aws

개발자 가이드

AWS HealthLake

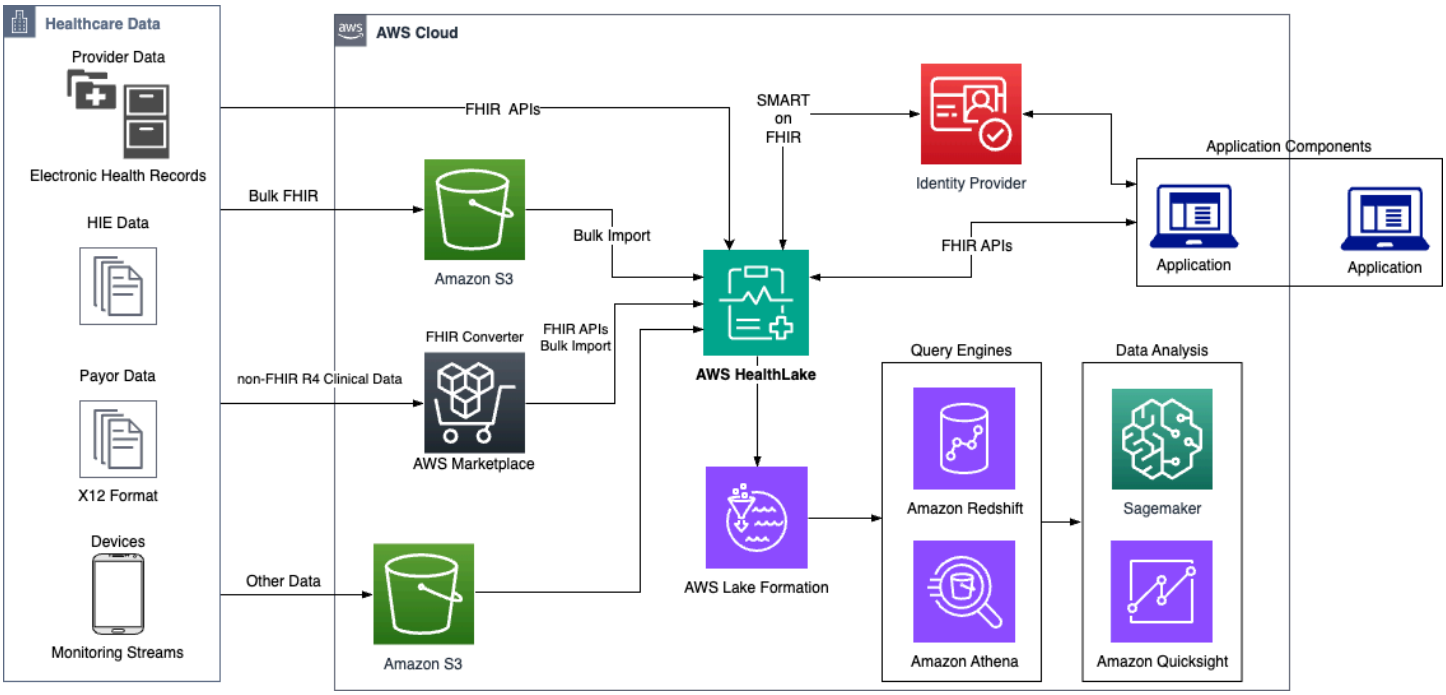
AWS HealthLake: 개발자 가이드

Table of Contents

AWS HealthLake란 무엇인가요?

AWS HealthLake 는 Fast Healthcare Interoperability Resources(FHIR) R4 사양을 사용하여 클라우드에서 상태 데이터를 저장, 분석 및 공유할 수 있는 HIPAA 적격 서비스입니다. HealthLake 사용 사례는 다음과 같습니다.

- 엔터프라이즈 상태 데이터 - 고성능과 가용성을 AWS 클라우드 유지하면서에서 직접 FHIR R4 상태 데이터를 관리하고 공유합니다.
- 의료 상호 운용성 - 완전 관리형 FHIR 데이터 스토어를 통한 환자 액세스를 위해 21세기 큐리 법 준수를 지원합니다.
- 자연어 처리(NLP) - 통합 NLP 모델을 활용하여 비정형 상태 데이터에서 의미 있는 의료 정보를 추출합니다.
- 멀티모달 분석 - HealthLake 데이터를 AWS HealthImaging 데이터 및 AWS HealthOmics 데이터와 결합하여 정밀 의학에 대한 인사이트를 제공합니다.



주제

- 중요 공지 사항
- 의 기능 AWS HealthLake
- 관련 AWS 서비스

- 액세스 AWS HealthLake
- HIPAA 자격 및 데이터 보안
- 가격 책정

중요 공지 사항

AWS HealthLake 는 전문적인 의학적 조언, 진단 또는 치료를 대체하지 않으며 질병 또는 건강 상태를 치료, 치료, 완화, 예방 또는 진단하기 위한 것이 아닙니다. 임상 의사 결정을 알리기 위한 타사 제품과 관련된 경우를 AWS HealthLake포함하여의 사용의 일부로 인적 검토를 도입할 책임이 있습니다. AWS HealthLake 는 적절한 의학적 판단을 적용하는 숙련된 의료 전문가의 검토 후에만 환자 치료 또는 임상 시나리오에 사용해야 합니다.

의 기능 AWS HealthLake

AWS HealthLake 는 다음과 같은 기능을 제공합니다.

FHIR R4 상태 데이터 가져오기

HealthLake 네이티브 가져오기 작업을 사용하면 임상 기록, 랩 보고서, 보험 청구 등을 포함하여 FHIR 데이터를 Amazon S3 버킷에서 HealthLake 데이터 스토어로 쉽게 마이그레이션할 수 있습니다. HealthLake는 의료 데이터 교환을 위한 FHIR R4 사양을 지원합니다. 필요한 경우 AWS HealthLake 파트너와 협력하여 상태 데이터를 FHIR R4 형식으로 변환할 수 있습니다.

안전하고 규정을 준수하며 감사 가능한 방식으로 상태 데이터 저장

HealthLake 데이터 스토어는 상태 데이터를 인덱싱하여 쿼리할 수 있도록 도와줍니다. 데이터 스토어는 시간순으로 각 환자의 의료 기록에 대한 전체 보기를 생성하고 FHIR R4 사양을 사용하여 정보 교환을 용이하게 합니다. 또한 인덱스를 최신 상태로 유지하기 위해 항상 실행되므로 내구성이 뛰어난 기본 스토리지 및 인덱스 조정과 함께 표준 FHIR R4 상호 작용을 사용하여 언제든지 정보를 검색할 수 있습니다.

트랜잭션 FHIR 서버 활용

표준 리소스 검증, SMART on FHIR 권한 부여 및 대량 데이터 FHIR APIs 내보내기 기능에 FHIR API를 활용하여 데이터 통합 및 분석을 지원함으로써 운영 비용을 절감하고 의사 결정을 개선합니다. HealthLake는 HL7 FHIR R4 APIs, FHIR 대량 데이터 액세스, US Core IG STU, HL7 SMART 앱 시작 프레임워크 IG, OAuth 2.0 및 OpenID Connect를 포함한 최신 ONC 및 CMS 규제 표준에 대한 고객 적합성을 지원합니다.

NLP를 사용하여 비정형 의료 데이터 변환

통합 의료 자연어 처리(NLP)는 HealthLake 데이터 스토어의 모든 원시 의료 텍스트 데이터를 변환하여 비정형 의료 데이터에서 의미 있는 정보를 이해하고 추출합니다. 통합 의료 NLP를 사용하면 의료 텍스트에서 엔터티, 엔터티 관계, 엔터티 특성 및 보호 대상 건강 정보(PHI)를 자동으로 추출할 수 있습니다. NLP 추출 엔터티는 HealthLake 데이터 스토어 내에 네이티브 FHIR R4 리소스로 저장되며 FHIR R4 APIs 또는 Amazon Athena(SQL)를 통해 액세스할 수 있습니다.

관련 AWS 서비스

AWS HealthLake 는 다른 AWS 서비스와 긴밀하게 통합됩니다. 다음 서비스에 대한 지식은 HealthLake를 완전히 활용하는 데 유용합니다.

- AWS Identity and Access Management - IAM을 사용하여 자격 증명과 HealthLake 리소스에 대한 액세스를 안전하게 관리합니다.
- Amazon Simple Storage Service - Amazon S3를 스테이징 영역으로 사용하여 DICOM 데이터를 HealthLake로 가져옵니다.
- AWS CloudTrail - CloudTrail을 사용하여 HealthLake 사용자 활동 및 API 사용량을 추적합니다.
- Amazon CloudWatch - CloudWatch를 사용하여 HealthLake 리소스를 관찰하고 모니터링합니다.
- AWS CloudFormation - CloudFormation 를 사용하여 코드형 인프라(IaC) 템플릿을 구현하여 HealthLake에서 리소스를 생성합니다.
- AWS PrivateLink - Amazon VPC를 사용하여 인터넷에 데이터를 노출하지 않고 HealthLake와 Amazon Virtual Private Cloud 간에 연결을 설정합니다.
- Amazon EventBridge - EventBridge를 사용하여 HealthLake 이벤트를 대상으로 라우팅하는 규칙을 생성하여 확장 가능한 이벤트 기반 애플리케이션을 생성합니다.
- AWS Lake Formation - Lake Formation을 사용하여 분석 및 기계 학습을 위해 HealthLake 데이터를 중앙에서 관리, 보호 및 공유합니다.
- Amazon Athena - Athena를 사용하여 심층 분석을 위해 SQL로 HealthLake 데이터를 쿼리합니다.

액세스 AWS HealthLake

AWS Management Console AWS Command Line Interface 및SDK를 AWS HealthLake 사용하여에 액세스할 수 있습니다. AWS SDKs 이 가이드에서는 AWS Management Console 및 SDK의 AWS CLI 및 AWS 코드 예제에 대한 절차 지침을 제공합니다. SDKs

AWS Command Line Interface (AWS CLI)

는 다양한 AWS 제품에 대한 명령을 AWS CLI 제공하며 Windows, Mac 및 Linux에서 지원됩니다. 자세한 내용은 AWS Command Line Interface 사용 설명서를 참조하십시오.

AWS SDKs

AWS SDKs 소프트웨어 개발자를 위한 라이브러리, 코드 예제 및 기타 리소스를 제공합니다. 이러한 라이브러리는 요청에 암호화 서명, 요청 재시도, 오류 응답 처리 등과 같은 작업을 자동으로 관리하는 기본 기능을 제공합니다. 자세한 내용은 빌드 기반 도구를 참조하세요 AWS.

AWS Management Console

는 HealthLake 및 관련 리소스를 관리하기 위한 웹 기반 사용자 인터페이스를 AWS Management Console 제공합니다. AWS 계정에 가입한 경우 HealthLake 콘솔에 로그인할 수 있습니다.

HIPAA 자격 및 데이터 보안

이것은 HIPAA 적격 서비스입니다. AWS미국 HIPAA(Health Insurance Portability and Accountability Act of 1996) 및 AWS 서비스를 사용하여 보호 대상 건강 정보(PHI)를 처리, 저장 및 전송하는 방법에 대한 자세한 내용은 HIPAA 개요를 참조하세요.

PHI 및 개인 식별 정보(PII)가 포함된 HealthLake에 대한 연결은 암호화되어야 합니다. 기본적으로 HealthLake에 대한 모든 연결은 TLS를 통한 HTTPS를 사용합니다. HealthLake는 암호화된 고객 콘텐츠를 저장하고 AWS 공동 책임 모델에 따라 작동합니다.

가격 책정

HealthLake 요금 정보는 AWS HealthLake 요금을 참조하세요. 비용을 추정하려면 HealthLake 요금 계산기를 사용합니다.

시작하기 AWS HealthLake

사용을 시작하려면 AWS 계정을 AWS HealthLake설정하고 AWS Identity and Access Management 사용자를 생성합니다. AWS CLI또는 AWS SDK 사용하려면 반드시 이를 설치하고 구성해야 합니다.

> **Note**
>
> 이 가이드의 레퍼런스 장에서는 FHIR, FHIR R4 및의 SMART에 대한 지원 콘텐츠를 제공합니다 AWS HealthLake. 예를 들어, FHIR 구성의 SMART, 지원되는 FHIR 프로필 검증 및 HealthLake 엔드포인트에 대한 정보를 찾을 수 있습니다.

HealthLake 개념에 대해 알아보고 설정한 후 코드 예제가 포함된 짧은 자습서를 사용하여 시작할 수 있습니다.

주제

- AWS HealthLake 개념
- 설 AWS HealthLake정
- AWS HealthLake 자습서

AWS HealthLake 개념

다음 용어와 개념은를 이해하고 사용하는 데 매우 중요합니다 AWS HealthLake.

개념

- 데이터 스토어 권한 부여 전략
- 통합 NLP
- 통합 분석

데이터 스토어 권한 부여 전략

HealthLake 데이터 스토어는 단일 내에 있는 FHIR R4 상태 데이터의 리포지토리입니다 AWS 리전. HealthLake는 다음과 같은 데이터 스토어 권한 부여 전략을 지원합니다.

- SigV4 권한 부여 - HealthLake는 AWS 서명 버전 4(SigV4) 권한 부여를 사용하여 FHIR API 호출을 승인합니다.

- FHIR 인증의 SMART - HealthLake는 FHIR 인증에서 대체 가능한 의료 애플리케이션 및 재사용 가능한 기술(SMART)을 사용하여 FHIR API 호출을 승인합니다.

자세한 내용은 HealthLake 데이터 스토어 생성 단원을 참조하십시오.

통합 NLP

AWS HealthLake 는 HIPAA 적격 자연어 처리(NLP) 라이브러리와 통합되어 비정형 의료 텍스트에서 의미 있는 상태 데이터를 추출합니다. NLP 라이브러리는 상태, 약물, 용량, 테스트, 치료 및 절차와 같은 의료 엔터티를 식별합니다. 이들은 개체 간의 관계를 인식하고 ICD-10-CM 및 RxNorm과 같은 의료 온톨로지 라이브러리에 연결합니다. 자세한 내용은 HealthLake용 통합 자연어 처리(NLP) 단원을 참조하십시오.

통합 분석

AWS HealthLake 는 FHIR `search` 및 `bundle` APIs를 넘어 대량의 상태 데이터를 쿼리하고 분석하기 위한 통합 분석을 제공합니다. 가져오는 동안 HealthLake는 SQL 인덱스 및 쿼리에 대한 테이블을 자동으로 생성합니다. 이를 통해 광범위한 데이터 엔지니어링 작업 없이 복잡한 의료 데이터에서 실행 가능한 인사이트를 얻을 수 있습니다. 자세한 내용은 Amazon Athena를 사용하여 HealthLake 데이터 쿼리 및 AWS HealthLake 샘플 프로젝트 섹션을 참조하세요.

설 AWS HealthLake정

이 장에서는 AWS Management Console 를 사용하여 사용을 시작하고 데이터 스토어를 AWS HealthLake 생성하는 데 필요한 권한을 설정합니다. 데이터 스토어를 생성할 수 있는 권한을 설정하려면 데이터 레이크 관리자 및 HealthLake 관리자인 IAM 사용자 또는 역할을 생성합니다. 이 사용자를 AWS Lake Formation의 데이터 레이크 관리자로 설정합니다. 데이터 레이크 관리자는 Amazon Athena를 사용하여 데이터 스토어를 쿼리하는 데 필요한 리소스에 대한 액세스 권한을 Lake Formation에 부여합니다. HealthLake 데이터 스토어를 생성한 후 파일을 가져오고 내보낼 수 있는 권한을 설정할 수 있습니다.

주제

- 에 가입 AWS 계정
- 관리자 액세스 권한이 있는 사용자 생성
- HealthLake를 사용하도록 IAM 사용자 또는 역할 구성(IAM 관리자)
- Lake Formation에서 Data Lake 관리자로 사용자 또는 역할 추가(IAM 관리자)
- S3 버킷 생성

- 데이터 스토어 생성
- 가져오기 작업에 대한 권한 설정
- 내보내기 작업에 대한 권한 설정
- 설치 AWS CLI

에 가입 AWS 계정

이 없는 경우 다음 단계를 AWS 계정완료하여 생성합니다.

에 가입하려면 AWS 계정

1. https://portal.aws.amazon.com/billing/signup을 엽니다.
2. 온라인 지시 사항을 따르세요.

   등록 절차 중 전화 또는 텍스트 메시지를 받고 전화 키패드로 확인 코드를 입력하는 과정이 있습니다.

   에 가입하면 AWS 계정AWS 계정 루트 사용자이 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스에 액세스할 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

AWS 는 가입 프로세스가 완료된 후 확인 이메일을 보냅니다. 언제든지 https://aws.amazon.com/으로 이동하고 내 계정을 선택하여 현재 계정 활동을 확인하고 계정을 관리할 수 있습니다.

관리자 액세스 권한이 있는 사용자 생성

에 가입한 후 일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 AWS 계정보호 AWS IAM Identity Center, AWS 계정 루트 사용자활성화 및 생성합니다.

보안 AWS 계정 루트 사용자

1. 루트 사용자를 선택하고 AWS 계정 이메일 주소를 입력하여 계정 소유자AWS Management Console로에 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.

   루트 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS Sign-In 사용 설명서의 루트 사용자로 로그인을 참조하세요.
2. 루트 사용자의 다중 인증(MFA)을 활성화합니다.

지침은 IAM 사용 설명서의 AWS 계정 루트 사용자(콘솔)에 대한 가상 MFA 디바이스 활성화를 참조하세요.

관리자 액세스 권한이 있는 사용자 생성

1. IAM Identity Center를 활성화합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 AWS IAM Identity Center설정을 참조하세요.
2. IAM Identity Center에서 사용자에게 관리 액세스 권한을 부여합니다.

   를 자격 증명 소스 IAM Identity Center 디렉터리 로 사용하는 방법에 대한 자습서는 사용 AWS IAM Identity Center 설명서의 기본값으로 사용자 액세스 구성을 IAM Identity Center 디렉터리 참조하세요.

관리 액세스 권한이 있는 사용자로 로그인

- IAM IDentity Center 사용자로 로그인하려면 IAM Identity Center 사용자를 생성할 때 이메일 주소로 전송된 로그인 URL을 사용합니다.

  IAM Identity Center 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS Sign-In 사용 설명서의 AWS 액세스 포털에 로그인을 참조하세요.

추가 사용자에게 액세스 권한 할당

1. IAM Identity Center에서 최소 권한 적용 모범 사례를 따르는 권한 세트를 생성합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 Create a permission set를 참조하세요.
2. 사용자를 그룹에 할당하고, 그룹에 Single Sign-On 액세스 권한을 할당합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 그룹 추가를 참조하세요.

HealthLake를 사용하도록 IAM 사용자 또는 역할 구성(IAM 관리자)

> 페르소나: IAM 관리자
>
> IAM 사용자 및 역할을 생성하고 데이터 레이크 관리자를 추가할 수 있는 사용자입니다.

이 주제의 이러한 단계는 IAM 관리자가 수행해야 합니다.

HealthLake 데이터 스토어를 Athena에 연결하려면 데이터 레이크 관리자 및 HealthLake 관리자인 IAM 사용자 또는 역할을 생성해야 합니다. 이 새 사용자 또는 역할은 AWS Lake Formation을 통해 데이터 스토어에 있는 리소스에 대한 액세스 권한을 부여하고 `AmazonHealthLakeFullAccess` AWS 관리형 정책을 사용자 또는 역할에 추가합니다.

> **Important**
>
> 데이터 레이크 관리자인 IAM 사용자 또는 역할은 새 데이터 레이크 관리자를 생성할 수 없습니다. 데이터 레이크 관리자를 추가하려면 `AdministratorAccess` 액세스 권한이 부여된 IAM 사용자 또는 역할을 사용해야 합니다.

관리자를 생성하려면

1. 조직의 사용자 또는 역할에 **`AmazonHealthlakeFullAccess`** IAM AWS 관리형 정책을 추가합니다.

   IAM 사용자 생성에 익숙하지 않은 경우 IAM 사용 설명서의 IAM 사용자 생성 및 IAM AWS 정책 개요를 참조하세요.
2. IAM 사용자 또는 역할에 AWS Lake Formation에 대한 액세스 권한을 부여합니다.

   - 조직의 사용자 또는 역할에 다음 IAM AWS 관리형 정책을 추가합니다. **`AWSLakeFormationDataAdmin`**

     > **Note**
     >
     > 이 `AWSLakeFormationDataAdmin` 정책은 모든 AWS Lake Formation 리소스에 대한 액세스 권한을 부여합니다. 작업을 완료하는 데 필요한 최소 권한을 항상 사용하는 것이 좋습니다. 자세한 설명은 IAM 사용자 가이드의 IAM 모범 사례 섹션을 참조하세요.
3. 사용자 또는 역할에 다음 인라인 정책을 추가합니다. 자세한 내용은 IAM 사용 설명서의 인라인 정책을 참조하세요.

