aws

开发人员指南

AWS HealthLake

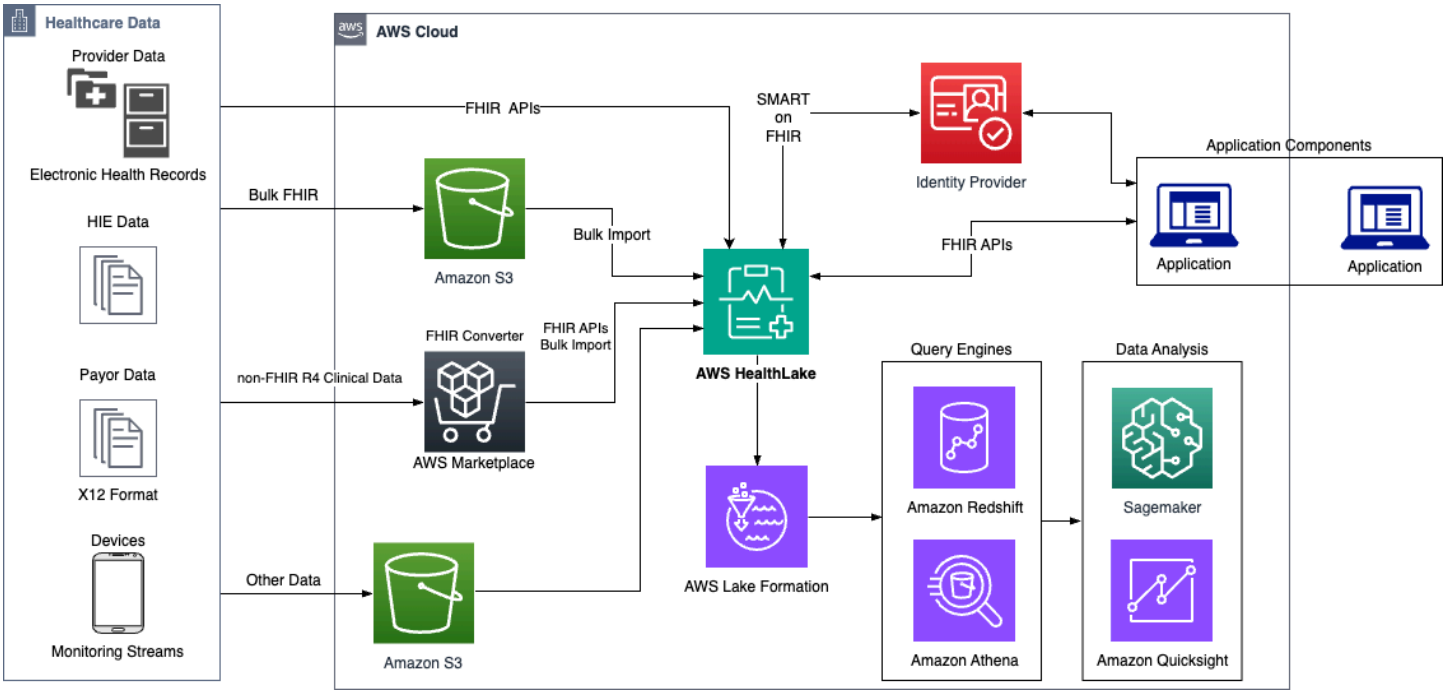
AWS HealthLake: 开发人员指南

Table of Contents

什么是 AWS HealthLake？

AWS HealthLake 是一项符合 HIPAA 资格的服务，可使用快速医疗互操作性资源 (FHIR) R4 规范在云中存储、分析和共享健康数据。 HealthLake 用例包括：

- 企业健康数据 — 直接从中管理和共享 FHIR R4 运行状况数据， AWS Cloud 同时保持高性能和可用性。
- 医疗互操作性 — 通过完全托管的 FHIR 数据存储支持客户遵守 21 世纪治疗法案，允许患者访问。
- 自然语言处理 (NLP)-利用集成的自然语言处理模型从非结构化健康数据中提取有意义的医疗信息。
- 多模态分析 — 将 HealthLake 数据与数据和 AWS HealthImaging AWS HealthOmics 数据相结合，为精准医疗提供见解。



主题

- 定价

重要提示

AWS HealthLake 不能替代专业医疗建议、诊断或治疗，也不能用于治愈、治疗、缓解、预防或诊断任何疾病或健康状况。您有责任将人工审查作为任何使用的一部分 AWS HealthLake，包括与任何旨在为临床决策提供依据的第三方产品相关使用。 AWS HealthLake 只有经过培训的医疗专业人员根据合理的医学判断进行审查后，才应在患者护理或临床场景中使用。

的特点 AWS HealthLake

AWS HealthLake 提供以下功能。

导入 FHIR R4 的健康数据

通过 HealthLake 原生导入操作，您可以轻松地将 FHIR 数据从 Amazon S3 存储桶迁移到 HealthLake 数据存储，包括临床记录、实验室报告、保险索赔等。 HealthLake 支持用于医疗保健数据交换的 FHIR R4 规范。如果需要，您可以与AWS HealthLake 合作伙伴合作，将您的健康数据转换为 FHIR R4 格式。

以安全、合规且可审计的方式存储健康数据

HealthLake 数据存储有助于为健康数据编制索引，以便对其进行查询。数据存储按时间顺序创建每位患者的病史的完整视图，并使用FHIR R4规范促进信息交换。而且它始终在运行以使您的索引保持最新，使您能够使用标准 FHIR R4 交互以及耐用的主存储和索引缩放随时搜索信息。

利用事务性 FHIR 服务器

利用 FHIR APIs 进行标准资源验证、SMART on FHIR 授权以及批量数据 FHIR API 导出功能，支持统一和分析数据，从而降低运营成本并改善决策。 HealthLake 支持客户遵守最新的 ONC 和 CMS 监管标准，包括： HL7 FHIR R4 APIs、FHIR 批量数据访问、美国核心 IG STU、SM HL7 ART App Launch Framework IG、2.0 OAuth 和 OpenID Connect。

使用 NLP 转换非结构化医疗数据

集成医学自然语言处理 (NLP) 可转换数据存储中的所有原始医学文本数据，以从非结构化医疗保健 HealthLake 数据中理解和提取有意义的信息。借助集成的医学 NLP，您可以自动从医学文本中提取实体、实体关系、实体特征和受保护的健康信息 (PHI)。NLP 提取的实体作为原生 FHIR R4 资源存储在 HealthLake 数据存储中，可以通过 FHIR R4 或 APIs Amazon Athena (SQL) 进行访问。

相关 AWS 服务

AWS HealthLake 与其他 AWS 服务紧密集成。了解以下服务对于充分利用很有用 HealthLake。

- AWS Identity and Access Management— 使用 IAM 安全地管理身份和对 HealthLake 资源的访问权限。
- 亚马逊简单存储服务 — 使用 Amazon S3 作为暂存区，将 DICOM 数据导入。 HealthLake
- AWS CloudTrail— CloudTrail 用于跟踪 HealthLake 用户活动和 API 使用情况。
- Amazon CloudWatch — CloudWatch 用于观察和监控 HealthLake 资源。
- AWS CloudFormation— 用于 CloudFormation 实现基础设施即代码 (IaC) 模板以在中创建资源。HealthLake
- AWS PrivateLink— 使用 Amazon VPC 在 HealthLake 和亚马逊虚拟私有云之间建立连接，而无需将数据暴露给互联网。
- Amazon EventBridge — 通过创建 EventBridge 将事件路由到目标的规则，用于创建可扩展、HealthLake 事件驱动的应用程序。
- AWS Lake Formation— 使用 Lake Formation 集中管理、保护和共享用于分析和机器学习 HealthLake 的数据。
- Amazon A thena — 使用 Athena 通过 SQL HealthLake 查询数据，以便进行更深入的分析。

正在访问 AWS HealthLake

您可以 AWS HealthLake 使用 AWS 管理控制台、 AWS Command Line Interface 和 AWS SDKs。本指南提供了程序说明 AWS 管理控制台 以及 AWS CLI 和的代码示例 AWS SDKs。

AWS Command Line Interface (AWS CLI)

AWS CLI 提供了适用于各种 AWS 产品的命令，并在 Windows、Mac 和 Linux 上受支持。有关更多信息，请参阅 AWS Command Line Interface 《用户指南》。

AWS SDKs

AWS SDKs 为软件开发人员提供库、代码示例和其他资源。这些库文件提供可自动执行任务的基本功能，例如以加密方式对请求签名、重试请求和处理错误响应。有关更多信息，请参阅构建工具 AWS。

AWS 管理控制台

AWS 管理控制台 提供了一个基于 Web 的用户界面，用于管理 HealthLake 及其相关资源。如果您已注册 AWS 帐户，则可以登录HealthLake 控制台。

HIPAA 资格和数据安全

这是一项符合 HIPAA 要求的服务。有关 AWS《1996 年美国健康保险流通与责任法案》(HIPAA) 以及使用 AWS 服务处理、存储和传输受保护的健康信息 (PHI) 的更多信息，请参阅 HIPAA 概述。

必须对与 HealthLake 包含 PHI 和个人身份信息 (PII) 的连接进行加密。默认情况下，所有连接都 HealthLake 使用通过 TLS 的 HTTPS。 HealthLake 存储加密的客户内容，并按照责任AWS 共担模式运营。

定价

有关 HealthLake 定价信息，请参阅AWS HealthLake 定价。要估算成本，请使用定 HealthLake 价计算器。

入门 AWS HealthLake

要开始使用 AWS HealthLake，请设置一个 AWS 帐户并创建一个 AWS Identity and Access Management 用户。要使用AWS CLI或 AWS SDKs，必须安装和配置它们。

> **Note**
>
> 本指南的参考章节提供了 SMART on FHIR、FHIR R4 和。 AWS HealthLake例如，您可以在 FHIR 配置、支持的 FHIR 配置文件验证和端点上找到有关 SMART 的信息。 HealthLake

在学习了 HealthLake 概念和设置之后，我们提供了一个包含代码示例的简短教程来帮助您入门。

主题

- AWS HealthLake 概念
- 设置 AWS HealthLake
- AWS HealthLake 教程

AWS HealthLake 概念

以下术语和概念对您了解和使用 AWS HealthLake非常有用。

概念

- 数据存储授权策略
- 集成自然语言处理
- 集成分析

数据存储授权策略

HealthLake 数据存储是 FHIR R4 运行状况数据的存储库，位于单个存储库中。 AWS 区域 HealthLake 支持以下数据存储授权策略。

- Sigv4 授权 — 使用AWS 签名版本 4 (Sigv4) HealthLake 授权授权 FHIR API 调用。
- SMART on FHIR 授权 — 在 FHIR HealthLake 授权上使用可替代医疗应用程序和可重复使用技术 (SMART) 授权 FHIR API 调用。

有关更多信息，请参阅 创建 HealthLake 数据存储。

集成自然语言处理

AWS HealthLake 与 HIPAA 合格的自然语言处理 (NLP) 库集成，可从非结构化医学文本中提取有意义的健康数据。自然语言处理库可以识别医疗实体，例如病症、药物、剂量、测试、治疗和程序。它们识别实体之间的关系，并将它们链接到医学本体库，例如 ICD-10-CM 和。 RxNorm有关更多信息，请参阅 集成的自然语言处理 (NLP) 用于 HealthLake。

集成分析

AWS HealthLake 超越了 FHIR`search`，它`bundle` APIs 为查询和分析大量健康数据提供了集成分析。在导入过程中， HealthLake 自动生成用于 SQL 索引和查询的表。这使您无需进行大量的数据工程工作，即可从复杂的医疗保健数据中获得切实可行的见解。有关更多信息，请参阅使用 Amazon Athena 查询 HealthLake 数据和AWS HealthLake 示例项目。

设置 AWS HealthLake

在本章中，您将使用 AWS 管理控制台 来设置开始使用 AWS HealthLake 和创建数据存储所需的权限。要设置创建数据存储的权限，您需要创建一个既是数据湖管理员又 HealthLake 是管理员的 IAM 用户或角色。您可以在 Lake Formation 中将此用户设置为数据 AWS 湖管理员。数据湖管理员授予 Lake Formation 访问使用亚马逊 Athena 查询数据存储所需的资源的权限。创建 HealthLake数据存储后，您可以设置导入和导出文件的权限。

主题

- 注册获取 AWS 账户
- 配置要使用的 IAM 用户或角色 HealthLake （ IAM 管理员 ）
- 在 Lake Formation 中添加用户或角色作为数据湖管理员（ IAM 管理员 ）
- 创建 S3 存储桶
- 创建数据存储
- 为导入任务设置权限
- 为导出任务设置权限
- 安装 AWS CLI

注册获取 AWS 账户

要开始使用 AWS，你需要一个 AWS 账户。有关创建的信息 AWS 账户，请参阅《AWS 账户管理 参考指南》 AWS 账户中的入门指南。

配置要使用的 IAM 用户或角色 HealthLake （ IAM 管理员 ）

> **角色：IAM 管理员**
>
> 可以创建 IAM 用户和角色并可以添加数据湖管理员的用户。

本主题中的这些步骤必须由 IAM 管理员执行。

要将您的 HealthLake 数据存储连接到 Athena，您需要创建一个既是数据湖管理员又是管理员的 IAM 用户或角色。 HealthLake 此新用户或角色授予通过 AWS Lake Formation 访问数据存储中资源的权限，并将`AmazonHealthLakeFullAccess` AWS 托管策略添加到其用户或角色中。

> **Important**
>
> 身为数据湖管理员的 IAM 用户或角色无法创建新的数据湖管理员。要添加其他数据湖管理员，您必须使用已被授予`AdministratorAccess`访问权限的 IAM 用户或角色。

创建管理员

1. 将 **`AmazonHealthlakeFullAccess`** IAM AWS 托管策略添加到组织中的用户或角色。

   如果您不熟悉创建 IAM 用户，请参阅 IA M 用户指南中的创建 I AWS AM 用户和 IAM 策略概述。
2. 向 IAM 用户或角色授予访问 AWS Lake Formation 的访问权限。

   - 向组织中的用户或角色添加以下 IAM AWS 托管策略：**`AWSLakeFormationDataAdmin`**

     > **Note**
     >
     > 该`AWSLakeFormationDataAdmin`政策允许访问所有 AWS Lake Formation 资源。建议您始终使用完成任务所需的最低权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM 最佳实操。

3. 向用户或角色添加以下内联策略。有关更多信息，请参阅 IAM 用户指南中的内联策略。

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": "*"
        }
    ]
}
```

有关该AWSLakeFormationDataAdmin政策的更多信息，请参阅 Lake Formation 开发者指南中的 La AWS ke Formation 角色和 IAM 权限参考。

在 Lake Formation 中添加用户或角色作为数据湖管理员（IAM 管理员）

> **Note**
>
> 如果您要集成，则必须执行此步骤SQL 索引和查询。

接下来，IAM 管理员必须将上一步中创建的用户或角色添加为 Lake Formation 中的数据湖管理员。

将 IAM 用户或角色添加为数据湖管理员

1. 打开 AWS Lake Formation 控制台：https://console.aws.amazon.com/lakeformation/

> **Note**
>
> 如果这是你第一次访问 Lake Formation，则会出现一个 “欢迎来到 Lake Form ation” 对话框，要求你定义 Lake Formation 管理员。

2. 将新用户或角色分配为 AWS Lake Formation 数据湖管理员。
   - 选项 1：如果你收到了 “欢迎来到 Lake Formation” 对话框。
     1. 选择添加其他 AWS 用户或角色。
     2. 选择向下箭头 (▾)。
     3. 选择你想同时成为 Lake Formation 管理员的管理员。 HealthLake
     4. 选择开始。
   - 选项 2：使用导航窗格 (☰)。
     1. 选择导航窗格 (☰)。
     2. 在 “权限” 下，选择 “管理角色和任务”。
     3. 在数据湖管理员部分中，选择选择管理员。
     4. 在 “管理数据湖管理员” 对话框中，选择向下箭头 (▾)。
     5. 接下来，选择或搜索您也想成为 Lake Formation HealthLake 管理员的管理员用户或角色。
     6. 选择保存。
3. 将默认安全设置更改为由 Lake Formation 管理。 HealthLake 数据存储资源需要由 Lake Formation 管理，而不是 IAM 管理。要进行更新，请参阅 La AWS ke Formation 开发者指南中的更改默认权限模型。

创建 S3 存储桶

要将 FHIR R4 数据导入 AWS HealthLake，建议使用两个 Amazon S3 存储桶。Amazon S3 输入存储桶保存要导入和从该存储桶 HealthLake 读取的 FHIR 数据。Amazon S3 输出存储桶存储导入任务的处理结果并向该存储桶 HealthLake 写入（日志）。

> **Note**
>
> 根据 AWS Identity and Access Management (IAM) 政策，您的 Amazon S3 存储桶名称必须是唯一的。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的存储桶命名规则。

出于本指南的目的，我们在本节后面设置导入权限时指定以下 Amazon S3 输入和输出存储桶。

- 输入存储桶: `arn:aws:s3:::amzn-s3-demo-source-bucket`
- 输出桶：`arn:aws:s3:::amzn-s3-demo-logging-bucket`

有关更多信息，请参阅《Amazon S3 用户指南》中的创建存储桶。

创建数据存储

HealthLake 数据存储是驻留在单个 AWS 区域内的 FHIR R4 数据的存储库。一个 AWS 账户可以有零个或多个数据存储。 HealthLake 支持两种数据存储授权策略。

> **重要提示**
>
> 在创建 HealthLake 数据存储之前，请查看 AWS 组织中可能限制 HealthLake 资源创建或管理的服务控制策略 (SCP)。即使您的 IAM 权限设置正确，SCP 也可能阻止成功创建 HealthLake 数据存储。
>
> 创建 HealthLake 数据存储时会生成 A。`datastoreID`在本节稍后设置导入权限`datastoreID`时，必须使用。

要创建 HealthLake 数据存储，请参阅创建 HealthLake 数据存储。

为导入任务设置权限

在将文件导入数据存储之前，必须授予访问您在 Amazon S3 中的输入和输出存储桶的 HealthLake 权限。要授予 HealthLake 访问权限，您需要为创建一个 IAM 服务角色 HealthLake，向该角色添加信任策略以授予 HealthLake 代入角色权限，并向角色附加权限策略以授予其访问您的 Amazon S3 存储桶的权限。

创建导入任务时，您可以为指定该角色的 Amazon 资源名称 (ARN)。`DataAccessRoleArn`有关 IAM 角色和信任策略的更多信息，请参阅 IAM 角色。

设置权限后，您就可以通过导入任务将文件导入数据存储了。有关更多信息，请参阅 启动 FHIR 导入任务。

设置导入权限

1. 如果还没有，请为输出日志文件创建一个目标 Amazon S3 存储桶。Amazon S3 存储桶必须与服务位于同一 AWS 区域，并且必须为所有选项开启阻止公共访问。要了解更多信息，请参阅使用 Amazon S3 阻止公共访问。还必须使用 Amazon-owned 或客户拥有的 KMS 密钥进行加密。要了解有关使用 KMS 密钥的更多信息，请参阅 Amazon 密钥管理服务。
2. 使用以下信任策略为其创建数据访问服务角色， HealthLake 并向该 HealthLake 服务授予代入该角色的权限。 HealthLake 使用它来写入输出 Amazon S3 存储桶。

   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. 向数据访问角色添加权限策略，使其能够访问 Amazon S3 存储桶。amzn-s3-demo-bucket替换为存储桶的名称。

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"
    }]
}
```

为导出任务设置权限

在从数据存储中导出文件之前，您必须授予访问您在 Amazon S3 中的输出存储桶的 HealthLake 权限。要授予 HealthLake 访问权限，您需要为创建一个 IAM 服务角色HealthLake，向该角色添加信任策略以授予 HealthLake 代入角色权限，并向角色附加权限策略以授予其访问您的 Amazon S3 存储桶的权限。

如果您已经为创建了角色 HealthLake，则可以重复使用该角色，并向其授予本主题中列出的导出 Amazon S3 存储桶的额外权限。要了解有关 IAM 角色和信任策略的更多信息，请参阅 IAM 策略和权限。

重要提示

HealthLake 支持本机 SDK 导出请求和 FHIR R4 操作`$export`。根据您决定使用的导出 API，必须提供单独的 IAM 操作。这允许您分别处理`allow`和`deny`权限。如果您想限制 HealthLake 软件开发工具包和 FHIR REST API 的导出，则必须对单独的 IAM 操作应用拒绝权限。如果您授予用户完全访问权限，则无需更改 IAM 用户的权限 HealthLake。

使用 AWS CLI and AWS 软件开发工具包：

以下本机 HealthLake 操作可用于使用 AWS CLI 和 AWS 软件开发工具包从数据存储中导出数据：

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

使用 FHIR API：

以下 IAM 操作可用于从数据存储中导出 HealthLake 数据以及使用 FHIR `$export` 操作取消（删除）导出任务：
POST:

- `StartFHIRExportJobWithPost`

GET:

- `StartFHIRExportJobWithGet`

- DescribeFHIRExportJobWithGet
- GetExportedFile

DELETE:

- CancelFHIRExportJobWithDelete

设置权限的用户或角色必须具有创建角色、创建策略和将策略附加到角色的权限。以下 IAM 策略授予这些权限。

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"
                }
            }
        }
    ]
}
```

设置导出权限

1. 如果还没有，请为要从数据存储中导出的数据创建一个目标 Amazon S3 存储桶。Amazon S3 存储桶必须与服务位于同一 AWS 区域，并且必须为所有选项开启阻止公共访问。要了解更多信息，请参阅使用 Amazon S3 阻止公共访问。还必须使用 Amazon-owned 或客户拥有的 KMS 密钥进行加密。要了解有关使用 KMS 密钥的更多信息，请参阅 Amazon 密钥管理服务。
2. 如果您尚未创建数据访问服务角色，请使用以下信任策略为 HealthLake 该 HealthLake 服务授予代入该角色的权限。 HealthLake使用它来写入输出 Amazon S3 存储桶。如果您已经在中创建了一个存储桶为导入任务设置权限，则可以在下一步中重复使用该存储桶并向其授予访问您的 Amazon S3 存储桶的权限。

   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. 向数据访问角色添加权限策略，使其能够访问您的输出 Amazon S3 存储桶。amzn-s3-demo-bucket替换为存储桶的名称。

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"
    }]
}
```

安装 AWS CLI

AWS CLI 是描述和列出 HealthLake 导入和导出任务属性所必需的。您也可以使用 HealthLake 软件开发工具包请求此信息。

要设置 AWS CLI

1. 下载并配置 AWS CLI。有关说明，请参阅AWS Command Line Interface 用户指南中的以下主题。

   - 安装或更新最新版本的 AWS CLI
   - 开始使用 AWS CLI

2. 在 AWS CLI `config`文件中，为管理员添加已命名的配置文件。运行 AWS CLI 命令时使用此配置文件。根据最低权限的安全原则，我们建议您创建一个单独的 IAM 角色，该角色具有特定于正在执行的任务的权限。有关已命名配置文件的更多信息，请参阅《AWS Command Line Interface 用户指南》中的配置和凭证文件设置。

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

3. 请使用以下 `help` 命令验证设置：

   ```
   aws healthlake help
   ```

   如果配置 AWS CLI 正确，您将看到的简要说明 AWS HealthLake 和可用命令的列表。

AWS HealthLake 教程

目标

在本教程中，您将使用本机 HealthLake操作将 FHIR R4 数据导入 HealthLake 数据存储。接下来，您将使用 FHIR 管理（创建、读取、更新、删除）FHIR 资源。 RESTful APIs在本教程结束时，您将使用本机 HealthLake操作导出 FHIR 数据。

先决条件

设置 中列出的所有步骤都是完成本教程所必需的。

教程步骤

1. 启动 FHIR 导入任务
2. 获取 FHIR 导入任务属性
3. 创建 FHIR 资源
4. 阅读 FHIR 资源
5. 更新 FHIR 资源
6. 删除 FHIR 资源
7. 导出 FHIR 数据
8. 删除数据存储

使用管理数据存储 AWS HealthLake

使用 AWS HealthLake，您可以创建和管理 FHIR R4 资源的数据存储。创建数据存储时，FHIR HealthLake 数据存储库将通过 RESTful API 端点提供。在创建 Synthea 开源 FHIR R4 健康数据时，你可以选择将其导入（预加载）到数据存储中。有关更多信息，请参阅 预加载的数据类型。

> **重要提示**
>
> HealthLake 支持两种类型的 FHIR 数据存储授权策略，即 AWS Sigv4 或 FHIR 上的 SMART。在创建 HealthLake FHIR 数据存储之前，必须选择一种授权策略。有关更多信息，请参阅 数据存储授权策略。

要查找活动 HealthLake 数据存储的 FHIR-related 功能（行为），请检索其能力声明。

以下主题介绍如何使用、 AWS SDK 和使用 HealthLake 云原生操作创建、描述、列出、更新、标记和删除 FHIR 数据存储。 AWS CLI AWS 管理控制台

主题

- 创建 HealthLake 数据存储
- 获取 HealthLake 数据存储属性
- 列出 HealthLake 数据存储
- 更新 HealthLake 数据存储
- 标记 HealthLake 数据存储
- 删除 HealthLake 数据存储

创建 HealthLake 数据存储

`CreateFHIRDatastore`用于创建符合 FHIR R4 规范 AWS HealthLake 的数据存储。 HealthLake 数据存储用于导入、管理、搜索和导出 FHIR 数据。在创建 Synthea 开源 FHIR R4 健康数据时，你可以选择将其导入（预加载）到数据存储中。有关更多信息，请参阅 预加载的数据类型。

> **重要提示**
>
> HealthLake 支持两种类型的 FHIR 数据存储授权策略，即 AWS Sigv4 或 FHIR 上的 SMART。在创建 HealthLake FHIR 数据存储之前，必须选择一种授权策略。有关更多信息，请参阅 数据存储授权策略。

创建数据存储时，FHIR HealthLake 数据存储库将通过 RESTful API 端点提供。创建 HealthLake 数据存储后，您可以请求其能力声明来查找所有关联的 FHIR-related 权能（行为）。

创建数据存储后，您可以更改其名称、默认 FHIR 验证配置文件、NLP 配置、分析配置和身份提供者配置。无法更改加密配置。有关更多信息，请参阅 更新数据存储。

以下菜单提供了 AWS CLI 和 AWS 软件开发工具包的示例以及操作步骤。 AWS 管理控制台有关更多信息，请参阅《AWS HealthLake API Reference》中的 `CreateFHIRDatastore`。

创建 HealthLake 数据存储

根据您的访问偏好选择菜单 AWS HealthLake。

AWS CLI 和 SDK

CLI

AWS CLI

示例 1：创建 SigV4-enabled HealthLake 数据存储

以下`create-fhir-datastore`示例演示了如何在中创建新的数据存储 AWS HealthLake。

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

输出：

```
{
    "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)"
}
```

示例 2：在 FHIR-enabled HealthLake 数据存储上创建 SMART

以下`create-fhir-datastore`示例演示了如何在中的 FHIR-enabled 数据存储上创建新的 SMART 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
```

identity_provider_configuration.json 的内容：

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

输出：

```
{
```

```
    "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)"
}
```

- 有关 API 详细信息，请参阅《AWS CLI Command Reference》中的 CreateFHIRDatastore。

Python

适用于 Python 的 SDK（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
```

以下代码显示了 FHIR-enabled HealthLake 数据存储上的 SMART 参数的示例。

```
            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
            )
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的 CreateFHIRDatastore。

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅适用于 SAP 的 AWS SDK 中的 CreateFhirDataStore ABAP API 参考。

> **示例可用性**
>
> 找不到所需的内容？ 使用本页右侧边栏上的 “提供反馈” 链接请求代码示例。

AWS 控制台

> **备注**
>
> 以下过程使用 AWS Sigv4 授权创建 HealthLake 数据存储。 HealthLake 控制台不支持在 FHIR 上创建 SMART 数据存储。

使用创建 HealthLake 数据存储 AWS Sigv4 授权

1. 登录 HealthLake 控制台上的 “创建数据存储” 页面。
2. 选择创建数据存储。
3. 在数据存储设置部分中，为数据存储名称指定一个名称。
4. （可选）在数据存储设置部分中，对于预加载样本数据，选中预加载合成数据的复选框。Synthea 数据是一个开源示例数据集。有关更多信息，请参阅 合成预加载的数据类型 HealthLake。
5. 在数据存储加密部分，选择使用 AWS 拥有的密钥（默认）或选择其他 AWS KMS 密钥（高级）。
6. 在 “标签-可选” 部分中，您可以向数据存储中添加标签。要了解有关为数据存储添加标签的更多信息，请参阅标记 HealthLake 数据存储。
7. 选择创建数据存储。

   数据存储的状态可在数据存储页面上找到。

获取 HealthLake 数据存储属性

`DescribeFHIRDatastore`用于获取 AWS HealthLake 数据存储的属性。以下菜单提供了操作步骤 AWS 管理控制台 和 AWS CLI 和的代码示例 AWS SDKs。有关更多信息，请参阅《AWS HealthLake API Reference》中的 `DescribeFHIRDatastore`。

获取 HealthLake 数据存储的属性

根据您的访问偏好选择菜单 AWS HealthLake。

AWS CLI 和 SDKs

CLI

AWS CLI

描述 FHIR 数据存储

以下`describe-fhir-datastore`示例演示了如何在中查找数据存储的属性 AWS HealthLake。

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

输出：

```
{
    "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
        }
    }
}
```

- 有关 API 的详细信息，请参阅FHIRDatastore《AWS CLI 命令参考》中的 “描述”。

Python

适用于 Python 的 SDK（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
```

- 有关 API 的详细信息，请参阅适用于 Python 的AWS SDK (Boto3) API 参考FHIRDatastore中的描述。

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅适用于 SAP 的 AWS SDK ABAP API 参考FHIRDatastore中的描述。

> **示例可用性**
>
> 找不到所需的内容？ 使用本页右侧边栏上的 “提供反馈” 链接请求代码示例。

AWS 控制台

1. 登录 HealthLake 控制台上的数据存储页面。
2. 选择数据存储。

   数据存储详细信息页面打开，所有 HealthLake 数据存储属性均可用。

列出 HealthLake 数据存储

`ListFHIRDatastores`用于列出用户账户中的所有 HealthLake 数据存储，无论数据存储状态如何。以下菜单提供了操作步骤 AWS 管理控制台 和 AWS CLI 和的代码示例 AWS SDKs。有关更多信息，请参阅《AWS HealthLake API Reference》中的 `ListFHIRDatastores`。

列出所有 HealthLake 数据存储

根据您的访问偏好选择菜单 AWS HealthLake。

AWS CLI 和 SDKs

CLI

AWS CLI

列出 FHIR 数据存储

以下list-fhir-datastores示例说明如何使用该命令以及用户如何根据中的数据存储状态筛选结果 AWS HealthLake。

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

输出：

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

- 有关 API 的详细信息，请参阅FHIRDatastores《AWS CLI 命令参考》中的 “列表”。

Python

适用于 Python 的 SDK（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
```

- 有关 API 的详细信息，请参阅《适用于 Python 的AWS 软件开发工具包 (Boto3) API 参考手册》FHIRDatastores中的列表。

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅适用于 SAP 的 AWS SDK FHIRDatastores 中列出 ABAP API 参考。

> **示例可用性**
>
> 找不到所需的内容？ 使用本页右侧边栏上的 “提供反馈” 链接请求代码示例。

AWS 控制台

- 登录 HealthLake 控制台上的数据存储页面。

  所有 HealthLake 数据存储都列在 “数据存储” 部分下。

更新 HealthLake 数据存储

`UpdateFHIRDatastore`用于更新现有 AWS HealthLake 数据存储的配置。您可以更新数据存储名称、自然语言处理 (NLP) 配置、分析配置、默认 FHIR 验证配置文件和身份提供者配置。无法更改您在创建数据存储时选择的加密配置。

> **Note**
>
> 更新身份提供商配置会将其全部替换，包括您要保留的每个字段。您省略的任何字段都将被清除。

以下菜单提供了 AWS CLI 和 AWS 软件开发工具包的示例。有关更多信息，请参阅《AWS HealthLake API Reference》中的 `UpdateFHIRDatastore`。

> **重要提示**
>
> NLP 和分析更改是通过异步工作流程应用的：数据存储状态更改为更新完成时`UPDATING`并返回到更新完成`ACTIVE`时，或者显示`UPDATE_FAILED`是否未完成。数据存储名称、FHIR 验证配置文件和身份提供者更改会立即生效，并且不会更改状态。一个数据存储一次只能进行一次更新；在数据存储运行时提交的第二次更新将返回`ConflictException`。`DescribeFHIRDatastore`用于跟踪更新的状态。

更新 HealthLake 数据存储

根据您的访问偏好选择菜单 AWS HealthLake。

AWS CLI 和 SDK

AWS CLI

示例 1：重命名数据存储

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

示例 2：启用 NLP

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

示例 3：暂停分析

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

示例 4：更新默认 FHIR 验证配置文件

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

响应返回完整的 DatastoreProperties — 与返回的形状相同DescribeFHIRDatastore。

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

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

有关 API 的详细信息，请参阅 CLI 命令参考中的 update-fhir-datastore。AWS

Python

适用于 Python (Boto3) 的 SDK

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

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

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

```
        )
        raise
```

有关 API 的详细信息，请参阅 Python SD AWS K (Boto3) API 参考中的 updateFhirDataStore。

> **示例可用性**
>
> 找不到所需的内容？ 使用本页右侧边栏上的 “提供反馈” 链接请求代码示例。

标记 HealthLake 数据存储

您可以以标签的形式将元 HealthLake 数据分配给数据存储。每个标签都是由用户定义的键和值组成的标签。标签可帮助您管理、识别、组织、搜索和筛选数据存储。

> **重要提示**
>
> 请勿在标签中存储受保护的健康信息（PHI）、个人身份信息（PII）或其他机密或敏感信息。标签的用途并非用于私人或敏感数据。

以下主题介绍如何使用 AWS 管理控制台 AWS CLI、和 AWS SDKs使用 HealthLake 标记操作。有关更多信息，请参阅AWS 一般参考 指南中的为AWS 资源添加标签。

主题

标记 HealthLake 数据存储

`TagResource`用于标记 HealthLake 数据存储。以下菜单提供了操作步骤 AWS 管理控制台 和 AWS CLI 和的代码示例 AWS SDKs。有关更多信息，请参阅《AWS HealthLake API Reference》中的 `TagResource`。

为 HealthLake 数据存储添加标签

根据您的访问偏好选择菜单 AWS HealthLake。

AWS CLI 和 SDKs

CLI

AWS CLI

向数据存储中添加标签

以下 tag-resource 示例演示如何向数据存储中添加标签。

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

此命令不生成任何输出。

。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考TagResource中的。

Python

适用于 Python 的 SDK（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
```

- 有关 API 的详细信息，请参阅适用TagResource于 Python 的AWS SDK (Boto3) API 参考。

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅适用TagResource于 S AP 的AWS SDK ABAP API 参考。

> **示例可用性**
>
> 找不到所需的内容？ 使用本页右侧边栏上的 “提供反馈” 链接请求代码示例。

AWS 控制台

1. 登录 HealthLake 控制台上的数据存储页面。
2. 选择数据存储。

   数据存储详细信息页面将会打开。
3. 在 标签 部分中，选择 管理标签。

   将打开管理标签页面。
4. 选择 添加新标签。
5. 输入一个 键和可选的 值（可选）。
6. 选择保存。

列出 HealthLake 数据存储的标签

`ListTagsForResource`用于列出 HealthLake 数据存储的标签。以下菜单提供了操作步骤 AWS 管理控制台 和 AWS CLI 和的代码示例 AWS SDKs。有关更多信息，请参阅《AWS HealthLake API Reference》中的 `ListTagsForResource`。

列出 HealthLake 数据存储的标签

根据您的访问偏好选择菜单 AWS HealthLake。

AWS CLI 和 SDKs

CLI

AWS CLI

列出数据存储的标签

以下 `list-tags-for-resource` 示例列出与指定数据存储关联的标签。

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

输出：

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

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ListTagsForResource中的。

Python

适用于 Python 的 SDK（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
```

- 有关 API 的详细信息，请参阅适用ListTagsForResource于 Python 的AWS SDK (Boto3) API 参考。

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅适用ListTagsForResource于 S AP 的AWS SDK ABAP API 参考。

> **示例可用性**
>
> 找不到所需的内容？ 使用本页右侧边栏上的 “提供反馈” 链接请求代码示例。

AWS 控制台

1. 登录 HealthLake 控制台上的数据存储页面。
2. 选择数据存储。

   数据存储详细信息页面将会打开。在标签部分下，列出了所有数据存储标签。

取消标记数据存储 HealthLake

UntagResource用于从 HealthLake 数据存储中移除标签。以下菜单提供了操作步骤 AWS 管理控制台 和 AWS CLI 和的代码示例 AWS SDKs。有关更多信息，请参阅《AWS HealthLake API Reference》中的 UntagResource。

取消对 HealthLake 数据存储的标记

根据您的访问偏好选择菜单 AWS HealthLake。

AWS CLI 和 SDKs

CLI

AWS CLI

从数据存储中移除标签

以下 untag-resource 示例演示如何从数据存储中移除标签。

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

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考UntagResource中的。

Python

适用于 Python 的 SDK（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
```

- 有关 API 的详细信息，请参阅适用UntagResource于 Python 的AWS SDK (Boto3) API 参考。

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅适用UntagResource于 S AP 的AWS SDK ABAP API 参考。

> **示例可用性**
>
> 找不到所需的内容？ 使用本页右侧边栏上的 “提供反馈” 链接请求代码示例。

AWS 控制台

1. 登录 HealthLake 控制台上的数据存储页面。

2. 选择数据存储。

   数据存储详细信息页面将会打开。

3. 在 标签 部分中，选择 管理标签。

   将打开管理标签页面。

4. 在标签旁选择 移除，以移除标签。

5. 选择保存。

删除 HealthLake 数据存储

DeleteFHIRDatastore用于删除 HealthLake 数据存储。以下菜单提供了操作步骤 AWS 管理控制台 和 AWS CLI 和的代码示例 AWS SDKs。有关更多信息，请参阅《AWS HealthLake API Reference》中的 DeleteFHIRDatastore。

删除 HealthLake 数据存储

根据您的访问偏好选择菜单 AWS HealthLake。

AWS CLI 和 SDKs

CLI

AWS CLI

删除 FHIR 数据存储

以下delete-fhir-datastore示例演示如何删除数据存储及其中的所有内容 AWS HealthLake。

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

输出：

```
{
    "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)"
}
```

- 有关 API 的详细信息，请参阅FHIRDatastore《AWS CLI 命令参考》中的 “删除”。

Python

适用于 Python 的 SDK（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
```

- 有关 API 的详细信息，请参阅 Python FHIRDatastore 版AWS SDK 中删除 (Boto3) API 参考。

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅在 SAP 的 AWS SDK FHIRDatastore 中删除 ABAP API 参考。

> **示例可用性**
>
> 找不到所需的内容？ 使用本页右侧边栏上的 “提供反馈” 链接请求代码示例。

AWS 控制台

1. 登录 HealthLake 控制台上的数据存储页面。
2. 选择数据存储。

   数据存储详细信息页面将会打开。
3. 选择删除。

   删除数据存储页面将会打开。
4. 要确认删除数据存储，请在文本输入字段中输入数据存储名称。
5. 选择删除。

在中管理 FHIR 订阅 AWS HealthLake

AWS HealthLake 支持 FHIR 订阅，允许您在特定医疗保健数据发生变化时收到实时通知。此功能实现了 FHIR R5 Backport 基于主题的订阅模型，与传统的 FHIR R4 订阅模式相比，提供了更好的可扩展性和灵活性。

借助 FHIR Subscriptions，您可以构建事件驱动的医疗保健应用程序，这些应用程序可以立即响应临床数据的变化，从而实现及时的干预、自动化的工作流程和增强的护理协调。

主题

FHIR 订阅的工作原理

FHIR 订阅在基于主题的模型上 HealthLake 运行，其中：

1. 创建主题以定义事件：创建订阅主题以指定可以触发通知的事件
2. 您订阅：使用特定的筛选条件创建对这些主题的订阅
3. HealthLake 监视器：该服务会持续监控符合您标准的事件
4. 已发送通知：发生 CWhen 匹配事件， HealthLake 通过您选择的渠道发送通知

关键组件

FHIR 订阅由以下组件组成。

订阅话题

订阅主题是通知系统的基础，它定义了：

- 触发事件：哪些更改会触发通知（例如：资源创建、更新、删除）
- 可用过滤器：订阅者可以使用哪些筛选选项
- 通知内容：通知中包含哪些数据

下表列出了常见的主题类型。

事件类型	说明	常见使用案例
资源创建	资源创建时触发	新患者登记，记录新的观察结果
资源更新	资源被修改时触发	状态变化、临床更新
资源删除	资源被删除时触发	审计和合规跟踪

订阅

订阅是指您请求接收由订阅主题定义的特定事件的通知。每份订阅包括：

- 主题参考：指定您要订阅的订阅主题
- 过滤器：选择哪些事件生成通知的标准
- 频道配置：应在何处以及如何发送通知
- 有效负载首选项：通知中应包含什么详细级别

通知渠道

HealthLake 支持以下通知渠道：

通道类型	使用案例	
EventBridge	企业集成、无服务器工作流程、跨服务编排AWS	

通道类型	使用案例	
REST Hook	直接端点通知，第三方系统集成	

通知负载

根据需要选择合适的有效载荷类型：

有效载荷类型	说明	安全注意事项
仅限身份证件	仅包含资源标识符	最小 PHI 暴露
完整资源	包含完整资源内容，最大大小为 256 KB。如果大小大于 256KB，它将恢复为 “仅限身份证”	包含 PHI；验证安全处理

最佳实践

性能优化

- 使用有针对性的过滤器：缩小标准以仅接收基本通知
- 选择适当的负载类型：尽可能使用仅限 ID 的有效载荷以获得更好的性能
- 实现高效的接收器：确保通知接收者快速处理消息

安全注意事项

- 安全端点：对 REST Hook 端点实施正确的身份验证
- PHI 保护：请谨慎使用完整资源有效负载，因为它们包含 PHI
- 访问控制：仅限授权用户创建订阅

卓越运营

- 设置适当的结束日期：使用临时订阅的结束日期

- 监控订阅状态：定期检查您的订阅状态
- 实现错误处理：设计应用程序以处理通知传送失败

FHIR 订阅生命周期 AWS HealthLake

请按照以下步骤了解 FHIR 订阅生命周期：

1. 创建 SubscriptionTopic
   - 使用状态SubscriptionTopic创建 "unknown"
2. 创建 Subscription
   - 使用状态Subscription创建 "requested"
   - HealthLake 验证配置 Subscription
   - Subscription必须引用已存在的主题（主题必须处于状态unknown、draft、active）。
3. Activation
   - 如果有效，则Subscription将的状态 HealthLake 更新为 "active"
   - 创建时Subscription，如果给定的主题处于状态"unknown"，则当订阅也处于活动状态时，状态会 HealthLake更新为 "active"
   - 订阅通常需要 5-10 分钟才能成功创建
   - 如果Subscription未成功创建，则状态将更改为应执行删除操作error的位置，然后重试创建订阅。您可以查看订阅资源中的"error"字段，以了解订阅未成功创建的原因。
4. 订阅时正在收录 active
5. Whil Subscription e 是 active
   - HealthLake 监视符合您标准的事件
   - 发生匹配时，通知会发送到已配置的端点
6. 错误处理
   - HealthLake 尝试重试 14 天，然后停止重试这些事件
7. 停用
   - Subscription可以通过以下方式停用 A：

     设置结束日期（自动停用）

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

删除Subscription资源

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

使用创建 FHIR 订阅 AWS HealthLake

以下指南向您展示如何使用 AWS HealthLake创建 FHIR 订阅。

创建 FHIR 订阅

1. 创建 SubscriptionTopic。

   订阅主题资源示例：

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

2. 准备您的通知终端节点（自定义渠道）。以下步骤是确保终端接收通知的必要步骤

   使用 REST Hook 时

   - 如果使用 CM_CMK 数据存储，请信任events.amazonaws.com您的 KMS 密钥策略。
   - 如果使用 CM_CMK 数据存储，则必须在 KMS 密钥中添加值为的EventBridgeApiDestinations标签 true
   - HealthLake 用于 OAuth 对您的 REST 挂钩端点进行身份验证。因此，在创建 REST 挂钩订阅时，必须传入客户端 ID、客户端密钥和 oAuth-endpoint-url频道。 _type.extension [*]。

   使用 CM_CMK 数据存储时的 KMS 密钥策略示例：

```
{
  "Sid": "AllowEventBridgeToUseKMSKey",
  "Effect": "Allow",
  "Principal": {
```

```
    "Service": ["events.amazonaws.com", "healthlake.amazonaws.com"]
  },
  "Action": ["kms:GenerateDataKey*", "kms:Decrypt", "kms:DescribeKey"],
  "Resource": "*"
}
```

使用时 EventBridge

- 如果使用 CM_CMK 数据存储，请信任`events.amazonaws.com`您的 KMS 密钥策略。
- 验证您的 EventBridge 资源策略是否信任`healthlake.amazonaws.com`服务主体。
- 使用 CM_CMK 和 EventBridge 作为终端节点时，请验证是否使用与数据存储 KMS 密钥相同的 KMS 密钥对 EventBridge 总线进行加密。
- 确认您的 EventBridge 总线至少有 1 条规则与生成的事件匹配 HealthLake。

EventBridge 频道总线的资源策略示例：

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

用于接收来自以下事件的 EventBridge 规则事件模式示例： HealthLake

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

> **Note**
>
> HealthLake 支持 2 个来源：
>
> - “healthlake”：仅适用于订阅。
> - “aws.healthlake”: 接收 HealthLake 服务事件。
>
> 在“healthlake”为 FHIR 订阅事件总线创建规则时用作源。

3. 创建 Subscription

使用以下方式提交订阅资源：

- 状态："requested"
- 引用您选择的SubscriptionTopic身份证件
- 筛选标准。有关更多信息，请参阅筛选受支持的过滤器的通知。
- 通道配置

订阅有效负载示例

以下代码示例展示了如何创建订阅负载。

EventBridge

使用event-bridge频道和id-only有效载荷类型进行订阅。

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

使用event-bridge终端节点和full-resource有效负载类型进行订阅。

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

休息挂钩

使用rest-hook终端节点和id-only有效负载类型进行订阅。

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

使用rest-hook频道和full-resource有效载荷类型进行订阅。

```
{
  "resourceType": "Subscription",
  "meta": {
    "profile": [
      "http://hl7.org/fhir/uv/subscriptions-backport/StructureDefinition/backport-
subscription"
    ]
```

```
  },
  "status": "requested",
  "end": "2026-07-31T05:38:17.2404292+00:00",
  "reason": "Test subscription for walkthrough",
  "criteria": "https://healthlake.<AWS_REGION>.amazonaws.com/datastore/<datastoreId>/
r4/SubscriptionTopic/<your topic id>",
  "_criteria": {
    "extension": [
      {
        "url": "http://hl7.org/fhir/uv/subscriptions-backport/StructureDefinition/
backport-filter-criteria",
        "valueString": "Encounter?subject=Patient/test-patient-id"
      }
    ]
  },
  "channel": {
    "type": "rest-hook",
    "_type": {
      "extension": [
        {
          "url": "http://healthlake.amazonaws.com/channel-type-clientId",
          "valueString": "<CLIENT_ID>"
        },
        {
          "url": "http://healthlake.amazonaws.com/channel-type-clientSecret",
          "valueString": "<CLIENT_SECRET>"
        },
        {
          "url": "http://healthlake.amazonaws.com/channel-type-oauth-endpoint",
          "valueUri": "<OAUTH_ENDPOINT_URL>"
        }
      ]
    },
    "endpoint": "<YOUR_REST_HOOK_ENDPOINT>",
    "payload": "application/fhir+json",
    "_payload": {
      "extension": [
        {
          "url": "http://hl7.org/fhir/uv/subscriptions-backport/StructureDefinition/
backport-payload-content",
          "valueCode": "full-resource"
        }
      ]
    }
```

```
  }
}
```

通知负载示例

在创建订阅时，通过向您配置的频道发送握手捆绑包来 HealthLake 检查订阅设置是否成功。以下有效载荷是握手捆绑包的示例。

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

```
      ]
    }
  }
}
```

仅限身份的通知包示例。

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

```
                "timestamp": "2025-09-05T00:18:43.393775234Z",
                "focus": "Encounter/c5ae898f-bd96-44dd-a509-87fdbcf23b19",
                "additionalContext": "Encounter/c5ae898f-bd96-44dd-a509-87fdbcf23b19/
_history/1",
                "id": "8f4e9c1a-2b3d-4e5f-6a7b-8c9d0e1f2a3b"
              }
            ]
          }
        }
      ]
    }
  }
}
```

完整资源通知包示例。

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

```
            "type": "event-notification",
            "eventsSinceSubscriptionStart": "-1",
            "subscription": {
              "reference": "https://healthlake.<AWS_REGION>.amazonaws.com/datastore/
<DS_ID>/r4/Subscription/<SUBSCRIPTION_ID>"
            },
            "topic": "https://healthlake.<AWS_REGION>.amazonaws.com/datastore/<DS_ID>/
r4/SubscriptionTopic/<TOPIC_ID>",
            "notificationEvent": [
              {
                "eventNumber": "0",
                "timestamp": "2025-09-05T00:18:43.845970754Z",
                "focus": "Encounter/82776529-59a0-4d63-bedb-82f6726d65b5",
                "additionalContext": "Encounter/82776529-59a0-4d63-bedb-82f6726d65b5/
_history/1",
                "id": "7a8b9c0d-1e2f-3a4b-5c6d-7e8f9a0b1c2d"
              }
            ]
          }
        },
        {
          "fullUrl": "Encounter/82776529-59a0-4d63-bedb-82f6726d65b5",
          "resource": {
            "resourceType": "Encounter",
            "id": "82776529-59a0-4d63-bedb-82f6726d65b5",
            "status": "finished",
            "class": {
              "system": "http://terminology.hl7.org/CodeSystem/v3-ActCode",
              "code": "AMB",
              "display": "ambulatory"
            },
            "subject": {
              "reference": "Patient/test-patient-id"
            },
            "meta": {
              "lastUpdated": "2025-09-05T00:18:43.219652906Z",
              "versionId": "1"
            }
          },
          "request": {
            "method": "CREATE",
            "url": "Encounter/82776529-59a0-4d63-bedb-82f6726d65b5"
          }
        }
```

```
      ]
    }
  }
}
```

事件版本控制

HealthLake 默认情况下支持 FHIR 历史记录。

要了解您在通知包中收到的资源版本，请执行以下操作：

- full-resource：由于完整资源包包含整个资源，因此该`entry[*]`版本将包含在捆绑包中每个资源的中。
- 仅限 id：捆绑包将不包含任何资源信息。 HealthLake包括通过`entry[0].notificationEvent[*].additionalContext`字段匹配并包含在捆绑包中的版本。此字段的格式为`<ResourceType>/<ResourceId>/_history/<Version Id>`。有关更多信息，请参阅仅限身份的有效负载示例中的 additionalContext 字段。

事件重复检测

HealthLake的 FHIR 订阅功能可保证至少一次交付。这意味着您可能会多次收到同一个事件，无论是在同一个捆绑包中还是在不同的捆绑包中。要识别重复项，请为中的通知包中的`entry[0].notificationEvent[*].id`每个事件 HealthLake 提供唯一的 ID。

此 ID 对于已匹配和交付的事件的特定版本是唯一的。例如，如果同一个遭遇战更新了两次，并且两个更新都符合筛选条件，那么您将收到两个具有相同遭遇参考的单独事件。他们会有同样的东西`notificationEvent[*].focus`，但会有独一无二的`notificationEvent[*].id`。此外，这些事件可以在单独的捆绑包中发送，也可以在同一个通知包中发送。

使用以下方式搜索 FHIR 订阅 AWS HealthLake

`Subscription`而且`SubscriptionTopic`资源也是可以搜索的。 HealthLake 支持订阅和 SubscriptionTopic 资源的所有常用搜索参数。

此外，我们还通过以下参数支持其他搜索功能：

订阅

搜索参数	说明	示例
contact	在 R4 核心规范中的 “订阅.contact” 字段中进行搜索	`Subscription?contact=phone`
criteria	在 R4 核心规范中的 “订阅.criteria” 字段中进行搜索	`Subscription?criteria=[baseUrl]/datastore/[datastoreId]/r4/SubscriptionTopic/[topicId]`
payload	在 R4 核心规范中的 subscription.Channel.payload 字段中搜索	`Subscription?payload=application/fhir+json`
status	在 R4 核心规格中的 “订阅.Status” 字段中搜索	`Subscription?status=error`
类型	在 R4 核心规格中的 subscription.Channel.type 字段中搜索	`Subscription?topic=event-bridge`
url	在 R4 核心规范中的 subscription.Channel.Endpoint 字段中搜索	`Subscription?url=[Subscription.channel.endpoint]`
筛选标准	在标准扩展字段中搜索网址 “http://hl7。 org/fhir/uv/subscriptions-backport/StructureDefinition/backport-filter-criteria” 作为字符串	`Subscription?filter-criteria=Encounter?`
自定义频道	在自定义频道类型附加信息字段中搜索网址 “http://hl7。 org/fhir/uv/subscriptions-backport/StructureDefinition/backport-channel-type” 作为编码 订阅负载示例	`Subscription?custom-channel=[System]|[code]`

搜索参数	说明	示例
有效载荷类型	在有效载荷类型扩展字段上进行搜索，您可以在其中指定有效负载的格式（`id-only`或`full-resource` ）	`Subscription?payload-type=full-resource`
topic	在 Subscription.criteria 字段中进行搜索，该字段添加了主题	`Subscription?topic=[topicId]`

SubscriptionTopic

搜索参数	说明	示例
date	在 SubscriptionTopic 资源中的日期字段上搜索	`SubscriptionTopic?date=[SubscriptionTopic.date]`
derived-or-self	搜索 SubscriptionTopic 资源中的`url`或`derivedFrom` 字段	`SubscriptionTopic?derived-or-self=[SubscriptionTopic.url | SubscriptionTopic.derivedFrom]`
identifier	在 SubscriptoinTopic .identifier 字段上搜索	`SubscriptionTopic?identifier=[SubscriptionTopic.identifier]`
资源	在 .resourceTri Subscript ionTopic gger.resource 字段中搜索	`SubscriptionTopic?resource=Encounter`
status	搜索的状态 SubscriptionTopic	`SubscriptionTopic?status=unknown`

搜索参数	说明	示例
删除实例快照	在标题上搜索 SubscriptionTopic	`SubscriptionTopic?title=admission`
触发器描述	在 .resou SubscriptionTopic rceTrigger.descrip	`SubscriptionTopic.trigger-description=resource moving to state 'in-progress'`
url	在网址上搜索 SubscriptionTopic	`SubscriptionTopic?url=[SubscriptionTopic.url]`
版本	在版本上搜索 SubscriptionTopic	`SubscriptionTopic?version=1`

使用筛选通知 AWS HealthLake

使用标准中的 FHIR 搜索参数来优化您的`Subscription`通知。

支持的过滤器

下表显示了支持订阅的支持的筛选条件列表。 HealthLake

搜索参数类型	FHIR 数据类型	修饰符	支持的前缀
字符串	字符串 HumanName 地址	精确，包含缺失	
令牌	布尔值 代码 字符串 编码	不是，文本，缺失	

搜索参数类型	FHIR 数据类型	修饰符	支持的前缀
	CodeableConcept 标识符 ContactPoint id		
数字	整数 decimal positiveInt unsignedInt	missing	“eq”、“ne”、“gt”、“lt”、“ge”、“le”、“sa”、“eb”、“ap”
日期	date dateTime 即时 周期 Timing		“eq”、“ne”、“gt”、“lt”、“ge”、“le”、“sa”、“eb”、“ap”
Quantity	数量 钱 Range SimpleQuantity		“eq”、“ne”、“gt”、“lt”、“ge”、“le”、“sa”、“eb”、“ap”
参考	参考	缺失、标识符、类型	

搜索参数类型	FHIR 数据类型	修饰符	支持的前缀
URI	uri url 权威性的 uid oid	missing	

支持的过滤器示例

下表显示了支持订阅的受 HealthLake 支持筛选条件的示例：

用途	筛选条件	说明
针对患者的观察	`Observation?patient=Patient/[id]&status=final`	当对特定患者的观察完成时收到通知
针对患者的观察	`Patient?birthdate=gt2021`	在 2021 年之后出生的患者注册或更新时收到通知
针对患者的观察	`Condition?code=http://snomed.info/sct|39065001`	使用特定 SNOMED 代码获取有关条件的通知
针对患者的观察	`Observation?code=http://loinc.org|8480-6&value-quantity=gt160`	获取高收缩压读数的通知

使用导入 FHIR 数据 AWS HealthLake

创建 HealthLake 数据存储后，下一步是从 Amazon Simple Storage Service (S3) 存储桶导入文件。您可以使用 AWS 管理控制台、 AWS CLI或 AWS SDKs启动 FHIR 导入任务。使用本机 AWS HealthLake 操作启动、描述和列出 FHIR 导入任务。

> **重要提示**
>
> HealthLake 支持用于医疗保健数据交换的 FHIR R4 规范。如果需要，您可以与AWS HealthLake 合作伙伴合作，在导入之前将您的健康数据转换为 FHIR R4 格式。

启动 FHIR 导入任务时，您可以指定 Amazon S3 存储桶输入位置、Amazon S3 存储桶输出位置（用于任务处理结果）、授予 HealthLake 访问您的 Amazon S3 存储桶访问权限的 IAM 角色以及客户拥有或 AWS 拥有 AWS Key Management Service 的密钥。有关更多信息，请参阅 为导入任务设置权限。

> **Note**
>
> 您可以对导入任务进行排队。异步导入任务以 FIFO（先入先出）方式处理。您可以像开始导入任务一样对任务进行排队。如果正在进行中，则只需排队即可。在导入任务进行期间，您可以创建、读取、更新或删除 FHIR 资源。

HealthLake 为每个 FHIR 导入任务生成一个`manifest.json`文件。该文件描述了 FHIR 导入任务的成功和失败。 HealthLake 将`manifest.json`文件输出到启动 FHIR 导入任务时指定的 Amazon S3 存储桶。日志文件分为两个文件夹，名为`SUCCESS`和`FAILURE`。使用该`manifest.json`文件作为对失败的导入任务进行故障排除的第一步，因为它提供了每个文件的详细信息。

