa todos os Nuvem AWS. Você é responsável por manter o controle sobre o conteúdo hospedado nessa infraestrutura. Você também é responsável pelas tarefas de configuração e gerenciamento de segurança dos Serviços da AWS que usa. Para obter mais informações sobre privacidade de dados, consulte Perguntas frequentes sobre privacidade de dados . Para obter informações sobre proteção de dados na Europa, consulte o Centro de Regulamento Geral sobre a Proteção de Dados (RGPD).

Para fins de proteção de dados, recomendamos que você proteja Conta da AWS as credenciais e configure usuários individuais com Centro de Identidade do AWS IAM ou AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use uma autenticação multifator (MFA) com cada conta.
- Use SSL/TLS para se comunicar com AWS os recursos. Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Configure a API e o registro de atividades do usuário com AWS CloudTrail. Para obter informações sobre o uso de CloudTrail trilhas para capturar AWS atividades, consulte Como trabalhar com CloudTrail trilhas no Guia AWS CloudTrail do usuário.
- Use soluções de AWS criptografia, juntamente com todos os controles de segurança padrão Serviços da AWS.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados sensíveis armazenados no Amazon S3.
- Se você precisar de módulos criptográficos validados pelo FIPS 140-3 ao acessar AWS por meio de uma interface de linha de comando ou de uma API, use um endpoint FIPS. Para saber mais sobre os endpoints FIPS disponíveis, consulte Federal Information Processing Standard (FIPS) 140-3.

É altamente recomendável que nunca sejam colocadas informações confidenciais ou sensíveis, como endereços de e-mail de clientes, em tags ou campos de formato livre, como um campo Nome.

Isso inclui quando você trabalha com HealthLake ou Serviços da AWS usa o console, a API ou AWS os SDKs. AWS CLI Quaisquer dados inseridos em tags ou em campos de texto de formato livre usados para nomes podem ser usados para logs de faturamento ou de diagnóstico. Se você fornecer um URL para um servidor externo, é fortemente recomendável que não sejam incluídas informações de credenciais no URL para validar a solicitação nesse servidor.

Criptografia em REST para AWS HealthLake

HealthLake fornece criptografia por padrão para proteger dados confidenciais de clientes em repouso usando uma chave do AWS Key Management Service (AWS KMS) de propriedade do serviço. Customer-managed As chaves KMS também são suportadas e são necessárias para importar e exportar arquivos de um armazenamento de dados. Para saber mais sobre a chave Customer-managed KMS, consulte Amazon Key Management Service. Os clientes podem escolher uma chave KMS de propriedade da AWS ou uma chave Customer-managed KMS ao criar um armazenamento de dados. A configuração de criptografia não pode ser alterada após a criação de um armazenamento de dados. Se um armazenamento de dados estiver usando uma chave KMS de propriedade da AWS, ela será indicada como AWS_OWNED_KMS_KEY e você não verá a chave específica usada para criptografia em repouso.

Chave KMS de propriedade da AWS

HealthLake usa essas chaves por padrão para criptografar automaticamente informações potencialmente confidenciais, como dados de identificação pessoal ou dados de Informações de Saúde Privadas (PHI) em repouso. As chaves KMS de propriedade da AWS não são armazenadas em sua conta. Elas fazem parte de uma coleção de chaves do KMS que a AWS possui e gerencia para uso em várias contas da AWS. Os serviços da AWS podem usar chaves KMS de propriedade da AWS para proteger seus dados. Você não pode visualizar, gerenciar, usar chaves KMS de propriedade da AWS nem auditar seu uso. No entanto, você não precisa fazer nenhum trabalho nem alterar nenhum programa para proteger as chaves que criptografam seus dados.

Não é cobrada uma taxa mensal ou uma taxa de uso se você usar chaves do KMS de propriedade da AWS, e elas não contam nas cotas do AWS KMS para sua conta. Para obter mais informações, consulte Chaves de propriedade da AWS.

Chaves do KMS gerenciadas pelo cliente

HealthLake suporta o uso de uma chave KMS simétrica gerenciada pelo cliente que você cria, possui e gerencia para adicionar uma segunda camada de criptografia sobre a criptografia existente de

propriedade da AWS. Como você tem controle total dessa camada de criptografia, você pode realizar tarefas como:

- Estabelecer e manter políticas de chaves, políticas do IAM e concessões
- Rotacionar os materiais de chave de criptografia
- Habilitar e desabilitar políticas de chaves
- Adicionar etiquetas
- Criar réplicas de chaves
- Chaves de agendamento para exclusão

Você também pode usar CloudTrail para rastrear as solicitações HealthLake enviadas AWS KMS em seu nome. AWS KMS Taxas adicionais se aplicam. Para obter mais informações, consulte as chaves de propriedade do cliente.

Criar uma chave gerenciada pelo cliente

Você pode criar uma chave simétrica gerenciada pelo cliente usando o AWS Management Console ou as AWS KMS APIs.

Siga as etapas para criar uma chave simétrica gerenciada pelo cliente no Guia do desenvolvedor do AWS Key Management Service.

As políticas de chaves controlam o acesso à chave gerenciada pelo cliente. Cada chave gerenciada pelo cliente deve ter exatamente uma política de chaves, que contém declarações que determinam quem pode usar a chave e como pode usá-la. Ao criar a chave gerenciada pelo cliente, é possível especificar uma política de chave. Para obter mais informações, consulte Gerenciamento do acesso às chaves gerenciadas pelo cliente no Guia do desenvolvedor do AWS Key Management Service.

Para usar sua chave gerenciada pelo cliente com seus HealthLake recursos, CreateGrant as operações kms: devem ser permitidas na política de chaves. Isso adiciona uma concessão a uma chave gerenciada pelo cliente que controla o acesso a uma chave KMS especificada, que dá ao usuário acesso às operações kms:grant necessárias. HealthLake Consulte Uso de subsídios para obter mais informações.

Para usar sua chave KMS gerenciada pelo cliente com seus HealthLake recursos, as seguintes operações de API devem ser permitidas na política de chaves:

- kms: CreateGrant adiciona concessões a uma chave KMS específica gerenciada pelo cliente, que permite acesso às operações de concessão.
- kms: DescribeKey fornece os detalhes da chave gerenciada pelo cliente necessários para validar a chave. Isso é necessário para todas as operações.
- kms: GenerateDataKey fornece acesso para criptografar recursos em repouso para todas as operações de gravação.
- O KMS:Decrypt fornece acesso às operações de leitura ou pesquisa de recursos criptografados.

Veja a seguir um exemplo de declaração de política que permite que um usuário crie e interaja com um armazenamento de AWS HealthLake dados criptografado por essa chave:

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

Permissões do IAM obrigatórias para usar uma chave KMS gerenciada pelo cliente

Ao criar um armazenamento de dados com AWS KMS criptografia habilitada usando uma chave KMS gerenciada pelo cliente, há permissões necessárias para a política de chaves e a política do IAM para o usuário ou função que está criando o armazenamento de HealthLake dados.

Você pode usar a chave de ViaService condição kms: para limitar o uso da chave KMS somente às solicitações originadas de. HealthLake

Para obter mais informações sobre políticas de chaves, consulte Habilitar políticas do IAM no Guia do desenvolvedor do AWS Key Management Service.

O usuário do IAM, a função do IAM ou a conta da AWS que cria seus repositórios devem ter as permissões kms:CreateGrant, kms: e kms:GenerateDataKey, além das DescribeKey permissões necessárias. HealthLake

Como HealthLake usa subsídios no AWS KMS

HealthLake exige uma concessão para usar sua chave KMS gerenciada pelo cliente. Quando você cria um armazenamento de dados criptografado com uma chave KMS gerenciada pelo cliente, HealthLake cria uma concessão em seu nome enviando uma CreateGrantsolicitação para o AWS KMS. Os subsídios no AWS KMS são usados para dar HealthLake acesso a uma chave do KMS em uma conta de cliente.

Os subsídios HealthLake criados em seu nome não devem ser revogados ou retirados. Se você revogar ou retirar a concessão que dá HealthLake permissão para usar as chaves do AWS KMS em sua conta HealthLake , não poderá acessar esses dados, criptografar novos recursos do FHIR enviados para o armazenamento de dados ou descriptografá-los quando forem retirados. Quando você revoga ou retira um subsídio HealthLake, a alteração ocorre imediatamente. Para revogar os direitos de acesso, você deve excluir o armazenamento de dados em vez de revogar a concessão. Quando um armazenamento de dados é excluído, HealthLake as concessões são retiradas em seu nome.

Monitorando suas chaves de criptografia para HealthLake

Você pode usar CloudTrail para rastrear as solicitações HealthLake enviadas AWS KMS em seu nome ao usar uma chave KMS gerenciada pelo cliente. As entradas de registro no CloudTrail registro mostram healthlake.amazonaws.com no campo UserAgent para distinguir claramente as solicitações feitas por. HealthLake

Os exemplos a seguir são CloudTrail eventos para CreateGrant, GenerateDataKey, Decrypt e DescribeKey para monitorar AWS KMS operações chamadas por HealthLake para acessar dados criptografados pela chave gerenciada pelo cliente.

A seguir, mostramos como usar CreateGrant para permitir HealthLake o acesso à chave KMS fornecida pelo cliente, permitindo usar essa chave KMS HealthLake para criptografar todos os dados do cliente em repouso.

Os usuários não precisam criar suas próprias concessões. HealthLake cria uma concessão em seu nome enviando uma CreateGrant solicitação para o AWS KMS. As concessões AWS KMS são usadas para dar HealthLake acesso a uma AWS KMS chave na conta do cliente.


```
         {
"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"
}
```

Os exemplos a seguir mostram como usar GenerateDataKey para garantir que o usuário tenha as permissões necessárias para criptografar dados antes de armazená-los.

```
            {
    "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"
}
```

O exemplo a seguir mostra como HealthLake chama a operação Decrypt para usar a chave de dados criptografada armazenada para acessar os dados criptografados.

```
         {
    "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"
}
```

O exemplo a seguir mostra como HealthLake usa a DescribeKey operação para verificar se a AWS KMS chave de propriedade do AWS KMS cliente está em um estado utilizável e para ajudar o usuário a solucionar problemas se ela não estiver funcionando.

```
         {
    "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"
}
```


Saiba mais

Os recursos a seguir fornecem mais informações sobre criptografia de dados em repouso.

Para obter mais informações sobre os conceitos básicos do AWS Key Management Service, consulte a AWS KMS documentação.

Para obter mais informações sobre as melhores práticas de segurança na AWS KMS documentação.

Criptografia em trânsito para AWS HealthLake

AWS HealthLake usa o TLS 1.2 para criptografar dados em trânsito pelo endpoint público e por meio de serviços de back-end.

Gerenciamento de identidade e acesso para AWS HealthLake

AWS Identity and Access Management (IAM) é uma ferramenta AWS service (Serviço da AWS) que ajuda o administrador a controlar com segurança o acesso aos AWS recursos. Os administradores do IAM controlam quem pode ser autenticado (conectado) e autorizado (tem permissões) a usar HealthLake os recursos. O IAM é um AWS service (Serviço da AWS) que você pode usar sem custo adicional.

Tópicos

- Público
- Autenticação com identidades
- Gerenciar o acesso usando políticas
- Como AWS HealthLake funciona com o IAM
- Exemplos de políticas baseadas em identidade para AWS HealthLake
- AWS políticas gerenciadas para AWS HealthLake
- Solução de problemas AWS HealthLake de identidade e acesso

Público

A forma como você usa AWS Identity and Access Management (IAM) difere com base na sua função:

- Usuário do serviço: solicite permissões ao seu administrador se você não conseguir acessar os atributos (consulte Solução de problemas AWS HealthLake de identidade e acesso).
- Administrador do serviço: determine o acesso do usuário e envie solicitações de permissão (consulte Como AWS HealthLake funciona com o IAM)
- Administrador do IAM: escreva políticas para gerenciar o acesso (consulte Exemplos de políticas baseadas em identidade para AWS HealthLake)

Autenticação com identidades

A autenticação é a forma como você faz login AWS usando suas credenciais de identidade. Você deve estar autenticado como usuário do IAM ou assumindo uma função do IAM. Usuário raiz da conta da AWS

Você pode fazer login como uma identidade federada usando credenciais de uma fonte de identidade como Centro de Identidade do AWS IAM (IAM Identity Center), autenticação de login único ou credenciais. Google/Facebook Para ter mais informações sobre como fazer login, consulte Como fazer login em sua Conta da AWS no Guia do usuário do Início de Sessão da AWS .

Para acesso programático, AWS fornece um SDK e uma CLI para assinar solicitações criptograficamente. Para ter mais informações, consulte AWS Signature Version 4 para solicitações de API no Guia do usuário do IAM.

Conta da AWS usuário root

Ao criar um Conta da AWS, você começa com uma identidade de login chamada usuário Conta da AWS raiz que tem acesso completo a todos Serviços da AWS os recursos. É altamente recomendável não usar o usuário-raiz em tarefas diárias. Consulte as tarefas que exigem credenciais de usuário-raiz em Tarefas que exigem credenciais de usuário-raiz no Guia do usuário do IAM.

Identidade federada

Como prática recomendada, exija que os usuários humanos usem a federação com um provedor de identidade para acessar Serviços da AWS usando credenciais temporárias.

Uma identidade federada é um usuário do seu diretório corporativo, provedor de identidade da web ou Directory Service que acessa Serviços da AWS usando credenciais de uma fonte de identidade. As identidades federadas assumem funções que oferecem credenciais temporárias.

Para o gerenciamento de acesso centralizado, recomendamos Centro de Identidade do AWS IAM. Para saber mais, consulte O que é o IAM Identity Center? no Guia do usuário do Centro de Identidade do AWS IAM .

Usuários e grupos do IAM

Um usuário do IAM é uma identidade com permissões específicas para uma única pessoa ou aplicação. É recomendável usar credenciais temporárias, em vez de usuários do IAM com credenciais de longo prazo. Para obter mais informações, consulte Exigir que usuários humanos

usem a federação com um provedor de identidade para acessar AWS usando credenciais temporárias no Guia do usuário do IAM.

Um grupo do IAM especifica um conjunto de usuários do IAM e facilita o gerenciamento de permissões para grandes conjuntos de usuários. Para ter mais informações, consulte Casos de uso de usuários do IAM no Guia do usuário do IAM.

Perfis do IAM

Uma perfil do IAM é uma identidade com permissões específicas que oferece credenciais temporárias. Você pode assumir uma função mudando de um usuário para uma função do IAM (console) ou chamando uma operação de AWS API AWS CLI ou. Para saber mais, consulte Métodos para assumir um perfil no Manual do usuário do IAM.

Os perfis do IAM são úteis para acesso de usuário federado, permissões de usuário do IAM temporárias, acesso entre contas, acesso entre serviços e aplicações em execução no Amazon EC2. Consulte mais informações em Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Gerenciar o acesso usando políticas

Você controla o acesso AWS criando políticas e anexando-as a AWS identidades ou recursos. Uma política define permissões quando associada a uma identidade ou recurso. AWS avalia essas políticas quando um diretor faz uma solicitação. A maioria das políticas é armazenada AWS como documentos JSON. Para ter mais informações sobre documentos de política JSON, consulte Visão geral das políticas JSON no Guia do usuário do IAM.

Por meio de políticas, os administradores especificam quem tem acesso a que, definindo qual entidade principal pode realizar ações em quais recursos e sob quais condições.

Por padrão, usuários e perfis não têm permissões. Um administrador do IAM cria políticas do IAM e as adiciona aos perfis, os quais os usuários podem então assumir. As políticas do IAM definem permissões, independentemente do método usado para realizar a operação.

Políticas baseadas em identidade

As políticas baseadas em identidade são documentos de políticas de permissão JSON que você anexa a uma identidade (usuário, grupo ou perfil). Essas políticas controlam quais ações as identidades podem realizar, em quais recursos e sob quais condições. Para saber como criar uma política baseada em identidade, consulte Definir permissões personalizadas do IAM com as políticas gerenciadas pelo cliente no Guia do Usuário do IAM.

As políticas baseadas em identidade podem ser políticas em linha (incorporadas diretamente em uma única identidade) ou políticas gerenciadas (políticas autônomas anexadas a várias identidades). Para saber como escolher entre uma política gerenciada e políticas em linha, consulte Escolher entre políticas gerenciadas e políticas em linha no Guia do usuário do IAM.

Políticas baseadas em recursos

Políticas baseadas em recursos são documentos de políticas JSON que você anexa a um recurso. Entre os exemplos estão políticas de confiança de perfil do IAM e políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. É necessário especificar uma entidade principal em uma política baseada em recursos.

Políticas baseadas em recursos são políticas em linha localizadas nesse serviço. Você não pode usar políticas AWS gerenciadas do IAM em uma política baseada em recursos.

Outros tipos de política

AWS oferece suporte a tipos de políticas adicionais que podem definir o máximo de permissões concedidas por tipos de políticas mais comuns:

- Limites de permissões: definem o número máximo de permissões que uma política baseada em identidade pode conceder a uma entidade do IAM. Para saber mais sobre limites de permissões, consulte Limites de permissões para identidades do IAM no Guia do usuário do IAM.
- Políticas de controle de serviço (SCPs) — Especifique as permissões máximas para uma organização ou unidade organizacional em AWS Organizations. Para saber mais, consulte Políticas de controle de serviço no Guia do usuário do AWS Organizations .
- Políticas de controle de recursos (RCPs) — Defina o máximo de permissões disponíveis para recursos em suas contas. Para obter mais informações, consulte Políticas de controle de recursos (RCPs) no Guia AWS Organizations do usuário.
- Políticas de sessão: políticas avançadas transmitidas como um parâmetro durante a criação de uma sessão temporária para um perfil ou um usuário federado. Para saber mais, consulte Políticas de sessão no Guia do usuário do IAM.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como AWS determinar se uma solicitação deve ser permitida

quando vários tipos de políticas estão envolvidos, consulte Lógica de avaliação de políticas no Guia do usuário do IAM.

Como AWS HealthLake funciona com o IAM

Antes de usar o IAM para gerenciar o acesso HealthLake, saiba com quais recursos do IAM estão disponíveis para uso HealthLake.

Recursos do IAM que você pode usar com AWS HealthLake

Recurso do IAM	HealthLake apoio
Políticas baseadas em identidade	Sim
Políticas baseadas em recurso	Não
Ações de políticas	Sim
Recursos de políticas	Sim
Chaves de condição de políticas	Sim
ACLs	Não
ABAC (tags em políticas)	Sim
Credenciais temporárias	Sim
Permissões de entidade principal	Sim
Perfis de serviço	Sim
Perfis vinculados a serviço	Não

Para ter uma visão de alto nível de como HealthLake e outros AWS serviços funcionam com a maioria dos recursos do IAM, consulte AWS os serviços que funcionam com o IAM no Guia do usuário do IAM.

Políticas baseadas em identidade para AWS HealthLake

Compatível com políticas baseadas em identidade: sim

As políticas baseadas em identidade são documentos de políticas de permissões JSON que podem ser anexados a uma identidade, como usuário do IAM, grupo de usuários ou perfil. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Definir permissões personalizadas do IAM com as políticas gerenciadas pelo cliente no Guia do Usuário do IAM.

Com as políticas baseadas em identidade do IAM, é possível especificar ações e recursos permitidos ou negados, assim como as condições sob as quais as ações são permitidas ou negadas. Para saber mais sobre todos os elementos que podem ser usados em uma política JSON, consulte Referência de elemento de política JSON do IAM no Guia do usuário do IAM.

Exemplos de políticas baseadas em identidade para AWS HealthLake

Para ver exemplos de políticas HealthLake baseadas em identidade, consulte. Exemplos de políticas baseadas em identidade para AWS HealthLake

Políticas baseadas em recursos dentro AWS HealthLake

Compatibilidade com políticas baseadas em recursos: não

Políticas baseadas em recursos são documentos de políticas JSON que você anexa a um recurso. São exemplos de políticas baseadas em recursos as políticas de confiança de perfil do IAM e as políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o atributo ao qual a política está anexada, a política define quais ações uma entidade principal especificado pode executar nesse atributo e em que condições. É necessário especificar uma entidade principal em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou. Serviços da AWS

Para permitir o acesso entre contas, é possível especificar uma conta inteira ou as entidades do IAM em outra conta como a entidade principal em uma política baseada em recursos. Consulte mais informações em Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Ações políticas para AWS HealthLake

Compatível com ações de políticas: sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Action` de uma política JSON descreve as ações que podem ser usadas para permitir ou negar acesso em uma política. Incluem ações em uma política para conceder permissões para executar a operação associada.

Para ver uma lista de HealthLake ações, consulte Ações definidas por AWS HealthLake na Referência de Autorização de Serviço.

As ações de política HealthLake usam o seguinte prefixo antes da ação:

```
healthlake
```

Para especificar várias ações em uma única declaração, separe cada ação com uma vírgula.

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

Para ver exemplos de políticas HealthLake baseadas em identidade, consulte. Exemplos de políticas baseadas em identidade para AWS HealthLake

Recursos políticos para AWS HealthLake

Compatível com recursos de políticas: sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento de política JSON `Resource` especifica o objeto ou os objetos aos quais a ação se aplica. Como prática recomendada, especifique um recurso usando seu nome do recurso da Amazon (ARN). Para ações que não oferecem compatibilidade com permissões em nível de recurso, use um curinga (*) para indicar que a instrução se aplica a todos os recursos.

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

Para ver uma lista dos tipos de HealthLake recursos e seus ARNs, consulte Recursos definidos por AWS HealthLake na Referência de Autorização de Serviço. Para saber as ações com as quais você pode especificar o ARN de cada recurso, consulte Ações definidas por. AWS HealthLake

Para ver exemplos de políticas HealthLake baseadas em identidade, consulte. Exemplos de políticas baseadas em identidade para AWS HealthLake

Chaves de condição de política para AWS HealthLake

Compatível com chaves de condição de política específicas de serviço: sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Condition` especifica quando as instruções são executadas com base em critérios definidos. É possível criar expressões condicionais que usem agentes de condição, como “igual a” ou “menor que”, para fazer a condição da política corresponder aos valores na solicitação. Para ver todas as chaves de condição AWS globais, consulte as chaves de contexto de condição AWS global no Guia do usuário do IAM.

Para ver uma lista de chaves de HealthLake condição, consulte Chaves de condição AWS HealthLake na Referência de autorização de serviço. Para conhecer as ações e os recursos com os quais você pode usar uma chave de condição, consulte Ações definidas por AWS HealthLake.

Para ver exemplos de políticas HealthLake baseadas em identidade, consulte. Exemplos de políticas baseadas em identidade para AWS HealthLake

Listas de controle de acesso (ACLs) em AWS HealthLake

Suportes ACLs: Não

As listas de controle de acesso (ACLs) controlam quais diretores (membros da conta, usuários ou funções) têm permissões para acessar um recurso. ACLs são semelhantes às políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

Controle de acesso baseado em atributos (ABAC) com AWS HealthLake

Compatível com ABAC (tags em políticas): sim

O controle de acesso por atributo (ABAC) é uma estratégia de autorização que define permissões com base em atributos chamados de tags. Você pode anexar tags a entidades e AWS recursos do IAM e, em seguida, criar políticas ABAC para permitir operações quando a tag do diretor corresponder à tag no recurso.

Para controlar o acesso baseado em tags, forneça informações sobre as tags no elemento de condição de uma política usando as `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` ou chaves de condição `aws:TagKeys`.

Se um serviço for compatível com as três chaves de condição para cada tipo de recurso, o valor será Sim para o serviço. Se um serviço for compatível com as três chaves de condição somente para alguns tipos de recursos, o valor será Parcial

Para saber mais sobre o ABAC, consulte Definir permissões com autorização do ABAC no Guia do usuário do IAM. Para visualizar um tutorial com etapas para configurar o ABAC, consulte Usar controle de acesso por atributo (ABAC) no Guia do usuário do IAM.

Usando credenciais temporárias com AWS HealthLake

Compatível com credenciais temporárias: sim

As credenciais temporárias fornecem acesso de curto prazo aos AWS recursos e são criadas automaticamente quando você usa a federação ou troca de funções. AWS recomenda que você gere credenciais temporárias dinamicamente em vez de usar chaves de acesso de longo prazo. Para ter mais informações, consulte Credenciais de segurança temporárias no IAM e Serviços da Serviços da AWS que funcionam com o IAM no Guia do usuário do IAM.

Permissões principais entre serviços para AWS HealthLake

Compatibilidade com o recurso de encaminhamento de sessões de acesso (FAS): sim

As sessões de acesso direto (FAS) usam as permissões do principal chamando um AWS service (Serviço da AWS), combinadas com a solicitação AWS service (Serviço da AWS) de fazer solicitações aos serviços posteriores. Para obter detalhes da política ao fazer solicitações de FAS, consulte Sessões de acesso direto.

Funções de serviço para AWS HealthLake

Compatível com perfis de serviço: sim

O perfil de serviço é um perfil do IAM que um serviço assume para executar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de serviço do IAM. Para saber mais, consulte Criar um perfil para delegar permissões a um AWS service (Serviço da AWS) no Guia do Usuário do IAM.

Para obter informações sobre funções de serviço e a política em linha necessária para acesso total AWS HealthLake, consulteConfigurar AWS HealthLake.

> **Warning**
>
> Alterar as permissões de uma função de serviço pode interromper HealthLake a funcionalidade. Edite as funções de serviço somente quando HealthLake fornecer orientação para fazer isso.

Funções vinculadas a serviços para AWS HealthLake

Compatível com perfis vinculados ao serviço: Não

Uma função vinculada ao serviço é um tipo de função de serviço vinculada a um. AWS service (Serviço da AWS) O serviço pode assumir o perfil de executar uma ação em seu nome. As funções vinculadas ao serviço aparecem em você Conta da AWS e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não editar as permissões para perfis vinculados ao serviço.

Para obter detalhes sobre como criar ou gerenciar perfis vinculados a serviços, consulte Serviços da AWS que funcionam com o IAM. Encontre um serviço na tabela que inclua um `Yes` na coluna Perfil vinculado ao serviço. Escolha o link Sim para visualizar a documentação do perfil vinculado a serviço desse serviço.

Exemplos de políticas baseadas em identidade para AWS HealthLake

Por padrão, usuários e perfis não têm permissão para criar ou modificar recursos do HealthLake. Para conceder permissão aos usuários para executar ações nos recursos que eles precisam, um administrador do IAM pode criar políticas do IAM.

Para aprender a criar uma política baseada em identidade do IAM ao usar esses documentos de política em JSON de exemplo, consulte Criar políticas do IAM (console) no Guia do usuário do IAM.

Para obter detalhes sobre ações e tipos de recursos definidos por HealthLake, incluindo o formato do ARNs para cada um dos tipos de recursos, consulte Ações, recursos e chaves de condição AWS HealthLake na Referência de Autorização de Serviço.

Tópicos

- Práticas recomendadas de política
- Usando o AWS HealthLake console
- Acessando um armazenamento AWS HealthLake de dados em Amazon Athena
- Permitir que os usuários visualizem suas próprias permissões

Práticas recomendadas de política

As políticas baseadas em identidade determinam se alguém pode criar, acessar ou excluir HealthLake recursos em sua conta. Essas ações podem incorrer em custos para sua Conta da AWS. Ao criar ou editar políticas baseadas em identidade, siga estas diretrizes e recomendações:

- Comece com as políticas AWS gerenciadas e avance para as permissões de privilégios mínimos — Para começar a conceder permissões aos seus usuários e cargas de trabalho, use as políticas AWS gerenciadas que concedem permissões para muitos casos de uso comuns. Eles estão disponíveis no seu Conta da AWS. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo AWS cliente que sejam específicas para seus casos de uso. Para saber mais, consulte Políticas gerenciadas pela AWS ou Políticas gerenciadas pela AWS para funções de trabalho no Guia do usuário do IAM.
- Aplique permissões de privilégio mínimo: ao definir permissões com as políticas do IAM, conceda apenas as permissões necessárias para executar uma tarefa. Você faz isso definindo as ações que podem ser executadas em recursos específicos sob condições específicas, também conhecidas como permissões de privilégio mínimo. Para saber mais sobre como usar o IAM para aplicar permissões, consulte Políticas e permissões no IAM no Guia do usuário do IAM.
- Use condições nas políticas do IAM para restringir ainda mais o acesso: é possível adicionar uma condição às políticas para limitar o acesso a ações e recursos. Por exemplo, é possível escrever uma condição de política para especificar que todas as solicitações devem ser enviadas usando SSL. Você também pode usar condições para conceder acesso às ações de serviço se elas forem usadas por meio de uma ação específica AWS service (Serviço da AWS), como CloudFormation. Para saber mais, consulte Elementos da política JSON do IAM: condição no Guia do usuário do IAM.
- Use o IAM Access Analyzer para validar suas políticas do IAM a fim de garantir permissões seguras e funcionais: o IAM Access Analyzer valida as políticas novas e existentes para que elas sigam a linguagem de política do IAM (JSON) e as práticas recomendadas do IAM. O IAM Access Analyzer oferece mais de cem verificações de política e recomendações práticas para ajudar a criar políticas seguras e funcionais. Para saber mais, consulte Validação de políticas do IAM Access Analyzer no Guia do Usuário do IAM.

- Exigir autenticação multifator (MFA) — Se você tiver um cenário que exija usuários do IAM ou um usuário root, ative Conta da AWS a MFA para obter segurança adicional. Para exigir MFA quando as operações de API forem chamadas, adicione condições de MFA às suas políticas. Para saber mais, consulte Configuração de acesso à API protegido por MFA no Guia do Usuário do IAM.

Para saber mais sobre as práticas recomendadas do IAM, consulte Práticas recomendadas de segurança no IAM no Guia do usuário do IAM.

Usando o AWS HealthLake console

Para acessar o AWS HealthLake console, você deve ter um conjunto mínimo de permissões. Essas permissões devem permitir que você liste e visualize detalhes sobre os HealthLake recursos em seu Conta da AWS. Caso crie uma política baseada em identidade mais restritiva que as permissões mínimas necessárias, o console não funcionará como pretendido para entidades (usuários ou perfis) com essa política.

Você não precisa permitir permissões mínimas do console para usuários que estão fazendo chamadas somente para a API AWS CLI ou para a AWS API. Em vez disso, permita o acesso somente a ações que correspondam à operação de API que estiverem tentando executar.

Para obter acesso total HealthLake, anexe as seguintes políticas a um usuário ou função do IAM: `AmazonHealthLakeFullAccess` `AWSLakeFormationDataAdmin` e. Você também precisa anexar a política HealthLake embutida, que é uma função de serviço. O perfil de serviço é um perfil do IAM que um serviço assume para executar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de serviço do IAM. Para saber mais, consulte Criar um perfil para delegar permissões a um AWS service (Serviço da AWS) no Guia do Usuário do IAM. Para obter informações sobre a política em linha que cria a função de serviço necessária, consulteConfigurar AWS HealthLake. Você também deve usar o AWS Lake Formation console ou a CLI para designar seu HealthLake administrador como administrador do AWS Lake Formation Data Lake. Para obter mais informações, consulte Configurar AWS HealthLake.

Acessando um armazenamento AWS HealthLake de dados em Amazon Athena

Se você quiser fornecer aos usuários e funções acesso aos armazenamentos de HealthLake dados em Amazon Athena, anexe as seguintes políticas do IAM à função ou ao usuário: `AmazonAthenaFullAccess` `AmazonS3FullAccess` e. `Selecte` `Describe` as permissões também são necessárias em tabelas gerenciadas pelo AWS Lake Formation. AWS Lake Formation as permissões de tabela são concedidas por um AWS Lake Formation administrador no AWS Lake

Formation console ou por meio da CLI. Para obter mais informações, consulte Configurar AWS HealthLake.

Permitir que os usuários visualizem suas próprias permissões

Este exemplo mostra como criar uma política que permita que os usuários do IAM visualizem as políticas gerenciadas e em linha anexadas a sua identidade de usuário. Essa política inclui permissões para concluir essa ação no console ou programaticamente usando a API AWS CLI ou 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 políticas gerenciadas para AWS HealthLake

Uma política AWS gerenciada é uma política autônoma criada e administrada por AWS. AWS as políticas gerenciadas são projetadas para fornecer permissões para muitos casos de uso comuns, para que você possa começar a atribuir permissões a usuários, grupos e funções.

Lembre-se de que as políticas AWS gerenciadas podem não conceder permissões de privilégio mínimo para seus casos de uso específicos porque estão disponíveis para uso de todos os AWS clientes. Recomendamos que você reduza ainda mais as permissões definindo as políticas gerenciadas pelo cliente que são específicas para seus casos de uso.

Você não pode alterar as permissões definidas nas políticas AWS gerenciadas. Se AWS atualizar as permissões definidas em uma política AWS gerenciada, a atualização afetará todas as identidades principais (usuários, grupos e funções) às quais a política está anexada. AWS é mais provável que atualize uma política AWS gerenciada quando uma nova AWS service (Serviço da AWS) é lançada ou novas operações de API são disponibilizadas para serviços existentes.

Para saber mais, consulte AWS Políticas gerenciadas pela no Guia do usuário do IAM.

AWS política gerenciada: AmazonHealthLakeFullAccess

A `AmazonHealthLakeFullAccess` política fornece acesso total HealthLake a. Com essa política anexada ao usuário ou função, os usuários podem HealthLake usá-la para acessar, consultar, importar e exportar dados em HealthLake. Para realizar muitas ações comuns em HealthLake, você deve adicionar políticas adicionais ao usuário ou à função. Para obter mais informações, consulte HealthLake operações Configurar AWS HealthLake e permissões.

É possível anexar a política `AmazonHealthLakeFullAccess` às suas identidades do IAM.

Essa política concede permissões administrativas e de colaboradores que permitem que usuários e funções consultem, pesquisem, importem e exportem HealthLake, além de possibilitar HealthLake a execução de ações em nome dos usuários e funções que têm essas permissões.

Detalhes das permissões

Essa política inclui a seguinte declaração.

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 política gerenciada: AmazonHealthLakeReadOnlyAccess

`AmazonHealthLakeReadOnlyAccess`a política concede acesso e permissões somente para leitura HealthLake e recursos relacionados em outros AWS serviços. Aplique essa política aos usuários aos quais você deseja conceder a capacidade de consultar e visualizar o armazenamento de HealthLake dados, mas não a capacidade de criar ou fazer alterações neles.

É possível anexar a política `AmazonHealthLakeReadOnlyAccess` às suas identidades do IAM.

Essa política concede *`read-only`* permissões que permitem que usuários e funções consultem HealthLake.

Detalhes das permissões

Essa política inclui a seguinte declaração.

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 operações e permissões

A tabela a seguir lista as operações típicas HealthLake e as permissões necessárias para executá-las.

HealthLake operações	Permissões obrigatórias
Crie um armazenamento de dados em HealthLake	`AmazonHealthLakeFullAccess` ,`AmazonLakeFormationDataAdmin` , política em linha e permissões de AWS Lake Formation administrador gerenciadas por AWS Lake Formation
Excluir um armazenamento de dados no HealthLake	`AmazonHealthLakeFullAccess` ,`AmazonLakeFormationDataAdmin` , política em linha e permissões de AWS Lake Formation administrador gerenciadas por AWS Lake Formation
Listar, pesquisar ou consultar um armazenamento de dados no HealthLake	`AmazonHealthLakeReadOnlyAccess`
Consulte um armazenamento de dados usando Amazon Athena	`AmazonAthenaFullAccess` ,`AmazonS3FullAccess` , AWS Lake Formation `Select` e `Describe` permissões em tabelas gerenciadas pelo AWS Lake Formation
Importar dados de HealthLake	Consulte Configurando permissões para trabalhos de importação.
Exportar dados de HealthLake	Consulte Configurando permissões para trabalhos de exportação.

HealthLake atualizações nas políticas AWS gerenciadas

Veja detalhes sobre as atualizações das políticas AWS gerenciadas a HealthLake partir do momento em que esse serviço começou a rastrear essas alterações. Para receber alertas automáticos sobre alterações nessa página, assine o feed RSS na página Histórico do HealthLake documento.

Alteração	Descrição	Data
AmazonHealthLakeFullAccess	`AmazonHealthLakeFullAccess` política necessária para permitir acesso total HealthLake a.	14 de novembro de 2022
AmazonHealthLakeReadOnlyAccess	`AmazonHealthLakeReadOnlyAccess` política necessária para acesso somente para leitura a. HealthLake	14 de novembro de 2022
HealthLake começou a rastrear alterações	HealthLake começou a rastrear as mudanças em suas políticas AWS gerenciadas.	14 de novembro de 2022

Solução de problemas AWS HealthLake de identidade e acesso

Use as informações a seguir para ajudá-lo a diagnosticar e corrigir problemas comuns que você pode encontrar ao trabalhar com HealthLake um IAM.

Tópicos

- Não estou autorizado a realizar uma ação em AWS HealthLake
- Não estou autorizado a realizar iam: PassRole
- Quero permitir que pessoas fora da minha AWS conta acessem meus AWS HealthLake recursos

Não estou autorizado a realizar uma ação em AWS HealthLake

Se isso Console de gerenciamento da AWS indicar que você não está autorizado a realizar uma ação, entre em contato com o administrador para obter ajuda. O administrador é a pessoa que forneceu o seu nome de usuário e senha.