   JSON

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-source-bucket/*",
                "arn:aws:s3:::amzn-s3-demo-logging-bucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ram:GetResourceShareInvitations",
                "ram:AcceptResourceShareInvitation",
                "glue:CreateDatabase",
                "glue:DeleteDatabase"
            ],
            "Resource": "*"
        }
    ]
}
```

AWSLakeFormationDataAdmin 정책에 대한 자세한 내용은 Lake Formation 개발자 안내서의 Lake Formation 페르소나 및 IAM 권한 참조를 참조하세요. AWS

Lake Formation에서 Data Lake 관리자로 사용자 또는 역할 추가(IAM 관리자)

> **Note**
>
> 를 통합하는 경우이 단계가 필요합니다SQL 인덱스 및 쿼리.

다음으로 IAM 관리자는 이전 단계에서 생성한 사용자 또는 역할을 Lake Formation의 데이터 레이크 관리자로 추가해야 합니다.

IAM 사용자 또는 역할을 데이터 레이크 관리자로 추가하려면

1. AWS Lake Formation 콘솔 열기: https://console.aws.amazon.com/lakeformation/

   > **Note**
   >
   > Lake Formation을 처음 방문하는 경우 Lake Formation 관리자를 정의하라는 Lake Formation 시작 대화 상자가 나타납니다.

2. AWS Lake Formation 데이터 레이크 관리자가 될 새 사용자 또는 역할을 할당합니다.
   - 옵션 1: Lake Formation 시작 대화 상자를 받은 경우.
     1. 다른 AWS 사용자 또는 역할 추가를 선택합니다.
     2. 아래쪽 화살표(")를 선택합니다.
     3. Lake Formation 관리자로도 사용할 HealthLake 관리자를 선택합니다.
     4. Get started를 선택합니다.
   - 옵션 2: 탐색 창(")을 사용합니다.
     1. 탐색 창(")을 선택합니다.
     2. 권한에서 관리 역할 및 작업을 선택합니다.
     3. 데이터 레이크 관리자 섹션에서 관리자 선택을 선택합니다.
     4. 데이터 레이크 관리자 관리 대화 상자에서 아래쪽 화살표(")를 선택합니다.
     5. 그런 다음 Lake Formation 관리자로도 사용할 HealthLake 관리자 사용자 또는 역할을 선택하거나 검색합니다.
     6. 저장을 선택합니다.
3. Lake Formation에서 관리할 기본 보안 설정을 변경합니다. HealthLake 데이터 스토어 리소스는 IAM이 아닌 Lake Formation에서 관리해야 합니다. 업데이트하려면 AWS Lake Formation 개발자 안내서의 기본 권한 모델 변경을 참조하세요.

S3 버킷 생성

FHIR R4 데이터를 로 가져오려면 Amazon S3 버킷 AWS HealthLake 2개를 사용하는 것이 좋습니다. Amazon S3 입력 버킷에는 가져올 FHIR 데이터와이 버킷의 HealthLake 읽기가 들어 있습니다. Amazon S3 출력 버킷은 가져오기 작업 및 HealthLake 쓰기(로그)의 처리 결과를이 버킷에 저장합니다.

> **Note**
>
> AWS Identity and Access Management (IAM) 정책으로 인해 Amazon S3 버킷 이름은 고유해야 합니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 버킷 이름 지정 규칙을 참조하세요.

이 가이드에서는이 섹션의 뒷부분에서 가져오기 권한을 설정할 때 다음과 같은 Amazon S3 입력 및 출력 버킷을 지정합니다.

- 입력 버킷: `arn:aws:s3:::amzn-s3-demo-source-bucket`
- 출력 버킷: `arn:aws:s3:::amzn-s3-demo-logging-bucket`

자세한 내용은 Amazon S3 사용 설명서의 버킷 생성하기를 참조하세요.

데이터 스토어 생성

HealthLake 데이터 스토어는 단일 AWS 리전 내에 있는 FHIR R4 데이터의 리포지토리입니다. AWS 계정은 데이터 스토어가 0개이거나 많을 수 있습니다. HealthLake는 두 가지 데이터 스토어 권한 부여 전략을 지원합니다.

> **중요**
>
> HealthLake 데이터 스토어를 생성하기 전에 HealthLake 리소스의 생성 또는 관리를 제한할 수 있는 AWS 조직의 서비스 제어 정책(SCPs)을 검토하세요. SCPs IAM 권한이 올바르게 설정된 경우에도 HealthLake 데이터 스토어를 성공적으로 생성하지 못하게 할 수 있습니다.
> HealthLake 데이터 스토어를 생성할 때 `datastoreID`가 생성됩니다. 이 섹션의 뒷부분에서 가져오기 권한을 설정할 `datastoreID` 때를 사용해야 합니다.

HealthLake 데이터 스토어를 생성하려면 섹션을 참조하세요HealthLake 데이터 스토어 생성.

가져오기 작업에 대한 권한 설정

파일을 데이터 스토어로 가져오기 전에 HealthLake에 Amazon S3의 입력 및 출력 버킷에 액세스할 수 있는 권한을 부여해야 합니다. HealthLake 액세스 권한을 부여하려면 HealthLake에 대한 IAM 서비스 역할을 생성하고, 역할에 신뢰 정책을 추가하여 HealthLake 수임 역할 권한을 부여하고, 역할에 권한 정책을 연결하여 Amazon S3 버킷에 대한 액세스 권한을 부여합니다.

가져오기 작업을 생성할 때에 대해이 역할의 Amazon 리소스 이름(ARN)을 지정합니다.`DataAccessRoleArn`. IAM 역할 및 신뢰 정책에 대한 자세한 내용은 IAM 역할을 참조하세요.

권한을 설정한 후에는 가져오기 작업을 통해 파일을 데이터 스토어로 가져올 준비가 된 것입니다. 자세한 내용은 FHIR 가져오기 작업 시작 단원을 참조하십시오.

가져오기 권한을 설정하려면

1. 아직 생성하지 않은 경우 출력 로그 파일에 대한 대상 Amazon S3 버킷을 생성합니다. Amazon S3 버킷은 서비스와 동일한 AWS 리전에 있어야 하며 모든 옵션에 대해 퍼블릭 액세스 차단을 켜야 합니다. 자세한 내용은 Amazon S3 퍼블릭 액세스 차단 사용을 참조하세요. Amazon 소유 또는 고객 소유 KMS 키를 암호화에도 사용해야 합니다. KMS 키 사용에 대한 자세한 내용은 Amazon Key Management Service를 참조하세요.
2. HealthLake에 대한 데이터 액세스 서비스 역할을 생성하고 HealthLake 서비스에 다음 신뢰 정책을 사용하여 이를 수임할 수 있는 권한을 부여합니다. HealthLake는 이를 사용하여 출력 Amazon S3 버킷을 작성합니다.

   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. Amazon S3 버킷에 액세스할 수 있도록 허용하는 권한 정책을 데이터 액세스 역할에 추가합니다. `amzn-s3-demo-bucket`를 버킷 이름으로 바꿉니다.

   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"
    }]
}
```

내보내기 작업에 대한 권한 설정

데이터 스토어에서 파일을 내보내기 전에 HealthLake에 Amazon S3의 출력 버킷에 액세스할 수 있는 권한을 부여해야 합니다. HealthLake 액세스 권한을 부여하려면 HealthLake에 대한 IAM 서비스 역할을 생성하고, 역할에 신뢰 정책을 추가하여 HealthLake 수임 역할 권한을 부여하고, 역할에 권한 정책을 연결하여 Amazon S3 버킷에 대한 액세스 권한을 부여합니다.

HealthLake에 대한 역할을 이미 생성한 경우 역할을 재사용하고이 주제에 나열된 Amazon S3 버킷 내보내기에 대한 추가 권한을 부여할 수 있습니다. IAM 역할 및 신뢰 정책에 대한 자세한 내용은 IAM 권한 및 정책을 참조하세요.

중요

HealthLake는 네이티브 SDK 내보내기 요청과 FHIR R4 `$export` 작업을 모두 지원합니다. 사용하려는 내보내기 API에 따라 별도의 IAM 작업을 제공해야 합니다. 이렇게 하면 `allow` 및 `deny` 권한을 별도로 처리할 수 있습니다. HealthLake SDK 및 FHIR REST API 내보내기를 모두 제한하려면 별도의 IAM 작업에 거부 권한을 적용해야 합니다. 사용자에게 HealthLake에 대한 전체 액세스 권한을 부여하는 경우 IAM 사용자 권한 변경이 필요하지 않습니다.

AWS CLI 및 AWS SDKs 사용:

및 AWS CLI SDK를 사용하여 데이터 스토어에서 데이터를 내보내는 데 사용할 수 있는 기본 HealthLake 작업은 다음과 같습니다. AWS SDKs

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

FHIR APIs 사용:

다음 IAM 작업은 HealthLake 데이터 스토어에서 데이터를 내보내고 FHIR 작업을 사용하여 내보내기 작업을 취소(삭제)하는 데 사용할 수 `$export` 있습니다.
POST:

- `StartFHIRExportJobWithPost`

GET:

- StartFHIRExportJobWithGet
- DescribeFHIRExportJobWithGet
- GetExportedFile

DELETE:

- CancelFHIRExportJobWithDelete

권한을 설정하는 사용자 또는 역할에는 역할을 생성하고, 정책을 생성하고, 정책을 역할에 연결할 수 있는 권한이 있어야 합니다. 다음 IAM 정책은 이러한 권한을 부여합니다.

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"
                }
            }
        }
    ]
}
```

내보내기 권한을 설정하려면

1. 아직 생성하지 않은 경우 데이터 스토어에서 내보낼 데이터에 대한 대상 Amazon S3 버킷을 생성합니다. Amazon S3 버킷은 서비스와 동일한 AWS 리전에 있어야 하며 모든 옵션에 대해 퍼블릭 액세스 차단을 켜야 합니다. 자세한 내용은 Amazon S3 퍼블릭 액세스 차단 사용을 참조하세요. Amazon 소유 또는 고객 소유 KMS 키도 암호화에 사용해야 합니다. KMS 키 사용에 대한 자세한 내용은 Amazon Key Management Service를 참조하세요.
2. 아직 생성하지 않은 경우 HealthLake에 대한 데이터 액세스 서비스 역할을 생성하고 HealthLake 서비스에 다음 신뢰 정책을 사용하여 수임할 수 있는 권한을 부여합니다. HealthLake는 이를 사용하여 출력 Amazon S3 버킷을 작성합니다. 에서 이미 생성한 경우 가져오기 작업에 대한 권한 설정다시 사용하고 다음 단계에서 Amazon S3 버킷에 대한 권한을 부여할 수 있습니다.

   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. 데이터 액세스 역할에 출력 Amazon S3 버킷에 액세스할 수 있는 권한 정책을 추가합니다. `amzn-s3-demo-bucket`를 버킷 이름으로 바꿉니다.

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"
    }]
}
```

설치 AWS CLI

AWS CLI 는 HealthLake 가져오기 및 내보내기 작업 속성을 설명하고 나열하는 데 필요합니다. HealthLake SDKs.

를 설정하려면 AWS CLI

1. AWS CLI를 다운로드하고 구성합니다. 관련 지침은AWS Command Line Interface 사용 설명서에서 다음 주제를 참조하세요.

   - 최신 버전의 설치 또는 업데이트 AWS CLI
   - 시작하기 AWS CLI

2. AWS CLI `config` 파일에 관리자의 명명된 프로필을 추가합니다. AWS CLI 명령을 실행할 때 이 프로파일을 사용합니다. 최소 권한이라는 보안 원칙에 따라, 수행 중인 작업과 관련된 권한을 가진 별도의 IAM 역할을 생성하는 것이 좋습니다. 명명된 프로파일에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 구성 및 자격 증명 파일 설정을 참조하세요.

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

3. 다음 `help` 명령을 사용하여 설정을 확인합니다.

   ```
   aws healthlake help
   ```

   AWS CLI 가 올바르게 구성된 경우에 대한 AWS HealthLake 간략한 설명과 사용 가능한 명령 목록이 표시됩니다.

AWS HealthLake 자습서

목표

이 자습서에서는 기본 HealthLake 작업을 사용하여 FHIR R4 데이터를 HealthLake 데이터 스토어로 가져옵니다. 다음으로 FHIR RESTful APIs. 자습서를 마치려면 네이티브 HealthLake 작업을 사용하여 FHIR 데이터를 내보냅니다.

사전 조건

이 튜토리얼을 완료하려면 설정에 나열된 모든 절차가 필요합니다.

튜토리얼 단계

1. FHIR 가져오기 작업 시작
2. FHIR 가져오기 작업 속성 가져오기
3. FHIR 리소스 생성
4. FHIR 리소스 읽기
5. FHIR 리소스 업데이트
6. FHIR 리소스 삭제
7. FHIR 데이터 내보내기
8. 데이터 스토어 삭제

를 사용하여 데이터 스토어 관리 AWS HealthLake

AWS HealthLake를 사용하면 FHIR R4 리소스에 대한 데이터 스토어를 생성하고 관리할 수 있습니다. HealthLake 데이터 스토어를 생성하면 RESTful API 엔드포인트를 통해 FHIR 데이터 리포지토리를 사용할 수 있습니다. Synthea 오픈 소스 FHIR R4 상태 데이터를 생성할 때 데이터 스토어로 가져오도록(사전 로드) 선택할 수 있습니다. 자세한 내용은 사전 로드된 데이터 형식 단원을 참조하십시오.

> **중요**
>
> HealthLake는 FHIR의 AWS SigV4 또는 SMART라는 두 가지 유형의 FHIR 데이터 스토어 권한 부여 전략을 지원합니다. HealthLake FHIR 데이터 스토어를 생성하기 전에 권한 부여 전략 중 하나를 선택해야 합니다. 자세한 내용은 데이터 스토어 권한 부여 전략 단원을 참조하십시오.

활성 HealthLake 데이터 스토어의 FHIR 관련 기능(동작)을 찾으려면 기능 설명을 검색합니다.

다음 주제에서는 HealthLake 클라우드 네이티브 작업을 사용하여 ,SDK 및를 사용하여 FHIR 데이터 스토어를 생성, 설명 AWS CLI, 나열, 태그 지정 및 삭제하는 방법을 설명합니다 AWS Management Console. AWS SDKs

주제

- HealthLake 데이터 스토어 생성
- HealthLake 데이터 스토어 속성 가져오기
- HealthLake 데이터 스토어 나열
- HealthLake 데이터 스토어에 태그 지정
- HealthLake 데이터 스토어 삭제

HealthLake 데이터 스토어 생성

`CreateFHIRDatastore`를 사용하여 FHIR R4 사양을 준수하는 AWS HealthLake 데이터 스토어를 생성합니다. HealthLake 데이터 스토어는 FHIR 데이터를 가져오고, 관리하고, 검색하고, 내보내는 데 사용됩니다. Synthea 오픈 소스 FHIR R4 상태 데이터를 생성할 때 데이터 스토어로 가져오도록(사전 로드) 선택할 수 있습니다. 자세한 내용은 사전 로드된 데이터 형식 단원을 참조하십시오.

> **중요**
>
> HealthLake는 FHIR의 AWS SigV4 또는 SMART라는 두 가지 유형의 FHIR 데이터 스토어 권한 부여 전략을 지원합니다. HealthLake FHIR 데이터 스토어를 생성하기 전에 권한 부여 전략 중 하나를 선택해야 합니다. 자세한 내용은 데이터 스토어 권한 부여 전략 단원을 참조하십시오.

HealthLake 데이터 스토어를 생성하면 RESTful API 엔드포인트를 통해 FHIR 데이터 리포지토리를 사용할 수 있습니다. HealthLake 데이터 스토어를 생성한 후 기능 설명을 요청하여 연결된 모든 FHIR 관련 기능(동작)을 찾을 수 있습니다.

다음 메뉴는 AWS CLI 및 AWS SDKs에 대한 예제와에 대한 프로시저를 제공합니다 AWS Management Console. 자세한 내용을 알아보려면 AWS HealthLake API 참조의 *CreateFHIRDatastore* 섹션을 참조하세요.

HealthLake 데이터 스토어를 생성하려면

액세스 기본 설정에 따라 메뉴를 선택합니다 AWS HealthLake.

AWS CLI 및 SDKs

CLI

AWS CLI

예제 1: SigV4-enabled HealthLake 데이터 스토어 생성

다음 `create-fhir-datastore` 예제에서는 AWS HealthLake에서 새 데이터 스토어를 생성하는 방법을 보여줍니다.

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

출력:

```
{
    "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)"
}
```

예제 2: FHIR 지원 HealthLake 데이터 스토어에서 SMART 생성

다음 `create-fhir-datastore` 예제에서는 AWS HealthLake에서 FHIR 지원 데이터 스토어에서 새 SMART를 생성하는 방법을 보여줍니다.

```
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.json의 콘텐츠:

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

출력:

```
{
```

```
    "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)"
}
```

자세한 내용은 AWS HealthLake 개발자 안내서의 FHIR 데이터 스토어 생성 및 모니터링을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 CreateFHIRDatastore 섹션을 참조하세요.

Python

SDK for Python(Boto3)

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

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


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

```
        - identity_provider_configuration: The identity provider configuration
for a SMART on FHIR-enabled data store.

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

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

다음 코드에서는 SMART on FHIR이 활성화된 HealthLake 데이터 저장소의 파라미터 예시를 보여줍니다.

```
            sse_configuration = {
                "KmsEncryptionConfig": {"CmkType": "AWS_OWNED_KMS_KEY"}
            }
            # TODO: Update the metadata to match your environment.
```

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

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 CreateFHIRDatastore를 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

- API 세부 정보는 SDK for SAP ABAP API 참조의 CreateFHIRDatastore를 참조하세요. AWS

> 예제 가용성
>
> 필요한 예제를 찾을 수 없습니까? 이 페이지의 오른쪽 사이드바에 있는 피드백 제공 링크를 사용하여 코드 예제를 요청합니다.

AWS 콘솔

> Note
>
> 다음 절차에서는 AWS SigV4 권한 부여를 사용하여 HealthLake 데이터 스토어를 생성합니다. HealthLake 콘솔은 FHIR 데이터 스토어에서 SMART 생성을 지원하지 않습니다.

AWS SigV4 권한 부여를 사용하여 HealthLake 데이터 스토어를 생성하려면

1. HealthLake 콘솔의 데이터 스토어 생성 페이지에 로그인합니다.
2. 데이터 스토어 생성을 선택합니다.
3. 데이터 스토어 설정 섹션의 데이터 스토어 이름에 이름을 지정합니다.
4. (선택 사항) 데이터 스토어 설정 섹션의 샘플 데이터 사전 로드에서 Synthea 데이터를 사전 로드할 확인란을 선택합니다. Synthea 데이터는 오픈 소스 샘플 데이터 세트입니다. 자세한 내용은 HealthLake용 Synthea 사전 로드된 데이터 형식 단원을 참조하십시오.
5. 데이터 스토어 암호화 섹션에서 AWS 소유 키 사용(기본값) 또는 다른 AWS KMS 키 선택(고급)을 선택합니다.
6. 태그 - 선택 사항 섹션에서 데이터 스토어에 태그를 추가할 수 있습니다. 데이터 스토어 태그 지정에 대한 자세한 내용은 섹션을 참조하세요HealthLake 데이터 스토어에 태그 지정.
7. 데이터 스토어 생성을 선택합니다.

   데이터 스토어의 상태는 데이터 스토어 페이지에서 확인할 수 있습니다.

HealthLake 데이터 스토어 속성 가져오기

`DescribeFHIRDatastore`를 사용하여 AWS HealthLake 데이터 스토어의 속성을 가져옵니다. 다음 메뉴는 AWS Management Console 및 SDKs의 AWS CLI 및 AWS 코드 예제에 대한 절차를 제공합니

다. 자세한 내용을 알아보려면 AWS HealthLake API 참조의 *DescribeFHIRDatastore* 섹션을 참조하세요.

HealthLake 데이터 스토어의 속성을 가져오는 방법

액세스 기본 설정에 따라 메뉴를 선택합니다 AWS HealthLake.

AWS CLI 및 SDKs

CLI

AWS CLI

FHIR 데이터 스토어를 설명하려면

다음 `describe-fhir-datastore` 예제에서는 AWS HealthLake에서 데이터 스토어의 속성을 찾는 방법을 보여줍니다.

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

출력:

```
{
    "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
        }
    }
}
```

자세한 내용은 AWS HealthLake 개발자 안내서의 FHIR 데이터 스토어 생성 및 모니터링을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DescribeFHIRDatastore 섹션을 참조하세요.

Python

SDK for Python(Boto3)

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

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


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

```
                "Couldn't describe data store with ID %s. Here's why %s",
                datastore_id,
                err.response["Error"]["Message"],
            )
            raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 DescribeFHIRDatastore를 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        oo_result = lo_hll->describefhirdatastore(
          iv_datastoreid = iv_datastore_id
        ).
        DATA(lo_datastore_properties) = oo_result->get_datastoreproperties( ).
        IF lo_datastore_properties IS BOUND.
          DATA(lv_datastore_name) = lo_datastore_properties-
>get_datastorename( ).
          DATA(lv_datastore_status) = lo_datastore_properties-
>get_datastorestatus( ).
          MESSAGE 'Data store described successfully.' TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
```

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

- API 세부 정보는 SDK for SAP ABAP API 참조의 DescribeFHIRDatastore를 참조하세요. AWS

> 예제 가용성
>
> 필요한 예제를 찾을 수 없습니까? 이 페이지의 오른쪽 사이드바에 있는 피드백 제공 링크를 사용하여 코드 예제를 요청합니다.

AWS 콘솔

1. HealthLake 콘솔의 데이터 스토어 페이지에 로그인합니다.
2. 데이터 스토어를 선택합니다.

   데이터 스토어 세부 정보 페이지가 열리고 모든 HealthLake 데이터 스토어 속성을 사용할 수 있습니다.

HealthLake 데이터 스토어 나열

데이터 스토어 상태에 관계없이 사용자 계정의 모든 HealthLake 데이터 스토어를 나열하는 `ListFHIRDatastores` 데 사용합니다. 다음 메뉴에서는 및 AWS SDKs의 AWS Management Console AWS CLI 및 코드 예제에 대한 절차를 제공합니다. 자세한 내용을 알아보려면 AWS HealthLake API 참조의 *`ListFHIRDatastores`* 섹션을 참조하세요.

모든 HealthLake 데이터 스토어를 나열하려면

액세스 기본 설정에 따라 메뉴를 선택합니다 AWS HealthLake.

AWS CLI 및 SDKs

CLI

AWS CLI

FHIR 데이터 스토어를 나열하려면

다음 `list-fhir-datastores` 예제에서는 명령을 사용하는 방법과 사용자가 AWS HealthLake의 데이터 스토어 상태를 기반으로 결과를 필터링하는 방법을 보여줍니다.

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

출력:

```
{
    "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
        }
    }
```

```
    ]
}
```

자세한 내용은 AWS HealthLake 개발자 안내서의 FHIR 데이터 스토어 생성 및 모니터링을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListFHIRDatastores 섹션을 참조하세요.

Python

SDK for Python(Boto3)

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

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


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

            # Loop through paginated results.
            while True:
                parameters = {}
                if next_token is not None:
                    parameters["NextToken"] = next_token
                response =
self.health_lake_client.list_fhir_datastores(**parameters)
                datastores.extend(response["DatastorePropertiesList"])
                if "NextToken" in response:
```

```
                    next_token = response["NextToken"]
                else:
                    break

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

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 ListFHIRDatastores를 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

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

- API 세부 정보는 SDK for SAP ABAP API 참조의 ListFHIRDatastores를 참조하세요. AWS

> **예제 가용성**
>
> 필요한 예제를 찾을 수 없습니까? 이 페이지의 오른쪽 사이드바에 있는 피드백 제공 링크를 사용하여 코드 예제를 요청합니다.

AWS 콘솔

- HealthLake 콘솔의 데이터 스토어 페이지에 로그인합니다.

  모든 HealthLake 데이터 스토어는 데이터 스토어 섹션 아래에 나열됩니다.

HealthLake 데이터 스토어에 태그 지정

태그 형태로 HealthLake 데이터 스토어에 메타데이터를 할당할 수 있습니다. 각 태그는 사용자 정의 키와 값으로 구성된 레이블입니다. 태그는 데이터 스토어를 관리, 식별, 구성, 검색 및 필터링하는 데 도움이 됩니다.

> **중요**
>
> 개인 식별 정보(PII)나 기타 기밀 정보 또는 민감한 정보를 태그에 저장하지 마십시오. 태그는 개인 데이터나 민감한 데이터에 사용하기 위한 것이 아닙니다.

다음 주제에서는 AWS Management Console AWS CLI, 및 AWS SDKs를 사용하여 HealthLake 태그 지정 작업을 사용하는 방법을 설명합니다. 자세한 내용은 AWS 일반 참조 가이드의 AWS 리소스 태그 지정을 참조하세요.

주제

HealthLake 데이터 스토어에 태그 지정

`TagResource`를 사용하여 HealthLake 데이터 스토어에 태그를 지정합니다. 다음 메뉴는 AWS Management Console 및 AWS SDKs의 AWS CLI 및 코드 예제에 대한 절차를 제공합니다. 자세한 내용을 알아보려면 AWS HealthLake API 참조의 *TagResource* 섹션을 참조하세요.

HealthLake 데이터 스토어에 태그를 지정하려면

액세스 기본 설정에 따라 메뉴를 선택합니다 AWS HealthLake.

AWS CLI 및 SDKs

CLI

AWS CLI

데이터 스토어에 태그를 추가하려면

다음 `tag-resource` 예제에서는 데이터 스토어에 태그를 추가하는 방법을 보여 줍니다.

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

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS HealthLake개발자 안내서의 데이터 스토어에 태그 추가를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 TagResource를 참조하세요.

Python

SDK for Python(Boto3)

```
    @classmethod
```

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

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


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

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 TagResource를 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 TagResource를 참조하세요.

> **예제 가용성**
>
> 필요한 예제를 찾을 수 없습니까? 이 페이지의 오른쪽 사이드바에 있는 피드백 제공 링크를 사용하여 코드 예제를 요청합니다.

AWS 콘솔

1. HealthLake 콘솔의 데이터 스토어 페이지에 로그인합니다.
2. 데이터 스토어를 선택합니다.

   데이터 스토어 세부 정보 페이지가 열립니다.
3. 태그 섹션에서 태그 관리를 선택합니다.

   태그 관리 페이지가 열립니다.
4. 새 태그 추가를 선택합니다.
5. 키와 값(선택 사항)을 입력합니다.
6. 저장을 선택합니다.