```
{
    "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
}
```


为导入配置验证级别

启动 FHIR 导入任务时，您可以选择指定应用`ValidationLevel`于每个资源。 AWS HealthLake 目前支持以下验证级别：

- `strict`：根据资源的配置文件元素对资源进行验证，如果不存在配置文件，则根据R4规格进行验证。这是的默认验证级别 AWS HealthLake。
- `structure-only`：根据 R4 对资源进行验证，忽略任何引用的配置文件。
- `minimal`：资源经过最低限度的验证，忽略了某些 R4 规则。未通过所需的结构检查的资源 search/analytics 将进行更新，以包括审计警告。

使用`minimal`验证级别导入时，可能会在名为的文件夹中生成其他日志文件`SUCCESS_WITH_SEARCH_VALIDATION_FAILURES`。尽管与搜索相关的验证检查失败，但该文件夹日志文件中的资源仍被提取到您的数据存储中。根据 FHIR，这意味着您的 FHIR 资源的某些方面无效，并且可能无法搜索格式错误的字段。这些资源后面会有一个描述上述失败的`extension`附件。

主题

- 启动 FHIR 导入任务
- 获取 FHIR 导入任务属性
- 列出 FHIR 导入任务

启动 FHIR 导入任务

StartFHIRImportJob用于启动 FHIR 导入 HealthLake 数据存储的任务。以下菜单提供了操作步骤 AWS 管理控制台 和 AWS CLI 和的代码示例 AWS SDKs。有关更多信息，请参阅《AWS HealthLake API Reference》中的 StartFHIRImportJob。

> **重要提示**
>
> HealthLake 支持用于医疗保健数据交换的 FHIR R4 规范。如果需要，您可以与AWS HealthLake 合作伙伴合作，在导入之前将您的健康数据转换为 FHIR R4 格式。

启动 FHIR 导入作业

根据您的访问偏好选择菜单 AWS HealthLake。

AWS CLI 和 SDKs

CLI

AWS CLI

启动 FHIR 导入作业

以下start-fhir-import-job示例说明如何使用 AWS HealthLake启动 FHIR 导入任务。

```
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)"
```

输出：

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

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 “启动 FHIRImport J ob”。

Python

适用于 Python 的 SDK（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
```

- 有关 API 的详细信息，请参阅 Python 版AWS SDK 中启动FHIRImport作业 (Boto3) API 参考。

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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.
```

- 有关 API 的详细信息，请参阅在 SAP 的 AWS SDK 中启动 FHIRImport J ob ABAP API 参考。

> **示例可用性**
>
> 找不到所需的内容？ 使用本页右侧边栏上的 “提供反馈” 链接请求代码示例。

AWS 控制台

1. 登录 HealthLake 控制台上的数据存储页面。
2. 选择数据存储。
3. 选择导入。

   将打开 “导入” 页面。
4. 在输入数据部分下，输入以下信息：

   - Amazon S3 中的输入数据位置
5. 在 “导入输出文件” 部分下，输入以下信息：

   - 导入输出文件在 Amazon S3 中的位置
   - 导入输出文件加密
6. 在访问权限部分下，选择使用现有 IAM 服务角色并从服务角色名称菜单中选择该角色或选择创建 IAM 角色。
7. 选择导入数据。

> **Note**
>
> 在导入过程中，在页面顶部的横幅上选择 “复制作业 ID”。您可以使用`JobID`来请求导入任务属性 AWS CLI。有关更多信息，请参阅 获取 FHIR 导入任务属性。

获取 FHIR 导入任务属性

用于获`DescribeFHIRImportJob`取 FHIR 导入任务属性。以下菜单提供了操作步骤 AWS 管理控制台 和 AWS CLI 和的代码示例 AWS SDKs。有关更多信息，请参阅《AWS HealthLake API Reference》中的 `DescribeFHIRImportJob`。

要获取 FHIR 导入任务属性

根据您的访问偏好选择菜单 AWS HealthLake。

AWS CLI 和 SDKs

CLI

AWS CLI

描述 FHIR 导入作业

以下`describe-fhir-import-job`示例说明如何使用 AWS HealthLake学习 FHIR 导入任务的属性。

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

输出：

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

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 “描述 FHIRImport J ob”。

Python

适用于 Python 的 SDK（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
```

- 有关 API 的详细信息，请参阅 Python 版AWS SDK 中描述FHIRImport作业 (Boto3) API 参考。

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅在 SAP 的 AWS SDK 中描述 FHIRImport J ob ABAP API 参考。

> **示例可用性**
>
> 找不到所需的内容？ 使用本页右侧边栏上的 “提供反馈” 链接请求代码示例。

AWS 控制台

> **Note**
>
> HealthLake 控制台上没有 FHIR 导入任务信息。而是使用 wit AWS CLI h `DescribeFHIRImportJob` 来请求导入任务属性，例如`JobStatus`。有关更多信息，请参阅本页上的 AWS CLI 示例。

列出 FHIR 导入任务

用于列`ListFHIRImportJobs`出活动 HealthLake 数据存储的 FHIR 导入任务。以下菜单提供了操作步骤 AWS 管理控制台 和 AWS CLI 和的代码示例 AWS SDKs。有关更多信息，请参阅《AWS HealthLake API Reference》中的 `ListFHIRImportJobs`。

列出 FHIR 导入任务

根据您的访问偏好选择菜单 AWS HealthLake。

AWS CLI 和 SDKs

CLI

AWS CLI

列出所有 FHIR 导入作业

以下 `list-fhir-import-jobs` 示例演示了如何使用该命令查看与账户关联的所有导入作业的列表。

```
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)
```

输出：

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

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 “列出FHIRImport作业”。

Python

适用于 Python 的 SDK（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
```

- 有关 API 的详细信息，请参阅 Python 版AWS SDK 中列出FHIRImport作业 (Boto3) API 参考。

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅在 SAP 的 AWS SDK 中列出FHIRImport作业 ABAP API 参考。

示例可用性

找不到所需的内容？ 使用本页右侧边栏上的 “提供反馈” 链接请求代码示例。

AWS 控制台

Note

HealthLake 控制台上没有 FHIR 导入任务信息。而是使用 with 列 AWS CLI `ListFHIRImportJobs`出所有 FHIR 导入任务。有关更多信息，请参阅本页上的 AWS CLI 示例。

管理 FHIR 的资源 AWS HealthLake

使用 FHIR R4 RESTful API 交互来管理数据存储中的 FHIR 资源。 HealthLake 以下各节描述了所有可用于 FH HealthLake IR 资源管理的 FHIR R4 RESTful API 交互。有关 HealthLake 数据存储功能及其支持 FHIR 规范的哪些部分的信息，请参见FHIR R4 能力声明 AWS HealthLake。

> **Note**
>
> 本章中列出的FHIR交互是按照 HL7 FHIR R4医疗保健数据交换标准构建的。由于它们是 HL7 FHIR 服务的代表，因此它们不是通过 AWS CLI 和 AWS SDKs提供的。有关更多信息，请参阅 FHIR R4 RESTful API 文档中的 RESTful API。

下表列出了支持的 FHIR R4 交互。 AWS HealthLake有关支持的 FHIR 资源类型的信息 HealthLake，请参见资源类型。

支持的 FHIR R4 交互作用 AWS HealthLake

相互作用	说明
整个系统的交互	
`capabilities`	获取系统的能力声明。请参阅FHIR R4 能力声明 AWS HealthLake。
`batch`	在一次交互中更新、创建或删除一组资源。请参阅捆绑 FHIR 资源。
类型级别的互动	
`create`	使用服务器分配的 ID 创建新资源。请参阅创建 FHIR 资源。
`search`	根据某些筛选条件搜索资源类型。请参阅正在搜索 FHIR 资源。
`history`	检索特定资源类型的更改历史记录。请参阅阅读 FHIR 资源历史记录。
实例级别的交互	
`read`	读取资源的当前状态。请参阅读取 FHIR 资源。
`history`	阅读特定资源的变更历史记录。请参阅阅读 FHIR 资源历史记录。

相互作用	说明
vread	读取资源特定版本的状态。请参阅读取特定版本的 FHIR 资源历史记录。
update	按资源的 ID 更新资源（如果资源是新的，则创建它）。请参阅更新 FHIR 资源。
delete	删除资源。请参阅删除 FHIR 资源。

主题

创建 FHIR 资源

FHIR `create` 交互会在 HealthLake 数据存储中创建新的 FHIR 资源。有关更多信息，请参阅 FHIR R4 RESTful API 文档`create`中的。

创建 FHIR 资源

1. 收集 HealthLake `region`和`datastoreId`价值。有关更多信息，请参阅 获取数据存储属性。
2. 确定要创建的 FHIR `Resource` 的类型。有关更多信息，请参阅 资源类型。
3. 使用收集的 HealthLake `region`和值为请求构造一个 URL `datastoreId`。还要包括要创建的 FHIR `Resource` 类型。要查看以下示例中的整个 URL 路径，请滚动到“复制”按钮。

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

4. 为请求构建 JSON 正文，指定新资源的 FHIR 数据。出于本过程的目的，我们使用的是 FHIR `Patient` 资源，因此请将文件另存为`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. 发送 请求。FHIR `create` 交互在 FHIR 授权上使用AWS 签名版本 4 或 SMART 的POST请求。以下示例 HealthLake 使用 curl 或控制台在中创建 FHIR `Patient` 资源。 HealthLake 要查看整个示例，请滚动到 “复制” 按钮。

   SigV4

   Sigv4 授权

   ```
   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

   该IdentityProviderConfiguration数据类型的 SMART on FHIR 授权示例。

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

调用者可以在授权 lambda 中分配权限。有关更多信息，请参阅 OAuth 2.0 作用域。

AWS Console

> **Note**
>
> HealthLake 控制台仅支持 AWS SigV4 授权。

1. 登录 HealthLake 控制台上的 “运行查询” 页面。

2. 在 “查询设置” 部分下，进行以下选择。

- 数据存储 ID-选择数据存储 ID 以生成查询字符串。
- 查询类型-选择`Create`。
- 资源类型-选择要创建的 FHIR 资源类型。
- 请求正文 — 为请求构建 JSON 正文，指定新资源的 FHIR 数据。

3. 选择运行查询。

为资源创建配置验证级别

创建 FHIR 资源时，您可以选择指定 `x-amzn-healthlake-fhir-validation-level` HTTP 标头来配置资源的验证级别。 AWS HealthLake 目前支持以下验证级别：

- `strict`：根据资源的配置文件元素对资源进行验证，如果不存在配置文件，则根据R4规格进行验证。这是的默认验证级别 AWS HealthLake。
- `structure-only`：根据 R4 对资源进行验证，忽略任何引用的配置文件。
- `minimal`：资源经过最低限度的验证，忽略了某些 R4 规则。未通过所需的结构检查的资源 search/analytics 将进行更新，以包括审计警告。

尽管搜索索引需要验证失败，但使用最低验证级别创建的资源仍可能被提取到数据存储中。在这种情况下，将更新资源，以包括专门用于记录上述故障的 Healthlake 扩展程序：

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

此外，将包含以下 HTTP 响应标头，其值为 “true”：

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

> **Note**
>
> 如果存在这些错误，则根据 R4 规范提取的数据可能无法按预期进行搜索。

读取 FHIR 资源

FHIR `read` 交互读取 HealthLake数据存储中资源的当前状态。有关更多信息，请参阅 FHIR R4 RESTful API 文档`read`中的。

要读取 FHIR 资源

1. 收集 HealthLake region和datastoreId价值。有关更多信息，请参阅 获取数据存储属性。
2. 确定Resource要读取的 FHIR 类型并收集相关id值。有关更多信息，请参阅 资源类型。
3. 使用收集到的 HealthLake region和值为请求构造一个 URL datastoreId。还要包括 FHIR Resource 类型及其关联id的。要查看以下示例中的整个 URL 路径，请滚动到 “复制” 按钮。

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

4. 发送 请求。FHIR read 交互在 FHIR 授权上使用AWS 签名版本 4 或 SMART 的GET请求。以下curl示例读取了 FHIR Patient 资源的当前状态。 HealthLake要查看整个示例，请滚动到 “复制” 按钮。

   SigV4

   Sigv4 授权

```
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

   该IdentityProviderConfiguration数据类型的 SMART on FHIR 授权示例。

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

```
ehr.example.com/user/introspect\",\"revocation_endpoint\":\"https://
ehr.example.com/user/revoke\",\"code_challenge_methods_supported\":[\"S256\"],
\"capabilities\":[\"launch-ehr\",\"sso-openid-connect\",\"client-public\",
\"permission-v2\"]}"
}
```

调用者可以在授权 lambda 中分配权限。有关更多信息，请参阅 OAuth 2.0 作用域。

AWS Console

1. 登录 HealthLake 控制台上的 “运行查询” 页面。
2. 在 “查询设置” 部分下，进行以下选择。

- 数据存储 ID-选择数据存储 ID 以生成查询字符串。
- 查询类型-选择Read。
- 资源类型-选择要读取的 FHIR 资源类型。
- 资源 ID — 输入 FHIR 资源 ID。

3. 选择运行查询。

阅读 FHIR 资源历史记录

FHIR history 交互会检索数据存储中特定 FHIR 资源的历史记录。 HealthLake 使用此交互，您可以确定 FHIR 资源的内容如何随着时间的推移而发生变化。它还可用于与审计日志协调以查看修改前后的资源状态。FHIR 的交互作用create和delete结果是要保存的资源的历史版本。update有关更多信息，请参阅 FHIR R4 RESTful API 文档history中的。

> **Note**
>
> 您可以选择不使用特定history的 FHIR 资源类型。要选择退出，请使用创建案例AWS Support Center Console。要创建您的案例，请登录您的 AWS 账户 并选择创建案例。

阅读 FHIR 资源历史记录

1. 收集 HealthLake region和datastoreId价值。有关更多信息，请参阅 获取数据存储属性。
2. 确定Resource要读取的 FHIR 类型并收集相关id值。有关更多信息，请参阅 资源类型。

3. 使用收集的 HealthLakeregion和值为请求构造一个 URL datastoreId。还应包括 FHIR Resource 类型、其关联id的和可选的搜索参数。要查看以下示例中的整个 URL 路径，请滚动到“复制”按钮。

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

HealthLake FHIR **history** 交互支持的搜索参数

参数	说明
_count : integer	一页上搜索结果的最大数量。服务器将返回请求的数量或数据存储默认允许的最大搜索结果数，以较低者为准。
_since : instant	仅包括在给定时刻或之后创建的资源版本。
_at : date(Time)	仅包括在日期时间值中指定的时间段内某个时刻处于最新状态的资源版本。有关更多信息，请参阅 HL7 FH RESTful IR API 文档*date*中的。

4. 发送 请求。FHIR history 交互在 FHIR 授权上使用AWS 签名版本 4 或 SMART 的GET请求。以下curl示例使用_count搜索参数为中的 HealthLake FHIR Patient 资源每页返回 100 个历史搜索结果。要查看整个示例，请滚动到“复制”按钮。

SigV4

Sigv4 授权

```
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

该IdentityProviderConfiguration数据类型的 SMART on FHIR 授权示例。

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

调用者可以在授权 lambda 中分配权限。有关更多信息，请参阅 OAuth 2.0 作用域。

history交互的返回内容包含在 FHIR 资源中Bundle，类型设置为。history它包含指定的版本历史记录，按最旧版本排序，并包含已删除的资源。有关更多信息，请参阅 FHIR R4 文档Resource Bundle中的。

读取特定版本的 FHIR 资源历史记录

FHIR vread 交互对数据存储中的资源执行特定版本的读取。 HealthLake 使用此交互，您可以像过去特定时间一样查看 FHIR 资源的内容。

> **Note**
>
> 如果您使用 FHIR `history` 交互而不使用`vread`，则 HealthLake 始终返回资源元数据的最新版本。

HealthLake 声明它支持对每个支持的资源`CapabilityStatement.rest.resource.versioning`进行版本控制。所有资源上的所有 HealthLake 数据存储都包含 `Resource.meta.versionId` (`vid`)。

启用 FHIR `history` 交互时（默认情况下，对于在 2024 年 10 月 25 日之后创建的数据存储或通过请求较旧的数据存储创建的数据存储），`Bundle`响应会将`vid`作为元素的一部分包括在内。`location`在以下示例中，`vid`显示为数字`1`。要查看完整示例，请参阅示例捆绑包/捆绑包响应 (JSON)。

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

阅读特定版本的 FHIR 资源历史记录

1. 收集 HealthLake `region`和`datastoreId`价值。有关更多信息，请参阅 获取数据存储属性。
2. 确定要读取和收集关联`id`的 and `vid` 值的 FHIR `Resource` 类型。有关更多信息，请参阅 资源类型。
3. 使用为 HealthLake 和 FHIR 收集的值为请求构造一个 URL。要查看以下示例中的整个 URL 路径，请滚动到“复制”按钮。

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

4. 发送 请求。FHIR `history` 交互在 FHIR 授权上使用AWS 签名版本 4 或 SMART 的`GET`请求。以下`vread`交互返回单个实例，其中包含为 FHIR `Patient` 资源指定的内容，该版本由指定的资源元数据。`vid`要查看以下示例中的整个 URL 路径，请滚动到“复制”按钮。

   SigV4

   Sigv4 授权

```
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

该IdentityProviderConfiguration数据类型的 SMART on FHIR 授权示例。

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

调用者可以在授权 lambda 中分配权限。有关更多信息，请参阅 OAuth 2.0 作用域。

更新 FHIR 资源

FHIR update 交互为现有资源创建新的当前版本，如果给定id资源尚不存在任何资源，则创建初始版本。有关更多信息，请参阅 FHIR R4 RESTful API 文档update中的。

更新 FHIR 资源

1. 收集 HealthLake `region`和`datastoreId`价值。有关更多信息，请参阅 获取数据存储属性。
2. 确定`Resource`要更新的 FHIR 类型并收集关联`id`值。有关更多信息，请参阅 资源类型。
3. 使用收集到的 HealthLake `region`和值为请求构造一个 URL `datastoreId`。还要包括 FHIR `Resource` 类型及其关联`id`的。要查看以下示例中的整个 URL 路径，请滚动到 “复制” 按钮。

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

4. 为请求构造`JSON`正文，指定要进行的 FHIR 数据更新。出于本过程的目的，请将文件另存为`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. 发送 请求。FHIR `update` 交互在 FHIR 授权上使用AWS 签名版本 4 或 SMART 的`PUT`请求。以下`curl`示例更新了中的`Patient`资源 HealthLake。要查看整个示例，请滚动到 “复制” 按钮。

   SigV4

   Sigv4 授权

```
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
```

如果更新了现有资源，则您的请求将返回 200 HTTP 状态码；如果创建了新资源，则将返回 201 HTTP 状态码。

SMART on FHIR

该IdentityProviderConfiguration数据类型的 SMART on FHIR 授权示例。

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

调用者可以在授权 lambda 中分配权限。有关更多信息，请参阅 OAuth 2.0 作用域。

AWS Console

1. 登录 HealthLake 控制台上的 “运行查询” 页面。

2. 在 “查询设置” 部分下，进行以下选择。

- 数据存储 ID-选择数据存储 ID 以生成查询字符串。
- 查询类型-选择Update （PUT）。
- 资源类型-选择要更新或创建的 FHIR 资源类型。
- 请求正文 — 为请求构建 JSON 正文，指定用于更新资源的 FHIR 数据。

3. 选择运行查询。

根据条件更新 FHIR 资源

条件更新允许您根据某些标识搜索条件而不是逻辑 FHIR id 更新现有资源。当服务器处理更新时，它会使用其标准搜索功能对资源类型执行搜索，目标是解析请求id的单个逻辑。

服务器采取的操作取决于它找到的匹配项数量：

- 没有匹配项，请求正文中未**id**提供匹配项：服务器创建 FHIR 资源。
- 未**id**提供匹配项，且资源尚不存在 **id**：服务器将交互视为 “更新即创建” 交互。
- 没有匹配项，已**id**提供且已存在：服务器以409 Conflict错误拒绝更新。
- One Match，未提供任何资源 OR（**id**提供的资源并且它与找到的资源相匹配）：服务器对匹配的资源执行更新，如上所述，如果资源已更新，则服务器应返回 a 200 OK。id
- One Match，已**id**提供资源但与找到的资源不匹配：服务器返回409 Conflict错误，表明客户端 ID 规范有问题，最好是 OperationOutcome
- 多个匹配项：服务器返回一个412 Precondition Failed错误，表明客户端的标准选择性不够好，最好是 OperationOutcome

以下示例更新了一个名为 peter、出生日期为 2000 年 1 月 1 日、电话号码为 1234567890 的Patient资源。

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

为资源更新配置验证级别

更新 FHIR 资源时，您可以选择指定 x-amzn-healthlake-fhir-validation-level HTTP 标头来配置资源的验证级别。 AWS HealthLake 目前支持以下验证级别：

- `strict`：根据资源的配置文件元素对资源进行验证，如果不存在配置文件，则根据R4规格进行验证。这是的默认验证级别 AWS HealthLake。
- `structure-only`：根据 R4 对资源进行验证，忽略任何引用的配置文件。
- `minimal`：资源经过最低限度的验证，忽略了某些 R4 规则。未通过所需的结构检查的资源 search/analytics 将进行更新，以包括审计警告。

尽管搜索索引需要验证失败，但使用最低验证级别更新的资源仍可能会被提取到数据存储中。在这种情况下，将更新资源，以包括专门用于记录上述故障的 Healthlake 扩展程序：

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

此外，将包含以下 HTTP 响应标头，其值为 “true”：

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

> **Note**
>
> 请注意，如果存在这些错误，则根据 R4 规范提取的数据可能无法按预期进行搜索。

使用 PATCH 操作修改资源

AWS HealthLake 支持 FHIR 资源的 PATCH 操作，使您能够通过将特定元素定位为添加、替换或删除来修改资源，而无需更新整个资源。当您需要执行以下操作时，此操作特别有用：

- 对大型资源进行有针对性的更新
- 减少网络带宽使用量
- 对特定资源元素进行原子修改
- 最大限度地降低覆盖并发更改的风险
- 作为批处理和事务工作流程的一部分更新资源

支持的补丁格式

AWS HealthLake 支持两种标准补丁格式：

JSON 补丁 (RFC 6902)

使用 JSON 指针语法按元素在资源结构中的位置来定位元素。

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

FHIRPath 补丁（FHIR R4 规范）

使用 FHIRPath 表达式根据元素的内容和关系来定位元素，从而提供了 FHIR 原生的修补方法。

Content-Type: `application/fhir+json`

用法

直接补丁操作

可以使用 PATCH HTTP 方法直接在 FHIR 资源上调用 PATCH 操作：

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

捆绑包中的补丁

补丁操作可以作为类型为 FHIR Bundle 的条目包含在内`transaction`，`batch`也可以在单个请求中将补丁操作与其他 FHIR 交互（创建、读取、更新、删除）结合起来。

- 交易捆绑包：所有条目都以原子方式成功或失败
- Batch 捆绑包：每个参赛作品均独立处理

JSON 补丁格式

支持的操作

操作	说明
add	为资源添加新值

操作	说明
remove	从资源中移除一个值
replace	替换资源中的现有值
move	从一个位置移除一个值并将其添加到另一个位置
copy	将值从一个位置复制到另一个位置
test	测试目标位置的值是否等于指定值

路径语法

JSON 补丁使用 JSON 指针语法 (RFC 6901) :

路径示例	说明
/name/0/family	名字的家族元素
/telecom/-	追加到电信阵列
/active	根级活动元素
/address/0/line/1	第一个地址的第二行

示例

包含多个操作的直接 JSON 补丁请求

```
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"
  }
]
```

通过单一操作直接请求 JSON 补丁

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

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

```
  }
]
```

套装中的 JSON 补丁

使用包含 base64 编码的 JSON 补丁负载的二进制资源：

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

FHIRPath 补丁格式

支持的操作

操作	说明
add	向资源添加新元素
insert	在列表的特定位置插入一个元素
delete	从资源中移除元素
replace	替换现有元素的值
move	对列表中的元素重新排序

路径语法

FHIRPath 补丁使用 FHIRPath 表达式，支持：

- 基于索引的访问：`Patient.name[0]`
- 使用以下@@ 方式进行筛选 **`where()`**：`Patient.name.where(use = 'official')`
- 布尔逻辑：`Patient.telecom.where(system = 'phone' and use = 'work')`
- 子集设置函数：`first()`，`last()`
- 存在性检查：`exists()`，`count()`
- 多态导航：`Observation.value`

示例

直接申请 FHIRPath 补丁

```
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 捆绑包中的补丁

使用带有`method: PATCH`以下内容的参数资源作为入口资源：

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

请求标头

标题	必需	描述
Content-Type	是	application/json-patch+json 用于 JSON 补丁或application/fhir+json FHIRPath 补丁
If-Match	否	使用特定版本的条件更新 ETag

示例响应

该操作将返回包含新版本信息的更新资源：

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

{
```

```
  "resourceType": "Patient",
  "id": "example",
  "active": true,
  "name": [
    {
      "family": "Smith",
      "given": ["John"]
    }
  ],
  "telecom": [
    {
      "system": "phone",
      "value": "555-555-5555",
      "use": "home"
    }
  ],
  "meta": {
    "versionId": "2",
    "lastUpdated": "2025-05-05T10:10:10Z"
  }
}
```

行为

补丁操作：

- 根据相应的规范验证补丁语法（JSON 补丁为 RFC 6902，补丁为 FHIR R4） FHIRPath
- 以原子方式应用操作-所有操作成功或全部失败
- 更新资源版本 ID 并创建新的历史记录条目
- 在应用更改之前，将原始资源保留在历史记录中
- 在应用补丁后验证 FHIR 资源限制
- 支持使用带有 If-Match 标头的条件更新 ETag

错误处理

该操作处理以下错误情况：

- 400 错误请求：补丁语法无效（请求不符合要求或补丁文档格式错误）
- 404 未找到：未找到资源（指定的 ID 不存在）

- 409 冲突：版本冲突（并发更新或提供了非当前版本 ID）
- 422 无法处理的实体：补丁操作无法应用于指定的资源元素

功能摘要

能力	json 补丁	FHIRPath 补丁
内容类型	`application/json-patch+json`	`application/fhir+json`
路径格式	JSON Pointer (RFC 6901)	FHIRPath 表达式
直接补丁 API	支持	支持
捆绑包 Batch	支持（通过二进制）	支持（通过参数）
捆绑交易	支持（通过二进制）	支持（通过参数）
操作	添加、删除、替换、移动、复制、测试	添加、插入、删除、替换、移动

限制

- 不支持使用搜索条件的条件补丁操作
- 捆绑包中的 JSON 补丁必须使用带有 base64 编码内容的二进制资源
- FHIRPath 捆绑包中的补丁必须使用参数资源

其他资源

有关 PATCH 操作的更多信息，请参阅：

- FHIR R4 补丁文档
- FHIR R4 补丁规范 FHIRPath
- RFC 6902-JSON 补丁
- RFC 6901-JSON Pointer

捆绑 FHIR 资源

FHIR Bundle 是存放 FHIR 资源集合的容器。 AWS HealthLake AWS HealthLake 支持两种具有不同处理行为的捆绑包。

Batch捆绑包独立处理每种资源。如果一个资源出现故障，其余资源仍然可以成功。每个操作都是单独处理的，即使某些操作失败，处理也会继续进行。在可以接受部分成功的情况下，使用批量捆绑包进行批量操作，例如上传多份无关的患者记录。

Transactionbundle 以原子方式将所有资源作为一个单元处理。要么所有资源操作都成功，要么都不 AWS HealthLake 提交。当您需要保证相关资源的参照完整性时，请使用交易捆绑包，例如创建具有相关观察结果和病症的患者，其中必须将所有数据记录在一起。

批量捆绑包和交易捆绑包之间的区别

功能	Batch	事务
处理模型	每个操作的成功或失败都是独立的。	所有操作都以单个原子单元的形式成功或失败。
故障处理	即使个别操作失败，处理也会继续。	如果任何一个操作失败，则整个捆绑包将失败。
执行顺序	不能保证执行顺序。	操作按指定的顺序处理。
参照完整性	未在所有操作中强制执行。	对捆绑包中本地引用的资源强制执行。
最适合用于	可以接受部分成功的批量操作。	必须一起创建或更新的相关资源。

您可以捆绑相同或不同类型的 FHIR 资源，它们可以混合使用 FHIR 操作，例如、create、readupdatedelete、和。patch有关更多信息，请参阅 FHIR R4 文档中的资源包。

以下是每种捆绑包类型的示例用例。

Batch 捆绑包

- 在夜间数据同步期间，上传来自不同设施的多份无关患者记录。
- 批量上传历史用药记录，其中一些记录可能存在验证问题。
- 加载参考数据，例如组织和从业人员，其中个人失败不会影响其他条目。

交易捆绑包

- 在急诊室住院期间，创建具有相关观察结果和状况的患者，所有数据都必须一起记录。
- 更新患者的药物清单和相关的过敏信息，这些信息必须保持一致。
- 将与患者的完整相遇、观察结果、程序和账单信息记录为单个原子单元。

> **Important**
>
> 批处理和交易捆绑包使用相同的Bundle资源结构。唯一的区别是type字段的值。

以下示例显示了具有多种资源类型和操作的事务捆绑包。

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

将 FHIR 资源捆绑为独立实体

将 FHIR 资源捆绑为独立实体

1. 收集 HealthLake `region`和`datastoreId`价值。有关更多信息，请参阅 获取数据存储属性。
2. 使用收集的 HealthLake`region`和值为请求构造一个 URL `datastoreId`。请勿在 URL 中指定 FHIR 资源类型。要查看以下示例中的整个 URL 路径，请滚动到“复制”按钮。

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

3. 为请求构造一个 JSON 正文，将每个 HTTP 动词指定为`method`元素的一部分。以下示例使用与`Bundle`资源的`batch`类型交互来创建新的`Patient`和`Medication`资源。所有必填部分都有相应的注释。出于本过程的目的，请将文件另存为`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. 发送 请求。FHIR Bundle 批处理类型使用具有AWS 签名版本 4 或 FHIR 授权的 SMART POST 请求。以下代码示例使用curl命令行工具进行演示。

SigV4

Sigv4 授权

```
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 on FHIR IdentityProviderConfiguration数据类型的授权示例。

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

调用者可以在授权 lambda 中分配权限。有关更多信息，请参阅 OAuth 2.0 作用域。

服务器返回一个响应，其中显示Patient了作为Bundle批处理类型请求的结果而创建的和Medication资源。

以捆绑包 PUTs 为条件

AWS HealthLake 支持使用以下查询参数在捆绑包中进行条件更新：

> **Note**
>
> 只有batch捆绑包中才支持 PUTs 有条件的。 Transaction捆绑包不支持有条件的 PUTs。

- _id（独立）
- _id与以下选项之一相结合：
  - _tag
  - _createdAt
  - _lastUpdated

当您在捆绑包 PUTs 中使用条件时，会根据现有资源 AWS HealthLake 评估查询参数，并根据匹配结果采取行动。

有条件的更新行为

场景	HTTP 状态	已采取的行动
提供没有 ID 的资源	201 已创建	始终创建新资源。
具有新 ID 的资源（不匹配）	201 已创建	使用指定 ID 创建新资源。
具有现有 ID 的资源（单一匹配）	200 OK (200 确定)	更新匹配的资源。
具有现有 ID 的资源（检测到冲突）	409 冲突	返回错误。未进行任何更改。
具有现有 ID 的资源（ID 不匹配）	400 错误请求	返回错误。未进行任何更改。
多个资源匹配条件	412 前提条件失败	返回错误。未进行任何更改。

在以下带有条件更新的捆绑包示例中，带有 FHIR ID 的Patient资源仅在满足条件_lastUpdated=lt2025-04-20时才会476更新。

```
{
```

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

将 FHIR 资源捆绑为一个实体

将 FHIR 资源捆绑为单个实体

1. 收集 HealthLake `region`和`datastoreId`价值。有关更多信息，请参阅 获取数据存储属性。
2. 使用收集的 HealthLake`region`和值为请求构造一个 URL `datastoreId`。将 FHIR 资源类型`Bundle`作为网址的一部分。要查看以下示例中的整个 URL 路径，请滚动到 “复制” 按钮。

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

3. 为请求构建 JSON 正文，指定要组合在一起的 FHIR 资源。以下示例将两个`Patient`资源分组在一起 HealthLake。出于本过程的目的，请将文件另存为`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. 发送 请求。FHIR Bundle 文档类型使用带有AWS 签名版本 4 签名协议的POST请求。以下代码示例使用curl命令行工具进行演示。

SigV4

Sigv4 授权

```
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 on FHIR IdentityProviderConfiguration数据类型的授权示例。

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

调用者可以在授权 lambda 中分配权限。有关更多信息，请参阅 OAuth 2.0 作用域。

服务器返回一个响应，其中显示了由于Bundle文档类型请求而创建的两个Patient资源。

为捆绑包配置验证级别

捆绑 FHIR 资源时，您可以选择指定 `x-amzn-healthlake-fhir-validation-level` HTTP 标头来配置资源的验证级别。将为捆绑包中的所有创建和更新请求设置此验证级别。 AWS HealthLake 目前支持以下验证级别：

- `strict`：根据资源的配置文件元素对资源进行验证，如果不存在配置文件，则根据R4规格进行验证。这是的默认验证级别 AWS HealthLake。
- `structure-only`：根据 R4 对资源进行验证，忽略任何引用的配置文件。
- `minimal`：资源经过最低限度的验证，忽略了某些 R4 规则。未通过所需的结构检查的资源 search/analytics 将进行更新，以包括审计警告。

尽管搜索索引需要验证失败，但与最低验证级别捆绑在一起的资源仍可能会被提取到数据存储中。在这种情况下，资源将更新为包含用于记录上述失败的 Healthlake 专用扩展程序，Bundle 响应中的条目将包括以下 OperationOutcome 资源：

```
{
    "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."
                        }
```

```
                    ]
                }
            }
        }
    ]
}
```

此外，将包含以下 HTTP 响应标头，其值为 “true”：

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

> **Note**
>
> 请注意，如果存在这些错误，则根据 R4 规范提取的数据可能无法按预期进行搜索。

对捆绑包类型 “消息” 的支持有限

HealthLake `message`通过内部转换过程对 FHIR 捆绑包类型提供有限的支持。此支持专为无法在源头重新格式化消息包的场景而设计，例如从传统医院系统中摄取 ADT（入院、出院、转移）提要。

> **Warning**
>
> 此功能需要明确的 AWS 账户许可名单，并且不强制执行 FHIR R4 消息语义或引用完整性。在使用消息包之前，请联系 Su AWS pport 申请启用您的账户。

与标准邮件处理的主要区别

- 消息包（FHIR 规范）：第一个条目必须是引用其他`MessageHeader`资源的条目。资源缺少单个`request`对象， MessageHeader 事件决定处理操作。
- HealthLake 处理：通过自动为每个资源条目分配 PUT 操作，将消息包转换为批量捆绑包。资源是独立处理的，不强制执行消息语义或引用完整性。

重要限制

- 未强制执行 FHIR R4 邮件特定的处理规则

- 资源之间没有交易完整性
- 资源间引用未经过验证
- 需要明确的账户许可名单

消息包结构示例

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

> **Note**
>
> 每个资源都是独立存储的，就像通过单独的 PUT 操作提交一样。如果需要完整的 FHIR 消息语义或参照完整性验证，请在提交之前对消息包进行预处理或实现应用程序级验证。

异步捆绑交易

AWS HealthLake 支持异步Bundle类型transaction，允许您提交包含最多 500 个资源的交易。当您提交异步事务时，会将其 HealthLake 排队等候处理，并立即返回轮询网址。您可以使用此 URL 来检查状态并检索响应。这遵循 FHIR 异步捆绑包模式。

何时使用异步事务

- 您需要在单笔交易中提交 100 个以上的资源（同步限制）。
- 您希望避免在等待事务处理完成时阻塞应用程序。
- 您需要以更高的吞吐量处理大量相关资源。

> **Important**
>
> 投票结果将在交易完成后的90天内公布。在这90天之后，轮询网址将不再返回结果。设计您的集成，以便在此窗口中检索和存储结果。

> **Note**
>
> 同步Bundle类型transaction继续支持多达 100 个资源，并且是默认的处理模式。如果您提交的资源超过 100 个transaction，但没有Prefer: respond-async标题的Bundle类型， HealthLake 则会返回422 Unprocessable Entity错误。异步处理batch不支持带类型的捆绑包，只能异步提交Bundle类型transaction（最多有 500 个操作）。

> **Note**
>
> PATCH异步捆绑交易中 PUTs 不支持运算和条件。

提交异步事务

要提交异步事务，请向数据存储端点发送带有Prefer: respond-async标头的POST请求。捆绑包必须有类型transaction。异步分发包处理batch不支持类型为 bundle。

HealthLake 在提交时对捆绑包进行初始验证。如果验证成功，则 HealthLake 返回 HTTP 202 Accepted，并返回包含轮询网址的`content-location`响应标头。

提交异步**Bundle**类型 **transaction**

1. 向 HealthLake 数据存储端点发送POST请求。

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

2. 使用捆绑包类型为请求构造一个 JSON 正文`transaction`。出于本过程的目的，请将文件另存为`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. 发送带有`Prefer: respond-async`标题的请求。FHIR `Bundle` 交易类型使用具有AWS 签名版本 4 或 FHIR 授权的 SMART POST 请求。以下代码示例使用`curl`命令行工具进行演示。

   SigV4

   Sigv4 授权

   ```
   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 on FHIR `IdentityProviderConfiguration`数据类型的授权示例。

   ```
   {
       "AuthorizationStrategy": "SMART_ON_FHIR",
       "FineGrainedAuthorizationEnabled": true,
       "IdpLambdaArn": "arn:aws:lambda:your-region:your-account-id:function:your-
   lambda-name",
   ```

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

调用者可以在授权 lambda 中分配权限。有关更多信息，请参阅 OAuth 2.0 作用域。

4. 成功提交后，服务器返回 HTTP 202 已接受。content-location响应标头包含轮询网址。响应正文是一种OperationOutcome资源。

```
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"
            ]
        }
    ]
}
```

轮询交易状态

提交异步事务后，使用content-location响应标头中的轮询网址来检查交易状态。向投票 URL 发送GET请求。

> **Note**
>
> 对于启用 SMART on FHIR 的数据存储，授权令牌必须包括对Transaction资源类型的read权限，以轮询交易状态。有关 FHIR 作用域上的 SMART 的更多信息，请参阅SMART on FHIR OAuth 2.0 瞄准镜支持 HealthLake。

向投票 URL 发送GET请求。以下示例使用curl命令行工具。

SigV4

Sigv4 授权

```
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 在 FHIR 授权上。授权令牌必须包含Transaction资源类型的read权限。

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

下表描述了可能的响应。

轮询响应码

HTTP 状态	含义	响应正文
202 已接受	交易已排队	OperationOutcome 诊断 “已提交”
202 已接受	交易正在处理中	OperationOutcome 诊断 “进行中”
200 OK (200 确定)	交易成功完成	Bundle带类型 transaction-response
4xx/5xx	交易失败	OperationOutcome 附有错误详情

以下示例显示了每种响应类型。

交易已排队 (202)

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

交易处理 (202)

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

```
    ]
}
```

交易已完成 (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"
            }
        }
    ]
}
```

交易失败 (4xx/5xx)

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

处理订单

类型的异步捆绑包`transaction`已排队，但不会按照严格的提交顺序进行处理。 HealthLake 根据可用容量和系统负载优化处理。

> **Important**
>
> 不要依赖交易按提交顺序进行处理。例如，如果您在上午 10:00 提交交易 A，在上午 10:01 提交交易 B，则事务 B 可能会在交易 A 之前完成。将您的申请设计为：
>
> - 处理 out-of-order完成。
> - 使用轮询网址独立跟踪每笔交易。
> - 如果顺序对您的用例很重要，请实施应用程序级排序。

配额和限制

以下配额和速率限制适用于异步交易。

异步交易配额

配额	值	可调整
每个异步事务的最大操作数	500	否
每个数据存储的最大待处理事务数	500	是

- 异步交易共享中定义的相同 API 速率限制服务限额。
- 轮询交易状态与 FHIR 资源上的 read (GET) 操作具有相同的 API 速率限制。
- 如果达到待处理交易限额，则后续提交将返回错误，直到现有交易完成。

错误处理

对于 “事务” 捆绑包，捆绑包中包含的所有 FHIR 资源都将作为原子操作进行处理。操作中的所有资源都必须成功，否则不处理捆绑包中的任何操作。

错误分为两类：同步 HealthLake 返回的提交错误和通过轮询检索的处理错误。

提交错误

HealthLake 在提交时验证捆绑包，并在事务排队之前同步返回错误。提交错误包括无效的 FHIR 资源验证错误、不支持的资源类型、超过 500 个操作限制以及将`Prefer: respond-async`标题与批处理捆绑包一起使用。如果已达到数据存储的待处理事务限制，则 HealthLake 返回 a `ThrottlingException`。当发生提交错误时，交易将不会排队。

处理错误

处理错误是在事务排队之后发生的，并通过轮询 URL 返回。其中包括事务冲突，即另一个操作修改了作为事务一部分的资源，以及处理期间的服务器错误。发生处理错误时，不会对事务中的资源进行资源变更。轮询网址将返回`OperationOutcome`包含错误详情的。

删除 FHIR 资源

FHIR `delete` 交互会将现有的 FHIR 资源从 HealthLake数据存储中移除。有关更多信息，请参阅 FHIR R4 RESTful API 文档`delete`中的。

删除 FHIR 资源

1. 收集 HealthLake `region`和`datastoreId`价值。有关更多信息，请参阅 获取数据存储属性。
2. 确定`Resource`要删除的 FHIR 类型并收集关联`id`值。有关更多信息，请参阅 资源类型。
3. 使用收集到的 HealthLake `region`和值为请求构造一个 URL `datastoreId`。还要包括 FHIR `Resource` 类型及其关联`id`的。要查看以下示例中的整个 URL 路径，请滚动到 “复制” 按钮。

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

4. 发送 请求。FHIR `delete` 交互在 FHIR 授权上使用AWS 签名版本 4 或 SMART 的`DELETE`请求。以下`curl`示例从 HealthLake 数据存储中移除现有 FHIR `Patient` 资源。要查看整个示例，请滚动到 “复制” 按钮。

   SigV4

   Sigv4 授权

   ```
   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'
```

服务器返回一个 204 HTTP 状态码，确认资源已从 HealthLake 数据存储中删除。如果删除请求失败，您将收到一400系列 HTTP 状态代码，说明请求失败的原因。

SMART on FHIR

该IdentityProviderConfiguration数据类型的 SMART on FHIR 授权示例。

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

调用者可以在授权 lambda 中分配权限。有关更多信息，请参阅 OAuth 2.0 作用域。

AWS Console

1. 登录 HealthLake 控制台上的 “运行查询” 页面。

2. 在 “查询设置” 部分下，进行以下选择。

- 数据存储 ID-选择数据存储 ID 以生成查询字符串。
- 查询类型-选择Delete。
- 资源类型-选择要删除的 FHIR 资源类型。
- 资源 ID — 输入 FHIR 资源 ID。

3. 选择运行查询。

根据条件删除 FHIR 资源

当您不知道特定的 FHIR 资源 ID 但有关于要删除的资源的其他识别信息时，条件删除特别有用。

有条件删除允许您根据搜索标准而不是逻辑 FHIR ID 删除现有资源。当服务器处理删除请求时，它会使用标准搜索功能对资源类型执行搜索，以解析请求的单个逻辑 ID。

有条件删除的工作原理

服务器的操作取决于它找到的匹配项数量：

1. 无匹配项：服务器尝试普通删除并做出相应响应（404 表示资源不存在，204 表示已删除的资源为 No Content）
2. 一场匹配：服务器对匹配的资源执行普通删除
3. 多个匹配项：返回 412 Precondition Failed 错误，表示客户端的标准选择性不足

响应场景

AWS HealthLake 使用以下响应模式处理有条件的删除操作：

成功运营

- 当您的搜索条件成功识别出单个活动资源时，系统将在完成删除后返回 204 No Conten t，就像标准删除操作一样。

基于身份的有条件删除

根据`id`附加参数（`createdAttag`、或`_lastUpdated`）执行有条件删除时：

- 204 无内容：资源已被删除
- 404 未找到：资源不存在
- 409 冲突：ID 匹配但其他参数不匹配

Non-ID-Based 有条件删除

当id未提供或使用createdAttag、或之外的参数时_lastUpdated：

- 404 未找到：未找到匹配项

冲突局势

有几种情况会导致 412 个先决条件失败的响应：

- 多个资源与您的搜索条件相匹配（条件不够具体）
- 将 ETag 标头与一起使用时会发生版本冲突 If-Match
- 在搜索和删除操作之间发生资源更新

成功执行有条件删除的示例

以下示例根据特定标准删除患者资源：

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

此请求会删除患者资源，其中：

- 名字叫 “彼得”
- 出生日期是 2000 年 1 月 1 日
- 电话号码是 1234567890

最佳实践

1. 使用特定的搜索条件来避免多个匹配项并防止 412 错误。
2. 当需要处理并发修改时，可以考虑使用 ETag 标头进行版本控制。
3. 适当处理错误响应：
   - 对于 404：优化搜索条件
   - 对于 412：使标准更加具体或解决版本冲突
4. 为高并发环境中的时间冲突做好准备，在这种环境中，可能会在搜索和删除操作之间修改资源。

等性与并发性

等能键

AWS HealthLake 支持 FHIR P0ST 操作的等效密钥，提供了一种强大的机制来确保资源创建期间的数据完整性。通过在请求标头中包含唯一的 UUID 作为等性密钥，医疗保健应用程序可以保证每个 FHIR 资源只创建一次，即使在涉及网络不稳定或自动重试的场景中也是如此。

此功能对于医疗保健系统尤其重要，因为在这些系统中，重复的病历可能会造成严重后果。当收到的请求与之前的请求具有相同的等效键时， HealthLake 将返回原始资源，而不是创建重复的资源。例如，这可能发生在重试循环中，或者由于请求管道冗余。使用等性键可以保持数据一致性 HealthLake ，同时为处理间歇性连接问题的客户端应用程序提供无缝体验。

实施

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

响应场景

第一个请求（已创建 201 个）

- 新资源已成功创建
- 响应包含资源 ID

重复请求（409 冲突）

- 检测到相同的等性密钥
- 已返回原始资源
- 未创建任何新资源

请求无效（400 错误请求）

- 格式错误的 UUID
- 缺少必填字段

最佳实践

- 为每个新创建的资源生成唯一的 UUID
- 存储用于重试逻辑的等性键
- 使用一致的密钥格式：推荐 UUID v4
- 在处理资源创建的客户端应用程序中实现

> **Note**
>
> 对于要求严格数据准确性并防止重复医疗记录的医疗保健系统而言，此功能特别有用。

ETag 在 AWS 中 HealthLake

AWS HealthLake ETags 用于对 FHIR 资源进行乐观的并发控制，为管理并发修改和维护数据一致性提供了一种可靠的机制。 ETag是代表资源的特定版本的唯一标识符，通过 HTTP 标头充当版本控制系统。在读取或修改资源时，应用程序可以使用 ETags 来防止意外重写并确保数据完整性，尤其是在可能并发更新的情况下。

实现示例

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

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

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

响应场景

成功操作（200 OK 或 204 无内容）

- ETag 匹配当前版本
- 操作按预期进行

版本冲突（412 前提条件失败）

- ETag 与当前版本不匹配
- 为了防止数据丢失，更新被拒绝

最佳实践

- 包含 ETags 在所有更新和删除操作中
- 实现用于处理版本冲突的重试逻辑
- 使用 If-None-Match:* 用于 create-if-not-exists场景
- 修改前务必验证 ETag 新鲜度

这种并发控制系统对于维护医疗保健数据的完整性至关重要，尤其是在有多个用户或系统访问和修改相同资源的环境中。

在中搜索 FHIR 资源 AWS HealthLake

使用 FHIR search交互根据某些筛选条件在 HealthLake 数据存储中搜索一组 FHIR 资源。可以使用GET或POST请求执行search交互。对于涉及个人身份信息 (PII) 或受保护的健康信息 (PHI) 的搜索，建议使用POST请求，因为 PII 和 PHI 已添加为请求正文的一部分，并在传输过程中进行加密。

> **Note**
>
> 本章中描述的 FHIR search 交互符合 HL7 FHIR R4 医疗保健数据交换标准。由于它代表 HL7 FHIR 服务，因此不是通过 AWS CLI 和 AWS SDKs提供的。有关更多信息，请参阅 FHIR R4 RESTful API 文档search中的。

HealthLake 支持 FHIR R4 搜索参数的子集。有关更多信息，请参阅 FHIR R4 的搜索参数为 HealthLake。

主题

- 使用 GET 搜索 FHIR 资源
- 使用 POST 搜索 FHIR 资源
- FHIR 搜索一致性级别

使用 GET 搜索 FHIR 资源

您可以使用GET请求来搜索 HealthLake 数据存储。使用时GET， HealthLake 支持将搜索参数作为网址的一部分提供，但不支持作为请求正文的一部分。有关更多信息，请参阅 FHIR R4 的搜索参数为 HealthLake。

> **重要提示**
>
> 对于涉及个人身份信息 (PII) 或受保护的健康信息 (PHI) 的搜索，安全最佳实践要求使用POST请求，因为 PII 和 PHI 是作为请求正文的一部分添加的，并在传输过程中进行加密。有关更多信息，请参阅 使用 POST 搜索 FHIR 资源。

以下过程之后是用于GET搜索 HealthLake 数据存储的示例。

使用搜索 HealthLake 数据存储 **GET**

1. 收集 HealthLake `region`和`datastoreId`价值。有关更多信息，请参阅 获取数据存储属性。
2. 确定要搜索和收集关联`id`值的 FHIR 资源类型。有关更多信息，请参阅 资源类型。
3. 使用收集的 HealthLake`region`和值为请求构造一个 URL `datastoreId`。还应包括 FHIR `Resource` 类型和支持的搜索参数。要查看以下示例中的整个 URL 路径，请滚动到 “复制” 按钮。

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

4. 使用AWS 签名版本 4 或 FHIR 授权的 SMART 发送`GET`请求。以下`curl`示例返回 HealthLake 数据存储中的`Patient`资源总数。要查看整个示例，请滚动到 “复制” 按钮。

   SigV4

   Sigv4 授权

```
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 on FHIR `IdentityProviderConfiguration`数据类型的授权示例。

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

```
ehr.example.com/user/manage\",\"introspection_endpoint\":\"https://
ehr.example.com/user/introspect\",\"revocation_endpoint\":\"https://
ehr.example.com/user/revoke\",\"code_challenge_methods_supported\":[\"S256\"],
\"capabilities\":[\"launch-ehr\",\"sso-openid-connect\",\"client-public\",
\"permission-v2\"]}"
}
```

调用者可以在授权 lambda 中分配权限。有关更多信息，请参阅 OAuth 2.0 作用域。

AWS Console

> **Note**
>
> HealthLake 控制台仅支持 SigV4 授权。通过 AWS CLI 和 AWS SDKs支持 FHIR 上的 SMART 授权。

1. 登录 HealthLake控制台上的 “运行查询” 页面。
2. 在 “查询设置” 部分下，进行以下选择。
   - 数据存储 ID-选择数据存储 ID 以生成查询字符串。
   - 查询类型-选择`Search with GET`。
   - 资源类型-选择要搜索的 FHIR 资源类型。
   - 搜索参数-选择搜索参数或搜索参数组合，将查询重点放在特定的记录上。
3. 选择运行查询。

示例：使用 GET 进行搜索

以下选项卡提供了使用搜索特定 FHIR 资源类型的示例。`GET`这些示例说明了如何在请求中指定搜索参数 URLs。

> **Note**
>
> HealthLake 控制台仅支持 SigV4 授权。通过 AWS CLI 和 AWS SDKs支持 FHIR 上的 SMART 授权。
> HealthLake 支持 FHIR R4 搜索参数的子集。有关更多信息，请参阅 搜索参数。

Patient (age)

尽管年龄不是在 FHIR 中定义的资源类型，但它被捕获为Patient资源类型中的一个元素。使用以下示例对Patient资源类型提出GET基于搜索的请求，使用 B irthD ate 元素和eq搜索比较器来搜索 1997 年出生的个人。

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

Condition

使用以下示例对Condition资源类型发出GET请求。搜索会在您的 HealthLake 数据存储中查找包含 SNOMED 医疗代码的条件72892002，该代码转换为。Normal pregnancy

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

DocumentationReference

以下示例说明如何根据DocumentReference资源类型为诊断为Patient链球菌且同时服用阿莫西林处方的人创建GET请求。

```
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

使用以下示例对Location资源类型发出GET请求。以下搜索将在您的 HealthLake 数据存储中查找地址中包含城市名称波士顿的位置。

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

Observation

使用以下示例对Observation资源类型发出GET基于搜索的请求。此搜索使用value-concept搜索参数来查找医疗代码266919005，该代码转换为。Never smoker