O erro do exemplo a seguir ocorre quando o usuário do IAM `mateojackson` tenta usar o console para visualizar detalhes sobre um recurso do *`my-example-widget`* fictício, mas não tem as permissões fictícias do `healthlake:`*`GetWidget`*.

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

Neste caso, Mateo pede ao administrador para atualizar suas políticas para permitir a ele o acesso ao recurso *`my-example-widget`* usando a ação `healthlake:`*`GetWidget`*.

Não estou autorizado a realizar iam: PassRole

Se você receber uma mensagem de erro informando que não está autorizado a executar a ação `iam:PassRole`, as suas políticas devem ser atualizadas para permitir que você passe uma função para o HealthLake.

Alguns Serviços da AWS permitem que você passe uma função existente para esse serviço em vez de criar uma nova função de serviço ou uma função vinculada ao serviço. Para fazê-lo, você deve ter permissões para passar o perfil para o serviço.

O exemplo de erro a seguir ocorre quando uma usuária do IAM chamada `marymajor` tenta utilizar o console para executar uma ação no HealthLake. No entanto, a ação exige que o serviço tenha permissões concedidas por um perfil de serviço. Mary não tem permissões para passar o perfil para o serviço.

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

Nesse caso, as políticas de Mary devem ser atualizadas para permitir que ela realize a ação `iam:PassRole`.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

Quero permitir que pessoas fora da minha AWS conta acessem meus AWS HealthLake recursos

É possível criar um perfil que os usuários de outras contas ou pessoas fora da organização podem usar para acessar seus recursos. É possível especificar quem é confiável para assumir o perfil. Para

serviços que oferecem suporte a políticas baseadas em recursos ou listas de controle de acesso (ACLs), você pode usar essas políticas para conceder às pessoas acesso aos seus recursos.

Para saber mais, consulte:

- Para saber se é HealthLake compatível com esses recursos, consulteComo AWS HealthLake funciona com o IAM.
- Para saber como fornecer acesso aos seus recursos em todos os Contas da AWS que você possui, consulte Como fornecer acesso a um usuário do IAM em outro Conta da AWS que você possui no Guia do usuário do IAM.
- Para saber como fornecer acesso aos seus recursos a terceiros Contas da AWS, consulte Como fornecer acesso Contas da AWS a terceiros no Guia do usuário do IAM.
- Para saber como conceder acesso por meio da federação de identidades, consulte Conceder acesso a usuários autenticados externamente (federação de identidades) no Guia do usuário do IAM.
- Para conhecer a diferença entre perfis e políticas baseadas em recurso para acesso entre contas, consulte Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Validação de conformidade AWS HealthLake

Third-party os auditores avaliam a segurança e a conformidade AWS HealthLake como parte de vários programas de AWS conformidade. Para HealthLake isso, inclui o HIPAA.

Para saber se um AWS service (Serviço da AWS) está dentro do escopo de programas de conformidade específicos, consulte Serviços da AWS Escopo por Programa de Conformidade Serviços da AWS e escolha o programa de conformidade em que você está interessado. Para obter informações gerais, consulte Programas de AWS conformidade Programas AWS de .

Você pode baixar relatórios de auditoria de terceiros usando AWS Artifact. Para obter mais informações, consulte Baixar relatórios em AWS Artifact .

Sua responsabilidade de conformidade ao usar Serviços da AWS é determinada pela confidencialidade de seus dados, pelos objetivos de conformidade de sua empresa e pelas leis e regulamentações aplicáveis. Para obter mais informações sobre sua responsabilidade de conformidade ao usar Serviços da AWS, consulte a documentação AWS de segurança.

Segurança da infraestrutura em AWS HealthLake

Como serviço gerenciado, AWS HealthLake é protegido pelos procedimentos AWS globais de segurança de rede descritos no whitepaper Amazon Web Services: Visão geral dos processos de segurança.

Você usa chamadas de API AWS publicadas para acessar HealthLake pela rede. Os clientes devem oferecer suporte a Transport Layer Security (TLS) 1.0 ou posterior. Recomendamos usar o TLS 1.2 ou posterior. Os clientes também devem oferecer suporte a pacotes de criptografia com sigilo direto perfeito (PFS), como Ephemeral (DHE) ou Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Diffie-Hellman A maioria dos sistemas modernos, como Java 7 e versões posteriores, comporta esses modos.

Além disso, as solicitações devem ser assinadas usando um ID da chave de acesso e uma chave de acesso secreta associada a uma entidade principal do IAM. Ou é possível usar o AWS Security Token Service (AWS STS) para gerar credenciais de segurança temporárias para assinar solicitações.

Criando AWS HealthLake recursos com AWS CloudFormation

AWS HealthLake é integrado com AWS CloudFormation, um serviço que ajuda você a modelar e configurar seus AWS recursos para que você possa gastar menos tempo criando e gerenciando seus recursos e infraestrutura. Você cria um modelo que descreve todos os AWS recursos que você deseja, CloudFormation provisiona e configura esses recursos para você.

Ao usar CloudFormation, você pode reutilizar seu modelo para configurar seus HealthLake recursos de forma consistente e repetida. Descreva seus recursos uma vez e, em seguida, provisione os mesmos recursos repetidamente em várias Contas da AWS regiões.

HealthLake e CloudFormation modelos

Para provisionar e configurar recursos HealthLake e serviços relacionados, você deve entender CloudFormation os modelos. Os modelos são arquivos de texto formatados em JSON ou YAML. Esses modelos descrevem os recursos que você deseja provisionar em suas CloudFormation pilhas. Se você não estiver familiarizado com JSON ou YAML, você pode usar o CloudFormation Designer para ajudá-lo a começar a usar modelos. CloudFormation Para obter mais informações, consulte O que é o Designer CloudFormation ? no Manual do usuário do AWS CloudFormation .

> **Note**
>
> AWS HealthLake suporta a criação de armazenamentos de dados com CloudFormation. Para obter mais informações, incluindo exemplos de modelos JSON e YAML para provisionar armazenamentos de HealthLake dados, consulte a referência do tipo de AWS HealthLake recurso no Guia do usuário.AWS CloudFormation

Saiba mais sobre CloudFormation

Para saber mais sobre isso CloudFormation, consulte os seguintes recursos:

- AWS CloudFormation
- AWS CloudFormation Guia do usuário
- Referência de API do CloudFormation
- AWS CloudFormation Guia do usuário da interface de linha de comando

AWS HealthLake e endpoints VPC de interface ()AWS PrivateLink

Você pode estabelecer uma conexão privada entre sua VPC e criar uma AWS HealthLake interface VPC endpoint. Os endpoints VPC de interface são alimentados por AWS PrivateLinkuma tecnologia que você pode usar para acessar de forma privada HealthLake; APIs sem um gateway de internet, dispositivo NAT, conexão VPN ou conexão. Direct Connect As instâncias em sua VPC não precisam de endereços IP públicos para se comunicar com HealthLake;. APIs O tráfego entre sua VPC e HealthLake; não sai da rede Amazon.

Cada endpoint de interface é representado por uma ou mais Interfaces de Rede Elástica nas sub-redes.

Para obter mais informações, consulte Interface VPC endpoints (AWS PrivateLink) no Guia do usuário da Amazon VPC.

Considerações sobre HealthLake VPC endpoints

Antes de configurar uma interface para o VPC endpoint HealthLake, certifique-se de revisar as propriedades e limitações do endpoint da interface no Guia do usuário do Amazon VPC.

HealthLake suporta fazer chamadas para todas as suas ações de API a partir de sua VPC.

Criação de uma interface VPC endpoint para; HealthLake

Você pode criar um VPC endpoint para o serviço HealthLake; usando o console Amazon VPC ou o (). AWS Command Line Interface AWS CLI Para obter mais informações, consulte Criar um endpoint de interface no Guia do usuário da Amazon VPC.

Crie um VPC endpoint para HealthLake; usando o seguinte nome de serviço:

- com.amazonaws. *region*.lago de saúde

Se você ativar o DNS privado para o endpoint, poderá fazer solicitações de API HealthLake usando seu nome DNS padrão para a região. Por exemplo, *.healthlake.us-east-1.amazonaws.com*

Para mais informações, consulte Acessar um serviço por um endpoint de interface no Guia do usuário da Amazon VPC.

Criação de uma política de VPC endpoint para HealthLake

É possível anexar uma política de endpoint ao endpoint da VPC que controla o acesso ao HealthLake. Essa política especifica as seguintes informações:

- A entidade principal que pode realizar ações.
- As ações que podem ser realizadas.
- Os recursos aos quais as ações podem ser aplicadas.

Para mais informações, consulte Controlar o acesso a serviços com VPC endpoints no Guia do usuário da Amazon VPC.

Exemplo: política de VPC endpoint para ações HealthLake

Veja a seguir um exemplo de uma política de endpoint para HealthLake. Quando anexada a um endpoint, essa política concede acesso à HealthLake `CreateFHIRDatastore` ação para todos os diretores em todos os recursos.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
```

```
            "Action":[
               "healthlake:create-fhir-datastore"
            ],
            "Resource":"*"
         }
      ]
}
```

Melhores práticas de segurança em AWS HealthLake

AWS HealthLake fornece vários recursos de segurança a serem considerados ao desenvolver e implementar suas próprias políticas de segurança. As práticas recomendadas a seguir são diretrizes gerais e não representam uma solução completa de segurança. Como essas práticas recomendadas podem não ser adequadas ou suficientes para o seu ambiente, trate-as como considerações úteis em vez de prescrições.

- Implemente o acesso de privilégio mínimo.
- Sempre que possível, use Customer-Managed-Keys (CMKs) para criptografar seus dados. Para saber mais sobre CMKs, consulte Amazon Key Management Service.
- Use Pesquisar com POST, não Pesquisar com GET ao consultar PHI ou PII em seu armazenamento de dados.
- Limite o acesso a funções de auditoria importantes e confidenciais.
- Ao criar recursos por meio das APIs de atualização ou importação em massa, não use PHI ou PII, incluindo os nomes dos armazenamentos de dados e trabalhos, em nenhum campo visível ou na ID lógica do FHIR (LID).
- Ao enviar solicitações de criação, leitura, atualização, exclusão ou pesquisa, não use PHI no cabeçalho HTTP.
- Permita AWS CloudTrail auditar o AWS HealthLake uso e garantir que não haja nenhuma atividade inesperada.
- Analise as melhores práticas para usar buckets do Amazon S3 com segurança. Para saber mais, consulte as melhores práticas de segurança no guia do usuário do Amazon S3.

Resiliência em AWS HealthLake

A infraestrutura AWS global é construída em torno de AWS regiões e zonas de disponibilidade. AWS As regiões fornecem várias zonas de disponibilidade fisicamente separadas e isoladas,

conectadas a redes de baixa latência, alta taxa de transferência e alta redundância. Com as zonas de disponibilidade, é possível projetar e operar aplicações e bancos de dados que automaticamente executam o failover entre as zonas sem interrupção. As zonas de disponibilidade são altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de data center tradicionais.

Se você precisa replicar seus dados ou aplicações para distâncias geográficas maiores, use as regiões locais da AWS . Uma região AWS local é um único data center projetado para complementar uma AWS região existente. Como todas as AWS regiões, as regiões AWS locais estão completamente isoladas das outras AWS regiões.

Para obter mais informações sobre AWS regiões e zonas de disponibilidade, consulte Infraestrutura AWS global.

AWS HealthLake referência

O seguinte material de referência de apoio está disponível para SMART em FHIR, FHIR e. AWS HealthLake

> **Note**
>
> Todas as HealthLake ações e tipos de dados nativos são descritos em uma referência separada. Para obter mais informações, consulte a AWS HealthLake Referência da API do.

Tópicos

- SMART no suporte FHIR para AWS HealthLake
- Suporte FHIR R4 para AWS HealthLake
- Referência de conformidade para AWS HealthLake
- Referência de suporte para AWS HealthLake

SMART no suporte FHIR para AWS HealthLake

Um armazenamento de HealthLake dados habilitado para aplicações médicas substituíveis e tecnologias reutilizáveis (SMART) permite o acesso ao SMART em aplicativos compatíveis com FHIR. HealthLake os dados são acessados por meio da autenticação e autorização de solicitações usando um servidor de autorização terceirizado. Portanto, em vez de gerenciar as credenciais do usuário via AWS Identity and Access Management, você está fazendo isso usando um SMART no servidor de autorização compatível com FHIR.

> **Note**
>
> HealthLake suporta SMART nas versões 1.0 e 2.0 do FHIR. Para saber mais sobre essas estruturas, consulte SMART App Launch na documentação do FHIR R4.
> HealthLake os armazenamentos de dados oferecem suporte às seguintes estruturas de autenticação e autorização para SMART em solicitações FHIR:
>
> - OpenID (AuthN): para autenticar a pessoa ou o aplicativo do cliente é quem (ou o que) ela afirma ser.

- OAuth 2.0 (AuthZ): para autorizar em quais recursos FHIR em seu armazenamento de HealthLake dados uma solicitação autenticada pode ler ou gravar. Isso é definido pelos escopos configurados em seu servidor de autorização.

Você pode criar um SMART no armazenamento de dados habilitado para FHIR usando o AWS CLI ou. AWS SDKs Para obter mais informações, consulte Criando um armazenamento HealthLake de dados.

Tópicos

- Começando a usar o SMART no FHIR
- HealthLake requisitos de autenticação para SMART no FHIR
- SMART em escopos FHIR OAuth 2.0 suportados por HealthLake
- Validação de token usando AWS Lambda
- Usando autorização refinada com um SMART no armazenamento de dados habilitado para FHIR HealthLake
- Buscando o SMART no documento FHIR Discovery
- Fazendo uma solicitação da API FHIR REST em um armazenamento de dados habilitado para Smart HealthLake

Começando a usar o SMART no FHIR

Os tópicos a seguir descrevem como começar a usar o SMART na autorização FHIR para. AWS HealthLake Eles incluem os recursos que você deve provisionar em sua AWS conta, a criação de um armazenamento de HealthLake dados habilitado para SMART on FHIR e um exemplo de como um aplicativo cliente SMART on FHIR interage com um servidor de autorização e um armazenamento de dados. HealthLake

Tópicos

- Configurando recursos para SMART no FHIR
- Fluxo de trabalho do aplicativo cliente para SMART no FHIR

Configurando recursos para SMART no FHIR

As etapas a seguir definem como as solicitações SMART on FHIR são tratadas HealthLake e os recursos necessários para que elas tenham sucesso. Os elementos a seguir funcionam juntos em um fluxo de trabalho para criar uma solicitação SMART na FHIR:

- O usuário final: geralmente, um paciente ou médico que usa um aplicativo SMART on FHIR de terceiros para acessar dados em um armazenamento de dados. HealthLake
- O aplicativo SMART on FHIR (conhecido como aplicativo cliente): um aplicativo que deseja acessar dados encontrados no armazenamento de HealthLake dados.
- O servidor de autorização: um servidor compatível com o OpenID Connect que é capaz de autenticar usuários e emitir tokens de acesso.
- O armazenamento de HealthLake dados: um armazenamento de HealthLake dados habilitado para SMART on FHIR que usa uma função Lambda para responder às solicitações REST do FHIR que fornecem um token portador.

Para que esses elementos funcionem juntos, você deve criar os seguintes recursos.

> **Note**
>
> Recomendamos criar seu SMART no armazenamento de HealthLake dados habilitado para FHIR depois de configurar o servidor de autorização, definir os escopos necessários nele e criar uma AWS Lambda função para lidar com a introspecção de tokens.

1. Configurar um endpoint do servidor de autorização

Para usar o SMART na estrutura FHIR, você precisa configurar um servidor de autorização terceirizado que possa validar as solicitações REST do FHIR feitas em um armazenamento de dados. Para obter mais informações, consulte HealthLake requisitos de autenticação para SMART no FHIR.

2. Defina escopos em seu servidor de autorização para controlar os níveis de acesso ao armazenamento de HealthLake dados

A estrutura SMART on FHIR usa OAuth escopos para determinar a quais recursos FHIR uma solicitação autenticada tem acesso e em que medida. Definir escopos é uma forma de projetar com

o mínimo de privilégios. Para obter mais informações, consulte SMART em escopos FHIR OAuth 2.0 suportados por HealthLake.

3. Configure uma AWS Lambda função capaz de realizar a introspecção de tokens

Uma solicitação FHIR REST enviada pelo aplicativo cliente em um SMART no armazenamento de dados habilitado para FHIR contém um JSON Web Token (JWT). Para obter mais informações, consulte Decodificando um JWT.

4. Crie um SMART no armazenamento de dados habilitado HealthLake para FHIR

Para criar um SMART no armazenamento de HealthLake dados FHIR, você precisa fornecer um. `IdentityProviderConfiguration` Para obter mais informações, consulte Criando um armazenamento HealthLake de dados.

Fluxo de trabalho do aplicativo cliente para SMART no FHIR

A seção a seguir explica como iniciar um aplicativo cliente e fazer uma solicitação FHIR REST bem-sucedida em um armazenamento de HealthLake dados dentro do contexto do SMART no FHIR.

1. Faça uma **`GET`** solicitação ao Known Uniform Resource Identifier usando o aplicativo cliente

Um aplicativo cliente habilitado para SMART deve fazer uma `GET` solicitação para encontrar os endpoints de autorização do seu armazenamento de HealthLake dados. Isso é feito por meio de uma solicitação de Identificador Uniforme de Recursos (URI) conhecido. Para obter mais informações, consulte Buscando o SMART no documento FHIR Discovery.

2. Solicitar acesso e escopos

O aplicativo cliente usa o ponto final de autorização do servidor de autorização, para que o usuário possa fazer login. Esse processo autentica o usuário. Os escopos são usados para definir quais recursos FHIR em seu armazenamento de HealthLake dados um aplicativo cliente pode acessar. Para obter mais informações, consulte SMART em escopos FHIR OAuth 2.0 suportados por HealthLake.

3. Tokens de acesso

Agora que o usuário foi autenticado, um aplicativo cliente recebe um token de acesso JWT do servidor de autorização. Esse token é fornecido quando o aplicativo cliente envia uma solicitação FHIR REST para o. HealthLake Para obter mais informações, consulte Validação de token.

4. Faça uma solicitação da API REST FHIR no SMART no armazenamento de dados habilitado para FHIR HealthLake

Agora, o aplicativo cliente pode enviar uma solicitação da API FHIR REST para um endpoint de armazenamento de HealthLake dados usando o token de acesso fornecido pelo servidor de autorização. Para obter mais informações, consulte Fazendo uma solicitação da API FHIR REST em um armazenamento de dados habilitado para Smart HealthLake.

5. Valide o token de acesso JWT

Para validar o token de acesso enviado na solicitação FHIR REST, use uma função Lambda. Para obter mais informações, consulte Validação de token usando AWS Lambda.

HealthLake requisitos de autenticação para SMART no FHIR

Para acessar recursos do FHIR em um SMART no armazenamento de HealthLake dados habilitado para FHIR, um aplicativo cliente deve ser autorizado por um servidor de autorização OAuth compatível com 2.0 e apresentar um token de OAuth portador como parte de uma solicitação da API FHIR REST. Para encontrar o endpoint do servidor de autorização, use o documento de descoberta HealthLake SMART on FHIR por meio de um identificador `Well-Known` uniforme de recursos. Para saber mais sobre esse processo, consulte Buscando o SMART no documento FHIR Discovery.

Ao criar um SMART no armazenamento de HealthLake dados FHIR, você deve definir o ponto final do servidor de autorização e o ponto final do token no `metadata` elemento da solicitação. `CreateFHIRDatastore` Para saber mais sobre como definir o `metadata` elemento, consulteCriando um armazenamento HealthLake de dados.

Usando os endpoints do servidor de autorização, o aplicativo cliente autenticará um usuário com o serviço de autorização. Depois de autorizado e autenticado, um JSON Web Token (JWT) é gerado pelo serviço de autorização e passado para o aplicativo cliente. Esse token contém escopos de recursos FHIR que o aplicativo cliente pode usar, o que, por sua vez, restringe quais dados o usuário pode acessar. Opcionalmente, se o escopo de lançamento foi fornecido, a resposta conterá esses detalhes. Para saber mais sobre o SMART em escopos FHIR suportados por HealthLake, consulte. SMART em escopos FHIR OAuth 2.0 suportados por HealthLake

Usando o JWT concedido pelo servidor de autorização, um aplicativo cliente faz chamadas da API FHIR REST para um SMART no armazenamento de dados habilitado para FHIR. HealthLake Para validar e decodificar o JWT, você precisa criar uma função Lambda. HealthLake invoca essa função Lambda em seu nome quando uma solicitação da API FHIR REST é recebida. Para ver um exemplo de função Lambda inicial, consulte. Validação de token usando AWS Lambda

Elementos do servidor de autorização necessários para criar um SMART no armazenamento de dados habilitado HealthLake para FHIR

Na `CreateFHIRDatastore` solicitação, você precisa fornecer o endpoint de autorização e o endpoint do token como parte do `metadata` elemento no `IdentityProviderConfiguration` objeto. Tanto o endpoint de autorização quanto o endpoint do token são obrigatórios. Para ver um exemplo de como isso é especificado na `CreateFHIRDatastore` solicitação, consulteCriando um armazenamento HealthLake de dados.

Declarações necessárias para concluir uma solicitação da API REST FHIR em um SMART no armazenamento de dados habilitado para FHIR HealthLake

Sua AWS Lambda função deve conter as seguintes declarações para que seja uma solicitação válida da API FHIR REST em um armazenamento de dados SMART no FHIR ativado. HealthLake

- `nbf`: Reivindicação (não anterior) — A reclamação “nbf” (não antes) identifica o momento antes do qual o JWT NÃO DEVE ser aceito para processamento. O processamento da reclamação “nbf” exige que a atual date/time DEVE ser posterior ou igual à não date/time listada antes na reivindicação “nbf”. O exemplo da função Lambda que fornecemos é convertido `iat` da resposta do servidor em. `nbf`
- `exp`: (Tempo de expiração) — A reivindicação “exp” (prazo de expiração) identifica o prazo de expiração a partir do qual o JWT não deve ser aceito para processamento.
- `isAuthorized`: Um booleano definido como. `True` Indica que a solicitação foi autorizada no servidor de autorização.
- `aud`: Reivindicação (Audiência) — A afirmação “aud” (audiência) identifica os destinatários aos quais o JWT se destina. Deve ser um terminal de armazenamento de HealthLake dados SMART no FHIR habilitado.
- `scope`: deve ser pelo menos um escopo relacionado a um recurso do FHIR. Esse escopo é definido em seu servidor de autorização. Para saber mais sobre os escopos relacionados a recursos do FHIR aceitos por HealthLake, consulte. SMART nos escopos de recursos do FHIR para HealthLake

SMART em escopos FHIR OAuth 2.0 suportados por HealthLake

HealthLake usa OAuth 2.0 como protocolo de autorização. O uso desse protocolo em seu servidor de autorização permite definir permissões de armazenamento de HealthLake dados (criar, ler, atualizar, excluir e pesquisar) para recursos FHIR aos quais um aplicativo cliente tem acesso.

A estrutura SMART on FHIR define um conjunto de escopos que podem ser solicitados do servidor de autorização. Por exemplo, um aplicativo cliente projetado apenas para permitir que os pacientes visualizem seus resultados laboratoriais ou visualizem seus detalhes de contato só deve ser autorizado a solicitar `read` escopos.

> **Note**
>
> HealthLake fornece suporte para SMART no FHIR V1 e V2, conforme descrito abaixo. O SMART no FHIR `AuthorizationStrategy`é definido para um dos três valores a seguir quando seu armazenamento de dados é criado:
>
> - `SMART_ON_FHIR_V1`— Support somente para SMART no FHIR V1, que inclui permissões `read` (leitura/pesquisa) e (). `write` create/update/delete
> - `SMART_ON_FHIR`— Support para SMART no FHIR V1 e V2, que inclui,`create`, `readupdate`, `delete` e permissões. `search`
> - `AWS_AUTH`— A estratégia de AWS HealthLake autorização padrão; não afiliada à SMART no FHIR.

Escopo de lançamento independente

HealthLake suporta o escopo `launch/patient` do modo de inicialização autônomo.

No modo de inicialização autônomo, um aplicativo cliente solicita acesso aos dados clínicos do paciente porque o usuário e o paciente não são conhecidos pelo aplicativo cliente. Assim, a solicitação de autorização do aplicativo do cliente solicita explicitamente que o escopo do paciente seja devolvido. Após a autenticação bem-sucedida, o servidor de autorização emite um token de acesso contendo o escopo do paciente de lançamento solicitado. O contexto necessário do paciente é fornecido junto com o token de acesso na resposta do servidor de autorização.

Escopos do modo de lançamento suportados

Escopo	Description
`launch/patient`	Um parâmetro em uma solicitação de autorização OAuth 2.0 solicitando que os dados do paciente sejam retornados na resposta de autorização.

SMART nos escopos de recursos do FHIR para HealthLake

HealthLake define três níveis de SMART nos escopos de recursos do FHIR.

- `patient`os escopos concedem acesso a dados específicos sobre um único paciente.
- `user`os escopos concedem acesso a dados específicos que um usuário pode acessar.
- `system`os escopos concedem acesso a todos os recursos do FHIR encontrados no armazenamento de dados. HealthLake

As seções a seguir listam a sintaxe para construir escopos de recursos FHIR usando SMART no FHIR V1 ou SMART no FHIR V2.

> **Note**
>
> A estratégia de autorização SMART on FHIR é definida quando seu armazenamento de dados é criado. Para obter mais informações, consulte `AuthorizationStrategy` na Referência de APIs do AWS HealthLake .

SMART em escopos FHIR V1 suportados por HealthLake

Ao usar o SMART no FHIR V1, a sintaxe geral para construir escopos de recursos do FHIR é a seguinte. HealthLake Para ver todo o caminho do URL no exemplo a seguir, role até o botão Copiar.

```
('patient' | 'user' | 'system') '/' (fhir-resource | '*') '.' ('read' | 'write' | '*')
```

SMART em escopos de autorização compatíveis com FHIR v1

Sintaxe do escopo	Exemplo de escopo	Resultado
`patient/(fhir-resource | '*').('read' | 'write' | '*')`	`patient/AllergyIntolerance.*`	O aplicativo cliente do paciente tem acesso de leitura/gravação em nível de instância a todas as alergias registradas.
`user/(fhir-resource | '*').('read' | 'write' | '*')`	`user/Observation.read`	O aplicativo cliente do usuário tem read/write e acesso em nível de instância a todas as observações registradas.
`system/('read' | 'write' | *)`	`system/*.*`	O aplicativo cliente do sistema tem read/write e acesso a todos os dados de recursos do FHIR.

SMART em escopos FHIR V2 suportados por HealthLake

Ao usar o SMART no FHIR V2, a sintaxe geral para construir escopos de recursos do FHIR é a seguinte. HealthLake Para ver todo o caminho do URL no exemplo a seguir, role até o botão Copiar.

```
('patient' | 'user' | 'system') '/' (fhir-resource | '*') '.' ('c' | 'r' | 'u' | 'd' | 's')
```

> **Note**
>
> Para usar o SMART no FHIR V2, você deve passar o valor `permission-v2`para a `capabilities` string de metadados, que é um membro do tipo de dados. `IdentityProviderConfiguration`

HealthLake suporta escopos granulares. Para obter mais informações, consulte os escopos granulares compatíveis no FHIR US Core Implementation Guide.

SMART em escopos de autorização compatíveis com FHIR V2

Sintaxe do escopo	Exemplo de escopo V1	Resultado
`patient/Observation.rs`	`user/Observation.read`	Permissão para ler e pesquisar `Observation` recursos para o paciente atual.
`system/*.cruds`	`system/*.*`	O aplicativo cliente do sistema tem create/read/update/delete/search acesso total a todos os dados de recursos do FHIR.

Validação de token usando AWS Lambda

Ao criar um HealthLake SMART no armazenamento de dados habilitado para FHIR, você deve fornecer o ARN da AWS Lambda função na solicitação. `CreateFHIRDatastore` O ARN da função Lambda é especificado no `IdentityProviderConfiguration` objeto usando o parâmetro. `IdpLambdaArn`

Você deve criar a função Lambda antes de criar seu SMART no armazenamento de dados habilitado para FHIR. Depois de criar o armazenamento de dados, o ARN do Lambda não pode ser alterado. Para ver o ARN do Lambda que você especificou quando o armazenamento de dados foi criado, use a ação da API. `DescribeFHIRDatastore`

Para que uma solicitação FHIR REST seja bem-sucedida em um armazenamento de dados SMART no FHIR ativado, sua função Lambda deve fazer o seguinte:

- Retorne uma resposta em menos de 1 segundo para o endpoint do armazenamento de HealthLake dados.
- Decodifique o token de acesso fornecido no cabeçalho de autorização da solicitação da API REST enviada pelo aplicativo cliente.
- Atribua uma função de serviço do IAM que tenha permissões suficientes para realizar a solicitação da API FHIR REST.
- As declarações a seguir são necessárias para concluir uma solicitação da API FHIR REST. Para saber mais, consulte Declarações necessárias.
  - `nbf`
  - `exp`
  - `isAuthorized`
  - `aud`
  - `scope`

Ao trabalhar com o Lambda, você precisa criar uma função de execução e uma política baseada em recursos, além da sua função Lambda. A função de execução de uma função do Lambda é uma função do IAM que concede à função permissão para acessar os serviços e recursos da AWS necessários em tempo de execução. A política baseada em recursos que você fornece deve permitir HealthLake invocar sua função em seu nome.

As seções deste tópico descrevem um exemplo de solicitação de um aplicativo cliente e uma resposta decodificada, as etapas necessárias para criar uma função AWS Lambda e como criar uma política baseada em recursos que possa assumir. HealthLake

- Parte 1: Criando uma função Lambda
- Parte 2: Criando uma função HealthLake de serviço usada pela função AWS Lambda
- Parte 3: Atualizando a função de execução da função Lambda
- Parte 4: Adicionando uma política de recursos à sua função Lambda
- Parte 5: Provisionando a simultaneidade para sua função Lambda

Criação de uma função AWS Lambda

A função Lambda criada neste tópico é acionada quando HealthLake recebe uma solicitação para um SMART no armazenamento de dados habilitado para FHIR. A solicitação do aplicativo cliente contém uma chamada à API REST e um cabeçalho de autorização contendo um token de acesso.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
Authorization: Bearer i8hweunweunweofiwweoijewiwe
```

O exemplo da função Lambda neste tópico é usado AWS Secrets Manager para obscurecer as credenciais relacionadas ao servidor de autorização. É altamente recomendável não fornecer detalhes de login do servidor de autorização diretamente em uma função Lambda.

Example validando uma solicitação FHIR REST contendo um token do portador da autorização

O exemplo da função Lambda mostra como validar uma solicitação FHIR REST enviada para um SMART no armazenamento de dados habilitado para FHIR. Para ver step-by-steps instruções sobre como implementar essa função Lambda, consulte. Criando uma função Lambda usando o Console de gerenciamento da AWS

Se a solicitação da API FHIR REST não contiver um endpoint de armazenamento de dados, token de acesso e operação REST válidos, a função Lambda falhará. Para saber mais sobre os elementos necessários do servidor de autorização, consulteDeclarações necessárias.

```
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()
```

Criando uma função Lambda usando o Console de gerenciamento da AWS

O procedimento a seguir pressupõe que você já tenha criado a função de serviço que deseja assumir HealthLake ao lidar com uma solicitação da API REST FHIR em um armazenamento de dados SMART no FHIR habilitado. Se você não criou a função de serviço, ainda pode criar a função Lambda. Você deve adicionar o ARN da função de serviço antes que a função Lambda funcione. Para saber mais sobre como criar uma função de serviço e especificá-la na função Lambda, consulte Criação de uma função HealthLake de serviço para uso na função AWS Lambda usada para decodificar um JWT

Para criar uma função Lambda ()Console de gerenciamento da AWS

1. Abra a página Funções do console do Lambda.
2. Escolha a opção Criar função.
3. Selecione Criar do zero.
4. Em Informações básicas, insira um nome de função. Em Tempo de execução, escolha um tempo de execução baseado em python.
5. Em Execution role (Perfil de execução), escolha Create a new role with basic Lambda permissions (Criar um novo perfil com as permissões básicas do Lambda).

   O Lambda cria uma função de execução que concede à função permissão para fazer upload de registros para a Amazon. CloudWatch A função Lambda assume a função de execução quando você invoca sua função e usa a função de execução para criar credenciais para o SDK. AWS
6. Escolha a guia Código e adicione o exemplo da função Lambda.

   Se você ainda não criou a função de serviço para usar a função Lambda, precisará criá-la antes que a função de amostra do Lambda funcione. Para saber mais sobre a criação de uma função de serviço para a função Lambda, consulte. Criação de uma função HealthLake de serviço para uso na função AWS Lambda usada para decodificar um 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()
```

Modificando a função de execução de uma função Lambda

Depois de criar a função Lambda, você precisa atualizar a função de execução para incluir as permissões necessárias para chamar o Secrets Manager. No Secrets Manager, cada segredo que você cria tem um ARN. Para aplicar o menor privilégio, a função de execução só deve ter acesso aos recursos necessários para a execução da função Lambda.

Você pode modificar a função de execução de uma função Lambda pesquisando-a no console do IAM ou escolhendo Configuração no console Lambda. Para saber mais sobre como gerenciar sua função de execução de funções Lambda, consulte. Função de execução do Lambda

Example Função de execução da função Lambda que concede acesso a **`GetSecretValue`**

Adicionar a ação do IAM `GetSecretValue` à função de execução concede a permissão necessária para que a função de amostra do Lambda funcione.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:secret-name-DKodTA"
        }
    ]
}
```

Neste ponto, você criou uma função Lambda que pode ser usada para validar o token de acesso fornecido como parte da solicitação FHIR REST enviada ao seu SMART no armazenamento de dados habilitado para FHIR.

Criação de uma função HealthLake de serviço para uso na função AWS Lambda usada para decodificar um JWT

Persona: administrador do IAM

Um usuário que pode adicionar ou remover políticas do IAM e criar novas identidades do IAM.

Perfil de serviço

O perfil de serviço é um perfil do IAM que um serviço assume para executar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de serviço do IAM. Para saber mais, consulte Criar um perfil para delegar permissões a um AWS service (Serviço da AWS) no Guia do Usuário do IAM.

Depois que o JSON Web Token (JWT) é decodificado, a autorização Lambda também precisa retornar um ARN de função do IAM. Essa função deve ter as permissões necessárias para realizar a solicitação da API REST ou falhará devido a permissões insuficientes.

Ao configurar uma política personalizada usando o IAM, é melhor conceder as permissões mínimas necessárias. Para saber mais, consulte Aplicar permissões de privilégios mínimos no Guia do usuário do IAM.

A criação HealthLake de uma função de serviço para designar na função Lambda de autorização requer duas etapas.

- Primeiro, você precisa criar uma política do IAM. A política deve especificar o acesso aos recursos do FHIR para os quais você forneceu escopos no servidor de autorização.
- Em segundo lugar, você precisa criar a função de serviço. Ao criar a função, você designa uma relação de confiança e anexa a política criada na etapa um. A relação de confiança é designada HealthLake como principal do serviço. Você precisa especificar um ARN do armazenamento de HealthLake dados e uma ID de AWS conta nesta etapa.

Criação de uma nova política do IAM

Os escopos que você define em seu servidor de autorização determinam a quais recursos FHIR um usuário autenticado tem acesso em um armazenamento de dados. HealthLake

A política do IAM que você cria pode ser personalizada para corresponder aos escopos que você definiu.

As ações a seguir no `Action` elemento de uma declaração de política do IAM podem ser definidas. Para cada um `Action` na tabela, você pode definir um`Resource types`. Em HealthLake um armazenamento de dados, é o único tipo de recurso compatível que pode ser definido no `Resource` elemento de uma declaração de política de permissão do IAM.

Recursos individuais do FHIR não são um recurso que você possa definir como um elemento em uma política de permissão do IAM.