HealthLake 데이터 스토어에 대한 태그 나열

`ListTagsForResource`를 사용하여 HealthLake 데이터 스토어의 태그를 나열합니다. 다음 메뉴는 AWS Management Console 및 AWS SDKs의 AWS CLI 및 코드 예제에 대한 절차를 제공합니다. 자세한 내용을 알아보려면 AWS HealthLake API 참조의 *`ListTagsForResource`* 섹션을 참조하세요.

HealthLake 데이터 스토어의 태그를 나열하려면

액세스 기본 설정에 따라 메뉴를 선택합니다 AWS HealthLake.

AWS CLI 및 SDKs

CLI

AWS CLI

데이터 스토어의 태그를 나열하려면

다음 `list-tags-for-resource` 예제에서는 지정된 데이터 스토어와 연결된 태그를 나열합니다.

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

출력:

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

자세한 내용은AWS HealthLake 개발자 안내서의HealthLake에서 리소스 태그 지정을 참조하세요. AWS HealthLake

- API 세부 정보는 AWS CLI 명령 참조의 ListTagsForResource를 참조하세요.

Python

SDK for Python(Boto3)

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

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


    def list_tags_for_resource(self, resource_arn: str) -> dict[str, str]:
        """
        Lists the tags for a HealthLake resource.
        :param resource_arn: The resource ARN.
        :return: The tags for the resource.
        """
        try:
            response = self.health_lake_client.list_tags_for_resource(
                ResourceARN=resource_arn
            )
            return response["Tags"]
        except ClientError as err:
            logger.exception(
```

```
                "Couldn't list tags for resource %s. Here's why %s",
                resource_arn,
                err.response["Error"]["Message"],
            )
            raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 ListTagsForResource를 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        " iv_resource_arn = 'arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        DATA(lo_result) = lo_hll->listtagsforresource(
          iv_resourcearn = iv_resource_arn
        ).
        ot_tags = lo_result->get_tags( ).
        DATA(lv_tag_count) = lines( ot_tags ).
        MESSAGE |Found { lv_tag_count } tag(s).| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
```

```
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- API 세부 정보는 SDK for SAP ABAP API 참조의 ListTagsForResource를 참조하세요. AWS

> **예제 가용성**
>
> 필요한 예제를 찾을 수 없습니까? 이 페이지의 오른쪽 사이드바에 있는 피드백 제공 링크를 사용하여 코드 예제를 요청합니다.

AWS 콘솔

1. HealthLake 콘솔의 데이터 스토어 페이지에 로그인합니다.
2. 데이터 스토어를 선택합니다.

   데이터 스토어 세부 정보 페이지가 열립니다. 태그 섹션 아래에 모든 데이터 스토어 태그가 나열됩니다.

HealthLake 데이터 스토어 태그 해제

`UntagResource`를 사용하여 HealthLake 데이터 스토어에서 태그를 제거합니다. 다음 메뉴는 AWS Management Console 및 AWS SDKs의 AWS CLI 및 코드 예제에 대한 절차를 제공합니다. 자세한 내용을 알아보려면 AWS HealthLake API 참조의 *`UntagResource`* 섹션을 참조하세요.

HealthLake 데이터 스토어의 태그를 해제하려면

액세스 기본 설정에 따라 메뉴를 선택합니다 AWS HealthLake.

AWS CLI 및 SDKs

CLI

AWS CLI

데이터 스토어에서 태그를 제거하려면

다음 `untag-resource` 예제에서는 데이터 스토어에서 태그를 제거하는 방법을 보여 줍니다.

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

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS HealthLake 개발자 안내서의 데이터 스토어에서 태그 제거를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 UntagResource를 참조하세요.

Python

SDK for Python(Boto3)

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

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


    def untag_resource(self, resource_arn: str, tag_keys: list[str]) -> None:
        """
        Untags a HealthLake resource.
        :param resource_arn: The resource ARN.
        :param tag_keys: The tag keys to remove from the resource.
```

```
        """
        try:
            self.health_lake_client.untag_resource(
                ResourceARN=resource_arn, TagKeys=tag_keys
            )
        except ClientError as err:
            logger.exception(
                "Couldn't untag resource %s. Here's why %s",
                resource_arn,
                err.response["Error"]["Message"],
            )
            raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 UntagResource를 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        " iv_resource_arn = 'arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        lo_hll->untagresource(
          iv_resourcearn = iv_resource_arn
          it_tagkeys = it_tag_keys
        ).
        MESSAGE 'Resource untagged successfully.' TYPE 'I'.
```

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

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 UntagResource를 참조하세요.

> 예제 가용성
>
> 필요한 예제를 찾을 수 없습니까? 이 페이지의 오른쪽 사이드바에 있는 피드백 제공 링크를 사용하여 코드 예제를 요청합니다.

AWS 콘솔

1. HealthLake 콘솔의 데이터 스토어 페이지에 로그인합니다.
2. 데이터 스토어를 선택합니다.

   데이터 스토어 세부 정보 페이지가 열립니다.
3. 태그 섹션에서 태그 관리를 선택합니다.

   태그 관리 페이지가 열립니다.
4. 제거할 태그 옆에 있는 제거를 선택합니다.
5. 저장을 선택합니다.

HealthLake 데이터 스토어 삭제

`DeleteFHIRDatastore`를 사용하여 HealthLake 데이터 스토어를 삭제합니다. 다음 메뉴에서는 AWS Management Console 및 AWS SDKs의 AWS CLI 및 코드 예제에 대한 절차를 제공합니다. 자세한 내용을 알아보려면 AWS HealthLake API 참조의 *`DeleteFHIRDatastore`* 섹션을 참조하세요.

HealthLake 데이터 스토어를 삭제하려면

액세스 기본 설정에 따라 메뉴를 선택합니다 AWS HealthLake.

AWS CLI 및 SDKs

CLI

AWS CLI

FHIR 데이터 스토어를 삭제하려면

다음 `delete-fhir-datastore` 예제에서는 AWS HealthLake에서 데이터 스토어와 모든 콘텐츠를 삭제하는 방법을 보여줍니다.

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

출력:

```
{
    "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)"
}
```

자세한 내용은 AWS HealthLake 개발자 안내서의 FHIR 데이터 스토어 <https://docs.aws.amazon.com/healthlake/latest/devguide/working-with-FHIR-healthlake.html> 생성 및 모니터링을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DeleteFHIRDatastore 섹션을 참조하세요.

Python

SDK for Python(Boto3)

```
    @classmethod
```

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

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


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

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

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 DeleteFHIRDatastore를 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        oo_result = lo_hll->deletefhirdatastore(
          iv_datastoreid = iv_datastore_id
        ).
        MESSAGE 'Data store deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_hllaccessdeniedex INTO DATA(lo_access_ex).
        DATA(lv_error) = |Access denied: { lo_access_ex->av_err_code }-
{ lo_access_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_access_ex.
      CATCH /aws1/cx_hllconflictexception INTO DATA(lo_conflict_ex).
        lv_error = |Conflict error: { lo_conflict_ex->av_err_code }-
{ lo_conflict_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_conflict_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- API 세부 정보는 SDK for SAP ABAP API 참조의 DeleteFHIRDatastore를 참조하세요. AWS

> **예제 가용성**
>
> 필요한 예제를 찾을 수 없습니까? 이 페이지의 오른쪽 사이드바에 있는 피드백 제공 링크를 사용하여 코드 예제를 요청합니다.

AWS 콘솔

1. HealthLake 콘솔의 데이터 스토어 페이지에 로그인합니다.
2. 데이터 스토어를 선택합니다.

   데이터 스토어 세부 정보 페이지가 열립니다.
3. 삭제를 선택합니다.

   데이터 스토어 삭제 페이지가 열립니다.
4. 데이터 스토어 삭제를 확인하려면 텍스트 입력 필드에 데이터 스토어 이름을 입력합니다.
5. 삭제를 선택합니다.

에서 FHIR 구독 관리 AWS HealthLake

AWS HealthLake 는 FHIR 구독을 지원하므로 특정 의료 데이터 변경이 발생할 때 실시간 알림을 받을 수 있습니다. 이 기능은 FHIR R5 백포트 주제 기반 구독 모델을 구현하여 기존 FHIR R4 구독 모델에 비해 향상된 확장성과 유연성을 제공합니다.

FHIR 구독을 사용하면 임상 데이터의 변화에 즉시 대응하는 이벤트 기반 의료 애플리케이션을 구축하여 적시에 개입하고 워크플로를 자동화하며 의료 조정을 강화할 수 있습니다.

주제

FHIR 구독 작동 방식

HealthLake의 FHIR 구독은 다음과 같은 주제 기반 모델에서 작동합니다.

1. 이벤트를 정의하는 주제 생성: 알림을 트리거할 수 있는 이벤트를 지정하는 구독 주제 생성
2. 구독: 특정 필터링 기준을 사용하여 이러한 주제에 대한 구독 생성
3. HealthLake 모니터: 서비스가 기준과 일치하는 이벤트를 지속적으로 모니터링합니다.
4. 전달된 알림: CWhen일치 이벤트가 발생하면 HealthLake는 선택한 채널을 통해 알림을 전송합니다.

핵심 구성 요소

FHIR 구독은 다음 구성 요소로 구성됩니다.

구독 주제

구독 주제는 알림 시스템의 기반이며 다음을 정의합니다.

- 트리거 이벤트: 알림을 트리거하는 변경 사항(예: 리소스 생성, 업데이트, 삭제)
- 사용 가능한 필터: 구독자가 사용할 수 있는 필터링 옵션
- 알림 콘텐츠: 알림에 포함되는 데이터

다음 표에는 일반적인 주제 유형이 나열되어 있습니다.

이벤트 유형	설명	일반 사용 사례
리소스 생성	리소스가 생성될 때 트리거됩니다.	새 환자 등록, 새 관찰 기록
리소스 업데이트	리소스가 수정될 때 트리거됩니다.	상태 변경, 임상 업데이트
리소스 삭제	리소스가 삭제될 때 트리거됩니다.	감사 및 규정 준수 추적

구독

구독은 구독 주제에서 정의한 특정 이벤트에 대한 알림을 수신하기 위한 요청입니다. 각 구독에는 다음이 포함됩니다.

- 주제 참조: 구독하려는 구독 주제를 지정합니다.
- 필터: 알림을 생성하는 이벤트를 선택하는 기준
- 채널 구성: 알림을 전달해야 하는 위치 및 방법
- 페이로드 기본 설정: 알림에 포함해야 하는 세부 정보 수준

알림 채널

HealthLake는 다음 알림 채널을 지원합니다.

채널 유형	사용 사례	
EventBridge	엔터프라이즈 통합, 서버리스 워크플로, 교차AWS 서비스 오케스트레이션	
REST Hook	직접 엔드포인트 알림, 타사 시스템 통합	

알림 페이로드

필요에 따라 적절한 페이로드 유형을 선택합니다.

페이로드 유형	설명	보안 고려 사항
ID 전용	리소스 식별자만 포함	최소 PHI 노출
전체 리소스	최대 크기가 256KB인 전체 리소스 콘텐츠를 포함합니다. 크기가 256KB보다 크면 ID 전용으로 돌아갑니다.	PHI 포함, 보안 처리 확인

모범 사례

성능 최적화

- 중점 필터 사용: 필수 알림만 수신하도록 기준 좁히기
- 적절한 페이로드 유형 선택: 성능 향상을 위해 가능하면 ID 전용 페이로드 사용
- 효율적인 수신자 구현: 알림 수신자가 메시지를 빠르게 처리하도록 보장

보안 고려 사항

- 보안 엔드포인트: REST Hook 엔드포인트에 대한 적절한 인증 구현
- PHI 보호: PHI가 포함되어 있으므로 전체 리소스 페이로드에 주의하세요.
- 액세스 제어: 구독 생성을 승인된 사용자로만 제한

운영 우수성

- 적절한 종료 날짜 설정: 임시 구독에 종료 날짜 사용
- 구독 상태 모니터링: 구독 상태를 정기적으로 확인합니다.
- 오류 처리 구현: 알림 전송 실패를 처리하도록 애플리케이션 설계

를 사용한 FHIR 구독 수명 주기 AWS HealthLake

다음 단계에 따라 FHIR 구독 수명 주기를 이해합니다.

1. SubscriptionTopic 생성
   - 상태가 SubscriptionTopic 인 생성 "unknown"
2. Subscription 생성
   - 상태가 Subscription 인 생성 "requested"
   - HealthLake는 Subscription 구성을 검증합니다.
   - Subscription는 이미 존재하는 주제를 참조해야 합니다(주제는 상태 unknown, draft,에 있어야 함active).
3. 활성화
   - 유효한 경우 HealthLake는 상태를 Subscription 로 업데이트합니다. "active"
   - 를 생성하는 동안 지정된 주제가 상태인 Subscription경우 "unknown" HealthLake는 구독도 활성화"active"되면 상태를 로 업데이트합니다.
   - 구독을 성공적으로 생성하는 데 보통 5~10분이 걸립니다.
   - Subscription가 성공적으로 생성되지 않으면 상태가 DELETE 작업을 수행해야 하는 error 위치로 변경된 다음 구독 생성을 다시 시도합니다. 구독 리소스의 "error" 필드를 보고 구독이 성공적으로 생성되지 않은 이유를 확인할 수 있습니다.
4. 구독이 인 동안 수집 active
5. Subscription가 인 동안 active
   - HealthLake는 기준과 일치하는 이벤트를 모니터링합니다.
   - 일치가 발생하면 구성된 엔드포인트로 알림이 전송됩니다.
6. 오류 처리
   - HealthLake는 14일 동안 재시도를 시도한 다음 해당 이벤트에 대한 재시도를 중지합니다.
7. 비활성화
   - 는 다음을 통해 비활성화할 Subscription 수 있습니다.

종료 날짜 설정(자동 비활성화)

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

Subscription 리소스 삭제

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

를 사용하여 FHIR 구독 생성 AWS HealthLake

다음 가이드에서는를 사용하여 FHIR 구독을 생성하는 방법을 보여줍니다 AWS HealthLake.

FHIR 구독을 생성하려면

1. SubscriptionTopic를 만듭니다.

   구독 주제 리소스의 예:

```
{
  "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. 알림 엔드포인트(사용자 지정 채널)를 준비합니다. 다음 단계는 엔드포인트가 알림을 수신하도록 하는 데 필요한 단계입니다.

   REST 후크를 사용하는 경우

   - CM_CMK 데이터 스토어events.amazonaws.com를 사용하는 경우 KMS 키 정책을 신뢰합니다.
   - CM_CMK 데이터 스토어를 사용하는 경우 값을 사용하여 KMS 키에 EventBridgeApiDestinations 태그를 추가해야 합니다. true

- HealthLake는 OAuth를 사용하여 REST 후크 엔드포인트를 인증합니다. 따라서 REST 후크 구독을 생성할 때 channel._type.extension[*]에서 client-id, client-secret 및 oAuth-endpoint-url을 전달해야 합니다.

CM_CMK 데이터 스토어를 사용하는 경우의 KMS 키 정책 예제:

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

EventBridge를 사용하는 경우

- CM_CMK 데이터 스토어`events.amazonaws.com`를 사용하는 경우 KMS 키 정책을 신뢰합니다.
- EventBridge 리소스 정책이 서비스 보안 주체`healthlake.amazonaws.com`로 신뢰하는지 확인합니다.
- CM_CMK를 사용하고 EventBridge가 엔드포인트인 경우 데이터 스토어 KMS 키와 동일한 KMS 키로 EventBridge 버스를 암호화하고 있는지 확인합니다.
- EventBridge 버스에 HealthLake에서 생성된 이벤트와 일치하는 규칙이 하나 이상 있는지 확인합니다.

EventBridge 채널 버스에 대한 리소스 정책 예제:

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

HealthLake에서 이벤트를 수신하는 EventBridge 규칙 이벤트 패턴의 예:

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

> **Note**
>
> HealthLake는 다음 두 가지 소스를 지원합니다.
>
> - “healthlake”: 구독에만 해당됩니다.
> - “aws.healthlake”: HealthLake 서비스 이벤트를 수신합니다.
>
> FHIR 구독 이벤트 버스에 대한 규칙을 생성할 때를 소스“healthlake”로 사용합니다.

3. Subscription을(를) 생성하기

   다음을 사용하여 구독 리소스를 제출합니다.

   - 상태: "requested"
   - 선택한 SubscriptionTopic ID에 대한 참조
   - 필터 기준. 자세한 내용은 지원되는 필터에 대한 알림 필터링을 참조하세요.
   - 채널 구성

구독 페이로드 예제

다음 코드 예제에서는 구독 페이로드를 생성하는 방법을 보여줍니다.

EventBridge

event-bridge 채널 및 id-only페이로드 유형이 포함된 구독.

```
{
  "resourceType": "Subscription",
```

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

event-bridge 엔드포인트 및 full-resource페이로드 유형이 포함된 구독.

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

Rest 후크

`rest-hook` 엔드포인트 및 `id-only`페이로드 유형이 포함된 구독.

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

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

```
    }
  }
}
```

rest-hook 채널 및 full-resource페이로드 유형이 포함된 구독.

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

알림 페이로드 예제

구독이 생성되는 동안 HealthLake는 구성된 채널로 핸드셰이크 번들을 전송하여 성공적인 구독 설정을 확인합니다. 다음 페이로드는 핸드셰이크 번들의 예입니다.

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

```
          "resource": {
            "resourceType": "SubscriptionStatus",
            "id": "<HANDSHAKE_RESOURCE_ID>",
            "status": "requested",
            "type": "handshake",
            "eventsSinceSubscriptionStart": "0",
            "subscription": {
              "reference": "https://healthlake.<AWS_REGION>.amazonaws.com/datastore/
<DS_ID>/r4/Subscription/<SUBSCRIPTION_ID>"
            },
            "topic": "https://healthlake.<AWS_REGION>.amazonaws.com/datastore/<DS_ID>/
r4/SubscriptionTopic/<TOPIC_ID>"
          }
        }
      ]
    }
  }
}
```

ID 전용 알림 번들의 예입니다.

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

전체 리소스 알림 번들의 예입니다.

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

이벤트 버전 관리

HealthLake는 기본적으로 FHIR 기록을 지원합니다.

알림 번들에서 받은 리소스 버전을 확인하려면:

- 전체 리소스: 전체 리소스 번들에는 전체 리소스가 포함되므로 번들에 있는 각 리소스에 `entry[*]` 대해 버전이 내에 포함됩니다.
- id-only: 번들에는 리소스 정보가 포함되지 않습니다. HealthLake에는 `entry[0].notificationEvent[*].additionalContext` 필드를 통해 매칭되고 번들에 포함된 버전이 포함됩니다. 이 필드는 형식입니다`<ResourceType>/<ResourceId>/_history/<Version Id>`. 자세한 내용은 예제 ID 전용 페이로드의 additionalContext 필드를 참조하세요.

이벤트 중복 감지

HealthLake의 FHIR 구독 기능은 하나 이상의 전송을 보장합니다. 즉, 동일한 번들 또는 다른 번들에서 동일한 이벤트를 여러 번 수신할 수 있습니다. 중복을 식별하기 위해 HealthLake는의 알림 번들에 있는 각 이벤트에 대해 고유한 ID를 제공합니다`entry[0].notificationEvent[*].id`.

이 ID는 일치 및 전송된 이벤트의 특정 버전에 고유합니다. 예를 들어 동일한 Encounter가 두 번 업데이트되고 두 업데이트가 모두 필터 기준과 일치하는 경우 동일한 Encounter 참조가 있는 두 개의 개별

이벤트를 받게 됩니다. 동일한를 가지notificationEvent[*].focus지만 고유한를 가지게 됩니다notificationEvent[*].id. 또한 이러한 이벤트는 별도의 번들로 또는 동일한 알림 번들 내에서 전송될 수 있습니다.

를 사용하여 FHIR 구독 검색 AWS HealthLake

Subscription 및 SubscriptionTopic 리소스도 검색할 수 있습니다. HealthLake는 Subscription 및 SubscriptionTopic 리소스에 대한 모든 공통 검색 파라미터를 지원합니다.

또한 다음 파라미터를 통해 추가 검색 기능을 지원합니다.

Subscription

검색 파라미터	설명	예제
contact	R4 코어 사양의 Subscription.contact 필드를 검색합니다.	`Subscription?contact=phone`
기준	R4 코어 사양의 Subscription.criteria 필드를 검색합니다.	`Subscription?criteria=[baseUrl]/datastore/[datastoreId]/r4/SubscriptionTopic/[topicId]`
payload	R4 코어 사양의 Subscription.channel.payload 필드를 검색합니다.	`Subscription?payload=application/fhir+json`
status	R4 코어 사양의 Subscription.status 필드를 검색합니다.	`Subscription?status=error`
type	R4 코어 사양의 Subscription.channel.type 필드를 검색합니다.	`Subscription?topic=event-bridge`
url	R4 코어 사양의 Subscription.channel.endpoint 필드에서 검색	`Subscription?url=[Subscription.channel.endpoint]`

검색 파라미터	설명	예제
filter-criteria	URL "http://hl7.org/fhir/uv/subscriptions-backport/StructureDefinition/backport-filter-criteria"을 문자열로 사용하여 기준 확장 필드를 검색합니다.	`Subscription?filter-criteria=Encounter?`
사용자 지정 채널	url "http://hl7.org/fhir/uv/subscriptions-backport/StructureDefinition/backport-channel-type"을 코딩으로 사용하여 사용자 지정 채널 유형 확장 필드를 검색합니다. 구독 페이로드 예제	`Subscription?custom-channel=[System]|[code]`
페이로드 유형	페이로드 형식 지정 방법을 지정하는 페이로드 유형 확장 필드를 검색합니다(`id-only` 또는 `full-resource`).	`Subscription?payload-type=full-resource`
주제	주제가 추가되는 Subscription.criteria 필드를 검색합니다.	`Subscription?topic=[topicId]`

SubscriptionTopic

검색 파라미터	설명	예제
날짜	SubscriptionTopic 리소스의 날짜 필드를 검색합니다.	`SubscriptionTopic?date=[SubscriptionTopic.date]`
derived-or-self	SubscriptionTopic 리소스의 `url` 또는 `derivedFrom` 필드를 검색합니다.	`SubscriptionTopic?derived-or-self=[SubscriptionTopic.url | Subscript`

검색 파라미터	설명	예제
		ionTopic.derivedFrom]
식별자	SubscriptoinTopic.identifier 필드에서 검색	SubscriptionTopic?identifier=[SubscriptionTopic.identifier]
리소스	SubscriptionTopic.resourceTrigger.resource 필드에서 검색	SubscriptionTopic?resource=Encounter
status	SubscriptionTopic의 상태 검색	SubscriptionTopic?status=unknown
제목	SubscriptionTopic의 제목 검색	SubscriptionTopic?title=admission
trigger-description	SubscriptionTopic.resourceTrigger.description에서 검색	SubscriptionTopic.trigger-description=resource moving to state 'in-progress'
url	URL에서 SubscriptionTopic 검색	SubscriptionTopic?url=[SubscriptionTopic.url]
버전	SubscriptionTopic 버전 검색	SubscriptionTopic?version=1

를 사용하여 알림 필터링 AWS HealthLake

Subscription 기준에서 표준 FHIR 검색 파라미터를 사용하여 알림을 구체화합니다.

지원되는 필터

다음 표에는 HealthLake가 구독에 대해 지원하는 지원되는 필터 기준 목록이 나와 있습니다.

검색 파라미터 유형	FHIR 데이터 형식	Modifiers(한정자)	지원되는 접두사
문자열	문자열 HumanName 주소	exact, 누락 포함	
토큰	부울 code 문자열 코딩 CodeableConcept 식별자 ContactPoint id	not, text, missing	
숫자	정수 decimal positiveInt unsignedInt	missing	"eq", "ne", "gt", "lt", "ge", "le", "sa", "eb", "ap"
Date	날짜 dateTime 인스턴트 Period Timing(타이밍)		"eq", "ne", "gt", "lt", "ge", "le", "sa", "eb", "ap"

검색 파라미터 유형	FHIR 데이터 형식	Modifiers(한정자)	지원되는 접두사
수량	수량 화폐 Range SimpleQuantity		"eq", "ne", "gt", "lt", "ge", "le", "sa", "eb", "ap"
레퍼런스	레퍼런스	누락, 식별자, 유형	
URI	uri url canonical uid oid	missing	

지원되는 필터 샘플

다음 표에는 HealthLake가 구독에 대해 지원하는 지원되는 필터 기준의 샘플이 나와 있습니다.