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

使用 POST 搜索 FHIR 资源

您可以使用 FHIR 与POST请求的search交互来搜索HealthLake 数据存储。使用时POST， HealthLake 支持在 URL 或请求正文中使用搜索参数，但不能在单个请求中同时使用这两个参数。

> **重要提示**
>
> 对于涉及个人身份信息 (PII) 或受保护的健康信息 (PHI) 的搜索，安全最佳实践要求使用POST请求，因为 PII 和 PHI 是作为请求正文的一部分添加的，并在传输过程中进行加密。

以下过程是使用 FHIR R4 与的search交互POST来搜索 HealthLake 数据存储的示例。这些示例展示了如何在 JSON 请求正文中指定搜索参数。

使用搜索 HealthLake 数据存储 **POST**

1. 收集 HealthLake region和datastoreId价值。有关更多信息，请参阅 获取数据存储属性。
2. 确定要搜索和收集关联id值的 FHIR 资源类型。有关更多信息，请参阅 资源类型。
3. 使用收集到的 HealthLakeregion和值为请求构造一个 URL datastoreId。还包括 FHIR Resource 类型和_search交互作用。要查看以下示例中的整个 URL 路径，请滚动到 “复制” 按钮。

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

4. 为请求构建 JSON 正文，指定要搜索的 FHIR 数据。出于此手术的目的，您将搜索Observation资源以发现从未吸烟的患者。要指定医疗代码状态Never smoker，请在 JSON 请求正文value-concept=266919005中进行设置。将该文件保存为 search-observation.json。

   ```
   value-concept=266919005
   ```

5. 发送 请求。FHIR search 交互使用带有AWS 签名版本 4 或 SMART 的 FHIR 授权GET请求。

   > **Note**
   >
   > 在POST请求正文中使用搜索参数发出请求时，请Content-Type: application/x-www-form-urlencoded将其用作标头的一部分。

以下curl示例对Observation资源类型发出基于 Post的搜索请求。该请求使用value-concept搜索参数来查找表示值的医疗代码266919005Never smoker。要查看整个示例，请滚动到“复制”按钮。

SigV4

Sigv4 授权

```
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

该IdentityProviderConfiguration数据类型的 SMART on FHIR 授权示例。

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

```
}
```

调用者可以在授权 lambda 中分配权限。有关更多信息，请参阅 OAuth 2.0 作用域。

示例：使用 POST 进行搜索

以下选项卡提供了使用搜索特定 FHIR 资源类型的示例。POST这些示例说明了如何在中指定请求 URLs。

> **Note**
>
> HealthLake 控制台仅支持 SigV4 授权。通过 AWS CLI 和 AWS SDKs支持 FHIR 上的 SMART 授权。
> HealthLake 支持 FHIR R4 搜索参数的子集。有关更多信息，请参阅 搜索参数。

Patient (age)

尽管年龄不是在 FHIR 中定义的资源类型，但它被捕获为`Patient`资源类型中的一个元素。使用以下示例对`Patient`资源类型发出`POST`基于搜索的请求。以下搜索示例使用搜索比较器来`eq`搜索 1997 年出生的个人。

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

要在搜索中指定 1997 年，请在请求正文中添加以下元素。

```
birthdate=eq1997
```

Condition

使用以下内容对`Condition`资源类型`POST`提出请求。此搜索可在您的 HealthLake 数据存储中查找包含医疗代码的位置`72892002`。

您必须指定请求网址和请求正文。以下是请求网址的示例。

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

要指定要搜索的医疗代码，请在请求正文中添加以下 JSON 元素。

```
code=72892002
```

DocumentReference

要在对 HealthLakeDocumentReference资源类型POST提出请求时查看集成自然语言处理 (NLP) 的结果，请按以下方式格式化请求。

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

要指定要引用的DocumentReference搜索参数，请参阅搜索参数类型。以下查询字符串使用多个搜索参数搜索用于生成集成 NLP 结果的 Amazon Comprehend Medical API 操作。

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

Location

使用以下示例对Location资源类型发出POST请求。搜索会在您的HealthLake 数据存储中查找地址中包含城市名称波士顿的位置。

您必须指定请求 URL 和请求正文。以下是请求网址的示例。

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

要在搜索Boston中指定，请在请求正文中添加以下元素：

```
address=Boston
```

Observation

使用以下示例对Observation资源类型发出POST基于搜索的请求。搜索使用value-concept搜索参数来查找医疗代码，266919005这表示出来Never smoker。

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

要指定状态Never smoker，请在 JSON 的正文value-concept=266919005中进行设置。

```
value-concept=266919005
```

FHIR 搜索一致性级别

AWS HealthLake 的搜索索引在搜索GET操作的最终一致性模型上运行。POST为了最终保持一致性，如果某个资源的搜索索引有待更新，则在索引更新完成之前，搜索结果将排除该资源的 N-1 版本。

AWS HealthLake 现在可以选择更新后的资源的一致性模型的行为方式。开发人员可以包含 “最终一致性”（上述默认行为）或 “强一致性”。强一致性将允许搜索索引更新待处理的资源的 N-1 版本包含在搜索结果中。这可用于即使搜索索引更新尚未完成也需要在结果中使用所有资源的用例场景。客户可以使用x-amz-fhir-history-consistency-level请求标头控制这种行为。

一致性级别

强一致性

设置为x-amz-fhir-history-consistency-level: strong返回所有匹配的记录，包括那些有待更新的搜索索引的记录。当需要在更新后立即搜索资源时，请使用此选项。

最终一致性

设置x-amz-fhir-history-consistency-level: eventual为仅返回已完成搜索索引更新的记录。如果未指定一致性级别，则这是默认行为。

用法示例

1. 更新资源时：

```
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. 后续搜索：

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

最佳实践

- 当需要立即搜索最近更新的资源时，请使用强一致性
- 在即时可见性并不重要的一般查询中，使用最终一致性
- 考虑一下即时可见性和潜在性能影响之间的权衡

> **Note**
>
> 一致性级别设置会影响更新的资源在搜索结果中的显示速度，但不会影响资源的实际存储。
> 将可选标`x-amz-fhir-history-consistency-level`头设置为 “strong” 会使每个资源的写入容量消耗增加一倍。
> 此功能仅适用于启用了版本历史记录的数据存储（默认情况下，在 2024 年 10 月 25 日之后创建的所有数据存储均启用该功能）。

使用导出 FHIR 数据 AWS HealthLake

使用本机 AWS HealthLake 操作启动、描述和列出 FHIR 导出作业。您可以对导出任务进行排队。异步导出任务以 FIFO（先入先出）方式处理。您可以像启动导出任务一样对任务进行排队。如果正在进行中，则只需排队即可。在导出任务进行期间，您可以创建、读取、更新或删除 FHIR 资源。

> **Note**
>
> 您也可以使用 FHIR R `$export` 4 操作从 HealthLake 数据存储中导出 FHIR 数据。有关更多信息，请参阅 使用 FHI HealthLake R 导出数据 `$export`。

主题

- 启动 FHIR 导出任务
- 获取 FHIR 导出任务属性
- 列出 FHIR 导出任务

启动 FHIR 导出任务

`StartFHIRExportJob`用于从 HealthLake 数据存储启动 FHIR 导出作业。以下菜单提供了操作步骤 AWS 管理控制台 和 AWS CLI 和的代码示例 AWS SDKs。有关更多信息，请参阅《AWS HealthLake API Reference》中的 `StartFHIRExportJob`。

> **备注**
>
> HealthLake 支持用于医疗保健数据交换的 FHIR R4 规范。因此，所有健康数据都以 FHIR R4 格式导出。

启动 FHIR 导出作业

根据您的访问偏好选择菜单 AWS HealthLake。

AWS CLI 和 SDKs

CLI

AWS CLI

启动 FHIR 导出作业

以下`start-fhir-export-job`示例说明如何使用 AWS HealthLake启动 FHIR 导出任务。

```
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)
```

输出：

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

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 “启动 FHIRExport J ob”。

Python

适用于 Python 的 SDK（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
```

- 有关 API 的详细信息，请参阅 Python 版AWS SDK 中启动FHIRExport作业 (Boto3) API 参考。

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅在 SAP 的 AWS SDK 中启动 FHIRExport J ob ABAP API 参考。

> **示例可用性**
>
> 找不到所需的内容？ 使用本页右侧边栏上的 “提供反馈” 链接请求代码示例。

AWS 控制台

1. 登录 HealthLake 控制台上的数据存储页面。
2. 选择数据存储。
3. 选择导出。

   将打开 “导出” 页面。
4. 在 “输出数据” 部分下，输入以下信息：
   - 输出数据在 Amazon S3 中的位置
   - 输出加密
5. 在访问权限部分下，选择使用现有 IAM 服务角色并从角色名称菜单中选择该角色或选择创建 IAM 角色。
6. 选择导出数据。

> **Note**
>
> 在导出过程中，在页面顶部的横幅上选择 “复制作业 ID”。您可以使用JobID来请求导出任务属性 AWS CLI。有关更多信息，请参阅 获取 FHIR 导出任务属性。

获取 FHIR 导出任务属性

DescribeFHIRExportJob用于从 HealthLake 数据存储中获取导出任务属性。以下菜单提供了操作步骤 AWS 管理控制台 和 AWS CLI 和的代码示例 AWS SDKs。有关更多信息，请参阅《AWS HealthLake API Reference》中的 DescribeFHIRExportJob。

> **备注**
>
> HealthLake 支持用于医疗保健数据交换的 FHIR R4 规范。因此，所有健康数据都以 FHIR R4 格式导出。

描述 FHIR 导出作业

根据您的访问偏好选择菜单 AWS HealthLake。

AWS CLI 和 SDKs

CLI

AWS CLI

描述 FHIR 导出作业

以下describe-fhir-export-job示例说明如何在中查找 FHIR 导出任务的 AWS HealthLake属性。

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

输出：

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

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 “描述 FHIRExport J ob”。

Python

适用于 Python 的 SDK（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
```

- 有关 API 的详细信息，请参阅 Python 版AWS SDK 中描述FHIRExport作业 (Boto3) API 参考。

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅在 SAP 的 AWS SDK 中描述 FHIRExport J ob ABAP API 参考。

> **示例可用性**
>
> 找不到所需的内容？ 使用本页右侧边栏上的 “提供反馈” 链接请求代码示例。

AWS 控制台

> **Note**
>
> HealthLake 控制台上没有 FHIR 导出任务信息。相反， AWS CLI 使用wit DescribeFHIRExportJob h来请求导出任务属性，例如JobStatus。有关更多信息，请参阅本页上的 AWS CLI 示例。

列出 FHIR 导出任务

用于列ListFHIRExportJobs出 HealthLake 数据存储的 FHIR 导出任务。以下菜单提供了操作步骤 AWS 管理控制台 和 AWS CLI 和的代码示例 AWS SDKs。有关更多信息，请参阅《AWS HealthLake API Reference》中的 ListFHIRExportJobs。

> **备注**
>
> HealthLake 支持用于医疗保健数据交换的 FHIR R4 规范。因此，所有健康数据都以 FHIR R4 格式导出。

列出 FHIR 导出任务

根据您的访问偏好选择菜单 AWS HealthLake。

AWS CLI 和 SDKs

CLI

AWS CLI

列出所有 FHIR 导出作业

以下 `list-fhir-export-jobs` 示例演示了如何使用该命令查看与账户关联的导出作业列表。

```
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)
```

输出：

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

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 “列出FHIRExport作业”。

Python

适用于 Python 的 SDK（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
```

- 有关 API 的详细信息，请参阅 Python 版AWS SDK 中列出FHIRExport作业 (Boto3) API 参考。

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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.
```

- 有关 API 的详细信息，请参阅在 SAP 的 AWS SDK 中列出FHIRExport作业 ABAP API 参考。

示例可用性

找不到所需的内容？ 使用本页右侧边栏上的 “提供反馈” 链接请求代码示例。

AWS 控制台

Note

HealthLake 控制台上没有 FHIR 导出任务信息。而是使用 with 列 AWS CLI `ListFHIRExportJobs`出所有 FHIR 导出任务。有关更多信息，请参阅本页上的 AWS CLI 示例。

使用的代码示 HealthLake 例 AWS SDKs

以下代码示例说明如何 HealthLake 使用 AWS 软件开发套件 (SDK)。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅HealthLake 与 AWS SDK 一起使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

代码示例

- 使用的基本示 HealthLake 例 AWS SDKs
  - HealthLake 使用的操作 AWS SDKs
    - CreateFHIRDatastore与 AWS SDK 或 CLI 配合使用
    - DeleteFHIRDatastore与 AWS SDK 或 CLI 配合使用
    - DescribeFHIRDatastore与 AWS SDK 或 CLI 配合使用
    - DescribeFHIRExportJob与 AWS SDK 或 CLI 配合使用
    - DescribeFHIRImportJob与 AWS SDK 或 CLI 配合使用
    - ListFHIRDatastores与 AWS SDK 或 CLI 配合使用
    - ListFHIRExportJobs与 AWS SDK 或 CLI 配合使用
    - ListFHIRImportJobs与 AWS SDK 或 CLI 配合使用
    - ListTagsForResource与 AWS SDK 或 CLI 配合使用
    - StartFHIRExportJob与 AWS SDK 或 CLI 配合使用
    - StartFHIRImportJob与 AWS SDK 或 CLI 配合使用
    - TagResource与 AWS SDK 或 CLI 配合使用
    - UntagResource与 AWS SDK 或 CLI 配合使用

使用的基本示 HealthLake 例 AWS SDKs

以下代码示例说明如何使用 with 的基础 AWS HealthLake 知识 AWS SDKs。

示例

- HealthLake 使用的操作 AWS SDKs

- CreateFHIRDatastore与 AWS SDK 或 CLI 配合使用
- DeleteFHIRDatastore与 AWS SDK 或 CLI 配合使用
- DescribeFHIRDatastore与 AWS SDK 或 CLI 配合使用
- DescribeFHIRExportJob与 AWS SDK 或 CLI 配合使用
- DescribeFHIRImportJob与 AWS SDK 或 CLI 配合使用
- ListFHIRDatastores与 AWS SDK 或 CLI 配合使用
- ListFHIRExportJobs与 AWS SDK 或 CLI 配合使用
- ListFHIRImportJobs与 AWS SDK 或 CLI 配合使用
- ListTagsForResource与 AWS SDK 或 CLI 配合使用
- StartFHIRExportJob与 AWS SDK 或 CLI 配合使用
- StartFHIRImportJob与 AWS SDK 或 CLI 配合使用
- TagResource与 AWS SDK 或 CLI 配合使用
- UntagResource与 AWS SDK 或 CLI 配合使用

HealthLake 使用的操作 AWS SDKs

以下代码示例演示了如何使用执行单个 HealthLake操作 AWS SDKs。每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关设置和运行代码的说明。

以下示例仅包括最常用的操作。有关完整列表，请参阅 AWS HealthLake API 参考。

示例

- CreateFHIRDatastore与 AWS SDK 或 CLI 配合使用
- DeleteFHIRDatastore与 AWS SDK 或 CLI 配合使用
- DescribeFHIRDatastore与 AWS SDK 或 CLI 配合使用
- DescribeFHIRExportJob与 AWS SDK 或 CLI 配合使用
- DescribeFHIRImportJob与 AWS SDK 或 CLI 配合使用
- ListFHIRDatastores与 AWS SDK 或 CLI 配合使用
- ListFHIRExportJobs与 AWS SDK 或 CLI 配合使用
- ListFHIRImportJobs与 AWS SDK 或 CLI 配合使用
- ListTagsForResource与 AWS SDK 或 CLI 配合使用
- StartFHIRExportJob与 AWS SDK 或 CLI 配合使用

- StartFHIRImportJob与 AWS SDK 或 CLI 配合使用
- TagResource与 AWS SDK 或 CLI 配合使用
- UntagResource与 AWS SDK 或 CLI 配合使用

`CreateFHIRDatastore`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateFHIRDatastore`。

CLI

AWS CLI

示例 1：创建支持 Sigv4 的数据存储 HealthLake

以下`create-fhir-datastore`示例演示了如何在中创建新的数据存储 AWS HealthLake。

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

输出：

```
{
    "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)"
}
```

示例 2：在支持 FHIR 的数据存储上创建 SM HealthLake ART

以下`create-fhir-datastore`示例演示了如何在中启用 FHIR 的数据存储上创建新的 SMART。 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
```

identity_provider_configuration.json 的内容：

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

输出：

```
{
    "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)"
}
```

- 有关 API 的详细信息，请参阅FHIRDatastore《AWS CLI 命令参考》中的 “创建”。

Python

适用于 Python 的 SDK（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
```

以下代码显示了启用 FH HealthLake IR 的数据存储上的 SMART 参数示例。

```
            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
            )
```

- 有关 API 的详细信息，请参阅在 Python AWS 开发工具包FHIRDatastore中创建 (Boto3) API 参考。

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅在 SAP 的 AWS SDK FHIRDatastore 中创建 ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅HealthLake 与 AWS SDK 一起使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DeleteFHIRDatastore`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeleteFHIRDatastore`。

CLI

AWS CLI

删除 FHIR 数据存储

以下`delete-fhir-datastore`示例演示如何删除数据存储及其中的所有内容 AWS HealthLake。

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

输出：

```
{
    "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)"
}
```

- 有关 API 的详细信息，请参阅FHIRDatastore《AWS CLI 命令参考》中的 “删除”。

Python

适用于 Python 的 SDK（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
```

- 有关 API 的详细信息，请参阅 Python FHIRDatastore 版AWS SDK 中删除 (Boto3) API 参考。

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅在 SAP 的 AWS SDK FHIRDatastore 中删除 ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅HealthLake 与 AWS SDK 一起使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeFHIRDatastore`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeFHIRDatastore`。

CLI

AWS CLI

描述 FHIR 数据存储

以下`describe-fhir-datastore`示例演示了如何在中查找数据存储的属性 AWS HealthLake。

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

输出：

```
{
    "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
        }
    }
}
```

- 有关 API 的详细信息，请参阅FHIRDatastore《AWS CLI 命令参考》中的 “描述”。

Python

适用于 Python 的 SDK（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
```

- 有关 API 的详细信息，请参阅适用于 Python 的AWS SDK (Boto3) API 参考FHIRDatastore中的描述。

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅适用于 SAP 的 AWS SDK ABAP API 参考FHIRDatastore中的描述。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅HealthLake 与 AWS SDK 一起使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeFHIRExportJob`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeFHIRExportJob`。

CLI

AWS CLI

描述 FHIR 导出作业

以下`describe-fhir-export-job`示例说明如何在中查找 FHIR 导出任务的 AWS HealthLake属性。

```
aws healthlake describe-fhir-export-job \
```

```
    --datastore-id (Data store ID) \
    --job-id 9b9a51943afaedd0a8c0c26c49135a31
```

输出：

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

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 “描述 FHIRExport J ob”。

Python

适用于 Python 的 SDK（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
```

- 有关 API 的详细信息，请参阅 Python 版AWS SDK 中描述FHIRExport作业 (Boto3) API 参考。

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅在 SAP 的 AWS SDK 中描述 FHIRExport J ob ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅HealthLake 与 AWS SDK 一起使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeFHIRImportJob`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeFHIRImportJob`。

CLI

AWS CLI

描述 FHIR 导入作业

以下`describe-fhir-import-job`示例说明如何使用 AWS HealthLake学习 FHIR 导入任务的属性。

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

输出：

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

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 “描述 FHIRImport J ob”。

Python

适用于 Python 的 SDK（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
```

- 有关 API 的详细信息，请参阅 Python 版AWS SDK 中描述FHIRImport作业 (Boto3) API 参考。

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅在 SAP 的 AWS SDK 中描述 FHIRImport J ob ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅HealthLake 与 AWS SDK 一起使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ListFHIRDatastores`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ListFHIRDatastores`。

CLI

AWS CLI

列出 FHIR 数据存储

以下`list-fhir-datastores`示例说明如何使用该命令以及用户如何根据中的数据存储状态筛选结果 AWS HealthLake。

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

输出：

```
{
    "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
```

```
        }
    }
    ]
}
```

- 有关 API 的详细信息，请参阅FHIRDatastores《AWS CLI 命令参考》中的 “列表”。

Python

适用于 Python 的 SDK（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
```

- 有关 API 的详细信息，请参阅适用于 Python 的AWS SDK (Boto3) API 参考FHIRDatastores 中的列表。

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅适用于 SAP 的 AWS SDK FHIRDatastores 中列出 ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅HealthLake 与 AWS SDK 一起使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ListFHIRExportJobs`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ListFHIRExportJobs`。

CLI

AWS CLI

列出所有 FHIR 导出作业

以下 `list-fhir-export-jobs` 示例演示了如何使用该命令查看与账户关联的导出作业列表。

```
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)
```

输出：

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

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 “列出FHIRExport作业”。

Python

适用于 Python 的 SDK（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
```

- 有关 API 的详细信息，请参阅 Python 版AWS SDK 中列出FHIRExport作业 (Boto3) API 参考。

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅在 SAP 的 AWS SDK 中列出FHIRExport作业 ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅HealthLake 与 AWS SDK 一起使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ListFHIRImportJobs`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ListFHIRImportJobs`。

CLI

AWS CLI

列出所有 FHIR 导入作业

以下 `list-fhir-import-jobs` 示例演示了如何使用该命令查看与账户关联的所有导入作业的列表。

```
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)
```

输出：

```
{
```

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

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 “列出FHIRImport作业”。

Python

适用于 Python 的 SDK（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
```

- 有关 API 的详细信息，请参阅 Python 版AWS SDK 中列出FHIRImport作业 (Boto3) API 参考。

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅在 SAP 的 AWS SDK 中列出FHIRImport作业 ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅HealthLake 与 AWS SDK 一起使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ListTagsForResource`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ListTagsForResource`。

CLI

AWS CLI

列出数据存储的标签

以下 `list-tags-for-resource` 示例列出与指定数据存储关联的标签。

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

输出：

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

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ListTagsForResource中的。

Python

适用于 Python 的 SDK（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
```

- 有关 API 的详细信息，请参阅适用ListTagsForResource于 Python 的AWS SDK (Boto3) API 参考。

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅适用ListTagsForResource于 S AP 的AWS SDK ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅HealthLake 与 AWS SDK 一起使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`StartFHIRExportJob`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `StartFHIRExportJob`。

CLI

AWS CLI

启动 FHIR 导出作业

以下`start-fhir-export-job`示例说明如何使用 AWS HealthLake启动 FHIR 导出任务。

```
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)
```

输出：

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

```
    "JobId": "9b9a51943afaedd0a8c0c26c49135a31"
}
```

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 “启动 FHIRExport J ob”。

Python

适用于 Python 的 SDK（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
```

- 有关 API 的详细信息，请参阅 Python 版AWS SDK 中启动FHIRExport作业 (Boto3) API 参考。

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅在 SAP 的 AWS SDK 中启动 FHIRExport J ob ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅HealthLake 与 AWS SDK 一起使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`StartFHIRImportJob`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `StartFHIRImportJob`。

CLI

AWS CLI

启动 FHIR 导入作业

以下`start-fhir-import-job`示例说明如何使用 AWS HealthLake启动 FHIR 导入任务。

```
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)"
```

输出：

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

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 “启动 FHIRImport J ob”。

Python

适用于 Python 的 SDK（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
```

- 有关 API 的详细信息，请参阅 Python 版AWS SDK 中启动FHIRImport作业 (Boto3) API 参考。

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅在 SAP 的 AWS SDK 中启动 FHIRImport J ob ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅HealthLake 与 AWS SDK 一起使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`TagResource`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `TagResource`。

CLI

AWS CLI

向数据存储中添加标签

以下 `tag-resource` 示例演示如何向数据存储中添加标签。

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

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

此命令不生成任何输出。

。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考TagResource中的。

Python

适用于 Python 的 SDK（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
```

- 有关 API 的详细信息，请参阅适用TagResource于 Python 的AWS SDK (Boto3) API 参考。

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅适用TagResource于 S AP 的AWS SDK ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅HealthLake 与 AWS SDK 一起使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`UntagResource`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `UntagResource`。

CLI

AWS CLI

从数据存储中移除标签

以下 `untag-resource` 示例演示如何从数据存储中移除标签。

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

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考UntagResource中的。

Python

适用于 Python 的 SDK（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
```

- 有关 API 的详细信息，请参阅适用UntagResource于 Python 的AWS SDK (Boto3) API 参考。

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅适用UntagResource于 S AP 的AWS SDK ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅HealthLake 与 AWS SDK 一起使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

整合 AWS HealthLake

以下 AWS 服务直接与集成 AWS HealthLake ，以实现集成的自然语言处理、SQL 查询和数据仓库。

- Amazon Comprehend Medical 是一项符合 H IPAA 资格的自然语言处理服务 (NLP)，它使用机器学习库从非结构化医学文本中提取有意义的健康数据。 HealthLake有关更多信息，请参阅亚马逊 Comprehend Medical 开发者指南。
- Amazon Athena 是一项交互式查询服务，允许您使用标准 SQ HealthLake L 直接分析亚马逊简单存储服务 (Amazon S3) 存储桶中的数据。有关更多信息，请参阅亚马逊 Athena 开发者指南。

主题

集成的自然语言处理 (NLP) 用于 HealthLake

AWS HealthLake 提供集成的自然语言处理 (NLP) 库，用于解析、识别和映射 F `DocumentReference`HIR 资源类型中存储的非结构化数据。

> **重要提示**
>
> 的集成 NLP 默认 HealthLake 处于关闭状态。要将其打开，请使用提交支持案例AWS Support Center Console。要创建您的案例，请登录您的 AWS 账户 并选择创建案例。

HealthLake 集成 NLP 的工作原理是调用 Amazon Comp `DetectEntities-V2` rehend M `InferICD10-CM` edical 和 API 操作。`InferRxNorm`这些操作会将其结果作为资源的扩展附加到`DocumentReference`资源中。当 Amazon Comprehend Medical API 操作检测到、`DIAGNOSIS`和/或的特征时`SYMPTOM`，`SIGN`它们会生成 FHIR 资源。`Linkage`新`Observation`资源`Condition`和资源由标识为`SIGN`、或特征的实体创建`SYMPTOM`，`DIAGNOSIS`并使用该`Linkage`资源将它们链接到源文档。

> **Note**
>
> 尽管支持通过 HealthLake集成 NLP 生成的 FHIR 资源GET请求，但 FHIR API `search` 功能却不支持。要详细了解如何使用与 Athena HealthLake 的集成搜索 NLP 扩展，请参阅。SQL 索引和查询

目录

HealthLake 集成的 NLP 库

HealthLake 使用 Amazon Comprehend Medical 库推断在`DocumentReference`资源类型中找到的数据。Amazon Comprehend Medical API *`DetectEntities-V2`* 操作*`InferICD10-CM`*，*`InferRxNorm`*并将疾病检测为特征。每项操作都提供了不同的见解。

> **⚠ 语言支持**
>
> Amazon Comprehend Medical API 操作仅检测英语文本中的医疗实体。

- DetectEntities-V2：检查各种医疗实体的临床文本，并返回有关它们的特定信息，例如实体类别、位置和置信度分数。
- 推断 ICD1 0-CM：以实体形式检测患者记录中的医疗状况，并将这些实体与经世界卫生组织 (WHO) 授权的疾病预防控制中心国家卫生统计中心的 ICD-10-CM 知识库中的标准化概念标识符关联起来。
- InferRxNorm: 将药物检测为患者记录中列出的实体，并将其与美国国家医学图书馆 RxNorm数据库中的标准化概念标识符关联起来。

每个 API 操作支持的特征是`SIGNSYMPTOM`、和`DIAGNOSIS`。如果检测到特征，则会将它们作为符合 FHIR 的扩展程序添加到数据存储中的不同位置。 HealthLake

添加扩展程序的位置。

- DocumentReference：Amazon Comprehend Medical API 操作的结果将作为extension一个添加到资源类型中的每个文档中。DocumentReference扩展的结果分为两组。你可以根据它们在结果中找到它们URL。
  - http://healthlake.amazonaws.com/system-generated-resources/
    - 这些是由创建或添加的资源类型 HealthLake。
  - http://healthlake.amazonaws.com/aws-cm/
    - 将 Amazon Comprehend Medical API 操作的原始输出添加到您的数据存储中。 HealthLake
- Linkage：此资源类型是由于集成 NLP 而添加或创建的。对特定项的GET请求Linkage会返回链接资源的列表。要确定是否添加Linkage了 HealthLake，请查找添加的"tag": [{"display": "SYSTEM_GENERATED"}]键值对。要了解有关 Linkage 的 FHIR 规范的更多信息，请参阅 FHIR R 4 文档Linkage中的。
- 因亚马逊 Comprehend Medical 运营而生成的 FHIR 资源类型。
  - Observation：包括 Amazon Comprehend Medical API DetectEntities-V2 操作的结果InferICD10-CM以及特征何时为或。SIGN SYMPTOM
  - Condition：包括 Amazon Comprehend Medical API DetectEntities-V2 操作InferICD10-CM的结果以及特征何时存在。DIAGNOSIS
  - MedicationStatement：包括亚马逊 Comprehend Medical API 操作的结果。InferRxNorm

使用 FHIR REST API 交互

默认情况下，在发出请求时不会返回 Amazon Comprehend Medical API 操作检测到的特征。GET要查看集成 NLP 操作的结果，必须ID为以下 FHIR 资源类型指定已知的。

- Linkage
- Observation
- Condition
- MedicationStatement

DocumentReference资源类型之外的 HealthLake 集成 NLP 操作的结果可通过GET请求获得，其中指定的ID已知包含来自 Amazon Comprehend Medical API 操作的结果。

HealthLake 集成 NLP 的搜索参数

下表列出了 HealthLake 集成 NLP 的可搜索属性。

HealthLake NLP 的搜索参数

搜索参数	查找以下项的匹配项
detecte ntities-entity-category	CM 扩展中 DetectEntities 子扩展中的实体类别 AWS
detecte ntities-entity-text	CM 扩展中 DetectEntities 子扩展中的实体文本 AWS
detecte ntities-entity-type	CM 扩展中 DetectEntities 子扩展中的实体类型 AWS
detecte ntities-entity-score	CM 扩展中 DetectEntities 子扩展中的实体分数 AWS
infer-icd10 cm-entity-text	CM 扩展中推断 ICD1 0CM 子扩展中的实体文本 AWS
infer-icd10 cm-entity-score	CM 扩展中推断 ICD1 0CM 子扩展中的实体分数 AWS
infer-icd10 cm-entity-concept-code	CM 扩展中的 Infer ICD1 0CM 子扩展中的实体概念代码 AWS
infer-icd10 cm-entity-concept-description	CM 扩展中的 Infer ICD1 0CM 子扩展中的实体概念描述 AWS
infer-icd10 cm-entity-concept-score	CM 扩展中的 Infer ICD1 0CM 子扩展中的实体概念分数 AWS
infer-rxnorm-entity-score	CM 扩展中 InferRxNorm 子扩展中的实体分数 AWS
infer-rxnorm-entity-text	CM 扩展中 InferRxNorm 子扩展中的实体文本 AWS
infer-rxnorm-entity-concept-代码	CM 扩展中 InferRxNorm 子扩展中的实体概念代码 AWS
infer-rxnorm-entity-concept-描述	CM 扩展中 InferRxNorm 子扩展中的实体概念描述 AWS
infer-rxnorm-entity-concept-分数	CM 扩展中 InferRxNorm 子扩展中的实体概念分数 AWS

HealthLake 提供特殊搜索以匹配EntityText和EntityCategory属于同一实体的条件。下表描述了支持的特殊搜索参数 HealthLake。

搜索参数

搜索参数	返回的匹配项
detecte ntities-entity-text-category	如果 DetectEntities 子扩展中至少有一个实体同时匹配 EntityText 和 EntityCategory。
detecte ntities-entity-type-score	如果子 DetectEntities 扩展中至少有一个实体同时匹配 EntityType 和 EntityScore。
detecte ntities-entity-text-score	如果子 DetectEntities 扩展中至少有一个实体同时匹配 EntityText 和 EntityScore。
detecte ntities-entity-text-type	如果 DetectEntities 子扩展中至少有一个实体同时匹配 EntityText 和 EntityType。
detecte ntities-entity-category-score	如果至少有一个实体同时与 EntityCategory 和 EntityScore 相匹配。
infer-icd10-cod cm-entity-text-concept e	如果 Infer ICD1 0CM 子扩展中至少有一个实体与 EntityText 匹配，并且该实体至少有一个与该代码匹配的 ConceptCode。
infer-icd10-scor cm-entity-text-concept e	如果 Infer ICD1 0CM 子扩展中至少有一个实体与 EntityText 匹配，并且该实体至少有一个与分数相匹配的 ConceptScore。
infer-icd10-concept-scor cm-entity-concept-description e	Infer ICD1 0CM 子扩展中的实体中是否至少有一个概念与概念描述和 ConceptScore 相匹配。
infer-rxnorm-entity-text-概念代码	如果 InferRxNorm 子扩展中至少有一个与 EntityText 匹配的实体，并且该实体至少有一个与该代码匹配的 ConceptCode。
infer-rxnorm-entity-text-概念分数	如果 InferRxNorm 子扩展中至少有一个实体与 EntityText 匹配，并且该实体至少有一个与分数相匹配的 ConceptScore。

搜索参数	返回的匹配项
infer-rxnorm-entity-concept-description-concept-score	如果 InferRxNorm子扩展中的实体中至少有一个概念与概念描述和 ConceptScore 相匹配。

HealthLake 集成 NLP 示例请求

示例 1：提取到 HealthLake 数据存储中的**`Patient`**记录

以下是基于与医疗保健专业人员会`Patient`面的临床记录示例。

> **合成数据**
>
> 以下示例中的文本是合成内容，不包含受保护的健康信息 (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)
```

提醒一下，这些信息在资源中以 base64 格式编码。DocumentReference当本文档被收录 HealthLake 并且 Amazon Comprehend Medical API 操作完成后，要查看结果，您可以从请求资源类型GET开始。DocumentReference

```
GET https://https://healthlake.region.amazonaws.com/datastore/datastoreId/
r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/DocumentReference
```

当 Amazon Comprehend Medical API 操作成功后，请在链接到以下内容的链接中查找这些键值对 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."
 }
```

以下选项卡显示了如何根据资源类型在 HealthLake 数据存储中报告摄取的医疗记录。

DocumentReference

要查看单个DocumentReference资源类型的结果，请在提供特定资源的地方GET发出请求。id

```
GET https://https://healthlake.region.amazonaws.com/datastore/datastoreId/
r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/DocumentReference/0e938f03-da7f-4178-acd8-
eea9586c46ed
```

成功后，您将获得一个 200 HTTP 响应代码和以下 JSON 响应（为清楚起见，该响应已被截断）。

这是http://healthlake.amazonaws.com/system-generated-resources/部分。你可以看到已经添加Linkage/*e366d29f-2c22-4c19-866e-09603937935a*了一个新的。您还可以查看在哪些地方向特定Observation和Condition资源类型添加 HealthLake 了基于推断的结果。

要查看这些资源类型是如何修改的，请选择相关选项卡。

```
{
  "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

要查看单个Linkage资源类型的结果，请在提供特定资源的地方GET发出请求。ID

```
GET https://https://healthlake.region.amazonaws.com/datastore/datastoreId/
r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/Linkage/e366d29f-2c22-4c19-866e-09603937935a
```

成功后，您将获得一个 200 HTTP 响应代码和以下截断的 JSON 响应。

响应包含item元素。其中，键值对"type": "source"表示用于修改的特定DocumentReference条目，Condition并Observations列在"type": "alternate"键值对下。

您还可以看到*meta*元素和相应的键值对*"tag": [{"display": "SYSTEM_GENERATED"}]*，表示这些资源是由创建的。 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

要查看单个Observation资源类型的结果，请在提供特定资源的地方GET发出请求。ID

```
GET https://https://healthlake.region.amazonaws.com/
datastore/datastoreId/r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/
Observation/e366d29f-2c22-4c19-866e-09603937935a
```

Amazon Comprehend Medical API 操作的结果已修改为以下要素：、和。code meta modifierExtension

code

类型的元素CodeableConcept。要了解更多信息，请参阅 FHIR R4 文档CodeableConcept中的。

HealthLake 附加以下三个键值对。

- "system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/": 其中 URL 指的是特定的亚马逊 Comprehend Medical API 操作。在本例中，推断 ICD1 0CM。
- "code": "A52.06": 用于识别疾病控制中心知识库中概念的 ICD-10-CM 代码在哪里A52.06。
- "display": "Other syphilitic heart involvement": 本"Other syphilitic heart involvement"体中对 ICD-10-CM 代码的详细描述在哪里。

以下截断的 JSON 响应仅包含元素。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"
 }
```

要了解模型对分配的 ICD-10-CM 代码正确性的置信度，请使用modifierExtension元素。

meta

该meta元素包含的元数据表明该code元素是否包含由 Amazon Comprehend Medical API 操作添加的详细信息。

以下截断的 JSON 响应仅包含元素。meta

```
"meta": {
  "lastUpdated": "2022-10-21T19:38:30.879Z",
  "tag": [{
   "display": "SYSTEM_GENERATED"
  }]
 }
```

modifierExtension

该modifierExtension元素包含有关code元素中已分配代码的可信度等级的更多详细信息。它还具有键值对，这些键值对提供了指向 DocumentReference 用于生成结果的原始值和相关的 Linkage 资源类型的链接。

对于添加的每个coding元素，你都会看到一个entity-score和一个被添加到 modifierE entity-Concept-Score xtension 中。对于键值对中的每个值，您会看到一个分数。因为entity-score，该分数是 Amazon Comprehend Medical 对检测准确性的可信度。因为entity-Concept-Score，该分数是亚马逊 Comprehend Medical 对该实体与 ICD-10-CM 概念准确关联的信心程度。

以下截断的 JSON 响应仅包含元素。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"
   }
  }
 ]
```

完整的 JSON 响应

```
{
  "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

要查看单个Condition资源类型的结果，请在提供特定资源的地方GET发出请求。ID

```
GET https://https://healthlake.region.amazonaws.com/datastore/datastoreId/
r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/Condition/b06d343d-
ddb8-4f36-82cb-853fcd434dfd
```

Amazon Comprehend Medical API 操作的结果已修改为以下要素：、和。code meta modifierExtension

code

类型的元素CodeableConcept。要了解更多信息，请参阅 FHIR R4 文档CodeableConcept中的。

HealthLake 附加以下三个键值对。

- "system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/": 其中 URL 指的是特定的亚马逊 Comprehend Medical API 操作。在本例中，推断 ICD1 0CM。
- "code": "I70.0": 用于识别疾病控制中心知识库中概念的 ICD-10-CM 代码在哪里A52.06。
- "display": "Atherosclerosis of aorta": 本"Other syphilitic heart involvement"体中对 ICD-10-CM 代码的详细描述在哪里。

以下截断的 JSON 响应仅包含元素。code

```
"code": {
  "coding":
  [
    {
   "system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/",
   "code": "I70.0",
   "display": "Atherosclerosis of aorta"
    }
  ],
  "text": "Atherosclerosis of aorta"
 }
```

要了解模型对分配的 ICD-10-CM 代码正确性的置信度，请使用modifierExtension元素。

meta

该meta元素包含的元数据表明该code元素是否包含由 Amazon Comprehend Medical API 操作添加的详细信息。

以下截断的 JSON 响应仅包含元素。meta

```
"meta": {
  "lastUpdated": "2022-10-21T19:38:30.877Z",
  "tag": [{
   "display": "SYSTEM_GENERATED"
  }]
 }
```

modifierExtension

该modifierExtension元素包含有关code元素中已分配代码的可信度等级的更多详细信息。它还具有键值对，这些键值对提供了指向 DocumentReference 用于生成结果的原始值和相关的 Linkage 资源类型的链接。

对于添加的每个coding元素，你都会看到一个entity-score和一个被添加到 modifierE entity-Concept-Score xtension 中。对于键值对中的每个值，您会看到一个分数。因为entity-score，该分数是 Amazon Comprehend Medical 对检测准确性的可信度。因为entity-Concept-Score，该分数是亚马逊 Comprehend Medical 对该实体与 ICD-10-CM 概念准确关联的信心程度。

以下截断的 JSON 响应仅包含元素。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"
   }
  }
 ]
```

完整的 JSON 响应

```
{
  "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"
 }
```


示例 2：**DocumentReference**包含**MedicationStatement**资源类型的 A

以下是根据患者与医疗专业人员的接触而编写的临床记录示例。

> ⚠ **合成数据**
>
> 此示例中的文本为合成内容，不包含受保护的健康信息 (PHI)。

```
Tom is not prescribed Advil
```

以下选项卡显示了如何根据资源类型在 HealthLake 数据存储中报告摄取的医疗记录。

DocumentReference

要查看单个DocumentReference资源类型的结果，请在提供特定资源的地方GET发出请求。ID

```
GET https://https://healthlake.region.amazonaws.com/datastore/datastoreId/
r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/DocumentReference/c549125d-a218-421f-
b8bf-23614c5e796c
```

成功后，您将获得一个 200 HTTP 响应代码和以下截断的 JSON 响应。

键值对表示其中的资源类型是由 Amazon Comprehen extension d Medical API 操作添加的。"url": "http://healthlake.amazonaws.com/system-generated-resources/"您可以看到新的Linkage资源类型和多个MedicationStatement资源。

```
"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

要查看单个Linkage资源类型的结果，请在提供特定资源的地方GET发出请求。ID

```
GET https://https://healthlake.region.amazonaws.com/datastore/datastoreId/
r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/Linkage/394bb244-177b-4409-8657-26b20ed56dd7
```

成功后，您将获得一个 200 HTTP 响应代码和以下 JSON 响应。

响应包含item元素。其中，键值对"type": "source"表示用于修改MedicationStatement资源DocumentReference类型的特定条目。

您还可以看到meta元素和相应的键值对"tag": [{"display": "SYSTEM_GENERATED"}]，表示这些资源是由创建的。 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

要查看单个MedicationStatement资源类型的结果，请在提供特定资源的地方GET发出请求。ID

```
GET https://https://healthlake.region.amazonaws.com/
datastore/datastoreId/r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/
MedicationStatement/9a89b0d3-6681-45ca-9926-27951edce5c7
```

MedicationStatement资源类型是查看 Amazon Comprehend Medical AP InferRxNorm I 操作结果的地方。将结果修改为以下要素：medicationCodeableConceptmeta、和modifierExtension。

medicationCodeableConcept

类型的元素CodeableConcept。要了解更多信息，请参阅 FHIR R4 文档CodeableConcept中的。

HealthLake 附加以下三个键值对。

- "system": ""http://healthlake.amazonaws.com/aws-cm/infer-rxnorm/: 其中 URL 指的是特定的亚马逊 Comprehend Medical API 操作。此处为 InferRxNorm。
- "code": "731533": 概 RxNorm 念 ID 在731533哪里，也称为 rxCUI。
- "display": "ibuprofen 200 MG Oral Capsule [Advil]": RxNorm 概念ibuprofen 200 MG Oral Capsule [Advil]的描述在哪里。

以下截断的 JSON 响应仅包含元素。MedicationStatement

```
"medicationCodeableConcept": {
  "coding": [
   {
    "system": "http://healthlake.amazonaws.com/aws-cm/infer-rxnorm/",
    "code": "731533",
    "display": "ibuprofen 200 MG Oral Capsule [Advil]"
   }
  ]
 }
```

meta

该meta元素包含的元数据表明该code元素是否包含由 Amazon Comprehend Medical API 操作添加的详细信息。

以下截断的 JSON 响应仅包含元素。meta

```
"meta": {
  "lastUpdated": "2022-10-24T20:05:02.800Z",
  "tag": [
   {
    "display": "SYSTEM_GENERATED"
   }
  ]
 }
```

modifierExtension

该modifierExtension元素包含键值对，这些键值对提供了指向DocumentReference用于生成结果的原始元素的链接以及相关的 Linkage 资源类型。

```
"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"
    }
   }
 ]
```

使用 Amazon Athena 查询 HealthLake 数据

在 HealthLake 导入作业期间，嵌套的 FHIR JSON 数据经过 ETL 处理并以 Apache Iceberg 开放表格式存储，其中每种 FHIR 资源类型在 Athena 中都表示为单独的表。这使用户可以使用 SQL 查询 FHIR 数据，但无需先将其导出。这很有价值，因为它使临床医生和科学家能够查询FHIR数据，以验证他们的决策或推进他们的研究。有关 Apache Iceberg 表在 Athena 中如何运行的更多信息，请参阅《Athena 用户指南》中的 “查询 Apache Iceberg 表”。

> **Note**
>
> HealthLake 支持 FHIR R4 在 Athena 中对你的 HealthLake 数据进行`read`交互。有关更多信息，请参阅 读取 FHIR 资源。

本节中的主题介绍如何将您的 HealthLake 数据存储连接到 Athena，如何使用 SQL 对其进行查询，以及如何将结果 AWS 与其他服务连接起来以供进一步分析。

主题

- 亚马逊 Athena 入门
- 使用 SQL 查询 HealthLake 数据
- 使用复杂筛选的 SQL 查询示例

亚马逊 Athena 入门

要 HealthLake 与 Amazon Athena 集成，您必须设置权限。为此，您需要创建 Athena 用户、群组或角色，并授予他们访问位于数据存储中的 FHIR 资源的权限。 HealthLake

- 向用户、群组或角色授予对 HealthLake 数据存储的访问权限（AWS Lake Formation 控制台）
- 设置 Athena 账号

向用户、群组或角色授予对 HealthLake 数据存储的访问权限（AWS Lake Formation 控制台）

> **角色： HealthLake 管理员**
>
> HealthLake 管理员角色是 Lake Formation 中的数据 AWS 湖管理员。他们授予对 Lake Formation 中 HealthLake 数据存储的访问权限。

对于创建的每个数据存储，La AWS ke Formation 控制台中都有两个条目可见。一个条目是资源链接。资源链接名称始终以斜体显示。每个资源链接都显示其链接的共享资源的名称和所有者。对于所有 HealthLake 数据存储，共享资源所有者是 HealthLake 服务帐号。另一个条目是 HealthLake服务帐号中的 HealthLake 数据存储。此过程中的步骤使用作为资源链接的数据存储。

要了解有关资源链接的更多信息，请参阅 Lake For mation 开发者指南中的资源链接在 Lak AWS e Formation 中的工作原理。

要使用户、群组或角色能够在 Athena 中查询数据，必须授予对资源数据库的 “描述” 权限。然后，您必须在表格上授予选择和描述权限。

步骤 1：授予对 HealthLake 数据存储资源链接数据库的 DESCRIBE 权限

1. 打开 AWS Lake Formation 控制台：https://console.aws.amazon.com/lakeformation/
2. 在主导航栏中，选择数据库。
3. 在数据库页面上，选择斜体数据存储名称旁边的单选按钮。
4. 选择 “操作” (▾)。
5. 选择授权。
6. 在授予数据权限页面的委托人下，选择 IAM 用户或角色。
7. 在 IAM 用户或角色下，使用向下箭头 (▾)，或者搜索您希望能够在 Athena 中进行查询的 IAM 用户、角色或群组。
8. 在 LF-Tags 或目录资源卡下，选择命名数据目录资源选项。
9. 在数据库下，使用向下箭头 (▾) 选择要共享访问权限 HealthLake 的数据存储数据库。
10. 在资源链接权限卡片的资源链接权限下，选择描述。

成功授予后，将出现 “授予权限” 成功横幅。要查看您刚刚授予的权限，请选择数据湖权限。在表格中找到用户、组和角色。在 “权限” 列下，您将看到列出的 “描述”。

现在，您必须使用 Grant on targ et 来对数据库中的所有表授予选择和描述权限。

步骤 2：授予对 HealthLake 数据存储资源链接中所有表的访问权限

1. 打开 AWS Lake Formation 控制台：https://console.aws.amazon.com/lakeformation/
2. 在主导航栏中，选择数据库。
3. 在数据库页面上，选择斜体数据存储名称旁边的单选按钮。
4. 选择 “操作” (▾)。
5. 选择向目标授予。
6. 在授予数据权限页面的委托人下，选择 IAM 用户或角色。
7. 在 IAM 用户或角色下，使用向下箭头 (▾) 或搜索您希望能够在 Athena 中进行查询的 IAM 用户、群组或角色。

8. 在 LF-Tags 或目录资源卡下，选择命名数据目录资源选项。
9. 在数据库下，使用向下箭头 (▾) 选择要授予访问权限 HealthLake 的数据存储数据库。
10. 在 “表” 下，选择 “所有表” 以与 HealthLake 用户共享所有表。
11. 在 “表权限” 卡的 “表格权限” 下，选择 “描述并选择”。
12. 选择授权。

选择授予后，将出现 “授予权限” 成功横幅。现在，指定的用户可以在 Athena 中的 HealthLake 数据存储上进行查询。

Athena 入门

> **HealthLake 用户**
>
> HealthLake 用户将使用 Athena 控制台 AWS CLI、 AWS SDKs 或来查询 HealthLake管理员与其共享的数据存储。 HealthLake

要使用 Athena 查询数据存储，必须执行以下三项操作。

- 通过 Lake Formation 向 IAM 用户或角色授予访问 HealthLake 数据存储的权限。要了解更多信息，请参阅向用户、群组或角色授予对 HealthLake 数据存储的访问权限（AWS Lake Formation 控制台）。
- 为您的 HealthLake 数据存储创建一个工作组。
- 指定一个 Amazon S3 存储桶来存储您的查询结果。

要开始使用 Athena，请将和 A FullAccess AWS mazonS3 托管策略添加到AmazonAthenaFullAccess您的用户、群组或角色中。使用 AWS 托管策略是开始使用新服务的好方法。请记住，AWS 托管策略可能不会为您的特定用例授予最低权限权限，因为它们可供所有 AWS 客户使用。在使用 IAM policy 设置权限时，请仅授予执行任务所需的许可。要了解有关 IAM 和应用最低权限的更多信息，请参阅 IAM 用户指南中的应用最低权限权限。

> **Important**
>
> 要查询 Athena 中的 HealthLake 数据存储，必须使用 A thena 引擎版本 3。

工作组是资源，因此您可以使用基于 IAM 的策略来控制对特定工作组的访问权限。要了解更多信息，请参阅《Athena 用户指南》中的使用工作组控制查询访问权限和费用。

要了解有关设置工作组的更多信息，请参阅https://docs.aws.amazon.com/athena/latest/ug/workgroups-procedure.html《Athena 用户指南》。

> **Note**
>
> 您的 Amazon S3 存储桶所在的区域和 Athena 控制台必须匹配。

在运行查询之前，必须指定 Amazon S3 中的查询结果存储桶位置，或者您必须使用已指定存储桶且其配置覆盖客户端设置的工作组。对于运行的每个查询，将自动保存输出文件。

有关在 Athena 控制台中指定查询结果位置的更多详情，请参阅 Amazon Athena 用户指南中的使用 Athena 控制台指定查询结果位置。

要查看如何在 Athena 中查询 HealthLake 数据存储的示例，请参阅。使用 SQL 查询 HealthLake 数据

使用 SQL 查询 HealthLake 数据

当你将 FHIR 数据导入 HealthLake 数据存储时，嵌套的 JSON FHIR 数据会同时经过 ETL 处理，并以 Apache Iceberg 开放表格式存储在 Amazon S3 中。您的 HealthLake 数据存储中的每种 FHIR 资源类型都将转换为一个表，可以在该表中使用 Amazon Athena 进行查询。可以使用基于 SQL 的查询对表进行单独查询，也可以按组进行查询。由于数据存储结构的原因，您的数据将作为多种不同的数据类型导入 Athena。要详细了解如何创建可访问这些数据类型的 SQL 查询，请参阅 Amazon Athen a 用户指南中的具有复杂类型和嵌套结构的查询数组。

> **Note**
>
> 本主题中的所有示例都使用使用 Synthea 创建的虚构数据。要了解有关创建预加载了 Synthea 数据的数据存储的更多信息，请参阅。创建 HealthLake 数据存储

对于资源类型中的每个元素，FHIR 规范都定义了一个基数。元素的基数定义了该元素可以出现的次数的下限和上限。构建 SQL 查询时，必须考虑到这一点。例如，让我们看看 “资源类型：患者” 中的一些元素。

- 元素：名称 FHIR 规范将基数设置为。`0..*`

该元素被捕获为数组。

```
[{
 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
}]
```

在 Athena 中，要查看资源类型是如何被摄取的，请在表格和视图下搜索该资源类型。要访问此数组中的元素，可以使用点表示法。以下是一个访问given和值的简单示例family。

```
SELECT
    name[1].given as FirstName,
    name[1].family as LastName
FROM Patient
```

- 元素： MaritalStatusFHIR 规范将基数设置为。0..1

此元素被捕获为 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
}
```

在 Athena 中，要查看资源类型是如何被摄取的，请在表格和视图下搜索该资源类型。要访问 JSON 中的键值对，您可以使用点表示法。因为它不是数组，所以不需要数组索引。以下是一个可以访问值的简单示例`text`。

```
SELECT
    maritalstatus.text as MaritalStatus
FROM Patient
```

要了解有关访问和搜索 JSON 的更多信息，请参阅《Athena 用户指南》中的查询 JSON。

Athena 数据操纵语言 (DML) 查询语句基于 Trino。Athena并不支持Trino的所有功能，并且存在显著差异。要了解更多信息，请参阅《A mazon Athena 用户指南》中的 DML 查询、函数和运算符。

此外，Athena 支持您在创建数据存储查询时可能会遇到的 HealthLake 多种数据类型。要了解有关 Athena 中数据类型的更多信息，请参阅亚马逊 Athena 用户指南中的亚马逊 Athena 中的数据类型。

要详细了解 SQL 查询在 Athena 中的工作原理，请参阅《亚马逊 Athena 用户指南》中的 Amazon Athena 的 SQL 参考。

每个选项卡都显示了如何使用 Athena 搜索指定资源类型和关联元素的示例。

Element: Extension

该元素`extension`用于在数据存储中创建自定义字段。

此示例向您展示如何访问在`Patient`资源类型中找到的`extension`元素的功能。

将 HealthLake 数据存储导入 Athena 时，对资源类型的元素的解析会有所不同。由于 element is 的结构变量，因此无法在架构中对其进行完全指定。为了处理这种可变性，数组中的元素作为字符串传递。

在的表描述中Patient，您可以看到extension描述为的元素array<string>，这意味着您可以使用索引值访问数组的元素。但是，要访问字符串的元素，必须使用json_extract。

以下是患者表中extension元素的单个条目。

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

尽管这是有效的 JSON，但 Athena 仍将其视为字符串。

此 SQL 查询示例演示了如何创建包含patient-mothersMaidenName和patient-birthPlace元素的表。要访问这些元素，你需要使用不同的数组索引和 json_extract.