Ações definidas por HealthLake

Ações	Descrição	Nível de acess	Tipo de recurso (obrigatório)
CreateResource	Concede permissão a um recurso de criação	Grava	ARN do armazenamento de dados: arn:aws:healthlake: ::datastore/fhir/ **your-region 111122223333 your-datastore-id**
DeleteResource	Concede permissão para excluir um recurso	Grava	ARN do armazenamento de dados: arn:aws:healthlake: ::datastore/fhir/ **your-region 111122223333 your-datastore-id**
ReadResource	Concede permissão para ler um recurso	Ler	ARN do armazenamento de dados: arn:aws:healthlake: ::datastore/fhir/ **your-region 111122223333 your-datastore-id**
SearchWithGet	Concede permissão para pesquisar recursos com o método GET	Ler	ARN do armazenamento de dados: arn:aws:healthlake: ::datastore/fhir/ **your-region 111122223333 your-datastore-id**

Ações	Descrição	Nível de acess	Tipo de recurso (obrigatório)
SearchWithPost	Concede permissão para pesquisar recursos com o método POST	Ler	ARN do armazenamento de dados: arn:aws:healthlake: ::datastore/fhir/ **your-region 111122223333 your-datastore-id**
IniciarFHIRExportJobWithPost	Concede permissão para iniciar um trabalho de exportação FHIR com GET	Grava	ARN do armazenamento de dados: arn:aws:healthlake: ::datastore/fhir/ **your-region 111122223333 your-datastore-id**
UpdateResource	Concede permissão para atualizar um recurso	Grava	ARN do armazenamento de dados: arn:aws:healthlake: ::datastore/fhir/ **your-region 111122223333 your-datastore-id**

Para começar, você pode usar`AmazonHealthLakeFullAccess`. Essa política permitiria leitura, gravação, pesquisa e exportação em todos os recursos do FHIR encontrados em um armazenamento de dados. Para conceder permissões somente de leitura em um armazenamento de dados, use. `AmazonHealthLakeReadOnlyAccess`

Para saber mais sobre como criar uma política personalizada usando o Console de gerenciamento da AWS, AWS CLI, ou IAM SDKs, consulte Como criar políticas do IAM no Guia do usuário do IAM.

Criação de uma função de serviço para HealthLake (console do IAM)

Use esse procedimento para criar uma função de serviço. Ao criar um serviço, você também precisará designar uma política do IAM.

Para criar a função de serviço para HealthLake (console do IAM)

1. Faça login no Console de gerenciamento da AWS e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação do console do IAM, escolha Perfis.
3. Depois, escolha Create role (Criar perfil).
4. Na página Selecionar entidade confiável, escolha Política de confiança personalizada.

5. Em seguida, em Política de confiança personalizada, atualize a política de amostra da seguinte forma. **`your-account-id`**Substitua pelo número da sua conta e adicione o ARN do armazenamento de dados que você deseja usar em seus trabalhos de importação ou exportação.

   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. Em seguida, escolha Próximo.
7. Na página Adicionar permissões, escolha a política que você deseja que o HealthLake serviço assuma. Para encontrar sua política, procure-a em Políticas de permissões.
8. Em seguida, escolha Anexar política.
9. Em seguida, na página Nome, revise e crie, em Nome da função, insira um nome.
10. (Opcional) Em seguida, em Descrição, adicione uma breve descrição para sua função.
11. Se possível, insira um nome de função ou sufixo de nome de função para ajudar a identificar o propósito desta função. Os nomes de função devem ser exclusivos em sua Conta da AWS. Eles não são diferenciados por maiúsculas e minúsculas. Por exemplo, não é possível criar perfis

denominados **`PRODROLE`** e **`prodrole`**. Como várias entidades podem fazer referência à função, não é possível editar o nome da função depois que ela é criada.

12. Revise os detalhes da função e escolha Criar função.

Para saber como especificar o ARN da função no exemplo da função Lambda, consulte. Criação de uma função AWS Lambda

Função de execução do Lambda

A função de execução de uma função Lambda é uma função do IAM que concede à função permissão para acessar AWS serviços e recursos. Esta página fornece informações sobre como criar, visualizar e gerenciar o perfil de execução de uma função do Lambda.

Por padrão, o Lambda cria uma função de execução com permissões mínimas quando você cria uma nova função do Lambda usando o. Console de gerenciamento da AWS Para gerenciar as permissões concedidas na função de execução, consulte Criação de uma função de execução no console do IAM no Lambda Developer Guide.

O exemplo da função Lambda fornecido neste tópico usa o Secrets Manager para ocultar as credenciais do servidor de autorização.

Como acontece com qualquer função do IAM que você criar, é importante seguir as melhores práticas de privilégios mínimos. Durante a frase de desenvolvimento, às vezes você pode conceder permissões além do necessário. Antes de publicar sua função no ambiente de produção, como prática recomendada, ajuste a política para incluir somente as permissões necessárias. Para obter mais informações, consulte Aplicar o menor privilégio no Guia do usuário do IAM.

Permitir HealthLake acionar sua função Lambda

Para que HealthLake possa invocar a função Lambda em seu nome, você deve fazer o seguinte:

- Você precisa definir `IdpLambdaArn` igual ao ARN da função Lambda que você deseja HealthLake invocar na solicitação. `CreateFHIRDatastore`
- Você precisa de uma política baseada em recursos que permita HealthLake invocar a função Lambda em seu nome.

Quando HealthLake recebe uma solicitação da API REST FHIR em um armazenamento de dados SMART no FHIR ativado, ele precisa de permissões para invocar a função Lambda especificada na criação do armazenamento de dados em seu nome. Para conceder HealthLake acesso, você

usará uma política baseada em recursos. Para saber mais sobre como criar uma política baseada em recursos para uma função do Lambda, consulte Permitir que um AWS serviço chame uma função do Lambda no Guia do desenvolvedor.AWS Lambda

Provisionando a simultaneidade para sua função Lambda

> **Important**
>
> HealthLake exige que o tempo máximo de execução da função Lambda seja inferior a um segundo (1000 milissegundos).
> Se sua função Lambda exceder o limite de tempo de execução, você receberá uma TimeOutexceção.

Para evitar essa exceção, recomendamos configurar a simultaneidade provisionada. Ao alocar simultaneidade provisionada antes de um aumento nas invocações, é possível garantir que todas as solicitações sejam atendidas por instâncias inicializadas com latência baixa. Para saber mais sobre como configurar a simultaneidade provisionada, consulte Como configurar a simultaneidade provisionada no Lambda Developer Guide

Para ver o tempo médio de execução de sua função Lambda atualmente, use a página de monitoramento de sua função Lambda no console Lambda. Por padrão, o console Lambda fornece um gráfico de duração que mostra a quantidade média, mínima e máxima de tempo que seu código de função gasta processando um evento. Para saber mais sobre o monitoramento das funções do Lambda, consulte Funções de monitoramento no console do Lambda no Guia do desenvolvedor do Lambda.

Se você já provisionou a simultaneidade para sua função do Lambda e deseja monitorá-la, consulte Monitoramento da simultaneidade no Guia do desenvolvedor do Lambda.

Usando autorização refinada com um SMART no armazenamento de dados habilitado para FHIR HealthLake

Os escopos por si só não fornecem a especificidade necessária sobre quais dados um solicitante está autorizado a acessar em um armazenamento de dados. O uso de autorização refinada permite um nível mais alto de especificidade ao conceder acesso a um SMART no armazenamento de dados habilitado para FHIR. HealthLake Para usar uma autorização refinada, defina `FineGrainedAuthorizationEnabled` igual a `True` no `IdentityProviderConfiguration` parâmetro da sua solicitação. `CreateFHIRDatastore`

Se você habilitou a autorização refinada, seu servidor de autorização retornará um `fhirUser` escopo `id_token` junto com o token de acesso. Isso permite que as informações sobre o usuário sejam recuperadas pelo aplicativo do cliente. O aplicativo cliente deve tratar a `fhirUser` declaração como o URI de um recurso FHIR representando o usuário atual. Pode ser `Patient`, `Practitioner` ou `RelatedPerson`. A resposta do servidor de autorização também inclui um `user/` escopo que define quais dados o usuário pode acessar. Isso usa a sintaxe definida para escopos relacionados aos escopos específicos de recursos do FHIR:

```
user/(fhir-resource | '*').('read' | 'write' | '*')
```

A seguir estão exemplos de como a autorização refinada pode ser usada para especificar ainda mais os tipos de recursos FHIR relacionados ao acesso a dados.

- Quando `fhirUser` é uma `Practitioner` autorização refinada determina a coleção de pacientes que o usuário pode acessar. `fhirUser`O acesso é permitido apenas para os pacientes em que o paciente se refere ao `fhirUser` como clínico geral.

  ```
  Patient.generalPractitioner : [{Reference(Practitioner)}]
  ```

- Quando `fhirUser` é um `Patient` ou `RelatedPerson` e o paciente referenciado na solicitação é diferente do`fhirUser`, uma autorização refinada determina o acesso `fhirUser` do paciente solicitado. O acesso é permitido quando há um relacionamento especificado no `Patient` recurso solicitado.

  ```
  Patient.link.other : {Reference(Patient|RelatedPerson)}
  ```

Buscando o SMART no documento FHIR Discovery

O SMART define um documento de descoberta que permite que os clientes conheçam o endpoint de autorização URLs e os recursos que um armazenamento HealthLake de dados suporta. Essas informações ajudam os clientes a direcionar as solicitações de autorização para o endpoint correto e a criar solicitações de autorização suportadas pelo armazenamento de HealthLake dados.

Para que um aplicativo cliente faça uma solicitação FHIR REST bem-sucedida HealthLake, ele deve reunir os requisitos de autorização definidos pelo armazenamento de HealthLake dados. Não é necessário um token de portador (autorização) para que essa solicitação seja bem-sucedida.

Para solicitar o Documento de Descoberta para um armazenamento HealthLake de dados

1. Colecione HealthLake `region` e `datastoreId` valorize. Para obter mais informações, consulte Obter propriedades do datastore.
2. Crie uma URL para a solicitação usando os valores coletados para HealthLake `region` `datastoreId` e. Anexe `/.well-known/smart-configuration` ao endpoint do URL. Para ver todo o caminho do URL no exemplo a seguir, role até o botão Copiar.

```
https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/.well-known/smart-configuration
```

3. Envie a solicitação usando o protocolo `GET` de AWS assinatura Signature Version 4. Para ver o exemplo inteiro, role até o botão Copiar.

   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'
```

   O documento de descoberta do armazenamento de HealthLake dados retorna como um blob JSON, onde você pode encontrar o `authorization_endpoint` e `otoken_endpoint`, junto com as especificações e os recursos definidos para o armazenamento de dados.

```
{
    "authorization_endpoint": "https://oidc.example.com/authorize",
    "token_endpoint": "https://oidc.example.com/oauth/token",
    "capabilities": [
        "launch-ehr",
        "client-public"
    ]
}
```

   Tanto o `authorization_endpoint` quanto o `token_endpoint` são necessários para iniciar um aplicativo cliente.

- Ponto final de autorização — O URL necessário para autorizar um aplicativo ou usuário cliente.
- Ponto final do token — O ponto final do servidor de autorização com o qual o aplicativo cliente usa para se comunicar.

Fazendo uma solicitação da API FHIR REST em um armazenamento de dados habilitado para Smart HealthLake

Você pode fazer solicitações da API REST FHIR em um SMART no armazenamento de dados habilitado para HealthLake FHIR. O exemplo a seguir mostra uma solicitação do aplicativo cliente contendo um JWT no cabeçalho de autorização e como o Lambda deve decodificar a resposta. Depois que a solicitação do aplicativo cliente for autorizada e autenticada, ela deverá receber um token portador do servidor de autorização. Use o token do portador no cabeçalho de autorização ao enviar uma solicitação da API REST FHIR em um SMART no armazenamento de dados habilitado para HealthLake FHIR.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/[ID]
Authorization: Bearer auth-server-provided-bearer-token
```

Como um token portador foi encontrado no cabeçalho de autorização e nenhuma identidade AWS do IAM foi detectada, HealthLake invoca a função Lambda especificada quando o SMART no armazenamento de dados habilitado para FHIR foi criado. HealthLake Quando o token é decodificado com sucesso pela sua função Lambda, o exemplo de resposta a seguir é enviado para. 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
}
```

Suporte FHIR R4 para AWS HealthLake

AWS HealthLake suporta a especificação FHIR R4 para troca de dados de saúde. As seções a seguir fornecem informações de suporte sobre como HealthLake utiliza a especificação FHIR R4 para ajudá-lo a gerenciar e pesquisar recursos FHIR em seu HealthLake armazenamento de dados usando o FHIR R4. RESTful APIs

Tópicos

- Declaração de capacidade do FHIR R4 para AWS HealthLake
- Validações de perfil FHIR para HealthLake
- Tipos de recursos compatíveis com FHIR R4 para HealthLake
- Parâmetros de pesquisa FHIR R4 para HealthLake
- FHIR R4 para $operations HealthLake

Declaração de capacidade do FHIR R4 para AWS HealthLake

Para encontrar os recursos (comportamentos) relacionados ao FHIR de um armazenamento de HealthLake dados ativo, você deve recuperar sua Declaração de Capacidade. A Declaração de Capacidade é usada como uma declaração da funcionalidade real do servidor ou uma declaração da implementação necessária ou desejada do servidor. A `capabilities`interação FHIR recupera informações sobre HealthLake os recursos do armazenamento de dados e quais partes da especificação FHIR ela suporta. HealthLake valida os tipos de recursos FHIR de acordo com o recurso FHIR R4. `StructureDefinition`

Para obter a Declaração de Capacidade para um armazenamento HealthLake de dados

1. Colecione HealthLake `region` e `datastoreId` valorize. Para obter mais informações, consulte Obter propriedades do datastore.
2. Crie uma URL para a solicitação usando os valores coletados para HealthLake `region` `datastoreId` e. Inclua também o `metadata` elemento FHIR no URL. Para ver todo o caminho do URL no exemplo a seguir, role até o botão Copiar.

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

3. Envie a solicitação . A `capabilities` interação FHIR usa uma `GET` solicitação com o protocolo de AWS assinatura Signature Version 4. O `curl` exemplo a seguir obtém a Declaração de Capacidade para o armazenamento de HealthLake dados especificado pelo`datastoreId`. Para ver o exemplo inteiro, role até o botão Copiar.

   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'
   ```

   Você receberá um código de resposta `200` HTTP e a Declaração de Capacidade para seu armazenamento HealthLake de dados. Para obter mais informações, consulte a `CapabilityStatement`documentação do FHIR R4.

Validações de perfil FHIR para HealthLake

AWS HealthLake suporta a especificação básica FHIR R4. Incluídos na especificação básica do FHIR R4 estão os perfis FHIR. Os perfis são usados em um tipo de recurso FHIR para definir uma definição de tipo de recurso mais específica usando and/or extensões de restrições no tipo de recurso base. Por exemplo, um perfil FHIR pode identificar campos obrigatórios, como extensões e conjuntos de valores. Um recurso pode oferecer suporte a vários perfis. Todos os armazenamentos HealthLake de dados suportam o uso de perfis FHIR.

> **Note**
>
> Não é necessário adicionar um perfil FHIR ao adicionar dados a um armazenamento de HealthLake dados. Se um perfil FHIR não for especificado quando um recurso for adicionado ou atualizado, o recurso será validado somente com base no esquema FHIR R4.

Os perfis FHIR, aos quais os recursos do FHIR estão em conformidade, são incluídos nos recursos antes de serem importados. HealthLake Portanto, os perfis FHIR são validados HealthLake durante a importação.

Os perfis FHIR são especificados em um guia de implementação. Um Guia de Implementação (IG) do FHIR é um conjunto de instruções que descrevem como usar o padrão FHIR para uma finalidade específica. HealthLake valida os perfis FHIR definidos nos guias de implementação a seguir.

Perfis FHIR suportados por AWS HealthLake

Nome	Ve	Guia de implement ação	Recurso	Les dos EU (Ol	Les dos EU (N. da Vir	Oe dos EU (Or	Ási Pa fico (Mı	Ási Pa fico (Sy	Ca (Ce	Eu (Irl	Europa (Londres)
Núcleo dos EUA	3.1	http://hl7.org/fhir/us/core/STU3.1.1/	Padrão	X	X	X	X	X	X	X	X
Núcleo dos EUA	4.0	https://hl7.org/fhir/us/core/STU4/index.html	Compatível	X	X	X	X	X	X	X	X
Núcleo dos EUA	5.0	https://hl7.org/fhir/us/core/STU5.0.1/index.html	Compatível	X	X	X	X	X	X	X	X
Núcleo dos EUA	6.1	https://hl7.org/fhir/us/core/STU6.1/index.html	Compatível	X	X	X	X	X	X	X	X
Núcleo dos EUA	7.0	https://hl7.org/fhir/us/core/STU7/	Compatível	X	X	X	X	X	X	X	X
Core do Reino Unido	2.0	https://simplifier.net/guide/uk-core-implementation-guide-stu2/Home/Profiles	Compatível	X	X	X	X			X	X

Nome	Ve	Guia de implement ação	Recurso	Les dos EU (Ol	Les dos EU (N. da Vir	Oe dos EU (Or	Ási Pa fico (Mi	Ási Pa fico (Sy	Ca (Ce	Eu (Irl	Europa (Londres)
		andExtensions/Profi lesIndex?version =2.0.1									
Botão azul CARIN	1.1	http://hl7.org/fhir/us/carin-bb/STU1.1/	Padrão	X	X	X	X	X	X	X	X
Botão azul CARIN	1.0 2.0 2.1	https://hl7.org/fhir/us/carin-bb/history.html	Compatível	X	X	X	X	X	X	X	X
Troca de Dados do Pagador Da Vinci	1.0	https://hl7.org/fhir/us/davinci-pdex/	Padrão	X	X	X	X	X	X	X	X
Troca de Dados do Pagador Da Vinci	2.0 2.1	https://hl7.org/fhir/us/davinci-pdex/history.html	Compatível	X	X	X	X	X	X	X	X
Bolsa de Registros de Saúde Da Vinci (HRex)	0.2	https://hl7.org/fhir/us/davinci-hrex/2020Sep/	Padrão	X	X	X	X	X	X	X	X
DaVinci Plano PDEX Net	1.1	https://hl7.org/fhir/us/davinci-pdex-plan-net/STU1.1/	Padrão	X	X	X	X	X	X	X	X

Nome	Ve	Guia de implement ação	Recurso	Les dos EU (Ol	Les dos EU (N. da Vir	Oe dos EU (Or	Ási Pa fico (Mı	Ási Pa fico (Sy	Ca (Ce	Eu (Irl	Europa (Londres)
DaVinci Plano PDEX Net	1.0	https://hl7.org/fhir/us/davinci-pdex-plan-net/STU1/	Compatível	X	X	X	X	X	X	X	X
DaVinci Formulário de medicamentos dos EUA do Payer Data Exchange (PDex)	1.1	https://hl7.org/fhir/us/davinci-drug-formulary/STU1.1/	Padrão	X	X	X	X	X	X	X	X
DaVinci Formulário de medicamentos dos EUA do Payer Data Exchange (PDex)	1.0 2.0 2.1	https://hl7.org/fhir/us/davinci-drug-formulary/history.html	Compatível	X	X	X	X	X	X	X	X
Troca de Dados Clínicos Da Vinci (CDex)	2.1	https://build.fhir.org/ig/HL7/davinci-ecdx/index.html	Padrão	X	X	X	X	X	X	X	X

Nome	Ve	Guia de implement ação	Recurso	Les dos EU (Ol	Les dos EU (N. da Vir	Oe dos EU (Or	Ási Pa fico (Mı	Ási Pa fico (Sy	Ca (Ce	Eu (Irl	Europa (Londres)
Da Vinci Previous Authorization Support (PAS) FHIR IG	2.1	https://hl7.org/fhir/us/davinci-pas/	Padrão	X	X	X	X	X	X	X	X
Guia de implement ação do NCQA HEDIS®	0.3	https://www.ncqa.org/resources/hedis-ig-resource-page/	Padrão	X	X	X	X			X	X
Resumo internaci onal do paciente (IPS)	2.0 - cé	https://hl7.org/fhir/uv/ips/2024Sep/	Padrão	X	X	X	X	X	X	X	X
Medida de qualidade	5.0	https://registry.fhir.org/package/hl7.fhir.us.cqfmeasures%7C5.0.0	Padrão	X	X	X	X			X	X
Relatórios genômicos	3.0	https://build.fhir.org/ig/HL7/genomics-reporting/index.html	Padrão	X	X	X	X			X	X

Nome	Ve	Guia de implement ação	Recurso	Les dos EU (Ol	Les dos EU (N. da Vir	Oe dos EU (Or	Ási Pa fico (Mi	Ási Pa fico (Sy	Ca (Ce	Eu (Irl	Europa (Londres)
Missão Digital Ayushman Bharat (ABDM) da Autoridade Nacional de Saúde	2,0	https://www.nrces.in/ndhm/fhir/r4/index.html	Padrão	X	X	X	X			X	X
CA Core +	1.1	https://simplifier.net/ca-core	Compatível						X		
CA:eReC Pan-Canadian eReferral-eConsult	1.1	https://simplifier.net/CA-eReC/~introduction	Compatível						X		
Resumo do paciente, edição canadense - (PS-CA)	2.1	https://simplifier.net/PS-CA-R1/~introduction	Compatível						X		
Repositório Digital de Medicamentos para Saúde de Ontário	4.0	https://simplifier.net/ca-on-dhdr-r4/~introduction	Compatível						X		

Nome	Ve	Guia de implement ação	Recurso	Les dos EU (Ol	Les dos EU (N. da Vir	Oe dos EU (Or	Ási Pa fico (Mu	Ási Pa fico (Sy	Ca (Ce	Eu (Irl	Europa (Londres)
Núcleo AU	1.0	https://hl7.org.au/fhir/core/	Compatível					X			
Gestão de práticas Magentus	1.2	https://fhir-versions.dev.geniesolutions.io/1.2.16/downloads.html	Compatível					X			

Validando perfis FHIR especificados em um recurso

Para que um perfil FHIR seja validado, adicione-o ao `profile` elemento de recursos individuais usando a URL do perfil designada no guia de implementação.

Os perfis FHIR são validados quando você adiciona um novo recurso ao seu armazenamento de dados. Para adicionar um novo recurso, você pode usar a operação de `StartFHIRImportJob` API, fazer uma `POST` solicitação para adicionar um novo recurso ou `PUT` atualizar um recurso existente.

Example— Para ver qual perfil FHIR é referenciado em um recurso

O URL do perfil é adicionado ao `profile` elemento no par de `"meta" : "profile"` valores-chave. Esse recurso foi truncado para maior clareza.

```
{
    "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— Como referenciar um perfil FHIR não suportado por padrão

Para validar em relação a um perfil não padrão compatível, adicione a URL do perfil versionado ao elemento. `meta.profile` O URL versionado inclui o URL do perfil base seguido por um caractere vertical (`|`) e o número da versão. Este recurso de exemplo foi truncado para maior clareza.

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

Você também pode incluir o URL versionado e o URL do perfil base. Ambos os formatos são válidos.

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

Tipos de recursos compatíveis com FHIR R4 para HealthLake

A tabela a seguir lista os tipos de recursos do FHIR R4 suportados pelo. AWS HealthLake Para obter mais informações, consulte Índice de recursos na documentação do FHIR R4.

Tipos de recursos FHIR R4 suportados pelo HealthLake

Conta	DetectedIssue	Fatura	Praticante

ActivityDefinition	Dispositivo	Biblioteca	PractitionerRole
AdverseEvent	DeviceDefinition	Ligação	Procedimento
AllergyIntolerance	DeviceMetric	Lista	Proveniência
Nomeação	DeviceUseStatement	Local	Questionário
AppointmentResponse	DeviceRequest	Medida	QuestionnaireResponse
AuditEvent - Veja a nota	DiagnosticReport	MeasureReport	RelatedPerson
Binário	DocumentManifest	Mídia	RequestGroup
BodyStructure	DocumentReference	Medicação	ResearchStudy
Pacote - Veja a nota	EffectEvidenceSynthesis	MedicationAdministration	ResearchSubject
CapabilityStatement	Encontro	MedicationDispense	RiskAssessment
CarePlan	Endpoint	MedicationKnowledge	RiskEvidenceSynthesis
CareTeam	EpisodeOfCare	MedicationRequest	Agendamento
ChargeItem	EnrollmentRequest	MedicationStatement	ServiceRequest
ChargeItemDefinition	EnrollmentResponse	MessageHeader	Slot
Reivindicar	ExplanationOfBenefit	MolecularSequence	Espécime
ClaimResponse	FamilyMemberHistory	NutritionOrder	StructureDefinition
Comunicação	Sinalizador	Observação	StructureMap
CommunicationRequest	Objetivo	OperationOutcome - Veja a nota	Substance

Composição	Group (Grupo)	Organização	SupplyDelivery
ConceptMap	GuidanceResponse	OrganizationAffiliation	SupplyRequest
Condição	HealthcareService	Parâmetros - Veja a nota	Tarefa
Consentimento	ImagingStudy	Paciente	ValueSet
Contrato	Imunização	PaymentNotice	VisionPrescription
Cobertura	ImmunizationEvaluation	PaymentReconciliation	VerificationResult - Veja a nota
CoverageEligibilityRequest	ImmunizationRecommendation	Pessoa	
CoverageEligibilityResponse	InsurancePlan	PlanDefinition	

⚠ Especificações FHIR e HealthLake

- Você não pode fazer `GET` `POST` solicitações com FHIR `OperationOutcome` e tipos de `Parameters` recursos.
- AuditEvent— Um AuditEvent recurso pode ser criado ou lido, mas não pode ser atualizado ou excluído.
- Bundle — Há várias maneiras de HealthLake gerenciar solicitações de Bundle. Consulte mais detalhes em Agrupando recursos do FHIR.
- VerificationResult— Esse tipo de recurso só é compatível com armazenamentos de dados criados após 09 de dezembro de 2023.

Parâmetros de pesquisa FHIR R4 para HealthLake

Use a `search`interação FHIR para pesquisar um conjunto de recursos FHIR em um armazenamento de HealthLake dados com base em alguns critérios de filtro. A `search` interação pode ser realizada usando uma `POST` solicitação `GET` ou. Para pesquisas que envolvam informações de identificação

pessoal (PII) ou informações de saúde protegidas (PHI), é recomendável usar POST solicitações, pois PII e PHI são adicionadas como parte do corpo da solicitação e criptografadas em trânsito.

> **Note**
>
> A `search` interação FHIR descrita neste capítulo é construída em conformidade com o padrão HL7 FHIR R4 para troca de dados de assistência médica. Por ser uma representação de um serviço HL7 FHIR, não é oferecido por meio AWS CLI de e. AWS SDKs Para obter mais informações, consulte a `search`documentação da API FHIR R4 RESTful .
> Você também pode consultar HealthLake datastores com SQL usando o Amazon Athena. Para obter mais informações, consulte Integrando.

HealthLake suporta o seguinte subconjunto de parâmetros de pesquisa do FHIR R4. Para obter mais informações, consulte Parâmetros de pesquisa FHIR R4 para HealthLake.

Tipos de parâmetros de pesquisa compatíveis

A tabela a seguir mostra os tipos de parâmetros de pesquisa suportados em HealthLake.

Tipos de parâmetros de pesquisa compatíveis

Parâmetro de pesquisa	Description
_id	ID do recurso (não é um URL completo)
_Última atualização	Data da última atualização. O servidor tem poder discricionário quanto à precisão do limite.
_tag	Pesquise por uma tag de recurso.
_perfil	Pesquise todos os recursos marcados com um perfil.
_segurança	Pesquise as etiquetas de segurança aplicadas a esse recurso.
_fonte	Pesquise de onde vem o recurso.
_texto	Pesquise a narrativa do recurso.

Parâmetro de pesquisa	Description
createdAt	Pesquise na extensão personalizada CreatedAt .

> **Note**
>
> Os seguintes parâmetros de pesquisa são compatíveis somente com datastores criados após 09 de dezembro de 2023: _security, _source, _text, createdAt.

A tabela a seguir mostra exemplos de como modificar cadeias de caracteres de consulta com base nos tipos de dados especificados para um determinado tipo de recurso. Para maior clareza, os caracteres especiais na coluna de exemplos não foram codificados. Para fazer uma consulta bem-sucedida, certifique-se de que a string de consulta tenha sido codificada corretamente.

Exemplos de parâmetros de pesquisa

Tipos de parâmetros de pesquisa	Detalhes	Exemplos

Número	Pesquisa um valor numérico em um recurso especific ado. Números significa tivos são observados. O número de dígitos significa tivos é específico por valor do parâmetro de pesquisa, excluindo zeros à esquerda. Prefixos de comparação são permitidos.	`[parameter]=100`
		`[parameter]=1e2`
		`[parameter]=lt100`

Data/ DateTime	Pesquisa uma data ou hora específica. O formato esperado éyyyy-mm-d dThh:mm:ss[Z|(+|-) hh:mm] , mas pode variar.	`[parameter]=eq2013 -01-14` `[parameter]=gt2013 -01-14T10:00`

Tipos de parâmetros de pesquisa	Detalhes	Exemplos
	Aceita os seguintes tipos de dados: `date` `dateTimeinstant`,`Period`,, `Timing` e. Para obter mais detalhes sobre o uso desses tipos de dados em pesquisas, consulte a data na documenta ção da API FHIR R4 RESTful . Prefixos de comparação são permitidos.	`[parameter]=ne2013 -01-14`

String	Pesquisa uma sequência de caracteres com distinção entre maiúsculas e minúsculas.
	
	Suporta ambos `HumanName` os `Address` tipos. Para obter mais detalhes, consulte a entrada do tipo de HumanName dados e as entradas do tipo de `Address` dados na documentação do FHIR R4.
	
	A pesquisa avançada é suportada usando `:text` modificadores.
		`[base]/Patient?giv en=eve`
		
		`[base]/Patient?giv en:contains=eve`

Tipos de parâmetros de pesquisa	Detalhes	Exemplos
Token	Pesquisa uma close-to-exact correspondência com uma sequência de caracteres, geralmente comparada a um par de valores de código médico. A distinção entre maiúsculas e minúsculas está vinculada ao sistema de código usado ao criar uma consulta. As consultas baseadas em subsunção podem ajudar a reduzir problemas relacionados à distinção entre maiúsculas e minúsculas. Para maior clareza, não `|` foi codificado.	`[parameter]=[system]|[code]` : Aqui `[system]` se refere a um sistema de codificação e `[code]` se refere ao valor do código encontrado nesse sistema específico. `[parameter]=[code]` : Aqui, sua entrada corresponderá a um código ou a um sistema. `[parameter]=|[code]` : aqui, sua entrada corresponderá a um código e a propriedade do sistema não terá identificador.

Composto	Pesquisa vários parâmetros em um único tipo de recurso, usando os modificadores `$` e a `,` operação.
	
	Prefixos de comparação são permitidos.
		`/Patient?language=FR,NL&language=EN`
		
		`Observation?component-code-value-quantity=http://loinc.org|8480-6$lt60`
		
		`[base]/Group?characteristic-value=gender$mixed`

Tipos de parâmetros de pesquisa	Detalhes	Exemplos

Quantidade	Pesquisa um número, sistema e código como valores. É necessário um número, mas o sistema e o código são opcionais. Com base no tipo de dados de quantidade. Para obter mais detalhes, consulte Quantidade na documentação do FHIR R4.
	
	Usa a seguinte sintaxe presumida `[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`

Referência	Pesquisa referências a outros recursos.	`[base]/Observation?subject=Patient/23`
		
		`test`

URI	Pesquisa uma sequência de caracteres que identifica inequivocamente um recurso específico.	`[base]/ValueSet?url=http://acme.org/fhir/ValueSet/123`
Especial	Pesquisas baseadas em extensões médicas integradas de PNL.	

Parâmetros de pesquisa avançada suportados por HealthLake

HealthLake suporta os seguintes parâmetros de pesquisa avançada.

Name (Nome)	Descrição	Exemplo	Recurso
`_includ`	Usado para solicitar que recursos adicionais sejam retornados em uma solicitaç ão de pesquisa. Ele retorna recursos que são referenci ados pela instância do recurso de destino.	`Encounter ?_include =Encounte r:subject`	
`_revinc de`	Usado para solicitar que recursos adicionais sejam retornados em uma solicitaç ão de pesquisa. Ele retorna recursos que fazem referênci a à instância do recurso primário.	`Patient?_ id=` **`patient-identifier`** `&_revincl ude=Encou nter:patient`	
`_summar`	O resumo pode ser usado para solicitar um subconjunto do recurso.	`Patient?_ summary=text`	Os seguintes parâmetro s de resumo são suportados:`_summary= true` ,`_summary= false` ,`_summary= text` ,`_summary=data` .
`_elemen`	Solicite que um conjunto específico de elementos seja retornado como parte de um recurso nos resultados da pesquisa.	`Patient?_ elements= identifie r,active, link`	

Name (Nome)	Descrição	Exemplo	Recurso
`_total`	Retorna o número de recursos que correspondem aos parâmetros de pesquisa.	`Patient?_total=accurate`	Support `_total=accurate` , `_total=none` .
`_sort`	Indique a ordem de classificação dos resultados da pesquisa retornados usando uma lista separada por vírgulas. O `-` prefixo pode ser usado para qualquer regra de classificação na lista separada por vírgulas para indicar a ordem decrescente.	`Observation?_sort=status,-date`	Support classificar por campos com tipos `Number` , `String` , `Quantity` , `Token` , `URI` , `Reference` . A classificação por só `Date` é compatível com armazenamentos de dados criados após 09 de dezembro de 2023. Support até 5 regras de classificação.
`_count`	Controle quantos recursos são retornados por página do pacote de pesquisa.	`Patient?_count=100`	O tamanho máximo da página é 100.
`chaining`	Elementos de pesquisa dos recursos referenciados. O `.` direciona a pesquisa em cadeia para o elemento dentro do recurso referenciado.	`DiagnosticReport?subject:Patient.name=peter`	
`reverse chaining (_has)`	Pesquise um recurso com base nos elementos dos recursos que se referem a ele.	`Patient?_has:Observation:patient:code=1234-5`	

`_include`

O uso `_include` em uma consulta de pesquisa permite que recursos adicionais do FHIR especificados também sejam retornados. Use `_include` para incluir recursos que estão vinculados diretamente.

Example— Para usar _include para encontrar os pacientes ou o grupo de pacientes que foram diagnosticados com tosse

Você pesquisaria o tipo de `Condition` recurso especificando o código de diagnóstico para tosse e, em seguida, `_include` especificaria que deseja que o `subject` diagnóstico também seja retornado. No tipo de `Condition` recurso, `subject` refere-se ao tipo de recurso do paciente ou ao tipo de recurso do grupo.

Para maior clareza, os caracteres especiais no exemplo não foram codificados. Para fazer uma consulta bem-sucedida, certifique-se de que a sequência de caracteres de consulta tenha sido codificada corretamente.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
Condition?code=49727002&_include=Condition:subject
```

`_include:iterate`Modificador

O `_include:iterate` modificador permite a inclusão recursiva de recursos referenciados em dois níveis. Por exemplo,

```
GET /ServiceRequest?
identifier=025C0931195&_include=ServiceRequest:requester&_include:iterate=PractitionerRole:prac
```

retornará o ServiceRequest, está associado PractitionerRole (por meio da referência do solicitante) e, em seguida, incluirá recursivamente o praticante referenciado por ela. PractitionerRole Esse modificador está disponível para todos os tipos de recursos em HealthLake.

`_include=*`Modificador

O `_include=*` modificador é um curinga que inclui automaticamente todos os recursos diretamente referenciados pelos resultados da pesquisa. Por exemplo,

```
GET /ServiceRequest?specimen.accession=12345&_include=*
```

retornará a correspondência ServiceRequest junto com todos os recursos referenciados (como paciente, médico, amostra etc.) sem precisar especificar cada caminho de referência individualmente. Esse modificador está disponível para todos os tipos de recursos em HealthLake.

`_revinclude`

O uso `_revinclude` em uma consulta de pesquisa permite que recursos adicionais do FHIR especificados também sejam retornados. Use `_revinclude` para incluir recursos vinculados de trás para frente.

Example— Usar _revinclude para incluir tipos de recursos relacionados de Encontro e Observação vinculados a um paciente específico

Para fazer essa pesquisa, você primeiro definiria o indivíduo `Patient` especificando seu identificador no parâmetro de `_id` pesquisa. Em seguida, você especificaria recursos adicionais do FHIR usando a estrutura e. `Encounter:patient` `Observation:patient`

Para maior clareza, os caracteres especiais no exemplo não foram codificados. Para fazer uma consulta bem-sucedida, certifique-se de que a sequência de caracteres de consulta tenha sido codificada corretamente.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
Patient?_id=patient-
identifier&_revinclude=Encounter:patient&_revinclude=Observation:patient
```

`_summary`

O uso `_summary` em uma consulta de pesquisa permite que o usuário solicite um subconjunto do recurso FHIR. Ele pode conter um dos seguintes valores:`true, text, data, false`. Quaisquer outros valores serão tratados como inválidos. Os recursos retornados serão marcados com `'SUBSETTED'` meta.tag, para indicar que os recursos estão incompletos.

- `true`: retorne todos os elementos suportados que estão marcados como 'resumo' na definição básica do (s) recurso (s).
- `text`: retorne somente os elementos 'text', 'id', 'meta' e somente os elementos obrigatórios de nível superior.
- `data`: Retorne todas as partes, exceto o elemento 'texto'.
- `false`: Retorne todas as partes do (s) recurso (s)

Em uma única solicitação de pesquisa, `_summary=text` não pode ser combinado com `_include` ou com parâmetros de `_revinclude` pesquisa.

Example— Obtenha o elemento “texto” dos recursos do paciente em um armazenamento de dados.

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

`_elements`

O uso `_elements` em uma consulta de pesquisa permite que elementos de recursos específicos do FHIR sejam solicitados. Os recursos retornados serão marcados com `'SUBSETTED'` meta.tag, para indicar que os recursos estão incompletos.

O `_elements` parâmetro consiste em uma lista separada por vírgulas de nomes de elementos básicos, como elementos definidos no nível raiz do recurso. Somente os elementos listados devem ser retornados. Se os valores dos `_elements` parâmetros contiverem elementos inválidos, o servidor os ignorará e retornará elementos obrigatórios e válidos.

`_elements`não será aplicável aos recursos incluídos (recursos retornados cujo modo de pesquisa é`include`).

Em uma única solicitação de pesquisa, `_elements` não pode ser combinado com os parâmetros `_summary` de pesquisa.

Example— Obtenha elementos “identificadores”, “ativos” e “vinculados” dos recursos do paciente em seu armazenamento de HealthLake dados.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?
_elements=identifier,active,link
```

`_total`

O uso `_total` em uma consulta de pesquisa retornará o número de recursos que correspondem aos parâmetros de pesquisa solicitados. HealthLake retornará o número total de recursos correspondentes (recursos retornados cujo modo de pesquisa é`match`) na resposta `Bundle.total` da pesquisa.