용도	필터 기준	설명
환자별 관찰	`Observation?patient=Patient/[id]&status=final`	특정 환자의 관찰이 완료되면 알림 받기
환자별 관찰	`Patient?birthdate=gt2021`	2021년 이후에 태어난 환자가 등록되거나 업데이트되면 알림 받기
환자별 관찰	`Condition?code=http://snomed.info/sct|39065001`	특정 SNOMED 코드가 있는 조건에 대한 알림 받기

용도	필터 기준	설명
환자별 관찰	`Observation?code=http://loinc.org|8480-6&value-quantity=gt160`	높은 수축성 혈압 측정값에 대한 알림 받기

를 사용하여 FHIR 데이터 가져오기 AWS HealthLake

HealthLake 데이터 스토어를 생성한 후 다음 단계는 Amazon Simple Storage Service(S3) 버킷에서 파일을 가져오는 것입니다. AWS Management Console AWS CLI또는 AWS SDKs. 기본 AWS HealthLake 작업을 사용하여 FHIR 가져오기 작업을 시작, 설명 및 나열합니다.

> **중요**
>
> HealthLake는 의료 데이터 교환을 위한 FHIR R4 사양을 지원합니다. 필요한 경우 AWS HealthLake 파트너와 협력하여 가져오기 전에 상태 데이터를 FHIR R4 형식으로 변환할 수 있습니다.

FHIR 가져오기 작업을 시작할 때 Amazon S3 버킷 입력 위치, Amazon S3 버킷 출력 위치(작업 처리 결과용), HealthLake에 Amazon S3 버킷에 대한 액세스 권한을 부여하는 IAM 역할, 고객 소유 또는 AWS 소유 AWS Key Management Service 키를 지정합니다. 자세한 내용은 가져오기 작업에 대한 권한 설정 단원을 참조하십시오.

> **Note**
>
> 가져오기 작업을 대기열에 넣을 수 있습니다. 비동기 가져오기 작업은 FIFO(선입선출) 방식으로 처리됩니다. 가져오기 작업을 시작하는 것과 동일한 방식으로 작업을 대기열에 넣을 수 있습니다. 진행 중인 경우 대기열에 추가하기만 하면 됩니다. 가져오기 작업이 진행되는 동안 FHIR 리소스를 생성, 읽기, 업데이트 또는 삭제할 수 있습니다.

HealthLake는 각 FHIR 가져오기 작업에 대한 `manifest.json` 파일을 생성합니다. 파일은 FHIR 가져오기 작업의 성공과 실패를 모두 설명합니다. HealthLake는 FHIR 가져오기 작업을 시작할 때 지정된 Amazon S3 버킷에 `manifest.json` 파일을 출력합니다. 로그 파일은 SUCCESS 및 라는 두 개의 폴더로 구성됩니다FAILURE. 각 `manifest.json` 파일에 대한 세부 정보를 제공하므로 실패한 가져오기 작업 문제를 해결하는 첫 번째 단계로 파일을 사용합니다.

```
{
    "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
}
```


가져오기에 대한 검증 수준 구성

FHIR 가져오기 작업을 시작할 때 각 리소스`ValidationLevel`에 적용할를 선택적으로 지정할 수 있습니다.는 AWS HealthLake 현재 다음 검증 수준을 지원합니다.

- `strict`: 리소스는 리소스의 프로필 요소 또는 프로필이 없는 경우 R4 사양에 따라 검증됩니다. 이는의 기본 검증 수준입니다 AWS HealthLake.
- `structure-only`: 리소스는 R4에 대해 검증되며 참조된 프로파일은 무시합니다.
- `minimal`: 리소스는 특정 R4 규칙을 무시하면서 최소한으로 검증됩니다. 검색/분석에 필요한 구조 검사에 실패한 리소스는 감사 경고를 포함하도록 업데이트됩니다.

`minimal` 검증 수준을 사용하여 가져올 때 라는 폴더에 추가 로그 파일이 생성될 수 있습니다`SUCCESS_WITH_SEARCH_VALIDATION_FAILURES`. 검색 관련 검증 검사에 실패했지만이 폴더의 로그 파일 내 리소스가 데이터 스토어에 수집되었습니다. 이는 FHIR 리소스의 특정 측면이 FHIR에 따

라 유효하지 않았으며 잘못된 형식의 필드를 검색할 수 없음을 의미합니다. 이러한 리소스에는 이러한 실패를 설명하는이 `extension` 추가됩니다.

주제

FHIR 가져오기 작업 시작

`StartFHIRImportJob`를 사용하여 HealthLake 데이터 스토어로 FHIR 가져오기 작업을 시작합니다. 다음 메뉴에서는 AWS Management Console 및 AWS SDKs의 AWS CLI 및 코드 예제에 대한 절차를 제공합니다. 자세한 내용을 알아보려면 AWS HealthLake API 참조의 *StartFHIRImportJob* 섹션을 참조하세요.

> **중요**
>
> HealthLake는 의료 데이터 교환을 위한 FHIR R4 사양을 지원합니다. 필요한 경우 AWS HealthLake 파트너와 협력하여 가져오기 전에 상태 데이터를 FHIR R4 형식으로 변환할 수 있습니다.

FHIR 가져오기 작업을 시작하는 방법

액세스 기본 설정에 따라 메뉴를 선택합니다 AWS HealthLake.

AWS CLI 및 SDKs

CLI

AWS CLI

FHIR 가져오기 작업을 시작하는 방법

다음 `start-fhir-import-job` 예제에서는 AWS HealthLake를 사용하여 FHIR 가져오기 작업을 시작하는 방법을 보여줍니다.

```
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)"
```

출력:

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

자세한 내용은 AWS HealthLake 개발자 안내서의 FHIR 데이터 스토어에 파일 가져오기를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 StartFHIRImportJob 섹션을 참조하세요.

Python

SDK for Python(Boto3)

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

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


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

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

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 StartFHIRImportJob을 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

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

- API 세부 정보는 SDK for SAP ABAP API 참조의 StartFHIRImportJob을 참조하세요. AWS

> **예제 가용성**
>
> 필요한 예제를 찾을 수 없습니까? 이 페이지의 오른쪽 사이드바에 있는 피드백 제공 링크를 사용하여 코드 예제를 요청합니다.

AWS 콘솔

1. HealthLake 콘솔의 데이터 스토어 페이지에 로그인합니다.
2. 데이터 스토어를 선택합니다.
3. 가져오기를 선택합니다.

   가져오기 페이지가 열립니다.
4. 입력 데이터 섹션에 다음 정보를 입력합니다.

   - Amazon S3의 입력 데이터 위치
5. 출력 파일 가져오기 섹션에 다음 정보를 입력합니다.

   - Amazon S3에서 출력 파일 위치 가져오기

- 출력 파일 암호화 가져오기

6. 액세스 권한 섹션에서 기존 IAM 서비스 역할 사용을 선택하고 서비스 역할 이름 메뉴에서 역할을 선택하거나 IAM 역할 생성을 선택합니다.
7. 데이터 가져오기를 선택합니다.

> **Note**
>
> 가져오는 동안 페이지 상단의 배너에서 작업 ID 복사를 선택합니다. 를 사용하여 `JobID`를 사용하여 가져오기 작업 속성을 요청할 수 있습니다 AWS CLI. 자세한 내용은 FHIR 가져오기 작업 속성 가져오기 단원을 참조하십시오.

FHIR 가져오기 작업 속성 가져오기

`DescribeFHIRImportJob`를 사용하여 FHIR 가져오기 작업 속성을 가져옵니다. 다음 메뉴는 AWS Management Console 및 AWS SDKs의 AWS CLI 및 코드 예제에 대한 절차를 제공합니다. 자세한 내용을 알아보려면 AWS HealthLake API 참조의 *DescribeFHIRImportJob* 섹션을 참조하세요.

FHIR 가져오기 작업 속성을 가져오는 방법

액세스 기본 설정에 따라 메뉴를 선택합니다 AWS HealthLake.

AWS CLI 및 SDKs

CLI

AWS CLI

FHIR 가져오기 작업을 설명하는 방법

다음 `describe-fhir-import-job` 예제에서는 AWS HealthLake를 사용하여 FHIR 가져오기 작업의 속성을 학습하는 방법을 보여줍니다.

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

출력:

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

자세한 내용은 AWS HealthLake 개발자 안내서의 FHIR 데이터 스토어에 파일 가져오기를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DescribeFHIRImportJob 섹션을 참조하세요.

Python

SDK for Python(Boto3)

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

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


    def describe_fhir_import_job(
        self, datastore_id: str, job_id: str
    ) -> dict[str, any]:
        """
        Describes a HealthLake import job.
```

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

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 DescribeFHIRImportJob을 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        " iv_job_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
```

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

- API 세부 정보는 SDK for SAP ABAP API 참조의 DescribeFHIRImportJob을 참조하세요. AWS

> 예제 가용성
>
> 필요한 예제를 찾을 수 없습니까? 이 페이지의 오른쪽 사이드바에 있는 피드백 제공 링크를 사용하여 코드 예제를 요청합니다.

AWS 콘솔

> Note
>
> HealthLake 콘솔에서는 FHIR 가져오기 작업 정보를 사용할 수 없습니다. 대신와 함께를 AWS CLI 사용하여와 같은 가져오기 작업 속성을 DescribeFHIRImportJob 요청합니다JobStatus. 자세한 내용은이 페이지의 AWS CLI 예제를 참조하세요.

FHIR 가져오기 작업 나열

`ListFHIRImportJobs`를 사용하여 활성 HealthLake 데이터 스토어에 대한 FHIR 가져오기 작업을 나열합니다. 다음 메뉴는 AWS Management Console 및 SDKs의 AWS CLI 및 AWS 코드 예제에 대한 절차를 제공합니다. 자세한 내용을 알아보려면 AWS HealthLake API 참조의 *ListFHIRImportJobs* 섹션을 참조하세요.

FHIR 가져오기 작업을 나열하려면

액세스 기본 설정에 따라 메뉴를 선택합니다 AWS HealthLake.

AWS CLI 및 SDKs

CLI

AWS CLI

모든 FHIR 가져오기 작업을 나열하는 방법

다음 `list-fhir-import-jobs` 예제에서는 명령을 사용하여 계정과 연결된 모든 가져오기 작업 목록을 보는 방법을 보여 줍니다.

```
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)
```

출력:

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

자세한 내용은 AWS HealthLake 개발자 안내서의 FHIR 데이터 스토어로 파일 가져오기를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListFHIRImportJobs 섹션을 참조하세요.

Python

SDK for Python(Boto3)

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

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

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


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

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

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 ListFHIRImportJobs을 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

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

- API 세부 정보는 SDK for SAP ABAP API 참조의 ListFHIRImportJobs를 참조하세요. AWS

> **예제 가용성**
>
> 필요한 예제를 찾을 수 없습니까? 이 페이지의 오른쪽 사이드바에 있는 피드백 제공 링크를 사용하여 코드 예제를 요청합니다.

AWS 콘솔

> **Note**
>
> FHIR 가져오기 작업 정보는 HealthLake 콘솔에서 사용할 수 없습니다. 대신와 함께를 AWS CLI 사용하여 모든 FHIR 가져오기 작업을 나열`ListFHIRImportJobs`합니다. 자세한 내용은 이 페이지의 AWS CLI 예제를 참조하세요.

에서 FHIR 리소스 관리 AWS HealthLake

FHIR R4 RESTful API 상호 작용을 사용하여 HealthLake 데이터 스토어에서 FHIR 리소스를 관리합니다. 다음 섹션에서는 FHIR 리소스 관리에 사용할 수 있는 모든 HealthLake 지원 FHIR R4 RESTful API 상호 작용을 설명합니다. HealthLake 데이터 스토어 기능과 지원하는 FHIR 사양의 부분에 대한 자세한 내용은 섹션을 참조하세요에 대한 FHIR R4 기능 설명 AWS HealthLake.

> **Note**
>
> 이 장에 나열된 FHIR 상호 작용은 의료 데이터 교환을 위한 HL7 FHIR R4 표준을 준수하도록 구축되었습니다. HL7 FHIR 서비스의 표현이므로 AWS CLI 및 AWS SDKs 통해 제공되지 않습니다. 자세한 내용은 FHIR R4 RESTful API 설명서의 RESTful API를 참조하세요. R4 RESTful

다음 표에는에서 지원하는 FHIR R4 상호 작용이 나열되어 있습니다 AWS HealthLake. HealthLake에서 지원하는 FHIR 리소스 유형에 대한 자세한 내용은 섹션을 참조하세요조건 키.

에서 지원하는 FHIR R4 상호 작용 AWS HealthLake

상호 작용	설명
전체 시스템 상호 작용	
`capabilities`	시스템에 대한 기능 설명을 가져옵니다. 에 대한 FHIR R4 기능 설명 AWS HealthLake을(를) 참조하세요.
`batch`	단일 상호 작용에서 리소스 세트를 업데이트, 생성 또는 삭제합니다. FHIR 리소스 번들링을(를) 참조하세요.
유형 수준 상호 작용	
`create`	서버 할당 ID로 새 리소스를 생성합니다. FHIR 리소스 생성을(를) 참조하세요.
`search`	일부 필터 기준에 따라 리소스 유형을 검색합니다. FHIR 리소스 검색을(를) 참조하세요.
`history`	특정 리소스 유형에 대한 변경 기록을 검색합니다. FHIR 리소스 기록 읽기을(를) 참조하세요.

상호 작용	설명
인스턴스 수준 상호 작용	
read	리소스의 현재 상태를 읽습니다. FHIR 리소스 읽기을(를) 참조하세요.
history	특정 리소스에 대한 변경 기록을 읽습니다. FHIR 리소스 기록 읽기을(를) 참조하세요.
vread	리소스의 특정 버전 상태를 읽습니다. 버전별 FHIR 리소스 기록 읽기을(를) 참조하세요.
update	ID로 리소스를 업데이트합니다(또는 새 리소스인 경우 생성). FHIR 리소스 업데이트을(를) 참조하세요.
delete	리소스를 삭제합니다. FHIR 리소스 삭제을(를) 참조하세요.

주제

FHIR 리소스 생성

FHIR `create` 상호 작용은 HealthLake 데이터 스토어에 새 FHIR 리소스를 생성합니다. 자세한 내용은 FHIR R4 RESTful API 설명서`create`의 섹션을 참조하세요.

FHIR 리소스를 생성하려면

1. HealthLake `region` 및 `datastoreId` 값을 수집합니다. 자세한 내용은 데이터 스토어 속성 가져오기 단원을 참조하십시오.

2. Resource 생성할 FHIR 유형을 결정합니다. 자세한 내용은 조건 키 단원을 참조하십시오.

3. HealthLake region 및에 대해 수집된 값을 사용하여 요청에 대한 URL을 구성합니다datastoreId. 생성할 FHIR Resource 유형도 포함합니다. 다음 예제에서 전체 URL 경로를 보려면 복사 버튼을 스크롤합니다.

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

4. 요청에 대한 JSON 본문을 구성하여 새 리소스에 대한 FHIR 데이터를 지정합니다. 이 절차의 목적상 FHIR Patient 리소스를 사용하므로 파일을 로 저장합니다create-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. 요청을 보냅니다. FHIR create 상호 작용은 FHIR 권한 부여 시 AWS 서명 버전 4 또는 SMART와 함께 POST 요청을 사용합니다. 다음 예제에서는 curl 또는 HealthLake 콘솔을 사용하여 HealthLake에서 FHIR Patient 리소스를 생성합니다. 전체 예제를 보려면 복사 버튼을 스크롤합니다.

   SigV4

   SigV4 권한 부여

```
curl --request POST \
```

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

SMART on FHIR

IdentityProviderConfiguration 데이터 형식에 대한 SMART on FHIR 권한 부여 예제입니다.

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

호출자는 권한 부여 Lambda에서 권한을 할당할 수 있습니다. 자세한 내용은 OAuth 2.0 범위 단원을 참조하십시오.

AWS Console

> **Note**
>
> HealthLake 콘솔은 AWS SigV4 권한 부여만 지원합니다.

1. HealthLake 콘솔의 쿼리 실행 페이지에 로그인합니다.

2. 쿼리 설정 섹션에서 다음을 선택합니다.

- 데이터 스토어 ID - 쿼리 문자열을 생성할 데이터 스토어 ID를 선택합니다.
- 쿼리 유형 -를 선택합니다`Create`.
- 리소스 유형 - 생성할 FHIR 리소스 유형을 선택합니다.
- 요청 본문 - 새 리소스에 대한 FHIR 데이터를 지정하여 요청에 대한 JSON 본문을 구성합니다.

3. 쿼리 실행을 선택합니다.

리소스 생성을 위한 검증 수준 구성

FHIR 리소스를 생성할 때 선택적으로 `x-amzn-healthlake-fhir-validation-level` HTTP 헤더를 지정하여 리소스에 대한 검증 수준을 구성할 수 있습니다.는 AWS HealthLake 현재 다음 검증 수준을 지원합니다.

- `strict`: 리소스는 리소스의 프로필 요소 또는 프로필이 없는 경우 R4 사양에 따라 검증됩니다. 이는에 대한 기본 검증 수준입니다 AWS HealthLake.
- `structure-only`: 리소스는 R4에 대해 검증되며 참조된 프로파일은 무시합니다.
- `minimal`: 리소스는 특정 R4 규칙을 무시하면서 최소한으로 검증됩니다. 검색/분석에 필요한 구조 검사에 실패한 리소스는 감사 경고를 포함하도록 업데이트됩니다.

최소 검증 수준으로 생성된 리소스는 검색 인덱싱에 필요한 검증에 실패하더라도 데이터 스토어에 수집될 수 있습니다. 이 경우 해당 실패를 문서화하기 위한 Healthlake별 확장을 포함하도록 리소스가 업데이트됩니다.

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

또한 다음 HTTP 응답 헤더는 "true" 값과 함께 포함됩니다.

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

> **Note**
>
> R4 사양에 따라 잘못된 형식으로 수집된 데이터는 이러한 오류가 있는 경우 예상대로 검색하지 못할 수 있습니다.

FHIR 리소스 읽기

FHIR `read` 상호 작용은 HealthLake 데이터 스토어에서 리소스의 현재 상태를 읽습니다. 자세한 내용은 FHIR R4 RESTful API 설명서`read`의 섹션을 참조하세요.

FHIR 리소스를 읽으려면

1. HealthLake `region` 및 `datastoreId` 값을 수집합니다. 자세한 내용은 데이터 스토어 속성 가져오기 단원을 참조하십시오.
2. 연결된 `id` 값을 `Resource` 읽고 수집할 FHIR 유형을 결정합니다. 자세한 내용은 조건 키 단원을 참조하십시오.
3. HealthLake `region` 및에 대해 수집된 값을 사용하여 요청에 대한 URL을 구성합니다`datastoreId`. FHIR `Resource` 유형 및 관련 도 포함합니다`id`. 다음 예제에서 전체 URL 경로를 보려면 복사 버튼을 스크롤합니다.

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

4. 요청을 보냅니다. FHIR `read` 상호 작용은 FHIR 권한 부여 시 AWS 서명 버전 4 또는 SMART와 함께 `GET` 요청을 사용합니다. 다음 `curl` 예제에서는 HealthLake에서 FHIR `Patient` 리소스의 현재 상태를 읽습니다. 전체 예제를 보려면 복사 버튼을 스크롤합니다.

   SigV4

   SigV4 권한 부여

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

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

SMART on FHIR

IdentityProviderConfiguration 데이터 형식에 대한 SMART on FHIR 권한 부여 예제입니다.

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

호출자는 권한 부여 Lambda에 권한을 할당할 수 있습니다. 자세한 내용은 OAuth 2.0 범위 단원을 참조하십시오.

AWS Console

1. HealthLake 콘솔의 쿼리 실행 페이지에 로그인합니다.

2. 쿼리 설정 섹션에서 다음을 선택합니다.

- 데이터 스토어 ID - 쿼리 문자열을 생성할 데이터 스토어 ID를 선택합니다.
- 쿼리 유형 -를 선택합니다Read.
- 리소스 유형 - 읽을 FHIR 리소스 유형을 선택합니다.

- 리소스 ID - FHIR 리소스 ID를 입력합니다.

3. 쿼리 실행을 선택합니다.

FHIR 리소스 기록 읽기

FHIR `history` 상호 작용은 HealthLake 데이터 스토어에서 특정 FHIR 리소스의 기록을 검색합니다. 이 상호 작용을 사용하여 시간이 지남에 따라 FHIR 리소스의 내용이 어떻게 변경되었는지 확인할 수 있습니다. 또한 감사 로그와 협력하여 수정 전후의 리소스 상태를 확인하는 데 유용합니다. FHIR 상호 작용 `create`, `update`및를 `delete` 수행하면 리소스의 기록 버전이 저장됩니다. 자세한 내용은 FHIR R4 RESTful API 설명서`history`의 섹션을 참조하세요.

> **Note**
>
> 특정 FHIR 리소스 유형에 `history` 대해 옵트아웃할 수 있습니다. 옵트아웃하려면를 사용하여 사례를 생성합니다AWS Support Center Console. 사례를 생성하려면에 로그인 AWS 계정하고 사례 생성을 선택합니다.

FHIR 리소스 기록을 읽으려면

1. HealthLake `region` 및 `datastoreId` 값을 수집합니다. 자세한 내용은 데이터 스토어 속성 가져오기 단원을 참조하십시오.
2. 연결된 `id` 값을 `Resource` 읽고 수집할 FHIR 유형을 결정합니다. 자세한 내용은 조건 키 단원을 참조하십시오.
3. HealthLake `region` 및에 대해 수집된 값을 사용하여 요청에 대한 URL을 구성합니다`datastoreId`. FHIR `Resource` 유형, 연결된 및 `id`선택적 검색 파라미터도 포함합니다. 다음 예제에서 전체 URL 경로를 보려면 복사 버튼을 스크롤합니다.

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

FHIR **history** 상호 작용에 대해 HealthLake에서 지원하는 검색 파라미터

파라미터	설명
`_count : integer`	페이지의 최대 검색 결과 수입니다. 서버는 요청된 수 또는 데이터 스토어에 기본적으로 허용되는 최대 검색 결과 수 중 더 적은 수를 반환합니다.
`_since : instant`	지정된 시점에 또는 그 이후에 생성된 리소스 버전만 포함합니다.
`_at : date(Time)`	날짜 시간 값에 지정된 기간 동안 특정 시점에 최신 상태였던 리소스 버전만 포함합니다. 자세한 내용은 HL7 FHIR RESTful API 설명서date의 섹션을 참조하세요.

4. 요청을 보냅니다. FHIR `history` 상호 작용은 FHIR 권한 부여에서 AWS 서명 버전 4 또는 SMART와 함께 `GET` 요청을 사용합니다. 다음 `curl` 예제에서는 `_count` 검색 파라미터를 사용하여 HealthLake의 FHIR `Patient` 리소스에 대한 페이지당 100개의 과거 검색 결과를 반환합니다. 전체 예제를 보려면 복사 버튼을 스크롤합니다.

   SigV4

   SigV4 권한 부여

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

   SMART on FHIR

   IdentityProviderConfiguration 데이터 형식에 대한 SMART on FHIR 권한 부여 예제입니다.