```
SELECT
    extension[1],
    json_extract(extension[1], '$.valueString') AS MothersMaidenName,
    extension[2],
    json_extract(extension[2], '$.valueAddress.city') AS birthPlace
FROM patient
```

要详细了解涉及 JSON 的查询，请参阅亚马逊 Athena 用户指南中的从 JSON 中提取数据。

Element: birthDate (Age)

在 FHIR 中，年龄不是 “患者” 资源类型的要素。以下是根据年龄进行筛选的两个搜索示例。

因为年龄不是一个元素，所以我们使用 SQL 查询。`birthDate`要查看元素是如何被引入 FHIR 的，请在表格和视图下搜索表名。你可以看到它的类型是字符串。

示例 1：计算年龄值

在这个示例 SQL 查询中，我们使用内置的 SQL 工具`year`来提取这些组件。`current_date`然后，我们减去它们以返回患者的实际年龄，列名为`age`。

```
SELECT
 (year(current_date) - year(date(birthdate))) as age
FROM patient
```

示例 2：筛选出生之前`2019-01-01`和现在出生的患者`male`。

SQL 查询向您展示如何使用`CAST`函数将`birthDate`元素转换为类型`DATE`，以及如何根据`WHERE`子句中的两个条件进行筛选。由于默认情况下，该元素是作为字符串类型提取的，`CAST`因此我们必须将其作为类型`DATE`。然后，您可以使用`<`运算符将其与其他日期进行比较,`2019-01-01`. 通过使用`AND`，您可以向子`WHERE`句添加第二个条件。

```
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

此示例显示了在 “位置” 资源类型中搜索城市名称为 Attleboro 的地点。

```
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

资源类型条件存储与已上升到令人担忧程度的问题相关的诊断数据。 HealthLake的集成医学自然语言处理 (NLP) 会根据Condition资源类型中的详细信息生成新 DocumentReference 资源。生成新资源时， HealthLake 将标签附加SYSTEM_GENERATED到meta元素。此示例 SQL 查询演示了如何搜索条件表并返回已删除SYSTEM_GENERATED结果的结果。

要了解有关 HealthLake集成自然语言处理 (NLP) 的更多信息，请参阅集成的自然语言处理 (NLP) 用于 HealthLake。

```
SELECT *
FROM condition
WHERE meta.tag[1] is NULL
```

您也可以在指定的字符串元素中进行搜索以进一步筛选查询。该modifierextension元素包含有关使用哪个DocumentReference资源生成一组条件的详细信息。同样，您必须使用json_extract来访问作为字符串引入 Athena 的嵌套 JSON 元素。

此示例 SQL 查询演示了如何搜索根据特定内容生成的所有内容DocumentReference。Condition用于CAST将 JSON 元素设置为字符串，以便您可以LIKE用来进行比较。

```
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

资源类型 “观察” 存储有关患者、设备或其他受试者的测量结果和简单断言。 HealthLake的集成自然语言处理 (NLP) 可根据Observation资源中的详细信息生成新DocumentReference资

源。此示例 SQL 查询包括WHERE meta.tag[1] is NULL注释掉的内容，这意味着包含了SYSTEM_GENERATED结果。

```
SELECT valueCodeableConcept.coding[1].code
FROM Observation
WHERE  valueCodeableConcept.coding[1].code = '266919005'
-- WHERE meta.tag[1] is NULL
```

此列是作为导入的struct。因此，您可以使用点表示法访问其中的元素。

Resource type: MedicationStatement

MedicationStatement 是一种 FHIR 资源类型，可用于存储有关患者已服用、正在服用或将来将要服用的药物的详细信息。 HealthLake的集成医学自然语言处理 (NLP) 会根据MedicationStatement资源类型中的文档生成新 DocumentReference资源。生成新资源时， HealthLake 将标签附加SYSTEM_GENERATED到meta元素。此示例 SQL 查询演示了如何创建一个查询，该查询使用患者标识符根据单个患者进行筛选，并查找由HealthLake的集成 NLP 添加的资源。

```
SELECT *
FROM medicationstatement
WHERE meta.tag[1].display = 'SYSTEM_GENERATED' AND subject.reference =
 'Patient/0679b7b7-937d-488a-b48d-6315b8e7003b';
```

要了解有关 HealthLake集成自然语言处理 (NLP) 的更多信息，请参阅集成的自然语言处理 (NLP) 用于 HealthLake。

使用复杂筛选的 SQL 查询示例

以下示例演示如何使用具有复杂筛选功能的 Amazon Athena SQL 查询从数据存储中查找 FHIR 数据。HealthLake

Example根据人口统计数据创建筛选标准

在创建患者群组时，确定正确的患者人口统计数据非常重要。此示例查询演示了如何使用 Trino 点符号和json_extract筛选数据存储中的 HealthLake 数据。

```
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;
```

使用 Athena 控制台，您可以进一步排序和下载结果。

Example为患者及其相关病症创建过滤器

以下示例查询演示了如何查找和排序在 HealthLake 数据存储中发现的患者的所有相关病症。

```
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;
```

您可以使用 Athena 控制台对结果进行进一步排序或下载以进行进一步分析。

Example为患者及其相关观察结果创建过滤器

以下示例查询演示如何对 HealthLake 数据存储中发现的患者的所有相关观察结果进行查找和排序。

```
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为患者及其相关程序创建筛选条件

将手术与患者联系起来是医疗保健的一个重要方面。以下 SQL 示例查询演示了如何使用 FHIR `Patient` 和`Procedure`资源类型来实现此目的。以下 SQL 查询将返回在您的 HealthLake 数据存储中找到的所有患者及其相关程序。

```
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;
```

您可以使用 Athena 控制台下载结果以供进一步分析，也可以对其进行排序以更好地了解结果。

Example为患者及其相关处方创建筛选条件

查看患者正在服用的药物的最新清单很重要。使用 Athena，您可以编写 SQL 查询，该查询同时使用`Patient`数据存储中的`MedicationRequest` HealthLake 和资源类型。

以下 SQL 查询连接了导入到 Athena `MedicationRequest` 中的`Patient`和表。它还使用点符号将处方组织到各自的条目中。

```
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
```

您可以使用 Athena 控制台对结果进行排序或下载以进行进一步分析。

Example查看MedicationStatement资源类型中找到的药物

以下示例查询向您展示了如何使用 SQL 整理导入到 Athena 的嵌套 JSON。该查询使用 FHIR `meta` 元素来指示何时通过 HealthLake集成自然语言处理 (NLP) 添加了药物。它还`json_extract`用于在 JSON 字符串数组中搜索数据。有关更多信息，请参阅 自然语言处理。

```
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;
```

您可以使用 Athena 控制台下载这些结果或对其进行排序。

Example筛选特定疾病类型

该示例显示了如何找到一组年龄在18至75岁之间、被诊断出患有糖尿病的患者。

```
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%');
```

现在，您可以使用 Athena 控制台对结果进行排序或下载以进行进一步分析。

监控 AWS HealthLake

监控和日志记录是维护的安全性、可靠性、可用性和性能的重要组成部分 AWS HealthLake。 AWS 提供以下服务，供您监视 HealthLake、报告问题并在适当时自动采取措施。

- AWS CloudTrail捕获由您的账户或代表您的 AWS 账户进行的 API 调用和相关事件，并将日志文件传输到您指定的 Amazon S3 存储桶。您可以识别哪些用户和帐户拨打了电话 AWS、发出呼叫的源 IP 地址以及呼叫发生的时间。有关更多信息，请参阅 AWS CloudTrail 《用户指南》。
- Amazon 会实时 CloudWatch监控您的 AWS 资源和您运行 AWS 的应用程序。您可以收集和跟踪指标，创建自定义的控制面板，以及设置警报以在指定的指标达到您指定的阈值时通知您或采取措施。例如，您可以 CloudWatch 跟踪您的 Amazon EC2 实例的 CPU 使用率或其他指标，并在需要时自动启动新实例。有关更多信息，请参阅 Amazon CloudWatch 用户指南。
- Amazon EventBridge 是一项无服务器事件总线服务，可以轻松地将您的应用程序与来自各种来源的数据连接起来。 EventBridge 提供来自您自己的应用程序、 Software-as-a-Service (SaaS) 应用程序和 AWS 服务的实时数据流，并将这些数据路由到 Lambda 等目标。这使您能够监控服务中发生的事件，并构建事件驱动的架构。有关更多信息，请参阅 Amazon EventBridge 用户指南。

主题

使用记录 HealthLake API 调用 AWS CloudTrail

AWS HealthLake 与 AWS CloudTrail一项服务集成，该服务提供用户、角色或 AWS 服务在中执行的操作的记录 HealthLake。 CloudTrail 将所有 API 调用捕获 HealthLake 为事件。捕获的调用包括来自 HealthLake 控制台的调用和对 HealthLake API 操作的代码调用。如果您创建了跟踪，则可以允许将 CloudTrail 事件持续传输到 Amazon S3 存储桶，包括的事件 HealthLake。如果您未配置跟踪，您仍然可以在 CloudTrail 控制台的 “事件历史记录” 中查看最新的事件。使用收集的信息 CloudTrail，您可以确定向哪个请求发出 HealthLake、发出请求的 IP 地址、谁发出了请求、何时发出请求以及其他详细信息。

要了解更多信息 CloudTrail，请参阅AWS CloudTrail 用户指南。

AWS HealthLake 中的信息 CloudTrail

CloudTrail 在您创建 AWS 账户时已在您的账户上启用。当活动发生在中时 HealthLake，该活动会与其他 AWS 服务 CloudTrail 事件一起记录在事件历史记录中。您可以在 AWS 账户中查看、搜索和下载最新事件。有关更多信息，请参阅使用事件历史记录查看 CloudTrail 事件。

要持续记录您 AWS 账户中的事件，包括的事件 HealthLake，请创建跟踪。跟踪允许 CloudTrail 将日志文件传输到 Amazon S3 存储桶。默认情况下，在控制台中创建跟踪记录时，此跟踪记录应用于所有 AWS 区域。跟踪记录 AWS 分区中所有区域的事件，并将日志文件传送到您指定的 Amazon S3 存储桶。此外，您可以配置其他 AWS 服务，以进一步分析和处理 CloudTrail 日志中收集的事件数据。有关更多信息，请参阅下列内容：

- 创建跟踪概述
- CloudTrail 支持的服务和集成
- 配置 Amazon SNS 通知 CloudTrail
- 接收来自多个区域的 CloudTrail 日志文件和接收来自多个账户的 CloudTrail日志文件

对于使用 FHIR HealthLake REST API 执行的 HealthLake 操作，所有操作均由 CloudTrail API 记录并记录在 API 参考和本开发人员指南中。例如，对以下操作的调用会在 CloudTrail 日志文件中生成条目：

- `DescribeFHIRImportJob`
- `DescribeFHIRExportJob`
- `StartFHIRImportJob`
- `ListFHIRImportJobs`
- `StartFHIRExportJob`
- `ListFHIRExportJobs`
- `CreateFHIRDatastore`
- `ListFHIRDatastores`
- `DeleteFHIRDatastore`
- `DescribeFHIRDatastore`
- `UpdateResource`
- `CreateResource`
- `DeleteResource`

- ReadResource
- GetCapabilities
- SearchWithGet
- SearchWithPost

每个事件或日志条目都包含有关生成请求的人员信息。身份信息有助于您确定以下内容：

- 请求是使用根证书还是 AWS Identity and Access Management (IAM) 用户凭证发出。
- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他 AWS 服务发出。

有关更多信息，请参阅 CloudTrail userIdentity 元素。

了解 AWS HealthLake 日志文件条目

跟踪是一种配置，允许将事件作为日志文件传输到您指定的 Amazon S3 存储桶。 CloudTrail 日志文件包含一个或多个日志条目。事件代表来自任何来源的单个请求，包括有关请求的操作、操作的日期和时间、请求参数等的信息。 CloudTrail 日志文件不是公共 API 调用的有序堆栈跟踪，因此它们不会按任何特定的顺序出现。

以下示例显示了演示该CreateFHIRDatastore操作的 CloudTrail 日志条目。

```
{
    "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"
}
```


使用 Amazon 监控 HealthLake 指标 CloudWatch

您可以 HealthLake 使用 Amazon 进行监控 CloudWatch，Amazon 会收集原始数据并将其处理为可读的近乎实时的指标。这些统计数据会保存 15 个月，从而使您能够使用历史信息，并能够更好地了解您

的 Web 应用程序或服务的执行情况。还可以设置特定阈值监视警报，在达到对应阈值时发送通知或采取行动。有关更多信息，请参阅 Amazon CloudWatch 用户指南。

> **Note**
>
> 报告所有原生 HealthLake操作的指标。

下表列出了报告的 HealthLake 指标和维度 CloudWatch。每个都以用户指定数据范围的频率计数表示。

向报告了以下 HealthLake 指标 CloudWatch。

HealthLake 报告给的指标 CloudWatch

指标	说明
调用计数	呼叫的次数 APIs。可以针对账户或指定的数据存储报告此问题。 单位：计数 有效统计数据：Sum、Count 维度：操作、数据存储 ID、数据存储类型
成功请求	成功完成的 API 请求次数。 单位：计数 有效统计数据：Sum、Average 维度：操作、数据存储 ID、数据存储类型
用户错误	由于用户错误而失败的请求数。 单位：计数 有效统计数据：Sum、Average 维度：操作、数据存储 ID、数据存储类型
服务器错误	由于服务器错误而失败的请求数。

指标	说明
	单位：计数 有效统计数据：Sum、Average 维度：操作、数据存储 ID、数据存储类型
限制的请求	已被限制的请求数。此指标不包含在用户或服务器错误计数中。 单位：计数 有效统计数据：Sum、Average 维度：操作、数据存储 ID、数据存储类型
延迟	处理用户请求所用的时间（以毫秒为单位）。 单位：毫秒 有效统计数据：Minimum、Maximum、Sum、Average 维度：操作、数据存储 ID、数据存储类型

向报告了以下 HealthLake 维度 CloudWatch。

HealthLake 报告给的尺寸 CloudWatch

维度	说明
操作	请求中使用的 API 操作
DataStore身份证	请求中使用的数据存储 ID
DataStoreType	请求中使用的数据存储类型

您可以通过 AWS 管理控制台 AWS CLI、或 CloudWatch API 获取指标。 HealthLake 您可以通过其中一个 Amazon AWS 软件开发套件 (SDKs) 或 CloudWatch API 工具来使用 API。 CloudWatch HealthLake 控制台根据来自 CloudWatch API 的原始数据显示图表。

您必须具有相应的 CloudWatch 权限才能 HealthLake 进行监控 CloudWatch。有关更多信息，请参阅《亚马逊 CloudWatch 用户指南》 CloudWatch中的亚马逊身份和访问管理。

查看 HealthLake 指标

查看指标（CloudWatch 控制台）

1. 登录 AWS 管理控制台 并打开CloudWatch控制台。
2. 选择 “指标”，选择 “所有指标”，然后选择 AWS/ HealthLake。
3. 选择维度、指标名称，然后选择 添加到图表。
4. 选择日期范围的值。所选日期范围的指标计数将显示在该图表中。

使用创建警报 CloudWatch

CloudWatch 警报在指定时间段内监视单个指标，并执行一项或多项操作：向亚马逊简单通知服务 (SNS) Simple Notification Service 主题或 Auto Scaling 策略发送通知。一个或多个操作基于指标在您指定的多个时间段内相对于给定阈值的值。 CloudWatch 还可以在警报状态发生变化时向您发送 SNS 消息。

> **Note**
>
> CloudWatch 警报仅在状态发生变化并且持续到您指定的时间段内时才会调用操作。

查看指标（CloudWatch 控制台）

1. 登录 CloudWatch 控制台。
2. 依次选择 Alarms 和 Create Alarm。
3. 选择 AWS/ HealthLake，然后选择一个指标。
4. 对于 Time Range，请选择要监控的时间范围，然后选择 Next。
5. 输入名称和描述。
6. 对于 Whenever，选择 >=，然后键入一个最大值。

7. 如果 CloudWatch 要在达到警报状态时发送电子邮件，请在 “操作” 部分的 “每当此警报” 中选择 “状态为警报”。在 “发送通知至” 中，选择一个邮件列表或选择 “新建列表” 并创建新的邮件列表。
8. 预览警报预览部分中的警报。如果对警报满意，请选择 Create Alarm (创建警报)。

使用 Amazon 监控 HealthLake 事件 EventBridge

Amazon EventBridge 是一项无服务器服务，它使用事件将应用程序组件连接在一起，使您可以更轻松地构建可扩展的事件驱动应用程序。的基础 EventBridge 是创建将事件路由到目标的规则。AWS HealthLake 提供对状态变化的持久传输 EventBridge。有关更多信息，请参阅什么是亚马逊 EventBridge？ 在《亚马逊 EventBridge 用户指南》中。

> **Note**
>
> 要了解如何向亚马逊发送 HealthLake 事件 EventBridge，请参阅 for Industries 博客 AWS HealthLake中的AWS 亚马逊 EventBridge 集成。

主题

HealthLake 事件已发送至 EventBridge

下表列出了发送 EventBridge 给处理的所有 HealthLake 事件。

HealthLake 事件类型	州
数据存储事件	
创建数据存储	`CREATING`
数据存储处于活动状态	`ACTIVE`
正在删除数据存储	`DELETING`
数据存储已删除	`DELETED`

HealthLake 事件类型	州
创建数据存储失败	CREATE_FAILED
有关更多信息，请参阅《AWS HealthLake API Reference》中的 datastoreStatus 。	
导入作业事件	
导入 Job 已提交	SUBMITTED
正在导入 Job	IN_PROGRESS
导入 Job 已完成但出现错误	COMPLETED_WITH_ERRORS
导入 Job 已完成	COMPLETED
导入 Job 失败	FAILED
有关更多信息，请参阅《AWS HealthLake API Reference》中的 jobStatus 。	
导出作业事件	
导出 Job 已提交	SUBMITTED
正在导出 Job	IN_PROGRESS
导出 Job 已完成但出现错误	COMPLETED_WITH_ERRORS
导出 Job 已完成	COMPLETED
导出 Job 失败	FAILED
有关更多信息，请参阅《AWS HealthLake API Reference》中的 jobStatus 。	

HealthLake 事件结构

HealthLake 事件是具有 JSON 结构的对象，其中还包含元数据详细信息。您可以使用元数据作为输入来重新创建事件或了解更多信息。所有关联的元数据字段都列在以下菜单的代码示例下的表格中。有关更多信息，请参阅 Amazon EventBridge 用户指南中的AWS 服务事件元数据。

> **Note**
>
> 要了解如何向亚马逊发送 HealthLake 事件 EventBridge，请参阅 for Industries 博客 AWS HealthLake中的AWS 亚马逊 EventBridge 集成。

数据存储事件

Data Store Creating

州-**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

州-**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

州-**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

州-**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/"
    }
}
```

数据存储事件-元数据描述

Name	Type	说明
`version`	字符串	EventBridge 事件架构版本。
`id`	字符串	为每个事件生成的版本 4 UUID。
`detail-type`	字符串	正在发送的事件的类型。
`source`	字符串	标识生成事件的服务。

Name	Type	说明
account	字符串	数据存储所有者的 12 位数 AWS 账户 ID。
time	字符串	事件发生的时间。
region	字符串	标识数据存储的 AWS 区域。
resources	数组（字符串）	包含数据存储的 ARN 的 JSON 数组。
detail	object	包含关于事件信息的 JSON 对象。
detail.datastoreId	字符串	与状态更改事件关联的数据存储 ID。
detail.datastoreName	字符串	数据存储名称。
detail.datastoreTypeVersion	字符串	数据存储 FHIR 版本。
detail.datastoreEndpoint	字符串	数据存储端点。

导入作业事件

Import Job Submitted

州-**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

州-**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

州-**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

州-**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

州-**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/"
        }
    }
```

```
}
```

导入任务事件-元数据描述

Name	Type	说明
version	字符串	EventBridge 事件架构版本。
id	字符串	为每个事件生成的版本 4 UUID。
detail-type	字符串	正在发送的事件的类型。
source	字符串	标识生成事件的服务。
account	字符串	数据存储所有者的 12 位数 AWS 账户 ID。
time	字符串	事件发生的时间。
region	字符串	标识数据存储的 AWS 区域。
resources	数组（字符串）	包含数据存储的 ARN 的 JSON 数组。
detail	object	包含关于事件信息的 JSON 对象。
detail.jobId	字符串	与状态更改事件关联的导入任务 ID。
detail.submitTime	字符串	提交导入任务的时间。
detail.datastoreId	字符串	生成状态更改事件的数据存储。
detail.inputDataConfig	字符串	包含要导入的 FHIR 文件的 Amazon S3 存储桶的输入前缀路径。

导出作业事件

Export Job Submitted

州-**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

州-**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

州-**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

州-**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

州-**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/"
        }
    }
}
```

导出作业事件-元数据描述

Name	Type	说明
version	字符串	EventBridge 事件架构版本。
id	字符串	为每个事件生成的版本 4 UUID。
detail-type	字符串	正在发送的事件的类型。
source	字符串	标识生成事件的服务。
account	字符串	数据存储所有者的 12 位数 AWS 账户 ID。
time	字符串	事件发生的时间。
region	字符串	标识数据存储的 AWS 区域。
resources	数组（字符串）	包含数据存储的 ARN 的 JSON 数组。

Name	Type	说明
detail	object	包含关于事件信息的 JSON 对象。
detail.jobId	字符串	与状态更改事件关联的导出任务 ID。
detail.submitTime	字符串	提交导出任务的时间。
detail.datastoreId	字符串	生成状态更改事件的数据存储。
detail.outputDataConfig	字符串	包含要导出的 FHIR 文件的 Amazon S3 存储桶的输出前缀路径。

中的安全性 AWS HealthLake

云安全 AWS 是重中之重。作为 AWS 客户，您可以受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

安全是双方共同承担 AWS 的责任。责任共担模式将其描述为云的安全性和云中的安全性：

- 云的安全性 AWS 负责保护在 AWS 云中运行 AWS 服务的基础架构。 AWS 还为您提供可以安全使用的服务。 Third-party 作为AWS 合规计划合规计划合规计划合的一部分，审计师定期测试和验证我们安全的有效性。要了解适用于的合规计划 HealthLake，请参阅按合规计划划分的 AWS 范围内服务 AWS 按合规计划。
- 云端安全-您的责任由您使用的 AWS 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

本文档可帮助您了解在使用时如何应用分担责任模型 HealthLake。以下主题向您介绍如何进行配置 HealthLake 以满足您的安全和合规性目标。您还将学习如何使用其他 AWS 服务来帮助您监控和保护您的 HealthLake 资源。

主题

中的数据保护 AWS HealthLake

AWS 责任共担模式适用于保护 AWS HealthLake 中的数据。如本模型所述 AWS ，负责保护运行所有内容的全球基础架构 AWS Cloud。您负责维护对托管在此基础结构上的内容的控制。您还负责您所

使用的 AWS 服务 的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题解答 AWS条款。 有关欧洲数据保护的信息，请参阅通用数据保护条例（GDPR）中心。

出于数据保护目的，我们建议您保护 AWS 账户 凭证并使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 设置个人用户。这样，每个用户只获得履行其工作职责所需的权限。还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证（MFA）。
- 用于 SSL/TLS 与 AWS 资源通信。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用设置 API 和用户活动日志 AWS CloudTrail。有关使用 CloudTrail 跟踪捕获 AWS 活动的信息，请参阅《AWS CloudTrail 用户指南》中的使用跟 CloudTrail 踪。
- 使用 AWS 加密解决方案以及其中的所有默认安全控件 AWS 服务。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
- 如果您在 AWS 通过命令行界面或 API 进行访问时需要经过 FIPS 140-3 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准（FIPS）第 140-3 版》https://aws.amazon.com/compliance/fips/。

强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如名称字段）。这包括您使用控制台、API HealthLake 或 SDK 或以其他 AWS 服务 方式使用控制台 AWS CLI、API 或 AWS SDK 的情况。在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供 URL，强烈建议您不要在网址中包含凭证信息来验证对该服务器的请求。

在 REST 时加密 AWS HealthLake

HealthLake 默认提供加密，使用服务拥有的 AWS Key Management Service (AWS KMS) 密钥保护敏感的静态客户数据。 Customer-managed 还支持 KMS 密钥，并且是从数据存储中导入和导出文件所必需的。要了解有关 Customer-managed KMS 密钥的更多信息，请参阅 Amazon 密钥管理服务。在创建数据存储时，客户可以选择 AWS 拥有的 Customer-managed KMS 密钥或 KMS 密钥。创建数据存储后，无法更改加密配置。如果数据存储使用的是 AWS 拥有的 KMS 密钥，则该密钥将表示为，AWS_OWNED_KMS_KEY并且您将看不到用于静态加密的特定密钥。

AWS 拥有的 KMS 密钥

HealthLake 默认使用这些密钥自动加密潜在的敏感信息，例如个人身份信息或静态私人健康信息 (PHI) 数据。AWS 拥有的 KMS 密钥不会存储在您的账户中。它们是 AWS 拥有和管理的 KMS 密钥集合的一

部分，可在多个 AWS 账户中使用。AWS 服务可以使用 AWS 拥有的 KMS 密钥来保护您的数据。您无法查看、管理、使用 AWS 拥有的 KMS 密钥或审核其使用情况。但是无需执行任何工作或更改任何计划即可保护用于加密数据的密钥。

如果您使用 AWS 拥有的 KMS 密钥，则无需支付月费或使用费，也不会计入您账户的 AWS KMS 配额。有关更多信息，请参阅 AWS 拥有的密钥。

客户托管式（KMS 密钥）

HealthLake 支持使用由您创建、拥有并管理的对称客户托管 KMS 密钥在现有 AWS 拥有的加密基础上添加第二层加密。由于您可以完全控制这层加密，因此可以执行以下任务：

- 建立和维护密钥政策、IAM Policy 和授权
- 轮换密钥加密材料
- 启用和禁用密钥政策
- 添加 标签
- 创建密钥别名
- 安排密钥删除

您还可以使用 CloudTrail 来跟踪代表您 HealthLake 发送 AWS KMS 的请求。需 AWS KMS 支付额外费用。有关更多信息，请参阅客户拥有的密钥。

创建客户托管密钥

您可以使用 AWS 管理控制台或 AWS KMS API 创建对称客户托管密钥。

按照 AWS Key Management Service 开发人员指南中创建对称客户托管密钥的步骤进行操作。

密钥政策控制对客户托管密钥的访问。每个客户托管式密钥必须只有一个密钥策略，其中包含确定谁可以使用密钥以及如何使用密钥的声明。创建客户托管式密钥时，可以指定密钥策略。有关更多信息，请参阅 AWS Key Management Service 开发人员指南中的管理客户托管密钥的访问权限。

要将客户托管密钥用于您的 HealthLake 资源，必须在密钥策略中允许 kms: CreateGrant 操作。这会向客户托管密钥添加授权，该密钥控制对指定 KMS 密钥的访问权限，从而允许用户访问 kms: gran t 操作所需的权限。 HealthLake 有关更多信息，请参阅使用授权。

要将客户托管的 KMS 密钥 HealthLake 用于您的资源，必须在密钥策略中允许以下 API 操作：

- kms：向特定的客户托管 KMS 密钥CreateGrant 添加授权，该密钥允许访问授予操作。

- km DescribeKey s：提供验证密钥所需的客户托管密钥详细信息。这是所有操作所必需的。
- kms：GenerateDataKey 为所有写入操作提供对静态加密资源的访问权限。
- KMS: Decrypt 提供对加密资源的读取或搜索操作的访问权限。

以下是一个策略声明示例，它允许用户创建由 AWS HealthLake 该密钥加密的数据存储并与之交互：

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

使用客户托管 KMS 密钥时所需的 IAM 权限

使用客户托管的 KMS 密钥创建启用 AWS KMS 加密的数据存储时，创建 HealthLake 数据存储的用户或角色需要密钥策略和 IAM 策略的权限。

您可以使用 k ms: ViaService 条件密钥将 KMS 密钥的使用限制为仅限来自 HealthLake的请求。

有关密钥策略的更多信息，请参阅 AWS Key Management Service 开发人员指南中的启用 IAM 策略。

创建存储库的 IAM 用户、IAM 角色或 AWS 账户必须拥有 kms: CreateGrant、kms: 和 kms: DescribeKey 权限以及必要的 HealthLake 权限。GenerateDataKey

如何在 AWS KMS 中 HealthLake 使用授权

HealthLake 需要获得授权才能使用您的客户托管的 KMS 密钥。当您创建使用客户托管的 KMS 密钥加密的数据存储时，通过向 AWS KMS 发送CreateGrant请求来代表您 HealthLake 创建授权。AWS KMS 中的赠款用于授予对客户账户中的 KMS 密钥的 HealthLake 访问权限。

代表您 HealthLake 创建的赠款不应被撤销或撤销。如果您撤销或取消授予在您的账户中使用 AWS KMS 密钥的 HealthLake 权限，则 HealthLake 无法访问这些数据、加密推送到数据存储的新 FHIR 资源，也无法在提取时对其进行解密。当您撤销或撤销的授予时 HealthLake，更改会立即生效。要撤消访问权限，应删除数据存储而不是撤消授权。删除数据存储后， HealthLake 将代表您停用授权。

监控您的加密密钥 HealthLake

使用 CloudTrail 客户托管的 KMS 密钥时，您可以使用来跟踪代表您 HealthLake 发送的请求。 AWS KMS 日志中的日志条目在 userA CloudTrail gent 字段中显示 healthlake.amazonaws.com，以明确区分由发出的请求。 HealthLake

以下示例是 CreateGrant、 GenerateDataKey、Decrypt 和 DescribeKey 监视 AWS KMS 操作 CloudTrail 的事件，这些操作被调用 HealthLake 以访问由您的客户托管密钥加密的数据。

以下内容显示了 CreateGrant 如何使用允许 HealthLake 访问客户提供的 KMS 密钥，从而 HealthLake 允许使用该 KMS 密钥加密所有静态客户数据。

用户无需创建自己的授权。 HealthLake 通过向 AWS KMS 发送 CreateGrant 请求来代表您创建资助。中的授权 AWS KMS 用于授予对客户账户中 AWS KMS 密钥的 HealthLake访问权限。

```
        {
"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"
}
```

以下示例说明如何使用 GenerateDataKey 来确保用户在存储数据之前拥有加密数据的必要权限。

```
         {
    "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"
}
```

以下示例显示了如何 HealthLake 调用 Decrypt 操作以使用存储的加密数据密钥来访问加密数据。

```
         {
    "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"
}
```

以下示例显示了如何 HealthLake 使用该 DescribeKey 操作来验证 AWS KMS 客户拥有的 AWS KMS 密钥是否处于可用状态，以及如何帮助用户对其无法运行进行故障排除。

```
          {
    "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"
}
```

了解详情

以下资源提供了有关静态数据加密的更多信息。

有关 AWS Key Management Service 基本概念的更多信息，请参阅 AWS KMS 文档。

有关安全最佳实践的更多信息， AWS KMS 请参阅文档。

正在对以下对象进行加密 AWS HealthLake

AWS HealthLake 使用 TLS 1.2 对通过公共终端节点和后端服务传输的数据进行加密。

的身份和访问管理 AWS HealthLake

AWS Identity and Access Management (IAM) AWS 服务 可帮助管理员安全地控制对 AWS 资源的访问权限。IAM 管理员控制谁可以进行身份验证（登录）和授权（有权限）使用 HealthLake 资源。您可以使用 IAM AWS 服务 ，无需支付额外费用。

主题

- 受众
- 使用身份进行身份验证
- 使用策略管理访问
- 如何 AWS HealthLake 与 IAM 配合使用
- 基于身份的策略示例 AWS HealthLake
- AWS 的托管策略 AWS HealthLake
- 对 AWS HealthLake 身份和访问进行故障排除

受众

您的使用方式 AWS Identity and Access Management (IAM) 因您的角色而异：

- 服务用户：如果您无法访问功能，请从管理员处请求权限（请参阅对 AWS HealthLake 身份和访问进行故障排除）
- 服务管理员：确定用户访问权限并提交权限请求（请参阅如何 AWS HealthLake 与 IAM 配合使用）

- IAM 管理员：编写用于管理访问权限的策略（请参阅基于身份的策略示例 AWS HealthLake）

使用身份进行身份验证

身份验证是您 AWS 使用身份凭证登录的方式。您必须以 IAM 用户身份进行身份验证 AWS 账户根用户，或者通过担任 IAM 角色进行身份验证。

您可以使用来自身份源的证书 AWS IAM Identity Center（例如（IAM Identity Center）、单点登录身份验证或 Google/Facebook 证书，以联合身份登录。有关登录的更多信息，请参阅《AWS 登录 用户指南》中的如何登录您的 AWS 账户。

对于编程访问， AWS 提供 SDK 和 CLI 来对请求进行加密签名。有关更多信息，请参阅《IAM 用户指南》中的适用于 API 请求的AWS 签名版本 4。

AWS 账户 root 用户

创建时 AWS 账户，首先会有一个名为 AWS 账户 root 用户的登录身份，该身份可以完全访问所有资源 AWS 服务 和资源。我们强烈建议不要使用根用户进行日常任务。有关需要根用户凭证的任务，请参阅《IAM 用户指南》中的需要根用户凭证的任务。

联合身份

作为最佳实践，要求人类用户使用与身份提供商的联合身份验证才能 AWS 服务 使用临时证书进行访问。

联合身份是指来自您的企业目录、Web 身份提供商的用户 Directory Service ，或者 AWS 服务 使用来自身份源的凭据进行访问的用户。联合身份代入可提供临时凭证的角色。

要集中管理访问权限，建议使用。 AWS IAM Identity Center有关更多信息，请参阅《AWS IAM Identity Center 用户指南》中的什么是 IAM Identity Center？。

IAM 用户和群组

IAM 用户是对某个人员或应用程序具有特定权限的一个身份。建议使用临时凭证，而非具有长期凭证的 IAM 用户。有关更多信息，请参阅 IAM 用户指南中的要求人类用户使用身份提供商的联合身份验证才能 AWS 使用临时证书进行访问。

IAM 组指定一组 IAM 用户，便于更轻松地对大量用户进行权限管理。有关更多信息，请参阅《IAM 用户指南》中的 IAM 用户使用案例。

IAM 角色

IAM 角色是具有特定权限的身份，可提供临时凭证。您可以通过从用户切换到 IAM 角色（控制台）或调用 AWS CLI 或 AWS API 操作来代入角色。有关更多信息，请参阅《IAM 用户指南》中的担任角色的方法。

IAM 角色对于联合用户访问、临时 IAM 用户权限、跨账户访问、跨服务访问以及在 Amazon EC2 上运行的应用程序非常有用。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

使用策略管理访问

您可以 AWS 通过创建策略并将其附加到 AWS 身份或资源来控制中的访问权限。策略定义了与身份或资源关联时的权限。 AWS 在委托人提出请求时评估这些政策。大多数策略都以 JSON 文档的 AWS 形式存储在中。有关 JSON 策略文档的更多信息，请参阅《IAM 用户指南》中的 JSON 策略概述。

管理员使用策略，通过定义哪个主体可以在什么条件下对哪些资源执行哪些操作来指定谁有权访问什么。

默认情况下，用户和角色没有权限。IAM 管理员创建 IAM 策略并将其添加到角色中，然后用户可以担任这些角色。IAM 策略定义权限，与执行操作所用的方法无关。

基于身份的策略

基于身份的策略是您附加到身份（用户、组或角色）的 JSON 权限策略文档。这些策略控制身份可以执行什么操作、对哪些资源执行以及在什么条件下执行。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的使用客户管理型策略定义自定义 IAM 权限。

基于身份的策略可以是内联策略（直接嵌入到单个身份中）或托管策略（附加到多个身份的独立策略）。要了解如何在托管策略和内联策略之间进行选择，请参阅《IAM 用户指南》中的在托管策略与内联策略之间进行选择。

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。您必须在基于资源的策略中指定主体。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用 IAM 中的 AWS 托管策略。

其他策略类型

AWS 支持其他策略类型，这些策略类型可以设置更常见的策略类型授予的最大权限：

- 权限边界 – 设置基于身份的策略可以授予 IAM 实体的最大权限。有关更多信息，请参阅《 IAM 用户指南》中的 IAM 实体的权限边界。
- 服务控制策略 (SCPs)-在中指定组织或组织单位的最大权限 AWS Organizations。有关更多信息，请参阅《AWS Organizations 用户指南》中的服务控制策略。
- 资源控制策略 (RCPs)-设置账户中资源的最大可用权限。有关更多信息，请参阅《AWS Organizations 用户指南》中的资源控制策略 (RCPs)。
- 会话策略 – 在为角色或联合用户创建临时会话时，作为参数传递的高级策略。有关更多信息，请参阅《IAM 用户指南》中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解在涉及多种策略类型时如何 AWS 确定是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

如何 AWS HealthLake 与 IAM 配合使用

在使用 IAM 管理访问权限之前 HealthLake，请先了解有哪些 IAM 功能可供使用 HealthLake。

您可以搭配使用的 IAM 功能 AWS HealthLake

IAM 功能	HealthLake 支持
基于身份的策略	是
基于资源的策略	否
策略操作	是
策略资源	是
策略条件键	是
ACLs	否
ABAC（策略中的标签）	是

IAM 功能	HealthLake 支持
临时凭证	是
主体权限	是
服务角色	是
服务关联角色	否

要全面了解 HealthLake 以及其他 AWS 服务如何与大多数 IAM 功能配合使用，请参阅 IAM 用户指南中的与 IAM 配合使用的AWS 服务。

基于身份的策略 AWS HealthLake

支持基于身份的策略：是

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的使用客户管理型策略定义自定义 IAM 权限。

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。要了解可在 JSON 策略中使用的所有元素，请参阅《IAM 用户指南》中的 IAM JSON 策略元素引用。

基于身份的策略示例 AWS HealthLake

要查看 HealthLake 基于身份的策略的示例，请参阅。基于身份的策略示例 AWS HealthLake

内部基于资源的政策 AWS HealthLake

支持基于资源的策略：否

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 AWS 服务。

要启用跨账户访问，您可以将整个账户或其他账户中的 IAM 实体指定为基于资源的策略中的主体。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

的政策行动 AWS HealthLake

支持策略操作：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

JSON 策略的 `Action` 元素描述可用于在策略中允许或拒绝访问的操作。在策略中包含操作以授予执行关联操作的权限。

要查看 HealthLake 操作列表，请参阅《服务授权参考》 AWS HealthLake中定义的操作。

正在执行的策略操作在操作前 HealthLake 使用以下前缀：

```
healthlake
```

要在单个语句中指定多项操作，请使用逗号将它们隔开。

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

要查看 HealthLake 基于身份的策略的示例，请参阅。基于身份的策略示例 AWS HealthLake

的政策资源 AWS HealthLake

支持策略资源：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

`Resource` JSON 策略元素指定要向其应用操作的一个或多个对象。作为最佳实践，请使用其 Amazon 资源名称（ARN）指定资源。对于不支持资源级权限的操作，请使用通配符 (*) 指示语句应用于所有资源。

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

要查看 HealthLake 资源类型及其列表 ARNs，请参阅《服务授权参考》 AWS HealthLake中定义的资源。要了解可用于指定每种资源的 ARN 的操作，请参阅由定义的操作。 AWS HealthLake

要查看 HealthLake 基于身份的策略的示例，请参阅。基于身份的策略示例 AWS HealthLake

的策略条件密钥 AWS HealthLake

支持特定于服务的策略条件键：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

`Condition` 元素根据定义的条件指定语句何时执行。您可以创建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。要查看所有 AWS 全局条件键，请参阅 IAM 用户指南中的AWS 全局条件上下文密钥。

要查看 HealthLake 条件键列表，请参阅《服务授权参考》 AWS HealthLake中的条件密钥。要了解可用于使用条件键的操作和资源，请参阅由定义的操作 AWS HealthLake。

要查看 HealthLake 基于身份的策略的示例，请参阅。基于身份的策略示例 AWS HealthLake

中的访问控制列表 (ACLs) AWS HealthLake

支持 ACLs：否

访问控制列表 (ACLs) 控制哪些委托人（账户成员、用户或角色）有权访问资源。 ACLs 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

基于属性的访问控制 (ABAC) AWS HealthLake

支持 ABAC（策略中的标签）：是

基于属性的访问权限控制（ABAC）是一种授权策略，该策略基于称为标签的属性来定义权限。您可以将标签附加到 IAM 实体和 AWS 资源，然后设计 ABAC 策略以允许在委托人的标签与资源上的标签匹配时进行操作。

要基于标签控制访问，您需要使用 `aws:ResourceTag/`*`key-name`*`aws:RequestTag/`*`key-name`* 或 `aws:TagKeys` 条件键在策略的条件元素中提供标签信息。

如果某个服务对于每种资源类型都支持所有这三个条件键，则对于该服务，该值为是。如果某个服务仅对于部分资源类型支持所有这三个条件键，则该值为部分。

有关 ABAC 的更多信息，请参阅《IAM 用户指南》中的使用 ABAC 授权定义权限。要查看设置 ABAC 步骤的教程，请参阅《IAM 用户指南》中的使用基于属性的访问权限控制（ABAC）。

将临时证书与 AWS HealthLake

支持临时凭证：是

临时证书提供对 AWS 资源的短期访问权限，并且是在您使用联合身份或切换角色时自动创建的。AWS 建议您动态生成临时证书，而不是使用长期访问密钥。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的临时安全凭证和使用 IAM 的。AWS 服务

的跨服务主体权限 AWS HealthLake

支持转发访问会话（FAS）：是

转发访问会话 (FAS) 使用调用主体的权限 AWS 服务，再加上 AWS 服务 向下游服务发出请求的请求。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。

的服务角色 AWS HealthLake

支持服务角色：是

服务角色是由一项服务担任、代表您执行操作的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务委派权限的角色。

有关服务角色和完全访问权限所需的内联策略的信息 AWS HealthLake，请参阅设置 AWS HealthLake。

> **Warning**
>
> 更改服务角色的权限可能会中断 HealthLake 功能。只有在 HealthLake 提供操作指导时才编辑服务角色。

的服务相关角色 AWS HealthLake

支持服务相关角色：否

服务相关角色是一种与服务相关联的 AWS 服务服务角色。服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务关联角色的权限。

有关创建或管理服务相关角色的详细信息，请参阅能够与 IAM 搭配使用的AWS 服务。在表中查找服务相关角色列中包含 Yes 的表。选择是链接以查看该服务的服务相关角色文档。

基于身份的策略示例 AWS HealthLake

默认情况下，用户和角色没有创建或修改 HealthLake 资源的权限。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。

要了解如何使用这些示例 JSON 策略文档创建基于 IAM 身份的策略，请参阅《IAM 用户指南》中的创建 IAM 策略（控制台）。

有关由 HealthLake定义的操作和资源类型（包括每种资源类型的格式）的详细信息，请参阅《服务授权参考》 AWS HealthLake中的操作、资源和条件密钥。 ARNs

主题

- 策略最佳实践
- 使用控制 AWS HealthLake 台
- 访问中的 AWS HealthLake 数据存储 Amazon Athena
- 允许用户查看他们自己的权限

策略最佳实践

基于身份的策略决定了某人是否可以在您的账户中创建、访问或删除 HealthLake 资源。这些操作可能会使 AWS 账户产生成本。创建或编辑基于身份的策略时，请遵循以下指南和建议：

- 开始使用 AWS 托管策略并转向最低权限权限 — 要开始向用户和工作负载授予权限，请使用为许多常见用例授予权限的AWS 托管策略。它们在你的版本中可用 AWS 账户。我们建议您通过定义针对您的用例的 AWS 客户托管策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管策略或工作职能的AWS 托管策略。
- 应用最低权限：在使用 IAM 策略设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的 IAM 中的策略和权限。
- 使用 IAM 策略中的条件进一步限制访问权限：您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果服务操作是通过特定的方式使用的，则也可以使用条件来授予对服务操作的访问权限 AWS 服务，例如 CloudFormation。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。
- 使用 IAM Access Analyzer 验证您的 IAM 策略，以确保权限的安全性和功能性：IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM 策略语言（JSON）和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的使用 IAM Access Analyzer 验证策略。

- 需要多重身份验证 (MFA)-如果 AWS 账户您的场景需要 IAM 用户或根用户，请启用 MFA 以提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的使用 MFA 保护 API 访问。

有关 IAM 中的最佳实操的更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实践。

使用控制 AWS HealthLake 台

要访问 AWS HealthLake 控制台，您必须拥有一组最低权限。这些权限必须允许您列出和查看有关您的 HealthLake 资源的详细信息 AWS 账户。如果创建比必需的最低权限更为严格的基于身份的策略，对于附加了该策略的实体（用户或角色），控制台将无法按预期正常运行。

对于仅调用 AWS CLI 或 AWS API 的用户，您无需为其设置最低控制台权限。相反，只允许访问与其尝试执行的 API 操作相匹配的操作。

要获得完全访问权限 HealthLake，请将以下策略附加到 IAM 用户或角色：`AmazonHealthLakeFullAccess`和`AWSLakeFormationDataAdmin`。您还需要附加作为服务角色的 HealthLake 内联策略。服务角色是由一项服务担任、代表您执行操作的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务委派权限的角色。有关创建所需服务角色的内联策略的信息，请参阅设置 AWS HealthLake。您还必须使用 AWS Lake Formation 控制台或 CLI 将 HealthLake 管理员分配为 AWS Lake Formation 数据湖管理员。有关更多信息，请参阅 设置 AWS HealthLake。

访问中的 AWS HealthLake 数据存储 Amazon Athena

如果您想为用户和角色提供对中 HealthLake 数据存储的访问权限 Amazon Athena，请将以下 IAM 策略附加到该角色或用户：`AmazonAthenaFullAccess`和`AmazonS3FullAccess`。 `Select`并且还需要对由管理的表`Describe`具有权限 AWS Lake Formation。 AWS Lake Formation 表权限由 AWS Lake Formation 管理员在 AWS Lake Formation 控制台中或通过 CLI 授予。有关更多信息，请参阅 设置 AWS HealthLake。

允许用户查看他们自己的权限

该示例说明了如何创建策略，以支持 IAM 用户查看附加到其用户身份的内联和托管策略。此策略包括在控制台上或使用 AWS CLI 或 AWS API 以编程方式完成此操作的权限。

```
{
    "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 的托管策略 AWS HealthLake

AWS 托管策略是由创建和管理的独立策略 AWS。 AWS 托管策略旨在为许多常见用例提供权限，以便您可以开始为用户、组和角色分配权限。

请记住， AWS 托管策略可能不会为您的特定用例授予最低权限权限，因为它们可供所有 AWS 客户使用。我们建议通过定义特定于使用案例的客户管理型策略来进一步减少权限。

您无法更改 AWS 托管策略中定义的权限。如果 AWS 更新 AWS 托管策略中定义的权限，则更新会影响该策略所关联的所有委托人身份（用户、组和角色）。 AWS 最有可能在启动新的 API 或现有服务可以使用新 AWS 服务 的 API 操作时更新 AWS 托管策略。

有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管式策略。

AWS 托管策略： AmazonHealthLakeFullAccess

该AmazonHealthLakeFullAccess策略提供对的完全访问权限 HealthLake。将此策略附加到其用户或角色后，用户 HealthLake 就可以使用来访问、查询、导入和导出中的数据 HealthLake。要在中执行许多常见操作 HealthLake，必须向用户或角色添加其他策略。有关更多信息，请参阅HealthLake 操作设置 AWS HealthLake和权限。

您可以将 AmazonHealthLakeFullAccess 策略附加到 IAM 身份。

此策略授予管理权限和参与者权限，允许用户和角色查询、搜索 HealthLake、导入和导出，还可以代表具有这些权限的用户和角色执行操作。 HealthLake

权限详细信息

本政策包括以下声明。

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 托管策略： AmazonHealthLakeReadOnlyAccess

AmazonHealthLakeReadOnlyAccess策略授予对其他 AWS 服务中 HealthLake 及相关资源的只读访问权限和权限。将此策略应用于您希望授予其查询和查看 HealthLake 数据存储的权限，但不允许其创建或更改数据的用户。

您可以将 AmazonHealthLakeReadOnlyAccess 策略附加到 IAM 身份。

此策略授予允许用户和角色进行查询的*read-only*权限 HealthLake。

权限详细信息

本政策包括以下声明。

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 操作和权限

下表列出了中的典型操作 HealthLake 以及执行这些操作所需的权限。

HealthLake 操作	所需的权限
在中创建数据存储 HealthLake	AmazonHealthLakeFullAccess 、AmazonLakeFormationDataAdmin 、内联策略和 AWS Lake Formation 管理员权限由 AWS Lake Formation
删除中的数据存储 HealthLake	AmazonHealthLakeFullAccess 、AmazonLakeFormationDataAdmin 、内联策略和 AWS Lake Formation 管理员权限由 AWS Lake Formation
在中列出、搜索或查询数据存储 HealthLake	AmazonHealthLakeReadOnlyAccess
使用查询数据存储 Amazon Athena	AmazonAthenaFullAccess AmazonS3FullAccess 、 AWS Lake Formation Select和对由管理的表的Describe权限 AWS Lake Formation
从中导入数据 HealthLake	请参阅为导入任务设置权限。
从中导出数据 HealthLake	请参阅为导出任务设置权限。

HealthLake AWS 托管策略的更新

查看自该服务开始跟踪这些更改之时 HealthLake 起的 AWS 托管策略更新的详细信息。要获得有关此页面变更的自动提醒，请订阅 “ HealthLake 文档历史记录” 页面上的 RSS feed。

更改	描述	日期
AmazonHealthLakeFullAccess	AmazonHealthLakeFullAccess 需要策略才能允许完全访问 HealthLake。	2022年11月14日
AmazonHealthLakeReadOnlyAccess	AmazonHealthLakeReadOnlyAccess 对的只读访问权限需要策略 HealthLake。	2022年11月14日
HealthLake 已开始跟踪更改	HealthLake 开始跟踪其 AWS 托管策略的更改。	2022年11月14日

对 AWS HealthLake 身份和访问进行故障排除

使用以下信息来帮助您诊断和修复在使用 HealthLake 和 IAM 时可能遇到的常见问题。

主题

- 我无权在以下位置执行操作 AWS HealthLake
- 我无权执行 iam：PassRole
- 我想允许 AWS 账户之外的人访问我的 AWS HealthLake 资源

我无权在以下位置执行操作 AWS HealthLake

如果 AWS 管理控制台 告诉您您无权执行某项操作，则必须联系管理员寻求帮助。管理员是指提供用户名和密码的人员。

当 mateojackson IAM 用户尝试使用控制台查看有关虚构 *my-example-widget* 资源的详细信息，但不拥有虚构 healthlake:*GetWidget* 权限时，会发生以下示例错误。

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

在这种情况下，Mateo 请求他的管理员更新其策略，以允许他使用 `healthlake:GetWidget` 操作访问 *my-example-widget* 资源。

我无权执行 iam：PassRole

如果您收到一个错误，表明您无权执行 `iam:PassRole` 操作，则必须更新策略以允许您将角色传递给。 HealthLake

有些 AWS 服务 允许您将现有角色传递给该服务，而不是创建新的服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为 `marymajor` 的 IAM 用户尝试使用控制台在 HealthLake 中执行操作时，会发生以下示例错误。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

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

在这种情况下，必须更新 Mary 的策略以允许她执行 `iam:PassRole` 操作。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

我想允许 AWS 账户之外的人访问我的 AWS HealthLake 资源

您可以创建一个角色，以便其他账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以代入角色。对于支持基于资源的策略或访问控制列表 (ACLs) 的服务，您可以使用这些策略向人们授予访问您的资源的权限。

要了解更多信息，请参阅以下内容：

- 要了解是否 HealthLake 支持这些功能，请参阅如何 AWS HealthLake 与 IAM 配合使用。
- 要了解如何提供对您拥有的资源的访问权限 AWS 账户 ，请参阅 IAM 用户指南中的向您拥有 AWS 账户 的另一个 IAM 用户提供访问权限。
- 要了解如何向第三方提供对您的资源的访问权限 AWS 账户，请参阅 IAM 用户指南中的向第三方提供访问权限。 AWS 账户
- 要了解如何通过身份联合验证提供访问权限，请参阅《IAM 用户指南》中的为经过外部身份验证的用户（身份联合验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

的合规性验证 AWS HealthLake

Third-party AWS HealthLake 作为多个合规计划的一部分，审计师对安全性和 AWS 合规性进行评估。为 HealthLake 此，包括 HIPAA。

要了解是否属于特定合规计划的范围，请参阅AWS 服务 “按合规计划划分的范围” ”，然后选择您感兴趣的合规计划。 AWS 服务 有关一般信息，请参阅AWS 合规计划AWS。

您可以使用下载第三方审计报告 AWS Artifact。有关更多信息，请参阅中的 “下载报告” 中的 “ AWS Artifact。

您在使用 AWS 服务 时的合规责任取决于您的数据的敏感性、贵公司的合规目标以及适用的法律和法规。有关您在使用时的合规责任的更多信息 AWS 服务，请参阅AWS 安全文档。

中的基础设施安全 AWS HealthLake

作为一项托管服务， AWS HealthLake 受到《Amazon Web Services：安全流程概述》白皮书中描述的 AWS 全球网络安全程序的保护。

您可以使用 AWS 已发布的 API 调用 HealthLake 通过网络进行访问。客户端必须支持传输层安全性协议（ TLS ）1.0 或更高版本。建议使用 TLS 1.2 或更高版本。客户端还必须支持具有完全向前保密 (PFS) 的密码套件，例如短暂的 (DHE) 或椭圆曲线短暂的 Diffie-Hellman (ECDHE)。 Diffie-Hellman 大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与 IAM 主体关联的秘密访问密钥来对请求进行签名。或者，您可以使用 AWS Security Token Service（ AWS STS ）生成临时安全凭证来对请求进行签名。

使用创建 AWS HealthLake 资源 AWS CloudFormation

AWS HealthLake 与一项服务集成 AWS CloudFormation，该服务可帮助您对 AWS 资源进行建模和设置，从而减少创建和管理资源和基础架构所花费的时间。您可以创建一个描述所需的所有 AWS 资源的模板，并为您 CloudFormation 预置和配置这些资源。

使用时 CloudFormation，您可以重复使用模板来一致且重复地设置 HealthLake 资源。只需描述一次您的资源，然后在多个 AWS 账户 区域中一遍又一遍地配置相同的资源。

HealthLake 和 CloudFormation 模板

要为和相关服务配置 HealthLake 和配置资源，必须了解CloudFormation 模板。模板是 JSON 或 YAML 格式的文本文件。这些模板描述了您要在 CloudFormation 堆栈中配置的资源。如果你不熟悉

JSON 或 YAML，可以使用 D CloudFormation esigner 来帮助你开始使用 CloudFormation 模板。有关更多信息，请参阅《AWS CloudFormation 用户指南》中的什么是 CloudFormation Designer？。

> **Note**
>
> AWS HealthLake 支持使用创建数据存储 CloudFormation。有关更多信息，包括用于配置 HealthLake 数据存储的 JSON 和 YAML 模板示例，请参阅AWS CloudFormation 用户指南中的AWS HealthLake资源类型参考。

了解更多关于 CloudFormation

要了解更多信息 CloudFormation，请参阅以下资源：

- AWS CloudFormation
- AWS CloudFormation 用户指南
- CloudFormation API 参考
- AWS CloudFormation 命令行界面用户指南

AWS HealthLake 和接口 VPC 终端节点 (AWS PrivateLink)

您可以通过创建接口 VPC 终端节点在您 AWS HealthLake 的 VPC 和之间建立私有连接。接口 VPC 终端节点由AWS PrivateLink一种可用于私密访问的技术提供支持 HealthLake； APIs 无需互联网网关、NAT 设备、VPN 连接或 Direct Connect 连接。您的 VPC 中的实例不需要公有 IP 地址即可与之通信 HealthLake； APIs。您的 VPC 和 HealthLake; 之间的流量不会离开亚马逊网络。

每个接口端点均由子网中的一个或多个弹性网络接口表示。

有关更多信息，请参阅 A mazon VPC 用户指南中的接口 VPC 终端节点 (AWS PrivateLink)。

HealthLake VPC 终端节点的注意事项

在为设置接口 VPC 终端节点之前 HealthLake，请务必查看 Amazon VPC 用户指南中的接口终端节点属性和限制。

HealthLake 支持从您的 VPC 调用其所有 API 操作。

为以下对象创建接口 VPC 终端节 HealthLake点：

您可以使用 Amazon VPC 控制台或 AWS Command Line Interface (AWS CLI) 为 HealthLake; 服务创建 VPC 终端节点。有关更多信息，请参阅《Amazon VPC User Guide》中的 Creating an interface endpoint。

使用以下服务名称为 HealthLake; 创建 VPC 终端节点：

- com.amazonaws。 *region*.healthl

如果您为终端节点开启私有 DNS，则可以使用该终端节点的默认 DNS 名称向 HealthLake发出 API 请求。例如 *healthlake.us-east-1.amazonaws.com*。

有关更多信息，请参阅《Amazon VPC 用户指南》中的通过接口端点访问服务。

为创建 VPC 终端节点策略 HealthLake

您可以为 VPC 端点附加控制对 HealthLake 的访问的端点策略。该策略指定以下信息：

- 可执行操作的主体。
- 可执行的操作。
- 可对其执行操作的资源。

有关更多信息，请参阅《Amazon VPC 用户指南》中的使用 VPC 端点控制对服务的访问。

示例：用于 HealthLake 操作的 VPC 终端节点策略

以下是的终端节点策略示例 HealthLake。当连接到终端节点时，此策略将授予所有资源的所有委托人访问该 HealthLake`CreateFHIRDatastore`操作的权限。

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "healthlake:create-fhir-datastore"
         ],
         "Resource":"*"
      }
```

```
    ]
}
```