`_total`suporta os `accurate` valores dos `none` parâmetros. `_total=estimate`não é suportado. Quaisquer outros valores serão tratados como inválidos. `_total`não é aplicável aos recursos incluídos (recursos retornados cujo modo de pesquisa é`include`).

Example— Obtenha o número total de recursos do paciente em um armazenamento de dados:

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

_sort

O uso `_sort` na consulta de pesquisa organiza os resultados em uma ordem específica. Os resultados são ordenados com base na lista separada por vírgulas das regras de classificação em ordem de prioridade. As regras de classificação devem ser parâmetros de pesquisa válidos. Quaisquer outros valores serão tratados como inválidos.

Em uma única solicitação de pesquisa, você pode usar até 5 parâmetros de pesquisa de classificação. Opcionalmente, você pode usar um `-` prefixo para indicar a ordem decrescente. O servidor classificará em ordem crescente por padrão.

Os tipos de parâmetros de pesquisa de classificação suportados são:`Number, String, Date, Quantity, Token, URI, Reference`. Se um parâmetro de pesquisa se referir a um elemento aninhado, esse parâmetro de pesquisa não terá suporte para classificação. Por exemplo, a pesquisa por “nome” do tipo de recurso Paciente se refere ao paciente. O elemento Nome com o tipo de HumanName dados é considerado aninhado. Portanto, não há suporte para classificar os recursos do paciente por 'nome'.

Example— Obtenha os recursos do paciente em um armazenamento de dados e classifique-os por data de nascimento em ordem crescente:

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

_count

O parâmetro `_count` é definido como uma instrução para o servidor sobre quantos recursos devem ser retornados em uma única página.

O tamanho máximo da página é 100. Qualquer valor maior que 100 é inválido. `_count=0`não é suportado.

Example— Pesquise o recurso do paciente e defina o tamanho da página de pesquisa para 25:

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?_count=25
```

Chaining and Reverse Chaining(_has)

O encadeamento e o encadeamento reverso no FHIR fornecem uma maneira mais eficiente e compacta de obter dados interconectados, reduzindo a necessidade de várias consultas separadas e tornando a recuperação de dados mais conveniente para desenvolvedores e usuários.

Se algum nível de recursão retornar mais de 100 resultados, HealthLake retornará 4xx para proteger o armazenamento de dados de ser sobrecarregado e causar várias paginações.

Example— Encadeamento - Obtém tudo DiagnosticReport o que se refere a um paciente em que o nome do paciente é peter.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/DiagnosticReport?
subject:Patient.name=peter
```

Example— Encadeamento reverso - Obtenha recursos do paciente, em que o recurso do paciente é referido por pelo menos uma observação em que a observação tem um código de 1234 e onde a observação se refere ao recurso do paciente no parâmetro de pesquisa do paciente.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?
_has:Observation:patient:code=1234
```

Modificadores de pesquisa compatíveis

Os modificadores de pesquisa são usados com campos baseados em strings. Todos os modificadores de pesquisa em HealthLake uso usam lógica baseada em booleanos. Por exemplo, você pode `:contains` especificar que um campo de string maior inclua uma string pequena para que seja incluído nos resultados da pesquisa.

Modificadores de pesquisa compatíveis

Modificador de pesquisa	Tipo
:ausente	Todos os parâmetros, exceto `Composite`
:exato	String
:contém	String
:não	Token

Modificador de pesquisa	Tipo
:texto	Token
:identificador	Referência
:abaixo	URI

Comparadores de pesquisa compatíveis

Você pode usar comparadores de pesquisa para controlar a natureza da correspondência em uma pesquisa. Você pode usar comparadores ao pesquisar nos campos de número, data e quantidade. A tabela a seguir lista os comparadores de pesquisa e suas definições que são suportadas pelo HealthLake.

Comparadores de pesquisa compatíveis

Comparador de pesquisa	Description
eq	O valor do parâmetro no recurso é igual ao valor fornecido.
ne	O valor do parâmetro no recurso não é igual ao valor fornecido.
gt	O valor do parâmetro no recurso é maior que o valor fornecido.
lt	O valor do parâmetro no recurso é menor que o valor fornecido.
idade	O valor do parâmetro no recurso é maior ou igual ao valor fornecido.
le	O valor do parâmetro no recurso é menor ou igual ao valor fornecido.
mar	O valor do parâmetro no recurso começa após o valor fornecido.

Comparador de pesquisa	Description
eb	O valor do parâmetro no recurso termina antes do valor fornecido.

Parâmetros de pesquisa FHIR não suportados pelo HealthLake

HealthLake suporta todos os parâmetros de pesquisa do FHIR, com exceção dos listados na tabela a seguir. Para obter uma lista completa dos parâmetros de pesquisa do FHIR, consulte o registro de parâmetros de pesquisa do FHIR.

Parâmetros de pesquisa não suportados

Composição do pacote	Localização próxima
Identificador de pacote	C onsent-source-reference
Mensagem do pacote	Paciente contratado
Tipo de pacote	Conteúdo do recurso
Carimbo de data/hora do pacote	Consulta de recursos

FHIR R4 para **`$operations`** HealthLake

As operações FHIR$ (também chamadas de “operações em dólares”) são funções especiais do lado do servidor que se estendem além das operações CRUD padrão (`Create`,, `ReadUpdate`,`Delete`) na especificação FHIR. Essas operações são identificadas pelo prefixo “$” e permitem processamento complexo, transformação de dados e operações em massa que seriam difíceis ou ineficientes de realizar usando chamadas de API REST padrão. $ As operações podem ser invocadas no nível do sistema, no nível do tipo de recurso ou em instâncias de recursos específicas, fornecendo maneiras flexíveis de interagir com os dados do FHIR. AWS HealthLake suporta vários FHIR$`operations`. Consulte cada página individual abaixo para obter detalhes adicionais.

Tópicos

- Operação FHIR R4 para $attribution-status HealthLake
- Excluindo tipos de recursos com $bulk-delete

- operação $bulk-member-match para HealthLake
- Operação FHIR R4 para $confirm-attribution-list HealthLake
- Operação FHIR R4 para $davinci-data-export HealthLake
- Gerando documentos clínicos com $document
- Removendo recursos permanentemente com $erase
- Obtendo dados do paciente com Patient/$everything
- Recuperando ValueSet códigos com $expand
- Exportação de HealthLake dados com o FHIR $export
- Operação FHIR para $inquire HealthLake
- Recuperando detalhes do conceito com $lookup
- $member-addoperação para HealthLake
- $member-matchoperação para HealthLake
- $member-removeoperação para HealthLake
- Removendo recursos do compartimento do paciente com $purge
- Operação FHIR para $questionnaire-package HealthLake
- Operação FHIR para $submit HealthLake
- Validando recursos do FHIR com $validate

Operação FHIR R4 para **`$attribution-status`** HealthLake

Recupera o status de atribuição de um membro específico, devolvendo um pacote contendo todos os recursos de atribuição relacionados ao paciente. Essa operação faz parte da implementação da Lista IG 2.1.0 da Lista de Atribuição de Membros (ATR) da FHIR.

Endpoint

```
POST [base]/Group/[id]/$attribution-status
```

Parâmetros da solicitação

A operação aceita os seguintes parâmetros opcionais:

Parâmetro	Tipo	Description
memberId	Identificador	O MemberId do membro para quem o status de atribuição é solicitado
Referência do paciente	Referência	Referência ao recurso do paciente no sistema do produtor

> **Note**
>
> Um `memberId` ou `patientReference` pode ser fornecido, ou ambos para fins de validação.

Exemplo de solicitação

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "memberId",
      "valueIdentifier": {
        "system": "http://example.org",
        "value": "MBR123456789"
      }
    },
    {
      "name": "patientReference",
      "valueReference": {
        "reference": "Patient/patient-123",
        "display": "John Doe"
      }
    }
  ]
}
```

Resposta

Retorna um pacote contendo recursos de atribuição relacionados ao paciente:

Recurso	Cardinalidade	Local
Paciente	1..1	grupo.membro.entidade
Cobertura	0,1	group.member.extension: referência de cobertura
Organization/Practitioner/PractitionerRole	0,1	group.member.extension: AttributedProvider
Qualquer recurso	0,1	group.member.extension: dados associados

Resposta da amostra


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

Tratamento de erros

A operação trata das seguintes condições de erro:

Erro	Status HTTP	Description
Solicitação de operação inválida	400	Parâmetros ou estrutura de solicitação não conformes
Recurso de grupo não encontrado	404	O ID de grupo especificado não existe
Recurso do paciente não encontrado	404	A referência de paciente especificada não existe

Autorização e segurança

Requisitos do SMART Scope

Os clientes devem ter privilégios apropriados para ler os recursos do Grupo e os recursos de atribuição relacionados

Os mecanismos de autorização padrão do FHIR se aplicam a todas as operações

Excluindo tipos de recursos com **`$bulk-delete`**

AWS HealthLake dá suporte à `$bulk-delete` operação, permitindo a exclusão de todos os recursos de um tipo específico em um armazenamento de dados. Essa operação é particularmente útil quando você precisa:

- Realize auditorias e limpezas sazonais
- Gerencie o ciclo de vida dos dados em grande escala
- Remover tipos de recursos específicos
- Cumpra as políticas de retenção de dados

Usage

A `$bulk-delete` operação pode ser invocada usando métodos POST:

```
POST [base]/[ResourceType]/$bulk-delete?isHardDelete=false&deleteAuditEvent=true
```

Parâmetros

Parâmetro	Tipo	Obrigatório	Padrão	Description
`isHardDelete`	booliano	Não	false	Quando verdadeiro, remove permanentemente os recursos do armazenamento
`deleteAuditEvent`	booleano	Não	true	Quando verdadeiro, exclui os eventos de auditoria associados
`_since`	string	Não	Hora de criação do	Quando inserido, seleciona a hora limite inicial para encontrar recursos com

Parâmetro	Tipo	Obrigatório	Padrão	Description
			armazenamento de dados	base na hora da Última Modificação. Não pode ser usado com início ou fim
`start`	string	Não	Hora de criação do armazenamento de dados	Quando inserido, seleciona o horário limite para encontrar recursos com base no horário da Última Modificação. Pode ser usado com extremidade
`end`	string	Não	Horário de envio do trabalho	Quando inserido, seleciona a hora limite final para encontrar recursos com base na hora da Última Modificação

Exemplos

Exemplo de solicitação

```
POST [base]/Observation/$bulk-delete?isHardDelete=false
```

Exemplo de resposta

```
{
      "jobId": "jobId",
      "jobStatus": "SUBMITTED"
    }
```

Status do trabalho

Para verificar o status de uma tarefa de exclusão em massa:

```
GET [base]/$bulk-delete/[jobId]
```

A operação retorna informações sobre o status do trabalho:

```
{
```

```
    "datastoreId": "datastoreId",
    "jobId": "jobId",
    "status": "COMPLETED",
    "submittedTime": "2025-10-09T15:09:51.336Z"
}
```

Comportamento

A `$bulk-delete` operação:

1. Processa de forma assíncrona para lidar com grandes volumes de recursos
2. Mantém as transações ACID para a integridade dos dados
3. Fornece rastreamento do status do trabalho com contagens de exclusões de recursos
4. Suporta os modos de exclusão temporária e definitiva
5. Inclui registro abrangente de auditoria das atividades de exclusão
6. Permite a exclusão seletiva de versões históricas e eventos de auditoria

Registro de auditoria

A `$bulk-delete` operação é registrada como Iniciar FHIRBulk DeleteJob e Descrever FHIRBulk DeleteJob com informações detalhadas da operação.

Limitações

- Quando `isHardDelete` definido como verdadeiro, os recursos excluídos permanentemente não aparecerão nos resultados da pesquisa ou `_history` nas consultas.
- Os recursos que estão sendo excluídos por meio dessa operação podem ficar temporariamente inacessíveis durante o processamento.
- A medição de armazenamento é ajustada somente nas versões históricas - deleteVersionHistory =false não ajusta o armazenamento do armazenamento de dados

operação `$bulk-member-match` para HealthLake

AWS HealthLake suporta a `$bulk-member-match` operação de processamento de solicitações de correspondência de vários membros de forma assíncrona. Essa operação permite que as organizações de saúde combinem com eficiência centenas de identificadores exclusivos de membros em diferentes sistemas de saúde usando informações demográficas e de cobertura em uma única solicitação em massa. Esse recurso é essencial para troca de dados de pagador para

pagador em grande escala, transições de membros e requisitos de conformidade do CMS e segue a especificação FHIR.

> **Note**
>
> A `$bulk-member-match` operação é baseada em uma especificação FHIR subjacente que atualmente é experimental e está sujeita a alterações. Conforme a especificação evolui, o comportamento e a interface dessa API serão atualizados adequadamente. Recomenda-se que os desenvolvedores monitorem as notas de HealthLake lançamento da AWS e as atualizações relevantes da especificação FHIR para se manterem informados sobre quaisquer alterações que possam afetar suas integrações.

Essa operação é particularmente útil quando você precisa:

- Processe a correspondência de membros em grande escala durante os períodos de inscrição aberta
- Facilite as transições de membros em massa entre pagadores
- Support os requisitos de troca de dados de conformidade do CMS em grande escala
- Combine eficientemente coortes de membros em todas as redes de saúde
- Minimize as chamadas de API e melhore a eficiência operacional para cenários de correspondência de alto volume

Usage

A `$bulk-member-match` operação é uma operação assíncrona invocada em recursos do Grupo usando o método POST:

```
POST [base]/Group/$bulk-member-match
```

Depois de enviar uma solicitação de correspondência em massa, você pode pesquisar o status do trabalho usando:

```
GET [base]/$bulk-member-match-status/{jobId}
```

Parâmetros compatíveis

HealthLake suporta os seguintes parâmetros FHIR: `$bulk-member-match`

Parâmetro	Tipo	Obrigatório	Description
`MemberPatient`	Paciente	Sim	Recurso para pacientes contendo informações demográficas para o membro a ser combinado.
`CoverageToMatch`	Cobertura	Sim	Recurso de cobertura que será usado para comparar com os registros existentes.
`CoverageToLink`	Cobertura	Não	Recurso de cobertura a ser vinculado durante o processo de correspondência.
`Consent`	Consentimento	Sim	O recurso de consentimento para fins de autorização também é armazenado. Isso é diferente da `$member-match` operação individual em que o consentimento não é necessário.

Solicitação POST para enviar uma vaga de correspondência de membros em massa

O exemplo a seguir mostra uma solicitação POST para enviar um trabalho por correspondência de membros em massa. Cada membro é encapsulado em um `MemberBundle` parâmetro contendo os `Consent` recursos e os recursos necessários `MemberPatientCoverageToMatch`, além de um opcional`CoverageToLink`.

```
POST [base]/Group/$bulk-member-match
Content-Type: application/fhir+json
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "MemberBundle",
      "part": [
        {
          "name": "MemberPatient",
          "resource": {
            "resourceType": "Patient",
            "identifier": [
              {
                "system": "http://example.org/patient-id",
                "value": "patient-0"
```

```
            }
          ],
          "name": [
            {
              "family": "Smith",
              "given": ["James"]
            }
          ],
          "gender": "male",
          "birthDate": "1950-01-01"
        }
      },
      {
        "name": "CoverageToMatch",
        "resource": {
          "resourceType": "Coverage",
          "status": "active",
          "identifier": [
            {
              "system": "http://example.org/coverage-id",
              "value": "cov-0"
            }
          ],
          "subscriberId": "sub-0",
          "beneficiary": {
            "reference": "Patient/patient123"
          },
          "relationship": {
            "coding": [
              {
                "system": "http://terminology.hl7.org/CodeSystem/subscriber-
relationship",
                "code": "self"
              }
            ]
          },
          "payor": [
            {
              "reference": "Organization/org123"
            }
          ]
        }
      },
      {
```

```
          "name": "Consent",
          "resource": {
            "resourceType": "Consent",
            "status": "active",
            "scope": {
              "coding": [
                {
                  "system": "http://terminology.hl7.org/CodeSystem/consentscope",
                  "code": "patient-privacy"
                }
              ]
            },
            "category": [
              {
                "coding": [
                  {
                    "system": "http://terminology.hl7.org/CodeSystem/v3-ActCode",
                    "code": "IDSCL"
                  }
                ]
              }
            ],
            "patient": {
              "reference": "Patient/patient123"
            },
            "performer": [
              {
                "reference": "Patient/patient123"
              }
            ],
            "sourceReference": {
              "reference": "http://example.org/DocumentReference/consent-source"
            },
            "policy": [
              {
                "uri": "http://hl7.org/fhir/us/davinci-hrex/StructureDefinition-hrex-
consent.html#regular"
              }
            ],
            "provision": {
              "type": "permit",
              "period": {
                "start": "2024-01-01",
                "end": "2025-12-31"
```

```
            },
            "actor": [
              {
                "role": {
                  "coding": [
                    {
                      "system": "http://terminology.hl7.org/CodeSystem/provenance-
participant-type",
                      "code": "performer"
                    }
                  ]
                },
                "reference": {
                  "identifier": {
                    "system": "http://hl7.org/fhir/sid/us-npi",
                    "value": "9876543210"
                  },
                  "display": "Old Health Plan"
                }
              },
              {
                "role": {
                  "coding": [
                    {
                      "system": "http://terminology.hl7.org/CodeSystem/v3-
ParticipationType",
                      "code": "IRCP"
                    }
                  ]
                },
                "reference": {
                  "identifier": {
                    "system": "http://hl7.org/fhir/sid/us-npi",
                    "value": "0123456789"
                  },
                  "display": "New Health Plan"
                }
              }
            ],
            "action": [
              {
                "coding": [
                  {
                    "system": "http://terminology.hl7.org/CodeSystem/consentaction",
```

```
                    "code": "disclose"
                  }
                ]
              }
            ]
          }
        }
      },
      {
        "name": "CoverageToLink",
        "resource": {
          "resourceType": "Coverage",
          "status": "active",
          "identifier": [
            {
              "system": "http://example.org/coverage-link-id",
              "value": "cov-link-0"
            }
          ],
          "subscriberId": "new-sub-0",
          "beneficiary": {
            "reference": "Patient/patient123"
          },
          "relationship": {
            "coding": [
              {
                "system": "http://terminology.hl7.org/CodeSystem/subscriber-
relationship",
                "code": "self"
              }
            ]
          },
          "payor": [
            {
              "identifier": {
                "system": "http://hl7.org/fhir/sid/us-npi",
                "value": "0123456789"
              },
              "display": "New Health Plan"
            }
          ]
        }
      }
    ]
```

```
    }
  ]
}
```

Resposta de trabalho concluída com saída

Quando o trabalho é concluído, a resposta inclui metadados do trabalho e um recurso de parâmetros FHIR contendo três recursos de grupo que categorizam os resultados da correspondência.

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

Como HealthLake classifica os membros em grupos de saída

Cada membro enviado em uma `$bulk-member-match` solicitação é avaliado por meio de um pipeline sequencial. O resultado de cada etapa determina em qual grupo de saída o membro é colocado.

1. Validação estrutural — Está em MemberBundle conformidade com os perfis exigidos? Em caso de falha: Erro (não em nenhum grupo).
2. Correspondência de pacientes — É possível HealthLake encontrar pacientes que correspondam aos dados demográficos enviados? Em caso de falha: NonMatchedMembers.
3. Confirmação de cobertura — Posso HealthLake restringir a exatamente um paciente com validade CoverageToMatch? Em caso de falha: NonMatchedMembers.
4. Avaliação do consentimento — O consentimento enviado é honroso no momento? (status = ativo, o período cobre a data atual, o artista pode ser validado). Em caso de falha: ConsentConstrainedMembers.
5. Sucesso — Todas as verificações são aprovadas. Consentimento armazenado no armazenamento de dados. Membro inserido MatchedMembers.

Princípio fundamental: um membro só pode aparecer em um destino. A primeira etapa com falha determina o posicionamento. Os membros que falham nas etapas 2 ou 3 nunca são colocados ConsentConstrainedMembers - esse grupo é exclusivamente para membros que se candidataram com sucesso, mas cujo consentimento não pode ser honrado.

Detalhes da avaliação do consentimento (Etapa 4):

- Verificação 1 — Status de consentimento: É `Consent.status` igual a “ativo”? Se não → ConsentConstrainedMembers.
- Verificação 2 — Período de provisão: `provision.period` Cobre a data atual? Se a data atual for antes `period.start` ou depois de `period.end` → ConsentConstrainedMembers.
- Verificação 3 — Validação do executante: A `Consent.performer` referência pode ser validada? Se o recurso referenciado não for encontrado no armazenamento de dados ou não estiver associado ao paciente correspondente →. ConsentConstrainedMembers

Todas as verificações devem ser aprovadas para que o membro seja inserido MatchedMembers e para que o consentimento seja armazenado.

Comportamento de correspondência de cobertura

Durante a correspondência de membros, somente `CoverageToMatch` é validado no armazenamento de dados do pagador respondente. `CoverageToLink`pertence ao new/requesting pagador e não é validado no armazenamento de dados do pagador antigo. A inclusão `CoverageToLink` na solicitação não afetará os resultados correspondentes.

Cada combinação de Paciente e Cobertura na solicitação é processada de forma independente. O mesmo paciente pode ser enviado várias vezes com planos de cobertura diferentes, e cada inscrição recebe seu próprio resultado com base em sua cobertura específica.

Tratamento de referências do executante de consentimento

O novo pagador pode enviar uma referência temporária ou local do paciente `Consent.performer` (por exemplo, a mesma referência usada em`Consent.patient`). HealthLake resolve essas referências automaticamente:

- Se `Consent.performer` contiver a mesma referência local de`Consent.patient`, HealthLake substitui-a pela referência real do paciente correspondente após a correspondência ser bem-sucedida.
- HealthLake suporta referências de artistas do tipo Paciente RelatedPerson, Profissional e Organização (referências diretas e referências de identificadores lógicos). PractitionerRole
- Se a validação do executor falhar (recurso não encontrado ou não associado ao paciente correspondente), o membro será inserido em ConsentConstrainedMembers vez de retornar um erro.

Recursos do Grupo de Saída

O trabalho concluído retorna um recurso de Parâmetros contendo três recursos do Grupo:

MatchedMembers Group (Grupo)

Contém referências de pacientes para todos os membros combinados com sucesso cujo consentimento está ativo e válido no momento da solicitação. O recurso Consent é criado e armazenado no armazenamento de dados para cada membro correspondente. Esse grupo é instanciado no armazenamento de dados e pode ser usado diretamente com o. `$davinci-data-export`

NonMatchedMembers Group (Grupo)

Contém referências a membros nos quais nenhuma correspondência exclusiva foi encontrada. Um membro é colocado aqui quando nenhum paciente no armazenamento de dados corresponde aos dados demográficos fornecidos, não existe cobertura válida para nenhum candidato a paciente compatível ou vários pacientes correspondem aos dados demográficos e vários têm cobertura válida (ambígua).

ConsentConstrainedMembers Group (Grupo)

Contém referências de pacientes para membros que foram combinados com sucesso (dados demográficos e cobertura confirmados), mas cujo consentimento não pode ser honrado no momento da solicitação. O recurso Consent não é armazenado para membros com restrições de consentimento. A identidade do membro correspondente (MemberIdentifier e MemberId) ainda está incluída para que o pagador solicitante saiba quem foi restringido.

O `Group.quantity` campo contém a contagem total de membros em cada um de seus respectivos grupos.

Referências dos membros do grupo:

- `Group.member.entity.reference`— Para MatchedMembers e ConsentConstrainedMembers, contém a ID do paciente do membro correspondente no sistema do pagador respondente. Para NonMatchedMembers, faz referência à entrada Patient contida.
- `Group.member.entity.extension (base-ext-match-parameters)`— Contém o ID do paciente da solicitação de entrada original (o ID enviado pelo pagador solicitante, derivado de `Patient.idCoverage.beneficiary.reference`, ou`Consent.patient.reference`).

Consent-Patient ligação

> **Important**
>
> O recurso de consentimento armazenado retém a referência do paciente exatamente conforme enviada pelo pagador solicitante. HealthLake não atualiza automaticamente o campo do paciente do consentimento para apontar para o paciente correspondente no armazenamento de dados receptor.

Para vincular um consentimento armazenado ao paciente correspondente, use o resultado do trabalho: cada membro do MatchedMembers grupo tem um `member.entity.reference` apontando para o paciente correspondente e um `member.entity.extension` (`base-ext-match-parameters`) apontando para o paciente de entrada contido. Cross-reference estes com o campo do paciente do Consent para criar o mapeamento em sua camada de aplicação.

O que é armazenado versus o que é transitório

A tabela a seguir documenta o que HealthLake persiste no armazenamento de dados durante o `$bulk-member-match` processamento e o que existe somente na resposta do trabalho:

Recurso	Armazenado?	Pode ser consultado via REST?	Observações
MemberPatient (entrada)	Não	Não	Usado somente para correspondência; não persiste
CoverageToMatch (entrada)	Não	Não	Usado somente para confirmação de cobertura
CoverageToLink (entrada)	Não	Não	Não validado em relação ao armazenamento de dados; pertence ao novo pagador
Consentimento (membros correspondentes)	Sim	Sim — GET [base] / Consent/ {id}	Armazenado como recebido do pagador solicitante
Consentimento (membros restritos)	Não	Não	Não armazenado. A identidade do membro ainda está incluída na resposta.
MatchedMembers Grupo (saída)	Sim	Sim — GET [base] / Group/ {id}	Instanciado; utilizável com $davinci-data-export

Recurso	Armazenado?	Pode ser consultado via REST?	Observações
NonMatchedMembers Grupo	Não	Não	Somente resposta de emprego
ConsentConstrained Members Grupo	Não	Não	Somente resposta de emprego

Integração com $davinci-data-export

O recurso de MatchedMembers grupo retornado por `$bulk-member-match` pode ser usado diretamente com a `$davinci-data-export` operação para recuperar dados de membros em massa:

```
POST [base]/Group/{matched-group-id}/$davinci-data-export
GET [base]/Group/{matched-group-id}
```

Essa integração permite fluxos de trabalho eficientes em que você primeiro identifica os membros correspondentes em massa e depois exporta seus registros de saúde completos usando o recurso de grupo resultante.

Usando $member-remove antes da exportação

Se você precisar excluir membros específicos da exportação após a correspondência (por exemplo, um membro revoga o consentimento entre a correspondência e a exportação), use `$member-remove` no MatchedMembers Grupo.

> **Important**
>
> Remover um membro via `$member-remove` marca o membro como inativo no Grupo, mas `$davinci-data-export` só exclui membros inativos depois que o Grupo é atualizado para o status “final”. Se você chamar `$davinci-data-export` um grupo que ainda tem o status padrão, os membros removidos ainda poderão aparecer nos resultados de exportação.

Fluxo de trabalho:

1. `POST [base]/Group/{id}/$member-remove`— marcar membros inativos
2. `PUT [base]/Group/{id}`— atualize o status do grupo para “final”
3. `POST [base]/Group/{id}/$davinci-data-export`— a exportação agora exclui membros removidos

Características de desempenho

A `$bulk-member-match` operação foi projetada para processamento de alto volume e é executada de forma assíncrona:

- Concorrência: máximo de 5 operações simultâneas por armazenamento de dados.
- Escalabilidade: lida com até 500 membros por solicitação (limite de carga útil de 5 MB).
- Operações paralelas: Compatível com operações simultâneas de importação, exportação ou exclusão em massa.

Autorização

A API usa SMART no protocolo de autorização FHIR com os seguintes escopos obrigatórios:

- `system/Patient.read`— Necessário para pesquisar e combinar os recursos do paciente.
- `system/Coverage.read`— Necessário para validar as informações de cobertura.
- `system/Group.write`— Necessário para criar recursos do Grupo de resultados.
- `system/Organization.read`— Condicional, obrigatório se a cobertura fizer referência a organizações.
- `system/Practitioner.read`— Condicional, exigido se a cobertura fizer referência a profissionais.
- `system/PractitionerRole.read`— Condicional, exigido se a cobertura fizer referência às funções dos profissionais.
- `system/Consent.write`— Condicional, exigido se os recursos de consentimento forem fornecidos.

A operação também oferece suporte à autorização AWS do IAM Signature Version 4 (SigV4) para acesso programático.

Regras de validação

As regras de validação a seguir são aplicadas a cada uma MemberBundle na Etapa 1. Os membros que falham na validação são relatados como erros e não aparecem em nenhum grupo de saída.

MemberPatient

Campo	Como HealthLake o usa	Falha na validação se...
`name.family`	Pesquisa demográfica	Missing (Ausente)
`name.given`	Pesquisa demográfica	Falta (é necessário pelo menos um)
`birthDate`	Pesquisa demográfica	Missing (Ausente)
`gender`	Pesquisa demográfica; se ausente, extensão Birth Sex usada	Nem sexo nem nascimento. Sexo presente (hrex-pat-1)
`identifier`	Incluído na pesquisa quando presente; melhora a confiança	Nunca causa falha (opcional)

CoverageToMatch / CoverageToLink

Campo	Como HealthLake o usa	Falha na validação se...
`status`	Confirma que a cobertura é acionável	Missing (Ausente)
`beneficiary`	Vincula a cobertura a um candidato a paciente	Missing (Ausente)
`payor`	Desambiguação quando existem vários candidatos	Falta ou mais de um pagador
`relationship`	Confirma a relação assinante-beneficiário	Missing (Ausente)

Campo	Como HealthLake o usa	Falha na validação se...
`identifier (MB) or subscriberId`	Chave primária de desambiguação	Nenhum presente

Consentimento

Campo	Como HealthLake o usa	Falha na validação se...
`scope`	Confirma que o escopo do consentimento é a privacidade do paciente	Código de privacidade do paciente ausente ou inexistente
`category`	Confirma a classificação da divulgação	Missing (Ausente)
`patient`	Identifica o sujeito do consentim ento	Missing (Ausente)
`performer`	Identifica quem está concordando	Missing (Ausente)
`sourceReference`	Documenta a fonte de consentim ento	Missing (Ausente)
`policy.uri`	Determina o escopo do compartil hamento de dados	Falta ou o URI não termina em #regular ou #sensitive
`provision.type`	Deve ser “permissão” de acordo com o perfil de consentimento da HRex	“Permissão” ausente ou não (incluindo “negar”)
`provision .period`	Avaliado na Etapa 4 para verificaç ão com restrição de consentimento	Desaparecido ou não start/end
`status`	Avaliado na Etapa 4 (NÃO na Etapa 1)	Nunca causa falha na Etapa 1 — HealthLake aceita qualquer status válido e avalia na Etapa 4

> **Note**
>
> O perfil HRex Consent define o status com um valor fixo de “ativo”. HealthLake relaxa intencionalmente essa restrição para que um consentimento não ativo receba uma classificação significativa (ConsentConstrainedMembers) em vez de uma rejeição geral de validação.

Comportamento de correspondência

- Pesquisa de pacientes (Etapa 2) — HealthLake pesquisas usando `name.family` + `name.given` (exato, sem distinção entre maiúsculas e minúsculas), `birthDate` (exato), `gender` (exato; sexo do nascimento usado se o sexo não existir) e `identifier` (quando presente, opcional).
- Desambiguação da cobertura (Etapa 3) — Quando vários candidatos a pacientes são encontrados, `CoverageToMatch` é usada para restringir a um. Uma cobertura é “válida” quando existe um recurso de cobertura ativo no armazenamento de dados que corresponda a `subscriberId` ou `identifier` (tipo MB) AND. `payor`
- Avaliação do consentimento (Etapa 4) - Realizada somente após uma combinação única bem-sucedida. Consulte a seção de detalhes da avaliação de consentimento acima.

Tratamento de erros

A operação trata das seguintes condições de erro:

- 400 Solicitação inválida: formato de solicitação inválido, parâmetros obrigatórios ausentes ou a carga excede os limites de tamanho (500 membros ou 5 MB).
- 422 Entidade não processável: erros de processamento durante a execução do trabalho.
- Erros individuais de membros: quando um membro específico não consegue processar, a operação continua com os membros restantes e inclui detalhes do erro no NonMatchedMembers Grupo com códigos de motivo apropriados. Por exemplo, um `MemberBundle` paciente sem o `birthDate` parâmetro retornará o seguinte erro:

```
"errors": [
  {
    "memberIndex": 1,
    "jsonBlob": {
      "resourceType": "OperationOutcome",
```

```
      "issue": [
        {
          "severity": "error",
          "code": "invalid",
          "diagnostics": "MemberPatient.birthDate is required"
        }
      ],
      "statusCode": 400
    }
  }
]
```

Os detalhes do erro estão disponíveis por meio do endpoint da pesquisa de status e incluem:

- `numberOfMembersWithCustomerError`: Contagem de membros com erros de validação ou entrada.
- `numberOfMembersWithServerError`: contagem de membros com erros de processamento no servidor.

Operações do relacionadas

- the section called “$member-match”— Operação de correspondência de membros individuais.
- the section called “$davinci-data-export”— Exportação de dados em massa usando recursos do Grupo.
- the section called “$Operações”— Lista completa das operações suportadas.

Operação FHIR R4 para **`$confirm-attribution-list`** HealthLake

Indica ao produtor que o consumidor não tem mais alterações a fazer na Lista de atribuição. Essa operação finaliza a lista de atribuição removendo membros inativos da lista, alterando o status para “final” e confirmando a operação. Essa operação faz parte da implementação da FHIR Member Attribution (ATR) List IG 2.1.0.

Endpoint

```
POST [base]/Group/[id]/$confirm-attribution-list
```

Solicitação

Não são necessários parâmetros.

```
POST [base]/Group/[id]/$confirm-attribution-list
```

Resposta

Retorna HTTP 201 com uma mensagem de confirmação.

Resposta da amostra

```
HTTP Status Code: 201

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "message",
      "valueString": "Accepted."
    }
  ]
}
```

Status do grupo após a confirmação

Após a confirmação bem-sucedida, o recurso do Grupo terá o status da lista de atribuições definido como “final”:

```
{
  "resourceType": "Group",
  "id": "fullexample",
  "extension": [
    {
      "url": "http://hl7.org/fhir/us/davinci-atr/StructureDefinition/ext-
attributionListStatus",
      "valueCode": "final"
    }
  ]
  // ... other Group properties
}
```

Tratamento de erros

A operação trata das seguintes condições de erro:

Erro	Status HTTP	Description
Solicitação de operação inválida	400	Parâmetros ou estrutura de solicitação não conformes
Recurso de grupo não encontrado	404	O ID de grupo especificado não existe

Autorização e segurança

Requisitos do SMART Scope

Os clientes devem ter privilégios apropriados para ler os recursos do Grupo e os recursos de atribuição relacionados

Pois`$confirm-attribution-list`, os clientes também devem ter privilégios de gravação para modificar os recursos do Grupo

Os mecanismos de autorização padrão do FHIR se aplicam a todas as operações

Operação FHIR R4 para **`$davinci-data-export`** HealthLake

A `$davinci-data-export` operação é uma operação FHIR assíncrona que você pode usar para exportar dados de saúde. AWS HealthLake Essa operação oferece suporte a vários tipos de exportação, incluindo Atribuição de Membro (ATR) Payer-to-Payer, Acesso ao PDex Provedor e Acesso ao Membro. APIs É uma versão especializada da `$export` operação padrão do FHIR, projetada para atender aos requisitos dos guias de DaVinci implementação.