   ```
   {
   ```

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

호출자는 권한 부여 Lambda에 권한을 할당할 수 있습니다. 자세한 내용은 OAuth 2.0 범위 단원을 참조하십시오.

history 상호 작용의 반환 콘텐츠는 Bundle유형이 로 설정된 FHIR 리소스에 포함되어 있습니다history. 지정된 버전 기록을 포함하고, 가장 오래된 버전을 마지막으로 정렬하며, 삭제된 리소스를 포함합니다. 자세한 내용은 FHIR R4 설명서Resource Bundle의 섹션을 참조하세요.

버전별 FHIR 리소스 기록 읽기

FHIR vread 상호 작용은 HealthLake 데이터 스토어에서 리소스의 버전별 읽기를 수행합니다. 이 상호 작용을 사용하면 과거 특정 시간에 있던 FHIR 리소스의 내용을 볼 수 있습니다.

> **Note**
>
> 없이 FHIR history 상호 작용을 사용하는 경우 vread HealthLake는 항상 리소스 메타데이터의 최신 버전을 반환합니다.

HealthLake는 지원되는 각 리소스에 대해에서 버전 관리를 지원한다고 선언CapabilityStatement.rest.resource.versioning합니다. 모든 HealthLake 데이터 스토어는 모든 리소스에 Resource.meta.versionId (vid)를 포함합니다.

FHIR history 상호 작용이 활성화된 경우(기본적으로 10/25/2024 이후에 생성된 데이터 스토어의 경우 또는 이전 데이터 스토어의 요청에 따라) Bundle 응답에는 location 요소의 vid 일부로가 포함됩니다. 다음 예제에서는 숫자 로 vid 표시됩니다1. 전체 예제를 보려면 예제 번들/번들 응답(JSON)을 참조하세요.

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

버전별 FHIR 리소스 기록을 읽으려면

1. HealthLake region 및 datastoreId 값을 수집합니다. 자세한 내용은 데이터 스토어 속성 가져오기 단원을 참조하십시오.
2. 연결된 및 id vid 값을 읽고 수집할 FHIR Resource 유형을 결정합니다. 자세한 내용은 조건 키 단원을 참조하십시오.
3. HealthLake 및 FHIR에 대해 수집된 값을 사용하여 요청에 대한 URL을 구성합니다. 다음 예제에서 전체 URL 경로를 보려면 복사 버튼을 스크롤합니다.

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

4. 요청을 보냅니다. FHIR history 상호 작용은 FHIR 권한 부여 시 AWS 서명 버전 4 또는 SMART와 함께 GET 요청을 사용합니다. 다음 vread 상호 작용은에서 지정한 Patient 리소스 메타데이터 버전의 FHIR 리소스에 지정된 콘텐츠가 있는 단일 인스턴스를 반환합니다vid. 다음 예제에서 전체 URL 경로를 보려면 복사 버튼을 스크롤합니다.

   SigV4

   SigV4 권한 부여

   ```
   curl --request GET \
     'https://healthlake.region.amazonaws.com/datastore/datastore-id/r4/Patient/id/
   _history/vid' \
   ```

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

SMART on FHIR

IdentityProviderConfiguration 데이터 형식에 대한 SMART on FHIR 권한 부여 예제입니다.

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

호출자는 권한 부여 Lambda에 권한을 할당할 수 있습니다. 자세한 내용은 OAuth 2.0 범위 단원을 참조하십시오.

FHIR 리소스 업데이트

FHIR update 상호 작용은 기존 리소스에 대한 새 현재 버전을 생성하거나 지정된에 대한 리소스가 아직 없는 경우 초기 버전을 생성합니다id. 자세한 내용은 FHIR R4 RESTful API 설명서update의 섹션을 참조하세요.

FHIR 리소스를 업데이트하려면

1. HealthLake region 및 datastoreId 값을 수집합니다. 자세한 내용은 데이터 스토어 속성 가져오기 단원을 참조하십시오.
2. 연결된 id 값을 Resource 업데이트하고 수집할 FHIR 유형을 결정합니다. 자세한 내용은 조건 키 단원을 참조하십시오.
3. HealthLake region 및에 대해 수집된 값을 사용하여 요청에 대한 URL을 구성합니다datastoreId. FHIR Resource 유형 및 관련 도 포함합니다id. 다음 예제에서 전체 URL 경로를 보려면 복사 버튼을 스크롤합니다.

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

4. 요청에 대한 JSON 본문을 구성하여 수행할 FHIR 데이터 업데이트를 지정합니다. 이 절차의 목적상 파일을 로 저장합니다update-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. 요청을 보냅니다. FHIR update 상호 작용은 FHIR 권한 부여에서 AWS 서명 버전 4 또는 SMART와 함께 PUT 요청을 사용합니다. 다음 curl 예시에서는 HealthLake에서 Patient 리소스를 업데이트합니다. 전체 예제를 보려면 복사 버튼을 스크롤합니다.

SigV4

SigV4 권한 부여

```
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
```

요청은 기존 리소스가 업데이트된 경우 200 HTTP 상태 코드를 반환하고 새 리소스가 생성된 경우 201 HTTP 상태 코드를 반환합니다.

SMART on FHIR

IdentityProviderConfiguration 데이터 형식에 대한 SMART on FHIR 권한 부여 예제입니다.

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

호출자는 권한 부여 Lambda에 권한을 할당할 수 있습니다. 자세한 내용은 OAuth 2.0 범위 단원을 참조하십시오.

AWS Console

1. HealthLake 콘솔의 쿼리 실행 페이지에 로그인합니다.

2. 쿼리 설정 섹션에서 다음을 선택합니다.

- 데이터 스토어 ID - 쿼리 문자열을 생성할 데이터 스토어 ID를 선택합니다.
- 쿼리 유형 -를 선택합니다`Update (PUT)`.
- 리소스 유형 - 업데이트하거나 생성할 FHIR 리소스 유형을 선택합니다.
- 요청 본문 - 리소스를 업데이트할 FHIR 데이터를 지정하여 요청에 대한 JSON 본문을 구성합니다.

3. 쿼리 실행을 선택합니다.

조건에 따라 FHIR 리소스 업데이트

조건부 업데이트를 사용하면 논리적 FHIR이 아닌 일부 식별 검색 기준에 따라 기존 리소스를 업데이트할 수 있습니다`id`. 서버는 업데이트를 처리할 때 요청에 대한 단일 논리를 해결하기 위해 리소스 유형에 `id` 대한 표준 검색 기능을 사용하여 검색을 수행합니다.

서버가 수행하는 작업은 검색한 일치 항목 수에 따라 달라집니다.

- 일치 항목 없음, 요청 본문에 **`id`** 제공되지 않음: 서버가 FHIR 리소스를 생성합니다.
- 일치 항목 없음, **`id`** 제공됨 및 리소스가에 아직 존재하지 않음**`id`**: 서버는 상호 작용을 업데이트로 상호 작용 생성을 처리합니다.
- 일치 항목 없음, **`id`** 제공됨 및 이미 존재: 서버가 `409 Conflict` 오류와 함께 업데이트를 거부합니다.
- 일치 항목 1개, 리소스**`id`**가 제공되지 않음 OR(리소스가 **`id`** 제공되고 찾은 리소스와 일치함): 서버는 위와 같이 일치하는 리소스에 대해 업데이트를 수행합니다. 리소스가 업데이트된 경우 서버는를 반환해야 합니다`200 OK`.
- One Match, 리소스가 **`id`** 제공되었지만 찾은 리소스와 일치하지 않음: 서버가 클라이언트 ID 사양이 가급적이면에서 문제가있음을 나타내는 `409 Conflict` 오류를 반환합니다. `OperationOutcome`

- 여러 일치 항목: 서버가 클라이언트의 기준이 가급적이면 OperationOutcome으로 충분히 선택되지 않았음을 나타내는 `412 Precondition Failed` 오류를 반환합니다.

다음 예제에서는 이름이 peter이고, 생년월일이 2000년 1월 1일이며, 전화번호가 1234567890인 `Patient` 리소스를 업데이트합니다.

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

리소스 업데이트에 대한 검증 수준 구성

FHIR 리소스를 업데이트할 때 선택적으로 `x-amzn-healthlake-fhir-validation-level` HTTP 헤더를 지정하여 리소스에 대한 검증 수준을 구성할 수 있습니다.는 AWS HealthLake 현재 다음 검증 수준을 지원합니다.

- `strict`: 리소스는 리소스의 프로필 요소 또는 프로필이 없는 경우 R4 사양에 따라 검증됩니다. 이는의 기본 검증 수준입니다 AWS HealthLake.
- `structure-only`: 리소스는 R4에 대해 검증되며 참조된 프로파일은 무시합니다.
- `minimal`: 리소스는 특정 R4 규칙을 무시하면서 최소한으로 검증됩니다. 검색/분석에 필요한 구조 검사에 실패한 리소스는 감사 경고를 포함하도록 업데이트됩니다.

최소 검증 수준으로 업데이트된 리소스는 검색 인덱싱에 필요한 검증에 실패하더라도 데이터 스토어에 수집될 수 있습니다. 이 경우 해당 실패를 문서화하기 위한 Healthlake별 확장을 포함하도록 리소스가 업데이트됩니다.

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

또한 다음 HTTP 응답 헤더는 "true" 값과 함께 포함됩니다.

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

> **Note**
>
> R4 사양에 따라 잘못된 형식으로 수집된 데이터는 이러한 오류가 있는 경우 예상대로 검색하지 못할 수 있습니다.

PATCH 작업을 사용하여 리소스 수정

AWS HealthLake 는 FHIR 리소스에 대한 PATCH 작업을 지원하므로 전체 리소스를 업데이트하지 않고 추가, 교체 또는 삭제할 특정 요소를 대상으로 하여 리소스를 수정할 수 있습니다. 이 작업은 다음이 필요한 경우에 특히 유용합니다.

- 대규모 리소스에 대한 대상 업데이트 수행
- 네트워크 대역폭 사용량 감소
- 특정 리소스 요소에 대한 원자성 수정 수행
- 동시 변경 사항 덮어쓰기 위험 최소화
- 배치 및 트랜잭션 워크플로의 일부로 리소스 업데이트

지원되는 패치 형식

AWS HealthLake 는 두 가지 표준 패치 형식을 지원합니다.

JSON 패치(RFC 6902)

JSON 포인터 구문을 사용하여 리소스 구조의 위치별로 요소를 대상으로 지정합니다.

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

FHIRPath 패치(FHIR R4 사양)

FHIRPath 표현식을 사용하여 콘텐츠 및 관계를 기준으로 요소를 대상으로 지정하여 패치 적용에 대한 FHIR 네이티브 접근 방식을 제공합니다.

Content-Type: `application/fhir+json`

사용법

직접 패치 작업

PATCH HTTP 메서드를 사용하여 FHIR 리소스에서 PATCH 작업을 직접 호출할 수 있습니다.

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

번들의 패치

패치 작업은 batch 또는 유형의 FHIR 번들 내에 항목으로 포함될 수 transaction있으므로 패치 작업을 단일 요청으로 다른 FHIR 상호 작용(생성, 읽기, 업데이트, 삭제)과 결합할 수 있습니다.

- 트랜잭션 번들: 모든 항목이 원자적으로 성공하거나 실패함
- 배치 번들: 각 항목은 독립적으로 처리됩니다.

JSON 패치 형식

지원되는 작업

연산	설명
add	리소스에 새 값 추가
remove	리소스에서 값 제거
replace	리소스의 기존 값 바꾸기
move	한 위치에서 값을 제거하고 다른 위치에 추가
copy	한 위치에서 다른 위치로 값 복사
test	대상 위치의 값이 지정된 값과 동일한지 테스트

경로 구문

JSON 패치는 JSON 포인터 구문(RFC 6901)을 사용합니다.

경로 예제	설명
/name/0/family	이름의 패밀리 요소
/telecom/-	통신 배열에 추가
/active	루트 수준 활성 요소
/address/0/line/1	첫 번째 주소의 두 번째 줄

예제

여러 작업을 사용한 직접 JSON 패치 요청

```
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"
  }
]
```

단일 작업으로 직접 JSON 패치 요청

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

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

번들의 JSON 패치

base64로 인코딩된 JSON 패치 페이로드를 포함하는 바이너리 리소스를 사용합니다.

```
{
  "resourceType": "Bundle",
  "type": "transaction",
  "entry": [{
    "resource": {
      "resourceType": "Binary",
      "contentType": "application/json-patch+json",
      "data":
 "W3sib3AiOiJhZGQiLCJwYXRoIjoiL2JpcnRoRGF0ZSIsInZhbHVlIjoiMTk5MC0wMS0wMSJ9XQ=="
    },
```

```
    "request": {
      "method": "PATCH",
      "url": "Patient/123"
    }
  }]
}
```

FHIRPath 패치 형식

지원되는 작업

연산	설명
add	리소스에 새 요소 추가
insert	목록의 특정 위치에 요소 삽입
delete	리소스에서 요소 제거
replace	기존 요소의 값 바꾸기
move	목록 내 요소 재정렬

경로 구문

FHIRPath 패치는 FHIRPath 표현식을 사용하여 다음을 지원합니다.

- 인덱스 기반 액세스: `Patient.name[0]`
- 를 사용한 필터링**`where()`**: `Patient.name.where(use = 'official')`
- 부울 로직: `Patient.telecom.where(system = 'phone' and use = 'work')`
- 함수 하위 설정: `first()`, `last()`
- 존재 확인: `exists()`, `count()`
- 다형성 탐색: `Observation.value`

예제

직접 FHIRPath 패치 요청

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

번들의 FHIRPath 패치

파라미터 리소스를의 항목 리소스로 사용합니다`method: PATCH`.

```
{
  "resourceType": "Bundle",
  "type": "transaction",
  "entry": [{
    "resource": {
      "resourceType": "Parameters",
      "parameter": [{
        "name": "operation",
        "part": [
          { "name": "type", "valueCode": "add" },
          { "name": "path", "valueString": "Patient" },
          { "name": "name", "valueString": "birthDate" },
          { "name": "value", "valueDate": "1990-01-01" }
        ]
      }]
    },
    "request": {
      "method": "PATCH",
      "url": "Patient/123"
    }
  }]
```

```
}
```

요청 헤더

헤더	필수	설명
Content-Type	예	application/json-patch+json JSON 패치의 경우 또는 FHIRPath 패치application/fhir+json 의 경우
If-Match	아니요	ETag를 사용한 버전별 조건부 업데이트

샘플 응답

작업은 새 버전 정보와 함께 업데이트된 리소스를 반환합니다.

```
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"
  }
}
```

동작

PATCH 작업:

- 적절한 사양에 따라 패치 구문을 검증합니다(JSON 패치의 경우 RFC 6902, FHIRPath 패치의 경우 FHIRPath R4).
- 원자적으로 작업 적용 - 모든 작업이 성공하거나 모두 실패함
- 리소스 버전 ID를 업데이트하고 새 기록 항목을 생성합니다.
- 변경 사항을 적용하기 전에 기록에서 원래 리소스를 보존합니다.
- 패치 적용 후 FHIR 리소스 제약 조건 검증
- ETag와 If-Match 헤더를 사용한 조건부 업데이트 지원

오류 처리

작업은 다음 오류 조건을 처리합니다.

- 400 잘못된 요청: 잘못된 패치 구문(부적합 요청 또는 잘못된 패치 문서)
- 404 찾을 수 없음: 리소스를 찾을 수 없음(지정된 ID가 존재하지 않음)
- 409 충돌: 버전 충돌(동시 업데이트 또는 비최신 버전 ID 제공)
- 422 처리 불가능한 엔터티: 지정된 리소스 요소에 패치 작업을 적용할 수 없습니다.

기능 요약

기능	JSON 패치	FHIRPath 패치
콘텐츠 유형	`application/json-patch+json`	`application/fhir+json`
경로 형식	JSON 포인터(RFC 6901)	FHIRPath 표현식
직접 패치 API	지원됨	지원됨

기능	JSON 패치	FHIRPath 패치
번들 배치	지원됨(바이너리를 통해)	지원됨(파라미터를 통해)
번들 트랜잭션	지원됨(바이너리를 통해)	지원됨(파라미터를 통해)
운영	추가, 제거, 교체, 이동, 복사, 테스트	추가, 삽입, 삭제, 교체, 이동

제한 사항

- 검색 조건을 사용하는 조건부 PATCH 작업은 지원되지 않습니다.
- 번들의 JSON 패치는 base64 인코딩 콘텐츠와 함께 바이너리 리소스를 사용해야 합니다.
- 번들의 FHIRPath 패치는 파라미터 리소스를 사용해야 합니다.

추가 리소스

PATCH 작업에 대한 자세한 내용은 다음을 참조하세요.

- FHIR R4 패치 설명서
- FHIR R4 FHIRPath 패치 사양
- RFC 6902 - JSON 패치
- RFC 6901 - JSON 포인터

FHIR 리소스 번들링

FHIRBundle은의 FHIR 리소스 모음을 위한 컨테이너입니다 AWS HealthLake.는 서로 다른 처리 동작을 가진 두 가지 유형의 번들을 AWS HealthLake 지원합니다.

Batch 번들은 각 리소스를 독립적으로 처리합니다. 리소스 하나가 실패하더라도 나머지 리소스는 여전히 성공할 수 있습니다. 각 작업은 개별적으로 처리되며 일부 작업이 실패하더라도 처리가 계속됩니다. 관련 없는 여러 환자 레코드를 업로드하는 등 부분적 성공이 허용되는 대량 작업에 배치 번들을 사용합니다.

`Transaction` 번들은 모든 리소스를 원자적으로 단일 단위로 처리합니다. 모든 리소스 작업이 성공하거나 커 AWS HealthLake 밋하지 않습니다. 모든 데이터를 함께 기록해야 하는 관련 관찰 및 조건이 있는 환자 생성과 같이 관련 리소스에서 참조 무결성을 보장해야 하는 경우 트랜잭션 번들을 사용합니다.

배치 번들과 트랜잭션 번들의 차이점

기능	배치	트랜잭션
처리 모델	각 작업은 독립적으로 성공하거나 실패합니다.	모든 작업은 단일 원자 단위로 성공하거나 실패합니다.
장애 처리	개별 작업이 실패하더라도 처리는 계속됩니다.	단일 작업이 실패하면 전체 번들이 실패합니다.
실행 순서	실행 순서는 보장되지 않습니다.	작업은 지정된 순서대로 처리됩니다.
참조 무결성	작업 간에 적용되지 않습니다.	번들 내에서 로컬로 참조되는 리소스에 적용됩니다.
가장 적합한 용도	부분적 성공이 허용되는 대량 작업.	함께 생성하거나 업데이트해야 하는 관련 리소스입니다.

동일하거나 다른 유형의 FHIR 리소스를 번들링할 수 있으며, `create`, , `read`, 등의 FHIR 작업을 혼합하여 포함할 수 `update` `delete`있습니다`patch`. 자세한 내용은 FHIR R4 설명서의 리소스 번들을 참조하세요.

다음은 각 번들 유형에 대한 사용 사례의 예입니다.

배치 번들

- 야간 데이터 동기화 중에 서로 다른 시설에서 관련 없는 여러 환자 레코드를 업로드합니다.
- 일부 레코드에 검증 문제가 있을 수 있는 과거 약물 레코드를 대량 업로드합니다.
- 개별 실패가 다른 항목에 영향을 미치지 않는 조직 및 실무자와 같은 참조 데이터를 로드합니다.

트랜잭션 번들

- 모든 데이터를 함께 기록해야 하는 응급 부서 승인 중에 관련 관찰 및 조건이 있는 환자를 생성합니다.
- 환자의 약물 목록과 일관성을 유지해야 하는 관련 알레르기 정보를 업데이트합니다.
- 환자, 관찰, 절차 및 결제 정보와의 완전한 접촉을 단일 원자 단위로 기록합니다.

> **Important**
>
> 배치 번들과 트랜잭션 번들 모두 동일한 `Bundle` 리소스 구조를 사용합니다. 유일한 차이점은 `type` 필드의 값입니다.

다음 예제에서는 여러 리소스 유형 및 작업이 있는 트랜잭션 번들을 보여줍니다.

```
{
  "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"
    }
  }
```

```
  ]
}
```

FHIR 리소스를 독립 엔터티로 번들링

FHIR 리소스를 독립 엔터티로 번들링하려면

1. HealthLake `region` 및 `datastoreId` 값을 수집합니다. 자세한 내용은 데이터 스토어 속성 가져오기 단원을 참조하십시오.
2. HealthLake `region` 및에 대해 수집된 값을 사용하여 요청에 대한 URL을 구성합니다`datastoreId`. URL에 FHIR 리소스 유형을 지정하지 마십시오. 다음 예제에서 전체 URL 경로를 보려면 복사 버튼을 스크롤합니다.

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

3. 각 HTTP 동사를 `method` 요소의 일부로 지정하여 요청에 대한 JSON 본문을 구성합니다. 다음 예제에서는 `Bundle` 리소스와의 `batch` 유형 상호 작용을 사용하여 새 `Patient` 및 `Medication` 리소스를 생성합니다. 그에 따라 모든 필수 섹션에 주석이 추가됩니다. 이 절차의 목적상 파일을로 저장합니다`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. 요청을 보냅니다. FHIR Bundle 배치 유형은 FHIR 권한 부여 시 AWS 서명 버전 4 또는 SMART로 POST 요청을 사용합니다. 다음 코드 예제에서는 데모용으로 curl 명령줄 도구를 사용합니다.

SigV4

SigV4 권한 부여

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

SMART on FHIR

IdentityProviderConfiguration 데이터 형식에 대한 SMART on FHIR 권한 부여 예제입니다.

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

호출자는 권한 부여 Lambda에 권한을 할당할 수 있습니다. 자세한 내용은 OAuth 2.0 범위 단원을 참조하십시오.

서버는 `Bundle` 배치 유형 요청의 결과로 생성된 `Patient` 및 `Medication` 리소스를 보여주는 응답을 반환합니다.

번들의 조건부 PUTs

AWS HealthLake 는 다음 쿼리 파라미터를 사용하여 번들 내에서 조건부 업데이트를 지원합니다.

- `_id` (독립 실행형)
- `_id` 다음 중 하나와 함께 사용할 수 있습니다.
  - `_tag`
  - `_createdAt`
  - `_lastUpdated`

번들에서 조건부 PUTs 사용하는 경우는 쿼리 파라미터를 기존 리소스와 비교하여 AWS HealthLake 평가하고 일치 결과를 기반으로 조치를 취합니다.

조건부 업데이트 동작

시나리오	HTTP 상태	취해진 조치
ID가 제공되지 않은 리소스	201 생성됨	는 항상 새 리소스를 생성합니다.
새 ID가 있는 리소스(일치하지 않음)	201 생성됨	지정된 ID로 새 리소스를 생성합니다.
기존 ID가 있는 리소스(단일 일치)	200 OK	일치하는 리소스를 업데이트합니다.
기존 ID가 있는 리소스(충돌 감지됨)	409 충돌	오류를 반환합니다. 변경 사항이 없습니다.
기존 ID가 있는 리소스(ID 불일치)	400 잘못된 요청	오류를 반환합니다. 변경 사항이 없습니다.
여러 리소스 일치 조건	412 사전 조건 실패	오류를 반환합니다. 변경 사항이 없습니다.