中的安全最佳实践 AWS HealthLake

AWS HealthLake 提供了许多安全功能，供您在制定和实施自己的安全策略时考虑。以下最佳实践是一般指导原则，并不代表完整安全解决方案。这些最佳实践可能不适合环境或不满足环境要求，请将其视为有用的考虑因素而不是惯例。

- 实施最低权限访问。
- 只要有可能，请使用 Customer-Managed-Keys (CMK) 加密您的数据。要了解有关 CMK 的更多信息，请参阅 Amazon 密钥管理服务。
- 在数据存储中查询 PHI 或 PII 时，请使用 “通过 POST 搜索”，而不是 “使用 GET 搜索”。
- 限制对敏感和重要的审计功能的访问。
- 通过更新或批量导入 API 创建资源时，请勿在任何可见字段或逻辑 FHIR ID (LID) 中使用 PHI 或 PII，包括数据存储和作业的名称。
- 发送创建、读取、更新、删除或搜索请求时，请勿在 HTTP 标头中使用 PHI。
- 启用 AWS CloudTrail 该选项可以审计 AWS HealthLake 使用情况并确保没有意外活动。
- 查看安全使用 Amazon S3 存储桶的最佳实践。要了解更多信息，请参阅 Amazon S3 用户指南中的安全最佳实践。

韧性在 AWS HealthLake

AWS 全球基础设施是围绕 AWS 区域和可用区构建的。 AWS 区域提供多个物理隔离和隔离的可用区，这些可用区通过低延迟、高吞吐量和高度冗余的网络相连。利用可用区，您可以设计和操作在可用区之间无中断地自动实现失效转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错能力和可扩展性。

如果需要跨更大的地理距离复制数据或应用程序，请使用 AWS 本地区域。 AWS 本地区域是旨在补充现有 AWS 区域的单一数据中心。与所有 AWS 区域一样， AWS 本地区域与其他 AWS 区域完全隔离。

有关 AWS 区域和可用区的更多信息，请参阅AWS 全球基础设施。

AWS HealthLake 参考

以下支持参考材料可用于 SMART 上的 FHIR、FHIR 和。 AWS HealthLake

> **Note**
>
> 所有原生 HealthLake 操作和数据类型均在单独的参考文献中进行了描述。有关更多信息，请参阅 AWS HealthLake API 参考。

主题

SMART on FHIR 支持 AWS HealthLake

支持 FHIR HealthLake 的数据存储上的可替代医疗应用程序和可重复使用技术 (SMART) 允许在符合 FHIR 的应用程序上访问 SMART。 HealthLake 通过使用第三方授权服务器对请求进行身份验证和授权来访问数据。因此，您不是通过管理用户凭证 AWS Identity and Access Management，而是使用符合 FHIR 标准的授权服务器上的 SMART。

> **Note**
>
> HealthLake 在 FHIR 版本 1.0 和 2.0 上支持 SMART。要了解有关这些框架的更多信息，请参阅 FHIR R4 文档中的 SMART App Laun ch。
> HealthLake 数据存储支持以下 SMART on FHIR 请求的身份验证和授权框架：
>
> - OpenID (authN)：用于对个人或客户应用程序进行身份验证，即他们声称的身份（或什么）。
> - OAuth 2.0 (authZ)：用于授权经过身份验证的请求可以读取或写入 HealthLake 数据存储中的哪些 FHIR 资源。这是由授权服务器中设置的范围定义的。

您可以使用 AWS CLI 或 AWS 软件开发工具包创建 SMART on 启用 FHIR 的数据存储。有关更多信息，请参阅 创建 HealthLake 数据存储。

创建启用 FHIR 的 SMART 数据存储后，您可以使用更新其身份提供商配置，包括授权服务器元数据、令牌验证 Lambda 函数和授权策略。`UpdateFHIRDatastore`有关更多信息，请参阅 更新 HealthLake 数据存储。

主题

- 在 FHIR 上使用 SMART 入门
- HealthLake FHIR 上的 SMART 的身份验证要求
- SMART on FHIR OAuth 2.0 瞄准镜支持 HealthLake
- 使用令牌验证 AWS Lambda
- 在启用 SMART on FHIR 的数据存储中使用细粒度授权 HealthLake
- 正在获取 SMART on FHIR 发现文档
- 在支持 Smart 的数据存储上发出 FHIR REST HealthLake API 请求

在 FHIR 上使用 SMART 入门

以下主题介绍如何开始使用 SMART on FHIR 授权。 AWS HealthLake它们包括必须在 AWS 账户中预置的资源、在 FHIR 上启用 SMART HealthLake 的数据存储的创建，以及一个 SMART on FHIR 客户端应用程序如何与授权服务器和数据存储交互的示例。 HealthLake

主题

- 在 FHIR 上为 SMART 设置资源
- FHIR 上的 SMART 的客户端应用程序工作流程

在 FHIR 上为 SMART 设置资源

以下步骤定义了如何处理 SMART on FHIR 请求 HealthLake 以及成功处理请求所需的资源。以下元素在工作流程中协同工作，以提出 SMART on FHIR 请求：

- 最终用户：通常，患者或临床医生使用第三方 SMART on FHIR 应用程序访问数据存储中的 HealthLake 数据。
- SMART on FHIR 应用程序（称为客户端应用程序）：想要访问数据存储中的HealthLake 数据的应用程序。

- 授权服务器：符合 OpenID Connect 标准的服务器，能够对用户进行身份验证并颁发访问令牌。
- HealthLake 数据存储：支持 SMART on FHIR HealthLake 的数据存储，它使用 Lambda 函数来响应提供不记名令牌的 FHIR REST 请求。

要使这些元素协同工作，必须创建以下资源。

> **Note**
>
> 我们建议在设置授权服务器、在其上定义必要的范围并创建处理令牌自省的 AWS Lambda 函数之后，在启用 FHIR HealthLake 的数据存储上创建 SMART。

1. 设置授权服务器端点

要使用 SMART on FHIR 框架，您需要设置第三方授权服务器，该服务器可以验证在数据存储上发出的 FHIR REST 请求。有关更多信息，请参阅 HealthLake FHIR 上的 SMART 的身份验证要求。

2. 在授权服务器上定义范围以控制 HealthLake 数据存储访问级别

SMART on FHIR 框架使用 OAuth 作用域来确定经过身份验证的请求可以访问哪些 FHIR 资源以及访问的范围。定义作用域是一种针对最低权限进行设计的方法。有关更多信息，请参阅 SMART on FHIR OAuth 2.0 瞄准镜支持 HealthLake。

3. 设置一个 AWS Lambda 能够执行令牌内省的函数

客户端应用程序在启用 SMART on FHIR 的数据存储上发送的 FHIR REST 请求包含 JSON Web 令牌 (JWT)。有关更多信息，请参阅解码 JWT。

4. 在启用 FHIR 时创建 SMART HealthLake 数据存储

要在 FHIR 上创建 SMART HealthLake 数据存储，您需要提供一个。`IdentityProviderConfiguration`有关更多信息，请参阅 创建 HealthLake 数据存储。

> **Note**
>
> 创建数据存储后，您可以使用更新身份提供商配置，例如，轮换令牌验证 Lambda 函数、更改授权服务器元数据或更改授权策略。`UpdateFHIRDatastore`有关更多信息，请参阅 更新 HealthLake 数据存储。

FHIR 上的 SMART 的客户端应用程序工作流程

以下部分介绍如何在 SMART on FHIR 的上下文中启动客户端应用程序并在 HealthLake 数据存储上成功发出 FHIR REST 请求。

1. 使用客户端应用程序向 Well-Known 统一资源标识符发出 **G** ET 请求

启用 SMART 的客户端应用程序必须GET请求查找 HealthLake 数据存储的授权端点。这是通过 Well-Known统一资源标识符 (URI) 请求完成的。有关更多信息，请参阅 正在获取 SMART on FHIR 发现文档。

2. 请求访问权限和范围

客户端应用程序使用授权服务器的授权端点，以便用户可以登录。此过程对用户进行身份验证。作用域用于定义客户端应用程序可以访问 HealthLake 数据存储中的哪些 FHIR 资源。有关更多信息，请参阅 SMART on FHIR OAuth 2.0 瞄准镜支持 HealthLake。

3. 访问令牌

现在，用户已通过身份验证，客户端应用程序将收到来自授权服务器的 JWT 访问令牌。此令牌是在客户端应用程序向发送 FHIR REST 请求时提供的。 HealthLake有关更多信息，请参阅 令牌验证。

4. 在启用 HealthLake FHIR 的数据存储上在 SMART 上发出 FHIR REST API 请求

现在，客户端应用程序可以使用授权服务器提供的访问令牌向 HealthLake 数据存储端点发送 FHIR REST API 请求。有关更多信息，请参阅 在支持 Smart 的数据存储上发出 FHIR REST HealthLake API 请求。

5. 验证 JWT 访问令牌

要验证在 FHIR REST 请求中发送的访问令牌，请使用 Lambda 函数。有关更多信息，请参阅 使用令牌验证 AWS Lambda。

HealthLake FHIR 上的 SMART 的身份验证要求

要访问 SMART on FHIR-enabled HealthLake 数据存储中的 FHIR 资源，客户端应用程序必须获得兼容 OAuth 2.0 的授权服务器的授权，并在 FHIR REST API 请求中出示 OAuth Bearer 令牌。要查找授权服务器的端点，请通过Well-Known统一资源标识符使用 SM HealthLake ART on FHIR 发现文档。要了解有关此过程的更多信息，请参阅正在获取 SMART on FHIR 发现文档。

在 FHIR HealthLake 数据存储上创建 SMART 时，必须在`CreateFHIRDatastore`请求的`metadata`元素中定义授权服务器的端点和令牌端点。要了解有关定义`metadata`元素的更多信息，请参阅创建 HealthLake 数据存储。

您也可以使用更新现有数据存储`metadata`中的元素（包括授权和令牌端点）`UpdateFHIRDatastore`。有关更多信息，请参阅 更新 HealthLake 数据存储。

使用授权服务器端点，客户端应用程序将使用授权服务对用户进行身份验证。授权和身份验证后，授权服务会生成 JSON Web 令牌 (JWT) 并传递给客户端应用程序。此令牌包含允许客户端应用程序使用的 FHIR 资源范围，这反过来又限制了用户能够访问的数据。或者，如果提供了启动范围，则响应中将包含这些详细信息。要了解有关支持的 SMART on FHIR 示波器的更多信息 HealthLake，请参阅SMART on FHIR OAuth 2.0 瞄准镜支持 HealthLake。

使用授权服务器授予的 JWT，客户端应用程序对启用 FHIR 的 SMART 数据存储进行 FHIR REST API 调用 HealthLake 。要验证和解码 JWT，您需要创建一个 Lambda 函数。 HealthLake 在收到 FHIR REST API 请求时代表你调用此 Lambda 函数。要查看起始 Lambda 函数的示例，请参阅。使用令牌验证 AWS Lambda

创建启用 FHIR 的 SMART HealthLake 数据存储所需的授权服务器元素

在`CreateFHIRDatastore`请求中，您需要提供授权端点和令牌端点作为`IdentityProviderConfiguration`对象中`metadata`元素的一部分。授权端点和令牌端点都是必需的。要查看`CreateFHIRDatastore`请求中如何指定这一点的示例，请参阅创建 HealthLake 数据存储。

在启用 HealthLake FHIR 的数据存储上的 SMART 上完成 FHIR REST API 请求所需的声明

您的 AWS Lambda 函数必须包含以下声明才能成为启用 SMART on FHIR HealthLake 的数据存储上的有效 FHIR REST API 请求。

- `nbf`: （Not Before）索赔 — “nbf”（不是之前）索赔指明了在此之前不得接受 JWT 进行处理的时间。“nbf” 索赔的处理要求当前索赔 date/time 必须高于或等于 “nbf” 索赔中 date/time 列出的前值。我们提供的示例 Lambda 函数`iat`从服务器响应转换为。`nbf`
- `exp`: （到期时间）索赔 — “exp”（到期时间）索赔确定了过期时间，在此时间或之后，JWT 不得被接受处理。
- `isAuthorized`: 布尔值设置为`True`。表示请求已在授权服务器上获得授权。
- `aud`: （受众）声明 — “aud”（受众）声明用于标识智威汤逊所针对的收件人。这必须是启用 SMART on FHIR HealthLake 的数据存储端点。

- `scope`：这必须是至少一个与 FHIR 资源相关的范围。此范围是在您的授权服务器上定义的。要了解有关接受的 FHIR 资源相关范围的更多信息 HealthLake，请参阅SMART 对 FHIR 的资源范围适用于 HealthLake。

SMART on FHIR OAuth 2.0 瞄准镜支持 HealthLake

HealthLake 使用 OAuth 2.0 作为授权协议。在授权服务器上使用此协议可以定义客户端应用程序有权访问的 FHIR 资源 HealthLake 的数据存储权限（创建、读取、更新、删除和搜索）。

SMART on FHIR 框架定义了一组可以向授权服务器请求的范围。例如，仅允许患者查看实验室结果或查看其联系方式的客户端应用程序应仅被授权申请`read`示波器。

> **Note**
>
> HealthLake 为 FHIR V1 和 V2 上的 SMART 提供支持，如下所述。创建数据存储时，FH `AuthorizationStrategy`IR 上的 SMART 设置为以下三个值之一：
>
> - `SMART_ON_FHIR_V1`— 在 FHIR V1 上仅支持 SMART，其中包括 `read` (read/search) 和 `write` (create/update/删除) 权限。
> - `SMART_ON_FHIR`— 在 FHIR V1 和 V2 上支持 SMART，其中包括`create`、、`read`、`update`、`delete`和权限。`search`
> - `AWS_AUTH`— 默认 AWS HealthLake 授权策略；与 FHIR 上的 SMART 无关。

独立发布范围

HealthLake 支持独立启动模式范围`launch/patient`。

在独立启动模式下，客户端应用程序请求访问患者的临床数据，因为客户端应用程序不知道用户和患者。因此，客户端应用程序的授权请求明确要求返回患者范围。成功进行身份验证后，授权服务器会发出包含请求的启动患者范围的访问令牌。所需的患者环境与访问令牌一起在授权服务器的响应中提供。

支持的启动模式范围

Scope	说明
`launch/patient`	OAuth 2.0 授权请求中的一个参数，请求在授权响应中返回患者数据。

SMART 对 FHIR 的资源范围适用于 HealthLake

HealthLake 在 FHIR 资源范围上定义了三个级别的 SMART。

- `patient`范围允许访问有关单个患者的特定数据。
- `user`作用域授予用户可以访问的特定数据的访问权限。
- `system`作用域授予对在 HealthLake 数据存储中找到的所有 FHIR 资源的访问权限。

以下各节列出了在 FHIR V1 上使用 SMART 或在 FHIR V2 上使用 SMART 构建 FHIR 资源范围的语法。

> **Note**
>
> SMART on FHIR 授权策略是在创建数据存储时设置的。有关更多信息，请参阅《AWS HealthLake API Reference》中的 `AuthorizationStrategy`。

支持 FHIR V1 瞄准镜的 SMART HealthLake

在 FHIR V1 上使用 SMART 时，构建 FHIR 资源范围的通用语法如下。 HealthLake 要查看以下示例中的整个 URL 路径，请滚动到“复制”按钮。

```
('patient' | 'user' | 'system') '/' (fhir-resource | '*') '.' ('read' | 'write' | '*')
```

SMART on FHIR v1 支持的授权范围

作用域语法	示例：作用域	结果
`patient/(fhir-resource | '*').('read' | 'write' | '*')`	`patient/AllergyIntolerance.*`	患者客户端应用程序具有对所有记录的过敏症的实例级 read/write访问权限。
`user/(fhir-resource | '*').('read' | 'write' | '*')`	`user/Observation.read`	用户客户端应用程序具有对所有记录的观察结果的实例级 read/write 访问权限。

作用域语法	示例：作用域	结果
`system/('read' | 'write' | *)`	`system/*.*`	系统客户端应用程序 read/write 可以访问所有 FHIR 资源数据。

支持 FHIR V2 瞄准镜的 SMART HealthLake

在 FHIR V2 上使用 SMART 时，构建 FHIR 资源范围的通用语法如下。 HealthLake 要查看以下示例中的整个 URL 路径，请滚动到“复制”按钮。

```
('patient' | 'user' | 'system') '/' (fhir-resource | '*') '.' ('c' | 'r' | 'u' | 'd' |
 's')
```

> **Note**
>
> 要在 FHIR V2 上使用 SMART，必须`permission-v2`将值传递到元数据`capabilities`字符串中，该字符串是`IdentityProviderConfiguration`数据类型的成员。
> HealthLake 支持精细作用域。有关更多信息，请参阅《FHIR 美国核心实施指南》中支持的粒度范围。

SMART on FHIR V2 支持的授权范围

作用域语法	示例 V1 作用域	结果
`patient/Observation.rs`	`user/Observation.read`	允许读取和搜索当前患者的`Observation` 资源。
`system/*.cruds`	`system/*.*`	系统客户端应用程序对所有 FHIR 资源数据具有完全 create/read update/delete //search 访问权限。

支持 FHIR V2.2 瞄准镜的 SMART HealthLake

V2.2 通过基于搜索参数的筛选扩展了 V2 范围。授权服务器现在可以发布通过特定数据特征而不仅仅是资源类型和 CRUDS 操作来限制访问的范围。

V2 中的所有内容都保持不变。 V2.2 纯粹是添加剂：

- 现有的 V2 作用域（不带过滤器）继续像以前一样工作。
- 使用可选的`?param=value`查询字符串对 V2 语法进行了扩展。
- 范围级别 (`patient/user/system`)、资源类型或 CRUDS 字母没有变化。

先决条件

在 FHIR 上启用 SMART 之前 V2.2，请确保满足以下条件：

- 您的数据存储是在`AuthorizationStrategy`设置为`SMART_ON_FHIR`（同时支持 V1 和 V2）的情况下创建的。使用`SMART_ON_FHIR_V1`或`AWS_AUTH`不符合条件的数据存储。
- 您的数据存储已包含`permission-v2`在`capabilities`数组中。 V2.2 是可加的，因为它扩展了 V2，不能单独使用。
- 您的 IDP Lambda 已配置为验证和传递 V2.2范围格式（?包含查询语法的范围）。

正在启用 V2.2

启用路径取决于您是在创建新的数据存储还是更新现有数据存储。

新的数据存储

创建新的数据存储时，请将以下`Metadata`字段`permission-v2.2`添加到`capabilities`数组中 `IdentityProviderConfiguration`：

```
"capabilities": [
  "launch-ehr",
  "sso-openid-connect",
  "client-public",
  "permission-v2",
  "permission-v2.2"
]
```

现有数据存储

要在现有数据存储上启用 SMART V2.2 on FHIR，请在的Metadata字段中添加permission-v2.2capabilities数组IdentityProviderConfiguration并使用UpdateFHIRDatastore提交更改。有关更多信息，请参阅 更新 HealthLake 数据存储。

要求：

- permission-v2必须保留在数组中。 V2.2 扩展了 V2，不能单独使用。
- AuthorizationStrategy必须是SMART_ON_FHIR（不是SMART_ON_FHIR_V1或AWS_AUTH）。
- 更新身份提供商配置会将其全部替换，因此请同时包括所有现有字段和功能permission-v2.2。

更改会立即生效，无需停机。要进行验证，请获取发现文档（需要 Sigv4）：

```
GET {healthlake-endpoint}/r4/.well-known/smart-configuration
```

如果响应中的capabilities数组包括permission-v2.2，则 FHIR 上的 SMART V2.2 处于活动状态。

扩展作用域语法

V2 语法使用可选的查询字符串进行了扩展：

```
V2:   (patient|user|system) / resource . cruds
V2.2: (patient|user|system) / resource . cruds [? param=value [& param=value ...]]
```

V2.2 作用域查询字符串组件

组件	说明
?param=value	Search-parameter 过滤器。只有符合此标准的资源才可访问。
&param=value	其他过滤器。多个过滤器采用 AND 运算 — 所有过滤器都必须匹配。

规则：

- 过滤器仅适用于作用域中指定的资源类型。不支持带过滤器的通配符 (*)。
- 根据数据存储的搜索配置，参数必须对资源类型有效（通过GET /r4/metadata）。

- 完整的作用域字符串（例如patient/Observation.rs?category=laboratory）是出现在 OAuth 2.0 范围参数和访问令牌声明中的文字值。scp
- URL-encode 授权请求中符合 RFC 6749 的特殊字符（例如，|→%7C）。
- 对于日期、数字和数量参数，V2.2 支持前缀比较器（例如，?date=eq2023-01-01）。V2.2 还支持搜索参数修饰符。

作用域示例

V2.2 作用域示例

Scope	已授予访问权限
patient/DiagnosticReport.rs?category=LAB	只有DiagnosticReport 资源在category哪里LAB。
patient/Observation.rs?_security=http://terminology.hl7.org/CodeSystem/v3-Confidentiality|R	仅限带有安全标签的Observation 资源Restricted 。
patient/Observation.rs?date=ge2023-01-01	仅限日期Observation 为 2023 年 1 月 1 日当天或之后的资源。
patient/Observation.rs?category=laboratory&status=final	仅限实验室和最终版本的Observation 资源。
user/Condition.rs?clinical-status=active	仅限活跃Condition 资源。

执法行为

当令牌包含 V2.2 作用域时，会按操作 HealthLake 应用筛选条件：

V2.2 每次行动强制执行

操作	行为
阅读 (r)	仅当资源与所有作用域过滤器匹配时才会成功。否则返回 403。

操作	行为
搜索 (s)	范围过滤器与查询相交。仅返回匹配的资源。
Create/Update (c/u)	资源必须满足作用域过滤器才能写入。否则返回 403。
删除 (d)	目标资源必须与范围过滤器匹配。否则返回 403。

作用域优先级

- 同一资源类型的多个 V2.2 作用域已合并（或者跨作用域）。
- 不带过滤器的更宽的 V2 作用域（例如`patient/Observation.rs`），无论同一个令牌中的 V2.2 范围是否较窄，都将授予完全访问权限。
- V2.2 未`permission-v2.2`启用的数据存储上的作用域将被静默忽略。

限制

V2.2 作用域过滤器不支持以下内容：

- 复合搜索参数（例如，`code-value-quantity`）。
- 链式搜索参数（例如，`subject:Patient.name=Smith`）。
- `_include`/`_revinclude`搜索参数。
- `$export`/`$davinci-data-export`（批量数据）- V2.2 筛选器不适用；批量导出使用 V2 范围。
- 与过滤器组合的通配符资源类型（例如，`patient/*.rs?category=LAB`无效）。使用搜索参数筛选器时，必须指定明确的资源类型（例如）。`patient/Observation.rs?category=LAB`

问题排查

症状	原因	Fix
无法识别令牌范围	V2.2 未在数据存储上启用	通过支持票证检查`/.well-known/smart-configuration` 并申请启用。
在存在的资源上使用 403	资源与作用域过滤器不匹配	根据范围参数验证资源值。

症状	原因	Fix
搜索结果为空	范围过滤器比查询更窄	结果是查询和范围筛选器的交集。
`InvalidScope` 错误	范围内的搜索参数无效	通过确认参数/`metadata` CapabilityStatement。

End-to-end 示例

场景：患者应用程序应仅显示 2023 年以后的最终实验室结果。

1. 授权服务器发布的令牌的作用域为：

```
patient/Observation.rs?category=laboratory&status=final&date=ge2023-01-01
```

2. 客户来电 HealthLake：

```
GET {endpoint}/r4/Observation?patient=Patient/123
```

3. HealthLake 强制使用范围过滤器。尽管客户端请求了所有观测值，但响应仅包含 AND `category=laboratory` `status=final` 和 `date ≥ 2023-01-01` AND 的`Observation`资源。

使用令牌验证 AWS Lambda

创建启用 FHIR 的 HealthLake SMART 数据存储时，必须在请求中`CreateFHIRDatastore`提供 AWS Lambda 该函数的 ARN。Lambda 函数的 ARN 是使用参数在`IdentityProviderConfiguration`对象中指定的。`IdpLambdaArn`

在创建 SMART on 启用 FHIR 的数据存储之前，必须创建 Lambda 函数。创建数据存储后，您可以通过更新`IdpLambdaArn`来指向不同的 Lambda 函数。`UpdateFHIRDatastore`要查看当前为数据存储配置的 Lambda ARN，请使用 API 操作。`DescribeFHIRDatastore`有关更多信息，请参阅 更新 HealthLake 数据存储。

要在启用了 SMART on FHIR 的数据存储上成功发出 FHIR REST 请求，您的 Lambda 函数必须执行以下操作：

- 在不到 1 秒的时间内向 HealthLake 数据存储端点返回响应。
- 对客户端应用程序发送的 REST API 请求的授权标头中提供的访问令牌进行解码。

- 分配一个具有足够权限来执行 FHIR REST API 请求的 IAM 服务角色。
- 完成 FHIR REST API 请求需要以下声明。要了解更多信息，请参阅所需声明。
  - nbf
  - exp
  - isAuthorized
  - aud
  - scope

使用 Lambda 时，除了您的 Lambda 函数外，您还需要创建执行角色和基于资源的策略。Lambda 函数的执行角色是一个 IAM 角色，它向函数授予访问运行时所需的 AWS 服务和资源的权限。您提供的基于资源的策略必须 HealthLake 允许代表您调用您的函数。

本主题中的各节描述了来自客户端应用程序的示例请求和解码后的响应、创建 Lamb AWS da 函数所需的步骤以及如何创建可以假设的基于资源的策略 HealthLake 。

- 第 1 部分：创建 Lambda 函数
- 第 2 部分：创建 AWS Lambda 函数使用的 HealthLake 服务角色
- 第 3 部分：更新 Lambda 函数的执行角色
- 第 4 部分：向您的 Lambda 函数添加资源策略
- 第 5 部分：为您的 Lambda 函数配置并发性

创建一个 AWS Lambda 函数

本主题中创建的 Lambda 函数在 HealthLake 收到对启用 FHIR 的 SMART 数据存储的请求时触发。来自客户端应用程序的请求包含一个 REST API 调用和一个包含访问令牌的授权标头。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
Authorization: Bearer i8hweunweunweofiwweoijewiwe
```

本主题中的示例 Lambda 函数 AWS Secrets Manager 用于掩盖与授权服务器相关的证书。我们强烈建议不要直接在 Lambda 函数中提供授权服务器登录详细信息。

Example验证包含授权持有者令牌的 FHIR REST 请求

示例 Lambda 函数向您展示了如何验证在启用 FHIR 的数据存储上发送到 SMART 的 FHIR REST 请求。要查看有关如何实现此 Lambda 函数的分步说明，请参阅。使用创建 Lambda 函数 AWS 管理控制台

如果 FHIR REST API 请求不包含有效的数据存储终端节点、访问令牌和 REST 操作，则 Lambda 函数将失败。要了解有关所需授权服务器元素的更多信息，请参阅所需声明。

```
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()
```

使用创建 Lambda 函数 AWS 管理控制台

以下过程假设您已经创建了在支持 SMART on FHIR 的数据存储上处理 FHIR REST API 请求时要 HealthLake 代入的服务角色。如果您尚未创建服务角色，则仍然可以创建 Lambda 函数。在 Lambda 函数起作用之前，您必须添加服务角色的 ARN。要了解有关创建服务角色并在 Lambda 函数中指定该角色的更多信息，请参阅 创建用于的 HealthLake 服务角色 AWS 用于解码 JWT 的 Lambda 函数

创建 Lambda 函数 (AWS 管理控制台)

1. 打开 Lamba 控制台的 Functions page（函数页面）。
2. 选择创建函数。
3. 选择从头开始编写。
4. 在基本信息下输入函数名称。在运行时下，选择基于 python 的运行时。
5. 在 Execution Role（执行角色）中，选择 Create a new role with basic Lambda permissions（创建具有基本 Lambda 权限的新角色）。

Lambda 创建了一个执行角色，该角色向该函数授予将日志上传到亚马逊的权限。CloudWatchLambda 函数在您调用函数时担任执行角色，并使用执行角色为 SDK 创建证书。AWS

6. 选择代码选项卡，然后添加示例 Lambda 函数。

   如果您尚未为 Lambda 函数创建要使用的服务角色，则需要先创建该角色，然后示例 Lambda 函数才能运行。要了解有关为 Lambda 函数创建服务角色的更多信息，请参阅。创建用于的 HealthLake 服务角色 AWS 用于解码 JWT 的 Lambda 函数

```
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()
```

修改 Lambda 函数的执行角色

创建 Lambda 函数后，您需要更新执行角色以包含调用 Secrets Manager 所需的权限。在 Secrets Manager 中，你创建的每个密钥都有一个 ARN。要应用最低权限，执行角色只能访问 Lambda 函数执行所需的资源。

您可以通过在 IAM 控制台中搜索或在 Lambda 控制台中选择配置来修改 Lambda 函数的执行角色。要了解有关管理 Lambda 函数执行角色的更多信息，请参阅。Lambda 执行角色

Example授予访问权限的 Lambda 函数执行角色 GetSecretValue

将 IAM 操作`GetSecretValue`添加到执行角色可授予示例 Lambda 函数运行所需的权限。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:secret-name-DKodTA"
        }
    ]
}
```

此时，您已经创建了一个 Lambda 函数，该函数可用于验证在 FHIR 启用了 FHIR 的数据存储上发送到您的 SMART 的 FHIR REST 请求中提供的访问令牌。

创建用于的 HealthLake 服务角色 AWS 用于解码 JWT 的 Lambda 函数

角色：IAM 管理员

可以添加或删除 IAM 策略并创建新 IAM 身份的用户。

服务角色

服务角色是由一项服务担任、代表您执行操作的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务委派权限的角色。

解码 JSON 网络令牌 (JWT) 后，Lambda 还需要返回 IAM 角色 ARN 的授权。此角色必须具有执行 REST API 请求所需的权限，否则将因权限不足而失败。

使用 IAM 设置自定义策略时，最好授予所需的最低权限。要了解更多信息，请参阅 IAM 用户指南中的应用最低权限权限。

创建要在授权 Lambda 函数中指定的 HealthLake 服务角色需要两个步骤。

- 首先，您需要创建 IAM 策略。该策略必须指定对您在授权服务器中为其提供作用域的 FHIR 资源的访问权限。

- 其次，你需要创建服务角色。创建角色时，您可以指定信任关系并附加您在第一步中创建的策略。信任关系指定 HealthLake 为服务主体。您需要在此步骤中指定 HealthLake 数据存储 ARN 和 AWS 账户 ID。

创建新的 IAM 策略

您在授权服务器中定义的范围决定了经过身份验证的用户在 HealthLake 数据存储中可以访问哪些 FHIR 资源。

您创建的 IAM 策略可以根据您定义的范围进行定制。

可以在 IAM 策略声明的`Action`元素中定义以下操作。您可以为表`Action`中的每一个定义一个`Resource types`。数据存储是唯一可以在 HealthLake IAM 权限策略声明的`Resource`元素中定义的受支持的资源类型。

单个 FHIR 资源不是您可以在 IAM 权限策略中定义为元素的资源。

操作定义为 HealthLake

操作	描述	访问级别	资源类型（必填）
CreateResource	向创建资源授予权限	写入	数据存储 ARN：arn: aws: healthlake:::/**your-region 111122223333** datastore/fhir**your-datastore-id**
DeleteResource	授予删除资源的权限	写入	数据存储 ARN：arn: aws: healthlake:::/**your-region 111122223333** datastore/fhir**your-datastore-id**
ReadResource	授予读取资源的权限	读取	数据存储 ARN：arn: aws: healthlake:::/**your-region 111122223333** datastore/fhir**your-datastore-id**
SearchWithGet	授予使用 GET 方法搜索资源的权限	读取	数据存储 ARN：arn: aws: healthlake:::/**your-region 111122223333** datastore/fhir**your-datastore-id**

操作	描述	访问级别	资源类型（必填）
SearchWithPost	授予使用 POST 方法搜索资源的权限	读取	数据存储 ARN：arn: aws: healthlake:::/**your-region 111122223333** datastore/fhir**your-datastore-id**
StartFHIRExportJobWithPost	授予使用 GET 开始 FHIR 导出任务的权限	写入	数据存储 ARN：arn: aws: healthlake:::/**your-region 111122223333** datastore/fhir**your-datastore-id**
UpdateResource	授予更新资源的权限	写入	数据存储 ARN：arn: aws: healthlake:::/**your-region 111122223333** datastore/fhir**your-datastore-id**

要开始使用，你可以使用`AmazonHealthLakeFullAccess`。该策略将允许对数据存储中找到的所有 FHIR 资源进行读取、写入、搜索和导出。要授予对数据存储的只读权限，请使用`AmazonHealthLakeReadOnlyAccess`。

要了解有关使用、或 IAM 开发工具包创建自定义策略的更多信息 AWS 管理控制台 AWS CLI，请参阅 I AM 用户指南中的创建 IAM 策略。

为 HealthLake (IAM 控制台) 创建服务角色

使用此过程创建服务角色。创建服务时，还需要指定 IAM 策略。

为 HealthLake （IAM 控制台）创建服务角色

1. 登录 AWS 管理控制台 并打开 IAM 控制台，网址为https://console.aws.amazon.com/iam/。
2. 在 IAM 控制台的导航窗格中，选择角色。
3. 然后，选择创建角色。
4. 在选择信任实体页面上，选择自定义信任策略。
5. 接下来，在“自定义信任策略”下更新示例策略，如下所示。**your-account-id**替换为您的账号，然后添加要在导入或导出任务中使用的数据存储的 ARN。

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. 然后选择下一步。
7. 在添加权限页面上，选择您希望 HealthLake 服务采用的策略。要查找您的策略，请在权限策略下进行搜索。
8. 然后，选择附加策略。
9. 然后在 “名称、查看和创建” 页面的 “角色名称” 下输入名称。
10. （可选）然后在描述下，为您的角色添加简短描述。
11. 如果可能，输入有助于识别该角色的作用的角色名称或角色名称后缀。角色名称在您的 AWS 账户内必须是唯一的。它们不按大小写区分。例如，您无法同时创建名为 **PRODROLE** 和 **prodrole** 的角色。由于多个单位可能引用该角色，角色创建完毕后无法编辑角色名称。
12. 查看角色详细信息，然后选择创建角色。

要了解如何在示例 Lambda 函数中指定角色 ARN，请参阅。[创建一个 AWS Lambda 函数]

Lambda 执行角色

Lambda 函数的执行角色是一个 IAM 角色，用于向函数授予访问 AWS 服务和资源的权限。此页面提供有关如何创建、查看和管理 Lambda 函数执行角色的信息。

默认情况下，当您使用创建新的 Lambda 函数时，Lambda 会创建具有最低权限的执行角色。 AWS 管理控制台要管理在执行角色中授予的权限，请参阅 Lambda 开发人员指南中的在 IAM 控制台中创建执行角色。

本主题中提供的示例 Lambda 函数使用 Secrets Manager 来掩盖授权服务器的证书。

与您创建的任何 IAM 角色一样，遵循最低权限最佳实践非常重要。在开发阶段，您有时可能会授予超出所需权限的权限。在生产环境中发布函数之前，最佳实践是调整策略，使其仅包含所需权限。有关更多信息，请参阅 IAM 用户指南中的应用最低权限。

HealthLake 允许触发你的 Lambda 函数

因此， HealthLake 可以代表您调用 Lambda 函数，您必须执行以下操作：

- 您需要将设置为`IdpLambdaArn`等于您要在请求中调用的 Lambda 函数的 ARN。 HealthLake `CreateFHIRDatastore`
- 您需要一个基于资源的策略， HealthLake 允许您代表您调用 Lambda 函数。

在启用 SMART on FHIR 的数据存储上 HealthLake 收到 FHIR REST API 请求时，它需要权限才能代表您调用创建数据存储时指定的 Lambda 函数。要授予 HealthLake 访问权限，您将使用基于资源的策略。要详细了解如何为 Lambda 函数创建基于资源的策略，请参阅开发者指南中的允许 AWS 服务调用 Lambda 函数。AWS Lambda

为您的 Lambda 函数配置并发性

> **Important**
>
> HealthLake 要求您的 Lambda 函数的最大运行时间必须小于一秒（ 1000 毫秒）。
> 如果您的 Lambda 函数超过了运行时间限制，则会出现异常。TimeOut

为避免出现此异常，我们建议配置预配置的并发性。通过在调用增加之前分配预置并发，您可以确保所有请求都由延迟较低的初始化实例来提供。要了解有关配置预配置并发的更多信息，请参阅 Lambda 开发人员指南中的配置预配置并发

要查看您的 Lambda 函数当前的平均运行时间，请使用 Lambda 控制台上您的 Lambda 函数的监控页面。默认情况下，Lambda 控制台提供持续时间图表，显示您的函数代码处理事件所花费的平均时间、最小时间和最大时间。要了解有关监控 Lambda 函数的更多信息，请参阅 Lambda 开发人员指南中的 Lambda 控制台中的监控函数。

如果您已经为 Lambda 函数配置了并发并想要对其进行监控，请参阅 Lambda 开发者指南中的监控并发性。

在启用 SMART on FHIR 的数据存储中使用细粒度授权 HealthLake

光靠@@ 作用域并不能为你提供必要的具体信息，说明请求者有权在数据存储中访问哪些数据。在授予对启用 FHIR 的 SMART 数据存储的访问权限时，使用细粒度授权可以提高特异性。 HealthLake 要使用细粒度授权，请在`FineGrainedAuthorizationEnabled`请求的`IdentityProviderConfiguration`参数`True`中设置等于。`CreateFHIRDatastore`

您还可以通过更新`FineGrainedAuthorizationEnabled`来启用或禁用对现有数据存储的细粒度授权。`UpdateFHIRDatastore`有关更多信息，请参阅 更新 HealthLake 数据存储。

如果您启用了细粒度授权，则您的授权服务器会返回一个`fhirUser`范围`id_token`以及访问令牌。这允许客户端应用程序检索有关用户的信息。客户端应用程序应将`fhirUser`声明视为代表当前用户的 FHIR 资源的 URI。这可以是 `Patient`、`Practitioner` 或 `RelatedPerson`。授权服务器的响应还包括一个`user/`范围，用于定义用户可以访问哪些数据。这使用为与 FHIR 资源特定作用域相关的作用域定义的语法：

```
user/(fhir-resource | '*').('read' | 'write' | '*')
```

以下是如何使用细粒度授权来进一步指定与数据访问相关的 FHIR 资源类型的示例。

- 何时`fhirUser`是`Practitioner`，细粒度的授权决定了用户可以访问的患者集合。只有患者以`fhirUser`全科医生的`fhirUser`身份提及的患者才允许进入。

  ```
  Patient.generalPractitioner : [{Reference(Practitioner)}]
  ```

- 何时`fhirUser`为`Patient`或，请求中提`RelatedPerson`及的患者与请求中提及的患者不同`fhirUser`，细粒度的授权决定了所请求患`fhirUser`者的访问权限。如果请求的`Patient`资源中指定了关系，则允许访问。

  ```
  Patient.link.other : {Reference(Patient|RelatedPerson)}
  ```

正在获取 SMART on FHIR 发现文档

SMART 定义了一个发现文档，允许客户端了解授权端点 URLs 和 HealthLake 数据存储支持的功能。此信息可帮助客户端将授权请求定向到正确的端点，并构造 HealthLake数据存储支持的授权请求。

要使客户端应用程序成功发出 FHIR REST 请求 HealthLake，它必须收集 HealthLake 数据存储所定义的授权要求。此请求不需要持有者令牌（授权）即可成功。

请求 HealthLake 数据存储的发现文档

1. 收集 HealthLake `region`和`datastoreId`价值。有关更多信息，请参阅 获取数据存储属性。
2. 使用收集到的 HealthLake`region`和值为请求构造一个 URL `datastoreId`。附加`/.well-known/smart-configuration`到 URL 的端点。要查看以下示例中的整个 URL 路径，请滚动到 “复制” 按钮。

   ```
   https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/.well-known/smart-configuration
   ```

3. 使用AWS 签名版本 4 GET 签名协议发送请求。要查看整个示例，请滚动到 “复制” 按钮。

   curl

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

   HealthLake 数据存储的 Discovery Document 以 JSON blob 的形式返回`token_endpoint`，您可以在其中找到`authorization_endpoint`和，以及数据存储的规格和已定义的功能。

   ```
   {
       "authorization_endpoint": "https://oidc.example.com/authorize",
       "token_endpoint": "https://oidc.example.com/oauth/token",
       "capabilities": [
           "launch-ehr",
           "client-public"
   ```

```
    ]
}
```

authorization_endpoint和都是启动客户端应用程序所必需的。token_endpoint

- 授权端点-授权客户端应用程序或用户所需的 URL。
- 令牌端点-客户端应用程序用来与之通信的授权服务器的端点。

在支持 Smart 的数据存储上发出 FHIR REST HealthLake API 请求

你可以在支持 FHIR 的数据存储上的 SMART 上发出 FH HealthLake IR REST API 请求。以下示例显示了来自客户端应用程序的请求，授权标头中包含 JWT，以及 Lambda 应如何解码响应。在客户端应用程序请求获得授权和身份验证后，它必须收到来自授权服务器的持有者令牌。在支持 FHIR 的 SMART 数据存储上发送 FH HealthLake IR REST API 请求时，请在授权标头中使用不记名令牌。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/[ID]
Authorization: Bearer auth-server-provided-bearer-token
```

由于在授权标头中找到了不记名令牌且未检测到 AWS IAM 身份，因此会 HealthLake 调用在创建启用 SMART on FHIR 的数据存储时指定的 Lambda 函数。 HealthLake 当您的 Lambda 函数成功解码令牌后，将向发送以下示例响应。 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
}
```

FHIR R4 支持 AWS HealthLake

AWS HealthLake 支持用于健康数据交换的 FHIR R4 规范。以下各节提供了有关如何 HealthLake 利用 FHIR R4 规范来帮助您使用 FHIR R4 管理和搜索 HealthLake 数据存储中的 FHIR 资源的支持信息。RESTful APIs

主题

- FHIR R4 能力声明 AWS HealthLake
- FHIR 个人资料验证 HealthLake
- FHIR R4 支持的资源类型适用于 HealthLake
- FHIR R4 的搜索参数为 HealthLake
- FHIR R4 适用于 $operations HealthLake

FHIR R4 能力声明 AWS HealthLake

要查找活动 HealthLake 数据存储的 FHIR 相关能力（行为），必须检索其能力声明。Capability 语句用作服务器实际功能的陈述或所需服务器实现的声明。FHIR `capabilities`交互检索有关 HealthLake 数据存储功能以及它支持 FHIR 规范的哪些部分的信息。 HealthLake 根据 FHIR R4 资源验证 FHIR 资源类型。`StructureDefinition`

获取 HealthLake 数据存储的能力声明

1. 收集 HealthLake `region`和`datastoreId`价值。有关更多信息，请参阅 获取数据存储属性。
2. 使用收集的 HealthLake `region`和值为请求构造一个 URL `datastoreId`。还要在网址中包含 FHIR `metadata` 元素。要查看以下示例中的整个 URL 路径，请滚动到 “复制” 按钮。

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

3. 发送 请求。FHIR `capabilities` 交互使用带有AWS 签名版本 4 签名协议的`GET`请求。以下`curl`示例获取由指定的 HealthLake 数据存储的能力声明`datastoreId`。要查看整个示例，请滚动到 “复制” 按钮。

   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'
```

您将收到 HealthLake 数据存储的 200 HTTP 响应代码和能力声明。有关更多信息，请参阅 FHIR R4 文档CapabilityStatement中的。

FHIR 个人资料验证 HealthLake

AWS HealthLake 支持基本的 FHIR R4 规范。基本的 FHIR R4 规范中包括 FHIR 配置文件。在 FHIR 资源类型上使用配置文件通过基本资源类型的约束 and/or 扩展来定义更具体的资源类型定义。例如，FHIR 配置文件可以识别必填字段，例如扩展名和值集。一个资源可以支持多个配置文件。所有 HealthLake 数据存储都支持使用 FHIR 配置文件。

> **Note**
>
> 向数据存储中添加数据时，不需要添加 FHIR 配置文件。 HealthLake如果在添加或更新资源时未指定 FHIR 配置文件，则仅根据基本 FHIR R4 架构对资源进行验证。
> FHIR 资源符合的 FHIR 配置文件在导入资源之前包含在资源中。 HealthLake因此，FHIR 配置文件将在导入 HealthLake 过程中通过进行验证。

FHIR 配置文件在实施指南中指定。FHIR 实施指南 (IG) 是一组说明，描述了如何将 FHIR 标准用于特定目的。 HealthLake 验证以下实施指南中定义的 FHIR 配置文件。

支持的 FHIR 配置文件 AWS HealthLake

Name	版本	实施指南	能力	美国东部（俄亥俄州）	美国东部（弗吉尼亚州北部	美国西部（俄勒冈州）	亚太地区（孟买	亚太地区（悉尼	加拿大（中部	欧洲地区（爱尔兰	欧洲地区（伦敦）
美国核心	3.1	http://hl7.org/fhir/us/core/STU3.1.1/	默认	X	X	X	X	X	X	X	X
美国核心	4.0	https://hl7.org/fhir/us/core/STU4/index.html	支持	X	X	X	X	X	X	X	X
美国核心	5.0	https://hl7.org/fhir/us/core/STU5.0.1/index.html	支持	X	X	X	X	X	X	X	X
美国核心	6.1	https://hl7.org/fhir/us/core/STU6.1/index.html	支持	X	X	X	X	X	X	X	X
美国核心	7.0	https://hl7.org/fhir/us/core/STU7/	支持	X	X	X	X	X	X	X	X
英国核心	2.0	https://simplifier.net/guide/uk-core-implementation-guide-stu2/Home/ProfilesandExtensions/ProfilesIndex?version=2.0.1	支持	X	X	X	X			X	X

Name	版本	实施指南	能力	美国东部（俄亥俄州）	美国东部（弗吉尼亚州北部	美国西部（俄勒冈州）	亚太地区（孟买	亚太地区（悉尼	加拿大（中部	欧洲地区（爱尔兰	欧洲地区（伦敦）
CARIN 蓝色按钮	1.1	http://hl7.org/fhir/us/carin-bb/STU1.1/	默认	X	X	X	X	X	X	X	X
CARIN 蓝色按钮	1.0.0、	https://hl7.org/fhir/us/carin-bb/history.html	支持	X	X	X	X	X	X	X	X
Da Vinci Payer Data Exchange	1.0	https://hl7.org/fhir/us/davinci-pdex/	默认	X	X	X	X	X	X	X	X
Da Vinci Payer Data Exchange	2.0.0	https://hl7.org/fhir/us/davinci-pdex/history.html	支持	X	X	X	X	X	X	X	X
达芬奇健康记录交易所 (HReX)	0.2	https://hl7.org/fhir/us/davinci-hrex/2020Sep/	默认	X	X	X	X	X	X	X	X
DaVinci PDEX Plan Net	1.1	https://hl7.org/fhir/us/davinci-pdex-plan-net/STU1.1/	默认	X	X	X	X	X	X	X	X
DaVinci PDEX Plan Net	1.0	https://hl7.org/fhir/us/davinci-pdex-plan-net/STU1/	支持	X	X	X	X	X	X	X	X

Name	版本	实施指南	能力	美国东部（俄亥俄州）	美国东部（弗吉尼亚州北部	美国西部（俄勒冈州）	亚太地区（孟买	亚太地区（悉尼	加拿大（中部	欧洲地区（爱尔兰	欧洲地区（伦敦）
DaVinci Payer Data Exchange (PDEx)《美国药品处方集》	1.1	https://hl7.org/fhir/us/davinci-drug-formulary/STU1.1/	默认	X	X	X	X	X	X	X	X
DaVinci Payer Data Exchange (PDEx)《美国药品处方集》	1.0.1、	https://hl7.org/fhir/us/davinci-drug-formulary/history.html	支持	X	X	X	X	X	X	X	X
Da Vinci Clinical Data Exchange (CDex)	2.1	https://build.fhir.org/ig/HL7/davinci-ecdx/index.html	默认	X	X	X	X	X	X	X	X
Da Vinci 事先授权支持 (PAS) FHIR IG	2.1	https://hl7.org/fhir/us/davinci-pas/	默认	X	X	X	X	X	X	X	X

Name	版本	实施指南	能力	美国东部（俄亥俄州）	美国东部（弗吉尼亚州北部	美国西部（俄勒冈州）	亚太地区（孟买	亚太地区（悉尼	加拿大（中部	欧洲地区（爱尔兰	欧洲地区（伦敦）
NCQA HEDIS® 实施指南	0.3	https://www.ncqa.org/resources/hedis-ig-resource-page/	默认	X	X	X	X			X	X
国际患者摘要 (IPS)	2.0 选票	https://hl7.org/fhir/uv/ips/2024Sep/	默认	X	X	X	X	X	X	X	X
质量衡量标准	5.0	https://registry.fhir.org/package/hl7.fhir.us.cqfmeasures%7C5.0.0	默认	X	X	X	X			X	X
基因组学报告	3.0	https://build.fhir.org/ig/HL7/genomics-reporting/index.html	默认	X	X	X	X			X	X
国家卫生局的阿育什曼·巴拉特数字使命（ABDM）	2.0	https://www.nrces.in/ndhm/fhir/r4/index.html	默认	X	X	X	X			X	X
CA Core+	1.1	https://simplifier.net/ca-core	支持						X		

Name	版本	实施指南	能力	美国东部（俄亥俄州）	美国东部（吉尼亚州北部	美国西部（勒冈州）	亚太地区（买	亚太地区（尼	加拿大（部	欧洲地区（尔兰	欧洲地区（伦敦）
CA:eReC Pan-Canadian e Referral-eConsult	1.1	https://simplifier.net/CA-eReC/~introduction	支持						X		
患者摘要加拿大版-(PS-CA)	2.1	https://simplifier.net/PS-CA-R1/~introduction	支持						X		
安大略省数字健康药物库	4.0	https://simplifier.net/ca-on-dhdr-r4/~introduction	支持						X		
AU 核心	1.0	https://hl7.org.au/fhir/core/	支持					X			
Magentus 实践管理	1.2	https://fhir-versions.dev.geniesolutions.io/1.2.16/downloads.html	支持					X			

验证资源中指定的 FHIR 配置文件

要验证 FHIR 配置文件，请使用实施指南中指定的配置文件 URL 将其添加到各个资源的profile元素中。

向数据存储中添加新资源时，会验证 FHIR 配置文件。要添加新资源，您可以使用StartFHIRImportJob API 操作、POST请求添加新资源或请求更新现有资源。 PUT

Example— 查看资源中引用了哪个 FHIR 配置文件

配置文件网址将添加到"meta" : "profile"键值对中的profile元素中。为清楚起见，此资源已被截断。

```
{
    "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— 如何引用非默认支持的 FHIR 配置文件

要根据支持的非默认配置文件进行验证，请将版本控制的配置文件 URL 添加到元素中。meta.profile版本控制的 URL 包括基本配置文件 URL，后跟一个竖线字符 (|) 和版本号。为清楚起见，此示例资源已被截断。

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

您还可以同时包含版本控制的 URL 和基本个人资料网址。两种格式均有效。

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

FHIR R4 支持的资源类型适用于 HealthLake

下表列出了支持的 FHIR R4 资源类型。 AWS HealthLake有关更多信息，请参阅 FHIR R4 文档中的资源索引。

支持的 FHIR R4 资源类型 HealthLake

Account	DetectedIssue	Invoice	从业者
ActivityDefinition	设备	图书馆	PractitionerRole
AdverseEvent	DeviceDefinition	联动	过程
AllergyIntolerance	DeviceMetric	列表	出处
预约	DeviceUseStatement	位置	问卷
AppointmentResponse	DeviceRequest	度量	QuestionnaireResponse
AuditEvent -参见备注	DiagnosticReport	MeasureReport	RelatedPerson
二元	DocumentManifest	媒体	RequestGroup
BodyStructure	DocumentReference	药物	ResearchStudy

捆绑包-参见备注	EffectEvidenceSynthesis	MedicationAdministration	ResearchSubject
CapabilityStatement	遭遇	MedicationDispense	RiskAssessment
CarePlan	端点	MedicationKnowledge	RiskEvidenceSynthesis
CareTeam	EpisodeOfCare	MedicationRequest	Schedule
ChargeItem	EnrollmentRequest	MedicationStatement	ServiceRequest
ChargeItemDefinition	EnrollmentResponse	MessageHeader	槽位
声明	ExplanationOfBenefit	MolecularSequence	标本
ClaimResponse	FamilyMemberHistory	NutritionOrder	StructureDefinition
Communication	标记	观察	StructureMap
CommunicationRequest	Goal	OperationOutcome -参见备注	Substance
合成	Group	Organization（组织）	SupplyDelivery
ConceptMap	GuidanceResponse	OrganizationAffiliation	SupplyRequest
条件	HealthcareService	参数-参见注释	Task
同意	ImagingStudy	病人	ValueSet
合同	免疫接种	PaymentNotice	VisionPrescription
涵盖	ImmunizationEvaluation	PaymentReconciliation	VerificationResult -参见备注
CoverageEligibilityRequest	ImmunizationRecommendation	人员	
CoverageEligibilityResponse	InsurancePlan	PlanDefinition	

⚠ FHIR 规格和 HealthLake

- 您不能使用 FHIR `OperationOutcome` 和`Parameters`资源类型发出`GET`或`POST`请求。
- AuditEvent— 可以创建或读取 AuditEvent 资源，但不能对其进行更新或删除。
- 捆绑包 — 有多种 HealthLake 管理捆绑包请求的方法。有关更多详细信息，请参阅捆绑 FHIR 资源。
- VerificationResult— 仅在 2023 年 12 月 9 日之后创建的数据存储支持此资源类型。

FHIR R4 的搜索参数为 HealthLake

使用 FHIR `search`交互根据某些筛选条件在 HealthLake 数据存储中搜索一组 FHIR 资源。可以使用`GET`或`POST`请求执行`search`交互。对于涉及个人身份信息 (PII) 或受保护的健康信息 (PHI) 的搜索，建议使用`POST`请求，因为 PII 和 PHI 是作为请求正文的一部分添加的，并在传输过程中进行加密。

Note

本章中描述的 FHIR `search` 交互符合 HL7 FHIR R4 的医疗保健数据交换标准。由于它代表 HL7 FHIR 服务，因此不是通过 AWS CLI 和 AWS SDKs提供的。有关更多信息，请参阅 FHIR R4 RESTful API 文档`search`中的。
您也可以使用 Amazon Athena 使用 SQL 查询 HealthLake 数据存储。有关更多信息，请参阅集成。