Recursos principais

- Processamento assíncrono: segue o padrão de solicitação assíncrona FHIR
- Exportação em nível de grupo: exporta dados para membros em um recurso de grupo específico
- Vários tipos de exportação: suporta ATR (atribuição de membros), acesso de PDex provedor e acesso de membros Payer-to-Payer APIs

- Suporte abrangente de perfis: inclui US Core, CARIN Blue Button e PDex perfis
- Filtragem flexível: suporta a filtragem por pacientes, tipos de recursos e intervalos de tempo
- Saída NDJSON: fornece dados no formato JSON delimitado por nova linha

Ponto final da operação

```
GET [base]/Group/[id]/$davinci-data-export
POST [base]/Group/[id]/$davinci-data-export
```

Parâmetros da solicitação

Parâmetro	Cardinalidade	Description
`patient`	0.. *	Membros específicos cujos dados devem ser exportados. Quando omitidos, todos os membros do Grupo são exportados.
`_type`	0,1	Lista delimitada por vírgula dos tipos de recursos FHIR a serem exportados. Quando omitidos, todos os tipos de recursos compatíve is com o tipo de exportação especificado são incluídos. Para exportações de ATR, o padrão é de 8 tipos de recursos de atribuição. Para PDex exportações, isso inclui todos os tipos de recursos de atribuição, além dos tipos de recursos clínicos e de sinistros do US Core, do CARIN Blue Button e dos perfis. PDex
`_since`	0,1	Inclua somente recursos atualizados após essa data e hora.
`_until`	0,1	Inclua somente recursos atualizados antes dessa data e hora.
`exportType`	0,1	Tipo de exportação a ser executada. Valores válidos: `hl7.fhir.us.davinci-atr`, `hl7.fhir.us.davinci-pdex` ,

Parâmetro	Cardinalidade	Description
		`hl7.fhir.us.davinci-pdex.p2p` , `hl7.fhir.us.davinci-pdex.member` . Padrão: `hl7.fhir.us.davinci-atr` .
`_includeEOB2xWoFinancial`	0,1	Especifica se os recursos do CARIN BB 2.x devem ser incluídos com ExplanationOfBenefit os dados financeiros removidos. Padrão: `false`.
`_security`	0.. *	Filtre os recursos exportados por valores de `meta.security` codificação. Use o `system|code` formato (o caractere de barra vertical deve ser codificado como URL). `%7C` Quando vários valores são fornecidos, os recursos devem corresponder a todos eles (E à semântica). Use `system|` (tubo de arrasto, sem código) para combinar qualquer código de um determinado sistema.
`_tag`	0.. *	Filtre os recursos exportados por valores de `meta.tag` codificação. Usa o mesmo `system|code` formato e semântica AND de`_security` . Quando ambos `_security` `_tag` são especificados, os recursos devem corresponder aos dois filtros.

Tipos de recursos compatíveis

Os tipos de recursos suportados dependem do tipo de exportação que você especificar. Para exportações de ATR, os seguintes tipos de recursos são suportados:

- `Group`
- `Patient`
- `Coverage`
- `RelatedPerson`

- `Practitioner`
- `PractitionerRole`
- `Organization`
- `Location`

Para PDex exportações (Provider Access e Member Access), todos os tipos de recursos clínicos e de sinistros são suportados, além dos tipos anteriores. Payer-to-Payer Para obter uma lista completa dos tipos de recursos suportados, consulte o US Core Implementation Guide (STU 6.1), o CARIN Blue Button Implementation Guide e o Da Vinci Prior Authorization Support Implementation Guide.

Tipos de exportação

A `$davinci-data-export` operação oferece suporte aos seguintes tipos de exportação. Você especifica o tipo de exportação usando o `exportType` parâmetro.

Tipo de exportação	Finalidade	Escopo de dados	Limite temporal
`hl7.fhir.us.davinci-atr`	Lista de atribuição de membros	Recursos relacionados à atribuição	Nenhum
`hl7.fhir.us.davinci-pdex`	API de acesso do provedor	Dados clínicos e de reclamações de pacientes atribuídos	5 anos
`hl7.fhir.us.davinci-pdex.p2p`	Payer-to-Payer Troca	Dados históricos de associados para transições de seguros	5 anos
`hl7.fhir.us.davinci-pdex.member`	API de acesso de membros	Dados de saúde do próprio membro	5 anos

> **Note**
>
> Para PDex exportações, o limite temporal de 5 anos não se aplica aos tipos de recursos ATR (`Group`,,`Patient`,`Coverage`,`RelatedPerson`,, `Practitioner`

`PractitionerRoleOrganization`,`Location`). Esses recursos estão sempre incluídos, independentemente da idade.

ATR (hl7.fhir.us.davinci-atr)

Com o tipo de exportação ATR, você pode exportar dados da Lista de Atribuição de Membros. Use esse tipo de exportação para recuperar recursos relacionados à atribuição para membros de um grupo. Para obter mais informações, consulte a Operação de exportação de ATR Da Vinci.

Tipos de recursos compatíveis

`Group`, `Patient`, `Coverage`, `RelatedPerson`, `Practitioner`, `PractitionerRole`, `Organization`, `Location`

Filtragem temporal

Nenhuma filtragem temporal é aplicada. Todos os recursos correspondentes são exportados independentemente da data.

PDex Tipos de exportação

Todos os tipos de PDex exportação compartilham os mesmos perfis suportados e a mesma lógica de filtragem. Para obter mais informações, consulte a API Da Vinci PDex Provider Access. Os seguintes perfis são compatíveis:

- US Core 3.1.1, 6.1.0 e 7.0.0
- PDex Autorização prévia (não suportada para acesso de membros)
- Perfis básicos do CARIN BB 2.x: internação institucional, ambulatorial institucional, profissional, oral, farmacêutica NonClinician

Para PDex exportações, os recursos clínicos e de reclamações são descobertos automaticamente para cada paciente do Grupo. Você não precisa referenciar explicitamente esses recursos no recurso do Grupo. A operação busca todos os recursos do compartimento do paciente (como`Observation`,,`Condition`, `Coverage` `RelatedPerson`,`MedicationRequest`, e`ExplanationOfBenefit`) que pertencem aos pacientes atribuídos. Somente os tipos `PatientGroup`, e non-patient-compartment ATR (`Practitioner`,, `PractitionerRoleOrganization`,`Location`) exigem referências explícitas no Grupo.

Acesso do provedor (`hl7.fhir.us.davinci-pdex`)

Permite que os provedores da rede recuperem dados de pacientes atribuídos.

Payer-to-Payer (`hl7.fhir.us.davinci-pdex.p2p`)

Permite a troca de dados entre pagadores quando um paciente muda de seguro.

Acesso de membro (`hl7.fhir.us.davinci-pdex.member`)

Permite que os membros acessem seus próprios dados de saúde. Esse tipo de exportação pode incluir dados financeiros nos recursos de sinistros.

Suporte de perfil e lógica de inclusão

Para PDex exportações, a `$davinci-data-export` operação usa declarações de perfil no `meta.profile` elemento para determinar quais recursos incluir na exportação.

ExplanationOfBenefit Tratamento de recursos

`ExplanationOfBenefit`Os recursos (EOB) são incluídos ou excluídos PDex das exportações com base em suas `meta.profile` declarações:

- ExplanationOfBenefit recursos com um perfil CARIN BB 1.x são excluídos da exportação.
- ExplanationOfBenefit recursos sem `meta.profile` conjunto são excluídos da exportação.
- ExplanationOfBenefit recursos com um perfil CARIN BB 2.x Basis estão sempre incluídos.
- ExplanationOfBenefit recursos com um perfil CARIN BB 2.x que contém dados financeiros são excluídos por padrão. Quando `_includeEOB2xWoFinancial=true` definido, eles são incluídos com os dados financeiros retirados e o recurso é transformado no perfil Basis correspondente.
- ExplanationOfBenefit recursos com um perfil de autorização PDex prévia estão sempre incluídos.

Transformação de dados financeiros

Quando você configura`_includeEOB2xWoFinancial=true`, a operação transforma os ExplanationOfBenefit recursos do CARIN BB 2.x em seus perfis Basis correspondentes, removendo dados financeiros. Por exemplo, um `C4BB ExplanationOfBenefit Oral` recurso é transformado em`C4BB ExplanationOfBenefit Oral Basis`, o que retira os dados financeiros do registro de acordo com a especificação FHIR.

Os seguintes elementos de dados financeiros são removidos durante a transformação:

- Todos os elementos são cortados `total`
- Todos os `adjudication` elementos com `amounttype` fatia
- Todos os `item.adjudication` elementos com informações de quantidade

A operação também atualiza os metadados do perfil durante a transformação:

- `meta.profile`é atualizado para o URL canônico do perfil básico
- A versão é atualizada para a versão base do CARIN BB 2.x
- Os recursos existentes no armazenamento de dados não são modificados
- Os recursos exportados não são mantidos de volta ao armazenamento de dados

Regras de detecção de perfil

A operação usa as seguintes regras para detectar e validar perfis:

- A detecção de versão é baseada no código `meta.profile` canônico URLs
- Um recurso é incluído se QUALQUER um de seus perfis declarados corresponder aos critérios de exportação.
- A validação do perfil ocorre durante o processamento da exportação

Filtragem temporal de cinco anos para exportações PDex

Para todos os tipos de PDex exportação, HealthLake aplica um filtro temporal de 5 anos com base na última atualização do recurso. O filtro temporal se aplica a todos os recursos, exceto aos seguintes tipos principais de recursos de atribuição, que são sempre exportados independentemente da idade:

- `Patient`
- `Coverage`
- `Organization`
- `Practitioner`
- `PractitionerRole`
- `RelatedPerson`
- `Location`

- `Group`

Esses recursos administrativos e demográficos são isentos porque fornecem contexto essencial para os dados exportados. As exportações de ATR não estão sujeitas a nenhuma filtragem temporal.

Solicitações de amostra

Os exemplos a seguir mostram como iniciar trabalhos de exportação para diferentes tipos de exportação.

Exportação 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"
    }
  }
}
```

Exportação do Provider Access com remoção de dados ExplanationOfBenefit financeiros

```
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 exportar

```
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
```

Exportação de acesso de membro para um paciente específico

```
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
```

Resposta da amostra

```
{
  "datastoreId": "eaee622d8406b41eb86c0f4741201ff9",
  "jobStatus": "SUBMITTED",
  "jobId": "48d7b91dae4a64d00d54b70862f33f61"
}
```

Relacionamentos de recursos

A operação exporta recursos com base em seus relacionamentos na Lista de atribuição de membros:

```
Group (Attribution List)
### Patient (Members)
### Coverage # RelatedPerson (Subscribers)
### Practitioner (Attributed Providers)
### PractitionerRole # Location
### Organization (Attributed Providers)
```

> **Note**
>
> O diagrama de relacionamento de recursos anterior se aplica às exportações de ATR. Para PDex exportações, os recursos clínicos e de reclamações são descobertos automaticamente por meio da pesquisa de pacientes e não exigem referências explícitas no recurso do Grupo.

Fontes de recursos

Recurso	Localização da fonte	Description
`Patient`	`Group.member.entity`	Os pacientes que são membros da lista de atribuição
`Coverage`	`Group.member.extension:coverageReference`	Cobertura que resultou na adesão do paciente
`Organization`	`Group.member.extension:attributedProvider`	Organizações às quais os pacientes são atribuídos
`Practitioner`	`Group.member.extension:attributedProvider`	Profissionais individuais aos quais os pacientes são atribuídos
`PractitionerRole`	`Group.member.extension:attributedProvider`	Funções profissionais às quais os pacientes são atribuídos
`RelatedPerson`	`Coverage.subscriber`	Assinantes da cobertura
`Location`	`PractitionerRole.location`	Locais associados às funções dos profissionais
`Group`	Ponto final de entrada	A lista de atribuições em si

Gestão de Job

Verifique o status do trabalho

```
GET [base]/export/[job-id]
```

Cancelar trabalho

```
DELETE [base]/export/[job-id]
```

Ciclo de vida da tarefa

- `SUBMITTED`- Job foi recebido e colocado na fila

- `IN_PROGRESS`- O trabalho está sendo processado ativamente
- `COMPLETED`- Job concluído com sucesso, arquivos disponíveis para download
- `FAILED`- Job encontrou um erro

Output Format

- Formato de arquivo: NDJSON (JSON delimitado por nova linha)
- Organização de arquivos: arquivos separados para cada tipo de recurso
- Extensão do arquivo: .ndjson
- Localização: bucket e caminho do S3 especificados

Tratamento de erros

A operação retorna HTTP 400 Bad Request com uma OperationOutcome das seguintes condições:

Erros de autorização

A função do IAM especificada em `DataAccessRoleArn` não tem permissões suficientes para realizar a operação de exportação. Para obter a lista completa das permissões necessárias do S3 e do KMS, consulte Configuração de permissões para trabalhos de exportação.

Erros de validação de parâmetros

- O `patient` parâmetro não está formatado como `Patient/id,Patient/id,...`
- Uma ou mais referências de pacientes são inválidas ou não pertencem ao Grupo especificado
- O valor do `exportType` parâmetro não é um tipo de exportação compatível
- O `_type` parâmetro contém tipos de recursos que não são compatíveis com o tipo de exportação especificado.
- O `_type` parâmetro não contém os tipos de recursos necessários (`Group`,`Patient`,`Coverage`) para o tipo de `hl7.fhir.us.davinci-atr` exportação
- O valor do `_includeE0B2xWoFinancial` parâmetro não é um booleano válido

Erros de validação de recursos

- O recurso de grupo especificado não existe no armazenamento de dados
- O recurso de grupo especificado não tem membros
- Um ou mais membros do Grupo fazem referência a recursos do paciente que não existem no armazenamento de dados

Segurança e autorização

- Os mecanismos de autorização padrão do FHIR se aplicam
- A função de acesso a dados deve ter as permissões necessárias do IAM para operações do S3 e do KMS. Para ver a lista completa das permissões necessárias, consulte Configuração de permissões para trabalhos de exportação.

Práticas recomendadas

- Seleção do tipo de recurso: solicite somente os tipos de recursos necessários para minimizar o tamanho da exportação e o tempo de processamento
- Filtragem baseada em tempo: use o `_since` parâmetro para exportações incrementais
- Filtragem de pacientes: use o `patient` parâmetro quando precisar apenas de dados de membros específicos
- Monitoramento de trabalhos: verifique regularmente o status do trabalho para grandes exportações
- Tratamento de erros: implemente a lógica de repetição adequada para trabalhos com falha
- Reconhecimento do filtro temporal: para PDex exportações, considere o filtro temporal de 5 anos ao selecionar os tipos de recursos
- Remoção de dados financeiros: use `_includeEOB2xWoFinancial=true` quando precisar de dados de sinistros sem informações financeiras
- Gerenciamento de perfil: garanta que os recursos tenham declarações de perfil apropriadas, valide os perfis de destino antes da ingestão e use o controle de versão do perfil para controlar o comportamento de exportação

Limitações

- Máximo de 500 pacientes pode ser especificado no `patient` parâmetro
- A exportação é limitada somente às operações em nível de grupo
- Suporta apenas o conjunto predefinido de tipos de recursos para cada tipo de exportação
- A saída está sempre no formato NDJSON
- PDex as exportações são limitadas a 5 anos de dados clínicos e de sinistros
- A transformação de dados financeiros só se aplica aos perfis CARIN BB 2.x ExplanationOfBenefit

Recursos adicionais

- Lista de atribuição de membros da Da Vinci (IG)
- Intercâmbio de Dados do Jogador Da Vinci
- CARIN Consumer Directed Player Data Exchange IG
- Guia de implementação principal dos EUA
- Especificação de acesso a dados em massa do FHIR

Gerando documentos clínicos com **`$document`**

AWS HealthLake agora suporta a `$document` operação de recursos de composição, permitindo gerar um documento clínico completo agrupando a composição com todos os recursos referenciados em um único pacote coeso. Essa operação é essencial para aplicativos de saúde que precisam:

- Crie documentos clínicos padronizados
- Troque registros completos de pacientes
- Armazene documentação clínica abrangente
- Gere relatórios que incluam todo o contexto relevante

Usage

A `$document` operação pode ser invocada em recursos de composição usando os métodos GET e POST:

Operações com Suporte

```
GET/POST [base]/Composition/[id]/$document
```

Parâmetros compatíveis

HealthLake suporta o seguinte parâmetro FHIR: `$document`

Parâmetro	Tipo	Obrigatório	Padrão	Description
`persist`	booliano	Não	false	Booleano indicando se o servidor deve armazenar o pacote de documentos gerado

Exemplos

Solicitação GET

```
GET [base]/Composition/180f219f-97a8-486d-99d9-ed631fe4fc57/$document?persist=true
```

Solicitação POST com parâmetros

```
POST [base]/Composition/180f219f-97a8-486d-99d9-ed631fe4fc57/$document
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "persist",
      "valueBoolean": true
    }
  ]
}
```

Resposta da amostra

A operação retorna um recurso Bundle do tipo “documento” contendo a composição e todos os recursos referenciados:

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

Comportamento

A `$document` operação:

1. Usa o recurso de composição especificado como base para o documento
2. Identifica e recupera todos os recursos diretamente referenciados pela Composição
3. Empacota a composição e todos os recursos referenciados em um pacote do tipo “documento”
4. Armazena o pacote de documentos gerado no armazenamento de dados quando o parâmetro persist é definido como verdadeiro
5. Identifica e recupera recursos referenciados indiretamente pela Composição para geração abrangente de documentos

Atualmente, a `$document` operação oferece suporte à recuperação de referências de recursos no seguinte formato:

1. ```
   GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Resource/id
   ```
2. Recurso/ID

As referências de recursos não suportadas no recurso de composição serão filtradas do documento gerado.

Tratamento de erros

A operação trata das seguintes condições de erro:

- 400 Solicitação inválida: `$document` operação inválida (solicitação não conforme) ou se o documento resultante falhar na validação do FHIR devido a referências filtradas quando persistir é definido como verdadeiro
- 404 Não encontrado: recurso de composição não encontrado

Para obter mais informações sobre a especificação da `$document` operação, consulte a documentação da composição do FHIR R4. `$document`

Removendo recursos permanentemente com **`$erase`**

AWS HealthLake suporta a `$erase` operação, permitindo a exclusão permanente de um recurso específico e de suas versões históricas. Essa operação é particularmente útil quando você precisa:

- Remova permanentemente recursos individuais
- Excluir históricos de versões específicos
- Gerencie ciclos de vida de recursos individuais
- Cumpra os requisitos específicos de remoção de dados

Usage

A `$erase` operação pode ser invocada em dois níveis:

Nível de instância do recurso

```
POST [base]/[ResourceType]/[ID]/$erase?deleteAuditEvent=true
```
```

Nível específico da versão

```
POST [base]/[ResourceType]/[ID]/_history/[VersionID]/$erase
```

Parâmetros

Parâmetro	Tipo	Obrigatório	Padrão	Description
`deleteAuditEvent`	booliano	Não	false	Quando verdadeiro, exclui os eventos de auditoria associados

Exemplos

Exemplo de solicitação

```
POST [base]/Patient/example-patient/$erase
```

Exemplo de resposta

```
{
      "jobId": "5df47e2f51ff3c731847678cb8cad48e",
      "jobStatus": "SUBMITTED"
    }
```

Status do trabalho

Para verificar o status de uma tarefa de exclusão:

```
GET [base]/$erase/[jobId]
```

A operação retorna informações sobre o status do trabalho:

```
{
      "datastoreId": "36622996b1fcecb7e12ee2ee085308d3",
      "jobId": "5df47e2f51ff3c731847678cb8cad48e",
      "status": "COMPLETED",
```

```
    "submittedTime": "2025-10-30T16:39:24.160Z"
  }
```

Comportamento

A `$erase` operação:

1. Processa de forma assíncrona para garantir a integridade dos dados
2. Mantém transações ACID
3. Fornece rastreamento do status do trabalho
4. Remove permanentemente o recurso especificado e suas versões
5. Inclui registro abrangente de auditoria das atividades de exclusão
6. Oferece suporte à exclusão seletiva de eventos de auditoria

Registro de auditoria

A `$erase` operação é registrada DeleteResource com o ID do usuário, o carimbo de data/hora e os detalhes do recurso.

Limitações

- `$erase`do recurso não aparecerá nos resultados da pesquisa ou `_history` nas consultas.
- Os recursos que estão sendo apagados podem ficar temporariamente inacessíveis durante o processamento
- A medição de armazenamento é ajustada imediatamente à medida que os recursos são removidos permanentemente

Obtendo dados do paciente com **`Patient/$everything`**

A `Patient/$everything` operação é usada para consultar um `Patient` recurso FHIR, junto com quaisquer outros recursos relacionados a ele. `Patient` A operação pode ser usada para fornecer ao paciente acesso a todo o prontuário ou para que um provedor realize um download em massa de dados relacionados a um paciente. HealthLakesuportes `Patient/$everything` para um paciente específico`id`.

`Patient/$everything`é uma operação da API FHIR REST que pode ser invocada conforme mostrado nos exemplos abaixo.

GET request

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything
```

> **Note**
>
> Os recursos em resposta são classificados por tipo de recurso e recursoid.
> A resposta é sempre preenchida comBundle.total.

Parâmetros do **`Patient/$everything`**

HealthLake suporta os seguintes parâmetros de consulta

Parâmetro	Detalhes
rápido	Obtenha todos os `Patient` dados após uma data de início especificada.
end	Obtenha todos os `Patient` dados antes de uma data de término especificada.
since	Atualize todos os `Patient` dados após uma data especificada.
_tipo	Obtenha `Patient` dados para tipos de recursos específicos.
_contar	Obtenha `Patient` dados e especifique o tamanho da página.

Example- Obtenha todos os dados do paciente após uma data de início especificada

`Patient/$everything`pode usar o `start` filtro para consultar somente dados após uma data específica.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?start=2024-03-15T00:00:00.000Z
```

Example- Obtenha todos os Patient dados antes de uma data de término especificada

O paciente $everything pode usar o end filtro para consultar apenas dados antes de uma data específica.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?end=2024-03-15T00:00:00.000Z
```

Example- Obtenha todos os Patient dados atualizados após uma data especificada

`Patient/$everything`pode usar o `since` filtro para consultar somente dados atualizados após uma data específica.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?since=2024-03-15T00:00:00.000Z
```

Example- Obtenha Patient dados para tipos de recursos específicos

O paciente $everything pode usar o `_type` filtro para especificar tipos de recursos específicos a serem incluídos na resposta. Vários tipos de recursos podem ser especificados em uma lista separada por vírgulas.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?_type=Observation,Condition
```

Example- Obtenha Patient dados e especifique o tamanho da página

O paciente $everything pode usar o `_count` para definir o tamanho da página.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?_count=15
```

`Patient/$everything``start`e `end` atributos

HealthLake oferece suporte aos seguintes atributos de recursos para `Patient/ $everything` `start` os parâmetros de `end` consulta e.

Recurso	Elemento de recursos
Conta	conta.período de serviço. Início
AdverseEvent	AdverseEvent.data

Recurso	Elemento de recursos
AllergyIntolerance	AllergyIntolerance.Data de gravação
Nomeação	Marcação. Início
AppointmentResponse	AppointmentResponse.iniciar
AuditEvent	AuditEvent.periodo.start
Básico	Básico. Criado
BodyStructure	SEM DATA
CarePlan	CarePlan.periodo.start
CareTeam	CareTeam.periodo.start
ChargeItem	ChargeItem. occurrenceDateTime, ChargeItem .OccurrencePeriod.start, .OccurrenceTiming.Event ChargeItem
Reivindicar	Reivindicação. Período faturável. Início
ClaimResponse	ClaimResponse.criado
ClinicalImpression	ClinicalImpression.data
Comunicação	Comunicação. Enviada
CommunicationRequest	CommunicationRequest. occurrenceDateTime, .Período de ocorrência.Início

Recurso	Elemento de recursos
Composição	Composição.Data
Condição	Condição.Data de gravação
Consentimento	Consentimento.Data/hora
Cobertura	Cobertura.Período.Início
CoverageEligibilityRequest	CoverageEligibilityRequest.criado
CoverageEligibilityResponse	CoverageEligibilityResponse.criado
DetectedIssue	DetectedIssue.identificado
DeviceRequest	DeviceRequest.Autor Edon
DeviceUseStatement	DeviceUseStatement. Gravado em
DiagnosticReport	DiagnosticReport.eficaz
DocumentManifest	DocumentManifest.criado
DocumentReference	DocumentReference.contexto.period.start
Encontro	Encounter.period.start

Recurso	Elemento de recursos
EnrollmentRequest	EnrollmentRequest.criado
EpisodeOfCare	EpisodeOfCare.periodo.start
ExplanationOfBenefit	ExplanationOfBenefit.Período faturável. Início
FamilyMemberHistory	SEM DATA
Sinalizador	FLAG.PERIOD.START
Objetivo	Objetivo.StatusData
Group (Grupo)	SEM DATA
ImagingStudy	ImagingStudy.iniciado
Imunização	Imunização. Registrada
ImmunizationEvaluation	ImmunizationEvaluation.data
ImmunizationRecommendation	ImmunizationRecommendation.data
Fatura	Data da fatura
Lista	Data da lista

Recurso	Elemento de recursos
MeasureReport	MeasureReport.periodo.start
Mídia	Mídia. Emitido
MedicationAdministration	MedicationAdministration.eficaz
MedicationDispense	MedicationDispense. Quando preparado
MedicationRequest	MedicationRequest.Autor Edon
MedicationStatement	MedicationStatement.Data declarada
MolecularSequence	SEM DATA
NutritionOrder	NutritionOrder.Data/hora
Observação	Observação. Eficaz
Paciente	SEM DATA
Pessoa	SEM DATA
Procedimento	Procedimento. Executado
Proveniência	Proveniência. Período ocorrido. Início, Proveniência. occurredDateTime

Recurso	Elemento de recursos
QuestionnaireResponse	QuestionnaireResponse.de autoria
RelatedPerson	SEM DATA
RequestGroup	RequestGroup.Autor Edon
ResearchSubject	ResearchSubject.período
RiskAssessment	RiskAssessment. occurrenceDateTime, .Período de RiskAssessment ocorrência.Início
Agendamento	Cronograma. Horizonte de planejamento
ServiceRequest	ServiceRequest.Autor Edon
Espécime	Espécime. Hora recebida
SupplyDelivery	SupplyDelivery. occurrenceDateTime, SupplyDelivery .OccurrencePeriod. start, .OccurrenceTiming.Event SupplyDelivery
SupplyRequest	SupplyRequest.Autor Edon
VisionPrescription	VisionPrescription.Data escrita

Recuperando ValueSet códigos com **$expand**

AWS HealthLake agora suporta a `$expand` operação ValueSets que foi ingerida por você como cliente, permitindo que você recupere a lista completa de códigos contidos nesses ValueSet recursos. Essa operação é particularmente útil quando você precisa:

- Recupere todos os códigos possíveis para fins de validação
- Exibir as opções disponíveis nas interfaces de usuário
- Realize pesquisas abrangentes de código dentro de um contexto terminológico específico

Usage

A `$expand` operação pode ser invocada em ValueSet recursos usando os métodos GET e POST:

Operações com Suporte

```
GET/POST [base]/ValueSet/[id]/$expand
GET [base]/ValueSet/$expand?url=http://example.com
POST [base]/ValueSet/$expand
```

Parâmetros compatíveis

HealthLake suporta um subconjunto de parâmetros FHIR R4: `$expand`

Parâmetro	Tipo	Obrigatório	Description
`url`	uri	Não	URL canônico do para expandir ValueSet
`id`	id	Não	ValueSet ID do recurso a ser expandido (para operações GET ou POST)
`filter`	string	Não	Filtrar o resultado da expansão do código
`count`	integer	Não	Número de códigos a serem devolvidos
`offset`	integer	Não	Número de códigos correspondentes a serem ignorados antes de devolver. Aplica-se após a filtragem e somente aos códigos correspon

Parâmetro	Tipo	Obrigatório	Description
			dentes, não ao conteúdo completo e não filtrado do original ValueSet

Exemplos

Solicitação GET por ID

```
GET [base]/ValueSet/example-valueset/$expand
```

Solicitação GET por URL com filtro

```
GET [base]/ValueSet/$expand?url=http://example.com/ValueSet/my-
valueset&filter=male&count=5
```

Solicitação POST com parâmetros (por ID)

```
POST [base]/ValueSet/example-valueset/$expand
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "count",
      "valueInteger": 10
    },
    {
      "name": "filter",
      "valueString": "admin"
    }
  ]
}
```

Solicitação POST com parâmetros (por 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
    }
  ]
}
```

Resposta da amostra

A operação retorna um ValueSet recurso com um `expansion` elemento contendo os códigos expandidos:

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

A resposta inclui:

- expansion.total: número total de códigos no expandido ValueSet
- expansion.contains: matriz de códigos expandidos com seus valores de sistema, código e exibição
- expansion.parameter: parâmetros usados na solicitação de expansão

Para obter mais informações sobre a especificação da `$expand` operação, consulte a documentação do FHIR ValueSet `$expand` R4.

Exportação de HealthLake dados com o FHIR **`$export`**

Você pode exportar dados em massa do seu armazenamento de HealthLake dados usando a operação FHIR $export. HealthLake suporta o `$export` uso `POST` e as solicitações do FHIR. `GET` Para fazer uma solicitação de exportação com`POST`, você precisa ter um usuário, grupo ou função do IAM com as permissões necessárias, especificar `$export` como parte da solicitação e incluir os parâmetros desejados no corpo da solicitação.

> **Note**
>
> Todas as solicitações de HealthLake exportação feitas usando o FHIR `$export` são retornadas em `ndjson` formato e exportadas para um bucket do Amazon S3, onde cada objeto do Amazon S3 contém somente um único tipo de recurso do FHIR.

Você pode enfileirar solicitações de exportação de acordo com as AWS cotas de serviço da conta. Para obter mais informações, consulte Cotas de serviço.

HealthLake suporta os três tipos de solicitações de endpoint de exportação em massa a seguir.

HealthLake **`$export`**tipos a granel

Tipo de exportação	Description	Sintaxe
Sistema	Exporte todos os dados do servidor HealthLake FHIR.	`POST https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/$export`

Todos os pacientes	Exporte todos os dados relacionados a todos os pacientes, incluindo os tipos de recursos associados ao tipo de recurso do paciente.	`POST https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/Patient/$export`
		
		`GET https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/Patient/$export`

Grupo de pacientes	Exporte todos os dados relacionados a um grupo de pacientes especificado com uma ID de grupo.	`POST https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/Group/ id/$export`
		
		`GET https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/Group/ id/$export`

Antes de começar

Atenda aos requisitos a seguir para fazer uma solicitação de exportação usando a API FHIR REST para. HealthLake

- Você deve ter configurado um usuário, grupo ou função que tenha as permissões necessárias para fazer a solicitação de exportação. Para saber mais, consulte Autorizando uma solicitação `$export`.
- Você deve ter criado uma função de serviço que conceda HealthLake acesso ao bucket do Amazon S3 para o qual você deseja que seus dados sejam exportados. A função de serviço também deve ser especificada HealthLake como principal do serviço. Para obter mais informações sobre a configuração de permissões, consulteConfigurando permissões para trabalhos de exportação.

Autorizando uma solicitação **`$export`**

Para fazer uma solicitação de exportação bem-sucedida usando a API REST FHIR, autorize seu usuário, grupo ou função usando IAM ou .0. OAuth2 Você também deve ter uma função de serviço.

Autorizar uma solicitação usando o IAM

Quando você faz uma `$export` solicitação, o usuário, o grupo ou a função devem ter ações do IAM incluídas na política. Para obter mais informações, consulte Configurando permissões para trabalhos de exportação.

Autorizando uma solicitação usando SMART no FHIR (2.0) OAuth

Ao fazer uma `$export` solicitação em um SMART no armazenamento de HealthLake dados habilitado para FHIR, você deve ter os escopos apropriados atribuídos. Para obter mais informações, consulte SMART nos escopos de recursos do FHIR para HealthLake.

> **Note**
>
> O FHIR `$export` com `GET` solicitações exige o mesmo método de autenticação ou token do portador (no caso do SMART no FHIR) para solicitar a exportação e a recuperação de arquivos. Os arquivos exportados usando o FHIR `$export` com `GET` estão disponíveis para download por 48 horas.

Fazendo uma **`$export`** solicitação

Esta seção descreve as etapas necessárias que você deve seguir ao fazer uma solicitação de exportação usando a API FHIR REST.

Para evitar cobranças acidentais em sua AWS conta, recomendamos testar suas solicitações fazendo uma `POST` solicitação sem fornecer a `$export` sintaxe.

Para fazer a solicitação, você deve fazer o seguinte:

1. Especifique `$export` no URL da `POST` solicitação para um endpoint compatível.
2. Especifique os parâmetros de cabeçalho necessários.
3. Especifique um corpo de solicitação que defina os parâmetros necessários.

Etapa 1: especifique **`$export`** no URL da **`POST`** solicitação para um endpoint compatível.

HealthLake oferece suporte a três tipos de solicitações de endpoint de exportação em massa. Para fazer uma solicitação de exportação em massa, você deve fazer uma solicitação `POST` com base em um dos três endpoints compatíveis. Os exemplos a seguir demonstram onde especificar `$export` no URL da solicitação.

- `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`

Você pode usar os seguintes parâmetros de pesquisa compatíveis na string de `POST` solicitação.

Parâmetros de pesquisa compatíveis

HealthLake suporta os seguintes modificadores de pesquisa em solicitações de exportação em massa.

Os exemplos a seguir incluem caracteres especiais que devem ser codificados antes de enviar sua solicitação.

Nome	Obrigatório?	Description	Exemplo
`_outputFormat`	Não	O formato dos arquivos de dados em massa solicitad	

Nome	Obrigatório?	Description	Exemplo
		os a serem gerados. Os valores aceitos sãoapplicati on/fhir+n djson ,applicati on/ndjson ,ndjson.	
_type	Não	Uma sequência de tipos de recursos FHIR delimitados por vírgula que você deseja incluir em seu trabalho de exportaçã o. Recomenda mos incluir _type porque isso pode ter uma implicaçã o de custo quando todos os recursos são exportados.	&_type=Me dicationS tatement, Observation
_since	Não	Tipos de recursos modificados em ou após o carimbo de data e hora. Se um tipo de recurso não tiver um horário de última atualização, ele será incluído na sua resposta.	&_since=2 024-05-09 T00%3A00%3A00Z

Nome	Obrigatório?	Description	Exemplo
`_until`	Não	Tipos de recursos modificados em ou antes do carimbo de data e hora. Usado em combinação com `_since` para definir um intervalo de tempo específico para exportação. Se um tipo de recurso não tiver um horário de última atualização, ele será excluído da sua resposta.	`&_until=2024-12-31T23%3A59%3A59Z`
`_security`	Não	Filtre os recursos exportados por valores de `meta.security` codificação. Use o `system|code` formato. Quando vários valores são fornecidos, os recursos devem corresponder a todos eles (E à semântica). Use `system|` (tubo de arrasto, sem código) para combinar qualquer código de um determinado sistema.	`&_security=https://myorg.com/tenant%7Cclinic-A`

Nome	Obrigatório?	Description	Exemplo
`_tag`	Não	Filtre os recursos exportados por valores de `meta.tag` codificação. Usa o mesmo `system|code` formato e semântica AND de `_security`. Quando ambos `_security` `_tag` são especificados, os recursos devem corresponder aos dois filtros.	`&_tag=https://myorg.com/dept%7Ccardiology`

Etapa 2: especificar os parâmetros de cabeçalho necessários

Para fazer uma solicitação de exportação usando a API REST FHIR, você deve especificar os seguintes parâmetros de cabeçalho.

- Content-Type: `application/fhir+json`
- Prefiro: `respond-async`

Em seguida, você deve especificar os elementos necessários no corpo da solicitação.

Etapa 3: especifique um corpo de solicitação que defina os parâmetros necessários.

A solicitação de exportação também exige um corpo em `JSON` formato. O corpo pode incluir os seguintes parâmetros.

Chave	Obrigatório?	Description	Valor
DataAccessRoleArn	Sim	Um ARN de uma função de HealthLake serviço. A função	`arn:aws:iam:: 444455556`

Chave	Obrigatório?	Description	Valor
		de serviço usada deve ser especific ada HealthLake como principal do serviço.	**666** :role/**your-healthlake-service-role**
JobName	Não	O nome da solicitação de exportação.	**your-export-job-name**
S3Uri	Sim	Parte de uma OutputDataConfig chave. O URI do S3 do bucket de destino em que seus dados exportados serão baixados.	s3://amzn-s3-demo-bucket/ **EXPORT-JOB** /
KmsKeyId	Sim	Parte de uma OutputDataConfig chave. O ARN da AWS KMS chave usada para proteger o bucket do Amazon S3.	arn:aws:kms: **region-of-bucket:123456789012** :key/**1234abcd-12ab-34cd-56ef-1234567890ab**

Example Corpo de uma solicitação de exportação feita usando a API FHIR REST

Para fazer uma solicitação de exportação usando a API REST FHIR, você deve especificar um corpo, conforme mostrado a seguir.

```
{
  "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"
```

```
    }
  }
}
```

Quando sua solicitação for bem-sucedida, você receberá a seguinte resposta.

cabeçalho de resposta

```
content-location: https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
export/your-export-request-job-id
```

Corpo de resposta

```
{
  "datastoreId": "your-data-store-id",
  "jobStatus": "SUBMITTED",
  "jobId": "your-export-request-job-id"
}
```

Gerenciando sua solicitação de exportação

Depois de fazer uma solicitação de exportação bem-sucedida, você pode gerenciar a solicitação usando `$export` para descrever o status de uma solicitação de exportação atual e `$export` cancelar uma solicitação de exportação atual.

Ao cancelar uma solicitação de exportação usando a API REST, você só é cobrado pela parte dos dados que foram exportados até o momento em que você enviou a solicitação de cancelamento.

Os tópicos a seguir descrevem como você pode obter o status de uma solicitação de exportação atual ou cancelá-la.

Cancelamento de uma solicitação de exportação

Para cancelar uma solicitação de exportação, faça uma `DELETE` solicitação e forneça o ID do trabalho na URL da solicitação.