조건부 업데이트가 포함된 다음 번들 예제에서는 조건이 충족되는 경우에만 FHIR ID가 인 Patient 리소스_lastUpdated=lt2025-04-20가 476 업데이트됩니다.

```
{
    "resourceType": "Bundle",
    "id": "bundle-batch",
    "meta": {
        "lastUpdated": "2014-08-18T01:43:30Z"
    },
    "type": "batch",
    "entry": [
        {
            "resource": {
                "resourceType": "Patient",
                "id": "476",
                "meta": {
                    "lastUpdated": "2022-06-03T17:53:36.724Z"
                },
                "active": true,
                "name": [
                    {
                        "use": "official",
                        "family": "Jackson",
                        "given": [
                            "Mateo",
                            "James"
                        ]
                    }
                ],
                "gender": "male",
                "birthDate": "1974-12-25"
            },
            "request": {
                "method": "PUT",
                "url": "Patient?_id=476&_lastUpdated=lt2025-04-20"
            }
        },
        {
            "resource": {
                "resourceType": "Medication",
                "id": "med0310",
                "contained": [
                    {
```

```
                "resourceType": "Substance",
                "id": "sub03",
                "code": {
                    "coding": [
                        {
                            "system": "http://snomed.info/sct",
                            "code": "55452001",
                            "display": "Oxycodone (substance)"
                        }
                    ]
                }
            }
        ],
        "code": {
            "coding": [
                {
                    "system": "http://snomed.info/sct",
                    "code": "430127000",
                    "display": "Oral Form Oxycodone (product)"
                }
            ]
        },
        "form": {
            "coding": [
                {
                    "system": "http://snomed.info/sct",
                    "code": "385055001",
                    "display": "Tablet dose form (qualifier value)"
                }
            ]
        },
        "ingredient": [
            {
                "itemReference": {
                    "reference": "#sub03"
                },
                "strength": {
                    "numerator": {
                        "value": 5,
                        "system": "http://unitsofmeasure.org",
                        "code": "mg"
                    },
                    "denominator": {
                        "value": 1,
```

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

FHIR 리소스를 단일 엔터티로 번들링

FHIR 리소스를 단일 엔터티로 번들링하려면

1. HealthLake `region` 및 `datastoreId` 값을 수집합니다. 자세한 내용은 데이터 스토어 속성 가져오기 단원을 참조하십시오.
2. HealthLake `region` 및에 대해 수집된 값을 사용하여 요청에 대한 URL을 구성합니다.`datastoreId`. FHIR 리소스 유형을 URL의 `Bundle` 일부로 포함합니다. 다음 예제에서 전체 URL 경로를 보려면 복사 버튼을 스크롤합니다.

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

3. 요청에 대한 JSON 본문을 구성하여 그룹화할 FHIR 리소스를 지정합니다. 다음 예제에서는 HealthLake에서 두 개의 `Patient` 리소스를 그룹화합니다. 이 절차의 목적상 파일을 로 저장합니다.`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. 요청을 보냅니다. FHIR Bundle 문서 유형은 AWS 서명 버전 4 서명 프로토콜과 함께 POST 요청을 사용합니다. 다음 코드 예제에서는 데모용으로 curl 명령줄 도구를 사용합니다.

   SigV4

   SigV4 권한 부여

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

   SMART on FHIR

   IdentityProviderConfiguration 데이터 형식에 대한 SMART on FHIR 권한 부여 예제입니다.

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

호출자는 권한 부여 Lambda에 권한을 할당할 수 있습니다. 자세한 내용은 OAuth 2.0 범위 단원을 참조하십시오.

서버는 Bundle 문서 유형 요청의 결과로 생성된 두 개의 Patient 리소스를 보여주는 응답을 반환합니다.

번들에 대한 검증 수준 구성

FHIR 리소스를 번들링할 때 선택적으로 x-amzn-healthlake-fhir-validation-level HTTP 헤더를 지정하여 리소스에 대한 검증 수준을 구성할 수 있습니다. 이 검증 수준은 번들 내의 모든 생성 및 업데이트 요청에 대해 설정됩니다.는 AWS HealthLake 현재 다음 검증 수준을 지원합니다.

- strict: 리소스는 리소스의 프로필 요소 또는 프로필이 없는 경우 R4 사양에 따라 검증됩니다. 이는의 기본 검증 수준입니다 AWS HealthLake.
- structure-only: 리소스는 R4에 대해 검증되며 참조된 프로파일은 무시합니다.
- minimal: 리소스는 특정 R4 규칙을 무시하면서 최소한으로 검증됩니다. 검색/분석에 필요한 구조 검사에 실패한 리소스는 감사 경고를 포함하도록 업데이트됩니다.

최소 검증 수준으로 번들링된 리소스는 검색 인덱싱에 필요한 검증에 실패하더라도 데이터 스토어에 수집될 수 있습니다. 이 경우 리소스는 해당 실패를 문서화하기 위한 Healthlake별 확장을 포함하도록 업데이트되며 번들 응답 내의 항목에는 다음과 같이 OperationOutcome 리소스가 포함됩니다.


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

또한 다음 HTTP 응답 헤더는 "true" 값과 함께 포함됩니다.

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

> **Note**
>
> R4 사양에 따라 잘못된 형식으로 수집된 데이터는 이러한 오류가 있는 경우 예상대로 검색하지 못할 수 있습니다.

번들 유형 "메시지"에 대한 제한된 지원

HealthLake는 내부 변환 프로세스를 `message` 통해 FHIR 번들 유형에 대한 제한된 지원을 제공합니다. 이 지원은 레거시 병원 시스템에서 ADT(Admission, Discharge, Transfer) 피드를 수집하는 등 소스에서 메시지 번들을 다시 포맷할 수 없는 시나리오를 위해 설계되었습니다.

> **Warning**
>
> 이 기능은 명시적 AWS 계정 허용 목록이 필요하며 FHIR R4 메시지 의미 체계 또는 참조 무결성을 적용하지 않습니다. 메시지 번들을 사용하기 전에 계정에 대한 활성화를 요청하려면 AWS Support에 문의하세요.

표준 메시지 처리의 주요 차이점

- 메시지 번들(FHIR 사양): 첫 번째 항목은 다른 리소스를 참조MessageHeader하는 이어야 합니다. 리소스에는 개별 request 객체가 없으며 MessageHeader 이벤트에 따라 처리 작업이 결정됩니다.
- HealthLake 처리: 각 리소스 항목에 PUT 작업을 자동으로 할당하여 메시지 번들을 배치 번들로 변환합니다. 리소스는 메시지 의미 체계 또는 참조 무결성을 적용하지 않고 독립적으로 처리됩니다.

중요 제한 사항

- FHIR R4 메시지별 처리 규칙이 적용되지 않음
- 리소스 간 트랜잭션 무결성 없음
- 리소스 간 참조가 검증되지 않음
- 명시적 계정 허용 목록 지정 필요

예제 메시지 번들 구조

```
{
              "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**
>
> 각 리소스는 개별 PUT 작업을 통해 제출된 것처럼 독립적으로 저장됩니다. 전체 FHIR 메시징 의미 체계 또는 참조 무결성 검증이 필요한 경우 제출 전에 메시지 번들을 사전 처리하거나 애플리케이션 수준 검증을 구현합니다.

비동기 번들 트랜잭션

AWS HealthLake 는 최대 500개의 리소스가 있는 트랜잭션을 제출할 수 `transaction` 있는 비동기 `Bundle` 유형을 지원합니다. 비동기 트랜잭션을 제출하면 HealthLake는 처리를 위해 이를 대기열에 넣고 즉시 폴링 URL을 반환합니다. 이 URL을 사용하여 상태를 확인하고 응답을 검색할 수 있습니다. 이는 FHIR 비동기 번들 패턴을 따릅니다.

비동기 트랜잭션을 사용해야 하는 경우

- 단일 트랜잭션에서 100개 이상의 리소스(동기 제한)를 제출해야 합니다.
- 트랜잭션 처리가 완료될 때까지 기다리는 동안 애플리케이션을 차단하지 않으려고 합니다.
- 처리량을 높이려면 대량의 관련 리소스를 처리해야 합니다.

> **Important**
>
> 폴링 결과는 트랜잭션이 완료된 후 90일 동안 사용할 수 있습니다. 이 90일 기간이 지나면 폴링 URL이 더 이상 결과를 반환하지 않습니다. 이 기간 내에 결과를 검색하고 저장하도록 통합을 설계합니다.

> **Note**
>
> 동기`Bundle`식 유형은 최대 100개의 리소스를 `transaction` 계속 지원하며 기본 처리 모드입니다. `Prefer: respond-async` 헤더 없이 리소스`transaction`가 100개 이상인 `Bundle` 유형을 제출하면 HealthLake가 `422 Unprocessable Entity` 오류를 반환합니다. 유형의 번들`batch`은 비동기 처리에 지원되지 않습니다. `Bundle` 유형만 비동기적으로 제출할 수 `transaction` 있습니다(최대 500개의 작업 사용).

비동기 트랜잭션 제출

비동기 트랜잭션을 제출하려면 `Prefer: respond-async` 헤더를 사용하여 데이터 스토어 엔드포인트에 POST 요청을 보냅니다. 번들에는 유형이 있어야 합니다`transaction`. 유형의 번들`batch`은 비동기 번들 처리에 지원되지 않습니다.

HealthLake는 제출 시 번들에 대한 초기 검증을 수행합니다. 검증에 성공하면 HealthLake는 폴링 URL이 포함된 `content-location` 응답 헤더와 함께 HTTP 202 Accepted를 반환합니다.

비동기 **`Bundle`** 유형을 제출하려면 **`transaction`**

1. HealthLake 데이터 스토어 엔드포인트에 POST 요청을 보냅니다.

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

2. 번들 유형이 인 요청에 대한 JSON 본문을 구성합니다`transaction`. 이 절차의 목적상 파일을 로 저장합니다`async-transaction.json`.

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

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

3. `Prefer: respond-async` 헤더와 함께 요청을 전송합니다. FHIR `Bundle` 트랜잭션 유형은 FHIR 권한 부여 시 AWS 서명 버전 4 또는 SMART인 `POST` 요청을 사용합니다. 다음 코드 예제에서는 데모용으로 `curl` 명령줄 도구를 사용합니다.

   SigV4

   SigV4 권한 부여

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

SMART on FHIR

IdentityProviderConfiguration 데이터 형식에 대한 SMART on FHIR 권한 부여 예제입니다.

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

호출자는 권한 부여 Lambda에 권한을 할당할 수 있습니다. 자세한 내용은 OAuth 2.0 범위 단원을 참조하십시오.

4. 제출에 성공하면 서버는 HTTP 202 Accepted를 반환합니다. `content-location` 응답 헤더에는 폴링 URL이 포함되어 있습니다. 응답 본문은 `OperationOutcome` 리소스입니다.

```
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"
            ]
        }
    ]
}
```

트랜잭션 상태에 대한 폴링

비동기 트랜잭션을 제출한 후 `content-location` 응답 헤더의 폴링 URL을 사용하여 트랜잭션 상태를 확인합니다. 폴링 URL에 `GET` 요청을 보냅니다.

> **Note**
>
> FHIR 지원 데이터 스토어의 SMART의 경우, 권한 부여 토큰에는 트랜잭션 상태를 폴링할 `Transaction` 리소스 유형에 대한 `read` 권한이 포함되어야 합니다. FHIR 범위의 SMART에 대한 자세한 내용은 섹션을 참조하세요HealthLake에서 지원하는 FHIR OAuth 2.0 범위의 SMART.

폴링 URL에 `GET` 요청을 보냅니다. 다음 예제에서는 `curl` 명령줄 도구를 사용합니다.

SigV4

SigV4 권한 부여

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

SMART on FHIR

FHIR 권한 부여에 대한 SMART. 권한 부여 토큰에는 `Transaction` 리소스 유형에 대한 `read` 권한이 포함되어야 합니다.

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

다음 표에서는 가능한 응답을 설명합니다.

응답 코드 폴링

HTTP 상태	의미	응답 본문
202 수락됨	트랜잭션이 대기열에 있음	OperationOutcome "SUBMITTED" 진단 포함
202 수락됨	트랜잭션 처리 중	OperationOutcome 진단 "IN_PROGRESS" 사용
200 OK	트랜잭션이 성공적으로 완료되었습니다.	Bundle 유형 포함 transaction-response
4xx/5xx	트랜잭션 실패	OperationOutcome 오류 세부 정보가 있는

다음 예제에서는 각 응답 유형을 보여줍니다.

트랜잭션 대기 중(202)

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

트랜잭션 처리(202)

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

트랜잭션 완료(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"
            }
        }
    ]
}
```

트랜잭션 실패(4xx/5xx)

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

주문 처리

유형의 비동기 번들`transaction`은 대기열에 있지만 엄격한 제출 순서로 처리되지 않습니다. HealthLake는 사용 가능한 용량 및 시스템 부하를 기반으로 처리를 최적화합니다.

> **Important**
>
> 제출된 순서대로 처리 중인 트랜잭션에 의존하지 마십시오. 예를 들어 오전 10시에 트랜잭션 A를 제출하고 오전 10시 1분에 트랜잭션 B를 제출하는 경우 트랜잭션 B는 트랜잭션 A 이전에 완료될 수 있습니다. 애플리케이션을 다음과 같이 설계하세요.
>
> - out-of-order 처리합니다.
> - 폴링 URL을 사용하여 각 트랜잭션을 독립적으로 추적합니다.
> - 사용 사례에 주문이 중요한 경우 애플리케이션 수준 시퀀싱을 구현합니다.

할당량 및 제한

다음 할당량 및 속도 제한은 비동기 트랜잭션에 적용됩니다.

비동기 트랜잭션 할당량

할당량	값	조정 가능
비동기 트랜잭션당 최대 작업 수	500	아니요
데이터 스토어당 최대 보류 중인 트랜잭션 수	500	예

- 비동기 트랜잭션은에 정의된 것과 동일한 API 속도 제한을 공유합니다Service Quotas.
- 트랜잭션 상태에 대한 폴링은 FHIR 리소스에 대한 읽기(GET) 작업과 동일한 API 속도 제한을 공유합니다.
- 보류 중인 트랜잭션 한도에 도달하면 기존 트랜잭션이 완료될 때까지 후속 제출에서 오류가 반환됩니다.

오류 처리

'트랜잭션' 번들의 경우 번들에 포함된 모든 FHIR 리소스는 원자성 작업으로 처리됩니다. 작업의 모든 리소스가 성공하거나 번들의 작업이 처리되지 않아야 합니다.

오류는 HealthLake가 동기적으로 반환하는 제출 오류와 폴링을 통해 검색하는 처리 오류의 두 가지 범주로 나뉩니다.

제출 오류

HealthLake는 제출 시 번들을 검증하고 트랜잭션이 대기열에 추가되기 전에 동기식으로 오류를 반환합니다. 제출 오류에는 잘못된 FHIR 리소스 검증 오류, 지원되지 않는 리소스 유형, 500개의 작업 제한 초과, 배치 번들과 함께 `Prefer: respond-async` 헤더 사용 등이 포함됩니다. 데이터 스토어에 대해 보류 중인 트랜잭션 한도에 도달하면 HealthLake는를 반환합니다`ThrottlingException`. 제출 오류가 발생하면 트랜잭션이 대기열에 추가되지 않습니다.

처리 오류

처리 오류는 트랜잭션이 대기열에 추가되고 폴링 URL을 통해 반환된 후에 발생합니다. 여기에는 다른 작업이 트랜잭션의 일부인 리소스를 수정한 트랜잭션 충돌과 처리 중 서버 오류가 포함됩니다. 처리 오류가 발생하면 트랜잭션의 리소스에 대해 리소스 변형이 수행되지 않습니다. 폴링 URL은 오류 세부 정보와 `OperationOutcome` 함께를 반환합니다.

FHIR 리소스 삭제

FHIR `delete` 상호 작용은 HealthLake 데이터 스토어에서 기존 FHIR 리소스를 제거합니다. 자세한 내용은 FHIR R4 RESTful API 설명서`delete`의 섹션을 참조하세요.

FHIR 리소스를 삭제하려면

1. HealthLake `region` 및 `datastoreId` 값을 수집합니다. 자세한 내용은 데이터 스토어 속성 가져오기 단원을 참조하십시오.

2. 연결된 id 값을 Resource 삭제하고 수집할 FHIR 유형을 결정합니다. 자세한 내용은 조건 키 단원을 참조하십시오.

3. HealthLake region 및에 대해 수집된 값을 사용하여 요청에 대한 URL을 구성합니다datastoreId. FHIR Resource 유형 및 관련 도 포함합니다id. 다음 예제에서 전체 URL 경로를 보려면 복사 버튼을 스크롤합니다.

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

4. 요청을 보냅니다. FHIR delete 상호 작용은 FHIR 권한 부여 시 AWS 서명 버전 4 또는 SMART와 함께 DELETE 요청을 사용합니다. 다음 curl 예시에서는 HealthLake 데이터 스토어에서 기존 FHIR Patient 리소스를 제거합니다. 전체 예제를 보려면 복사 버튼을 스크롤합니다.

   SigV4

   SigV4 권한 부여

```
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'
```

   서버는 리소스가 HealthLake 데이터 스토어에서 제거되었음을 확인하는 204 HTTP 상태 코드를 반환합니다. 삭제 요청이 실패하면 요청이 실패한 이유를 나타내는 400 일련의 HTTP 상태 코드를 받게 됩니다.

   SMART on FHIR

   IdentityProviderConfiguration 데이터 형식에 대한 SMART on FHIR 권한 부여 예제입니다.

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

호출자는 권한 부여 Lambda에서 권한을 할당할 수 있습니다. 자세한 내용은 OAuth 2.0 범위 단원을 참조하십시오.

AWS Console

1. HealthLake 콘솔의 쿼리 실행 페이지에 로그인합니다.
2. 쿼리 설정 섹션에서 다음을 선택합니다.

- 데이터 스토어 ID - 쿼리 문자열을 생성할 데이터 스토어 ID를 선택합니다.
- 쿼리 유형 -를 선택합니다`Delete`.
- 리소스 유형 - 삭제할 FHIR 리소스 유형을 선택합니다.
- 리소스 ID - FHIR 리소스 ID를 입력합니다.

3. 쿼리 실행을 선택합니다.

조건에 따라 FHIR 리소스 삭제

조건부 삭제는 특정 FHIR 리소스 ID를 모르지만 삭제하려는 리소스에 대한 다른 식별 정보가 있는 경우에 특히 유용합니다.

조건부 삭제를 사용하면 논리적 FHIR ID가 아닌 검색 기준에 따라 기존 리소스를 삭제할 수 있습니다. 서버는 삭제 요청을 처리할 때 리소스 유형에 대한 표준 검색 기능을 사용하여 검색을 수행하여 요청에 대한 단일 논리적 ID를 확인합니다.

조건부 삭제 작동 방식

서버의 작업은 검색한 일치 항목 수에 따라 달라집니다.

1. 일치 항목 없음: 서버가 일반 삭제를 시도하고 적절하게 응답합니다(404 존재하지 않는 리소스의 경우 찾을 수 없음, 204 이미 삭제된 리소스의 경우 콘텐츠 없음).
2. 일치하는 항목 하나: 서버가 일치하는 리소스에서 일반 삭제를 수행합니다.
3. 여러 일치 항목: 클라이언트의 기준이 충분히 선택적이지 않음을 나타내는 412 사전 조건 실패 오류를 반환합니다.

대응 시나리오

AWS HealthLake 는 다음과 같은 응답 패턴으로 조건부 삭제 작업을 처리합니다.

성공한 작업

- 검색 기준이 단일 활성 리소스를 성공적으로 식별하면 시스템은 표준 삭제 작업과 마찬가지로 삭제를 완료한 후 204 콘텐츠 없음을 반환합니다.

ID 기반 조건부 삭제

추가 파라미터(`createdAt`, `tag`또는)`id`를 사용하여를 기반으로 조건부 삭제를 수행하는 경우`_lastUpdated`:

- 204 콘텐츠 없음: 리소스가 이미 삭제되었습니다.
- 404 찾을 수 없음: 리소스가 존재하지 않음
- 409 충돌: ID는 일치하지만 다른 파라미터는 일치하지 않음

Non-ID-Based 조건부 삭제

`id`가 제공되지 않거나 `createdAt`, 또는 이외의 파라미터를 사용하는 경우`tag_lastUpdated`:

- 404 찾을 수 없음: 일치하는 항목을 찾을 수 없음

충돌 상황

여러 시나리오에서 412개의 사전 조건 실패 응답이 발생합니다.

- 여러 리소스가 검색 기준과 일치(기준이 충분히 구체적이지 않음)

- 에서 ETag 헤더 사용 시 버전 충돌 `If-Match`
- 검색 및 삭제 작업 사이에 발생하는 리소스 업데이트

성공적인 조건부 삭제의 예

다음 예시에서는 특정 기준에 따라 환자 리소스를 삭제합니다.

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

이 요청은 다음과 같은 환자 리소스를 삭제합니다.

- 이름은 "peter"입니다.
- 생년월일은 2000년 1월 1일입니다.
- 전화번호는 1234567890입니다.

모범 사례

1. 특정 검색 기준을 사용하여 여러 일치 항목을 방지하고 412 오류를 방지합니다.
2. 동시 수정을 처리하는 데 필요한 경우 버전 관리를 위해 ETag 헤더를 고려합니다.
3. 오류 응답을 적절하게 처리합니다.
   - 404의 경우: 검색 기준 구체화
   - 412의 경우: 기준을 더 구체적으로 지정하거나 버전 충돌 해결
4. 검색 작업과 삭제 작업 간에 리소스를 수정할 수 있는 동시성이 높은 환경에서 타이밍 충돌에 대비합니다.

자격 증명 및 동시성

자격 증명 키

AWS HealthLake 는 FHIR POST 작업에 대한 멱등성 키를 지원하여 리소스 생성 중에 데이터 무결성을 보장하는 강력한 메커니즘을 제공합니다. 고유한 UUID를 요청 헤더에 멱등성 키로 포함하면 의료 애플리케이션은 네트워크 불안정 또는 자동 재시도와 관련된 시나리오에서도 각 FHIR 리소스가 정확히 한 번 생성되도록 보장할 수 있습니다.

이 기능은 중복 의료 기록이 심각한 결과를 초래할 수 있는 의료 시스템에 특히 중요합니다. 이전 요청과 동일한 멱등성 키로 요청이 수신되면 HealthLake는 복제본을 생성하는 대신 원본 리소스를 반환합니다. 예를 들어 재시도 루프 중에 또는 중복 요청 파이프라인으로 인해 발생할 수 있습니다. 멱등성 키를 사용하면 HealthLake가 데이터 일관성을 유지하면서 간헐적인 연결 문제를 처리하는 클라이언트 애플리케이션에 원활한 환경을 제공할 수 있습니다.

구현

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

응답 시나리오

첫 번째 요청(201 생성됨)

- 새 리소스가 성공적으로 생성됨
- 응답에 리소스 ID 포함

중복 요청(409 충돌)

- 동일한 멱등성 키가 감지됨
- 반환된 원본 리소스
- 새 리소스가 생성되지 않음

잘못된 요청(400 잘못된 요청)

- 잘못된 UUID
- 필수 필드 누락

모범 사례

- 각 새 리소스 생성에 대해 고유한 UUID 생성
- 재시도 로직에 대한 멱등성 키 저장
- 일관된 키 형식 사용: UUID v4 권장
- 리소스 생성을 처리하는 클라이언트 애플리케이션에서 구현

> **Note**
>
> 이 기능은 엄격한 데이터 정확도를 요구하고 중복 의료 기록을 방지하는 의료 시스템에 특히 유용합니다.

AWS HealthLake의 ETag

AWS HealthLake 는 FHIR 리소스의 낙관적 동시성 제어에 ETags 사용하여 동시 수정을 관리하고 데이터 일관성을 유지하는 신뢰할 수 있는 메커니즘을 제공합니다. ETag는 리소스의 특정 버전을 나타내는 고유 식별자로, HTTP 헤더를 통해 버전 관리 시스템으로 작동합니다. 리소스를 읽거나 수정할 때 애플리케이션은 ETags 사용하여 의도하지 않은 덮어쓰기를 방지하고 특히 잠재적 동시 업데이트가 있는 시나리오에서 데이터 무결성을 보장할 수 있습니다.