HealthLake 支持以下 FHIR R4 搜索参数子集。有关更多信息，请参阅 FHIR R4 的搜索参数为 HealthLake。

支持的搜索参数类型

下表显示了中支持的搜索参数类型 HealthLake。

支持的搜索参数类型

搜索参数	说明
_id	资源 ID（不是完整网址）

搜索参数	说明
_last Upd	上次更新日期。服务器可以自行决定边界精度。
_tag	按资源标签搜索。
_个人资料	搜索所有标有个人资料的资源。
_安全	搜索应用于此资源的安全标签。
_来源	搜索资源来源。
_text	搜索资源的叙述。
createdAt	在自定义扩展程序上搜索 createdAt。

> **Note**
>
> 以下搜索参数仅适用于 2023 年 12 月 9 日之后创建的数据存储：_security、_source、_text、createAt。

下表显示了如何根据给定资源类型的指定数据类型修改查询字符串的示例。为清楚起见，示例列中的特殊字符未经过编码。要成功查询，请确保查询字符串已正确编码。

搜索参数示例

搜索参数类型	Details	示例

数字	在指定资源中搜索数值。可以观察到重要的数字。有效位数是按搜索参数值确定的，不包括前导零。允许使用比较前缀。	`[parameter]=100`
		`[parameter]=1e2`
		`[parameter]=lt100`

日期/ DateTime	搜索特定的日期或时间。预期的格式是，`yyyy-mm-ddThh:mm:ss[Z|(+|-)hh:mm]` 但可能有所不同。	`[parameter]=eq2013-01-14`

搜索参数类型	Details	示例

	接受以下数据类型：date、dateTimeinstant、Pe和Timing。有关在搜索中使用这些数据类型的更多详细信息，请参阅 FHIR R4 RESTful API 文档中的日期。
	
	允许使用比较前缀。
		`[parameter]=gt2013-01-14T10:00`
		
		`[parameter]=ne2013-01-14`

字符串	以区分大小写的方式搜索字符序列。
	
	同时支持HumanName和Address类型。有关更多详细信息，请参阅 FHIR R4 文档中的Address数据类型条目和数据类型条目。HumanName
	
	使用`:text`修饰符支持高级搜索。
		`[base]/Patient?given=eve`
		
		`[base]/Patient?given:contains=eve`

令牌	搜索 close-to-exact与一串字符的匹配项，通常与一对医疗代码值进行比较。 区分大小写与创建查询时使用的代码系统有关。基于Subsumption的查询可以帮助减少与区分大小写有关的问题。为清楚起见|，尚未编码。	`[parameter]=[system]|[code]`：这里[system]指的是编码系统，[code]指的是在该特定系统中找到的代码值。 `[parameter]=[code]`：在这里，您的输入将匹配代码或系统。 `[parameter]=|[code]`：此处您的输入将与代码匹配，并且系统属性没有标识符。

搜索参数类型	Details	示例
复合键	使用修饰符$和,运算在单个资源类型中搜索多个参数。 允许使用比较前缀。	/Patient?language=FR,NL&language=EN Observation?component-code-value-quantity=http://loinc.org|8480-6$lt60 [base]/Group?characteristic-value=gender$mixed
Quantity	以值形式搜索数字、系统和代码。必须输入数字，但系统和代码是可选的。基于数量数据类型。有关更多详细信息，请参阅 FHIR R4 文档中的数量。 使用以下假设语法 [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
参考	搜索对其他资源的引用。	[base]/Observation?subject=Patient/23 test

搜索参数类型	Details	示例
URI	搜索可明确标识特定资源的字符串。	`[base]/ValueSet?url=http://acme.org/fhir/ValueSet/123`
特殊	基于集成的医疗 NLP 扩展进行搜索。	

支持的高级搜索参数 HealthLake

HealthLake 支持以下高级搜索参数。

Name	说明	示例	能力
`_includ`	用于请求在搜索请求中返回其他资源。它返回目标资源实例引用的资源。	`Encounter?_include=Encounter:subject`	
`_revinc de`	用于请求在搜索请求中返回其他资源。它返回引用主资源实例的资源。	`Patient?_id=` **`patient-identifier`** `&_revinclude=Encounter:patient`	
`_summar`	摘要可用于请求资源的子集。	`Patient?_summary=text`	支持以下摘要参数：`_summary=true` 、`_summary=false` 、`_summary=text` 、`_summary=data` 。
`_elemen`	请求在搜索结果中将一组特定的元素作为资源的一部分返回。	`Patient?_elements=identifie`	

Name	说明	示例	能力
		r,active, link	
_total	返回与搜索参数匹配的资源数量。	Patient?_ total=acc urate	Supp _total=accurate ort，_total=none 。
_sort	使用逗号分隔的列表表示返回的搜索结果的排序顺序。该-前缀可用于逗号分隔列表中的任何排序规则，以表示降序。	Observati on?_sort= status,-date	Support 支持按带有类型的字段进行排序Number, String, Quantity, Token, URI, Reference 。Date仅在 2023 年 12 月 9 日之后创建的数据存储支持排序依据。Support 最多支持 5 个排序规则。
_count	控制搜索包中每页返回多少资源。	Patient?_ count=100	最大页面大小为 100。
chainin	搜索引用资源的元素。将链接搜索.定向到引用资源中的元素。	Diagnosti cReport?s ubject:Pa tient.nam e=peter	
reverse chainin (_has)	根据引用资源的元素搜索资源。	Patient?_ has:Obser vation:pa tient:cod e=1234-5	

_include

_include在搜索查询中使用还允许返回其他指定的 FHIR 资源。用于包括_include向前链接的资源。

Example— 用于_include查找被诊断为咳嗽的患者或患者群体

你可以在指定咳嗽诊断代码的Condition资源类型上进行搜索，然后使用_include指定也要返回该诊断subject的代码。在Condition资源类型中，subject指的是患者资源类型或组资源类型。

为清楚起见，示例中的特殊字符未经过编码。要成功查询，请确保查询字符串已正确编码。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
Condition?code=49727002&_include=Condition:subject
```

_include:iterate修饰符

该_include:iterate修饰符允许以递归方式包含两个级别的引用资源。例如，

```
GET /ServiceRequest?
identifier=025C0931195&_include=ServiceRequest:requester&_include:iterate=PractitionerRole:prac
```

将返回与之 ServiceRequest关联的 PractitionerRole （通过请求者引用），然后递归地包含由其引用的从业者。 PractitionerRole此修改器适用于中的所有资源类型 HealthLake。

_include=*修饰符

_include=*修饰符是一个通配符，它自动包含搜索结果直接引用的所有资源。例如，

```
GET /ServiceRequest?specimen.accession=12345&_include=*
```

将返回匹配项 ServiceRequest 及其引用的所有资源（例如患者、从业人员、标本等），而无需单独指定每个参考路径。此修改器适用于中的所有资源类型 HealthLake。

_revinclude

_revinclude在搜索查询中使用还允许返回其他指定的 FHIR 资源。用于包含_revinclude向后链接的资源。

Example— 用于包括_revinclude与特定患者关联的相关遭遇和观察资源类型

要进行此搜索，首先要Patient通过在_id搜索参数中指定个人的标识符来定义个人。然后，您可以使用结构Encounter:patient和Observation:patient来指定其他 FHIR 资源。

为清楚起见，示例中的特殊字符未经过编码。要成功查询，请确保查询字符串已正确编码。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
Patient?_id=patient-
identifier&_revinclude=Encounter:patient&_revinclude=Observation:patient
```

_summary

_summary在搜索查询中使用允许用户请求 FHIR 资源的子集。它可以包含以下值之一:true, text, data, false. 任何其他值都将被视为无效。返回的资源将在 meta.tag 'SUBSETTED' 中标记，以表示资源不完整。

- true：返回所有在资源基本定义中标记为 “摘要” 的受支持元素。
- text: 仅返回 “文本”、“id”、“meta” 元素，仅返回顶级必填元素。
- data: 返回除 “文本” 元素之外的所有部分。
- false: 返回资源的所有部分

在单个搜索请求中，_summary=text不能与_include或_revinclude搜索参数组合使用。

Example— 获取数据存储中患者资源的 “文本” 元素。

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

_elements

_elements在搜索查询中使用允许请求特定的 FHIR 资源元素。返回的资源将在 meta.tag 'SUBSETTED' 中标记，以表示资源不完整。

该_elements参数由以逗号分隔的基本元素名称列表组成，例如在资源中根级别定义的元素。只有列出的元素才会被返回。如果_elements参数值包含无效元素，服务器将忽略它们并返回必需元素和有效元素。

_elements不适用于包含的资源（搜索模式为的返回资源include）。

在单个搜索请求中，_elements不能与_summary搜索参数组合使用。

Example— 获取 HealthLake 数据存储中患者资源的 “标识符”、“活动”、“链接” 元素。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?
_elements=identifier,active,link
```

_total

_total在搜索查询中使用将返回与请求的搜索参数相匹配的资源数量。 HealthLake 将返回 of search 响应中匹配资源的总数（搜索模式为Bundle.total的返回资源match）。

_total支持accurate、none参数值。 _total=estimate不支持。任何其他值都将被视为无效。 _total不适用于包含的资源（搜索模式为的返回资源include）。

Example— 获取数据存储中患者资源的总数：

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

_sort

_sort在搜索查询中使用可按特定顺序排列结果。结果根据以逗号分隔的排序规则列表按优先顺序排序。排序规则应该是有效的搜索参数。任何其他值都将被视为无效。

在单个搜索请求中，您最多可以使用 5 个排序搜索参数。您可以选择使用-前缀来表示降序。默认情况下，服务器将按升序排序。

支持的排序搜索参数类型为:Number, String, Date, Quantity, Token, URI, Reference. 如果搜索参数指的是嵌套的元素，则排序不支持此搜索参数。例如，搜索资源类型的 “名称” 患者指的是患者。 HumanName 数据类型的名称元素被视为嵌套。因此，不支持按 “姓名” 对患者资源进行排序。

Example— 在数据存储中获取患者资源，并按出生日期升序对其进行排序：

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

_count

该参数_count被定义为向服务器发出的有关应在单个页面中返回多少资源的指令。

最大页面大小为 100。任何大于 100 的值均无效。 _count=0不支持。

Example— 搜索患者资源并将搜索页面大小设置为 25：

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?_count=25
```

Chaining and Reverse Chaining(_has)

FHIR 中的链接和反向链接提供了一种更高效、更紧凑的方式来获取相互关联的数据，从而减少了对多个单独查询的需求，并使开发人员和用户更方便地检索数据。

如果任何级别的递归返回的结果超过 100 个，则 HealthLake 将返回 4xx，以防止数据存储过载并导致多次分页。

Example— Chaining-获取所有 DiagnosticReport 指向患者姓名为 peter 的患者的内容。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/DiagnosticReport?
subject:Patient.name=peter
```

Example— 反向链接-获取患者资源，其中患者资源由至少一个观察点引用，其中观察结果的代码为 1234，其中观察结果指患者搜索参数中的患者资源。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?
_has:Observation:patient:code=1234
```

支持的搜索修饰符

搜索修饰符用于基于字符串的字段。中的所有搜索修饰符都 HealthLake 使用基于布尔值的逻辑。例如，您可以指定:contains较大的字符串字段应包含一个小字符串，以便将其包含在搜索结果中。

支持的搜索修饰符

搜索修饰符	Type
:缺失	除外的所有参数 Composite
:精确	字符串
:包含	字符串
:不是	令牌
:文本	令牌
:标识符	参考

搜索修饰符	Type
:以下	URI

支持的搜索比较器

您可以使用搜索比较器来控制搜索中匹配的性质。在搜索数字、日期和数量字段时，您可以使用比较器。下表列出了支持的搜索比较器及其定义。 HealthLake

支持的搜索比较器

搜索比较器	说明
eq	资源中参数的值等于提供的值。
ne	资源中参数的值不等于提供的值。
gt	资源中参数的值大于提供的值。
lt	资源中参数的值小于提供的值。
ge	资源中参数的值大于或等于提供的值。
le	资源中参数的值小于或等于提供的值。
sa	资源中参数的值从提供的值之后开始。
eb	资源中参数的值在提供的值之前结束。

不支持 FHIR 搜索参数 HealthLake

HealthLake 支持所有 FHIR 搜索参数，但下表中列出的参数除外。有关 FHIR 搜索参数的完整列表，请参阅 FHIR 搜索参数注册表。

不支持的搜索参数

捆绑包构成	位置-附近
捆绑包标识符	C onsent-source-reference

捆绑消息	合同患者
捆绑包类型	资源内容
捆绑包时间戳	资源查询

FHIR R4 适用于 **$operations** HealthLake

FHIR $ 操作（也称为 “美元操作”）是特殊的服务器端函数，其扩展范围超出了 FHIR 规范中的标准 CRUD（`CreateReadUpdate`、、、`Delete`）操作。这些操作由 “$” 前缀标识，支持使用标准 REST API 调用难以执行或效率低下的复杂处理、数据转换和批量操作。$ Operations 可以在系统级别、资源类型级别或特定资源实例上调用，从而提供了与 FHIR 数据交互的灵活方式。 AWS HealthLake 支持多个 FHIR `$operations`。有关更多详细信息，请参阅下面的各个页面。

主题

- FHIR R4 操作适用于 $attribution-status HealthLake
- 使用删除资源类型 $bulk-delete
- $bulk-m ember-Match 操作适用于 HealthLake
- FHIR R4 操作适用于 $confirm-attribution-list HealthLake
- FHIR R4 操作适用于 $davinci-data-export HealthLake
- 使用生成临床文档 $document
- 使用永久删除资源 $erase
- 使用获取患者数据 Patient/$everything
- 使用检索 ValueSet 验证码 $expand
- 使用 FHI HealthLake R 导出数据 $export
- FHIR 的$inquire手术 HealthLake
- 使用检索概念细节 $lookup
- $member-add操作用于 HealthLake
- $member-match操作为 HealthLake
- $member-remove操作用于 HealthLake
- 使用移除患者隔间资源 $purge
- FHIR 的$questionnaire-package手术 HealthLake
- FHIR 的$submit手术 HealthLake

- 使用验证 FHIR 资源 $validate

FHIR R4 操作适用于 **$attribution-status** HealthLake

检索特定成员的归因状态，返回包含与患者相关的所有归因资源的捆绑包。此操作是 FHIR 成员归因 (ATR) 名单 IG 2.1.0 实施的一部分。

端点

```
POST [base]/Group/[id]/$attribution-status
```

请求参数

该操作接受以下可选参数：

参数	类型	说明
memberId	标识符	申请归因状态的成员的 MemberId
患者参考	参考	引用制作人系统中的患者资源

> **Note**
>
> patientReference可以提供memberId或之一，或者两者兼而有之，用于验证目的。

示例请求

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "memberId",
      "valueIdentifier": {
        "system": "http://example.org",
        "value": "MBR123456789"
      }
    },
    {
```

```
      "name": "patientReference",
      "valueReference": {
        "reference": "Patient/patient-123",
        "display": "John Doe"
      }
    }
  ]
}
```

响应

返回包含与患者相关的归因资源的捆绑包：

资源	基数	位置
病人	1..1	群组成员实体
涵盖	0.. 1	Group.member.extension: co
Organization/Practitioner/PractitionerRole	0.. 1	Group.member.extensions:
任何资源	0.. 1	Group.member. 扩展：关联数据

示例响应

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

错误处理

该操作处理以下错误情况：

错误	HTTP 状态	说明
操作请求无效	400	不符合要求的请求参数或结构
未找到群组资源	404	指定的群组 ID 不存在

错误	HTTP 状态	说明
未找到患者资源	404	指定的患者参考文献不存在

授权和安全

智能瞄准镜要求

客户必须具有相应的权限才能读取群组资源和相关归因资源

标准的 FHIR 授权机制适用于所有操作

使用删除资源类型 **$bulk-delete**

AWS HealthLake 支持该$bulk-delete操作，允许删除数据存储中特定类型的所有资源。当您需要执行以下操作时，此操作特别有用：

- 进行季节性审计和清理
- 大规模管理数据生命周期
- 移除特定的资源类型
- 遵守数据保留政策

用法

可以使用 POST 方法调用该$bulk-delete操作：

```
POST [base]/[ResourceType]/$bulk-delete?isHardDelete=false&deleteAuditEvent=true
```

参数

参数	Type	必需	默认值	说明
isHardDelete	布尔值	否	false	如果为 true，则从存储中永久删除资源
deleteAuditEvent	布尔值	否	true	如果为 true，则删除关联的审计事件

参数	Type	必需	默认值	说明
_since	字符串	否	数据存储创建时间	输入后，选择开始截止时间，根据资源的上次修改时间来查找资源。不能与开头或结尾一起使用
start	字符串	否	数据存储创建时间	输入后，根据资源的上次修改时间选择查找资源的截止时间。可以与 end 一起使用
end	字符串	否	Job 提交时间	输入后，选择结束截止时间，根据资源的上次修改时间来查找资源

示例

请求示例

```
POST [base]/Observation/$bulk-delete?isHardDelete=false
```

响应示例

```
{
      "jobId": "jobId",
      "jobStatus": "SUBMITTED"
    }
```

作业状态

要检查批量删除任务的状态，请执行以下操作：

```
GET [base]/$bulk-delete/[jobId]
```

该操作返回任务状态信息：

```
{
      "datastoreId": "datastoreId",
      "jobId": "jobId",
      "status": "COMPLETED",
```

```
    "submittedTime": "2025-10-09T15:09:51.336Z"
  }
```

行为

该$bulk-delete操作：

1. 异步处理以处理大量资源
2. 维护 ACID 事务以保证数据完整性
3. 提供任务状态跟踪，包括资源删除次数
4. 支持软删除和硬删除模式
5. 包括删除活动的全面审核记录
6. 允许有选择地删除历史版本和审计事件

审核日志

$bulk-delete操作记录为 “开始” FHIRBulk DeleteJob 和 “描述” FHIRBulkDeleteJob ，其中包含详细的操作信息。

限制

- 如果设置isHardDelete为 true，则硬删除的资源将不会出现在搜索结果或_history查询中。
- 通过此操作删除的资源在处理过程中可能暂时无法访问
- 存储计量仅针对历史版本进行调整- deleteVersionHistory =false 不会调整数据存储存储

$bulk-member-Match 操作适用于 HealthLake

AWS HealthLake 支持异步处理多个成员匹配请求的$bulk-member-match操作。该操作使医疗保健组织能够在单个批量请求中使用人口统计和保险信息，有效地匹配不同医疗保健系统中的数百个成员的唯一标识符。此功能对于大规模的付款人到付款人数据交换、成员过渡和 CMS 合规性要求至关重要，并且符合 FHIR 规范。

> **Note**
>
> 该$bulk-member-match操作基于FHIR的基本规范，该规范目前处于实验阶段，可能会发生变化。随着规范的发展，此 API 的行为和接口将相应更新。建议开发人员监视 AWS HealthLake 发行说明和相关的 FHIR 规范更新，以随时了解可能影响其集成的任何更改。

当您需要执行以下操作时，此操作特别有用：

- 在开放注册期间大规模处理成员匹配
- 促进付款人之间的批量成员过渡
- Support 支持大规模 CMS 合规数据交换要求
- 在医疗保健网络中高效匹配成员群组
- 最大限度地减少 API 调用并提高大批量匹配场景的运营效率

用法

该$bulk-member-match操作是使用 POST 方法对组资源调用的异步操作：

```
POST [base]/Group/$bulk-member-match
```

提交批量匹配请求后，您可以使用以下方式轮询任务状态：

```
GET [base]/$bulk-member-match-status/{jobId}
```

支持的参数

HealthLake 支持以下 FHIR $bulk-member-match 参数：

参数	Type	必需	说明
MemberPatient	病人	是	包含要匹配的成员的人口统计信息的患者资源。
CoverageToMatch	涵盖	是	覆盖范围资源，将用于与现有记录进行匹配。
CoverageToLink	涵盖	否	在匹配过程中要链接的覆盖范围资源。
Consent	同意	是	还会存储用于授权目的的同意资源。这与不需要征得同意的个人$member-match 操作不同。

POST 请求提交批量成员匹配任务

以下示例显示了提交批量成员匹配作业的 POST 请求。每个成员都封装在包含必需的MemberPatientCoverageToMatch、和Consent资源以及可选MemberBundle参数的参数中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"
              }
            ]
          }
        }
      ]
    }
  ]
}
```

已完成任务响应，并附有输出

任务完成后，响应将包括任务元数据和一个 FHIR Parameters 资源，其中包含三个对匹配结果进行分类的组资源。

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

如何 HealthLake 将成员分类为输出组

在$bulk-member-match请求中提交的每个成员都将通过顺序管道进行评估。每个步骤的结果决定了将该成员置于哪个输出组中。

1. 结构验证 — 是否 MemberBundle 符合要求的配置文件？ 失败时：错误（不在任何组中）。
2. 患者匹配 —能否 HealthLake 找到与提交的人口统计数据相匹配的患者？ 失败时: NonMatchedMembers.
3. 承保范围确认 ——能否将 HealthLake 范围缩小到只有一名有效的患者 CoverageToMatch？ 失败时: NonMatchedMembers.
4. 同意评估 —提交的同意书现在是否合格？（状态 = 活跃，期间涵盖当前日期，可以验证执行者）。失败时: ConsentConstrainedMembers.
5. 成功-所有检查均通过。同意存储在数据存储中。成员已放入 MatchedMembers。

关键原则：一个成员只能出现在一个目的地。第一个失败的步骤决定了位置。未能通过第 2 步或第 3 步的成员永远不会进入该群组 ConsentConstrainedMembers ——该群组专为成功匹配但无法兑现其同意的成员而设。

同意评估详情（步骤 4）：

- 检查 1-同意状态：Consent.status等于 “激活” 吗？ 如果不是 → ConsentConstrainedMembers。

- 支票 2 — 供应期：是否provision.period涵盖当前日期？ 如果当前日期早于period.start或之后 period.end → ConsentConstrainedMembers.
- 检查 3-执行者验证：能否验证Consent.performer参考文献？ 如果在数据存储中找不到引用的资源，或者与匹配的患者没有关联 → ConsentConstrainedMembers。

必须通过所有支票才能将成员安置在里 MatchedMembers 面，同意书才能被存储。

覆盖范围匹配行为

在成员匹配期间，CoverageToMatch仅根据响应付款人的数据存储进行验证。 CoverageToLink属于 new/requesting 付款人，且未根据旧付款人的数据存储进行验证。包含CoverageToLink在请求中不会影响匹配结果。

申请中的每个 “患者+保险” 组合都是独立处理的。同一位患者可以多次提交不同的保险计划，并且每个参赛作品都会根据其特定的承保范围获得自己的结果。

同意执行者推荐人处理

新的付款人可能会在中发送临时或本地患者推荐信Consent.performer（例如，中使用的同一推荐信Consent.patient）。 HealthLake 会自动解析这些引用：

- 如果Consent.performer包含与相同的局部参考文献Consent.patient，则在匹配成功后将其 HealthLake 替换为实际匹配的患者参考文献。
- HealthLake 支持 “患者” RelatedPerson、“从业者” 和 “组织” 类型的表演者引用（包括直接引用和逻辑标识符引用）。 PractitionerRole
- 如果执行者验证失败（未找到资源或资源与匹配的患者无关），则会将成员置于 ConsentConstrainedMembers 其中，而不是返回错误。

输出组资源

已完成的任务将返回一个包含三个组资源的参数资源：

MatchedMembers Group

包含所有成功匹配的成员的患者推荐信，这些成员的同意在请求时已激活且有效。将为每个匹配的成员创建意见征求资源并将其存储在数据存储中。此组在数据存储中实例化，可以直接与一起使用。$davinci-data-export

NonMatchedMembers Group

包含对未找到唯一匹配项的成员的引用。当数据存储中没有患者与所提供的人口统计数据相匹配，任何匹配的候选患者都没有有效的保险，或者多名患者与人口统计数据相匹配且多名患者具有有效的覆盖范围（模棱两可）时，将成员置于此处。

ConsentConstrainedMembers Group

包含成功匹配（已确认人口统计和覆盖范围）但在请求时无法兑现其同意的成员的患者推荐信。不会为受许可限制的成员存储意见征求资源。匹配的成员身份（MemberIdentifier 和 MemberId）仍包含在内，因此提出请求的付款人知道谁受到了限制。

该`Group.quantity`字段包含其各自组中的成员总数。

小组成员参考资料：

- `Group.member.entity.reference`— 对于 MatchedMembers 和 ConsentConstrainedMembers，包含响应付款人系统中匹配成员的患者 ID。对于 NonMatchedMembers，引用包含的输入 “患者”。
- `Group.member.entity.extension (base-ext-match-parameters)`— 包含原始输入请求中的患者 ID（请求付款人提交的身份证，源自`Patient.idCoverage.beneficiary.reference`、或`Consent.patient.reference`）。

Consent-Patient 链接

> **Important**
>
> 存储的同意资源保留的患者参考信息与申请付款人提交的信息完全相同。 HealthLake 不会自动更新 “同意” 的 “患者” 字段以指向接收数据存储中匹配的患者。

要将存储的同意书链接到匹配的患者，请使用任务输出： MatchedMembers 组中的每个成员都有一个`member.entity.reference`指向匹配患者的指向，一个 `member.entity.extension (base-ext-match-parameters)` 指向包含的输入患者。 Cross-reference 这些带有同意的患者字段，用于在您的应用程序层中构建映射。

存储的内容与瞬态存储的内容

下表记录了`$bulk-member-match`处理期间数据存储中 HealthLake 保留的内容以及仅存在于任务响应中的内容：

资源	已存储？	可以通过 REST 查询吗？	注意
MemberPatient（输入）	否	否	仅用于匹配；不用于保存
CoverageToMatch（输入）	否	否	仅用于确认承保范围
CoverageToLink（输入）	否	否	未针对数据存储进行验证；属于新的付款人
同意（匹配的成员）	是	是 — GET [base] / Consent/ {id}	按收到请求的付款人收到的款项存储
同意（受限制的成员）	否	否	未存储。回复中仍包含成员身份。
MatchedMembers 组（输出）	是	是 — GET [base] / Group/ {id}	已实例化；可与 $davinci-data-export 配合使用
NonMatchedMembers 群组	否	否	仅限 Job 响应
ConsentConstrained Members 群组	否	否	仅限 Job 响应

与 $davinci-data-export

返回的 MatchedMembers 组资源`$bulk-member-match`可以直接用于检索批量成员数据的`$davinci-data-export`操作：

```
POST [base]/Group/{matched-group-id}/$davinci-data-export
GET [base]/Group/{matched-group-id}
```

这种集成可实现高效的工作流程，您可以首先批量识别匹配的成员，然后使用生成的群组资源导出他们的完整健康记录。

在导出之前使用 $member-remobe

如果您需要在匹配后将特定成员排除在导出范围之外（例如，成员在匹配和导出之间撤消同意），请在 MatchedMembers 群组$member-remove上使用。

> **Important**
>
> 通过删除成员会将该成员$member-remove标记为在群组中处于非活动状态，但$davinci-data-export仅在群组更新为 “最终” 状态后才会排除不活跃的成员。如果您调用$davinci-data-export仍处于默认状态的群组，则已移除的成员仍可能出现在导出结果中。

工作流：

1. POST [base]/Group/{id}/$member-remove— 将成员标记为非活动状态
2. PUT [base]/Group/{id}— 将群组状态更新为 “最终”
3. POST [base]/Group/{id}/$davinci-data-export— 导出现在不包括已删除的成员

性能特征

该$bulk-member-match操作专为大容量处理而设计，并且是异步运行的：

- 并@@ 发：每个数据存储最多 5 个并发操作。
- 可扩展性：每个请求最多可处理 500 个成员（有效载荷限制 5 MB）。
- 并行操作：与并行导入、导出或批量删除操作兼容。

Authorization

API 在 FHIR 上使用 SMART 授权协议，其所需范围如下：

- system/Patient.read— 搜索和匹配患者资源所必需的。
- system/Coverage.read— 验证覆盖范围信息所必需的。
- system/Group.write— 创建结果组资源所必需的。
- system/Organization.read— 视情况而定，如果承保范围涉及组织，则为必填项。
- system/Practitioner.read— 有条件，如果承保范围涉及从业人员，则为必填项。
- system/PractitionerRole.read— 视情况而定，如果保险范围涉及从业者的角色，则为必填项。

- `system/Consent.write`— 有条件的，如果提供了同意资源，则为必填项。

该操作还支持用于编程 AWS 访问的 IAM 签名版本 4 (Sigv4) 授权。

验证规则

在步骤 1 中，以下验证规则适用于每个 MemberBundle 规则。未通过验证的成员会被报告为错误，并且不会出现在任何输出组中。

MemberPatient

字段	如何 HealthLake 使用它	如果出现以下情况，则验证失败
`name.family`	人口统计搜索	缺失
`name.given`	人口统计搜索	缺失（至少需要一个）
`birthDate`	人口统计搜索	缺失
`gender`	人口统计搜索；如果没有，则使用出生性别扩展名	既不存在性别也不存在出生性别（hrex-pat-1）
`identifier`	在搜索中包含在搜索中；提高信心	从不导致故障（可选）

CoverageToMatch / CoverageToLink

字段	如何 HealthLake 使用它	如果出现以下情况，则验证失败
`status`	确认承保范围可付诸行动	缺失
`beneficiary`	将承保范围与患者候选人联系起来	缺失
`payor`	存在多个候选人时消除歧义	缺失或不止一个付款人
`relationship`	确认订阅者与受益人的关系	缺失
`identifier (MB) or subscriberId`	主要消除歧义的关键	都不在场

同意

字段	如何 HealthLake 使用它	如果出现以下情况，则验证失败
`scope`	确认同意范围为患者隐私	缺少或没有患者隐私码
`category`	确认披露分类	缺失
`patient`	标识同意主体	缺失
`performer`	确定谁表示同意	缺失
`sourceReference`	记录同意来源	缺失
`policy.uri`	确定数据共享范围	缺失或 URI 未以 #regular 或 #sensitive 结尾
`provision.type`	必须根据 HreX 同意资料获得 “许可”	缺少或不是 “许可”（包括 “拒绝”）
`provision.period`	在步骤 4 中针对受同意约束的检查进行了评估	缺失或没有 start/end
`status`	在步骤 4（不是步骤 1）中进行评估	从不导致步骤 1 失败 — HealthLake 接受任何有效状态并在步骤 4 处进行评估

> **Note**
>
> HRex 同意配置文件使用固定值为 “活跃” 来定义状态。 HealthLake 故意放宽这一限制，使非主动同意获得有意义的分类 (ConsentConstrainedMembers)，而不是一揽子验证拒绝。

匹配行为

- 患者搜索（步骤 2）— 使用 `name.family` +`name.given`（精确、不区分大小写）、（精确）、`birthDate`（精确；如果不存在性别则使用出生性别）和`identifier`（如果存在则可选）进行 HealthLake 搜索。`gender`

- 消除@@ 覆盖范围歧义（步骤 3）— 当找到多个候选患者CoverageToMatch时，使用缩小到一个。如果数据存储中存在与subscriberId或identifier（MB 类型）和（MB 类型）匹配的活动 Coverage 资源，则覆盖范围为 “有效”。payor
- 同意评估（步骤 4）— 仅在成功完成唯一匹配后执行。请参阅上面的用户意见征求评估详情部分。

错误处理

该操作处理以下错误情况：

- 400 错误请求：请求格式无效、缺少必填参数或有效负载超过大小限制（500 个成员或 5 MB）。
- 422 无法处理的实体：任务执行期间的处理错误。
- 个人成员错误：当特定成员未能处理时，将继续对剩余成员进行操作，并在 NonMatchedMembers 群组中包含错误详细信息以及相应的原因代码。例如，MemberBundle如果患者缺少该birthDate参数，则会返回以下错误：

```
"errors": [
  {
    "memberIndex": 1,
    "jsonBlob": {
      "resourceType": "OperationOutcome",
      "issue": [
        {
          "severity": "error",
          "code": "invalid",
          "diagnostics": "MemberPatient.birthDate is required"
        }
      ],
      "statusCode": 400
    }
  }
]
```

错误详情可通过状态轮询端点获得，包括：

- numberOfMembersWithCustomerError：存在验证或输入错误的成员数量。
- numberOfMembersWithServerError：出现服务器端处理错误的成员数。

相关 操作

- the section called “$member-Match”— 个人成员匹配操作。
- the section called “$davinci-data-export”— 使用群组资源批量导出数据。
- the section called “$运营”— 支持的操作的完整列表。

FHIR R4 操作适用于 **$confirm-attribution-list** HealthLake

向制作人表明，消费者无需对归因列表进行任何更改。此操作通过从列表中移除不活跃的成员、将状态更改为 “最终” 并确认操作来最终确定归因列表。此操作是 FHIR 成员归因 (ATR) 名单 IG 2.1.0 实施的一部分。

端点

```
POST [base]/Group/[id]/$confirm-attribution-list
```

请求

无需任何参数。

```
POST [base]/Group/[id]/$confirm-attribution-list
```

响应

返回带有确认消息的 HTTP 201。

示例响应

```
HTTP Status Code: 201

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "message",
      "valueString": "Accepted."
    }
  ]
}
```

确认后的群组状态

成功确认后，群组资源的归因列表状态将设置为 “最终”：

```
{
  "resourceType": "Group",
  "id": "fullexample",
  "extension": [
    {
      "url": "http://hl7.org/fhir/us/davinci-atr/StructureDefinition/ext-
attributionListStatus",
      "valueCode": "final"
    }
  ]
  // ... other Group properties
}
```

错误处理

该操作处理以下错误情况：

错误	HTTP 状态	说明
无效的操作请求	400	不符合要求的请求参数或结构
未找到群组资源	404	指定的群组 ID 不存在

授权和安全

智能瞄准镜要求

客户必须具有相应的权限才能读取群组资源和相关归因资源

对于$confirm-attribution-list，客户机还必须具有写入权限才能修改组资源

标准的 FHIR 授权机制适用于所有操作

FHIR R4 操作适用于 **$davinci-data-export** HealthLake

该$davinci-data-export操作是一种异步 FHIR 操作，可用于从中 AWS HealthLake导出医疗保健数据。此操作支持多种导出类型，包括成员归因 (ATR)、 PDex 提供商访问和成员访问权限 APIs。Payer-to-Payer它是标准 FHIR $export 操作的专用版本，旨在满足 DaVinci 实施指南的要求。

主要功能

- 异步处理：遵循标准的 FHIR 异步请求模式
- 组级导出：导出特定群组资源内成员的数据
- 多种导出类型：支持 ATR（成员归因）、 PDex 提供商访问和成员访问权限 Payer-to-Payer APIs
- 全面的个人资料 Supp ort：包括 US Core、CARIN Blue Button 和 PDex 个人资料
- 灵活筛选：支持按患者、资源类型和时间范围进行筛选
- NDJSON 输出：以换行符分隔的 JSON 格式提供数据

操作终端节点

```
GET [base]/Group/[id]/$davinci-data-export
POST [base]/Group/[id]/$davinci-data-export
```

请求参数

参数	基数	说明
patient	0。。 *	要导出其数据的特定成员。如果省略，则导出组中的所有成员。
_type	0.. 1	要导出的 FHIR 资源类型列表，以逗号分隔。省略时，将包括指定导出类型的所有支持的资源类型。对于 ATR 导出，这默认为 8 种归因资源类型。对于 PDex 出口，这包括所有归因资源类型以及来自美国核心、CARIN Blue Button 和 PDex 概况的临床和索赔资源类型。
_since	0.. 1	仅包括在此日期和时间之后更新的资源。
_until	0.. 1	仅包括在此日期和时间之前更新的资源。

参数	基数	说明
exportType	0.. 1	要执行的导出类型。有效值：hl7.fhir.us.davinci-atr 、hl7.fhir.us.davinci-pdex 、hl7.fhir.us.davinci-pdex.p2p 、hl7.fhir.us.davinci-pdex.member 。默认值：hl7.fhir.us.davinci-atr 。
_includeEOB2xWoFinancial	0.. 1	指定是否包含已删除财务数据的 CARIN BB 2.x ExplanationOfBenefit 资源。默认值：false。
_security	0。。 *	按meta.security 编码值筛选导出的资源。使用system|code 格式（管道字符必须以 URL 编码为）。%7C当提供多个值时，资源必须匹配所有值（和语义）。使用system|（尾随管道，无代码）来匹配给定系统中的任何代码。
_tag	0。。 *	按meta.tag编码值筛选导出的资源。使用与相同的system|code 格式和 AND 语义。_security 如果同时指定_security 了和，_tag则资源必须与两个过滤器匹配。

支持的资源类型

支持的资源类型取决于您指定的导出类型。对于 ATR 导出，支持以下资源类型：

- Group
- Patient
- Coverage
- RelatedPerson
- Practitioner
- PractitionerRole
- Organization
- Location

对于 PDex 导出（提供者访问权限和成员访问权限），除了上述类型外，还支持所有临床和索赔资源类型。 Payer-to-Payer有关支持的资源类型的完整列表，请参阅《美国核心实施指南》(STU 6.1)、《CARIN Blue Button 实施指南》和《达芬奇事先授权支持实施指南》。

导出类型

该$davinci-data-export操作支持以下导出类型。您可以使用exportType参数指定导出类型。

导出类型	用途	数据范围	时间限制
hl7.fhir.us.davinci-atr	成员归因列表	与归因相关的资源	无
hl7.fhir.us.davinci-pdex	提供商访问权限 API	归因患者的临床和索赔数据	5 年了
hl7.fhir.us.davinci-pdex.p2p	Payer-to-Payer 交易所	保险过渡的成员历史数据	5 年了
hl7.fhir.us.davinci-pdex.member	成员访问权限 API	会员自己的健康数据	5 年了

> **Note**
>
> 对于 PDex 出口，5 年期限不适用于 ATR 资源类型（Group、、Patient、、Coverage、RelatedPerson、Practitioner、PractitionerRoleOr无论年龄大小，这些资源始终包括在内。

ATR (hl7.fhir.us.davinci-atr)

使用 ATR 导出类型，您可以导出成员归因列表数据。使用此导出类型为群组中的成员检索与归因相关的资源。欲了解更多信息，请参阅达芬奇 ATR 出口业务。

支持的资源类型

```
Group, Patient, Coverage, RelatedPerson, Practitioner, PractitionerRole,
Organization, Location
```

时间过滤

不应用任何时间过滤。无论日期如何，都会导出所有匹配的资源。

PDex 导出类型

所有 PDex 导出类型都使用相同的支持的配置文件和筛选逻辑。有关更多信息，请参阅 Da Vinci PDex 提供商访问权限 API。支持以下配置文件：

- 美国核心 3.1.1、6.1.0 和 7.0.0
- PDex 事先授权（不支持会员访问）
- CARIN BB 2.x 基本概况：住院机构、门诊机构、专业、口腔 NonClinician、药房

对于 PDex 出口，系统会自动发现集团中每位患者的临床和理赔资源。您无需在组资源中明确引用这些资源。该手术会搜索属于归因患者的所有患者隔间资源（例如`ObservationConditionCoverageRelatedPersonMedicationRequest`、、、、和`ExplanationOfBenefit`）。只有`PatientGroup`、和 non-patient-compartment ATR 类型（`Practitioner`、、`PractitionerRoleOrganization`、`Location`）需要在组中进行显式引用。

提供商访问权限 (`hl7.fhir.us.davinci-pdex`)

使网络内提供者能够检索归因患者的患者数据。

Payer-to-Payer (`hl7.fhir.us.davinci-pdex.p2p`)

当患者更换保险时，允许付款人之间进行数据交换。

成员访问权限 (`hl7.fhir.us.davinci-pdex.member`)

允许成员访问自己的健康数据。这种导出类型可能包括索赔资源中的财务数据。

个人资料 Support 和包含逻辑

对于 PDex 导出，该`$davinci-data-export`操作使用`meta.profile`元素中的配置文件声明来确定要在导出中包含哪些资源。

ExplanationOfBenefit 资源处理

`ExplanationOfBenefit`(EOB) 资源根据其`meta.profile`申报被纳入或排除在 PDex 出口之外：

- ExplanationOfBenefit 带有 CARIN BB 1.x 配置文件的资源不包括在导出范围内。
- ExplanationOfBenefit 未`meta.profile`设置的资源将从导出中排除。
- ExplanationOfBenefit 始终包含具有 CARIN BB 2.x Basis 配置文件的资源。
- ExplanationOfBenefit 默认情况下，不包括包含财务数据的 CARIN BB 2.x 配置文件的资源。设置后`_includeEOB2xWoFinancial=true`，它们将包含在去除的财务数据中，并将资源转换为相应的 Basis 配置文件。
- ExplanationOfBenefit 带有 “ PDex 事先授权” 配置文件的资源始终包括在内。

财务数据转换

设置后`_includeEOB2xWoFinancial=true`，该操作会通过删除财务数据将 CARIN BB 2.x ExplanationOfBenefit 资源转换为相应的 Basis 配置文件。例如，将`C4BB ExplanationOfBenefit Oral`资源转换为`C4BB ExplanationOfBenefit Oral Basis`，这会根据 FHIR 规范从记录中删除财务数据。

转换期间会移除以下财务数据元素：

- 对元素进行所有切片 `total`
- 所有带`amounttype`切片的`adjudication`元素
- 所有`item.adjudication`包含金额信息的元素

该操作还会在转换过程中更新配置文件元数据：

- `meta.profile`已更新为 Basis 配置文件规范 URL
- 版本已更新至 CARIN BB 2.x Basis 版本
- 数据存储中的现有资源未被修改
- 导出的资源不会保留回数据存储中

配置文件检测规则

该操作使用以下规则来检测和验证配置文件：

- 版本检测基于规`meta.profile`范 URLs
- 如果资源声明的任何配置文件符合导出标准，则包含该资源
- 配置文件验证是在导出处理过程中进行的

为期五年的 PDex 出口时间过滤

对于所有 PDex 导出类型，根据资源上次更新时间 HealthLake 应用 5 年时间过滤器。时间过滤器适用于除以下核心归因资源类型之外的所有资源，这些资源无论使用年限如何，都将始终导出：

- Patient
- Coverage
- Organization
- Practitioner
- PractitionerRole
- RelatedPerson
- Location
- Group

这些管理和人口资源是免税的，因为它们为导出的数据提供了基本的背景信息。ATR 导出不受任何时间筛选。

示例请求

以下示例说明如何启动不同导出类型的导出任务。

ATR 导出

```
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"
    }
  }
}
```

提供商访问导出并删除 ExplanationOfBenefit 财务数据

```
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 出口

```
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
```

导出特定患者的成员访问权限

```
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
```

示例响应

```
{
  "datastoreId": "eaee622d8406b41eb86c0f4741201ff9",
  "jobStatus": "SUBMITTED",
  "jobId": "48d7b91dae4a64d00d54b70862f33f61"
}
```

资源关系

该操作根据资源在成员归因列表中的关系导出资源：

```
Group (Attribution List)
### Patient (Members)
### Coverage # RelatedPerson (Subscribers)
### Practitioner (Attributed Providers)
### PractitionerRole # Location
### Organization (Attributed Providers)
```

> **Note**
>
> 前面的资源关系图适用于 ATR 导出。对于 PDex 导出，临床和索赔资源是通过患者搜索自动发现的，不需要在集团资源中明确引用。

资源来源

资源	来源位置	说明
Patient	Group.member.entity	属于归因列表成员的患者
Coverage	Group.member.extension:coverageReference	导致患者会员资格的承保范围
Organization	Group.member.extension:attributedProvider	患者归因于的组织
Practitioner	Group.member.extension:attributedProvider	患者归因于的个人从业者
PractitionerRole	Group.member.extension:attributedProvider	患者所扮演的从业者角色
RelatedPerson	Coverage.subscriber	报道的订阅者
Location	PractitionerRole.location	与从业者角色相关的地点
Group	输入端点	归因列表本身

Job 管理

查看 Job 状态

```
GET [base]/export/[job-id]
```

取消作业

```
DELETE [base]/export/[job-id]
```

作业生命周期

- SUBMITTED-已接收 Job 并已排队
- IN_PROGRESS-Job 正在处理中
- COMPLETED-Job 成功完成，文件可供下载
- FAILED-Job 遇到了错误

输出格式

- 文件格式：NDJSON（以换行符分隔的 JSON）
- 文件组织：每种资源类型都有单独的文件
- 文件扩展名：.ndjson
- 位置：指定的 S3 存储桶和路径

错误处理

OperationOutcome 对于以下情况，该操作会返回 HTTP 400 错误请求，其中包含一个：

授权错误

中指定的 IAM 角色`DataAccessRoleArn`没有足够的权限来执行导出操作。有关所需的 S3 和 KMS 权限的完整列表，请参阅为导出任务设置权限。

参数验证错误

- 该`patient`参数的格式未设置为 `Patient/id,Patient/id,...`
- 一个或多个患者推荐无效或不属于指定群体
- `exportType`参数值不是支持的导出类型

- 该_type参数包含指定导出类型不支持的资源类型
- _type参数缺少hl7.fhir.us.davinci-atr导出类型所需的资源类型 (GroupPatient、、Coverage)
- _includeEOB2xWoFinancial参数值不是有效的布尔值

资源验证错误

- 数据存储中不存在指定的组资源
- 指定的群组资源没有成员
- 一个或多个小组成员引用了数据存储中不存在的患者资源

安全和授权

- 适用标准的 FHIR 授权机制
- 数据访问角色必须具有 S3 和 KMS 操作所需的 IAM 权限。有关所需权限的完整列表，请参阅为导出任务设置权限。

最佳实践

- 资源类型选择：仅请求所需的资源类型，以最大限度地减少导出大小和处理时间
- 基于时间的筛选：使用_since参数进行增量导出
- 患者筛选：当您只需要特定成员的数据时，请使用该patient参数
- Job 监控：定期检查大宗出口的任务状态
- 错误处理：为失败的作业实现正确的重试逻辑
- 时间过滤器感知：对于 PDex 导出，在选择资源类型时，请考虑使用 5 年时间过滤器
- 删除财务数据：_includeEOB2xWoFinancial=true当您需要不包含财务信息的索赔数据时使用
- 配置文件管理：确保资源具有适当的配置文件声明，在摄取之前根据目标配置文件进行验证，并使用配置文件版本控制导出行为

限制

- patient参数中最多可以指定 500 名患者
- 导出仅限于组级操作
- 仅支持每种导出类型的预定义资源类型集

- 输出始终采用 NDJSON 格式
- PDex 出口仅限于 5 年的临床和索赔数据
- 财务数据转换仅适用于 CARIN BB 2.x 配置文件 ExplanationOfBenefit

其他资源

- 达芬奇会员归因列表 IG
- Da Vinci Payer Data Exchange IG
- CARIN 消费者导向付款人 Data Exchange IG
- 美国核心实施指南
- FHIR 批量数据访问规范

使用生成临床文档 **$document**

AWS HealthLake 现在支持组合资源的$document操作，使您能够通过将组合及其所有引用资源捆绑到一个统一的包中来生成完整的临床文档。此操作对于需要满足以下要求的医疗保健应用程序至关重要：

- 创建标准化临床文档
- 交换完整的患者记录
- 存储全面的临床文档
- 生成包含所有相关背景的报告

用法

可以使用 GET 和 POST 方法在组合资源上调用该$document操作：

支持的操作

```
GET/POST [base]/Composition/[id]/$document
```

支持的参数

HealthLake 支持以下 FHIR $document 参数：

参数	Type	必需	默认值	说明
persist	布尔值	否	false	表示服务器是否应存储生成的文档包的布尔值

示例

获取请求

```
GET [base]/Composition/180f219f-97a8-486d-99d9-ed631fe4fc57/$document?persist=true
```

带参数的 POST 请求

```
POST [base]/Composition/180f219f-97a8-486d-99d9-ed631fe4fc57/$document
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "persist",
      "valueBoolean": true
    }
  ]
}
```

示例响应

该操作返回一个 “文档” 类型的 Bundle 资源，其中包含构图和所有引用的资源：

```
{
  "resourceType": "Bundle",
  "id": "180f219f-97a8-486d-99d9-ed631fe4fc57",
  "type": "document",
  "identifier": {
    "system": "urn:ietf:rfc:3986",
    "value": "urn:uuid:0c3151bd-1cbf-4d64-b04d-cd9187a4c6e0"
  },
  "timestamp": "2024-06-21T15:30:00Z",
```

```
  "entry": [
    {
      "fullUrl": "http://example.org/fhir/Composition/180f219f-97a8-486d-99d9-
ed631fe4fc57",
      "resource": {
        "resourceType": "Composition",
        "id": "180f219f-97a8-486d-99d9-ed631fe4fc57",
        "status": "final",
        "type": {
          "coding": [
            {
              "system": "http://loinc.org",
              "code": "34133-9",
              "display": "Summary of Episode Note"
            }
          ]
        },
        "subject": {
          "reference": "Patient/example"
        },
        "section": [
          {
            "title": "Allergies",
            "entry": [
              {
                "reference": "AllergyIntolerance/123"
              }
            ]
          }
        ]
      }
    },
    {
      "fullUrl": "http://example.org/fhir/Patient/example",
      "resource": {
        "resourceType": "Patient",
        "id": "example",
        "name": [
          {
            "family": "Smith",
            "given": ["John"]
          }
        ]
      }
```

```
    },
    {
      "fullUrl": "http://example.org/fhir/AllergyIntolerance/123",
      "resource": {
        "resourceType": "AllergyIntolerance",
        "id": "123",
        "patient": {
          "reference": "Patient/example"
        },
        "code": {
          "coding": [
            {
              "system": "http://snomed.info/sct",
              "code": "418689008",
              "display": "Allergy to penicillin"
            }
          ]
        }
      }
    }
  ]
}
```

行为

该$document操作：

1. 将指定的合成资源作为文档的基础
2. 识别并检索组合直接引用的所有资源
3. 将构图和所有引用的资源打包成一个 “文档” 类型的捆绑包
4. 当 persist 参数设置为 true 时，将生成的文档包存储在数据存储中
5. 识别和检索合成间接引用的资源，以生成全面的文档

该$document操作目前支持按以下格式检索资源引用：

1. ```
   GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Resource/id
   ```
2. 资源/ID

合成资源中不支持的资源引用将从生成的文档中过滤掉。
   ```

错误处理

该操作处理以下错误情况：

- 400 错误请求：$document操作无效（请求不一致），或者如果在 persist 设置为 true 时由于筛选出引用而导致生成的文档未通过 FHIR 验证
- 404 未找到：未找到合成资源

有关$document操作规范的更多信息，请参阅 FHIR R4 组合$document文档。

使用永久删除资源 $erase

AWS HealthLake 支持该$erase操作，允许永久删除特定资源及其历史版本。当您需要执行以下操作时，此操作特别有用：

- 永久移除个别资源
- 删除特定的版本历史记录
- 管理单个资源生命周期
- 遵守特定的数据删除要求

用法

可以在两个级别上调用该$erase操作：

资源实例级别

```
POST [base]/[ResourceType]/[ID]/$erase?deleteAuditEvent=true
```

特定版本级别

```
POST [base]/[ResourceType]/[ID]/_history/[VersionID]/$erase
```

参数

参数	Type	必需	默认值	说明
deleteAuditEvent	布尔值	否	false	如果为 true，则删除关联的审计事件

示例

请求示例

```
POST [base]/Patient/example-patient/$erase
```

响应示例

```
{
      "jobId": "5df47e2f51ff3c731847678cb8cad48e",
      "jobStatus": "SUBMITTED"
    }
```

作业状态

要检查擦除作业的状态，请执行以下操作：

```
GET [base]/$erase/[jobId]
```

该操作返回任务状态信息：

```
{
      "datastoreId": "36622996b1fcecb7e12ee2ee085308d3",
      "jobId": "5df47e2f51ff3c731847678cb8cad48e",
      "status": "COMPLETED",
      "submittedTime": "2025-10-30T16:39:24.160Z"
    }
```

行为

该$erase操作：

1. 异步处理以确保数据完整性
2. 维护 ACID 交易
3. 提供作业状态跟踪
4. 永久移除指定资源及其版本
5. 包括删除活动的全面审核记录
6. 支持选择性删除审计事件

审核日志

$erase操作记录 DeleteResource 与用户 ID、时间戳和资源详细信息相同。

限制

- $erased资源不会出现在搜索结果或_history查询中。
- 正在删除的资源在处理过程中可能暂时无法访问
- 资源被永久删除后，存储计量会立即调整

使用获取患者数据 Patient/$everything

该Patient/$everything操作用于查询 FHIR Patient 资源以及与之相关的任何其他资源。Patient该手术可用于为患者提供访问其全部记录的权限，也可以让提供者执行与患者相关的批量数据下载。 HealthLakePatient/$everything为特定患者提供支持id。

Patient/$everything是一个 FHIR REST API 操作，可以调用该操作，如以下示例所示。

GET request

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything
```

> **Note**
>
> 响应中的资源按资源类型和资源排序id。
> 响应中总是填充有Bundle.total。

`Patient/$everything` 参数

HealthLake 支持以下查询参数

参数	Details
开启	获取指定开始日期之后的所有Patient数据。
最终	获取指定结束日期之前的所有Patient数据。
since	获取指定日期之后更新的所有Patient数据。
_type	获取特定资源类型的Patient数据。
_count	获取Patient数据并指定页面大小。

Example-获取指定开始日期之后的所有患者数据

Patient/$everything可以使用start筛选器仅查询特定日期之后的数据。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?start=2024-03-15T00:00:00.000Z
```

Example-获取指定结束日期之前的所有Patient数据

patient $every end thing 只能使用过滤器来查询特定日期之前的数据。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?end=2024-03-15T00:00:00.000Z
```

Example-获取在指定日期之后更新所有Patient数据

Patient/$everything可以使用since筛选器仅查询在特定日期之后更新的数据。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?since=2024-03-15T00:00:00.000Z
```

Example-获取特定资源类型的Patient数据

patient $everything 可以使用_type过滤器来指定要包含在响应中的特定资源类型。可以在逗号分隔的列表中指定多种资源类型。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?_type=Observation,Condition
```