```
DELETE https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/export/your-
export-request-job-id
```

Quando sua solicitação for bem-sucedida, você receberá o seguinte.

```
{
```

```
  "exportJobProperties": {
    "jobId": "your-original-export-request-job-id",
    "jobStatus": "CANCEL_SUBMITTED",
    "datastoreId": "your-data-store-id"
  }
}
```

Quando sua solicitação não for bem-sucedida, você receberá o seguinte.

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "not-supported",
      "diagnostics": "Interaction not supported."
    }
  ]
}
```

Descrever uma solicitação de exportação

Para obter o status de uma solicitação de exportação, faça uma `GET` solicitação usando `export` e seu**`export-request-job-id`**.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/export/your-export-request-id
```

A resposta JSON conterá um `ExportJobProperties` objeto. Ele pode conter os seguintes pares chave-valor.

Nome	Obrigatório?	Description	Valor
DataAccessRoleArn	Não	Um ARN de uma função de HealthLake serviço. A função de serviço usada deve ser especificada HealthLake como principal do serviço.	`arn:aws:iam:: 444455556666 :role/your-healthlake-service-role`

Nome	Obrigatório?	Description	Valor
SubmitTime	Não	A data e hora em que um trabalho de exportação foi enviado.	`Apr 21, 2023 5:58:02`
EndTime	Não	A hora em que um trabalho de exportação foi concluído.	`Apr 21, 2023 6:00:08 PM`
JobName	Não	O nome da solicitação de exportação.	**`your-export-job-name`**
JobStatus	Não		Os valores válidos são: `SUBMITTED | IN_PROGRESS | COMPLETED_WITH_ERRORS | COMPLETED | FAILED`
S3Uri	Sim	Parte de um OutputDataConfigobjeto. O URI do Amazon S3 do bucket de destino em que seus dados exportados serão baixados.	`s3://amzn-s3-demo-bucket/` **`EXPORT-JOB`** `/`

Nome	Obrigatório?	Description	Valor
KmsKeyId	Sim	Parte de um OutputDataConfigobjeto. O ARN da AWS KMS chave usada para proteger o bucket do Amazon S3.	`arn:aws:kms: region-of-bucket:123456789012 :key/1234abcd-12ab-34cd-56ef-1234567890ab`

Example: Corpo de uma solicitação de descrição de exportação feita usando a API REST FHIR

Quando for bem-sucedido, você receberá a seguinte resposta JSON.

```
{
  "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",
  }
}
```

Operação FHIR para **`$inquire`** HealthLake

A `$inquire` operação permite que você verifique o status de uma solicitação de autorização prévia enviada anteriormente. Essa operação implementa o Guia de Implementação do Da Vinci Prior Authorization Support (PAS), fornecendo um fluxo de trabalho padronizado baseado em FHIR para recuperar a decisão de autorização atual.

Como funciona

- Enviar consulta: Você envia um pacote FHIR contendo a reclamação que deseja verificar e informações de apoio
- Pesquisa: HealthLake pesquisa o correspondente ClaimResponse em seu armazenamento de dados
- Recuperar: o status de autorização mais recente foi recuperado
- Responder: você recebe uma resposta imediata com o status atual da autorização (em fila, aprovada, negada etc.)

> **Note**
>
> `$inquire`é uma operação somente de leitura que recupera o status de autorização existente. Ele não modifica nem atualiza nenhum recurso em seu armazenamento de dados.

Ponto final da API

```
POST /datastore/{datastoreId}/r4/Claim/$inquire
Content-Type: application/fhir+json
```

Estrutura de solicitações

Requisitos do pacote

Sua solicitação deve ser um recurso do FHIR Bundle com:

- Bundle.type: Deve ser `"collection"`
- Bundle.entry: deve conter exatamente um recurso de reivindicação com:
  - `use = "preauthorization"`
  - `status = "active"`
- Recursos referenciados: todos os recursos referenciados pela reivindicação devem ser incluídos no pacote

Consulta por exemplo

Os recursos em seu pacote de entrada servem como um modelo de pesquisa. HealthLake usa as informações fornecidas para localizar o correspondente ClaimResponse.

Recursos necessários

Recurso	Cardinalidade	Perfil	Description
Reclamação	1	Consulta de reclamação do PAS	A autorização prévia sobre a qual você está perguntando
Paciente	1	Paciente beneficiário do PAS	Informações demográficas do paciente
Organização (Seguradora)	1	Organização segurador a PAS	Companhia de seguros
Organização (Provedor)	1	Organização solicitante do PAS	Profissional de saúde que enviou a solicitação

Critérios de pesquisa importantes

HealthLake pesquisa o ClaimResponse uso de:

- Referência do paciente a partir da reclamação
- Referência da seguradora a partir da reclamação
- Referência do provedor a partir da reclamação
- Data de criação a partir da reclamação (como filtro de tempo)

Somente consultas específicas do paciente

Todas as consultas devem estar vinculadas a um paciente específico. Consultas em todo o sistema sem identificação do paciente não são permitidas.

Exemplo de solicitação

```
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"
      }
```

```
    }
  ]
}
```

Formato de resposta

Resposta de sucesso (200 OK)

Você receberá um pacote de respostas de consulta do PAS contendo:

- ClaimResponsecom o status de autorização atual; múltiplo ClaimResponsese corresponder aos critérios de pesquisa
- Todos os recursos originais de sua solicitação (repetidos de volta)
- Registro de data e hora de quando a resposta foi montada

Possíveis ClaimResponse resultados

Outcome	Description
`queued`	A solicitação de autorização ainda está pendente de análise
`complete`	A decisão de autorização foi tomada (verifique se foi `disposition` aprovada/negada)
`error`	Ocorreu um erro durante o processamento
`partial`	Autorização parcial concedida

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

Respostas de erro

400 solicitação inválida

Retornado quando o formato da solicitação é inválido ou a validação falha.

```
{
    "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 Não autorizado

Retornado quando as credenciais de autenticação estão ausentes ou são inválidas.

```
{
    "resourceType": "OperationOutcome",
    "issue": [
        {
            "severity": "error",
            "code": "forbidden",
            "diagnostics": "Invalid authorization header"
```

```
        }
    ]
}
```

403 proibido

Retornado quando o usuário autenticado não tem permissão para acessar o recurso solicitado.

```
{
    "resourceType": "OperationOutcome",
    "issue": [
        {
            "severity": "error",
            "code": "exception",
            "diagnostics": "Insufficient SMART scope permissions."
        }
    ]
}
```

400 Quando nenhum foi encontrado

Retornado quando nenhuma correspondência ClaimResponse foi encontrada para a consulta.

```
{
  "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 Tipo de mídia não suportado

Retornado quando o cabeçalho Content-Type não é application/fhir+json.

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

```
    "code": "value",
    "diagnostics": "Incorrect MIME-type. Update request Content-Type header."
  }]
}
```

429, muitas solicitações

Retornado quando os limites de taxa são excedidos.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "throttled",
    "diagnostics": "Rate limit exceeded. Please retry after some time."
  }]
}
```

Regras de validação

HealthLake realiza uma validação abrangente em sua consulta:

Validação do pacote

- Deve estar em conformidade com o perfil do PAS Inquiry Request Bundle
- `Bundle.type`deve ser `"collection"`
- Deve conter exatamente um recurso de reclamação
- Todos os recursos referenciados devem ser incluídos no pacote

Validação da solicitação

- Deve estar em conformidade com o perfil PAS Claim Inquiry
- `Claim.use`deve ser `"preauthorization"`
- `Claim.status`deve ser `"active"`
- Campos obrigatórios:`patient`,`insurer`,`provider`, `created`

Validação de recursos

- Todos os recursos devem estar em conformidade com seus respectivos perfis de consulta do PAS

- Os recursos de apoio necessários devem estar presentes (paciente, seguradora, organização prestadora)
- As referências cruzadas devem ser válidas e solucionáveis dentro do pacote

Especificações de performance

Métrica	Especificação
Limite de contagem de recursos	500 recursos por pacote
Limite de tamanho do pacote	Máximo de 5 MB

Permissões obrigatórias

Para usar a `$inquire` operação, certifique-se de que sua função do IAM tenha:

- `healthlake:InquirePreAuthClaim`- Para chamar a operação

SMART em escopos FHIR

Escopos mínimos exigidos:

- SMART v1: `user/ClaimResponse.read`
- SMART v2: `user/ClaimResponse.s`

Notas de implementação importantes

Comportamento de pesquisa

Quando você envia uma consulta, HealthLake pesquisa ClaimResponse usando:

- Referência do paciente a partir da reivindicação de entrada
- Referência da seguradora a partir da reclamação de entrada
- Referência do provedor a partir da reivindicação de entrada
- Data de criação a partir da declaração de entrada (como um filtro de tempo)

Várias correspondências: se várias ClaimResponses corresponderem aos seus critérios de pesquisa, HealthLake retornará todos os resultados correspondentes. Você deve usar o `ClaimResponse.created` carimbo de data/hora mais recente para identificar o status mais recente.

Reivindicações atualizadas

Se você enviou várias atualizações para a mesma autorização anterior (por exemplo, Claim v1.1, v1.2, v1.3), a `$inquire` operação recuperará a versão ClaimResponse associada à versão mais recente com base nos critérios de pesquisa fornecidos.

Operação somente de leitura

A `$inquire` operação:

- Recupera o status de autorização existente
- Retorna o mais recente ClaimResponse
- Não modifica nem atualiza nenhum recurso
- Não cria novos recursos
- Não aciona o processamento de novas autorizações

Exemplo de fluxo de trabalho

Fluxo de trabalho típico de consulta de autorização prévia

```
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"
```

Operações do relacionadas

- `Claim/$submit`- Envie uma nova solicitação de autorização prévia ou atualize uma existente
- `Patient/$everything`- Recupere dados abrangentes do paciente para contexto de autorização prévia

Recuperando detalhes do conceito com **`$lookup`**

AWS HealthLake agora suporta a `$lookup` operação de CodeSystem recursos, permitindo que você recupere detalhes sobre um conceito específico em um sistema de código fornecendo informações de identificação, como seu código. Essa operação é particularmente útil quando você precisa:

- Recupere informações detalhadas sobre códigos médicos específicos
- Valide os significados e propriedades do código
- Acesse definições e relacionamentos de conceitos
- Support a tomada de decisões clínicas com dados terminológicos precisos

Usage

A `$lookup` operação pode ser invocada em CodeSystem recursos usando os métodos GET e POST:

Operações com Suporte

```
GET [base]/CodeSystem/$lookup?system=http://snomed.info/
sct&code=73211009&version=20230901
POST [base]/CodeSystem/$lookup
```

Parâmetros compatíveis

HealthLake suporta um subconjunto de parâmetros FHIR R4: `$lookup`

Parâmetro	Tipo	Obrigatório	Description
`code`	código	Sim	O código conceitual que você está procurando (por exemplo, “71620000" no SNOMED CT)

Parâmetro	Tipo	Obrigatório	Description
`system`	uri	Sim	O URL canônico do sistema de código (por exemplo, "http://snomed.info/sct “)
`version`	string	Não	Versão específica do sistema de código

Exemplos

Solicitação GET

```
GET [base]/CodeSystem/$lookup?system=http://snomed.info/
sct&code=71620000&version=2023-09
```

Solicitação 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"
    }
  ]
}
```

Resposta da amostra

A operação retorna um recurso de Parâmetros contendo os detalhes do conceito:

```
{
    "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"
```

```
                }
            ]
        }
    ]
}
```

Parâmetros de resposta

A resposta inclui os seguintes parâmetros, quando disponíveis:

Parâmetro	Tipo	Description
`name`	string	Nome do sistema de código
`version`	string	Versão do sistema de código
`display`	string	Nome de exibição do conceito
`designation`	BackboneElement	Representações adicionais para esse conceito.
`property`	BackboneElement	Propriedades adicionais do conceito (definição, relacionamentos, etc.)

Comportamento

A `$lookup` operação:

1. Valida os parâmetros necessários (`codeesystem`)
2. Pesquisa o conceito dentro do sistema de código especificado armazenado no armazenamento de dados
3. Retorna informações detalhadas do conceito, incluindo nome de exibição, designações e propriedades.
4. Suporta pesquisas específicas da versão quando o parâmetro é fornecido `version`
5. Opera somente em sistemas de código armazenados explicitamente no armazenamento de HealthLake dados

Tratamento de erros

A operação trata das seguintes condições de erro:

- 400 Solicitação inválida: `$lookup` operação inválida (solicitação não conforme ou ausência de parâmetros obrigatórios)
- 404 Não encontrado: sistema de código não encontrado ou código não encontrado no sistema de código especificado

Advertências

Para esta versão, não há suporte para o seguinte:

- `$lookup`operação chamando servidores de terminologia externos
- `$lookup`operação CodeSystems gerenciada por HealthLake , mas não armazenada explicitamente no armazenamento de dados

Para obter mais informações sobre a especificação da `$lookup` operação, consulte a documentação do FHIR CodeSystem `$lookup` R4.

`$member-add`operação para HealthLake

A `$member-add` operação FHIR adiciona um membro (paciente) a um recurso do Grupo, especificamente uma Lista de Atribuição de Membros. Essa operação faz parte do Guia de Implementação de Atribuição de DaVinci Membros e apoia o processo de reconciliação para gerenciar as atribuições de membros.

Ponto final da operação

```
POST [base]/datastore/{datastoreId}/r4/Group/{groupId}/$member-add
Content-Type: application/json
```

Parâmetros

A operação aceita um recurso de parâmetros FHIR com as seguintes combinações de parâmetros:

Opções de parâmetros

Você pode usar uma das seguintes combinações de parâmetros:

Opção 1: ID de membro + NPI do provedor

`memberId` + `providerNpi`

`memberId` + `providerNpi` + `attributionPeriod`

Opção 2: Referência do paciente + Referência do provedor

`patientReference` + `providerReference`

`patientReference` + `providerReference` + `attributionPeriod`

Detalhes do parâmetro

ID do membro (opcional)

O identificador do membro a ser adicionado ao Grupo.

Tipo: Identificador

Sistema: Sistema de identificação de pacientes

```
{
  "name": "memberId",
  "valueIdentifier": {
    "system": "http://example.org/patient-id",
    "value": "patient-new"
  }
}
```

ProviderNPI (opcional)

O Identificador Nacional do Provedor (NPI) do provedor atribuído.

Tipo: Identificador

Sistema: http://terminology.hl7. org/CodeSystem/NPI

```
{
  "name": "providerNpi",
  "valueIdentifier": {
    "system": "http://terminology.hl7.org/CodeSystem/NPI",
    "value": "1234567890"
```

```
  }
}
```

Referência do paciente (opcional)

Referência direta ao recurso do paciente a ser adicionado.

Tipo: Referência

```
{
  "name": "patientReference",
  "valueReference": {
    "reference": "Patient/patient-123"
  }
}
```

Referência do provedor (opcional)

Referência direta ao recurso do provedor.

Tipo: Referência

```
{
  "name": "providerReference",
  "valueReference": {
    "reference": "Practitioner/provider-456"
  }
}
```

Período de atribuição (opcional)

O período de tempo durante o qual o paciente é atribuído ao provedor.

Tipo: Período

```
{
  "name": "attributionPeriod",
  "valuePeriod": {
    "start": "2024-07-15",
    "end": "2025-07-14"
  }
}
```

Exemplos de solicitações

Usando ID de membro e NPI do provedor

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

Usando referências de pacientes e provedores

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

Formato de resposta

Resposta de adição bem-sucedida

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

Respostas de erro

Sintaxe de solicitação inválida

Status HTTP: 400 Solicitação inválida

```
{
```

```
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "invalid",
      "details": {
        "text": "Invalid parameter combination provided"
      }
    }
  ]
}
```

Recurso não encontrado

Status HTTP: 404 não encontrado

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "not-found",
      "details": {
        "text": "Resource not found."
      }
    }
  ]
}
```

Conflito de versão

Status HTTP: 409 Conflito

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "conflict",
      "details": {
        "text": "Resource version conflict detected"
      }
    }
```

```
  ]
}
```

Status de atribuição inválido

Status HTTP: 422 Entidade não processável

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

Regras de negócios

Validação do status de atribuição

A operação só pode ser realizada quando o Status de Atribuição do Grupo é:

- `draft`
- `open`

As operações não são permitidas quando o status é`final`.

Prevenção de membros duplicados

O sistema evita a adição de membros duplicados com base na combinação exclusiva de:

- Identificador de membro
- Identificador do pagador
- Identificador de cobertura

Validação do período de cobertura

Quando um `attributionPeriod` é fornecido, ele deve estar dentro dos limites do período de cobertura do membro. O sistema vai:

- Pesquise o recurso de cobertura do membro
- Use a cobertura mais recente (versionID mais alta) se existirem várias
- Valide que o período de atribuição não exceda o período de cobertura

Validação de referência

Quando a ID e a referência são fornecidas para o mesmo recurso (paciente ou profissional), o sistema valida que elas correspondem ao mesmo recurso.

Quando os campos ID e reference.identifier são fornecidos para o mesmo recurso (paciente ou provedor), ocorre um erro.

Autenticação e autorização

A operação requer autorização SMART on FHIR para:

- Permissões de leitura - Para validar recursos de pacientes, provedores e grupos
- Permissões de pesquisa - Para encontrar recursos por identificador
- Atualizar permissões - Para modificar o recurso do Grupo

Comportamento operacional

Atualizações de recursos

- Atualiza o ID da versão do recurso do grupo
- Cria uma entrada de histórico com o estado original do recurso antes da operação
- Adiciona informações do membro à matriz group.Member com:
  - Referência do paciente em entity.reference
  - Período de atribuição em período
  - Cobertura e informações do provedor em campos de extensão

Passos de validação

- Validação de parâmetros - Garante combinações de parâmetros válidas
- Existência de recursos - Valida a existência de recursos para pacientes, provedores e grupos
- Status de atribuição - Confirma que o status do grupo permite modificações
- Verificação duplicada - Impede a adição de membros existentes

- Validação da cobertura - Garante que o período de atribuição esteja dentro dos limites da cobertura

Limitações

- Todos os recursos referenciados devem existir no mesmo armazenamento de dados
- A operação só funciona com recursos do Grupo de Lista de Atribuição de Membros
- Os períodos de atribuição devem estar dentro dos limites de cobertura
- Não é possível modificar grupos com status “final”

`$member-match`operação para HealthLake

AWS HealthLake agora oferece suporte à `$member-match` operação de recursos para pacientes, permitindo que as organizações de saúde encontrem o identificador exclusivo de um membro em diferentes sistemas de saúde usando informações demográficas e de cobertura. Essa operação é essencial para alcançar a conformidade com o CMS e facilitar a troca segura de payer-to-payer dados, mantendo a privacidade do paciente.

Essa operação é particularmente útil quando você precisa:

- Permita a troca segura de dados de saúde entre organizações
- Mantenha a continuidade do atendimento ao paciente em diferentes sistemas
- Support os requisitos de conformidade do CMS
- Facilite a identificação precisa dos membros em todas as redes de saúde

Usage

A `$member-match` operação pode ser invocada nos recursos do paciente usando o método POST:

```
POST [base]/Patient/$member-match
```

Parâmetros compatíveis

HealthLake suporta os seguintes parâmetros FHIR: `$member-match`

Parâmetro	Tipo	Obrigatório	Padrão	Description
MemberPatient	Paciente	Sim	—	Recurso para pacientes contendo informações demográficas para o membro a ser combinado
CoverageToMatch	Cobertura	Sim	—	Recurso de cobertura que será usado para comparar com os registros existentes
CoverageToLink	Cobertura	Não	—	Recurso de cobertura a ser vinculado durante o processo de correspondência
Consentimento	Consentimento	Não	—	Recurso de consentimento para fins de autorização

Exemplos

Solicitação POST com parâmetros

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

Resposta da amostra

A operação retorna um recurso de Parâmetros contendo os resultados correspondentes:

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

Parâmetros de resposta

A resposta inclui os seguintes parâmetros quando uma correspondência é encontrada:

Parâmetro	Tipo	Description
MemberIdentifier	Identificador	O identificador exclusivo do membro correspon dente
MemberId	Referência	Referência ao recurso do paciente
algoritmo de correspondência	String	Tipo de algoritmo de correspondência usado (EXACT_MATCH, STRONG_MATCH ou DEMOGRAPHIC_MATCH)
Detalhes da partida	String	Informações detalhadas sobre o processo de correspondência
Campos correspon dentes	String	Lista de campos específicos que foram combinados com sucesso

Algoritmos correspondentes

A `$member-match` API emprega uma abordagem de correspondência em vários níveis para garantir a identificação precisa dos membros:

COINCIDÊNCIA EXATA

Usa o identificador do paciente combinado com a cobertura SubscriberId

Fornece o mais alto nível de confiança para correspondência de membros

STRONG_MATCH

Usa o Patient Identifier com informações mínimas de cobertura

Oferece alta confiança quando os critérios de correspondência exata não são atendidos

CORRESPONDÊNCIA DEMOGRÁFICA

Baseia-se em informações demográficas básicas

Usado quando a correspondência baseada em identificador não é possível

Comportamento

A `$member-match` operação:

- Aceita dados demográficos do paciente, detalhes da cobertura e informações de consentimento opcionais como entrada
- Retorna um identificador de membro exclusivo que pode ser usado para interações subsequentes
- Implementa a correspondência em vários níveis (exata, forte, demográfica) para garantir a identificação precisa dos membros em diferentes sistemas de saúde
- Salva todas as informações de consentimento fornecidas para fins de autorização futura
- Oferece suporte à troca segura de payer-to-payer dados enquanto mantém a privacidade do paciente
- Cumpre os requisitos do CMS para troca de dados de saúde

Autorização

A API usa SMART no protocolo de autorização FHIR com os seguintes escopos obrigatórios:

- `system/Patient.read`
- `system/Coverage.read`
- `system/Organization.read`(condicional)
- `system/Practitioner.read`(condicional)
- `system/PractitionerRole.read`(condicional)
- `system/Consent.write`(condicional)

Tratamento de erros

A operação trata das seguintes condições de erro:

- `400 Bad Request`: `$member-match` operação inválida (solicitação não conforme ou ausência de parâmetros obrigatórios)
- `422 Unprocessable Entity`: Nenhuma correspondência ou várias correspondências encontradas

`$member-remove`operação para HealthLake

A `$member-remove` operação permite que você remova membros de uma Lista de Atribuição de Membros do FHIR (recurso de grupo) em. AWS HealthLake Essa operação faz parte do Guia de Implementação de Atribuição de DaVinci Membros e apoia o processo de reconciliação para gerenciar as atribuições de membros.

Pré-requisitos

- AWS HealthLake Armazenamento de dados FHIR
- Permissões apropriadas do IAM para HealthLake operações
- Lista de atribuição de membros (recurso do grupo) em status de rascunho ou aberto

Detalhes da operação

Endpoint

```
POST /Group/{id}/$member-remove
```

Tipo de conteúdo

```
application/fhir+json
```

Parâmetros

A operação aceita um recurso de parâmetros FHIR com os seguintes parâmetros opcionais:

Parâmetro	Cardinalidade	Tipo	Description
memberId	0,1	Identificador	Identificador comercial do membro a ser removido
NPI do provedor	0,1	Identificador	NPI do provedor atribuído
Referência do paciente	0,1	Referência	Referência direta ao recurso do paciente
Referência do provedor	0,1	Referência	Referência direta ao recurso do provedor (profissional ou organizaç ão) PractitionerRole
Referência de cobertura	0,1	Referência	Referência ao recurso de cobertura

Combinações de parâmetros suportadas

As seguintes combinações de parâmetros são suportadas:

- `memberId`somente - Remove todas as atribuições do membro especificado
- `memberId`+ `providerNpi` - Remove as atribuições da combinação específica de membro-provedor
- `patientReference`somente - Remove todas as atribuições do paciente especificado
- `patientReference`+ `providerReference` - Remove as atribuições da combinação específica paciente-provedor
- `patientReference`+ `providerReference` + `coverageReference` - Remove a atribuição específica com base no paciente, no provedor e na cobertura

Exemplo de solicitação

```
{
```

```
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "patientReference",
      "valueReference": {
        "reference": "Patient/12345"
      }
    },
    {
      "name": "providerReference",
      "valueReference": {
        "reference": "Practitioner/67890"
      }
    }
  ]
}
```

Resposta

Resposta bem-sucedida

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

Comportamento

Requisitos de status

- A operação só funciona em listas de atribuição com status ou `draft` `open`
- Listas com `final` status rejeitarão a operação com um erro 422

Processo de remoção de membros

- Listas de status preliminares: os membros são marcados como inativos (`inactive: true`) e sua `changeType` extensão é atualizada para `changed`
- Listas de status abertas: comportamento semelhante ao status de rascunho
- Listas de status finais: A operação foi rejeitada

Validação

- As referências são validadas para garantir que existam no armazenamento de dados HealthLake
- Se o identificador e a referência forem fornecidos para o mesmo tipo de recurso, eles deverão corresponder ao mesmo recurso
- As combinações de parâmetros são validadas de acordo com os padrões suportados

Tratamento de erros

Respostas de erro comuns

Recurso não encontrado (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"]
    }
  ]
}
```

Status final da lista de atribuições (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"]
    }
  ]
}
```

Operação inválida (400)

Retornado quando as combinações de parâmetros são inválidas ou malformadas.

Várias correspondências encontradas (412)

Retornado quando os parâmetros fornecidos correspondem a vários membros na lista de atribuição.

```
{
  "resourceType": "OperationOutcome",
```

```
  "issue": [
    {
      "severity": "error",
      "code": "processing",
      "diagnostics": "Multiple members found matching the criteria"
    }
  ]
}
```

Práticas recomendadas

- Use parâmetros específicos: quando possível, use a combinação de parâmetros mais específica para evitar remoções não intencionais
- Status da lista de verificação: verifique o status da lista de atribuição antes de tentar remoções
- Lide com erros normalmente: implemente o tratamento adequado de erros para todas as condições de erro possíveis
- Validar referências: certifique-se de que todos os recursos referenciados existam antes de fazer a solicitação

Removendo recursos do compartimento do paciente com **`$purge`**

AWS HealthLake suporta a `$purge` operação, permitindo a exclusão permanente de todos os recursos dentro do compartimento do paciente. Essa operação é particularmente útil quando você precisa:

- Remova todos os dados associados a um paciente
- Cumpra as solicitações de remoção de dados do paciente
- Gerencie o ciclo de vida dos dados do paciente
- Execute uma limpeza abrangente do prontuário do paciente

Usage

A `$purge` operação pode ser invocada nos recursos do paciente:

```
POST [base]/Patient/[ID]/$purge?deleteAuditEvent=true
```

Parâmetros

Parâmetro	Tipo	Obrigatório	Padrão	Description
`deleteAuditEvent`	booliano	Não	false	Quando verdadeiro, exclui os eventos de auditoria associados
`_since`	string	Não	Horário de criação do armazenamento de dados	Quando inserido, seleciona a hora limite inicial para encontrar recursos com base na hora da Última Modificação. Não pode ser usado com início ou fim
`start`	string	Não	Horário de criação do armazenamento de dados	Quando inserido, seleciona o horário limite para encontrar recursos com base no horário da Última Modificação. Pode ser usado com extremidade
`end`	string	Não	Horário de envio do trabalho	Quando inserido, seleciona a hora limite final para encontrar recursos com base na hora da Última Modificação

Exemplos

Exemplo de solicitação

```
POST [base]/Patient/example-patient/$purge?deleteAuditEvent=true
```

Exemplo de resposta

```
{
  "resourceType": "OperationOutcome",
  "id": "purge-job",
  "issue": [
    {
      "severity": "information",
      "code": "informational",
```

```
      "diagnostics": "Purge job started successfully. Job ID:
 12345678-1234-1234-1234-123456789012"
    }
  ]
}
```

Status do trabalho

Para verificar o status de uma tarefa de expurgação:

```
GET [base]/$purge/[jobId]
```

A operação retorna informações sobre o status do trabalho:

```
{
      "datastoreId": "36622996b1fcecb7e12ee2ee085308d3",
      "jobId": "3dd1c7a5b6c0ef8c110f566eb87e2ef9",
      "status": "COMPLETED",
      "submittedTime": "2025-10-31T18:43:21.822Z"
    }
```

Comportamento

A `$purge` operação:

1. Processa de forma assíncrona para lidar com vários recursos
2. Mantém as transações ACID para a integridade dos dados
3. Fornece rastreamento do status do trabalho com contagens de exclusões de recursos
4. Remove permanentemente todos os recursos no compartimento do paciente
5. Inclui registro abrangente de auditoria das atividades de exclusão
6. Oferece suporte à exclusão seletiva de eventos de auditoria

Registro de auditoria

A `$purge` operação é registrada como Iniciar FHIRBulk DeleteJob e Descrever FHIRBulk DeleteJob com informações detalhadas da operação.

Limitações

- Os recursos eliminados não aparecerão nas respostas de pesquisa

- Os recursos que estão sendo eliminados podem ficar temporariamente inacessíveis durante o processamento
- Todos os recursos no compartimento do paciente são removidos permanentemente

Operação FHIR para **$questionnaire-package** HealthLake

A `$questionnaire-package` operação recupera um pacote abrangente contendo um questionário FHIR e todas as dependências necessárias para renderizar e processar o questionário. Essa operação implementa o Guia de Implementação de Modelos e Regras de Documentação Da Vinci (DTR), permitindo a renderização dinâmica de formulários para requisitos de documentação em fluxos de trabalho de saúde.

Como funciona

- Solicitação: você envia parâmetros identificando o (s) questionário (s) necessário (s), juntamente com a cobertura e o contexto do pedido
- Recuperar: HealthLake reúne o questionário e todas as dependências (bibliotecas CQLValueSets, etc.)
- Package: Todos os recursos são agrupados em um formato padronizado
- Responder: Você recebe um pacote completo pronto para renderização e coleta de dados

Casos de uso

- Documentação de autorização prévia: colete as informações clínicas necessárias para solicitações de autorização prévia
- Requisitos de cobertura: reúna a documentação necessária para satisfazer os requisitos de cobertura do pagador
- Clinical Data Exchange: Estruture os dados clínicos para envio aos pagadores
- Formulários dinâmicos: renderize questionários com dados pré-preenchidos do paciente e lógica condicional

Ponto final da API

```
POST /datastore/{datastoreId}/r4/Questionnaire/$questionnaire-package
Content-Type: application/fhir+json
```

Parâmetros de solicitação

Parâmetros de entrada

O corpo da solicitação deve conter um recurso de parâmetros FHIR com os seguintes parâmetros:

Parâmetro	Tipo	Cardinalidade	Description
`coverage`	Cobertura	1.. * (Obrigatório)	Recurso (s) de cobertura para estabelecer o membro e cobertura para documentação
`questionnaire`	canônico	0.. *	URL (s) canônica (s) para que questionários específicos retornem (podem incluir a versão)
`order`	Recurso	0.. *	Solicite recursos (DeviceRequest,, ServiceRequest MedicationRequest, Encontro, Nomeação) para estabelecer o contexto
`changedSince`	dateTime	0,1	Se estiver presente, retorne somente os recursos alterados após esse registro de data e hora

Regras de validação de parâmetros

Pelo menos UM dos seguintes itens deve ser fornecido (além do obrigatório`coverage`):

- Um ou mais `questionnaire` canônicos URLs
- Um ou mais `order` recursos

Combinações de solicitações válidas:

- `coverage` + `questionnaire`
- `coverage` + `order`
- `coverage` + `questionnaire` + `order`

Exemplo de solicitação

```
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"
    }
  ]
}
```

Formato de resposta

Resposta de sucesso (200 OK)

A operação retorna um recurso FHIR Parameters contendo um ou mais Package Bundles. Cada pacote Package inclui:

Tipo de entrada	Cardinalidade	Description
Questionário	1	O questionário a ser entregue
QuestionnaireResponse	0,1	Resposta pré-preenchida ou parcialmente preenchida (se aplicável)
Ferramentas	0.. *	Bibliotecas CQL contendo pré-população e lógica condicional
ValueSet	0.. *	Expandido ValueSets (para opções de resposta com menos de 40 expansões)

Example Exemplo de resposta


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

Fluxo de trabalho operacional

Como HealthLake processa sua solicitação

Quando você liga`$questionnaire-package`, HealthLake executa as seguintes etapas:

1. Identifique o paciente e o pagador: extrai o paciente e a organização de seguros do seu `coverage` parâmetro.
2. Encontre o questionário certo:
   - Com **`questionnaire`** parâmetro: usa o URL canônico que você forneceu
   - Com **`order`** parâmetro: combina o código do pedido (CPT/HCPCS/LOINC) e o pagador para encontrar o questionário apropriado
3. Coletar dependências: recupera automaticamente tudo o que é necessário para renderizar o questionário:
   - Bibliotecas CQL - Lógica para questões pré-populacionais e condicionais
   - ValueSets- Opções de resposta (expandidas automaticamente se <40 opções)
   - QuestionnaireResponse- Qualquer resposta existente em andamento ou concluída
4. Package tudo junto:
   - Agrupa todos os recursos (cada recurso incluído apenas uma vez)
   - Filtra por `changedSince` data e hora, se fornecido
   - Adiciona avisos `Outcome` se algum recurso estiver faltando

Resultado: um pacote completo e independente pronto para renderização.

Respostas de erro

400 solicitação inválida

Retornado quando a validação da solicitação falha.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "required",
    "details": {
      "text": "At least one of 'questionnaire' or 'order' must be provided along with
 'coverage'"
    }
  }]
}
```

424 Dependência falhada

Retornado quando um recurso dependente não pode ser recuperado.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "warning",
    "code": "not-found",
    "details": {
      "text": "Referenced Library 'http://example.org/fhir/Library/missing-library'
 could not be retrieved"
    }
  }]
}
```

401 Não autorizado

Retornado quando as credenciais de autenticação estão ausentes ou são inválidas.

403 proibido

Retornado quando o usuário autenticado não tem permissão para acessar os recursos solicitados.

406 Não aceitável

Retornado quando o tipo de conteúdo solicitado não pode ser fornecido.

409 Conflito

Retornado quando há um conflito de versão ou simultaneidade.

410 Desaparecido

Retornado quando o recurso solicitado foi excluído permanentemente.

429, muitas solicitações

Retornado quando os limites de taxa são excedidos.

500 Internal Server Error

Retornado quando ocorre um erro inesperado no servidor.

501 Não implementado

Retornado quando a operação solicitada ainda não foi implementada.

Regras de validação

Validação de entrada

- `coverage`parâmetro é obrigatório (1.. * cardinalidade)
- Pelo menos um dos `questionnaire` ou `order` deve ser fornecido
- Todos os recursos de cobertura devem ser recursos FHIR válidos
- Todos os recursos do Pedido devem ser recursos FHIR válidos
- O canônico URLs deve ser formatado corretamente
- `changedSince`deve ser um ISO 8601 DateTime válido

QuestionnaireResponse validação

- `status`deve ser apropriado (`in-progress`,`completed`,`amended`)
- A estrutura deve corresponder ao questionário referenciado
- `based0n`deve fazer referência a recursos válidos do Pedido
- `subject`deve referenciar recursos válidos para pacientes

Desduplicação de recursos

- Cada recurso aparece somente uma vez no pacote
- Exceção: versões diferentes do mesmo recurso podem ser incluídas
- Os recursos são identificados por seu URL e versão canônicos

Especificações de performance

Métrica	Especificação
Limite de contagem de recursos	500 recursos por pacote
Limite de tamanho do pacote	Máximo de 5 MB

Permissões obrigatórias

Para usar a `$questionnaire-package` operação, certifique-se de que sua função do IAM tenha:

- `healthlake:QuestionnairePackage`- Para chamar a operação
- `healthlake:ReadResource`- Para recuperar o questionário e os recursos dependentes
- `healthlake:SearchWithPost`- Para pesquisar QuestionnaireResponse e recursos relacionados

SMART em escopos FHIR

Escopos mínimos exigidos:

- SMART v1: `user/Questionnaire.read user/Library.read user/ValueSet.read user/QuestionnaireResponse.read`
- SMART v2: `user/Questionnaire.rs user/Library.rs user/ValueSet.rs user/QuestionnaireResponse.rs`

Notas de implementação importantes

Estratégia de recuperação de recursos

Prioridade de identificação do questionário:

- URL canônica (se o `questionnaire` parâmetro for fornecido) - Prioridade mais alta
- Análise do pedido (se o `order` parâmetro for fornecido):
  - Combine os códigos de pedido (CPT, HCPCS, LOINC) com as políticas médicas do pagador
  - Use o pagador de cobertura para filtrar questionários específicos do pagador
  - Considere os códigos de motivo para um contexto adicional

Resolução de dependências

Bibliotecas CQL:

- Recuperado por meio da `cqf-library` extensão nos recursos do questionário

- Busca recursivamente bibliotecas dependentes por meio de tipo `Library.relatedArtifact` `depends-on`
- Todas as dependências da biblioteca estão incluídas no pacote

ValueSets:

- Expandido automaticamente se eles contiverem menos de 40 conceitos
- Maiores ValueSets estão incluídos sem expansão
- ValueSets referenciados no questionário e nos recursos da biblioteca estão incluídos

QuestionnaireResponse pré-população

A operação pode retornar um QuestionnaireResponse com dados pré-preenchidos quando:

- Uma resposta existente em andamento ou concluída foi encontrada
- A lógica CQL nas bibliotecas associadas pode extrair dados dos registros dos pacientes
- A resposta está vinculada ao pedido e à cobertura relevantes

Critérios de pesquisa para QuestionnaireResponse:

Parâmetro de pesquisa	Caminho FHIR	Description
`based-on`	`QuestionnaireResponse.basedOn`	Links para ServiceRequest ou CarePlan
`patient`	`QuestionnaireResponse.subject`	O paciente que é o sujeito
`questionnaire`	`QuestionnaireResponse.questionnaire`	O questionário que está sendo respondido

Filtragem de recursos alterada

Quando o `changedSince` parâmetro é fornecido:

- Somente os recursos modificados após o registro de data e hora especificado são incluídos
- Se nenhum recurso tiver sido alterado, retorna `200 OK` com um pacote vazio
- Útil para atualizações incrementais e estratégias de armazenamento em cache
- A comparação de timestamp usa o campo de recurso `meta.lastUpdated`

Vários pacotes

A operação pode retornar vários Package Bundles quando:

- Vários questionários são solicitados via canonical URLs
- Vários pedidos exigem questionários diferentes
- Versões diferentes do mesmo questionário são aplicáveis

Cada Package Bundle é independente com todas as dependências necessárias.

Casos de uso comuns

Caso de uso 1: documentação de autorização prévia

Cenário: Um provedor precisa coletar a documentação para uma autorização prévia de oxigenoterapia domiciliar.

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

```
    }
  ]
}
```

Resultado: Retorna um pacote com o questionário de oxigenoterapia, pré-preenchido com os sinais vitais do paciente e os códigos de diagnóstico do EHR.

Caso de uso 2: Recuperar a versão específica do questionário

Cenário: um provedor precisa de uma versão específica de um questionário para fins de conformidade.

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "coverage",
      "resource": { /* Coverage resource */ }
    },
    {
      "name": "questionnaire",
      "valueCanonical": "http://example.org/fhir/Questionnaire/dme-request|3.1.0"
    }
  ]
}
```

Resultado: retorna exatamente a versão 3.1.0 do questionário de solicitação do DME com todas as dependências.