구현 예제

```
// 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
```

응답 시나리오

성공한 작업(200 OK 또는 204 No Content)

- ETag가 현재 버전과 일치
- 작업이 의도한 대로 진행됨

버전 충돌(412 사전 조건 실패)

- ETag가 현재 버전과 일치하지 않음
- 데이터 손실을 방지하기 위해 업데이트가 거부됨

모범 사례

- 모든 업데이트 및 삭제 작업에 ETags 포함
- 버전 충돌을 처리하기 위한 재시도 로직 구현
- If-None-Match 사용: create-if-not-exists 시나리오의 경우 *
- 수정하기 전에 항상 ETag 최신 상태 확인

이 동시성 제어 시스템은 특히 여러 사용자 또는 시스템이 동일한 리소스에 액세스하고 수정하는 환경에서 의료 데이터의 무결성을 유지하는 데 필수적입니다.

에서 FHIR 리소스 검색 AWS HealthLake

FHIR search 상호 작용을 사용하여 일부 필터 기준에 따라 HealthLake 데이터 스토어에서 FHIR 리소스 세트를 검색합니다. 상호 search 작용은 GET 또는 POST 요청을 사용하여 수행할 수 있습니다. 개인 식별 정보(PII) 또는 보호 대상 건강 정보(PHI)가 포함된 검색의 경우 PII 및 PHI가 POST 요청 본문의 일부로 추가되고 전송 중에 암호화되므로 요청을 사용하는 것이 좋습니다.

> **Note**
>
> 이 장에서 설명하는 FHIR search 상호 작용은 의료 데이터 교환을 위한 HL7 FHIR R4 표준을 준수하도록 구축되었습니다. HL7 FHIR 서비스의 표현이므로 AWS CLI 및 AWS SDKs 통해 제공되지 않습니다. 자세한 내용은 FHIR R4 RESTful API 설명서search의 섹션을 참조하세요.

HealthLake는 FHIR R4 검색 파라미터의 하위 집합을 지원합니다. 자세한 내용은 HealthLake에 대한 FHIR R4 검색 파라미터 단원을 참조하십시오.

주제

- GET을 사용하여 FHIR 리소스 검색
- POST를 사용하여 FHIR 리소스 검색
- FHIR 검색 일관성 수준

GET을 사용하여 FHIR 리소스 검색

GET 요청을 사용하여 HealthLake 데이터 스토어를 검색할 수 있습니다. GET를 사용하는 경우 HealthLake는 URL의 일부로 검색 파라미터를 제공하는 것을 지원하지만 요청 본문의 일부로는 제공하지 않습니다. 자세한 내용은 HealthLake에 대한 FHIR R4 검색 파라미터 단원을 참조하십시오.

> **중요**
>
> 개인 식별 정보(PII) 또는 보호 대상 건강 정보(PHI)가 포함된 검색의 경우 PII 및 PHI가 POST 요청 본문의 일부로 추가되고 전송 중에 암호화되므로 보안 모범 사례는 요청을 사용하기 위해를 호출합니다. 자세한 내용은 POST를 사용하여 FHIR 리소스 검색 단원을 참조하십시오.

다음 절차에는를 사용하여 HealthLake 데이터 스토어를 GET 검색하는 예제가 나와 있습니다.

를 사용하여 HealthLake 데이터 스토어를 검색하려면 **GET**

1. HealthLake `region` 및 `datastoreId` 값을 수집합니다. 자세한 내용은 데이터 스토어 속성 가져오기 단원을 참조하십시오.
2. 연결된 `id` 값을 검색하고 수집할 FHIR 리소스 유형을 결정합니다. 자세한 내용은 조건 키 단원을 참조하십시오.
3. HealthLake `region` 및에 대해 수집된 값을 사용하여 요청에 대한 URL을 구성합니다.datastoreId. FHIR `Resource` 유형과 지원되는 검색 파라미터도 포함합니다. 다음 예제에서 전체 URL 경로를 보려면 복사 버튼을 스크롤합니다.

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

4. FHIR 권한 부여 시 AWS 서명 버전 4 또는 SMART를 사용하여 `GET` 요청을 보냅니다. 다음 `curl` 예시에서는 HealthLake 데이터 스토어의 총 `Patient` 리소스 수를 반환합니다. 전체 예제를 보려면 복사 버튼을 스크롤합니다.

   SigV4

   SigV4 권한 부여

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

   SMART on FHIR

   `IdentityProviderConfiguration` 데이터 형식에 대한 SMART on FHIR 권한 부여 예제입니다.

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

호출자는 권한 부여 Lambda에 권한을 할당할 수 있습니다. 자세한 내용은 OAuth 2.0 범위 단원을 참조하십시오.

AWS Console

> **Note**
>
> HealthLake 콘솔은 SigV4 권한 부여만 지원합니다. SMART on FHIR 권한 부여는 AWS CLI 및 AWS SDKs 통해 지원됩니다.

1. HealthLake 콘솔의 쿼리 실행 페이지에 로그인합니다.

2. 쿼리 설정 섹션에서 다음을 선택합니다.

- 데이터 스토어 ID - 쿼리 문자열을 생성할 데이터 스토어 ID를 선택합니다.
- 쿼리 유형 -를 선택합니다`Search with GET`.
- 리소스 유형 - 검색할 FHIR 리소스 유형을 선택합니다.
- 검색 파라미터 - 검색 파라미터 또는 검색 파라미터 조합을 선택하여 쿼리를 특정 레코드에 집중합니다.

3. 쿼리 실행을 선택합니다.

예: GET으로 검색

다음 탭은를 사용하여 특정 FHIR 리소스 유형을 검색하는 예제를 제공합니다GET. 이 예제에서는 요청 URLs에서 검색 파라미터를 지정하는 방법을 보여줍니다.

> **Note**
>
> HealthLake 콘솔은 SigV4 권한 부여만 지원합니다. SMART on FHIR 권한 부여는 AWS CLI 및 AWS SDKs 통해 지원됩니다.
> HealthLake는 FHIR R4 검색 파라미터의 하위 집합을 지원합니다. 자세한 내용은 검색 파라미터 단원을 참조하십시오.

Patient (age)

수명은 FHIR에서 정의된 리소스 유형이 아니지만 `Patient` 리소스 유형의 요소로 캡처됩니다. 다음 예제를 사용하여 1997년에 태어난 개인을 검색하기 위해 birthDate 요소와 `eq` 검색 비교기를 사용하여 `Patient` 리소스 유형에 대한 GET기반 검색을 요청합니다.

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

Condition

다음 예제를 사용하여 `Condition` 리소스 유형을 GET 요청합니다. 검색은 HealthLake 데이터 스토어에서 로 변환72892002되는 SNOMED 의료 코드가 포함된 조건을 찾습니다`Normal pregnancy`.

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

DocumentationReference

다음 예제는 (`Patients`)에 대한 `DocumentReference` 리소스 유형에 대해 연쇄상 알균 진단을 받고 아목시실린도 처방받은에 대한 GET 요청을 생성하는 방법을 보여줍니다.

```
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

다음 예제를 사용하여 Location 리소스 유형을 GET 요청합니다. 다음 검색은 주소의 일부로 도시 이름 보스턴이 포함된 HealthLake 데이터 스토어의 위치를 찾습니다.

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

Observation

다음 예제를 사용하여 Observation 리소스 유형에 대한 GET기반 검색 요청을 수행합니다. 이 검색은 value-concept 검색 파라미터를 사용하여 로 변환266919005되는 의료 코드를 찾습니다Never smoker.

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

POST를 사용하여 FHIR 리소스 검색

FHIR search 상호 작용을 POST 요청과 함께 사용하여 HealthLake 데이터 스토어를 검색할 수 있습니다. POST를 사용하는 경우 HealthLake는 URL 또는 요청 본문에서 검색 파라미터를 지원하지만 단일 요청에서 둘 다 사용할 수는 없습니다.

> **중요**
>
> 개인 식별 정보(PII) 또는 보호 대상 건강 정보(PHI)가 포함된 검색의 경우 PII 및 PHI가 POST 요청 본문의 일부로 추가되고 전송 중에 암호화되므로 보안 모범 사례는 요청을 사용하기 위해 를 호출합니다.

다음은 FHIR R4 상호 search 작용을 사용하여 HealthLake 데이터 스토어POST를 검색하는 예제입니다. 이 예제에서는 JSON 요청 본문에서 검색 파라미터를 지정하는 방법을 보여줍니다.

를 사용하여 HealthLake 데이터 스토어를 검색하려면 **POST**

1. HealthLake region 및 datastoreId 값을 수집합니다. 자세한 내용은 데이터 스토어 속성 가져오기 단원을 참조하십시오.

2. 연결된 id 값을 검색하고 수집할 FHIR 리소스 유형을 결정합니다. 자세한 내용은 조건 키 단원을 참조하십시오.

3. HealthLake region 및에 대해 수집된 값을 사용하여 요청에 대한 URL을 구성합니다datastoreId. FHIR Resource 유형 및 _search 상호 작용도 포함합니다. 다음 예제에서 전체 URL 경로를 보려면 복사 버튼을 스크롤합니다.

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

4. 검색할 FHIR 데이터를 지정하여 요청에 대한 JSON 본문을 구성합니다. 이 절차의 목적상 Observation 리소스를 검색하여 담배를 피운 적이 없는 환자를 찾습니다. 의료 코드 상태를 지정하려면 JSON 요청 본문value-concept=266919005에서를 Never smoker설정합니다. 파일을 search-observation.json(으)로 저장합니다.

```
value-concept=266919005
```

5. 요청을 보냅니다. FHIR search 상호 작용은 FHIR 권한 부여 시 AWS 서명 버전 4 또는 SMART와 함께 GET 요청을 사용합니다.

> **Note**
>
> POST 요청 본문에 검색 파라미터를 사용하여 요청할 때는를 헤더의 Content-Type: application/x-www-form-urlencoded 일부로 사용합니다.

다음 curl 예제에서는 Observation 리소스 유형에 대한 POST 기반 검색 요청을 수행합니다. 요청은 value-concept 검색 파라미터를 사용하여 값을 266919005 나타내는 의료 코드를 찾습니다Never smoker. 전체 예제를 보려면 복사 버튼을 스크롤합니다.

SigV4

SigV4 권한 부여

```
curl --request POST \
  'https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Observation/
_search' \
  --aws-sigv4 'aws:amz:region:healthlake' \
  --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
  --header "x-amz-security-token:$AWS_SESSION_TOKEN" \
```

```
  --header "Content-Type: application/x-www-form-urlencoded"
  --header "Accept: application/json"
  --data @search-observation.json
```

SMART on FHIR

[IdentityProviderConfiguration](#) 데이터 형식에 대한 SMART on FHIR 권한 부여 예제입니다.

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

호출자는 권한 부여 Lambda에 권한을 할당할 수 있습니다. 자세한 내용은 [OAuth 2.0 범위](#) 단원을 참조하십시오.

예: POST로 검색

다음 탭은를 사용하여 특정 FHIR 리소스 유형을 검색하는 예제를 제공합니다POST. 이 예제에서는 URLs에서 요청을 지정하는 방법을 보여줍니다.

> **Note**
>
> HealthLake 콘솔은 SigV4 권한 부여만 지원합니다. SMART on FHIR 권한 부여는 AWS CLI 및 AWS SDKs 통해 지원됩니다.

HealthLake는 FHIR R4 검색 파라미터의 하위 집합을 지원합니다. 자세한 내용은 검색 파라미터 단원을 참조하십시오.

Patient (age)

수명은 FHIR에서 정의된 리소스 유형이 아니지만 `Patient` 리소스 유형의 요소로 캡처됩니다. 다음 예제를 사용하여 `Patient` 리소스 유형에 대한 POST기반 검색을 요청합니다. 다음 검색 예제에서는 `eq` 검색 비교기를 사용하여 1997년에 태어난 개인을 검색합니다.

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

검색에서 1997년을 지정하려면 요청 본문에 다음 요소를 추가합니다.

```
birthdate=eq1997
```

Condition

다음을 사용하여 `Condition` 리소스 유형을 POST 요청합니다. 이 검색은 HealthLake 데이터 스토어에서 의료 코드가 포함된 위치를 찾습니다72892002.

요청 URL과 요청 본문을 지정해야 합니다. 다음은 요청 URL의 예입니다.

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

검색하려는 의료 코드를 지정하려면 요청 본문에 다음 JSON 요소를 추가합니다.

```
code=72892002
```

DocumentReference

`DocumentReference` 리소스 유형에 대해 POST 요청할 때 HealthLake의 통합 자연어 처리(NLP) 결과를 보려면 다음과 같이 요청의 형식을 지정합니다.

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

참조할 DocumentReference 검색 파라미터를 지정하려면 섹션을 참조하세요검색 파라미터 유형. 다음 쿼리 문자열은 여러 검색 파라미터를 사용하여 통합 NLP 결과를 생성하는 데 사용되는 Amazon Comprehend Medical API 작업을 검색합니다.

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

Location

다음 예제를 사용하여 Location 리소스 유형을 POST 요청합니다. 검색은 주소의 일부로 도시 이름 보스턴이 포함된 HealthLake 데이터 스토어의 위치를 찾습니다.

요청 URL과 요청 본문을 지정해야 합니다. 다음은 요청 URL의 예입니다.

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

검색Boston에서를 지정하려면 요청 본문에 다음 요소를 추가합니다.

```
address=Boston
```

Observation

다음 예제를 사용하여 Observation 리소스 유형에 대한 POST기반 검색을 요청합니다. 검색은 value-concept 검색 파라미터를 사용하여를 266919005 나타내는 의료 코드를 찾습니다Never smoker.

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

상태를 지정하려면 JSON 본문value-concept=266919005에 Never smoker를 설정합니다.

```
value-concept=266919005
```

FHIR 검색 일관성 수준

AWS HealthLake의 검색 인덱스는 GET 및 POST SEARCH 작업에 대한 최종 일관성 모델에서 작동합니다. 최종 일관성을 위해 리소스에 대해 보류 중인 검색 인덱스 업데이트가 있는 경우 인덱스 업데이트가 완료될 때까지 검색 결과에 리소스의 버전 N-1이 제외됩니다.

AWS HealthLake에는 이제 업데이트된 리소스에 대한 일관성 모델의 작동 방식을 선택할 수 있는 기능이 포함되어 있습니다. 개발자는 '이벤트 일관성', 위에서 설명한 기본 동작 또는 '강력한 일관성'을 포함할 수 있습니다. 강력한 일관성은 대기 중인 검색 인덱스 업데이트가 있는 리소스에 대한 리소스의 N-1 버전을 검색 결과에 포함할 수 있도록 허용합니다. 검색 인덱스 업데이트가 아직 완료되지 않은 경우에도 결과에 모든 리소스가 필요한 사용 사례 시나리오에 사용할 수 있습니다. 고객은 `x-amz-fhir-history-consistency-level` 요청 헤더를 사용하여이 동작을 제어할 수 있습니다.

일관성 수준

강력한 일관성

대기 중인 검색 인덱스 업데이트가 있는 레코드를 포함하여 일치하는 모든 레코드를 반환`x-amz-fhir-history-consistency-level: strong`하도록 설정합니다. 업데이트 직후 리소스를 검색해야 하는 경우이 옵션을 사용합니다.

최종 일관성

검색 인덱스 업데이트를 완료한 레코드만 반환`x-amz-fhir-history-consistency-level: eventual`하도록 설정합니다. 일관성 수준이 지정되지 않은 경우의 기본 동작입니다.

사용 예

1. 리소스를 업데이트할 때:

```
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. 후속 검색:

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

모범 사례

- 최근에 업데이트된 리소스를 즉시 검색해야 하는 경우 강력한 일관성 사용
- 즉각적인 가시성이 중요하지 않은 일반 쿼리에 최종 일관성 사용
- 즉각적인 가시성과 잠재적 성능 영향 간의 장단점을 고려하세요.

> **Note**
>
> 일관성 수준 설정은 업데이트된 리소스가 검색 결과에 표시되는 속도에 영향을 주지만 리소스의 실제 스토리지에는 영향을 주지 않습니다.
> 선택적 `x-amz-fhir-history-consistency-level` 헤더를 '강력'으로 설정하면 리소스당 쓰기 용량 소비가 두 배로 늘어납니다.
> 이 기능은 버전 기록이 활성화된 데이터 스토어에만 적용됩니다(2024년 10월 25일 이후에 생성된 모든 데이터 스토어에는 기본적으로 활성화되어 있음).

를 사용하여 FHIR 데이터 내보내기 AWS HealthLake

기본 AWS HealthLake 작업을 사용하여 FHIR 내보내기 작업을 시작, 설명 및 나열합니다. 내보내기 작업을 대기열에 넣을 수 있습니다. 비동기 내보내기 작업은 FIFO(선입선출) 방식으로 처리됩니다. 내보내기 작업을 시작하는 것과 동일한 방식으로 작업을 대기열에 넣을 수 있습니다. 진행 중인 경우 대기열에 추가하기만 하면 됩니다. 내보내기 작업이 진행되는 동안 FHIR 리소스를 생성, 읽기, 업데이트 또는 삭제할 수 있습니다.

> **Note**
>
> FHIR R4 `$export` 작업을 사용하여 HealthLake 데이터 스토어에서 FHIR 데이터를 내보낼 수도 있습니다. 자세한 내용은 FHIR을 사용하여 HealthLake 데이터 내보내기 `$export` 단원을 참조하십시오.

주제

FHIR 내보내기 작업 시작

`StartFHIRExportJob`를 사용하여 HealthLake 데이터 스토어에서 FHIR 내보내기 작업을 시작합니다. 다음 메뉴는 AWS Management Console 및 AWS SDKs의 AWS CLI 및 코드 예제에 대한 절차를 제공합니다. 자세한 내용을 알아보려면 AWS HealthLake API 참조의 *`StartFHIRExportJob`* 섹션을 참조하세요.

> **Note**
>
> HealthLake는 의료 데이터 교환을 위한 FHIR R4 사양을 지원합니다. 따라서 모든 상태 데이터는 FHIR R4 형식으로 내보내집니다.

FHIR 내보내기 작업을 시작하는 방법

액세스 기본 설정에 따라 메뉴를 선택합니다 AWS HealthLake.

AWS CLI 및 SDKs

CLI

AWS CLI

FHIR 내보내기 작업을 시작하는 방법

다음 `start-fhir-export-job` 예제에서는 AWS HealthLake를 사용하여 FHIR 내보내기 작업을 시작하는 방법을 보여줍니다.

```
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)
```

출력:

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

자세한 내용은 AWS HealthLake 개발자 안내서의 FHIR 데이터 스토어에서 파일 내보내기를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 StartFHIRExportJob 섹션을 참조하세요.

Python

SDK for Python(Boto3)

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

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


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

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

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 StartFHIRExportJob을 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

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

- API 세부 정보는 SDK for SAP ABAP API 참조의 StartFHIRExportJob을 참조하세요. AWS

> 예제 가용성
>
> 필요한 예제를 찾을 수 없습니까? 이 페이지의 오른쪽 사이드바에 있는 피드백 제공 링크를 사용하여 코드 예제를 요청합니다.

AWS 콘솔

1. HealthLake 콘솔의 데이터 스토어 페이지에 로그인합니다.
2. 데이터 스토어를 선택합니다.
3. 내보내기를 선택합니다.

   내보내기 페이지가 열립니다.
4. 출력 데이터 섹션에서 다음 정보를 입력합니다.

   - Amazon S3의 출력 데이터 위치
   - 출력 암호화
5. 액세스 권한 섹션에서 기존 IAM 서비스 역할 사용을 선택하고 역할 이름 메뉴에서 역할을 선택하거나 IAM 역할 생성을 선택합니다.

6. 데이터 내보내기를 선택합니다.

> **Note**
>
> 내보내는 동안 페이지 상단의 배너에서 작업 ID 복사를 선택합니다. 를 사용하여 `JobID`를 사용하여 내보내기 작업 속성을 요청할 수 있습니다 AWS CLI. 자세한 내용은 FHIR 내보내기 작업 속성 가져오기 단원을 참조하십시오.

FHIR 내보내기 작업 속성 가져오기

HealthLake 데이터 스토어에서 내보내기 작업 속성을 가져오는 `DescribeFHIRExportJob` 데 사용합니다. 다음 메뉴는 AWS Management Console 및 AWS SDKs의 AWS CLI 및 코드 예제에 대한 절차를 제공합니다. 자세한 내용을 알아보려면 AWS HealthLake API 참조의 *`DescribeFHIRExportJob`* 섹션을 참조하세요.

> **Note**
>
> HealthLake는 의료 데이터 교환을 위한 FHIR R4 사양을 지원합니다. 따라서 모든 상태 데이터는 FHIR R4 형식으로 내보내집니다.

FHIR 내보내기 작업을 설명하는 방법

액세스 기본 설정에 따라 메뉴를 선택합니다 AWS HealthLake.

AWS CLI 및 SDKs

CLI

AWS CLI

FHIR 내보내기 작업을 설명하는 방법

다음 `describe-fhir-export-job` 예제에서는 AWS HealthLake에서 FHIR 내보내기 작업의 속성을 찾는 방법을 보여줍니다.

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

출력:

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

자세한 내용은 AWS HealthLake 개발자 안내서의 FHIR 데이터 스토어에서 파일 내보내기를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DescribeFHIRExportJob 섹션을 참조하세요.

Python

SDK for Python(Boto3)

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

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

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

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 DescribeFHIRExportJob을 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

- API 세부 정보는 SDK for SAP ABAP API 참조의 DescribeFHIRExportJob을 참조하세요. AWS

> **예제 가용성**
>
> 필요한 예제를 찾을 수 없습니까? 이 페이지의 오른쪽 사이드바에 있는 피드백 제공 링크를 사용하여 코드 예제를 요청합니다.

AWS 콘솔

> **Note**
>
> FHIR 내보내기 작업 정보는 HealthLake 콘솔에서 사용할 수 없습니다. 대신와 함께를 AWS CLI 사용하여와 같은 내보내기 작업 속성을 `DescribeFHIRExportJob` 요청합니다`JobStatus`. 자세한 내용은이 페이지의 AWS CLI 예제를 참조하세요.

FHIR 내보내기 작업 나열

`ListFHIRExportJobs`를 사용하여 HealthLake 데이터 스토어에 대한 FHIR 내보내기 작업을 나열합니다. 다음 메뉴는 AWS Management Console 및 SDKs의 AWS CLI 및 AWS 코드 예제에 대한 절차를 제공합니다. 자세한 내용을 알아보려면 AWS HealthLake API 참조의 *`ListFHIRExportJobs`* 섹션을 참조하세요.

> **Note**
>
> HealthLake는 의료 데이터 교환을 위한 FHIR R4 사양을 지원합니다. 따라서 모든 상태 데이터는 FHIR R4 형식으로 내보내집니다.

FHIR 내보내기 작업을 나열하려면

액세스 기본 설정에 따라 메뉴를 선택합니다 AWS HealthLake.

AWS CLI 및 SDKs

CLI

AWS CLI

모든 FHIR 내보내기 작업을 나열하는 방법

다음 `list-fhir-export-jobs` 예제에서는 명령을 사용하여 계정과 연결된 내보내기 작업 목록을 보는 방법을 보여 줍니다.

```
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)
```

출력:

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

자세한 내용은 AWS HealthLake 개발자 안내서의 FHIR 데이터 스토어에서 파일 내보내기를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListFHIRExportJobs 섹션을 참조하세요.

Python

SDK for Python(Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
```

```
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
client.

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


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

```
                parameters["NextToken"] = next_token
            response =
self.health_lake_client.list_fhir_export_jobs(**parameters)
            jobs.extend(response["ExportJobPropertiesList"])
            if "NextToken" in response:
                next_token = response["NextToken"]
            else:
                break
        return jobs
    except ClientError as err:
        logger.exception(
            "Couldn't list export jobs. Here's why %s",
            err.response["Error"]["Message"],
        )
        raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 ListFHIRExportJobs을 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
  TRY.
      " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
      IF iv_submitted_after IS NOT INITIAL.
        oo_result = lo_hll->listfhirexportjobs(
```

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

- API 세부 정보는 SDK for SAP ABAP API 참조의 ListFHIRExportJobs를 참조하세요. AWS

> **예제 가용성**
>
> 필요한 예제를 찾을 수 없습니까? 이 페이지의 오른쪽 사이드바에 있는 피드백 제공 링크를 사용하여 코드 예제를 요청합니다.