Example-获取Patient数据并指定页面大小

病人 $everything 都可以使用_count来设置页面大小。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?_count=15
```

Patient/$everythingstart和end属性

HealthLake 支持Patient/ $everythingstart和end查询参数的以下资源属性。

资源	资源元素
Account	账户。服务周期。开始
AdverseEvent	AdverseEvent. 日期
AllergyIntolerance	AllergyIntolerance. RecordedDate
预约	预约. 开始
AppointmentResponse	AppointmentResponse.start
AuditEvent	AuditEvent.period.start
Basic	basic.created
BodyStructure	NO_DATE
CarePlan	CarePlan.period.start
CareTeam	CareTeam.period.start

资源	资源元素
ChargeItem	ChargeItem。 occurrenceDateTime， ChargeItem.occurrencePeriod.start，.occurrenceTiming.event ChargeItem
声明	Claim.billablePeriod.star
ClaimResponse	ClaimResponse. 已创建
ClinicalImpression	ClinicalImpression. 日期
Communication	通信. 已发送
CommunicationRequest	CommunicationRequest。 occurrenceDateTime， CommunicationRequest.occurrencePeriod.start
合成	作文. 日期
条件	条件. 录制日期
同意	同意。dateTime
涵盖	Coverage.period.Start
CoverageEligibilityRequest	CoverageEligibilityRequest. 已创建
CoverageEligibilityResponse	CoverageEligibilityResponse. 已创建
DetectedIssue	DetectedIssue. 已识别

资源	资源元素
DeviceRequest	DeviceRequest.authoredon
DeviceUseStatement	DeviceUseStatement.recordedOn
DiagnosticReport	DiagnosticReport。有效
DocumentManifest	DocumentManifest. 已创建
DocumentReference	DocumentReference.context.period.start
遭遇	Encounter.period.Start
EnrollmentRequest	EnrollmentRequest. 已创建
EpisodeOfCare	EpisodeOfCare.period.start
ExplanationOfBenefit	ExplanationOfBenefit.billablePeriod.start
FamilyMemberHistory	NO_DATE
标记	flag.period.start
Goal	goal.statusDate
Group	NO_DATE
ImagingStudy	ImagingStudy。已开始

资源	资源元素
免疫接种	免疫接种。已记录
ImmunizationEvaluation	ImmunizationEvaluation. 日期
ImmunizationRecommendation	ImmunizationRecommendation. 日期
Invoice	发票日期
列表	List.date
MeasureReport	MeasureReport.period.start
媒体	Media. 已发布
MedicationAdministration	MedicationAdministration。有效
MedicationDispense	MedicationDispense. 准备好时
MedicationRequest	MedicationRequest.authoredon
MedicationStatement	MedicationStatement.dateAss
MolecularSequence	NO_DATE
NutritionOrder	NutritionOrder.dateTime

资源	资源元素
观察	观察。有效
病人	NO_DATE
人员	NO_DATE
过程	程序. 已执行
出处	出处。发生时期。开始，出处。 occurredDateTime
QuestionnaireResponse	QuestionnaireResponse. 创作
RelatedPerson	NO_DATE
RequestGroup	RequestGroup.authoredon
ResearchSubject	ResearchSubject. 句点
RiskAssessment	RiskAssessment。 occurrenceDateTime， RiskAssessment.occurrencePeriod.start
Schedule	Schedule.PlanningHorizon
ServiceRequest	ServiceRequest.authoredon
标本	标本。接收时间
SupplyDelivery	SupplyDelivery。 occurrenceDateTime， SupplyDelivery.occurrencePeriod.start，.occurrenceTiming.event SupplyDelivery
SupplyRequest	SupplyRequest.authoredon

资源	资源元素
VisionPrescription	VisionPrescription.dateWrit

使用检索 ValueSet 验证码 $expand

AWS HealthLake 现在支持您作为客户提取的 ValueSets 内容的$expand操作，使您能够检索这些 ValueSet 资源中包含的完整代码列表。当您需要执行以下操作时，此操作特别有用：

- 检索所有可能的代码以进行验证
- 在用户界面中显示可用选项
- 在特定的术语上下文中执行全面的代码查找

用法

可以使用 GET 和 POST 方法对 ValueSet 资源调用该$expand操作：

支持的操作

```
GET/POST [base]/ValueSet/[id]/$expand
GET [base]/ValueSet/$expand?url=http://example.com
POST [base]/ValueSet/$expand
```

支持的参数

HealthLake 支持 FHIR R4 $expand 参数的子集：

参数	Type	必需	说明
url	uri	否	待扩展的权威网址 ValueSet
id	id	否	ValueSet 要扩展的资源 ID（用于 GET 或 POST 操作）
filter	字符串	否	筛选代码扩展结果
count	整数	否	要返回的代码数量

参数	Type	必需	说明
offset	整数	否	返回之前要跳过的匹配代码的数量。过滤后仅适用于匹配的代码，不适用于原始代码中未经过滤的完整内容 ValueSet

示例

通过 ID 获取请求

```
GET [base]/ValueSet/example-valueset/$expand
```

使用过滤器通过 URL 获取请求

```
GET [base]/ValueSet/$expand?url=http://example.com/ValueSet/my-
valueset&filter=male&count=5
```

带参数的 POST 请求（按 ID）

```
POST [base]/ValueSet/example-valueset/$expand
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "count",
      "valueInteger": 10
    },
    {
      "name": "filter",
      "valueString": "admin"
    }
  ]
}
```

带参数的 POST 请求（通过 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
    }
  ]
}
```

示例响应

该操作返回一个 ValueSet 资源，其expansion元素包含扩展代码：

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

响应包括以下内容：

- expansion.total：扩展版中的代码总数 ValueSet
- expansion.contains：扩展代码数组及其系统、代码和显示值
- expansion.parameter：扩展请求中使用的参数

有关$expand操作规范的更多信息，请参阅 FHIR R4 文档 ValueSet $expand。

使用 FHI HealthLake R 导出数据 **$export**

您可以使用 FHIR $export 操作从 HealthLake 数据存储中批量导出数据。 HealthLake 支持 FHIR $export 使用POST和GET请求。要向发出导出请求POST，您必须拥有具有所需权限的 IAM 用户、群组或角色，在请求中指定$export，并在请求正文中包含所需的参数。

> **Note**
>
> 使用 FHIR 发出的所有 HealthLake 导出请求$export都将以ndjson格式返回并导出到 Amazon S3 存储桶，其中每个 Amazon S3 对象仅包含一个 FHIR 资源类型。
> 您可以根据 AWS 账户服务配额对导出请求进行排队。有关更多信息，请参阅 服务限额。

HealthLake 支持以下三种类型的批量导出端点请求。

HealthLake 批量**$export**类型

导出类型	说明	语法
系统	从 HealthLake FHIR 服务器导出所有数据。	`POST https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/$export`

所有患者	导出与所有患者相关的所有数据，包括与患者资源类型相关的资源类型。	`POST https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/$export`
		
		`GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/$export`

患者群体	导出与使用群组 ID 指定的一组患者相关的所有数据。	`POST https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Group/id/$export`
		
		`GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Group/id/$export`

开始前的准备工作

要使用适用 HealthLake的 FHIR REST API 提出导出请求，请满足以下要求。

- 您必须已设置具有必要权限的用户、组或角色才能发出导出请求。要了解更多信息，请参阅对请求进行授$export权。
- 您必须已创建服务角色来授予 HealthLake 访问要将数据导出到的 Amazon S3 存储桶的权限。服务角色还必须指定 HealthLake 为服务主体。有关设置权限的更多信息，请参阅为导出任务设置权限。

对请求进行授**$export**权

要使用 FHIR REST API 成功发出导出请求，请使用 IAM 或 OAuth2 .0 对您的用户、群组或角色进行授权。您还必须具有服务角色。

使用 IAM 对请求进行授权

当您提出`$export`请求时，用户、群组或角色的策略中必须包含 IAM 操作。有关更多信息，请参阅 为导出任务设置权限。

在 FHIR 上使用 SMART 授权请求 (2.0) OAuth

在支持 FHIR 的 SMART HealthLake 数据存储上`$export`发出请求时，必须分配相应的范围。有关更多信息，请参阅 SMART 对 FHIR 的资源范围适用于 HealthLake。

> **Note**
>
> `$export`带有`GET`请求的 FHIR 需要相同的身份验证方法或持有者令牌（对于 FHIR 上的 SMART 而言）才能请求导出和检索文件。使用 FHIR 导出的文件`$export``GET`可在 48 小时内下载。

提出**$export**请求

本节介绍使用 FHIR REST API 发出导出请求时必须采取的必要步骤。

为避免意外向您的 AWS 账户收费，我们建议您在不提供`$export`语法的情况下通过提出`POST`请求来测试您的请求。

要提出请求，您必须执行以下操作：

1. `$export`在`POST`请求网址中指定支持的终端节点。
2. 指定所需的标题参数。
3. 指定用于定义所需参数的请求正文。

步骤 1：**$export**在**POST**请求网址中指定支持的终端节点。

HealthLake 支持三种类型的批量导出端点请求。要发出批量导出请求，您必须在三个支持的终端节点之一上发出`POST`基于请求的请求。以下示例演示了在请求网址`$export`中指定何处。

- 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

您可以在POST请求字符串中使用以下支持的搜索参数。

支持的搜索参数

HealthLake 在批量导出请求中支持以下搜索修饰符。

以下示例包括特殊字符，在提交请求之前必须对其进行编码。

Name	必填？	说明	示例
_outputFormat	否	要生成的请求的批量数据文件的格式。可接受的值为application/fhir+ndjson 、application/ndjson 、ndjson。	
_type	否	要包含在导出任务中的以逗号分隔的 FHIR 资源类型字符串。我们建议将其包括在内，_type因为在导出所有资源时，这可能会影响成本。	&_type=MedicationStatement,Observation
_since	否	在日期时间戳当天或之后修改的资源类型。如果某个资源类型没有上次更新时间，	&_since=2024-05-09T00%3A00%3A00Z

Name	必填？	说明	示例
		则会将其包含在您的响应中。	
_until	否	在日期时间戳当天或之前修改的资源类型。与结合使用_since以定义特定的导出时间范围。如果某个资源类型没有上次更新时间，则它们将被排除在您的响应之外。	&_until=2024-12-31T23%3A59%3A59Z
_security	否	按meta.security 编码值筛选导出的资源。使用system|code 格式。当提供多个值时，资源必须匹配所有值（和语义）。使用system|（尾随管道，无代码）来匹配给定系统中的任何代码。	&_security=https://myorg.com/tenant%7Cclinic-A
_tag	否	按meta.tag编码值筛选导出的资源。使用与相同的system|code 格式和 AND 语义。_security 如果同时指定_security 了和，_tag则资源必须与两个过滤器匹配。	&_tag=https://myorg.com/dept%7Ccardiology

步骤 2：指定所需的标题参数

要使用 FHIR REST API 发出导出请求，必须指定以下标头参数。

- Content-Type：`application/fhir+json`
- 首选：`respond-async`

接下来，您必须在请求正文中指定所需的元素。

步骤 3：指定用于定义所需参数的请求正文。

导出请求还需要JSON格式化的正文。正文可以包含以下参数。

Key	必填？	说明	值
DataAccessRoleArn	是	HealthLake 服务角色的 ARN。使用的服务角色必须指定 HealthLake 为服务主体。	`arn:aws:iam:: 444455556666 :role/your-healthlake-service-role`
JobName	否	导出请求的名称。	**your-export-job-name**
S3Uri	是	OutputDataConfig 钥匙的一部分。将下载导出数据的目标存储桶的 S3 URI。	`s3://amzn-s3-demo-bucket/ EXPORT-JOB /`
KmsKeyId	是	OutputDataConfig 钥匙的一部分。用于保护 Amazon S3 存储桶的 AWS KMS 密钥的 ARN。	`arn:aws:kms: region-of-bucket:123456789012 :key/1234abcd-12ab-34cd-56ef-1234567890ab`

Example使用 FHIR REST API 发出的导出请求的正文

要使用 FHIR REST API 发出导出请求，必须指定正文，如下所示。

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

请求成功后，您将收到以下回复。

响应标头

```
content-location: https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
export/your-export-request-job-id
```

响应正文

```
{
  "datastoreId": "your-data-store-id",
  "jobStatus": "SUBMITTED",
  "jobId": "your-export-request-job-id"
}
```

管理您的导出请求

成功发出导出请求后，您可以使用描述当前导$export出请求的状态和取消当前导出请求$export来管理请求。

使用 REST API 取消导出请求时，您只需为提交取消请求之前导出的部分数据付费。

以下主题介绍如何获取当前导出请求的状态或取消当前导出请求。

取消导出请求

要取消导出请求，DELETE请提出请求并在请求网址中提供任务 ID。

```
DELETE https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/export/your-export-request-job-id
```

请求成功后，您将收到以下信息。

```
{
  "exportJobProperties": {
    "jobId": "your-original-export-request-job-id",
    "jobStatus": "CANCEL_SUBMITTED",
    "datastoreId": "your-data-store-id"
  }
}
```

当您的请求失败时，您会收到以下信息。

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "not-supported",
      "diagnostics": "Interaction not supported."
    }
  ]
}
```

描述导出请求

要获取导出请求的状态，GET请使用export和您的**export-request-job-id**。

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/export/your-export-request-id
```

JSON 响应将包含一个ExportJobProperties对象。它可能包含以下键:值对。

Name	必填？	说明	值
DataAccessRoleArn	否	HealthLake 服务角色的 ARN。使用的服务角色必须指定	arn:aws:iam:: **444455556666** :role/**your-**

Name	必填？	说明	值
		HealthLake 为服务主体。	**healthlake-service-role**
SubmitTime	否	提交导出任务的日期。	`Apr 21, 2023 5:58:02`
EndTime	否	导出任务完成的时间。	`Apr 21, 2023 6:00:08 PM`
JobName	否	导出请求的名称。	**your-export-job-name**
JobStatus	否		有效值为： `SUBMITTED | IN_PROGRESS | COMPLETED_WITH_ERRORS | COMPLETED | FAILED`
S3Uri	是	OutputDataConfig物体的一部分。将下载导出数据的目标存储桶的 Amazon S3 URI。	`s3://amzn-s3-demo-bucket/` **EXPORT-JOB /**
KmsKeyId	是	OutputDataConfig物体的一部分。用于保护 Amazon S3 存储桶的 AWS KMS 密钥的 ARN。	`arn:aws:kms:` **region-of-bucket:123456789012** `:key/`**1234abcd-12ab-34cd-56ef-1234567890ab**

Example: 使用 FHIR REST API 发出的描述导出请求的正文

成功后，您将收到以下 JSON 响应。

```
{
  "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",
  }
}
```

FHIR 的**$inquire**手术 HealthLake

该$inquire操作使您可以检查先前提交的先前授权请求的状态。该操作实施了《达芬奇事先授权支持 (PAS) 实施指南》，提供了一个基于 FHIR 的标准化工作流程来检索当前的授权决策。

工作原理

- 提交查询：您发送了一份 FHIR 捆绑包，其中包含您要查看的索赔和支持信息
- HealthLake 搜索： ClaimResponse 在您的数据存储中搜索相应的内容
- 检索：已检索到最新的授权状态
- 回应：您会立即收到包含当前授权状态（已排队、已批准、已拒绝等）的回复

> **Note**
>
> $inquire是一个只读操作，用于检索现有授权状态。它不会修改或更新您的数据存储中的任何资源。

API 端点

```
POST /datastore/{datastoreId}/r4/Claim/$inquire
Content-Type: application/fhir+json
```

请求结构

捆绑包要求

您的请求必须是 FHIR 捆绑包资源，其中包含以下内容：

- Bundle.type：必须是 `"collection"`
- Bundle.entry：必须恰好包含一个索赔资源，其中包含：
  - `use = "preauthorization"`
  - `status = "active"`
- 参考资源：声明中引用的所有资源都必须包含在捆绑包中

> **按示例查询**
>
> 您的输入 Bundle 中的资源可用作搜索模板。 HealthLake 使用提供的信息来查找相应的 ClaimResponse。

所需的资源

资源	基数	配置文件	说明
索赔	1	PAS 索赔查询	您要询问的事先授权
病人	1	PAS 受益患者	患者人口统计信息
组织（保险公司）	1	PAS 保险公司组织	保险公司
组织（提供商）	1	PAS 申请组织	提交申请的医疗保健提供者

重要的搜索条件

HealthLake 搜索 ClaimResponse 使用：

- 索赔中的@@ 患者推荐信
- 索赔中的@@ 保险公司推荐信
- 索赔中的@@ 提供者参考信息
- 索赔的@@ 创建日期（作为时间筛选器）

> **仅针对患者的查询**
>
> 所有查询都必须针对特定的患者。不允许在没有患者身份的情况下进行全系统查询。

示例请求

```
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"
      }
    }
  ]
}
```

响应格式

成功响应 (200 OK)

您将收到一个 PAS 查询回复包，其中包含：

- ClaimResponse当前授权状态；ClaimResponse如果符合搜索条件，则为多个
- 您请求中的所有原始资源（回显）
- 收集响应的时间戳

可能的 ClaimResponse 结果

结果	说明
queued	授权请求仍在等待审核
complete	已做出授权决定（检查是否已批准 disposition /拒绝）

结果	说明
error	处理过程中出错
partial	已授予部分授权


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

错误响应

400 错误请求

当请求格式无效或验证失败时返回。

```
{
    "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 未经授权

当身份验证凭证缺失或无效时返回。

```
{
    "resourceType": "OperationOutcome",
    "issue": [
        {
            "severity": "error",
            "code": "forbidden",
            "diagnostics": "Invalid authorization header"
        }
    ]
}
```

403 禁止访问

当经过身份验证的用户无权访问所请求的资源时返回。

```
{
    "resourceType": "OperationOutcome",
    "issue": [
        {
            "severity": "error",
            "code": "exception",
            "diagnostics": "Insufficient SMART scope permissions."
        }
    ]
}
```

400 当没有找到时

未找到与查询 ClaimResponse 相匹配的内容时返回。

```
{
```

```
  "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 不支持的媒体类型

当内容类型标头不是 application/fhir+json 时返回。

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "value",
    "diagnostics": "Incorrect MIME-type. Update request Content-Type header."
  }]
}
```

429 请求过多

超过速率限制时返回。

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "throttled",
    "diagnostics": "Rate limit exceeded. Please retry after some time."
  }]
}
```

验证规则

HealthLake 对您的询问进行全面验证：

捆绑包验证

- 必须符合 PAS 查询请求捆绑包配置文件

- Bundle.type必须是 "collection"
- 必须恰好包含一个 Claim 资源
- 所有引用的资源都必须包含在捆绑包中

索赔验证

- 必须符合 PAS 索赔查询资料
- Claim.use必须是 "preauthorization"
- Claim.status必须是 "active"
- 必填字段：patient、insurer、provider、created

资源验证

- 所有资源都必须符合各自的 PAS 查询资料
- 必须有所需的支持资源（患者、保险公司组织、提供者组织）
- 交叉引用必须有效且可在 Bundle 中解析

性能规格

指标	规范
资源数量限制	每个捆绑包 500 个资源
捆绑包大小限制	最大 5 MB

所需的权限

要使用该$inquire操作，请确保您的 IAM 角色具有：

- healthlake:InquirePreAuthClaim-调用该操作

在 FHIR 瞄准镜上使用 S

所需的最低范围：

- SMART v1：user/ClaimResponse.read

- SMART v2：`user/ClaimResponse.s`

重要的实施说明

搜索行为

当您提交查询时， ClaimResponse 使用以下方式进行 HealthLake 搜索：

- 输入索赔中的@@ 患者参考信息
- 输入索赔中的@@ 保险公司参考信息
- 来自输入索赔的@@ 提供者参考
- 根据输入的 Claim 创建日期（作为时间筛选器）

多个匹配项：如果多个 ClaimResponses 匹配您的搜索条件，则 HealthLake 返回所有匹配的结果。您应该使用最新的`ClaimResponse.created`时间戳来识别最新状态。

更新的索赔

如果您已向相同的先前授权（例如，Claim v1.1、v1.2、v1.3）提交了多个更新，则该`$inquire`操作将根据提供的搜索条件检索与最新版本 ClaimResponse 关联的更新。

只读操作

该`$inquire`操作：

- 是否检索现有的授权状态
- 是否会返回最新的 ClaimResponse
- 不修改或更新任何资源
- 不创建新资源
- 不会触发新的授权处理

工作流程示例

典型的事先授权查询工作流程

```
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"
```

相关 操作

- `Claim/$submit`-提交新的事先授权请求或更新现有的事先授权申请
- `Patient/$everything`-检索全面的患者数据以获取事先授权的上下文

使用检索概念细节 **$lookup**

AWS HealthLake 现在支持 CodeSystem 资源`$lookup`操作，使您能够通过提供代码等识别信息来检索有关代码系统中特定概念的详细信息。当您需要执行以下操作时，此操作特别有用：

- 检索有关特定医疗法规的详细信息
- 验证代码含义和属性
- 访问概念定义和关系
- 利用准确的术语数据支持临床决策

用法

可以使用 GET 和 POST 方法对 CodeSystem 资源调用该`$lookup`操作：

支持的操作

```
GET [base]/CodeSystem/$lookup?system=http://snomed.info/
sct&code=73211009&version=20230901
POST [base]/CodeSystem/$lookup
```

支持的参数

HealthLake 支持 FHIR R4 $lookup 参数的子集：

参数	Type	必需	说明
code	代码	是	你正在查找的概念代码（例如，康涅狄格州 SNOMED 中的 “716200000”）
system	uri	是	代码系统的权威网址（例如，“http://snomed.info/sct”）
version	字符串	否	代码系统的特定版本

示例

获取请求

```
GET [base]/CodeSystem/$lookup?system=http://snomed.info/
sct&code=71620000&version=2023-09
```

POST 请求

```
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"
    }
  ]
}
```

示例响应

该操作返回一个包含概念详细信息的参数资源：

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

响应参数

响应包括以下参数（如果有）：

参数	类型	说明
name	字符串	代码系统的名称
version	字符串	代码系统的版本
display	字符串	概念的显示名称
designation	BackboneElement	此概念的其他表述。
property	BackboneElement	概念的其他属性（定义、关系等）

行为

该$lookup操作：

1. 验证所需的参数（code和system）
2. 在存储在数据存储中的指定代码系统中搜索概念
3. 返回详细的概念信息，包括显示名称、名称和属性。
4. 提供参数时支持特定版本的version查找

5. 仅在显式存储在 HealthLake 数据存储中的代码系统上运行

错误处理

该操作处理以下错误情况：

- 400 错误请求：$lookup操作无效（请求不符合要求或缺少必填参数）
- 404 未找到：未找到代码系统或在指定的代码系统中找不到代码

警告

此版本不支持以下内容：

- $lookup通过调用外部术语服务器进行操作
- $lookup操作由 CodeSystems 管理 HealthLake 但未显式存储在数据存储中

有关$lookup操作规范的更多信息，请参阅 FHIR R4 文档 CodeSystem $lookup。

$member-add操作用于 HealthLake

FHIR $member-add 操作将成员（患者）添加到群组资源，特别是成员归因列表。此操作是《DaVinci 成员归因实施指南》的一部分，支持管理成员归因的协调流程。

操作终端节点

```
POST [base]/datastore/{datastoreId}/r4/Group/{groupId}/$member-add
Content-Type: application/json
```

参数

该操作接受具有以下参数组合的 FHIR 参数资源：

参数选项

您可以使用以下参数组合之一：

选项 1：会员 ID + 提供商 NPI

memberId + providerNpi

memberId + providerNpi + attributionPeriod

选项 2：患者推荐+提供者参考

`patientReference` + `providerReference`

`patientReference` + `providerReference` + `attributionPeriod`

参数详情

会员 ID（可选）

要添加到群组的成员的标识符。

类型：标识符

系统：患者识别系统

```
{
  "name": "memberId",
  "valueIdentifier": {
    "system": "http://example.org/patient-id",
    "value": "patient-new"
  }
}
```

ProviderNPI（可选）

归因提供商的国家提供商标识符 (NPI)。

类型：标识符

系统：http://terminology.hl7。 org/CodeSystem/NPI

```
{
  "name": "providerNpi",
  "valueIdentifier": {
    "system": "http://terminology.hl7.org/CodeSystem/NPI",
    "value": "1234567890"
  }
}
```

患者参考（可选）

直接引用待添加的患者资源。

类型：参考

```
{
  "name": "patientReference",
  "valueReference": {
    "reference": "Patient/patient-123"
  }
}
```

提供者引用（可选）

直接引用提供者资源。

类型：参考

```
{
  "name": "providerReference",
  "valueReference": {
    "reference": "Practitioner/provider-456"
  }
}
```

归因周期（可选）

将患者归因于提供者的时间段。

类型：时期

```
{
  "name": "attributionPeriod",
  "valuePeriod": {
    "start": "2024-07-15",
    "end": "2025-07-14"
  }
}
```

索取示例

使用会员 ID 和提供商 NPI

```
{
```

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

使用患者和提供者推荐信

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

响应格式

成功添加响应

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

错误响应

请求语法无效

HTTP 状态：400 错误请求

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

```
      "code": "invalid",
      "details": {
        "text": "Invalid parameter combination provided"
      }
    }
  ]
}
```

未找到资源

HTTP 状态：404 未找到

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "not-found",
      "details": {
        "text": "Resource not found."
      }
    }
  ]
}
```

版本冲突

HTTP 状态：409 冲突

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "conflict",
      "details": {
        "text": "Resource version conflict detected"
      }
    }
  ]
}
```

归因状态无效

HTTP 状态：422 无法处理的实体

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

商业规则

归因状态验证

只有当群组的归因状态为：时，才能执行该操作：

- draft
- open

当状态为时，不允许进行操作final。

防止重复会员

系统可防止根据以下各项的唯一组合添加重复成员：

- 成员标识符
- 付款人标识符
- 保险标识符

承保期验证

提供attributionPeriod时，它必须在成员的保险期限范围内。系统将：

- 搜索会员的承保范围资源
- 如果存在多个覆盖率，则使用最新的覆盖率（最高版本 ID）
- 确认归因期限未超过保险期限

参考文献验证

当为同一个资源（患者或提供者）同时提供身份证和参考资料时，系统会验证它们是否对应于相同的资源。

如果为同一资源（患者或提供者）同时提供 ID 和 reference.identifier 字段，则会引发错误。

身份验证和授权

该操作需要 FHIR 上的 SMART 授权，以便：

- 读取权限-用于验证患者、提供者和小组资源
- 搜索权限-按标识符查找资源
- 更新权限-修改群组资源

操作行为

资源更新

- 更新群组资源版本 ID
- 使用操作前的原始资源状态创建历史条目
- 使用以下命令将成员信息添加到 Group.member 数组：
  - entity.reference 中的患者参考资料
  - 期间内的归因周期
  - 扩展字段中的覆盖范围和提供者信息

验证步骤

- 参数验证-确保有效的参数组合
- 资源存在-验证患者、提供者和小组资源的存在
- 归因状态-确认群组状态允许修改
- 重复检查-防止添加现有成员
- 覆盖范围验证-确保归因期在覆盖范围内

限制

- 所有引用的资源必须存在于同一个数据存储中
- 操作仅适用于成员归因列表组资源

- 归因期限必须在覆盖范围内
- 无法修改处于 “最终” 状态的群组

`$member-match`操作为 HealthLake

AWS HealthLake 现在支持患者资源`$member-match`运营，使医疗保健组织能够使用人口统计和保险信息在不同的医疗保健系统中查找成员的唯一标识符。该操作对于实现CMS合规性以及在维护患者隐私的同时促进安全 payer-to-payer的数据交换至关重要。

当您需要执行以下操作时，此操作特别有用：

- 在组织之间实现安全的医疗保健数据交换
- 在不同系统中保持患者护理的连续性
- 支持 CMS 合规性要求
- 促进跨医疗保健网络的准确成员识别

用法

可以使用 POST 方法在患者资源上调用该`$member-match`操作：

```
POST [base]/Patient/$member-match
```

支持的参数

HealthLake 支持以下 FHIR `$member-match` 参数：

参数	Type	必需	默认值	说明
MemberPatient	病人	是	—	包含要匹配的成员的人口统计信息的患者资源
CoverageToMatch	涵盖	是	—	将用于与现有记录进行匹配的覆盖率资源
CoverageToLink	涵盖	否	—	在匹配过程中要链接的覆盖范围资源
同意	同意	否	—	用于授权目的的同意资源

示例

带参数的 POST 请求

```
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"
          }
```

```
        ]
      }
    }
  ]
}
```

示例响应

该操作返回一个包含匹配结果的参数资源：

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

响应参数

找到匹配项时，响应包含以下参数：

参数	类型	说明
MemberIdentifier	标识符	匹配成员的唯一标识符
MemberId	参考	参考患者资源
匹配算法	字符串	使用的匹配算法类型（EXACT_MATCH、STRONG_MATCH 或 DEMOGRAPHIC_MATCH）
比赛详情	字符串	有关匹配过程的详细信息
匹配字段	字符串	成功匹配的特定字段列表

匹配算法

$member-matchAPI 采用多层匹配方法来确保准确识别成员：

EXACT_MATCH

- 结合使用患者识别码和承保范围 SubscriberId
- 为成员匹配提供最高的置信度

强匹配

- 使用包含最少覆盖信息的患者标识符
- 当不符合完全匹配标准时，可信度很高

人口统计学匹配

- 依赖基本的人口统计信息
- 在无法进行基于标识符的匹配时使用

行为

该$member-match操作：

- 接受患者人口统计、承保范围详细信息和可选同意信息作为输入
- 返回可用于后续交互的唯一成员标识符
- 实施多层匹配（精确、强烈、人口统计），确保不同医疗保健系统中的成员身份准确识别
- 保存所提供的任何同意信息，以备将来授权之用
- 支持安全 payer-to-payer的数据交换，同时保护患者隐私
- 符合 CMS 对医疗保健数据交换的要求

Authorization

API 在 FHIR 上使用 SMART 授权协议，其所需范围如下：

- `system/Patient.read`
- `system/Coverage.read`
- `system/Organization.read`（有条件的）
- `system/Practitioner.read`（有条件的）
- `system/PractitionerRole.read`（有条件的）
- `system/Consent.write`（有条件的）

错误处理

该操作处理以下错误情况：

- `400 Bad Request`: `$member-match` 操作无效（请求不符合要求或缺少必填参数）
- `422 Unprocessable Entity`: 未找到匹配项或多个匹配项

`$member-remove`操作用于 HealthLake

该`$member-remove`操作允许您从中的 FHIR 成员归因列表（群组资源）中 AWS HealthLake移除成员。此操作是《 DaVinci 成员归因实施指南》的一部分，支持用于管理成员归因的协调流程。

先决条件

- AWS HealthLake FHIR 数据存储库
- 相应的 IAM HealthLake 操作权限

- 处于草稿或开放状态的成员归因列表（群组资源）

操作详情

端点

```
POST /Group/{id}/$member-remove
```

内容类型

```
application/fhir+json
```

参数

该操作接受带有以下可选参数的 FHIR 参数资源：

参数	基数	Type	说明
memberId	0.. 1	标识符	要移除的成员的企业标识符
ProviderNPI	0.. 1	标识符	归因提供商的 NPI
患者参考	0.. 1	参考	直接参考患者资源
提供者参考	0.. 1	参考	直接引用提供者资源（从 PractitionerRole业者或组织）
保险范围参考	0.. 1	参考	对覆盖范围资源的引用

支持的参数组合

支持以下参数组合：

- `memberIdonly`-删除指定成员的所有属性
- `memberId`+ `providerNpi`-移除特定成员-提供者组合的归因
- `patientReferenceonly`-移除指定患者的所有归因
- `patientReference`+ `providerReference`-移除特定患者-提供者组合的归因
- `patientReference`+ `providerReference` + `coverageReference`-删除基于患者、提供者和承保范围的特定归因

请求示例

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "patientReference",
      "valueReference": {
        "reference": "Patient/12345"
      }
    },
    {
      "name": "providerReference",
      "valueReference": {
        "reference": "Practitioner/67890"
      }
    }
  ]
}
```

响应

成功响应

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "result",
      "valueBoolean": true
    },
    {
      "name": "effectiveDate",
      "valueDate": "2024-06-30"
    },
    {
      "name": "status",
      "valueCode": "inactive"
    },
    {
      "name": "message",
      "valueString": "Member successfully removed from attribution list"
    }
```

```
  ]
}
```

行为

身份要求

该操作仅适用于状态为draft或的归因列表 open

带有final状态的列表将拒绝该操作，错误为 422

成员移除流程

草稿状态列表：成员被标记为不活跃 (inactive: true)，其changeType扩展名更新为 changed

打开状态列表：行为与草稿状态类似

最终状态列表：操作被拒绝

验证

对引用进行验证以确保它们存在于 HealthLake 数据存储中

如果为相同的资源类型同时提供了标识符和引用，则它们必须对应于相同的资源

参数组合根据支持的模式进行验证

错误处理

常见错误响应

未找到资源 (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"]
    }
  ]
}
```

归因列表最终状态 (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"]
    }
  ]
}
```

无效的操作 (400)

当参数组合无效或格式错误时返回。

找到多个匹配项 (412)

当提供的参数与归因列表中的多个成员匹配时返回。

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

```
      "severity": "error",
      "code": "processing",
      "diagnostics": "Multiple members found matching the criteria"
    }
  ]
}
```

最佳实践

- 使用特定参数：如果可能，请使用最具体的参数组合，以避免意外删除
- 检查列表状态：在尝试移除之前验证归因列表状态
- 优雅地处理错误：对所有可能的错误情况实施适当的错误处理
- 验证引用：在提出请求之前，请确保所有引用的资源都存在

使用移除患者隔间资源 $purge

AWS HealthLake 支持$purge手术，允许永久删除患者隔间内的所有资源。当您需要执行以下操作时，此操作特别有用：

- 删除与患者相关的所有数据
- 遵守患者数据删除请求
- 管理患者数据生命周期
- 执行全面的患者记录清理

用法

可以在患者资源上调用该$purge操作：

```
POST [base]/Patient/[ID]/$purge?deleteAuditEvent=true
```

参数

参数	Type	必需	默认值	说明
deleteAuditEvent	布尔值	否	false	如果为 true，则删除关联的审计事件

参数	Type	必需	默认值	说明
_since	字符串	否	数据存储创建时间	输入后，选择开始截止时间，根据资源的上次修改时间来查找资源。不能与开头或结尾一起使用
start	字符串	否	数据存储创建时间	输入后，根据资源的上次修改时间选择查找资源的截止时间。可以与 end 一起使用
end	字符串	否	Job 提交时间	输入后，选择结束截止时间，根据资源的上次修改时间来查找资源

示例

请求示例

```
POST [base]/Patient/example-patient/$purge?deleteAuditEvent=true
```

响应示例

```
{
  "resourceType": "OperationOutcome",
  "id": "purge-job",
  "issue": [
    {
      "severity": "information",
      "code": "informational",
      "diagnostics": "Purge job started successfully. Job ID:
 12345678-1234-1234-1234-123456789012"
    }
  ]
}
```

作业状态

要检查清除任务的状态，请执行以下操作：

```
GET [base]/$purge/[jobId]
```

该操作返回任务状态信息：

```
{
      "datastoreId": "36622996b1fcecb7e12ee2ee085308d3",
      "jobId": "3dd1c7a5b6c0ef8c110f566eb87e2ef9",
      "status": "COMPLETED",
      "submittedTime": "2025-10-31T18:43:21.822Z"
    }
```

行为

该$purge操作：

1. 异步处理以处理多个资源
2. 维护 ACID 事务以确保数据完整性
3. 提供任务状态跟踪，包括资源删除次数
4. 永久移除患者隔间内的所有资源
5. 包括删除活动的全面审核记录
6. 支持选择性删除审计事件

审核日志

$purge操作记录为 “开始” FHIRBulk DeleteJob 和 “描述” FHIRBulkDeleteJob ，其中包含详细的操作信息。

限制

- 已清除的资源不会出现在搜索响应中
- 正在清除的资源在处理过程中可能暂时无法访问
- 患者隔间的所有资源都将被永久移除

FHIR 的**$questionnaire-package**手术 HealthLake

该$questionnaire-package操作会检索一个包含FHIR调查问卷及其呈现和处理问卷所需的所有依赖项的综合包。该操作实现了《达芬奇文档模板和规则 (DTR) 实施指南》，从而实现了医疗工作流程中文档要求的动态表单呈现。

工作原理

- 请求：您发送标识所需调查问卷的参数，以及覆盖范围和订单上下文
- 检索： HealthLake 收集调查问卷和所有依赖项（ValueSets、CQL 库等）
- P@@ ac kage：所有资源都以标准格式捆绑在一起
- 回复：您会收到一个完整的软件包，可以进行渲染和数据收集

应用场景

- 事先授权文件：收集事先授权请求所需的临床信息
- 保险要求：收集满足付款人保险要求所需的文件
- Cl@@ inical Data Exchange：整理临床数据以提交给付款人
- 动态表单：使用预先填充的患者数据和条件逻辑呈现问卷

API 端点

```
POST /datastore/{datastoreId}/r4/Questionnaire/$questionnaire-package
Content-Type: application/fhir+json
```

请求参数

输入参数

请求正文必须包含带有以下参数的 FHIR 参数资源：

参数	Type	基数	说明
coverage	涵盖	1.. *（必填）	用于确定文件成员和覆盖范围的覆盖范围的覆盖资源
questionnaire	权威性的	0。。 *	要返回的特定问卷的权威网址（可能包括版本）
order	资源	0。。 *	订购资源（DeviceRequest、ServiceRequest、 MedicationRequest、遭遇、约会）以建立上下文

参数	Type	基数	说明
changedSince	dateTime	0.. 1	如果存在，则仅返回在此时间戳之后更改的资源

参数验证规则

必须提供以下至少一项（除必填项外coverage）：

- 一个或多个规questionnaire范 URLs
- 一个或多个order资源

有效的请求组合：

- coverage + questionnaire
- coverage + order
- coverage + questionnaire + order

示例请求

```
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"
    }
  ]
}
```

响应格式

成功响应 (200 OK)

该操作返回包含一个或多个 Pack age Bund le 的 FHIR 参数资源。每个 Package 套装包括：

参赛类型	基数	说明
问卷	1	要填写的问卷
QuestionnaireRespo nse	0.. 1	预填充或部分完成的回复（如果适用）
Library	0。。*	包含预填充和条件逻辑的 CQL 库
ValueSet	0。。*	已扩展 ValueSets （适用于扩展范围小于 40 的答案选项）

Example响应示例

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "PackageBundle",
      "resource": {
        "resourceType": "Bundle",
        "id": "questionnaire-package-example",
        "meta": {
          "profile": ["http://hl7.org/fhir/us/davinci-dtr/StructureDefinition/DTR-
QPackageBundle"]
        },
        "type": "collection",
        "timestamp": "2024-03-15T10:30:00Z",
        "entry": [
          {
            "fullUrl": "http://example.org/fhir/Questionnaire/home-oxygen-therapy",
            "resource": {
              "resourceType": "Questionnaire",
              "id": "home-oxygen-therapy",
              "url": "http://example.org/fhir/Questionnaire/home-oxygen-therapy",
              "version": "2.0",
```

```
        "status": "active",
        "title": "Home Oxygen Therapy Documentation",
        "item": [
          {
            "linkId": "1",
            "text": "Patient diagnosis",
            "type": "choice",
            "answerValueSet": "http://example.org/fhir/ValueSet/oxygen-diagnoses"

          }
        ]
      }
    },
    {
      "fullUrl": "http://example.org/fhir/Library/oxygen-prepopulation",
      "resource": {
        "resourceType": "Library",
        "id": "oxygen-prepopulation",
        "url": "http://example.org/fhir/Library/oxygen-prepopulation",
        "version": "1.0",
        "type": {
          "coding": [{
            "system": "http://terminology.hl7.org/CodeSystem/library-type",
            "code": "logic-library"
          }]
        },
        "content": [{
          "contentType": "text/cql",
          "data": "bGlicmFyeSBPeHlnZW5QcmVwb3B1bGF0aW9u..."
        }]
      }
    },
    {
      "fullUrl": "http://example.org/fhir/ValueSet/oxygen-diagnoses",
      "resource": {
        "resourceType": "ValueSet",
        "id": "oxygen-diagnoses",
        "url": "http://example.org/fhir/ValueSet/oxygen-diagnoses",
        "status": "active",
        "expansion": {
          "timestamp": "2024-03-15T10:30:00Z",
          "contains": [
            {
              "system": "http://hl7.org/fhir/sid/icd-10",
```

```
                    "code": "J44.0",
                    "display": "COPD with acute lower respiratory infection"
                  },
                  {
                    "system": "http://hl7.org/fhir/sid/icd-10",
                    "code": "J96.01",
                    "display": "Acute respiratory failure with hypoxia"
                  }
                ]
              }
            }
          },
          {
            "fullUrl": "http://example.org/fhir/QuestionnaireResponse/example-
prepopulated",
            "resource": {
              "resourceType": "QuestionnaireResponse",
              "id": "example-prepopulated",
              "questionnaire": "http://example.org/fhir/Questionnaire/home-oxygen-
therapy|2.0",
              "status": "in-progress",
              "subject": {
                "reference": "Patient/example-patient"
              },
              "basedOn": [{
                "reference": "ServiceRequest/example-service-request"
              }],
              "item": [
                {
                  "linkId": "1",
                  "text": "Patient diagnosis",
                  "answer": [{
                    "valueCoding": {
                      "system": "http://hl7.org/fhir/sid/icd-10",
                      "code": "J44.0",
                      "display": "COPD with acute lower respiratory infection"
                    }
                  }]
                }
              ]
            }
          }
        ]
      }
```

```
    },
    {
      "name": "Outcome",
      "resource": {
        "resourceType": "OperationOutcome",
        "issue": [{
          "severity": "information",
          "code": "informational",
          "details": {
            "text": "Successfully retrieved questionnaire package"
          }
        }]
      }
    }
  ]
}
```

操作工作流程

如何 HealthLake 处理您的请求

拨打电话时$questionnaire-package， HealthLake 执行以下步骤：

1. 识别患者和付款人：从您的coverage参数中提取患者和保险组织。
2. 找到合适的问卷：
   - 带**questionnaire**参数：使用您提供的权威网址
   - 带**order**参数：匹配订单代码 (CPT/HCPCS/LOINC) 和付款人以找到相应的问卷
3. 收集依赖关系：自动检索呈现调查问卷所需的所有内容：
   - CQL 库-填充前问题和条件问题的逻辑
   - ValueSets-答案选项（如果选项小于 40 个，则自动扩展）
   - QuestionnaireResponse-任何现有正在处理或已完成的回复
4. P@@ ackage 把所有东西放在一起：
   - 捆绑所有资源（每种资源仅包含一次）
   - 按changedSince时间戳过滤（如果提供）
   - 在Outcome缺少任何资源时添加警告

结果：一个完整、独立的软件包可以渲染了。

错误响应

400 错误请求

请求验证失败时返回。

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "required",
    "details": {
      "text": "At least one of 'questionnaire' or 'order' must be provided along with
 'coverage'"
    }
  }]
}
```

424 失败的依赖关系

当无法检索依赖资源时返回。

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "warning",
    "code": "not-found",
    "details": {
      "text": "Referenced Library 'http://example.org/fhir/Library/missing-library'
 could not be retrieved"
    }
  }]
}
```

401 未经授权

当身份验证凭证缺失或无效时返回。

403 禁止访问

当经过身份验证的用户无权访问所请求的资源时返回。

406 不可接受

当无法提供请求的内容类型时返回。

409 冲突

存在版本或并发冲突时返回。

410 Gone

当请求的资源已被永久删除时返回。

429 请求过多

超过速率限制时返回。

500 内部服务器错误

服务器发生意外错误时返回。

501 未实施

当请求的操作尚未实现时返回。

验证规则

输入验证

- `coverage`参数为必填项（1.. * 基数）
- `order`必须至少提供其中一个`questionnaire`或一个
- 所有覆盖范围资源都必须是有效的 FHIR 资源
- 所有订单资源都必须是有效的 FHIR 资源
- 规范的格式 URLs 必须正确
- `changedSince`必须是有效的 ISO 8601 日期时间

QuestionnaireResponse 验证

- `status`必须合适 (`in-progress`、`completed`、`amended`)
- 结构必须与引用的问卷相匹配
- `basedOn`必须引用有效的订单资源
- `subject`必须引用有效的患者资源

资源重复数据删除

- 每个资源在捆绑包中仅出现一次
- 例外：可能同时包含同一资源的不同版本
- 资源由其规范 URL 和版本标识

性能规格

指标	规范
资源数量限制	每个捆绑包 500 个资源
捆绑包大小限制	最大 5 MB

所需的权限

要使用该`$questionnaire-package`操作，请确保您的 IAM 角色具有：

- `healthlake:QuestionnairePackage`-调用该操作
- `healthlake:ReadResource`-检索问卷和相关资源
- `healthlake:SearchWithPost`-搜索 QuestionnaireResponse 和相关资源

FHIR 瞄准镜上的智能

所需的最低范围：

- SMART v1：`user/Questionnaire.read user/Library.read user/ValueSet.read user/QuestionnaireResponse.read`
- SMART v2：`user/Questionnaire.rs user/Library.rs user/ValueSet.rs user/QuestionnaireResponse.rs`

重要的实施说明

资源检索策略

问卷识别优先级：

- 规范网址（如果提供了questionnaire参数）-最高优先级
- 订单分析（如果提供了order参数）：
  - 将订单代码（CPT、HCPCS、LOINC）与付款人的医疗保单相匹配
  - 使用保险付款人筛选特定于付款人的问卷
  - 考虑原因代码以获取更多上下文

依赖关系解决方案

CQL 库：

- 通过问卷资源上的cqf-library扩展程序检索
- 使用类型递归获取依赖库 Library.relatedArtifact depends-on
- 所有库依赖项都包含在软件包中

ValueSets:

- 如果包含的概念少于 40 个，则会自动展开
- 包括 ValueSets 较大的不带扩展功能
- ValueSets 包括问卷和图书馆资源中引用的内容

QuestionnaireResponse 种群前

在以下情况下，该操作可能会返回 QuestionnaireResponse 带有预填充数据的a：

- 已找到现有正在处理或已完成的响应
- 关联库中的 CQL 逻辑可以从患者记录中提取数据
- 回复与相关的订单和覆盖范围相关

搜索标准 QuestionnaireResponse：

搜索参数	FHIR 路径	说明
based-on	QuestionnaireResponse.basedOn	链接到 ServiceRequest 或 CarePlan
patient	QuestionnaireResponse.subject	作为受试者的患者
questionnaire	QuestionnaireResponse.questionnaire	正在回答的问卷

已更改的资源筛选

提供changedSince参数时：

- 仅包含在指定时间戳之后修改的资源
- 如果未更改任何资源，则返回200 OK一个空包
- 对于增量更新和缓存策略很有用
- 时间戳比较使用资源meta.lastUpdated字段

多个套餐捆绑包

在以下情况下，该操作可能会返回多个 Package Bundle：

- 通过 canonical 申请了多份问卷 URLs
- 多个订单需要不同的问卷
- 同一份问卷的不同版本适用

每个 Package Bundle 都是独立的，具有所有必要的依赖关系。

常见使用案例

用例 1：事先授权文档

场景：提供者需要收集家庭氧气疗法的文件，事先获得授权。

```
{
```

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

结果：返回包含氧气治疗问卷的包裹，其中预先填充了电子病历中的患者生命体征和诊断代码。

用例 2：检索特定的调查问卷版本

场景：提供者需要特定版本的问卷以保证合规性。

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "coverage",
      "resource": { /* Coverage resource */ }
    },
    {
      "name": "questionnaire",
      "valueCanonical": "http://example.org/fhir/Questionnaire/dme-request|3.1.0"
    }
  ]
}
```

结果：精确返回包含所有依赖项的 DME 请求调查问卷的 3.1.0 版本。

用例 3：检查更新

场景：提供者想要检查自上次检索以来是否有任何问卷资源已更新。

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

结果：仅返回 2024 年 3 月 1 日之后修改过的资源，如果没有任何更改，则返回空包。

用例 4：多个订单

场景：提供商提交多个服务请求，可能需要不同的调查表。

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "coverage",
      "resource": { /* Coverage resource */ }
    },
    {
      "name": "order",
      "resource": { /* ServiceRequest for imaging */ }
    },
    {
      "name": "order",
      "resource": { /* ServiceRequest for DME */ }
    }
```

```
  ]
}
```

结果：返回多个 Package Bundle，每个适用的问卷对应一个。

与其他达芬奇集成 IGs

保险需求发现 (CRD)

工作流程集成：

- 提供者在其 EHR 中订购服务
- CRD 挂钩开火，正在检查覆盖要求
- 付款人回复表示需要文件
- 提供者`$questionnaire-package`致电检索文档表单
- 提供者填写问卷
- 文件通过 PAS 提交或 CDex

事先授权支持 (PAS)

工作流程集成：

- `$questionnaire-package`用于检索文档要求
- 填写所需的临床数据 QuestionnaireResponse
- 使用`Claim/$submit`填写完毕后提交事先授权 QuestionnaireResponse
- 使用检查状态 `Claim/$inquire`

临床数据交换 (CDex)

工作流程集成：

- 付款人要求为索赔提供其他文件
- 提供者`$questionnaire-package`用于检索结构化数据收集表单
- 提供商完成 QuestionnaireResponse
- 文件通过 CDex 附件工作流程提交给付款人

故障排除指南

问题：未返回问卷

可能的原因：

- 权威网址与数据存储中的任何调查问卷都不匹配
- 订单代码未映射到付款人医疗政策中的任何问卷
- 保险付款人没有相关的问卷

解决方案：

- 验证规范 URL 是否正确且问卷是否存在
- 检查订单代码 (CPT/HCPCS) 的指定是否正确
- 确认付款人组织已配置调查表

问题：Package 中缺少依赖关系

可能的原因：

- 引用的库或者数据存储中 ValueSet 不存在
- 图书馆参考文献已损坏或不正确
- ValueSet 扩展失败

解决方案：

- 检查`Outcome`参数中是否有关于资源缺失的警告
- 验证您的数据存储中是否存在所有引用的资源
- 确保正确 ValueSet URLs 且可解决

问题：带有 changedSince 的空包

可能的原因：

- 这是预期行为-自指定时间戳以来未修改任何资源

解决方案：

- 使用软件包的缓存版本
- 移除changedSince参数以检索完整软件包

问题： QuestionnaireResponse 未预先填充

可能的原因：

- 未 QuestionnaireResponse 找到现有的
- CQL 库逻辑无法提取所需数据
- 患者数据缺失或不完整

解决方案：

- 这可能是预料之中的——并非所有问卷都有人口前的逻辑
- 检查数据存储中是否存在患者数据
- 查看 CQL 库逻辑以了解数据提取要求

最佳实践

1. 将 Canonical 与版本一起 URLs 使用

在申请特定问卷时，请务必指定版本号：

```
{
  "name": "questionnaire",
  "valueCanonical": "http://example.org/fhir/Questionnaire/dme|2.1.0"
}
```

原因：确保问卷更新时的一致性并防止意外更改。

2. 为了提高性能，杠杆率已改变

对于经常访问的调查表，请使用changedSince以最大限度地减少数据传输：

```
{
  "name": "changedSince",
  "valueDateTime": "2024-03-10T15:30:00Z"
}
```

原因：通过仅检索更新的资源来减少延迟和带宽使用量。

3. 包括完整的承保范围信息

提供全面的覆盖范围详情，以确保正确选择问卷：

```
{
  "name": "coverage",
  "resource": {
    "resourceType": "Coverage",
    "beneficiary": { "reference": "Patient/123" },
    "payor": [{ "reference": "Organization/payer-abc" }],
    "class": [{ /* Group/plan information */ }]
  }
}
```

原因：帮助 HealthLake 确定付款人特定的调查问卷和要求。

相关 操作

- `Claim/$submit`-提交事先授权申请并附上完整的文件
- `Claim/$inquire`-查看已提交的先前授权的状态
- `ValueSet/$expand`-展开 ValueSets 答案选择

FHIR 的**`$submit`**手术 HealthLake

该`$submit`操作使您能够以电子方式向付款人提交事先授权请求以供批准。该操作实施了《达芬奇事先授权支持（PAS）实施指南》，为事先授权提交提供了基于FHIR的标准化工作流程。

工作原理

- 提交：您发送一份 FHIR 捆绑包，其中包含您先前的授权申请和辅助临床数据
- 验证：根据 PAS 要求 HealthLake 验证提交的内容
- 保留：所有资源都存储在您的 HealthLake 数据存储中
- 回复：您会立即收到状态为 “已排队” 的回复

- 流程：授权决策由付款人异步处理

API 端点

```
POST /datastore/{datastoreId}/r4/Claim/$submit
Content-Type: application/fhir+json
```

请求结构

捆绑包要求

您的请求必须是 FHIR 捆绑包资源，其中包含以下内容：

- Bundle.type：必须是 `"collection"`
- b@@ undle.entr y：必须恰好包含一个索赔资源 `use = "preauthorization"`
- 参考资源：声明中引用的所有资源都必须包含在捆绑包中

所需的资源

资源	基数	配置文件	说明
索赔	1	PAS 索赔	事先授权请求
病人	1	PAS 患者	患者人口统计信息
组织（保险公司）	1	PAS 保险公司	保险公司
组织（提供商）	1	PAS 请求者	医疗保健提供者提交申请
覆盖范围	1 或更多	PAS 覆盖范围	保险范围详情

可选资源

资源	基数	配置文件	说明
从业者	0 或更多	PAS从业人员	医疗保健从业人员

资源	基数	配置文件	说明
PractitionerRole	0 或更多	PAS PractitionerRole	从业者角色
ServiceRequest	0 或更多	PAS ServiceRequest	要求的医疗服务
DeviceRequest	0 或更多	PAS DeviceRequest	要求的医疗设备
MedicationRequest	0 或更多	PAS MedicationRequest	要求的药物
DocumentReference	0 或更多	PAS DocumentReference	支持临床文档

示例请求

```
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"
    }
  }]
}
```

响应格式

成功响应 (200 OK)

您将收到一个 PAS 响应包，其中包含：

- ClaimResponse使用outcome: "queued"和 status: "active"
- 您请求中的所有原始资源
- 确认收货的时间戳


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

错误响应

400 错误请求

当请求格式无效或格式错误时返回。

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "invalid",
    "diagnostics": "The provided payload was invalid and could not be parsed
 correctly."
  }]
}
```

412 前提条件失败

当已提交相同的事先授权请求（检测到重复提交）时返回。

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "processing",
    "diagnostics": "PreAuth Claim already exists"
  }]
}
```

幂等性

该$submit操作是等效的。多次提交同一个请求不会创建重复的事先授权请求。相反，你会收到一个 412 错误，指示你使用$inquire来检查原始提交的状态。

422 无法处理的实体

当 FHIR 验证失败时返回。

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "required",
    "diagnostics": "Bundle contains more than one preauthorization claim"
  }]
```

```
}
```

验证规则

HealthLake 对您提交的内容进行全面验证：

捆绑包验证

- 必须符合 PAS 请求捆绑包配置文件
- `Bundle.type`必须是 `"collection"`
- 可以包含多个 Claim 资源
- 但是，必须仅包含一个可使用预授权的 Claim 资源
  - 而且这个 Claim 资源必须是捆绑包中的第一个条目
- 所有引用的资源都必须包含在捆绑包中

索赔验证

- 必须符合 PAS 索赔档案
- `Claim.use`必须是 `"preauthorization"`
- 必填字段：`patient`、`insurer`、`provider`、`created`、`priority`
- 企业标识符必须存在且有效

资源验证

- 所有资源都必须符合各自的 PAS 配置文件
- 必须有所需的支持资源（患者、承保范围、组织）
- 交叉引用必须有效且可在 Bundle 中解析

性能规格

指标	规范
捆绑包大小限制	最大 5 MB
资源数量限制	每个捆绑包 500 个资源

所需的权限

要使用该$submit操作，可以在 FHIR 上使用 AWS Sigv4 或 SMART：

- 确保您的 IAM 角色具有：healthlake:SubmitPreAuthClaim-调用操作

在 FHIR 瞄准镜上使用 S

所需的最低范围：

- SMART v1：user/Claim.write & <all_resourceTypes_in_Bundle>.write
- SMART v2：user/Claim.c & <all_resourceTypes_in_Bundle>.c or system/*.*

重要的实施说明

资源持久性

- 所有 Bundle 条目都作为单独的 FHIR 资源保留在您的数据存储中
- 客户提供的信息在提供时 IDs 会被保留
- 维护版本历史记录以供审计
- 重复检测可防止资源冲突

处理行为

- 每份有效的提交结果恰好是一份 ClaimResponse ，其中包含"queued"结果
- 无效的提交会返回 400 或 422 状态码以及详细的错误信息
- 系统错误会返回相应的 5xx 状态码
- 所有成功提交的内容都会返回 200 状态，且状态为待处理 ClaimResponse

捆绑包要求

- Bundle.entry.fullUrl值必须是 REST URLs 或"urn:uuid:[guid]"格式
- 所有提交内容都 GUIDs 必须是唯一的（相同的资源实例除外）
- 引用的资源必须存在于捆绑包中或者可以解析

相关 操作

- Claim/$inquire-查询已提交的事先授权请求的状态
- Patient/$everything-检索全面的患者数据以了解事先授权的情况

使用验证 FHIR 资源 **$validate**

AWS HealthLake 现在支持 FHIR 资源的$validate操作，使您无需执行任何存储操作即可根据 FHIR 规范验证资源并检查其是否符合指定的配置文件或基本资源定义。当您需要执行以下操作时，此操作特别有用：

- 验证 FHIR CMS 合规性要求
- 在生产环境中使用资源之前对其进行测试
- 在用户编辑临床数据时提供实时验证反馈
- 减少要创建和更新的无效数据提交 APIs