Caso de uso 3: verifique se há atualizações

Cenário: um provedor deseja verificar se algum recurso do questionário foi atualizado desde a última recuperação.

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

Resultado: retorna somente recursos que foram modificados após 1º de março de 2024 ou um pacote vazio se nada tiver sido alterado.

Caso de uso 4: vários pedidos

Cenário: um provedor envia várias solicitações de serviço que podem exigir questionários diferentes.

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "coverage",
      "resource": { /* Coverage resource */ }
    },
    {
      "name": "order",
      "resource": { /* ServiceRequest for imaging */ }
    },
    {
      "name": "order",
      "resource": { /* ServiceRequest for DME */ }
    }
  ]
}
```

Resultado: retorna vários Package Bundles, um para cada questionário aplicável.

Integração com outros Da Vinci IGs

Descoberta de requisitos de cobertura (CRD)

Integração do fluxo de trabalho:

- O provedor solicita um serviço em seu EHR
- O gancho do CRD dispara, verificando os requisitos de cobertura
- O pagador responde indicando que a documentação é necessária
- O provedor liga `$questionnaire-package` para recuperar o formulário de documentação
- O provedor preenche o questionário
- A documentação é enviada via PAS ou CDex

Suporte de autorização prévia (PAS)

Integração do fluxo de trabalho:

- Use `$questionnaire-package` para recuperar os requisitos de documentação
- Preencha o QuestionnaireResponse com os dados clínicos necessários
- Envie a autorização prévia usando `Claim/$submit` com o preenchido QuestionnaireResponse
- Verifique o status usando `Claim/$inquire`

Troca de dados clínicos (CDex)

Integração do fluxo de trabalho:

- O pagador solicita documentação adicional para uma reclamação
- O provedor usa `$questionnaire-package` para recuperar o formulário estruturado de coleta de dados
- O provedor conclui o QuestionnaireResponse
- A documentação é enviada ao pagador por meio do fluxo de trabalho de CDex anexos

Guia de solução de problemas

Problema: Nenhum questionário foi devolvido

Causas possíveis:

- O URL canônico não corresponde a nenhum questionário no armazenamento de dados

- O código do pedido não é mapeado para nenhum questionário na política médica do pagador
- O pagador da cobertura não tem questionários associados

Soluções:

- Verifique se o URL canônico está correto e se o questionário existe
- Verifique se os códigos de pedido (CPT/HCPCS) estão especificados corretamente
- Confirme se a organização pagadora tem questionários configurados

Problema: dependências ausentes no Package

Causas possíveis:

- Biblioteca referenciada ou ValueSet não existe no armazenamento de dados
- As referências da biblioteca estão quebradas ou incorretas
- ValueSet a expansão falhou

Soluções:

- Verifique o `Outcome` parâmetro para ver se há avisos sobre recursos ausentes
- Verifique se todos os recursos referenciados existem em seu armazenamento de dados
- Certifique-se de que ValueSet URLs estão corretos e solucionáveis

Problema: Empty Package com ChangedSince

Causas possíveis:

- Esse é o comportamento esperado - nenhum recurso foi modificado desde o timestamp especificado

Soluções:

- Use a versão em cache do pacote

- Remova o `changedSince` parâmetro para recuperar o pacote completo

Problema: QuestionnaireResponse Não pré-preenchido

Causas possíveis:

- Não foi QuestionnaireResponse encontrado nenhum existente
- A lógica da biblioteca CQL não conseguiu extrair os dados necessários
- Os dados do paciente estão ausentes ou incompletos

Soluções:

- Isso pode ser esperado - nem todos os questionários têm lógica pré-populacional
- Verifique se os dados do paciente existem no armazenamento de dados
- Revise a lógica da Biblioteca CQL para requisitos de extração de dados

Práticas recomendadas

1. Use Canonical com versões URLs

Sempre especifique os números das versões ao solicitar questionários específicos:

```
{
  "name": "questionnaire",
  "valueCanonical": "http://example.org/fhir/Questionnaire/dme|2.1.0"
}
```

Por quê: Garante a consistência e evita alterações inesperadas quando os questionários são atualizados.

2. Aproveite o ChangedSince para desempenho

Para questionários acessados com frequência, use `changedSince` para minimizar a transferência de dados:

```
{
  "name": "changedSince",
```

```
  "valueDateTime": "2024-03-10T15:30:00Z"
}
```

Por quê: Reduz a latência e o uso da largura de banda recuperando somente recursos atualizados.

3. Inclua informações completas de cobertura

Forneça detalhes abrangentes da cobertura para garantir a seleção correta do questionário:

```
{
  "name": "coverage",
  "resource": {
    "resourceType": "Coverage",
    "beneficiary": { "reference": "Patient/123" },
    "payor": [{ "reference": "Organization/payer-abc" }],
    "class": [{ /* Group/plan information */ }]
  }
}
```

Por quê: Ajuda a HealthLake identificar questionários e requisitos específicos do pagador.

Operações do relacionadas

- `Claim/$submit`- Envie solicitações de autorização prévia com a documentação completa
- `Claim/$inquire`- Verifique o status das autorizações anteriores enviadas
- `ValueSet/$expand`- Expandir ValueSets para opções de resposta

Operação FHIR para **`$submit`** HealthLake

A `$submit` operação permite que você envie eletronicamente solicitações de autorização prévia aos pagadores para aprovação. Essa operação implementa o Guia de Implementação do Da Vinci Prior Authorization Support (PAS), fornecendo um fluxo de trabalho padronizado baseado em FHIR para envios de autorização prévia.

Como funciona

- Enviar: Você envia um pacote FHIR contendo sua solicitação de autorização prévia e dados clínicos de apoio
- Validar: HealthLake valida o envio de acordo com os requisitos do PAS
- Persistir: todos os recursos são armazenados em seu armazenamento HealthLake de dados

- Responder: você recebe uma resposta imediata com o status “em fila”
- Processo: A decisão de autorização é processada de forma assíncrona pelo pagador

Ponto final da API

```
POST /datastore/{datastoreId}/r4/Claim/$submit
Content-Type: application/fhir+json
```

Estrutura de solicitações

Requisitos do pacote

Sua solicitação deve ser um recurso do FHIR Bundle com:

- Bundle.type: Deve ser `"collection"`
- Bundle.entry: deve conter exatamente um recurso de reivindicação com `use = "preauthorization"`
- Recursos referenciados: todos os recursos referenciados pela reivindicação devem ser incluídos no pacote

Recursos necessários

Recurso	Cardinalidade	Perfil	Description
Reclamação	1	Reivindicação do PAS	A solicitação de autorização prévia
Paciente	1	Paciente PAS	Informações demográficas do paciente
Organização (Seguradora)	1	Seguradora PAS	Companhia de seguros
Organização (Provedor)	1	Solicitante do PAS	Provedor de serviços de saúde enviando a solicitação
Cobertura	1 ou mais	Cobertura do PAS	Detalhes da cobertura do seguro

Recursos opcionais

Recurso	Cardinalidade	Perfil	Description
Praticante	0 ou mais	Praticante do PAS	Profissionais de saúde
Practitio nerRole	0 ou mais	PAS PractitionerRole	Funções do profissional
ServiceRe quest	0 ou mais	PAS ServiceRequest	Serviços médicos solicitados
DeviceRequest	0 ou mais	PAS DeviceRequest	Dispositivos médicos solicitados
Medicatio nRequest	0 ou mais	PAS MedicationRequest	Medicamentos solicitados
DocumentR eference	0 ou mais	PAS DocumentR eference	Documentação clínica de apoio

Exemplo de solicitação

```
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"
    }
  }]
```

```
}
```

Formato de resposta

Resposta de sucesso (200 OK)

Você receberá um pacote de respostas do PAS contendo:

- ClaimResponsecom `outcome: "queued"` e `status: "active"`
- Todos os recursos originais de sua solicitação
- Carimbo de data e hora confirmando o recebimento

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

Respostas de erro

400 solicitação inválida

Retornado quando o formato da solicitação é inválido ou está incorreto.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "invalid",
    "diagnostics": "The provided payload was invalid and could not be parsed
 correctly."
  }]
}
```

412 Falha na pré-condição

Retornado quando a mesma solicitação de autorização prévia já foi enviada (envio duplicado detectado).

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "processing",
    "diagnostics": "PreAuth Claim already exists"
  }]
}
```

> **Idempotência**
>
> A `$submit` operação é idempotente. Enviar a mesma solicitação várias vezes não criará solicitações de autorização prévia duplicadas. Em vez disso, você receberá um erro 412 solicitando que você use `$inquire` para verificar o status do seu envio original.

422 Entidade não processável

Retornado quando a validação do FHIR falha.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "required",
    "diagnostics": "Bundle contains more than one preauthorization claim"
  }]
}
```

Regras de validação

HealthLake realiza uma validação abrangente em seu envio:

Validação do pacote

- Deve estar em conformidade com o perfil do PAS Request Bundle
- `Bundle.type`deve ser `"collection"`
- Pode conter vários recursos de reivindicação
- No entanto, deve conter exatamente um recurso de reclamação com uso pré-autorizado
  - E esse recurso de reivindicação deve ser a primeira entrada no pacote
- Todos os recursos referenciados devem ser incluídos no pacote

Validação da solicitação

- Deve estar em conformidade com o perfil de reclamação do PAS
- `Claim.use`deve ser `"preauthorization"`
- Campos obrigatórios: `patient`,`insurer`,`provider`,`created`, `priority`
- Os identificadores comerciais devem estar presentes e válidos

Validação de recursos

- Todos os recursos devem estar em conformidade com seus respectivos perfis PAS
- Os recursos de apoio necessários devem estar presentes (paciente, cobertura, organização)
- As referências cruzadas devem ser válidas e solucionáveis dentro do pacote

Especificações de performance

Métrica	Especificação
Limite de tamanho do pacote	Máximo de 5 MB
Limite de contagem de recursos	500 recursos por pacote

Permissões obrigatórias

Para usar a `$submit` operação, é possível usar o AWS Sigv4 ou o SMART no FHIR:

- Certifique-se de que sua função do IAM tenha: `healthlake:SubmitPreAuthClaim` - Para chamar a operação

SMART em escopos FHIR

Escopos mínimos exigidos:

- SMART v1: `user/Claim.write & <all_resourceTypes_in_Bundle>.write`
- SMART v2: `user/Claim.c & <all_resourceTypes_in_Bundle>.c or system/*.*`

Notas de implementação importantes

Persistência de recursos

- Todas as entradas do pacote são mantidas como recursos individuais do FHIR em seu armazenamento de dados
- Os fornecidos pelo cliente IDs são preservados quando fornecidos
- O histórico de versões é mantido para fins de auditoria
- A detecção de duplicatas evita conflitos de recursos

Comportamento de processamento

- Cada envio válido resulta em exatamente um ClaimResponse com `"queued"` resultado
- Envios inválidos retornam códigos de status 400 ou 422 com informações detalhadas de erro
- Os erros do sistema retornam os códigos de status 5xx apropriados

- Todos os envios bem-sucedidos retornam o status 200 com um valor pendente ClaimResponse

Requisitos do pacote

- `Bundle.entry.fullUrl`os valores devem ser REST URLs ou `"urn:uuid:[guid]"` formatados
- Todos GUIDs devem ser exclusivos em todos os envios (exceto nas mesmas instâncias de recursos)
- Os recursos referenciados devem existir dentro do pacote ou ser solucionáveis

Operações do relacionadas

- `Claim/$inquire`- Consulte o status de uma solicitação de autorização prévia enviada
- `Patient/$everything`- Recupere dados abrangentes do paciente para contexto de autorização prévia

Validando recursos do FHIR com **`$validate`**

AWS HealthLake agora suporta a `$validate` operação de recursos FHIR, permitindo que você valide um recurso de acordo com a especificação FHIR e verifique sua conformidade com um perfil específico ou definição de recurso básico sem realizar nenhuma operação de armazenamento. Essa operação é particularmente útil quando você precisa:

- Valide os requisitos de conformidade do FHIR CMS
- Teste os recursos antes de usá-los na produção
- Forneça feedback de validação em tempo real à medida que os usuários editam dados clínicos
- Reduza os envios de dados inválidos para criar e atualizar APIs

Usage

A `$validate` operação pode ser invocada em recursos FHIR usando métodos POST:

Operações com Suporte

```
POST [base]/[type]/[id]/$validate
```

```
POST [base]/[type]/$validate
```

Cargas úteis suportadas

Recurso de parâmetros

HealthLake suporta os seguintes parâmetros FHIR: `$validate`

Parâmetro	Tipo	Obrigatório	Description
`resource`	Recurso	Sim	O recurso a ser validado
`profile`	canônico	Não	URL canônico do perfil a ser validado
`mode`	código	Não	Modo de validação:`create`, ou `update`

Recurso direto com parâmetros de consulta

Parâmetro	Tipo	Obrigatório	Description
`profile`	canônico	Não	URL canônico do perfil a ser validado
`mode`	código	Não	Modo de validação:`create`, ou `update`

Exemplos

Solicitação POST de recurso com ID e carga útil de parâmetros

```
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"
    }
  ]
}
```

Solicitação POST para tipo de recurso e carga útil de parâmetros

```
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"
    }
  ]
}
```

Solicitação POST de recurso com ID e carga direta de recursos

```
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"
}
```

Solicitação POST para tipo de recurso e carga direta do recurso

```
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"
}
```

Resposta da amostra

A operação retorna um OperationOutcome recurso com resultados de validação:

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "information",
      "code": "informational",
      "diagnostics": "Validation successful"
    }
  ]
}
```

Exemplo de resposta com erros de validação

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

Comportamento

A `$validate` operação:

1. Valida o recurso em relação à especificação FHIR e à definição do recurso básico
2. Verifica a conformidade com os perfis especificados quando o `profile` parâmetro é fornecido
3. Valida com base no modo especificado (`createouupdate`)
4. Retorna resultados detalhados da validação, incluindo erros, avisos e mensagens informativas
5. Não executa nenhuma operação de armazenamento - somente validação
6. Retorna HTTP 200 OK quando a validação pode ser realizada, independentemente de problemas de validação terem sido encontrados

Modos de validação

- criar: valida o recurso como se estivesse sendo criado (novo recurso)
- atualização: valida o recurso como se estivesse sendo atualizado (recurso existente)

Tratamento de erros

A operação retorna:

- 200 OK: A validação foi realizada com sucesso (independentemente do resultado da validação)
- 400 Solicitação inválida: formato ou parâmetros de solicitação inválidos

- 404 Não encontrado: tipo de recurso ou perfil não encontrado

Para obter mais informações sobre a especificação da `$validate` operação, consulte a documentação do recurso FHIR R4. `$validate`

Referência de conformidade para AWS HealthLake

AWS HealthLake fornece recursos projetados para ajudá-lo a rastrear e relatar o uso da API de acordo com os requisitos de interoperabilidade do CMS (Centers for Medicare & Medicaid Services). Esses recursos permitem que você categorize as transações de API por categorias exigidas pelo CMS e capture automaticamente métricas de uso para fins de relatórios de conformidade.

⚠ Entendendo suas responsabilidades de conformidade

O uso AWS HealthLake de seus endpoints de interoperabilidade do CMS não é suficiente por si só para alcançar a conformidade com o CMS. Você é responsável por:

- Mapeando corretamente seus fluxos de trabalho de API para os endpoints apropriados da categoria CMS com base em seus casos de uso específicos e obrigações regulatórias
- Implementando controles adequados de autenticação e autorização que atendam aos requisitos do CMS
- Garantir que seus recursos e trocas de dados do FHIR estejam em conformidade com os regulamentos e guias de implementação aplicáveis do CMS
- Configurando e monitorando as CloudWatch métricas para atender às suas necessidades de relatórios de conformidade
- Compreender quais regras do CMS se aplicam à sua organização e implementar os controles técnicos e operacionais apropriados

AWS HealthLake fornece a infraestrutura e as ferramentas para apoiar seus esforços de conformidade, mas você deve usar esses recursos adequadamente com base em seus requisitos regulatórios específicos. O simples roteamento de chamadas de API por meio desses endpoints não torna automaticamente seu aplicativo compatível com os regulamentos do CMS.

Tópicos

- Recursos de conformidade do CMS

Recursos de conformidade do CMS

AWS HealthLake fornece recursos para ajudá-lo a atender aos requisitos de interoperabilidade e conformidade do CMS (Centers for Medicare & Medicaid Services). Esses recursos permitem que você acompanhe o uso da API por categoria de CMS e, posteriormente, relate as métricas de uso para fins de conformidade.

Tópicos

- Endpoints de interoperabilidade do CMS
- CloudWatch Métricas aprimoradas para conformidade com o CMS

Endpoints de interoperabilidade do CMS

Visão geral do

HealthLake fornece quatro endpoints de interoperabilidade do CMS que correspondem às categorias de API exigidas pelo CMS. O URL base subjacente do seu armazenamento de HealthLake dados não muda. Esses endpoints simplesmente fornecem uma maneira de categorizar e rastrear suas chamadas de API para fins de geração de relatórios do CMS.

Finalidade

O objetivo principal desses endpoints de interoperabilidade é permitir que os clientes:

- Rastreie facilmente as transações de API por categoria de CMS
- Relate automaticamente as métricas de uso para conformidade com o CMS
- Mantenha os fluxos de trabalho existentes do FHIR com mudanças mínimas

Todas as chamadas de API funcionam de forma idêntica, independentemente de você usar o endpoint de interoperabilidade ou o endpoint FHIR padrão. A única diferença é como as transações são categorizadas em métricas. CloudWatch

Endpoints de interoperabilidade CMS compatíveis

Categoria CMS	Endpoint de interoperabilidade	Exemplo de uso
Acesso do paciente	`/patientaccess/v2/r4`	`baseURL/patientaccess/v2/r4/Patient/123`

Categoria CMS	Endpoint de interoperabilidade	Exemplo de uso
Acesso do provedor	`/provideraccess/v2/r4`	`baseURL/provideraccess/v2/r4/Observation?patient=123`
De jogador para jogador	`/payertopayerdx/v2/r4`	`baseURL/payertopayerdx/v2/r4/Practitioner/456`
Serviço de autenticação prévia	`/priorauthservice/v2/r4`	`baseURL/priorauthservice/v2/r4/ExplanationOfBenefit?patient=789`

Como funcionam os endpoints de interoperabilidade

Chamada HealthLake de API padrão:

```
baseURL/resourceType/[id]
baseURL/resourceType?[parameters]
```

Com o CMS Interoperability Endpoint:

```
baseURL/interoperability-endpoint/resourceType/[id]
baseURL/interoperability-endpoint/resourceType?[parameters]
```

O caminho do endpoint de interoperabilidade é simplesmente inserido entre o URL base e o tipo de recurso. Tudo após o caminho do endpoint de interoperabilidade permanece exatamente igual às suas chamadas de API atuais.

Exemplos de uso

Exemplo 1: API de acesso do paciente

Chamada de API atual (ainda funciona):

```
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"
```

Com o endpoint de interoperabilidade Patient Access (para rastreamento de 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"
```

Pontos-chave:

- O URL base permanece: `https://healthlake.us-east-1.amazonaws.com/datastore/abc123`
- Ponto final de interoperabilidade inserido: `/patientaccess/v2/r4`
- Caminho do recurso inalterado: `/Patient/123`
- Ambas as chamadas retornam respostas idênticas
- A chamada do endpoint de interoperabilidade é rastreada automaticamente em `URIType=patient-access` CloudWatch
- As operações POST, PUT, PATCH, DELETE funcionam de forma idêntica.
- O corpo da solicitação permanece inalterado.

Referência de tradução de endpoint

Endpoint de interoperabilidade	Traduz para	Categoria CMS
`baseURL/patientaccess/v2/r4/Patient`	`baseURL/r4/Patient`	Acesso do paciente
`baseURL/provideraccess/v2/r4/Observation`	`baseURL/r4/Observation`	Acesso do provedor
`baseURL/payertopayerdx/v2/r4/Practitioner/456`	`baseURL/r4/Practitioner/456`	Troca de Dados de Pagador para Pagador
`baseURL/priorauthservice/v2/r4/Expla`	`baseURL/r4/ExplanationOfBenefit?patient=789`	Autorização prévia

Endpoint de interoperabilidade	Traduz para	Categoria CMS
`nationOfBenefit?patient=789`		

Observações importantes

- Sem diferença funcional: os endpoints de interoperabilidade e os endpoints FHIR padrão retornam respostas idênticas e oferecem suporte a operações idênticas
- URL base inalterado: seu endpoint HealthLake de armazenamento de dados permanece o mesmo
- Integração simples: insira o caminho do endpoint de interoperabilidade entre seu URL base e o tipo de recurso
- Rastreamento automático: CloudWatch as métricas categorizam automaticamente as chamadas por endpoint de interoperabilidade usado
- Compatível com versões anteriores: as chamadas de API existentes sem endpoints de interoperabilidade continuam funcionando normalmente

CloudWatch Métricas aprimoradas para conformidade com o CMS

Visão geral do

Quando você usa os endpoints de interoperabilidade do CMS, emite HealthLake automaticamente CloudWatch métricas aprimoradas com dimensões adicionais para suportar os requisitos de relatórios do CMS. Essas métricas rastreiam o uso da API por identidade do chamador, aplicativo e tipos de URI específicos do CMS, sem a necessidade de nenhuma configuração adicional.

Como funciona

Quando você faz uma chamada de API usando um endpoint de interoperabilidade:

```
# 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.)
```

Nenhum código ou configuração adicional é necessário. Basta usar os endpoints de interoperabilidade e as métricas aprimoradas serão capturadas automaticamente.

> **Note**
>
> Para armazenamentos de dados não inteligentes em FHIR, a `URIType` dimensão ainda é capturada, permitindo que você acompanhe o uso da API por categoria de CMS. `ClientId`As dimensões `Sub` e só estão disponíveis ao usar SMART na autenticação FHIR com tokens de portador contendo essas declarações.

Novas dimensões métricas

Além das dimensões existentes (`DatastoreId`,,`Operation`)`DatastoreType`, as seguintes dimensões são adicionadas automaticamente ao usar endpoints de interoperabilidade:

Dimensão	Description	Valores de exemplo	Fonte
URIType	Categoria de conformidade do CMS	`patient-access` , `provider-access` , `payer-to-payer` , `prior-authorization`	Determinado automaticamente a partir do caminho do endpoint de interoperabilidade
Sub	Identidade do chamador	Identificador de usuário/entidade	Extraído da reivindicação do token do portador `sub`
ClientId	Identificador da aplicação	`portal_app` , `ehr_system`	Extraído da reivindicação do token do portador `client_id`

Métricas disponíveis

Todas as HealthLake métricas existentes agora incluem as dimensões adicionais ao usar endpoints de interoperabilidade:

- CallCount- Número total de chamadas de API
- Latência - tempo de resposta da API em milissegundos
- UserErrors- Contagem de 4xx erros do cliente
- SystemErrors- Contagem de 5xx erros do servidor
- Throttles - Contagem de solicitações limitadas
- SuccessfulRequests- Contagem de chamadas de API bem-sucedidas

Consultando métricas em CloudWatch

CloudWatch Exemplo de consulta do Insights

Consulte todas as chamadas da Patient Access API por aplicativo:

```
SELECT SUM(CallCount)
FROM "AWS/HealthLake"
WHERE DatastoreId = '75c1cf9b0d71cd38fec8f7fb317c4c1a'
    AND URIType = 'patient-access'
GROUP BY ClientId
```

Referência de suporte para AWS HealthLake

O seguinte material de referência de apoio está disponível para AWS HealthLake.

> **Note**
>
> Todas as HealthLake ações e tipos de dados nativos são descritos em uma referência separada. Para obter mais informações, consulte a Referência da API do AWS HealthLake .

Tópicos

- AWS HealthLake Endpoints e cotas
- Tipos de dados pré-carregados do Synthea para HealthLake
- AWS HealthLake projetos de amostra
- Solução de problemas AWS HealthLake

- Usando HealthLake com um AWS SDK

AWS HealthLake Endpoints e cotas

Os tópicos a seguir contêm informações sobre pontos AWS HealthLake de extremidade e cotas de serviço.

Tópicos

- Service endpoints
- Cotas de serviço

Service endpoints

Um endpoint de serviço é um URL que identifica um host e a porta como o ponto de entrada para um serviço web. Cada solicitação de serviço da web contém um endpoint. A maioria dos AWS serviços fornece endpoints para regiões específicas para permitir uma conectividade mais rápida. A tabela a seguir lista os endpoints de serviço para AWS HealthLake.

Nome da região	Região	Endpoint	Protocolo	
Leste dos EUA (Ohio)	us-east-2	healthlake.us-east-2.amazonaws.com	HTTPS	
		healthlake-fips.us-east-2.amazonaws.com	HTTPS	
Leste dos EUA (Norte da Virgínia)	us-east-1	healthlake.us-east-1.amazonaws.com	HTTPS	
		healthlake-fips.us-east-1.amazonaws.com	HTTPS	
Oeste dos EUA (Oregon)	us-west-2	healthlake.us-west-2.amazonaws.com	HTTPS	
		healthlake-fips.us-west-2.amazonaws.com	HTTPS	
Ásia-Pacífico (Mumbai)	ap-south-1	healthlake.ap-south-1.amazonaws.com	HTTPS	

Nome da região	Região	Endpoint	Protocolo	
Ásia-Pacífico (Sydney)	ap-southeast-2	healthlake.ap-southeast-2.amazonaws.com	HTTPS	
Canadá (Central)	ca-central-1	healthlake.ca-central-1.amazonaws.com	HTTPS	
Europa (Irlanda)	eu-west-1	healthlake.eu-west-1.amazonaws.com	HTTPS	
Europa (Londres)	eu-west-2	healthlake.eu-west-2.amazonaws.com	HTTPS	

Cotas de serviço

As cotas de serviço são definidas como o valor máximo para recursos, ações e itens em sua AWS conta.

> **Note**
>
> É possível solicitar aumentos de cota para cotas ajustáveis do IAM usando o console do Service Quotas. Para obter mais informações, consulte Solicitando um Aumento de Cota no Guia do Usuário do Service Quotas.
> A taxa da API Sync Write aumenta proporcionalmente com o tamanho da carga, com cada incremento de 1 KB consumindo capacidade adicional (por exemplo, uma carga útil de 4 KB usa 4 vezes a capacidade de gravação). Definir o `x-amz-fhir-history-consistency-level` cabeçalho opcional para `strong` dobrar o consumo de capacidade de gravação por recurso.
> Os recursos dentro dos pacotes seguem read/write limites padrão com base no tamanho da carga útil de 1 KB. A transação do tipo pacote consome o dobro da capacidade de gravação em comparação com o tipo lote, o que significa que os pacotes em lote podem processar duas vezes mais recursos por segundo do que os pacotes de transação.

A tabela a seguir lista as cotas padrão para AWS HealthLake.

Nome	Padrão	Ajustá	Description
Número de assinaturas ativas por conta	Cada região compatível: 100	Sim	O número máximo de recursos de assinatura ativos por conta.
Número de recursos de assinatura ativos por armazenamento de dados	Cada região compatível: 50	Sim	O número máximo de recursos de assinatura ativos por armazenamento de dados.
Número de ativos SubscriptionTopic por conta	Cada região compatível: 100	Sim	O número máximo de SubscriptionTopic recursos ativos por conta.
Número de SubscriptionTopic recursos ativos por armazenamento de dados	Cada região compatível: 50	Sim	O número máximo de SubscriptionTopic recursos ativos por armazenamento de dados.
Número de caracteres em uma observação médica	Cada região compatível: 10.000	Não	O número máximo de caracteres em um atestado médico individual dentro do tipo de DocumentReference recurso (POST/PUT solicitações).
Número de trabalhos simultâneos StartFHIRExportJob	Cada região compatível: 1	Não	O máximo de StartFHIR ExportJob trabalhos simultâneos.
Número de trabalhos simultâneos StartFHIRImportJob	Cada região compatível: 1	Não	O máximo de StartFHIR ImportJob trabalhos simultâneos.

Nome	Padrão	Ajustá	Description
Número de armazenamentos de dados por conta	Cada região com suporte: 10	Sim	O número máximo padrão de armazenamentos de dados ativos por conta.
Número de arquivos em um StartFHIRImportJob	Cada região com suporte: 1.000.000	Sim	O número máximo de arquivos em um StartFHIRImportJob.
Número de recursos por pacote	Cada região com suporte: 500	Não	O número máximo de recursos permitidos em uma solicitação de pacote.
Taxa de CancelFHIRExportJob solicitações usando DELETE por conta	Cada região compatível: 1	Não	O número máximo de CancelFHIRExportJob solicitações usando DELETE que você pode fazer por segundo por conta.
Taxa de solicitações de createFhirDatastore por conta	Cada região compatível: 1	Não	Número máximo de solicitações CreateFHIRDatastore que é possível fazer por minuto por conta.
Taxa de solicitações DeleteFhirDatastore por conta	Cada região compatível: 1	Não	Número máximo de solicitações DeleteFHIRDatastore que é possível fazer por minuto por conta.

Nome	Padrão	Ajustá	Description
Taxa de DescribeFHIRBulkDeleteJob solicitações por conta	Cada região com suporte: 10	Sim	O número máximo de DescribeFHIRBulkDeleteJob solicitações que você pode fazer por segundo por conta.
Taxa de DescribeFHIRBulkDeleteJob solicitações por armazenamento de dados	Cada região com suporte: 10	Sim	O número máximo de DescribeFHIRBulkDeleteJob solicitações que você pode fazer por segundo por armazenamento de dados.
Taxa de solicitações DescribeFhirDatastore por conta	Cada região com suporte: 10	Não	Número máximo de solicitações DescribeFHIRDatastore que é possível fazer por segundo por conta.
Taxa de DescribeFHIRExportJob solicitações por conta	Cada região com suporte: 10	Não	O número máximo de DescribeFHIRExportJob solicitações que você pode fazer por segundo por conta.
Taxa de DescribeFHIRExportJob solicitações usando GET por conta	Cada região com suporte: 10	Não	O número máximo de DescribeFHIRExportJob solicitações usando GET que você pode fazer por segundo por conta.
Taxa de DescribeFHIRImportJob solicitações por conta	Cada região com suporte: 10	Não	O número máximo de DescribeFHIRImportJob solicitações que você pode fazer por segundo por conta.

Nome	Padrão	Ajustá	Description
Taxa de solicitações de descoberta por conta	Cada região com suporte: 10	Não	Número máximo de solicitações Discovery que é possível fazer por minuto por conta.
Taxa de solicitações GET por conta	Cada região compatível: 6.000	Sim	Número máximo de solicitações GET que é possível fazer por segundo por conta.
Taxa de solicitações GET por armazenamento de dados	Cada região compatível: 3.000	Sim	Número máximo de solicitações GET que é possível fazer por segundo por armazenamento de dados. Os armazenamentos de dados criados antes de 8/21 /2023 serão limitados a 100 solicitações por segundo.
Taxa de GetCapabilities solicitações por conta	Cada região com suporte: 10	Não	O número máximo de GetCapabilities solicitações que você pode fazer por segundo por conta.
Taxa de GetExportedFile solicitações por armazenamento de dados	Cada região com suporte: 10	Não	O número máximo de GetExportedFile solicitações que você pode fazer por segundo por armazenamento de dados.

Nome	Padrão	Ajustá	Description
Taxa de solicitações ListFhirDatastores por conta	Cada região com suporte: 10	Não	O número máximo de solicitações ListFHIRD atastores que é possível fazer por segundo por conta.
Taxa de ListFHIRExportJobs solicitaç ões por conta	Cada região com suporte: 10	Não	O número máximo de ListFHIRExportJobs solicitações que você pode fazer por segundo por conta.
Taxa de ListFHIRImportJobs solicitaç ões por conta	Cada região com suporte: 10	Não	O número máximo de ListFHIRImportJobs solicitações que você pode fazer por segundo por conta.
Taxa de ListTagsforResource solicitaç ões por conta	Cada região com suporte: 10	Não	O número máximo de ListTagsforResource solicitações que você pode fazer por segundo por conta.
Taxa de SearchEverything solicitações por conta	Cada região com suporte: 10	Sim	O número máximo de SearchEverything solicitações que você pode fazer por segundo por conta.
Taxa de SearchEverything solicitações por armazenamento de dados	Cada região com suporte: 10	Sim	O número máximo de SearchEverything solicitações que você pode fazer por segundo por armazenamento de dados.

Nome	Padrão	Ajustá	Description
Taxa de StartFHIRBulkDeleteJob solicitações por conta	Cada região com suporte: 10	Sim	O número máximo de StartFHIRBulkDeleteJob solicitações que você pode fazer por segundo por conta.
Taxa de StartFHIRBulkDeleteJob solicitações por armazenamento de dados	Cada região com suporte: 10	Sim	O número máximo de StartFHIRBulkDeleteJob solicitações que você pode fazer por segundo por armazenamento de dados.
Taxa de StartFHIRBulkMembe rMatchJob solicitações por conta	Cada região com suporte: 10	Sim	O número máximo de StartFHIRBulkMembe rMatchJob solicitações que você pode fazer por segundo por conta.
Taxa de StartFHIRBulkMembe rMatchJob solicitações por armazenam ento de dados	Cada região com suporte: 10	Sim	O número máximo de StartFHIRBulkMembe rMatchJob solicitações que você pode fazer por segundo por armazenam ento de dados.
Taxa de StartFHIRExportJob solicitaç ões por conta	Cada região compatível: 1	Não	O número máximo de StartFHIRExportJob solicitações que você pode fazer por segundo por conta.

Nome	Padrão	Ajustá	Description
Taxa de StartFHIRExportJob solicitações usando GET por conta	Cada região compatível: 1	Não	O número máximo de StartFHIRExportJob solicitações usando GET que você pode fazer por segundo por conta.
Taxa de StartFHIRExportJob solicitações usando POST por conta	Cada região compatível: 1	Não	O número máximo de StartFHIRExportJob solicitações usando POST que você pode fazer por segundo por conta.
Taxa de StartFHIRImportJob solicitações por conta	Cada região compatível: 25	Não	O número máximo de StartFHIRImportJob solicitações que você pode fazer por segundo por conta.
Taxa de TagResource solicitações por conta	Cada região com suporte: 10	Não	O número máximo de TagResource solicitações que você pode fazer por segundo por conta.
Taxa de UntagResource solicitações por conta	Cada região com suporte: 10	Não	O número máximo de UntagResource solicitações que você pode fazer por segundo por conta.