用法

可以使用 POST 方法在 FHIR 资源上调用该$validate操作：

支持的操作

```
POST [base]/[type]/[id]/$validate
POST [base]/[type]/$validate
```

支持的负载

参数资源

HealthLake 支持以下 FHIR $validate 参数：

参数	Type	必需	说明
resource	资源	是	要验证的资源
profile	权威性的	否	要验证的个人资料的规范网址

参数	Type	必需	说明
mode	代码	否	验证模式：create，或 update

带有查询参数的直接资源

参数	Type	必需	说明
profile	权威性的	否	要验证的个人资料的规范网址
mode	代码	否	验证模式：create，或 update

示例

带有 ID 和参数负载的 POST 资源请求

```
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"
    }
  ]
}
```

资源类型和参数负载的 POST 请求

```
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"
    }
  ]
```

```
}
```

POST 请求带有 ID 和直接资源负载的资源

```
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"
}
```

资源类型和直接资源负载的 POST 请求

```
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"
}
```

示例响应

该操作返回一个包含验证结果的 OperationOutcome 资源：

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "information",
      "code": "informational",
      "diagnostics": "Validation successful"
    }
  ]
}
```

包含验证错误的示例响应

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "required",
      "details": {
        "text": "Missing required element"
      },
      "diagnostics": "Patient.identifier is required by the US Core Patient profile",
      "location": [
        "Patient.identifier"
      ]
    },
    {
      "severity": "warning",
      "code": "code-invalid",
      "details": {
        "text": "Invalid code value"
      },
      "diagnostics": "The provided gender code is not from the required value set",
      "location": [
        "Patient.gender"
      ]
    }
  ]
}
```

行为

该$validate操作：

1. 根据 FHIR 规范和基础资源定义验证资源
2. 提供profile参数时检查与指定配置文件的一致性
3. 根据指定的模式（create或）进行update验证
4. 返回详细的验证结果，包括错误、警告和信息性消息
5. 不执行任何存储操作-仅限验证
6. 无论是否发现验证问题，当可以执行验证时，都返回 HTTP 200 OK

验证模式

- 创建：像创建资源一样验证资源（新资源）
- 更新：像更新一样验证资源（现有资源）

错误处理

该操作返回：

- 200 OK：验证已成功执行（无论验证结果如何）
- 400 错误请求：请求格式或参数无效
- 404 未找到：未找到资源类型或配置文件

有关$validate操作规范的更多信息，请参阅 FHIR R4 资源$validate文档。

的合规性参考 AWS HealthLake

AWS HealthLake 提供的功能旨在帮助您跟踪和报告 API 使用情况，以符合 CMS（医疗保险和医疗补助服务中心）互操作性要求。这些功能使您能够按照 CMS 规定的类别对 API 交易进行分类，并自动捕获使用量指标以进行合规性报告。

> **了解您的合规责任**
>
> 仅靠使用 AWS HealthLake 及其 CMS 互操作性端点不足以实现 CMS 合规性。您有责任做到以下几点：

- 根据您的特定用例和监管义务，将您的 API 工作流程正确映射到相应的 CMS 类别端点
- 实施符合 CMS 要求的适当身份验证和授权控制
- 确保您的 FHIR 资源和数据交换符合适用的 CMS 法规和实施指南
- 配置和监控 CloudWatch 指标以支持您的合规性报告需求
- 了解哪些 CMS 规则适用于您的组织并实施适当的技术和运营控制

AWS HealthLake 提供了支持您的合规工作的基础架构和工具，但您必须根据具体的监管要求适当地使用这些功能。仅仅通过这些端点路由 API 调用并不能自动使您的应用程序符合 CMS 法规。

主题

CMS 合规性功能

AWS HealthLake 提供的功能可帮助您满足 CMS（医疗保险和医疗补助服务中心）互操作性和合规性要求。这些功能使您能够按内容管理系统类别跟踪 API 使用情况，然后报告使用情况指标以实现合规性。

主题

CMS 互操作端点

概述

HealthLake 提供了四个 CMS 互操作性端点，它们对应于 CMS 规定的 API 类别。 HealthLake 数据存储的底层基本 URL 不会改变。这些端点只是提供了一种对您的 API 调用进行分类和跟踪的方法，以便进行 CMS 报告。

用途

这些互操作性端点的主要目的是使客户能够：

- 按内容管理系统类别@@ 轻松跟踪 API 交易

- 自动报告 CMS 合规性的使用量指标
- 只需最少的更改即可@@ 维护现有的 FHIR 工作流程

无论您使用互操作性端点还是标准的 FHIR 端点，所有 API 调用的功能都是一样的，唯一的区别是如何按指标对交易进行分类。 CloudWatch

支持的 CMS 互操作端点

内容管理系统类别	互操作性端点	示例用法
患者通道	/patientaccess/v2/r4	baseURL/patientaccess/v2/r4/Patient/123
提供商访问权限	/provideraccess/v2/r4	baseURL/provideraccess/v2/r4/Observation?patient=123
付款人对付款人	/payertopayerdx/v2/r4	baseURL/payertopayerdx/v2/r4/Practitioner/456
先前的身份验证服务	/priorauthservice/v2/r4	baseURL/priorauthservice/v2/r4/ExplanationOfBenefit?patient=789

互操作性端点的工作原理

标准 HealthLake API 调用：

```
baseURL/resourceType/[id]
baseURL/resourceType?[parameters]
```

使用 CMS 互操作性端点：

```
baseURL/interoperability-endpoint/resourceType/[id]
baseURL/interoperability-endpoint/resourceType?[parameters]
```

只需在您的基本 URL 和资源类型之间插入互操作性端点路径即可。互操作性端点路径之后的所有内容都与您当前的 API 调用完全相同。

用法示例

示例 1：患者访问 API

当前的 API 调用（仍然有效）：

```
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"
```

使用患者访问互操作性端点（用于 CMS 跟踪）：

```
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"
```

要点：

- 基本网址保持不变：`https://healthlake.us-east-1.amazonaws.com/datastore/abc123`
- 已插入互操作性端点：`/patientaccess/v2/r4`
- 资源路径未更改：`/Patient/123`
- 两个呼叫都返回相同的响应
- 互操作性端点调用将在下方`URIType=patient-access`自动跟踪 CloudWatch
- POST、PUT、PATCH、DELETE 操作的工作原理相同。
- 请求正文保持不变。

端点翻译参考

互操作性端点	翻译为	内容管理系统类别
`baseURL/patientaccess/v2/r4/Patient`	`baseURL/r4/Patient`	患者通道

互操作性端点	翻译为	内容管理系统类别
baseURL/providera ccess/v2/r4/Observ ation	baseURL/r4/Observation	提供商访问权限
baseURL/payertopayerdx/ v2/r4/Practitioner/456	baseURL/r4/Practit ioner/456	付款人对付款人 Data Exchange
baseURL/priorauths ervice/v2/r4/Expla nationOfBenefit?pa tient=789	baseURL/r4/Explana tionOfBenefit?pati ent=789	事先授权

重要提示

- 无功能差异：互操作性端点和标准 FHIR 端点返回相同的响应并支持相同的操作
- 基本 URL 未更改：您的 HealthLake 数据存储端点保持不变
- 简单集成：在基本 URL 和资源类型之间插入互操作性端点路径
- 自动跟踪： CloudWatch 指标根据使用的互操作性端点自动对呼叫进行分类
- 向后兼容：没有互操作端点的现有 API 调用可以继续正常工作

增强了 CMS 合规性 CloudWatch 指标

概述

当您使用 CMS 互操作性端点时， HealthLake 会自动生成带有额外维度的增强 CloudWatch 指标，以支持 CMS 报告要求。这些指标按呼叫者身份、应用程序和特定于 CMS 的 URI 类型跟踪 API 使用情况，无需进行任何其他配置。

工作原理

当你使用互操作性端点进行 API 调用时：

```
# 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.)
```

不需要额外的代码或配置。只需使用互操作性端点，即可自动捕获增强的指标。

> **Note**
>
> 对于非 Smart on FHIR 数据存储，仍会捕获URIType维度，从而允许您按内容管理系统类别跟踪 API 使用情况。Sub和ClientId维度仅在使用包含这些声明的不记名令牌进行 FHIR 身份验证的 SMART 时可用。

新的指标维度

使用互操作性端点时DatastoreId，DatastoreType除了现有维度（、、Operation）之外，还会自动添加以下维度：

维度	说明	示例值	来源
URIType	CMS 合规性类别	patient-access，provider-access，payer-to-payer，prior-authorization	根据互操作性端点路径自动确定
Sub	来电者身份	用户/实体标识符	摘自不记名代币索赔 sub
ClientId	应用程序标识符	portal_app，ehr_system	摘自不记名代币索赔 client_id

可用指标

现在，使用互操作性终端节点时，所有现有 HealthLake 指标都包含其他维度：

- CallCount-API 调用总数

- 延迟-API 响应时间（以毫秒为单位）
- UserErrors-4xx 个客户端错误的计数
- SystemErrors-5xx 服务器错误计数
- Throttles-受限制的请求数
- SuccessfulRequests-成功的 API 调用次数

在中查询指标 CloudWatch

CloudWatch 见解查询示例

按应用程序查询所有 Patient Access API 调用：

```
SELECT SUM(CallCount)
FROM "AWS/HealthLake"
WHERE DatastoreId = '75c1cf9b0d71cd38fec8f7fb317c4c1a'
    AND URIType = 'patient-access'
GROUP BY ClientId
```

的 Support 参考资料 AWS HealthLake

以下支持参考材料可用于 AWS HealthLake。

> **Note**
>
> 所有原生 HealthLake 操作和数据类型均在单独的参考文献中进行了描述。有关更多信息，请参阅 AWS HealthLake API 参考。

主题

- AWS HealthLake 终端节点和配额
- 合成预加载的数据类型 HealthLake
- AWS HealthLake 示例项目
- 故障排除 AWS HealthLake
- HealthLake 与 AWS SDK 一起使用

AWS HealthLake 终端节点和配额

以下主题包含有关 AWS HealthLake 服务终端节点和配额的信息。

主题

服务端点

服务端点是指定作为某一 Web 服务入口点的主机和端口的 URL。每个 Web 服务请求都包含一个端点。大多数 AWS 服务都为特定区域提供终端节点，以实现更快的连接。下表列出了的服务终端节点 AWS HealthLake。

区域名称	区域	端点	协议
美国东部（俄亥俄州）	us-east-2	healthlake.us-east-2.amazonaws.com healthlake-fips.us-east-2.amazonaws.com	HTTPS HTTPS
美国东部（弗吉尼亚州北部）	us-east-1	healthlake.us-east-1.amazonaws.com healthlake-fips.us-east-1.amazonaws.com	HTTPS HTTPS
美国西部（俄勒冈州）	us-west-2	healthlake.us-west-2.amazonaws.com healthlake-fips.us-west-2.amazonaws.com	HTTPS HTTPS
亚太地区（孟买）	ap-south-1	healthlake.ap-south-1.amazonaws.com	HTTPS
亚太地区（悉尼）	ap-southeast-2	healthlake.ap-southeast-2.amazonaws.com	HTTPS
加拿大（中部）	ca-central-1	healthlake.ca-central-1.amazonaws.com	HTTPS

区域名称	区域	端点	协议	
欧洲地区（爱尔兰）	eu-west-1	healthlake.eu-west-1.amazonaws.com	HTTPS	
欧洲地区（伦敦）	eu-west-2	healthlake.eu-west-2.amazonaws.com	HTTPS	

服务限额

服务配额定义为 AWS 账户中资源、操作和项目的最大值。

> **Note**
>
> 对于可调限额，您可以使用服务限额控制台请求增加限额。有关更多信息，请参阅服务限额用户指南 中的请求增加限额。
> Sync Write API 速率随有效载荷大小成比例增加，每 1KB 增量消耗额外容量（例如，4KB 的负载使用 4 倍的写入容量）。将可选标`x-amz-fhir-history-consistency-level`头设置为每种资源的写入容量消耗量增加一`strong`倍。
> 捆绑包中的资源遵循基于 1 KB 有效载荷大小的标准 read/write 限制。与批处理类型相比，捆绑包类型事务消耗的写入容量是批处理容量的两倍，这意味着批处理捆绑包每秒可以处理的资源量是事务捆绑包的两倍。

下表列出了的默认配额 AWS HealthLake。

Name	默认值	可调整	说明
每个账户的有效订阅数量	每个受支持的区域：100 个	是	每个账户的最大活跃订阅资源数量。
每个数据存储的有效订阅资源数量	每个受支持的区域：50 个	是	每个数据存储的最大活跃订阅资源数。

Name	默认值	可调整	说明
SubscriptionTopic 每个账户的活跃人数	每个受支持的区域：100 个	是	每个账户的最大活跃 SubscriptionTopic 资源数。
每个数据存储的活动 SubscriptionTopic 资源数量	每个受支持的区域：50 个	是	每个数据存储的最大活动 SubscriptionTopic 资源数。
医疗记录中的字符数	每个受支持的区域：1 万个	否	DocumentReference 资源类型（POST/PUT 请求）中单个医疗记录中的最大字符数。
并发 StartFHIRExportJob 任务数	每个受支持的区域：1 个	否	最大并发 StartFHIR ExportJob 任务数。
并发 StartFHIRImportJob 任务数	每个受支持的区域：1 个	否	最大并发 StartFHIR ImportJob 任务数。
每个账户的数据存储数量	每个受支持的区域：20 个	是	每个账户的默认最大活动数据存储数量。
a 中的文件数 StartFHIRImportJob	每个受支持的区域：100 万个	是	中的最大文件数 StartFHIRImportJob。
每个捆绑包的资源数量	每个受支持的区域：500 个	否	捆绑包请求中允许的最大资源数量。
每个账户使用 DELETE 的 CancelFHIRExportJob 请求率	每个受支持的区域：1 个	否	每个账户每秒可以使用 DELETE 发出的最大 CancelFHIRExportJob 请求数。

Name	默认值	可调整	说明
每个账户的 CreateFHIRDatastore 请求速率	每个受支持的区域：1 个	否	每账户每分钟可发出的最大 CreateFHIRDatastore 请求数量。
每个账户的 DeleteFHIRDatastore 请求速率	每个受支持的区域：1 个	否	每个账户每分钟可发出的最大 DeleteFHIRDatastore 请求数量。
每个账户的 DescribeFHIRBulkDeleteJob 请求率	每个受支持的区域：10 个	是	每个账户每秒可以发出的最大 DescribeFHIRBulkDeleteJob 请求数。
每个数据存储的 DescribeFHIRBulkDeleteJob 请求率	每个受支持的区域：10 个	是	每个数据存储每秒可以发出的最大 DescribeFHIRBulkDeleteJob 请求数。
每个账户的 DescribeFHIRDatastore 请求速率	每个受支持的区域：10 个	否	每个账户每秒可发出的最大 DescribeFHIRDatastore 请求数量。
每个账户的 DescribeFHIRExportJob 请求率	每个受支持的区域：10 个	否	每个账户每秒可以发出的最大 DescribeFHIRExportJob 请求数。
每个账户使用 GET 的 DescribeFHIRExportJob 请求率	每个受支持的区域：10 个	否	每个账户每秒可以使用 GET 发出的最大 DescribeFHIRExportJob 请求数。
每个账户的 DescribeFHIRImportJob 请求率	每个受支持的区域：10 个	否	每个账户每秒可以发出的最大 DescribeFHIRImportJob 请求数。

Name	默认值	可调整	说明
每个账户的 Discovery 请求速率	每个受支持的区域：10 个	否	每个账户每分钟可发出的最大 Discovery 请求数量。
每个账户的 GET 请求速率	每个受支持的区域：6,000 个	是	每个账户每秒可发出的最大 GET 请求数量。
每个数据存储的 GET 请求速率	每个受支持的区域：3,000 个	是	每个数据存储每秒可发出的最大 GET 请求数量。8/21/2023 之前创建的数据存储将限制为每秒 100 个请求。
每个账户的 GetCapabilities 请求率	每个受支持的区域：10 个	否	每个账户每秒可以发出的最大 GetCapabilities 请求数。
每个数据存储的 GetExportedFile 请求率	每个受支持的区域：10 个	否	每个数据存储每秒可以发出的最大 GetExportedFile 请求数。
每个账户的 ListFHIRDatastores 请求速率	每个受支持的区域：10 个	否	每个账户每秒可发出的最大 ListFHIRDatastores 请求数量。
每个账户的 ListFHIRExportJobs 请求率	每个受支持的区域：10 个	否	每个账户每秒可以发出的最大 ListFHIRExportJobs 请求数。
每个账户的 ListFHIRImportJobs 请求率	每个受支持的区域：10 个	否	每个账户每秒可以发出的最大 ListFHIRImportJobs 请求数。

Name	默认值	可调整	说明
每个账户的 ListTagsforResource 请求率	每个受支持的区域：10 个	否	每个账户每秒可以发出的最大 ListTagsforResource 请求数。
每个账户的 SearchEverything 请求率	每个受支持的区域：10 个	是	每个账户每秒可以发出的最大 SearchEverything 请求数。
每个数据存储的 SearchEverything 请求率	每个受支持的区域：10 个	是	每个数据存储每秒可以发出的最大 SearchEverything 请求数。
每个账户的 StartFHIRBulkDeleteJob 请求率	每个受支持的区域：10 个	是	每个账户每秒可以发出的最大 StartFHIRBulkDeleteJob 请求数。
每个数据存储的 StartFHIRBulkDeleteJob 请求率	每个受支持的区域：10 个	是	每个数据存储每秒可以发出的最大 StartFHIRBulkDeleteJob 请求数。
每个账户的 StartFHIRBulkMemberMatchJob 请求率	每个受支持的区域：10 个	是	每个账户每秒可以发出的最大 StartFHIRBulkMemberMatchJob 请求数。
每个数据存储的 StartFHIRBulkMemberMatchJob 请求率	每个受支持的区域：10 个	是	每个数据存储每秒可以发出的最大 StartFHIRBulkMemberMatchJob 请求数。
每个账户的 StartFHIRExportJob 请求率	每个受支持的区域：1 个	否	每个账户每秒可以发出的最大 StartFHIRExportJob 请求数。

Name	默认值	可调整	说明
每个账户使用 GET 的 StartFHIR ExportJob 请求率	每个受支持的区域：1 个	否	每个账户每秒可以使用 GET 发出的最大 StartFHIRExportJob 请求数。
每个账户使用 POST 的 StartFHIR ExportJob 请求率	每个受支持的区域：1 个	否	每个账户每秒可以使用 POST 发出的最大 StartFHIRExportJob 请求数。
每个账户的 StartFHIRImportJob 请求率	每个受支持的区域：25 个	否	每个账户每秒可以发出的最大 StartFHIRImportJob 请求数。
每个账户的 TagResource 请求率	每个受支持的区域：10 个	否	每个账户每秒可以发出的最大 TagResource 请求数。
每个账户的 UntagResource 请求率	每个受支持的区域：10 个	否	每个账户每秒可以发出的最大 UntagResource 请求数。
每个账户的 ValidateResource 请求率	每个支持的区域：2000 个	是	每个账户每秒可以发出的最大 ValidateResource 请求数。 8/21/2023 之前创建的数据存储将限制为每秒 300 个请求。
每个数据存储的 ValidateResource 请求率	每个受支持的区域：1,000 个	是	每个数据存储每秒可以发出的最大 ValidateResource 请求数。 8/21/2023 之前创建的数据存储将限制为每秒 300 个请求。

Name	默认值	可调整	说明
每个账户的 WRITE 请求速率	每个受支持的区域：6000 个	是	每个账户每秒可发出的最大 CREATE|UPDATE|DELETE 请求数量。
每个数据存储的 WRITE 请求速率	每个受支持的区域：3000 个	是	每个数据存储每秒可发出的最大 CREATE|UPDATE|DELETE 请求数量。 8/21/2023 之前创建的数据存储将限制为每秒 300 个请求。
每个账户使用 GET 的搜索请求速率	每个受支持的区域：200 个	是	每个账户每秒可发出的使用 GET 的最大搜索请求数量。
每个数据存储使用 GET 的搜索请求速率	每个受支持的区域：100 个	是	每个数据存储每秒可发出的使用 GET 的最大搜索请求数量。
每个账户使用 POST 的搜索请求速率	每个受支持的区域：200 个	是	每个账户每秒可发出的使用 POST 的最大搜索请求数量。
每个数据存储使用 POST 的搜索请求速率	每个受支持的区域：100 个	是	每个数据存储每秒可发出的使用 POST 的最大搜索请求数量。
单个导入文件的大小	每个受支持的区域：50 GB	否	包含在中的单个文件的最大大小（以 GB 为单位）StartFHIRImportJob。
每个数据存储的排队异步捆绑包事务总数	每个受支持的区域：500 个	是	在任何给定时间，每个数据存储区排队的异步捆绑包事务的最大数量。

Name	默认值	可调整	说明
每个数据存储的排队批量导出作业总数	每个受支持的区域：25 个	是	每个数据存储在任何给定时间的最大排队批量导出作业数量。
每个数据存储的排队批量导入作业总数	每个受支持的区域：25 个	是	每个数据存储在任何给定时间的最大排队批量导入作业数量。
总导入任务大小	每个支持的区域：5,000 千兆字节	是	导入任务中包含的所有文件的最大大小（以 GB 为单位）。

合成预加载的数据类型 HealthLake

HealthLake 仅支持 SYNTHEA 作为预加载的数据类型。Synth ea 是一款用于对`Patient`病史进行建模的合成患者生成器。它托管在开源 Git 存储库中， HealthLake 允许生成符合 FHIR R4 的资源，`Bundle`这样用户就可以在不使用实际患者数据的情况下测试模型。

预加载 HealthLake 的数据存储中提供以下资源类型。有关使用 Synthea HealthLake 数据预加载数据存储的更多信息，请参阅。创建 HealthLake 数据存储

> **Note**
>
> 要查看 HealthLake支持的 FHIR R4 资源的完整列表，请参见。FHIR R4 支持的资源类型适用于 HealthLake

合成支持的 FHIR 资源类型 HealthLake

AllergyIntolerance	位置
CarePlan	MedicationAdministration
CareTeam	MedicationRequest

声明	观察
条件	Organization（组织）
设备	病人
DiagnosticReport	从业者
遭遇	PractitionerRole
ExplanationofBenefit	过程
ImagingStudy	出处
免疫接种	

AWS HealthLake 示例项目

为了进一步进行分析，您可以与其他 AWS 服务 HealthLake 一起使用，如以下博客文章示例所示。

HealthLake 集成分析

- 人口健康应用程序 AWS HealthLake — 第 1 部分：使用 Amazon Quick 进行分析和监控。
- 使用 Amazon A SageMaker I 和 AWS HealthLake 标准化数据构建预测性疾病模型。
- 使用 A AWS I 服务构建认知搜索和健康知识图表。

HealthLake 事件监控

- 亚马逊 EventBridge 与... 集成 AWS HealthLake。

故障排除 AWS HealthLake

以下主题针对您在使用、或 HealthLake 控制台时可能遇到的错误和问题提供了疑难解答建议。 AWS CLI AWS SDKs如果您发现本节中未列出的问题，请使用本页右侧边栏上的 “提供反馈” 按钮进行举报。

主题

- 数据存储操作

- 导入操作
- FHIR APIs
- 自然语言处理集成
- SQL 集成

数据存储操作

问题：当我尝试创建 HealthLake 数据存储时，我收到以下错误：

```
AccessDeniedException: Insufficient Lake Formation permission(s): Required Database on
 Catalog
```

2022 年 11 月 14 日， HealthLake 更新了创建新数据存储所需的 IAM 权限。有关更多信息，请参阅配置要使用的 IAM 用户或角色 HealthLake （ IAM 管理员 ）。

问题：使用创建 HealthLake 数据存储时 AWS SDKs，数据存储创建状态会返回异常或未知状态。

如果您的调用`DescribeFHIRDatastore`或 `ListFHIRDatastores` API 调用返回异常或未知数据存储状态，请将您的 AWS SDK 更新到最新版本。

导入操作

问题： HealthLake 如果我的数据不是 FHIR R4 格式，我还能使用吗？

只有 FHIR R4 格式的数据可以导入到 HealthLake 数据存储中。有关可以帮助将现有健康数据转换为 FHIR R4 格式的合作伙伴列表，请参阅AWS HealthLake 合作伙伴。

问题：为什么我的 FHIR 导入任务失败了？

成功的导入任务将生成一个`.ndjson`格式为结果（输出日志）的文件夹，但是，单个记录可能无法导入。发生这种情况时，将生成第二个`FAILURE`文件夹，其中包含未能导入的记录清单。有关更多信息，请参阅 使用导入 FHIR 数据 AWS HealthLake。

要分析导入任务失败的原因，请使用 `DescribeFHIRImportJob` API 进行分析 JobProperties。建议采取以下措施：

- 如果状态为`FAILED`且存在消息，则失败与作业参数有关，例如输入数据大小或输入文件数量超出 HealthLake 配额。

- 如果导入任务状态为COMPLETED_WITH_ERRORS，请查看清单文件manifest.json，以了解哪些文件未成功导入。
- 如果导入任务状态为FAILED，但消息不存在，请前往任务输出位置访问清单文件manifest.json。

对于每个输入文件，都有失败输出文件，其中包含任何未能导入的资源的输入文件名。响应包含与输入数据位置相对应的行号 (lineID)、FHIR 响应对象 (UpdateResourceResponse) 和响应的状态码 (StatusCode)。

示例输出文件可能类似于以下内容：

```
{"lineId":3, UpdateResourceResponse:{"jsonBlob":
{"resourceType":"OperationOutcome","issue":
[{"severity":"error","code":"processing","diagnostics":"1 validation error detected:
 Value 'Patient123' at 'resourceType' failed to satisfy constraint: Member must satisfy
 regular expression pattern: [A-Za-z]{1,256}"}]}, "statusCode":400}
{"lineId":5, UpdateResourceResponse:{"jsonBlob":
{"resourceType":"OperationOutcome","issue":
[{"severity":"error","code":"processing","diagnostics":"This property must be an
 simple value, not a com.google.gson.JsonArray","location":["/EffectEvidenceSynthesis/
name"]},{"severity":"error","code":"processing","diagnostics":"Unrecognised
 property '@telecom'","location":["/EffectEvidenceSynthesis"]},
{"severity":"error","code":"processing","diagnostics":"Unrecognised
 property '@gender'","location":["/EffectEvidenceSynthesis"]},
{"severity":"error","code":"processing","diagnostics":"Unrecognised
 property '@birthDate'","location":["/EffectEvidenceSynthesis"]},
{"severity":"error","code":"processing","diagnostics":"Unrecognised
 property '@address'","location":["/EffectEvidenceSynthesis"]},
{"severity":"error","code":"processing","diagnostics":"Unrecognised
 property '@maritalStatus'","location":["/EffectEvidenceSynthesis"]},
{"severity":"error","code":"processing","diagnostics":"Unrecognised
 property '@multipleBirthBoolean'","location":["/EffectEvidenceSynthesis"]},
{"severity":"error","code":"processing","diagnostics":"Unrecognised
 property '@communication'","location":["/EffectEvidenceSynthesis"]},
{"severity":"warning","code":"processing","diagnostics":"Name should be usable as an
 identifier for the module by machine processing applications such as code generation
 [name.matches('[A-Z]([A-Za-z0-9_]){0,254}')]","location":["EffectEvidenceSynthesis"]},
{"severity":"error","code":"processing","diagnostics":"Profile http://hl7.org/fhir/
StructureDefinition/EffectEvidenceSynthesis, Element 'EffectEvidenceSynthesis.status':
 minimum required = 1, but only found 0","location":["EffectEvidenceSynthesis"]},
{"severity":"error","code":"processing","diagnostics":"Profile
 http://hl7.org/fhir/StructureDefinition/EffectEvidenceSynthesis,
```

```
 Element 'EffectEvidenceSynthesis.population': minimum required
 = 1, but only found 0","location":["EffectEvidenceSynthesis"]},
{"severity":"error","code":"processing","diagnostics":"Profile
 http://hl7.org/fhir/StructureDefinition/EffectEvidenceSynthesis,
 Element 'EffectEvidenceSynthesis.exposure': minimum required =
 1, but only found 0","location":["EffectEvidenceSynthesis"]},
{"severity":"error","code":"processing","diagnostics":"Profile http://
hl7.org/fhir/StructureDefinition/EffectEvidenceSynthesis, Element
 'EffectEvidenceSynthesis.exposureAlternative': minimum required
 = 1, but only found 0","location":["EffectEvidenceSynthesis"]},
{"severity":"error","code":"processing","diagnostics":"Profile http://hl7.org/fhir/
StructureDefinition/EffectEvidenceSynthesis, Element 'EffectEvidenceSynthesis.outcome':
 minimum required = 1, but only found 0","location":["EffectEvidenceSynthesis"]},
{"severity":"information","code":"processing","diagnostics":"Unknown
 extension http://synthetichealth.github.io/synthea/disability-adjusted-
life-years","location":["EffectEvidenceSynthesis.extension[3]"]},
{"severity":"information","code":"processing","diagnostics":"Unknown extension
 http://synthetichealth.github.io/synthea/quality-adjusted-life-years","location":
["EffectEvidenceSynthesis.extension[4]"]}]}, "statusCode":400}
{"lineId":7, UpdateResourceResponse:{"jsonBlob":
{"resourceType":"OperationOutcome","issue":
[{"severity":"error","code":"processing","diagnostics":"2 validation errors detected:
 Value at 'resourceId' failed to satisfy constraint: Member must satisfy regular
 expression pattern: [A-Za-z0-9-.]{1,64}; Value at 'resourceId' failed to satisfy
 constraint: Member must have length greater than or equal to 1"}]}, "statusCode":400}
{"lineId":9, UpdateResourceResponse:{"jsonBlob":
{"resourceType":"OperationOutcome","issue":
[{"severity":"error","code":"processing","diagnostics":"Missing required id field in
 resource json"}]}, "statusCode":400}
{"lineId":15, UpdateResourceResponse:{"jsonBlob":
{"resourceType":"OperationOutcome","issue":
[{"severity":"error","code":"processing","diagnostics":"Invalid JSON found in input
 file"}]}, "statusCode":400}
```

上面的示例显示，输入文件中相应输入行的第 3、4、7、9、15 行出现故障。对于每行，解释如下：

- 在第 3 行，响应解释了输入文件第 3 行中`resourceType`提供的内容无效。
- 在第 5 行，响应说明输入文件的第 5 行存在 FHIR 验证错误。
- 在第 7 行，响应解释说，作为输入`resourceId`提供的存在验证问题。
- 在第 9 行，响应说明输入文件必须包含有效的资源 ID。
- 在第 15 行，输入文件的响应是该文件不是有效的 JSON 格式。

FHIR APIs

问题：如何对 FHIR RESTful APIs 实施授权？

确定数据存储授权策略要使用的。

要使用创建 Sigv4 授权 适用于 Python (Boto3) 的 AWS SDK，请创建一个类似于以下示例的脚本。

```
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())
```

问题：为什么在使用客户托管的 KMS 密钥加密的数据存储中使用 FHIR RESTful APIs 时会收到*AccessDenied*错误？

用户或角色需要拥有客户托管密钥和 IAM 策略的权限才能访问数据存储。用户必须拥有使用客户托管密钥所需的 IAM 权限。如果用户撤销或取消了授予使用客户托管的 KMS 密钥 HealthLake 权限的授权，则 HealthLake 会返回AccessDenied错误。

HealthLake 必须拥有访问客户数据、加密导入到数据存储的新 FHIR 资源以及在请求时解密 FHIR 资源的权限。有关更多信息，请参阅AWS KMS 权限疑难解答。

问题： HealthLake 使用 10MB 文档的 FH *POST* IR API 操作返回了错误。*413 Request Entity Too Large*

AWS HealthLake 同步创建和更新 API 限制为 5MB，以避免延迟和超时增加。您可以使用批量导入 API 使用`Binary`资源类型提取最大 164MB 的大型文档。

自然语言处理集成

问题：如何开启 HealthLake集成的自然语言处理功能？

自 2022 年 11 月 14 日起， HealthLake 数据存储的默认行为发生了变化。

当前数据存储：所有当前 HealthLake 的数据存储都将停止对 base64 `DocumentReference` 编码的资源使用自然语言处理 (NLP)。这意味着不会使用 NLP 分析新资源，也不会根据资源类型中的文本生成新`DocumentReference`资源。`DocumentReference`对于现有`DocumentReference`资源，通过 NLP 生成的数据和资源将保留，但不会在 2023 年 2 月 20 日之后更新。

新数据存储：2023 年 2 月 20 日之后创建 HealthLake 的数据存储将不会对 base64 `DocumentReference` 编码的资源执行自然语言处理 (NLP)。

要启用 HealthLake NLP 集成，请使用AWS Support Center Console创建支持案例。要创建您的案例，请登录您的 AWS 账户，然后选择创建案例。要了解有关创建案例和案例管理的更多信息，请参阅支持 用户指南中的创建支持案例和案例管理。

问题：>如何找到无法通过集成 NLP 处理的*`DocumentReference`*资源？

如果`DocumentReference`资源无效，则 HealthLake 提供表示验证错误的扩展名，而不是在综合医疗 NLP 输出中提供。要查找在 NLP 处理过程中导致验证错误的**`DocumentReference`**资源，您可以使用带有搜索键cm-decoration-status和搜索值 VALIDATION_ERROR HealthLake 的 FHIR **`search`** 函数。此搜索将列出导致验证错误的所有`DocumentReference`资源，以及描述错误性质的错误消息。那些存在验证错误的`DocumentReference`资源中扩展字段的结构将类似于以下示例。

```
"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**
>
> 如果 NLP 装饰创建的嵌套对象超过 10,000 个，也会出现 A VALIDATION_ERROR。发生这种情况时，必须先将文档拆分成较小的文档，然后再进行处理。

SQL 集成

问题：为什么我在添加新的数据湖管理员*permissions error: lakeformation:PutDataLakeSettings*时会获得 Lake Formation？

如果您的 IAM 用户或角色包含AWSLakeFormationDataAdmin AWS 托管策略，则无法添加新的数据湖管理员。您将收到一条包含以下内容的错误：

```
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
```

需要使用 AWS 托管策略AdministratorAccess才能将 IAM 用户或角色添加为 AWS Lake Formation 数据湖管理员。如果您的 IAM 用户或角色也包含AWSLakeFormationDataAdmin该操作，则操作将失败。AWSLakeFormationDataAdmin AWS 托管策略包含对 La AWS ke Formation API 操作的明确拒绝PutDataLakeSetting。即使管理员拥有 AWS 使用AdministratorAccess托管策略的完全访问权限，也可能受到该AWSLakeFormationDataAdmin策略的限制。

问题：如何迁移现有 HealthLake 数据存储以使用 Amazon Athen a SQL 集成？

HealthLake 2022 年 11 月 14 日之前创建的数据存储可以正常运行，但无法在 Athena 中使用 SQL 进行查询。要使用 Athena 查询先前存在的数据存储，必须先将其迁移到新的数据存储。

将您的 HealthLake 数据迁移到新的数据存储

1. 创建新的数据存储。
2. 将数据从先前存在的存储桶导出到 Amazon S3 存储桶。
3. 将数据从 Amazon S3 存储桶导入到新的数据存储中。

> **Note**
>
> 将数据导出到 Amazon S3 存储桶需要支付额外费用。额外费用取决于您导出的数据的大小。

问题：为 SQL 集成创建新的 HealthLake 数据存储时，数据存储状态未改变*Creating*。

如果您尝试创建新的 HealthLake 数据存储，但您的数据存储状态未从 “正在创建” 发生变化，则需要更新 Athena 才能使用。 AWS Glue Data Catalog有关更多信息，请参阅亚马逊 Athena AWS 用户指南step-by-step中的升级到 Glue 数据目录。

成功升级后 AWS Glue Data Catalog，您可以创建 HealthLake 数据存储。

要移除旧 HealthLake 的数据存储，请使用创建支持案例AWS Support Center Console。要创建您的案例，请登录您的 AWS 账户，然后选择创建案例。要了解更多信息，请参阅支持 用户指南中的创建支持案例和案例管理。

问题：将数据导入新数据存储后，Athena 控制台无法正常工作 HealthLake

将数据导入新的 HealthLake 数据存储后，这些数据可能无法立即使用。这是为了留出时间将数据提取到 Apache Iceberg 表中。请稍后再试。

问题：如何将 Athena 中的搜索结果连接到其他服务？ AWS

AWS 与其他服务共享来自 Athena 的搜索结果时，当您将搜索结果`json_extract[1]`用作 SQL 搜索查询的一部分时，可能会出现问题。要修复此问题，必须更新到`CATVAR`。

在尝试创建保存结果、表（静态）或视图（动态）时，您可能会遇到此问题。

HealthLake 与 AWS SDK 一起使用

AWS 软件开发套件 (SDKs) 可用于许多流行的编程语言。每个软件开发工具包都提供 API、代码示例和文档，使开发人员能够更轻松地以其首选语言构建应用程序。

SDK 文档	代码示例
适用于 C++ 的 AWS SDK	适用于 C++ 的 AWS SDK 代码示例
AWS CLI	AWS CLI 代码示例
适用于 Go 的 AWS SDK	适用于 Go 的 AWS SDK 代码示例
适用于 Java 的 AWS SDK	适用于 Java 的 AWS SDK 代码示例
适用于 JavaScript 的 AWS SDK	适用于 JavaScript 的 AWS SDK 代码示例
适用于 Kotlin 的 AWS SDK	适用于 Kotlin 的 AWS SDK 代码示例
适用于 .NET 的 AWS SDK	适用于 .NET 的 AWS SDK 代码示例
适用于 PHP 的 AWS SDK	适用于 PHP 的 AWS SDK 代码示例
AWS Tools for PowerShell	AWS Tools for PowerShell 代码示例
适用于 Python (Boto3) 的 AWS SDK	适用于 Python (Boto3) 的 AWS SDK 代码示例
适用于 Ruby 的 AWS SDK	适用于 Ruby 的 AWS SDK 代码示例
适用于 Rust 的 AWS SDK	适用于 Rust 的 AWS SDK 代码示例
适用于 SAP ABAP 的 AWS SDK	适用于 SAP ABAP 的 AWS SDK 代码示例
适用于 Swift 的 AWS SDK	适用于 Swift 的 AWS SDK 代码示例

示例可用性

找不到所需的内容？ 通过使用此页面底部的提供反馈链接请求代码示例。

AWS HealthLake 发布

下表显示了的功能和更新的发布时间 AWS HealthLake。有关版本的更多信息，请参阅链接的主题。

变更	说明	日期
$export 和 $ 的安全和标签过滤 davinci-data-export	$export和$davinci-data-export 操作现在支持_security 和_tag查询参数，用于按meta.security 和meta.tag编码值筛选导出的资源。这支持使用标准 FH system|code IR 格式进行多租户导出和精细访问控制。	2026年4月30日
$ bulk-member-match 操作	AWS HealthLake 现在支持异步处理多个成员匹配请求的$bulk-member-match 操作。此操作使医疗保健组织能够在单个批量请求中使用人口统计和保险信息，有效地匹配不同医疗保健系统中的数百个成员的唯一标识符。 • 每个请求最多可处理 500 个成员，每个数据存储最多可处理 5 个并发操作 • 结果分为 MatchedMembers NonMatchedMembers、和 ConsentConstrained Members 组 • 与$davinci-data-export end-to-end批量数据工作流程集成	2026年4月1日

	有关更多信息，请参阅 the section called “$bulk-Member-match”。	
异步捆绑交易	AWS HealthLake 现在支持异步Bundle类型transaction ，允许您提交包含最多 500 个资源的交易。 HealthLake 将事务排队等候处理，并立即返回轮询 URL 以检查状态和检索结果。有关更多信息，请参阅异步捆绑交易。	2026年3月24日
包操作中的 Patch Support	HealthLake 现在支持 Bundle Patch，为 Transaction FHIRPath 和 Batch 捆绑包以及 FHIRPath 补丁 API 启用了 JSON 补丁。此功能允许客户直接在 Bundle 操作中修补资源，而无需更换全部资源，从而减少了有效负载大小，简化了集成工作流程，并加快了数据摄取速度。	2026 年 3 月 20 日
DaVinci PDex $ 的导出类型 davinci-data-export	该$davinci-data-export 操作现在支持 “提供者访问权限” 和 “成员访问权限” 的 PDex 导出类型 APIs。 Payer-to-Payer • 基于配置文件的资源包含逻辑 ExplanationOfBenefit • 使用_includeEOB2xWoFinancial 参数进行财务数据转换 • 对临床和索赔数据进行为期 5 年的时间筛选	2026 年 3 月 20 日

_until $export 和 $ 中的参数 davinci-data-export	导出操作的时间过滤参数	2026 年 2 月 26 日
_include 搜索参数	HealthLake 现在支持包括:* 和包含:iterate	2026 年 2 月 26 日
CMS 互操作端点	此功能使您能够按内容管理系统类别跟踪 API 使用情况，并随后报告使用情况指标以实现合规性。	2026 年 2 月 26 日
捆绑包消息类型 Support	对消息类型的 FHIR 捆绑包资源的支持有限	2026 年 2 月 26 日
增加了对新功能的支持 IGs	AWS HealthLake 扩展了其 FHIR 实施指南 (IGs) 对 CMS 0057F 的支持： • 支持 CARIN Blue Button 2.0.0 和 2.1.0 • 支持 Da Vinci Payer Data Exchange 2.0.0 和 2.1.0 • 支持 DaVinci Payer Data Exchange (PDex) 美国药品处方集 2.0.1 和 2.1.0 • 支持 Da Vinci Clinical Data Exchange (CDex) 2.1.0 • 支持达芬奇事先授权支持 (PAS) FHIR IG 2.1.0	2026 年 2 月 26 日
$submit 操作	该$submit操作使您能够以电子方式向付款人提交事先授权请求以供批准。	2026 年 2 月 26 日

$问卷包操作	该$questionnaire-package 操作会检索一个包含FHIR调查问卷及其呈现和处理问卷所需的所有依赖项的综合包。	2026 年 2 月 26 日
$inquire 操作	该$inquire操作使您可以检查先前提交的先前授权请求的状态。	2026 年 2 月 26 日
$member-移除操作	$member-remove 操作允许您从中的 FHIR 成员归因列表（群组资源）中移除成员。AWS HealthLake	2025 年 11 月 12 日
$member-match 操作	AWS HealthLake 现在支持患者资源的 $member-Match 操作，使医疗保健组织能够使用人口统计和保险信息在不同的医疗保健系统中查找成员的唯一标识符。	2025 年 11 月 12 日
$member-add 操作	FHIR $member-add 操作将成员（患者）添加到群组资源，特别是成员归因列表。	2025 年 11 月 12 日
$ davinci-data-export 操作	$ davinci-data-export 操作是一种异步 FHIR 操作，允许从AWS HealthLake中导出成员归因列表数据。	2025 年 11 月 12 日
$ confirm-attribution-list 操作	向制作人表明消费者无需对归因列表进行任何更改，通过移除不活跃的成员并将状态更改为 “最终” 来最终确定归因列表。	2025 年 11 月 12 日

$归因状态操作	检索特定成员的归因状态，返回包含与患者相关的所有归因资源的捆绑包。	2025 年 11 月 12 日
FHIR 订阅	HealthLake 支持 FHIR 订阅，允许您在特定医疗保健数据发生变化时收到实时通知，并构建事件驱动的工作流程。	2025 年 10 月 30 日
地区扩展到加拿大蒙特利尔	HealthLake 已在加拿大（蒙特利尔）地区推出。有关更多信息，请参阅服务端点。	2025 年 10 月 17 日
增加了对新功能的支持 IGs	AWS HealthLake 已将其 FHIR 实施指南 (IGs) 支持范围扩大到以下加拿大市场： • CA Core+Canadian 基准 FHIR 概况定义了加拿大医疗保健系统之间互操作性的核心数据元素和限制。 • ca: erec Pan-Canadian eReferlar-eConsultStandardized FHIR规范，用于加拿大各地医疗保健提供者之间的电子转诊和咨询。 • 加拿大版《患者摘要》（PS-CA）加拿大改编版《国际患者摘要》（IPS），用于在各个护理环境中共享基本的患者健康信息。 • RepositoryOntarioOntario Digital Health Drug 特定的 FHIR 简介，用于省级卫生系统内的标准化药物和处方数据交换。	2025 年 10 月 17 日

特定区域的 IG 支持	HealthLake 现在支持特定区域 IGs。有关更多信息，请参阅配置文件验证。	2025 年 10 月 8 日
补丁操作	HealthLake 允许使用 JSON 补丁操作修改 FHIR 资源的特定元素，而无需更新整个资源。	2025 年 8 月 18 日
$验证操作	HealthLake 无需执行存储操作即可根据规格和配置文件验证 FHIR 资源，从而返回详细的验证结果。	2025 年 8 月 18 日
$清除操作	HealthLake 包括从数据存储中永久删除患者隔间内的所有资源的功能。	2025 年 8 月 18 日
$lookup 操作	HealthLake 提供了通过提供代码和系统标识符 CodeSystem 来检索有关中特定概念的详细信息的功能。	2025 年 8 月 18 日
$expand 操作	HealthLake 现在允许扩展 ValueSet 资源以检索客户 ValueSets提取的代码中包含的完整列表。	2025 年 8 月 18 日
$擦除操作	HealthLake 现在提供从数据存储中永久删除特定资源及其所有历史版本的功能。	2025 年 8 月 18 日
$文档操作	HealthLake 通过将组合资源及其所有引用资源捆绑到单个文档捆绑包中，支持生成完整的临床文档。	2025 年 8 月 18 日

:below 搜索修饰符	HealthLake 介绍在术语系统中搜索按层次结构低于指定 URI 的 URI 值。	2025 年 8 月 8 日
有条件删除	HealthLake 现在支持 FHIR 有条件删除，允许医疗保健组织根据搜索条件而不是逻辑 FHIR ID 删除现有资源。有关更多信息，请参阅根据条件删除 FHIR 资源。	2025 年 7 月 7 日
增加了对新功能的支持 IGs	AWS HealthLake 已扩展其 FHIR 实施指南 (IGs) 对以下内容的支持： • 美国核心 7.0.0，它规定了如何使用 FHIR 来实现 USCDI 4.0 标准 • UK Core 2.0.1 实施指南，提供全英国 FHIR 实施指导	2025 年 7 月 7 日
地区扩展到爱尔兰都柏林	HealthLake 已在欧洲（都柏林）地区推出。有关更多信息，请参阅服务端点。	2025 年 6 月 9 日
捆绑包类型交易	HealthLake 现在支持 FHIR Bundle 类型“交易”，允许医疗保健组织将多个资源作为单个原子操作提交。这可以实现更高效的数据交换和集成工作流程。例如，医疗保健提供者现在可以在一次交易中更新患者记录、药物清单和预约，从而降低复杂性和潜在错误。有关更多信息，请参阅捆绑 FHIR 资源。	2025 年 4 月 28 日

增加了对新功能的支持 IGs	AWS HealthLake AWS HealthLake 已扩展其 FHIR 实施指南 (IGs) 对以下内容的支持： • NCQA HEDIS® 实施指南 (0.3.1)：支持医疗保健有效性数据和信息集 (HEDIS) 的质量衡量和报告。 • 国际患者摘要 (IPS) (2.0.0)：允许交换基本的健康信息，以支持患者护理的连续性。 • 质量测量 (5.0.0)：支持质量衡量标准定义和数据的表示和交换。 • 基因组学报告 (3.0.0)：促进基因组数据和报告的交换。	2025 年 4 月 28 日
等能键	HealthLake 现在支持 FHIR POST 操作的等性密钥，提供了一种强大的机制来确保资源创建期间的数据完整性。有关更多信息，请参阅等性和并发性。	2025 年 4 月 18 日
FHIR 历史一致性	HealthLake 现在支持通过新的`x-amz-fhir-history-consistency-level` 标头为启用历史记录的数据存储提供强一致性。设置为 “strong” 时，无论更新状态如何，FHIR 搜索结果都将包含所有已编入索引的记录。有关更多信息，请参阅 FHIR 搜索一致性级别。	2025 年 4 月 18 日

Etag 和 “if-match”	HealthLake 现在提供 ETag 支持，允许客户端使用 “If-Match ” 标头来确保等性更新。这可防止并发更新期间意外覆盖，从而有助于维护数据的完整性。这在高容量医疗保健环境中尤其有价值，在这种环境中，多个系统可能会尝试同时更新同一记录。有关更多信息 ETag ，请参阅 AWS HealthLake 中的。	2025 年 4 月 18 日
以捆绑包 PUTs 为条件	HealthLake 现在支持 FHIR Bundle 的有条件更新，使医疗保健组织能够更灵活地管理和更新数据。现在，客户可以指定条件，在捆绑包事务中有条件地创建、更新或删除资源。这简化了系统之间的数据同步过程，减少了对复杂客户端逻辑的需求。有关更多信息，请参阅捆绑包 PUTs 中的条件性内容。	2025 年 4 月 18 日

FHIR V2 瞄准镜上的 SMART	HealthLake 支持 SMART on FHIR V2 作用域，用于创建、读取、更新、删除和搜索 FHIR 资源。有关更多信息，请参阅有关 FHIR 资源范围的 SMART。 HealthLake • SMART on FHIR V2 示波器适用于 2025 年 1 月 22 日之后创建的所有 HealthLake 数据存储。如果您的数据存储是在此日期之前创建的，则可以提交支持请求以启用 SMART on FHIR V2 范围。使用创建案例AWS Support Center Console。要创建您的案例，请登录您的 AWS 账户 并选择创建案例。	2025 年 1 月 22 日
FHIR 美国核心配置文件，版本 6.1.0	HealthLake 支持 FHIR 美国核心配置文件 6.1.0 版。有关更多信息，请参阅 FHIR 配置文件验证。 HealthLake	2025 年 1 月 22 日
FHIR $使用 GET 导出	HealthLake 支持 FHIR `$export` 使用。`GET`有关更多信息，请参阅使用 FHIR `$export` 导出 HealthLake 数据。	2025 年 1 月 22 日

包含经过测试的代码示例的重构开发者指南	HealthLake 引入了重构后的开发者指南，其中包含针对原生操作 AWS CLI 和 AWS SDK 操作的测试代码示例。此外，程序现在可用于所有支持的 FHIR API 交互。有关更多信息，请参阅代码示例和管理 FHIR 资源。	2024 年 12 月 18 日
FHIR history 和互动 vread	HealthLake 支持用于检索特定资源历史记录的 FHIR 交history互以及用于执行特定版本的资源读取的vread交互。有关更多信息，请参阅阅读 FHIR 资源历史记录。 • 在 2024 年 10 月 history 25 日之后创建的所有 HealthLake 数据存储中，默认启用 FHIR 资源。如果您的数据存储是在此日期之前创建的，则可以提交支持请求以启用 FHIR history 交互。使用创建案例AWS Support Center Console。要创建您的案例，请登录您的 AWS 账户 并选择创建案例。	2024 年 10 月 25 日

FHIR Patient/$everything 手术	HealthLake 支持用于搜索Patient资源及其所有相关资源的 FHIR Patient/$everything 操作。使用此操作，您可以访问患者的全部记录或批量下载Patient数据。有关更多信息，请参阅使用获取患者数据Patient/$everything 。 • 默认情况下Patient/$everything ，在 2024 年 2 月 27 日之后创建的所有 HealthLake 数据存储都启用 FHIR。如果您的数据存储是在此日期之前创建的，则可以提交支持请求以启用该Patient/$everything 操作。使用创建案例AWS Support Center Console。要创建您的案例，请登录您的 AWS 账户 并选择创建案例。	2024 年 2 月 27 日
FHIR 资源 VerificationResult	HealthLake 支持 FHIR VerificationResult 资源类型，用于描述一个或多个元素的验证要求、来源、状态和日期。有关更多信息，请参阅 FHIR R4 的资源类型。HealthLake	2023 年 12 月 9 日

FHIR $export 手术	HealthLake 支持 FHIR $export 操作，用于从数据存储中批量导出健康 HealthLake 数据。有关更多信息，请参阅使用 FHIR $export 导出 HealthLake 数据。 • 在 2023 年 6 月 1 日之后创建的所有 HealthLake 数据存储中，默认启用 FHIR $export。如果您的数据存储是在此日期之前创建的，则可以提交支持请求以启用该$export操作。使用创建案例AWS Support Center Console。要创建您的案例，请登录您的 AWS 账户 并选择创建案例。 • HealthLake 在 06/01/23 之前创建的数据存储仅支持系统范围导出的$export任务请求。 • HealthLake 在 06/01/23 之前创建的数据存储不支持$export使用数据存储端点上的GET请求获取 FHIR 的状态。	2023 年 6 月 1 日
SMART on FHIR 支持	HealthLake 添加了对 SMART on FHIR 授权的支持。有关更多信息，请参阅 SMART on FHIR 支持。 AWS HealthLake	2023 年 5 月 31 日

FHIR 个人资料验证	HealthLake 支持 FHIR 配置文件验证，用于使用基础资源类型的约束 and/or 扩展来定义特定的资源类型定义。有关更多信息，请参阅配置文件验证。	2023 年 5 月 31 日
亚太地区（孟买）区域	HealthLake 已在亚太地区（孟买）区域推出。有关更多信息，请参阅服务端点。	2023 年 4 月 4 日
默认情况下，自然语言处理处于关闭状态	HealthLake 自 2023 年 2 月 20 日起，已在所有数据存储上关闭集成自然语言处理 (NLP)。您可以提交支持请求以开启集成的 NLP 功能。使用创建案例AWS Support Center Console。要创建您的案例，请登录您的 AWS 账户 并选择创建案例。要了解有关集成 NLP 的更多信息，请参阅将 NLP 与集成。 HealthLake •	2023 年 2 月 20 日

使用 Amazon Athena 进行 SQL 索引和查询	HealthLake 支持使用 Amazon Athena 使用 SQL 查询 FHIR 数据。有关更多信息，请参阅使用 Amazon Athena 查询 HealthLake 数据。 • 在 2022 年 11 月 14 日之后创建的所有 HealthLake 数据存储中，默认启用 SQL 查询功能。如果您的数据存储是在此日期之前创建的，则可以提交支持票证以启用 SQL 查询功能。使用创建案例AWS Support Center Console。要创建您的案例，请登录您的 AWS 账户 并选择创建案例。 • 使用 SQL 查询功能，HealthLake 必须更新要访问的 IAM 设置。要在 Athena 中创建 HealthLake 数据存储并授予访问权限，您必须`AWSLakeFormationDataAdmin` 将托管策略添加到您的 IAM 用户、群组或角色中。您可以使用该`AWSLakeFormationDataAdmin` 策略创建数据湖管理员并授予对 Athena 中数据存储的访问权限。有关更多信息，请参阅配置 IAM 用户或角色。	2022 年 11 月 14 日

导入任务总量增加了	HealthLake 将StartFHIR ImportJob 请求Total import job size的，更新为 500 GB。有关更多信息，请参阅服务配额。	2022 年 10 月 3 日
FHIR 资源 Bundle	HealthLake 支持 FHIR Bundle 资源类型，用于同时处理多个 FHIR 资源。有关更多信息，请参阅捆绑 FHIR 资源。	2022 年 8 月 5 日
FHIR 互动的配额更新	HealthLake 更新 FHIR 资源管理交互的配额。有关更多信息，请参阅服务配额。	2022年7月16日
FHIR _include 搜索参数	HealthLake 添加了对 FHIR _include 搜索参数的支持，以便在search请求中返回其他资源。有关更多信息，请参阅高级搜索参数。	2022年7月16日
AWS HealthLake 已正式上市	HealthLake 已在所有支持的地区正式推出。有关更多信息，请参阅服务端点。	2021 年 7 月 15 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。