Nome	Padrão	Ajustá	Description
Taxa de ValidateResource solicitações por conta	Cada região com suporte: 2.000	Sim	O número máximo de ValidateResource solicitações que você pode fazer por segundo por conta. Os armazenamentos de dados criados antes de 8/21 /2023 serão limitados a 300 solicitações por segundo.
Taxa de ValidateResource solicitações por armazenamento de dados	Cada região com suporte: 1.000	Sim	O número máximo de ValidateResource solicitações que você pode fazer por segundo por armazenamento de dados. Os armazenamentos de dados criados antes de 8/21 /2023 serão limitados a 300 solicitações por segundo.
Taxa de solicitações de WRITE por conta	Cada região compatível: 6.000	Sim	Número máximo de solicitações de CREATE|UPDATE|DELETE que você pode fazer por segundo, por conta.

Nome	Padrão	Ajustá	Description
Taxa de solicitações de WRITE por armazenamento de dados	Cada região compatível: 3.000	Sim	Número máximo de solicitações de CREATE|UPDATE|DELETE que você pode fazer por segundo por datastore . Os armazenamentos de dados criados antes de 8/21 /2023 serão limitados a 300 solicitaç ões por segundo.
Taxa de solicitações de pesquisa usando GET por conta	Cada região compatível: 200	Sim	Número máximo de solicitações de pesquisa usando GET que é possível fazer por segundo por conta.
Taxa de solicitações de pesquisa usando GET por armazenamento de dados	Cada região compatível: 100	Sim	Número máximo de solicitações de pesquisa usando GET que é possível fazer por segundo por armazenam ento de dados.
Taxa de solicitações de pesquisa usando POST por conta	Cada região compatível: 200	Sim	Número máximo de solicitações de pesquisa usando POST que é possível fazer por segundo por conta.

Nome	Padrão	Ajustá	Description
Taxa de solicitações de pesquisa usando POST por armazenamento de dados	Cada região compatível: 100	Sim	Número máximo de solicitações de pesquisa usando POST que é possível fazer por segundo por armazenamento de dados.
Tamanho do arquivo importado individual	Cada região compatível: 50 gigabytes	Não	O tamanho máximo (em GB) de um arquivo individual incluído em um StartFHIRImportJob.
Total de transações do pacote assíncrono enfileirado por armazenamento de dados	Cada região com suporte: 500	Sim	O número máximo de transações de pacotes assíncronos em fila por armazenamento de dados a qualquer momento.
Total de trabalhos de exportação em massa enfileirados por datastore	Cada região com suporte: 25	Sim	Número máximo de trabalhos de exportação em massa enfileirados por datastore em um determinado momento.
Total de trabalhos de importação em massa enfileirados por datastore	Cada região com suporte: 25	Sim	Número máximo de trabalhos de importação em massa enfileirados por datastore em um determinado momento.
Como visualizar resultados de trabalho de importação	Cada região suportada: 5.000 Gigabytes	Sim	O tamanho máximo (em GB) de todos os arquivos incluídos na tarefa de importação.

Tipos de dados pré-carregados do Synthea para HealthLake

HealthLake suporta somente SYNTHEA como um tipo de dados pré-carregado. O Synthea é um gerador sintético de pacientes que modela o histórico `Patient` médico. Ele está hospedado em um repositório Git de código aberto que HealthLake permite gerar um `Bundle` recurso compatível com FHIR R4 para que os usuários possam testar modelos sem usar dados reais do paciente.

Os seguintes tipos de recursos estão disponíveis em armazenamentos de HealthLake dados pré-carregados. Para obter mais informações sobre o pré-carregamento de armazenamentos HealthLake de dados com dados Synthea, consulte. Criando um armazenamento HealthLake de dados

> **Note**
>
> Para ver uma lista completa dos recursos HealthLake compatíveis com o FHIR R4, consulte. Tipos de recursos compatíveis com FHIR R4 para HealthLake

Tipos de recursos Synthea FHIR suportados por HealthLake

AllergyIntolerance	Local
CarePlan	MedicationAdministration
CareTeam	MedicationRequest
Reivindicar	Observação
Condição	Organização
Dispositivo	Paciente
DiagnosticReport	Praticante
Encontro	PractitionerRole
ExplanationofBenefit	Procedimento
ImagingStudy	Proveniência
Imunização	

AWS HealthLake projetos de amostra

Para aprofundar sua análise, você pode usar HealthLake com outros AWS serviços, conforme demonstrado nos exemplos de postagens de blog a seguir.

HealthLake análise integrada

- Aplicativos de saúde populacional com AWS HealthLake — Parte 1: Análise e monitoramento usando o Amazon Quick.
- Criação de modelos preditivos de doenças usando Amazon SageMaker AI com dados AWS HealthLake normalizados.
- Crie uma pesquisa cognitiva e um gráfico de conhecimento de saúde usando serviços de AWS IA.

HealthLake monitoramento de eventos

- EventBridge Integração da Amazon com AWS HealthLake.

Solução de problemas AWS HealthLake

Os tópicos a seguir fornecem dicas de solução de problemas para erros e problemas que você pode encontrar ao usar o HealthLake console AWS CLI AWS SDKs, ou. Se você encontrar um problema que não esteja listado nesta seção, use o botão Fornecer feedback na barra lateral direita desta página para denunciá-lo.

Tópicos

- Ações do armazenamento de dados
- Importar ações
- FHIR APIs
- Integrações de PNL
- Integrações SQL

Ações do armazenamento de dados

Problema: quando tento criar um armazenamento de HealthLake dados, recebo o seguinte erro:

```
AccessDeniedException: Insufficient Lake Formation permission(s): Required Database on
 Catalog
```

Em 14 de novembro de 2022, HealthLake atualizei as permissões necessárias do IAM para criar um novo armazenamento de dados. Para obter mais informações, consulte Configurar um usuário ou uma função do IAM para usar HealthLake (administrador do IAM).

Problema: Ao criar um armazenamento de HealthLake dados usando o AWS SDKs, o status de criação do armazenamento de dados retorna uma exceção ou um status desconhecido.

Atualize seu AWS SDK para a versão mais recente se suas chamadas `DescribeFHIRDatastore` ou de `ListFHIRDatastores` API retornarem uma exceção ou um status desconhecido do armazenamento de dados.

Importar ações

Problema: ainda posso usar HealthLake se meus dados não estiverem no formato FHIR R4?

Somente dados formatados em FHIR R4 podem ser importados para um armazenamento de dados. HealthLake Para obter uma lista de parceiros que podem ajudar a transformar os dados de saúde existentes no formato FHIR R4, consulte Parceiros.AWS HealthLake

Problema: Por que meu trabalho de importação do FHIR falhou?

Um trabalho de importação bem-sucedido gerará uma pasta com resultados (log de saída) no `.ndjson` formato, no entanto, registros individuais podem falhar na importação. Quando isso acontecer, uma segunda `FAILURE` pasta será gerada com um manifesto de registros que falharam na importação. Para obter mais informações, consulte Importando dados FHIR com AWS HealthLake.

Para analisar por que um trabalho de importação falhou, use a `DescribeFHIRImportJob` API para analisar JobProperties o. O seguinte é recomendado:

- Se o status for `FAILED` e uma mensagem estiver presente, as falhas estão relacionadas a parâmetros do trabalho, como tamanho dos dados de entrada ou número de arquivos de entrada, que estão além das HealthLake cotas.
- Se o status do trabalho de importação for`COMPLETED_WITH_ERRORS`, verifique o arquivo de manifesto,`manifest.json`, para obter informações sobre quais arquivos não foram importados com êxito.

- Se o status do trabalho de importação for `FAILED` e uma mensagem não estiver presente, vá até o local de saída do trabalho para acessar o arquivo de manifesto,`manifest.json`.

Para cada arquivo de entrada, há um arquivo de saída de falha com o nome do arquivo de entrada para qualquer recurso que falhe na importação. As respostas contêm o número da linha (lineID) correspondente à localização dos dados de entrada, objeto de resposta FHIR (UpdateResourceResponse) e código de status (statusCode) da resposta.

Um exemplo de arquivo de saída pode ser semelhante ao seguinte:


```
{"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}
```


O exemplo acima mostra que houve falhas nas linhas 3, 4, 7, 9, 15 das linhas de entrada correspondentes do arquivo de entrada. Para cada uma dessas linhas, as explicações são as seguintes:

- Na linha 3, a resposta explica que o `resourceType` fornecido na linha 3 do arquivo de entrada não é válido.
- Na linha 5, a resposta explica que há um erro de validação FHIR na linha 5 do arquivo de entrada.
- Na linha 7, a resposta explica que há um problema de validação com o `resourceId` fornecimento como entrada.

- Na linha 9, a resposta explica que o arquivo de entrada deve conter um ID de recurso válido.
- Na linha 15, a resposta do arquivo de entrada é que o arquivo não está em um formato JSON válido.

FHIR APIs

Problema: Como faço para implementar a autorização para o FHIR RESTful APIs?

Determine o Estratégia de autorização do armazenamento de dados a ser usado.

Para criar a autorização SigV4 usando o AWS SDK para Python (Boto3), crie um script semelhante ao exemplo a seguir.

```
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())
```

Problema: Por que estou recebendo *`AccessDenied`* erros ao usar o FHIR RESTful APIs para um armazenamento de dados criptografado com uma chave KMS gerenciada pelo cliente?

As permissões para chaves gerenciadas pelo cliente e políticas do IAM são necessárias para que um usuário ou função acesse um armazenamento de dados. Um usuário deve ter as permissões necessárias do IAM para usar uma chave gerenciada pelo cliente. Se um usuário revogou ou retirou uma concessão que dava HealthLake permissão para usar a chave KMS gerenciada pelo cliente, HealthLake retornará um `AccessDenied` erro.

HealthLake deve ter a permissão para acessar os dados do cliente, criptografar novos recursos do FHIR importados para um armazenamento de dados e descriptografar os recursos do FHIR quando solicitados. Para obter mais informações, consulte Solução de problemas de AWS KMS permissões.

Problema: uma operação *`POST`* da API FHIR HealthLake usando um documento de 10 MB está retornando o erro. `413 Request Entity Too Large`

AWS HealthLake tem um limite síncrono de criação e atualização da API de 5 MB para evitar maiores latências e tempos limite. Você pode ingerir documentos grandes, de até 164 MB, usando o tipo de `Binary` recurso usando a API de importação em massa.

Integrações de PNL

Problema: Como faço para ativar HealthLake o recurso integrado de processamento de linguagem natural do?

Em 14 de novembro de 2022, o comportamento padrão dos armazenamentos de HealthLake dados mudou.

Armazenamentos de dados atuais: todos os armazenamentos de HealthLake dados atuais deixarão de usar o processamento de linguagem natural (NLP) em recursos codificados em base64`DocumentReference`. Isso significa que novos `DocumentReference` recursos não serão analisados usando a PNL e nenhum novo recurso será gerado com base no texto do tipo de `DocumentReference` recurso. Para `DocumentReference` os recursos existentes, os dados e recursos gerados via PNL permanecem, mas não serão atualizados após 20 de fevereiro de 2023.

Novos armazenamentos de HealthLake dados: os armazenamentos de dados criados após 20 de fevereiro de 2023 não executarão processamento de linguagem natural (NLP) em recursos codificados em base64`DocumentReference`.

Para ativar a integração HealthLake da PNL, crie um caso de suporte usando AWS Support Center Console. Para criar seu caso, faça login no seu Conta da AWS e escolha Criar caso. Para saber mais sobre a criação de um caso e o gerenciamento de casos, consulte Criação de casos de suporte e gerenciamento de casos no Guia Suporte do usuário.

Problema: >Como faço para encontrar *`DocumentReference`* recursos que não puderam ser processados pela PNL integrada?

Se um `DocumentReference` recurso não for válido, HealthLake fornece uma extensão indicando um erro de validação em vez de fornecê-la na saída médica integrada de PNL. Para encontrar **`DocumentReference`** recursos que levaram a um erro de validação durante o processamento da PNL, você pode usar a **`search`** função FHIR com a chave HealthLake de pesquisa cm-decoration-statuse o valor de pesquisa VALIDATION_ERROR. Essa pesquisa listará todos os `DocumentReference` recursos que levaram a erros de validação, junto com uma mensagem de erro descrevendo a natureza do erro. A estrutura do campo de extensão nesses `DocumentReference` recursos com erros de validação será semelhante ao exemplo a seguir.

```
"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**
>
> A também `VALIDATION_ERROR` pode ocorrer se a decoração de PNL criar mais de 10.000 objetos aninhados. Quando isso acontece, o documento deve ser dividido em documentos menores antes do processamento.

Integrações SQL

Problema: Por que recebo um Lake Formation *`permissions error: lakeformation:PutDataLakeSettings`* ao adicionar um novo administrador de data lake?

Se seu usuário ou função do IAM contiver a política `AWSLakeFormationDataAdmin` AWS gerenciada, você não poderá adicionar novos administradores de data lake. Você receberá um erro contendo o seguinte:

```
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
```

A política AWS gerenciada `AdministratorAccess` é necessária para adicionar um usuário ou uma função do IAM como administrador de data lake do AWS Lake Formation. Se seu usuário ou função do IAM também contiver`AWSLakeFormationDataAdmin`, a ação falhará. A política `AWSLakeFormationDataAdmin` AWS gerenciada contém uma negação explícita da operação da API AWS Lake Formation,`PutDataLakeSetting`. Até mesmo administradores com acesso total ao AWS uso da política `AdministratorAccess` gerenciada podem ser limitados pela `AWSLakeFormationDataAdmin` política.

Problema: Como faço para migrar um HealthLake datastore existente para usar a integração do Amazon Athena SQL?

HealthLake os armazenamentos de dados criados antes de 14 de novembro de 2022 são funcionais, mas não podem ser consultados no Athena usando SQL. Para consultar um armazenamento de dados preexistente com o Athena, você deve primeiro migrá-lo para um novo armazenamento de dados.

Para migrar seus HealthLake dados para um novo armazenamento de dados

1. Crie um novo armazenamento de dados.
2. Exporte os dados do pré-existente para um bucket do Amazon S3.
3. Importe os dados para o novo armazenamento de dados do bucket do Amazon S3.

> **Note**
>
> A exportação de dados para um bucket do Amazon S3 incorre em uma taxa extra. A taxa extra depende do tamanho dos dados que você exporta.

Problema: ao criar um novo armazenamento de HealthLake dados para integração com SQL, o status do armazenamento de dados não muda de*`Creating`*.

Se você tentar criar um novo armazenamento de HealthLake dados e o status do armazenamento de dados não mudar de Criando, você precisará atualizar o Athena para usar o. AWS Glue Data Catalog Para obter mais informações, consulte Atualização para o catálogo de dados AWS Glue step-by-step no Guia do usuário do Amazon Athena.

Depois de atualizar com sucesso o AWS Glue Data Catalog, você pode criar um armazenamento de HealthLake dados.

Para remover um armazenamento de HealthLake dados antigo, crie um caso de suporte usando AWS Support Center Console. Para criar seu caso, faça login no seu Conta da AWS e escolha Criar caso. Para saber mais, consulte Criação de casos de suporte e gerenciamento de casos no Guia Suporte do usuário.

Problema: o console do Athena não está funcionando depois de importar dados para um novo armazenamento de dados HealthLake

Depois de importar dados para um novo armazenamento de HealthLake dados, os dados podem não estar disponíveis para uso imediato. Isso é para dar tempo para que os dados sejam ingeridos nas tabelas do Apache Iceberg. Tente novamente mais tarde.

Problema: Como faço para conectar os resultados da pesquisa no Athena a outros AWS serviços?

Ao compartilhar seus resultados de pesquisa do Athena com outros AWS serviços, podem ocorrer problemas quando você usa `json_extract[1]` como parte de uma consulta de pesquisa SQL. Para corrigir esse problema, você deve atualizar para `CATVAR` o.

Você pode encontrar esse problema ao tentar criar resultados salvos, uma tabela (estática) ou uma exibição (dinâmica).

Usando HealthLake com um AWS SDK

AWS kits de desenvolvimento de software (SDKs) estão disponíveis para muitas linguagens de programação populares. Cada SDK fornece uma API, exemplos de código e documentação que permitem que os desenvolvedores criem facilmente aplicações em seu idioma de preferência.

Documentação do SDK	Exemplos de código
AWS SDK para C++	AWS SDK para C++ exemplos de código

Documentação do SDK	Exemplos de código
AWS CLI	AWS CLI exemplos de código
AWS SDK para Go	AWS SDK para Go exemplos de código
AWS SDK para Java	AWS SDK para Java exemplos de código
AWS SDK para JavaScript	AWS SDK para JavaScript exemplos de código
AWS SDK para Kotlin	AWS SDK para Kotlin exemplos de código
AWS SDK para .NET	AWS SDK para .NET exemplos de código
AWS SDK para PHP	AWS SDK para PHP exemplos de código
Ferramentas da AWS para PowerShell	Ferramentas da AWS para PowerShell exemplos de código
AWS SDK para Python (Boto3)	AWS SDK para Python (Boto3) exemplos de código
AWS SDK para Ruby	AWS SDK para Ruby exemplos de código
AWS SDK para Rust	AWS SDK para Rust exemplos de código
AWS SDK para SAP ABAP	AWS SDK para SAP ABAP exemplos de código
AWS SDK para Swift	AWS SDK para Swift exemplos de código

Exemplo de disponibilidade

Não consegue encontrar o que precisa? Solicite um exemplo de código usando o link Fornecer feedback na parte inferior desta página.

AWS HealthLake lançamentos

A tabela a seguir mostra quando os recursos e atualizações foram lançados para AWS HealthLake. Para obter mais informações sobre uma versão, consulte o tópico vinculado.

Alteração	Descrição	Data
Segurança e filtragem de tags para $export e $ davinci-data-export	As `$davinci-data-export` operações `$export` e agora oferecem suporte `_security` e `_tag` consultam parâmetros para filtrar recursos exportados `meta.security` e valores de `meta.tag` codificação. Isso permite a exportação multilocatária e o controle de acesso granular usando o formato FHIR padrão. `system|code`	30 de abril de 2026
bulk-member-match operação $	AWS HealthLake agora suporta a `$bulk-member-match` operação para processar solicitações de correspondência de vários membros de forma assíncrona. Essa operação permite que as organizações de saúde combinem com eficiência centenas de identificadores exclusivos de membros em diferentes sistemas de saúde usando informações demográficas e de cobertura em uma única solicitação em massa.	1 de abril de 2026

	• Lida com até 500 membros por solicitação com até 5 operações simultâneas por armazenamento de dados • Resultados categorizados em MatchedMembers, NonMatchedMembers, e grupos ConsentConstrained Members • Integra-se com fluxos de `$davinci-data-export` trabalho de dados end-to-end em massa Para obter mais informações, consulte the section called "$bulk-member-match".	
Transações assíncronas de pacotes	AWS HealthLake agora suporta o `Bundle` tipo assíncrono`transaction` , permitindo que você envie transações com até 500 recursos. HealthLake e coloca a transação em fila para processamento e retorna imediatamente uma URL de pesquisa para verificar o status e recuperar os resultados. Para obter mais informações, consulte Transações assíncronas de pacotes.	24 de março de 2026

Patch Support em operações de pacote	HealthLake agora oferece suporte ao Bundle Patch, habilitando o JSON Patch para pacotes Transaction FHIRPath e Batch, bem como FHIRPath para a API Patch. Esse recurso permite que os clientes corrijam recursos diretamente nas operações do Bundle sem exigir a substituição total dos recursos, reduzindo o tamanho da carga útil, simplificando os fluxos de trabalho de integração e permitindo a ingestão mais rápida de dados.	20 de março de 2026
DaVinci PDex Tipos de exportação para $ davinci-data-export	A `$davinci-data-export` operação agora oferece suporte aos tipos de PDex exportação para Provider Access e Member Access APIs. Payer-to-Payer • Lógica de inclusão baseada em perfil para recursos ExplanationOfBenefit • Transformação de dados financeiros com o `_includeEOB2xWoFinancial` parâmetro • Filtro temporal de 5 anos em dados clínicos e de sinistros	20 de março de 2026

Parâmetro _until em $export & $ davinci-data-export	Parâmetro de filtragem temporal para operações de exportação	26 de fevereiro de 2026
_incluir parâmetro de pesquisa	HealthLake agora suporta include: * e include:iterate	26 de fevereiro de 2026
Endpoints de interoperabilidade do CMS	Esse recurso permite que você acompanhe o uso da API por categoria de CMS e, posteriormente, relate as métricas de uso para fins de conformidade.	26 de fevereiro de 2026
Bundle Message Type Support	Suporte limitado para recursos do pacote FHIR com tipo de mensagem	26 de fevereiro de 2026
Adicionado suporte para novos IGs	AWS HealthLake expandiu o suporte do FHIR Implementation Guides (IGs) para o CMS 0057F: • Support para CARIN Blue Button 2.0.0 e 2.1.0 • Support para Da Vinci Payer Data Exchange 2.0.0 e 2.1.0 • Support for DaVinci Payer Data Exchange (PDex) US Drug Formulary 2.0.1 e 2.1.0 • Support for Da Vinci Clinical Data Exchange (CDex) 2.1.0 • Suporte para Suporte de Autorização Prévia Da Vinci (PAS) FHIR IG 2.1.0	26 de fevereiro de 2026

Operação $submit	A `$submit` operação permite que você envie eletronic amente solicitações de autorização prévia aos pagadores para aprovação.	26 de fevereiro de 2026
Operação $questionnaire-pac kage	A `$questionnaire-pac kage` operação recupera um pacote abrangente contendo um questionário FHIR e todas as dependências necessárias para renderizar e processar o questionário.	26 de fevereiro de 2026
Operação $inquire	A `$inquire` operação permite que você verifique o status de uma solicitação de autorização prévia enviada anteriormente.	26 de fevereiro de 2026
Operação $member-remove	A operação $member-remove permite que você remova membros de uma Lista de Atribuição de Membros do FHIR (recurso de grupo) em. AWS HealthLake	12 de novembro de 2025
Operação $member-match	AWS HealthLake agora oferece suporte à operação $member-match para recursos de pacientes, permitindo que organizações de saúde encontrem o identificador exclusivo de um membro em diferentes sistemas de saúde usando informações demográfi cas e de cobertura.	12 de novembro de 2025

Operação $member-add	A operação FHIR $member-add adiciona um membro (paciente) a um recurso do Grupo, especificamente a uma Lista de Atribuição de Membros.	12 de novembro de 2025
$ davinci-data-export Operação	A davinci-data-export operação $ é uma operação FHIR assíncrona que permite a exportação dos dados da Lista de Atribuição de Membros de. AWS HealthLake	12 de novembro de 2025
$ confirm-attribution-list Operação	Indica ao Produtor que o Consumidor não tem mais alterações a fazer na Lista de Atribuições, finalizando a lista de atribuição removendo membros inativos e alterando o status para “final”.	12 de novembro de 2025
Operação $attribution-status	Recupera o status de atribuição de um membro específico, devolvendo um pacote contendo todos os recursos de atribuição relacionados ao paciente.	12 de novembro de 2025
Assinaturas FHIR	HealthLake suporta assinaturas FHIR, permitindo que você receba notificações em tempo real quando ocorrerem alterações específicas nos dados de saúde e crie fluxos de trabalho orientados por eventos.	30 de outubro de 2025

Expansão da região para Montreal, Canadá	HealthLake está disponível na região Canadá (Montreal). Para obter mais informações, consulte Endpoints de serviço.	17 de outubro de 2025

Adicionado suporte para novos IGs	AWS HealthLake expandiu o suporte dos Guias de Implementação do FHIR (IGs) para o seguinte mercado canadense: • Perfil FHIR básico do CA Core+Canadian que define os principais elementos de dados e as restrições de interoperabilidade nos sistemas de saúde canadenses. • CA:EREC Pan-Canadian eReferral - Especificação eConsultStandardized FHIR para referências e consultas eletrônicas entre profissionais de saúde em todo o Canadá. • Resumo do paciente Edição canadense (PS-CA) Adaptação canadense do Resumo Internacional do Paciente (IPS) para compartilhar informações essenciais de saúde do paciente em todos os ambientes de atendimento. • Perfil FHIR RepositoryOntario específico do Ontario Digital Health Drug para troca padronizada de dados de medicamentos e prescrições no sistema de saúde provincial.	17 de outubro de 2025

Suporte de IG específico da região	HealthLake agora oferece suporte para regiões específicas IGs. Para obter mais informações, consulte validações de perfil.	8 de outubro de 2025
Operação de patch	HealthLake permite modificar elementos específicos dos recursos do FHIR usando operações de patch JSON sem atualizar o recurso inteiro.	18 de agosto de 2025
Operação $validate	HealthLake permite validar os recursos do FHIR em relação à especificação e aos perfis sem realizar operações de armazenamento, retornando resultados de validação detalhados.	18 de agosto de 2025
Operação $purge	HealthLake inclui a funcionalidade para excluir permanentemente todos os recursos do compartimento de um paciente do armazenamento de dados.	18 de agosto de 2025
Operação $lookup	HealthLake fornece a capacidade de recuperar informações detalhadas sobre um conceito específico em um CodeSystem fornecendo o código e o identificador do sistema.	18 de agosto de 2025

Operação $expand	HealthLake agora permite expandir ValueSet os recursos para recuperar a lista completa de códigos contidos nos códigos ingeridos pelo cliente ValueSets.	18 de agosto de 2025
Operação $erase	HealthLake agora oferece a exclusão permanente de um recurso específico e de todas as suas versões históricas do armazenamento de dados.	18 de agosto de 2025
Operação $document	HealthLake suporta a geração de documentos clínicos completos agrupando um recurso de composição com todos os seus recursos referenciados em um único pacote de documentos.	18 de agosto de 2025
:abaixo do modificador de pesquisa	HealthLake introduz a busca por valores de URI que estão hierarquicamente abaixo do URI especificado em um sistema terminológico.	8 de agosto de 2025

Exclusão condicional	HealthLake agora oferece suporte à exclusão condicional do FHIR, permitindo que as organizações de saúde excluam um recurso existente com base nos critérios de pesquisa e não pelo ID lógico do FHIR. Consulte Excluindo recursos do FHIR com base nas condições para obter mais informações.	7 de julho de 2025
Adicionado suporte para novos IGs	AWS HealthLake expandiu o suporte dos Guias de Implementação do FHIR (IGs) para o seguinte: • US Core 7.0.0, que especifica como usar o FHIR para implementar o padrão USCDI 4.0 • Guia de implementação do UK Core 2.0.1 para fornecer orientação de implementação de FHIR em todo o Reino Unido	7 de julho de 2025
Expansão da região para Dublin, Irlanda	HealthLake está disponível na região da UE (Dublin). Para obter mais informações, consulte Endpoints de serviço.	9 de junho de 2025

Transações do tipo pacote	HealthLake agora suporta o tipo de pacote FHIR 'Transaction', permitindo que organizações de saúde enviem vários recursos como uma única operação atômica. Isso permite fluxos de trabalho de integração e troca de dados mais eficientes. Por exemplo, um profissional de saúde agora pode atualizar o prontuário do paciente, a lista de medicamentos e a consulta em uma única transação, reduzindo a complexidade e os possíveis erros. Consulte Agrupando recursos do FHIR para obter mais informações.	28 de abril de 2025

Adicionado suporte para novos IGs	AWS HealthLake HealthLak e A AWS expandiu o suporte dos Guias de Implementação do FHIR (IGs) para o seguinte: • Guia de implementação do NCQA HEDIS® (0.3.1): Suporta medições e relatórios de qualidade para o Conjunto de Dados e Informações de Eficácia em Saúde (HEDIS). • Resumo Internacional do Paciente (IPS) (2.0.0): Permite a troca de informações essenciais de saúde para apoiar a continuidade dos cuidados aos pacientes. • Medida de qualidade (5.0.0): suporta a representação e a troca de definições e dados de medidas de qualidade. • Relatórios genômicos (3.0.0): facilita a troca de dados e relatórios genômicos.	28 de abril de 2025

Chaves de idempotência	HealthLake agora suporta chaves de idempotência para operações FHIR POST, fornecendo um mecanismo robusto para garantir a integridade dos dados durante a criação de recursos. Consulte Idempotência e simultaneidade para obter mais informações.	18 de abril de 2025
Consistência do histórico do FHIR	HealthLake agora oferece suporte a uma forte consistência para armazenamentos de dados habilitados para histórico por meio do novo `x-amz-fhir-history-consistency-level` cabeçalho. Quando definidos como “fortes”, os resultados da pesquisa FHIR incluem todos os registros indexados, independentemente do status da atualização. Consulte Níveis de consistência de pesquisa do FHIR para obter mais informações.	18 de abril de 2025

Etag e 'if-match'	HealthLake agora fornece suporte para ETag, permitindo que os clientes usem o cabeçalho 'If-Match' para garantir atualizações idempotentes. Isso ajuda a manter a integridade dos dados evitando substituições acidentais durante atualizações simultâneas. Isso é particularmente valioso em ambientes de saúde de alto volume, nos quais vários sistemas podem tentar atualizar o mesmo registro simultaneamente. Consulte ETag na AWS HealthLake para obter mais informações.	18 de abril de 2025

Condicional PUTs em pacotes	HealthLake agora oferece suporte a atualizações condicionais para pacotes FHIR, oferecendo às organizações de saúde mais flexibilidade para gerenciar e atualizar seus dados. Agora, os clientes podem especificar critérios para criar, atualizar ou excluir recursos condicion almente como parte de uma transação de pacote. Isso simplifica os processos de sincronização de dados entre sistemas e reduz a necessida de de uma lógica complexa do lado do cliente. Consulte Condicional PUTs em pacotes para obter mais informações.	18 de abril de 2025

SMART em escopos FHIR V2	HealthLake suporta SMART nos escopos do FHIR V2 para criar, ler, atualizar, excluir e pesquisar recursos do FHIR. Para obter mais informaçõ es, consulte SMART sobre os escopos de recursos do FHIR para. HealthLake • Os escopos SMART on FHIR V2 estão disponíveis para todos os armazenam entos de HealthLake dados criados após 22/01/2025. Se seu armazenamento de dados foi criado antes dessa data, você pode enviar um ticket de suporte para habilitar o SMART nos escopos do FHIR V2. Crie um caso usando AWS Support Center Consoleo. Para criar seu caso, faça login no seu Conta da AWS e escolha Criar caso.	22 de janeiro de 2025
FHIR US Core Profile, versão 6.1.0	HealthLake suporta a versão 6.1.0 do FHIR US Core Profile. Para obter mais informações, consulte Validações de perfil FHIR para. HealthLake	22 de janeiro de 2025
FHIR $export com GET	HealthLake suporta FHIR com. `$export` `GET` Para obter mais informações, consulte Exportação de HealthLake dados com o `$export` FHIR.	22 de janeiro de 2025

Guia do desenvolvedor refatorado com exemplos de códigos testados	HealthLake apresenta um Guia do Desenvolvedor reformulado com exemplos de códigos testados para ações nativas AWS CLI e do SDK. AWS Além disso, os procedimentos agora estão disponíveis para todas as interações da API FHIR suportadas. Para obter mais informações, consulte Exemplos de código e Gerenciamento de recursos do FHIR.	18 de dezembro de 2024

FHIR `history` e interações `vread`	HealthLake suporta a `history` interação FHIR para recuperar o histórico de um recurso específico e a `vread` interação para realizar uma leitura específica da versão de um recurso. Para obter mais informações, consulte Lendo o histórico de recursos do FHIR. • O recurso FHIR `history` é ativado por padrão para todos os armazenamentos de HealthLake dados criados após 25/10/2024. Se seu armazenamento de dados foi criado antes dessa data, você pode enviar um ticket de suporte para que a `history` interação com o FHIR seja ativada. Crie um caso usando AWS Support Center Console. Para criar seu caso, faça login no seu Conta da AWS e escolha Criar caso.	25 de outubro de 2024

Operação FHIR `Patient/$everything`	HealthLake suporta a `Patient/$everything` operação FHIR para pesquisar um `Patient` recurso e todos os recursos relacionados. Usando essa operação, você pode acessar o prontuário completo de um paciente ou baixar `Patient` dados em massa. Para obter mais informações, consulte Obter dados do paciente com `Patient/$everything`. • O FHIR `Patient/$everything` está habilitado por padrão para todos os armazenamentos de HealthLake dados criados após 27/02/2024. Se seu armazenamento de dados foi criado antes dessa data, você pode enviar um ticket de suporte para que a `Patient/$everything` operação seja ativada. Crie um caso usando AWS Support Center Console. Para criar seu caso, faça login no seu Conta da AWS e escolha Criar caso.	27 de fevereiro de 2024

Recurso FHIR `VerificationResult`	HealthLake suporta o tipo de `VerificationResult` recurso FHIR para descrever requisitos de validação, fontes, status e datas para um ou mais elementos. Para obter mais informações, consulte Tipos de recursos do FHIR R4 para. HealthLake	9 de dezembro de 2023

Operação FHIR `$export`	HealthLake suporta a `$export` operação FHIR para exportar dados de saúde em massa de um HealthLake armazenamento de dados. Para obter mais informaçõ es, consulte Exportação de HealthLake dados com o `$export` FHIR. • O FHIR `$export` está ativado por padrão em todos os armazenamentos HealthLake de dados criados após 1º de junho de 2023. Se seu armazenam ento de dados foi criado antes dessa data, você pode enviar um ticket de suporte para que a `$export` operação seja ativada. Crie um caso usando AWS Support Center Consoleo. Para criar seu caso, faça login no seu Conta da AWS e escolha Criar caso. • HealthLake os armazenam entos de dados criados antes de 01/06/23 oferecem suporte somente a solicitaç ões de `$export` trabalho para exportações em todo o sistema. • HealthLake os armazenam entos de dados criados	1º de junho de 2023

	antes de 01/06/23 não suportam a obtenção do status de um FHIR `$export` usando uma `GET` solicitação no endpoint do armazenamento de dados.	
SMART no suporte FHIR	HealthLake adiciona suporte para SMART na autorizaç ão FHIR. Para obter mais informações, consulte SMART no suporte do FHIR para. AWS HealthLake	31 de maio de 2023
Validações do perfil FHIR	HealthLake suporta validaçõe s de perfil FHIR para definir definições específicas de tipos de recursos usando and/or extensões de restrições nos tipos de recursos básicos. Para obter mais informações, consulte Validações de perfil.	31 de maio de 2023
Região Ásia-Pacífico (Mumbai)	HealthLake está disponível na região Ásia-Pacífico (Mumbai). Para obter mais informações, consulte Endpoints de serviço.	4 de abril de 2023

O processamento de linguagem natural está desativado por padrão	HealthLake desativou o processamento integrado de linguagem natural (NLP) em todos os armazenamentos de dados a partir de 20 de fevereiro de 2023. Você pode enviar um ticket de suporte para ativar a funcionalidade de PNL integrada. Crie um caso usando AWS Support Center Consoleo. Para criar seu caso, faça login no seu Conta da AWS e escolha Criar caso. Para saber mais sobre a PNL integrada, consulte Integrando a PNL com. HealthLake •	20 de fevereiro de 2023

Índice e consulta SQL com o Amazon Athena	HealthLake suporta a consulta de dados FHIR com SQL usando o Amazon Athena. Para obter mais informaçõ es, consulte Consulta de HealthLake dados com o Amazon Athena. • A funcionalidade de consulta SQL está habilitad a por padrão para todos os armazenamentos HealthLake de dados criados após 14/11/2022. Se seu armazenamento de dados foi criado antes dessa data, você pode enviar um ticket de suporte para habilitar a funcionalidade de consulta SQL. Crie um caso usando AWS Support Center Consoleo. Para criar seu caso, faça login no seu Conta da AWS e escolha Criar caso. • Com a funcionalidade de consulta SQL, as configura ções do IAM a serem acessadas HealthLake devem ser atualizadas. Para criar armazenamentos de HealthLake dados e conceder acesso a eles no Athena, você deve ter a política `AWSLakeFo rmationDataAdmin`	14 de novembro de 2022

	gerenciada adicionada ao seu usuário, grupo ou função do IAM. Você pode usar a `AWSLakeFormationDataAdmin` política para criar administradores de data lake e conceder acesso aos armazenamentos de dados no Athena. Para obter mais informações, consulte Configurar um usuário ou uma função do IAM.	
O tamanho total do trabalho de importação aumentou	HealthLake atualiza o `Total import job size` para uma `StartFHIRImportJob` solicitação para 500 GB. Para obter mais informações, consulte Cotas de serviço.	3 de outubro de 2022
Recurso FHIR `Bundle`	HealthLake suporta o tipo de `Bundle` recurso FHIR para processar vários recursos FHIR simultaneamente. Para obter mais informações, consulte Agrupando recursos do FHIR.	5 de agosto de 2022
Atualizações de cotas para interações com o FHIR	HealthLake atualiza as cotas para interações de gerenciamento de recursos do FHIR. Para obter mais informações, consulte Cotas de serviço.	16 de julho de 2022

Parâmetro de pesquisa FHIR `_include`	HealthLake adiciona suporte ao parâmetro de `_include` pesquisa FHIR para retornar recursos adicionais em uma `search` solicitação. Para obter mais informações, consulte Parâmetros de pesquisa avançada.	16 de julho de 2022
AWS HealthLake está geralmente disponível	HealthLake geralmente está disponível em todas as regiões com suporte. Para obter mais informações, consulte Endpoints de serviço.	15 de julho de 2021

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.