AWS 콘솔

> **Note**
>
> FHIR 내보내기 작업 정보는 HealthLake 콘솔에서 사용할 수 없습니다. 대신와 AWS CLI 함께를 사용하여 모든 FHIR 내보내기 작업을 나열`ListFHIRExportJobs`합니다. 자세한 내용은 이 페이지의 AWS CLI 예제를 참조하세요.

AWS SDKs를 사용한 HealthLake 코드 예제

다음 코드 예제에서는 HealthLake를 AWS 소프트웨어 개발 키트(SDK)와 함께 사용하는 방법을 보여 줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 HealthLake 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

코드 예시

- AWS SDKs를 사용한 HealthLake의 기본 예제
  - AWS SDKs를 사용한 HealthLake 작업
    - AWS SDK 또는 CLI와 CreateFHIRDatastore 함께 사용
    - AWS SDK 또는 CLI와 DeleteFHIRDatastore 함께 사용
    - AWS SDK 또는 CLI와 DescribeFHIRDatastore 함께 사용
    - AWS SDK 또는 CLI와 DescribeFHIRExportJob 함께 사용
    - AWS SDK 또는 CLI와 DescribeFHIRImportJob 함께 사용
    - AWS SDK 또는 CLI와 ListFHIRDatastores 함께 사용
    - AWS SDK 또는 CLI와 ListFHIRExportJobs 함께 사용
    - AWS SDK 또는 CLI와 ListFHIRImportJobs 함께 사용
    - AWS SDK 또는 CLI와 ListTagsForResource 함께 사용
    - AWS SDK 또는 CLI와 StartFHIRExportJob 함께 사용
    - AWS SDK 또는 CLI와 StartFHIRImportJob 함께 사용
    - AWS SDK 또는 CLI와 TagResource 함께 사용
    - AWS SDK 또는 CLI와 UntagResource 함께 사용

AWS SDKs를 사용한 HealthLake의 기본 예제

다음 코드 예제에서는 AWS HealthLake SDKs에서의 기본 사항을 AWS 사용하는 방법을 보여줍니다.

예제

- AWS SDKs를 사용한 HealthLake 작업

- AWS SDK 또는 CLI와 CreateFHIRDatastore 함께 사용
- AWS SDK 또는 CLI와 DeleteFHIRDatastore 함께 사용
- AWS SDK 또는 CLI와 DescribeFHIRDatastore 함께 사용
- AWS SDK 또는 CLI와 DescribeFHIRExportJob 함께 사용
- AWS SDK 또는 CLI와 DescribeFHIRImportJob 함께 사용
- AWS SDK 또는 CLI와 ListFHIRDatastores 함께 사용
- AWS SDK 또는 CLI와 ListFHIRExportJobs 함께 사용
- AWS SDK 또는 CLI와 ListFHIRImportJobs 함께 사용
- AWS SDK 또는 CLI와 ListTagsForResource 함께 사용
- AWS SDK 또는 CLI와 StartFHIRExportJob 함께 사용
- AWS SDK 또는 CLI와 StartFHIRImportJob 함께 사용
- AWS SDK 또는 CLI와 TagResource 함께 사용
- AWS SDK 또는 CLI와 UntagResource 함께 사용

AWS SDKs를 사용한 HealthLake 작업

다음 코드 예제에서는 AWS SDKs를 사용하여 개별 HealthLake 작업을 수행하는 방법을 보여줍니다. 각 예시에는 GitHub에 대한 링크가 포함되어 있습니다. 여기에서 코드 설정 및 실행에 대한 지침을 찾을 수 있습니다.

다음 예제에는 가장 일반적으로 사용되는 작업만 포함되어 있습니다. 전체 목록은 AWS HealthLake API 참조를 참조하세요.

예제

- AWS SDK 또는 CLI와 CreateFHIRDatastore 함께 사용
- AWS SDK 또는 CLI와 DeleteFHIRDatastore 함께 사용
- AWS SDK 또는 CLI와 DescribeFHIRDatastore 함께 사용
- AWS SDK 또는 CLI와 DescribeFHIRExportJob 함께 사용
- AWS SDK 또는 CLI와 DescribeFHIRImportJob 함께 사용
- AWS SDK 또는 CLI와 ListFHIRDatastores 함께 사용
- AWS SDK 또는 CLI와 ListFHIRExportJobs 함께 사용
- AWS SDK 또는 CLI와 ListFHIRImportJobs 함께 사용
- AWS SDK 또는 CLI와 ListTagsForResource 함께 사용

- AWS SDK 또는 CLI와 StartFHIRExportJob 함께 사용
- AWS SDK 또는 CLI와 StartFHIRImportJob 함께 사용
- AWS SDK 또는 CLI와 TagResource 함께 사용
- AWS SDK 또는 CLI와 UntagResource 함께 사용

AWS SDK 또는 CLI와 **CreateFHIRDatastore** 함께 사용

다음 코드 예시는 CreateFHIRDatastore의 사용 방법을 보여 줍니다.

CLI

AWS CLI

예제 1: SigV4-enabled HealthLake 데이터 스토어 생성

다음 create-fhir-datastore 예제에서는 AWS HealthLake에서 새 데이터 스토어를 생성하는 방법을 보여줍니다.

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

출력:

```
{
    "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)"
}
```

예제 2: FHIR 지원 HealthLake 데이터 스토어에서 SMART 생성

다음 create-fhir-datastore 예제에서는 AWS HealthLake에서 FHIR 지원 데이터 스토어에서 새 SMART를 생성하는 방법을 보여줍니다.

```
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.json의 콘텐츠:

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

출력:

```
{
    "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)"
}
```

자세한 내용은 AWS HealthLake 개발자 안내서의 FHIR 데이터 스토어 생성 및 모니터링을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 CreateFHIRDatastore 섹션을 참조하세요.

Python

SDK for Python(Boto3)

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

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


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

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

```
                and identity_provider_configuration is not None
            ):
                # Creating a SMART on FHIR-enabled data store
                parameters["SseConfiguration"] = sse_configuration
                parameters[
                    "IdentityProviderConfiguration"
                ] = identity_provider_configuration

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

다음 코드에서는 SMART on FHIR이 활성화된 HealthLake 데이터 저장소의 파라미터 예시를 보여줍니다.

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

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

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 CreateFHIRDatastore를 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

- API 세부 정보는 SDK for SAP ABAP API 참조의 CreateFHIRDatastore를 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 HealthLake 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **DeleteFHIRDatastore** 함께 사용

다음 코드 예시는 DeleteFHIRDatastore의 사용 방법을 보여 줍니다.

CLI

AWS CLI

FHIR 데이터 스토어를 삭제하려면

다음 delete-fhir-datastore 예제에서는 AWS HealthLake에서 데이터 스토어와 모든 콘텐츠를 삭제하는 방법을 보여줍니다.

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

출력:

```
{
    "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)"
}
```

자세한 내용은 AWS HealthLake 개발자 안내서의 FHIR 데이터 스토어 <https://docs.aws.amazon.com/healthlake/latest/devguide/working-with-FHIR-healthlake.html> 생성 및 모니터링을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DeleteFHIRDatastore 섹션을 참조하세요.

Python

SDK for Python(Boto3)

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

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


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

```
        try:

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

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 DeleteFHIRDatastore를 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

```
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_access_ex.
      CATCH /aws1/cx_hllconflictexception INTO DATA(lo_conflict_ex).
        lv_error = |Conflict error: { lo_conflict_ex->av_err_code }-
{ lo_conflict_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_conflict_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- API 세부 정보는 SDK for SAP ABAP API 참조의 DeleteFHIRDatastore를 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 HealthLake 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **DescribeFHIRDatastore** 함께 사용

다음 코드 예시는 DescribeFHIRDatastore의 사용 방법을 보여 줍니다.

CLI

AWS CLI

FHIR 데이터 스토어를 설명하려면

다음 describe-fhir-datastore 예제에서는 AWS HealthLake에서 데이터 스토어의 속성을 찾는 방법을 보여줍니다.

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

출력:

```
{
    "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
        }
    }
}
```

자세한 내용은 AWS HealthLake 개발자 안내서의 FHIR 데이터 스토어 생성 및 모니터링을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DescribeFHIRDatastore 섹션을 참조하세요.

Python

SDK for Python(Boto3)

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

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

```
        return cls(health_lake_client)


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

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 DescribeFHIRDatastore를 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        oo_result = lo_hll->describefhirdatastore(
          iv_datastoreid = iv_datastore_id
        ).
        DATA(lo_datastore_properties) = oo_result->get_datastoreproperties( ).
        IF lo_datastore_properties IS BOUND.
          DATA(lv_datastore_name) = lo_datastore_properties-
>get_datastorename( ).
          DATA(lv_datastore_status) = lo_datastore_properties-
>get_datastorestatus( ).
          MESSAGE 'Data store described successfully.' TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        DATA(lv_error) = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        lv_error = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
    ENDTRY.
```

- API 세부 정보는 SDK for SAP ABAP API 참조의 DescribeFHIRDatastore를 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 HealthLake 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **DescribeFHIRExportJob** 함께 사용

다음 코드 예시는 DescribeFHIRExportJob의 사용 방법을 보여 줍니다.

CLI

AWS CLI

FHIR 내보내기 작업을 설명하는 방법

다음 `describe-fhir-export-job` 예제에서는 AWS HealthLake에서 FHIR 내보내기 작업의 속성을 찾는 방법을 보여줍니다.

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

출력:

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

자세한 내용은 AWS HealthLake 개발자 안내서의 FHIR 데이터 스토어에서 파일 내보내기를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DescribeFHIRExportJob 섹션을 참조하세요.

Python

SDK for Python(Boto3)

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

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


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

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 DescribeFHIRExportJob을 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

- API 세부 정보는 SDK for SAP ABAP API 참조의 DescribeFHIRExportJob을 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 HealthLake 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **DescribeFHIRImportJob** 함께 사용

다음 코드 예시는 DescribeFHIRImportJob의 사용 방법을 보여 줍니다.

CLI

AWS CLI

FHIR 가져오기 작업을 설명하는 방법

다음 describe-fhir-import-job 예제에서는 AWS HealthLake를 사용하여 FHIR 가져오기 작업의 속성을 학습하는 방법을 보여줍니다.

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

출력:

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

자세한 내용은 AWS HealthLake 개발자 안내서의 FHIR 데이터 스토어에 파일 가져오기를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DescribeFHIRImportJob 섹션을 참조하세요.

Python

SDK for Python(Boto3)

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

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


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

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 DescribeFHIRImportJob을 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

- API 세부 정보는 SDK for SAP ABAP API 참조의 DescribeFHIRImportJob을 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 HealthLake 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 `ListFHIRDatastores` 함께 사용

다음 코드 예시는 `ListFHIRDatastores`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

FHIR 데이터 스토어를 나열하려면

다음 `list-fhir-datastores` 예제에서는 명령을 사용하는 방법과 사용자가 AWS HealthLake의 데이터 스토어 상태를 기반으로 결과를 필터링하는 방법을 보여줍니다.

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

출력:

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

자세한 내용은 AWS HealthLake 개발자 안내서의 FHIR 데이터 스토어 생성 및 모니터링을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListFHIRDatastores 섹션을 참조하세요.

Python

SDK for Python(Boto3)

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

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


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

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

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

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 ListFHIRDatastores를 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

- API 세부 정보는 SDK for SAP ABAP API 참조의 ListFHIRDatastores를 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 HealthLake 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **ListFHIRExportJobs** 함께 사용

다음 코드 예시는 ListFHIRExportJobs의 사용 방법을 보여 줍니다.

CLI

AWS CLI

모든 FHIR 내보내기 작업을 나열하는 방법

다음 list-fhir-export-jobs 예제에서는 명령을 사용하여 계정과 연결된 내보내기 작업 목록을 보는 방법을 보여 줍니다.

```
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)
```

출력:

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

자세한 내용은 AWS HealthLake 개발자 안내서의 FHIR 데이터 스토어에서 파일 내보내기를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListFHIRExportJobs 섹션을 참조하세요.

Python

SDK for Python(Boto3)

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

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


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

```
                response =
self.health_lake_client.list_fhir_export_jobs(**parameters)
                jobs.extend(response["ExportJobPropertiesList"])
                if "NextToken" in response:
                    next_token = response["NextToken"]
                else:
                    break
            return jobs
        except ClientError as err:
            logger.exception(
                "Couldn't list export jobs. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 ListFHIRExportJobs을 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

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

- API 세부 정보는 SDK for SAP ABAP API 참조의 ListFHIRExportJobs를 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 HealthLake 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **ListFHIRImportJobs** 함께 사용

다음 코드 예시는 ListFHIRImportJobs의 사용 방법을 보여 줍니다.

CLI

AWS CLI

모든 FHIR 가져오기 작업을 나열하는 방법

다음 list-fhir-import-jobs 예제에서는 명령을 사용하여 계정과 연결된 모든 가져오기 작업 목록을 보는 방법을 보여 줍니다.

```
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)
```

출력:

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

자세한 내용은 AWS HealthLake 개발자 안내서의 FHIR 데이터 스토어로 파일 가져오기를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListFHIRImportJobs 섹션을 참조하세요.

Python

SDK for Python(Boto3)

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

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


    def list_fhir_import_jobs(
        self,
        datastore_id: str,
        job_name: str = None,
        job_status: str = None,
        submitted_before: datetime = None,
        submitted_after: datetime = None,
    ) -> list[dict[str, any]]:
        """
        Lists HealthLake import jobs satisfying the conditions.
        :param datastore_id: The data store ID.
        :param job_name: The import job name.
        :param job_status: The import job status.
        :param submitted_before: The import job submitted before the specified
date.
        :param submitted_after: The import job submitted after the specified
date.
        :return: A list of import jobs.
        """
        try:
            parameters = {"DatastoreId": datastore_id}
```

```
            if job_name is not None:
                parameters["JobName"] = job_name
            if job_status is not None:
                parameters["JobStatus"] = job_status
            if submitted_before is not None:
                parameters["SubmittedBefore"] = submitted_before
            if submitted_after is not None:
                parameters["SubmittedAfter"] = submitted_after
            next_token = None
            jobs = []
            # Loop through paginated results.
            while True:
                if next_token is not None:
                    parameters["NextToken"] = next_token
                response =
self.health_lake_client.list_fhir_import_jobs(**parameters)
                jobs.extend(response["ImportJobPropertiesList"])
                if "NextToken" in response:
                    next_token = response["NextToken"]
                else:
                    break
            return jobs
        except ClientError as err:
            logger.exception(
                "Couldn't list import jobs. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 ListFHIRImportJobs을 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        IF iv_submitted_after IS NOT INITIAL.
          oo_result = lo_hll->listfhirimportjobs(
            iv_datastoreid = iv_datastore_id
            iv_submittedafter = iv_submitted_after
          ).
        ELSE.
          oo_result = lo_hll->listfhirimportjobs(
            iv_datastoreid = iv_datastore_id
          ).
        ENDIF.
        DATA(lt_import_jobs) = oo_result->get_importjobpropertieslist( ).
        DATA(lv_job_count) = lines( lt_import_jobs ).
        MESSAGE |Found { lv_job_count } import job(s).| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- API 세부 정보는 SDK for SAP ABAP API 참조의 ListFHIRImportJobs를 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 HealthLake 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **ListTagsForResource** 함께 사용

다음 코드 예시는 ListTagsForResource의 사용 방법을 보여 줍니다.

CLI

AWS CLI

데이터 스토어의 태그를 나열하려면

다음 list-tags-for-resource 예제에서는 지정된 데이터 스토어와 연결된 태그를 나열합니다.

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

출력:

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

자세한 내용은AWS HealthLake 개발자 안내서의 HealthLake에서 리소스 태그 지정을 참조하세요. AWS HealthLake

- API 세부 정보는 AWS CLI 명령 참조의 ListTagsForResource를 참조하세요.

Python

SDK for Python(Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
```

```
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
client.

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


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

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 ListTagsForResource를 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

- API 세부 정보는 SDK for SAP ABAP API 참조의 ListTagsForResource를 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 HealthLake 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **StartFHIRExportJob** 함께 사용

다음 코드 예시는 StartFHIRExportJob의 사용 방법을 보여 줍니다.

CLI

AWS CLI

FHIR 내보내기 작업을 시작하는 방법

다음 `start-fhir-export-job` 예제에서는 AWS HealthLake를 사용하여 FHIR 내보내기 작업을 시작하는 방법을 보여줍니다.

```
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)
```

출력:

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

자세한 내용은 AWS HealthLake 개발자 안내서의 FHIR 데이터 스토어에서 파일 내보내기를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 StartFHIRExportJob 섹션을 참조하세요.

Python

SDK for Python(Boto3)

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

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

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


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

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

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 StartFHIRExportJob을 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

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

- API 세부 정보는 SDK for SAP ABAP API 참조의 StartFHIRExportJob을 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 HealthLake 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **StartFHIRImportJob** 함께 사용

다음 코드 예시는 StartFHIRImportJob의 사용 방법을 보여 줍니다.

CLI

AWS CLI

FHIR 가져오기 작업을 시작하는 방법

다음 start-fhir-import-job 예제에서는 AWS HealthLake를 사용하여 FHIR 가져오기 작업을 시작하는 방법을 보여줍니다.

```
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)"
```

출력:

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

자세한 내용은 AWS HealthLake 개발자 안내서의 FHIR 데이터 스토어에 파일 가져오기를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 StartFHIRImportJob 섹션을 참조하세요.

Python

SDK for Python(Boto3)

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

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


    def start_fhir_import_job(
        self,
        job_name: str,
        datastore_id: str,
        input_s3_uri: str,
        job_output_s3_uri: str,
        kms_key_id: str,
        data_access_role_arn: str,
    ) -> dict[str, str]:
        """
        Starts a HealthLake import job.
        :param job_name: The import job name.
        :param datastore_id: The data store ID.
```

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

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 StartFHIRImportJob을 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

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

- API 세부 정보는 SDK for SAP ABAP API 참조의 StartFHIRImportJob을 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 HealthLake 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 `TagResource` 함께 사용

다음 코드 예시는 TagResource의 사용 방법을 보여 줍니다.

CLI

AWS CLI

데이터 스토어에 태그를 추가하려면

다음 tag-resource 예제에서는 데이터 스토어에 태그를 추가하는 방법을 보여 줍니다.

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

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS HealthLake개발자 안내서의 데이터 스토어에 태그 추가를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 TagResource를 참조하세요.

Python

SDK for Python(Boto3)

```
    @classmethod
```

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

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


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

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 TagResource를 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 TagResource를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 HealthLake 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **`UntagResource`** 함께 사용

다음 코드 예시는 `UntagResource`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

데이터 스토어에서 태그를 제거하려면

다음 `untag-resource` 예제에서는 데이터 스토어에서 태그를 제거하는 방법을 보여 줍니다.

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

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS HealthLake 개발자 안내서의 데이터 스토어에서 태그 제거를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 UntagResource를 참조하세요.

Python

SDK for Python(Boto3)

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

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


    def untag_resource(self, resource_arn: str, tag_keys: list[str]) -> None:
        """
        Untags a HealthLake resource.
        :param resource_arn: The resource ARN.
        :param tag_keys: The tag keys to remove from the resource.
        """
```

```
        try:
            self.health_lake_client.untag_resource(
                ResourceARN=resource_arn, TagKeys=tag_keys
            )
        except ClientError as err:
            logger.exception(
                "Couldn't untag resource %s. Here's why %s",
                resource_arn,
                err.response["Error"]["Message"],
            )
            raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 UntagResource를 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        " iv_resource_arn = 'arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        lo_hll->untagresource(
          iv_resourcearn = iv_resource_arn
          it_tagkeys = it_tag_keys
        ).
        MESSAGE 'Resource untagged successfully.' TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
```

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

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 UntagResource를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 HealthLake 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

통합 AWS HealthLake

다음 AWS 서비스는와 직접 통합되어 통합 자연어 처리, SQL 쿼리 및 데이터 웨어하우징 AWS HealthLake 을 지원합니다.

- Amazon Comprehend Medical은 기계 학습 라이브러리를 사용하여 HealthLake의 비정형 의료 텍스트에서 의미 있는 상태 데이터를 추출하는 HIPAA 적격 자연어 처리 서비스(NLP)입니다. 자세한 내용은 Amazon Comprehend Medical 개발자 안내서를 참조하세요.
- Amazon Athena는 표준 SQL을 사용하여 Amazon Simple Storage Service(Amazon S3) 버킷에서 HealthLake 데이터를 직접 분석할 수 있는 대화형 쿼리 서비스입니다. 자세한 내용은 Amazon Athena 개발자 안내서를 참조하세요.

주제

HealthLake용 통합 자연어 처리(NLP)

AWS HealthLake 는 FHIR `DocumentReference` 리소스 유형에 저장된 비정형 데이터를 구문 분석, 식별 및 매핑하기 위한 통합 자연어 처리(NLP) 라이브러리를 제공합니다.

> **중요**
>
> HealthLake용 통합 NLP는 기본적으로 꺼져 있습니다. 활성화하려면를 사용하여 지원 사례를 제출합니다AWS Support Center Console. 사례를 생성하려면에 로그인 AWS 계정 하고 사례 생성을 선택합니다.

HealthLake 통합 NLP는 Amazon Comprehend Medical `DetectEntities-V2`, `InferICD10-CM`및 `InferRxNorm` API 작업을 호출하여 작동합니다. 작업은 결과를 `DocumentReference` 리소스에 확장명으로 추가합니다. Amazon Comprehend Medical API 작업이 `SIGN`, `SYMPTOM`및/또는 인 특성을 감지하면 FHIR `Linkage` 리소스를 `DIAGNOSIS`생성합니다. 새 `Condition` 및 `Observation` 리소스는 `SIGN`, `SYMPTOM`또는 특성으로 식별된 엔터티에서 생성`DIAGNOSIS`되며 `Linkage` 리소스를 사용하여 소스 문서에 연결됩니다.

> **Note**
>
> HealthLake 통합 NLP에서 생성한 FHIR 리소스에 대한 GET 요청은 지원되지만 FHIR API search 기능은 지원되지 않습니다. HealthLake와 Athena의 통합을 사용하여 NLP 확장을 검색하는 방법에 대한 자세한 내용은 섹션을 참조하세요SQL 인덱스 및 쿼리.

목차

HealthLake 통합 NLP 라이브러리

HealthLake는 Amazon Comprehend Medical 라이브러리를 사용하여 DocumentReference 리소스 유형에서 찾은 데이터를 유추합니다. Amazon Comprehend Medical API 작업 DetectEntities-V2, InferICD10-CM및는 의학적 상태를 특성으로 InferRxNorm 감지합니다. 각 작업은 서로 다른 인사이트를 제공합니다.

> **⚠ 언어 지원**
>
> Amazon Comprehend Medical API 작업은 영어 텍스트로 된 의료 엔터티만 감지합니다.

- DetectEntities-V2: 임상 텍스트에서 다양한 의료 엔터티를 검사하고 엔터티 범주, 위치, 신뢰도 점수와 같은 특정 정보를 반환합니다.
- InferICD10-CM: 환자 레코드의 의학적 상태를 개체로 감지하고, 세계 보건 기구(WHO)의 권한 부여에 따라 CDC의 국립 건강 통계 센터의 ICD-10-CM 지식 기반의 정규화된 개념 식별자에 해당 개체를 연결합니다.
- InferRxNorm: 환자 레코드에 나열된 엔터티로 약물을 감지하고 National Library of Medicine에서 RxNorm 데이터베이스의 정규화된 개념 식별자에 연결합니다.

각 API 작업에 지원되는 특성은 SIGN, SYMPTOM및 입니다DIAGNOSIS. 특성이 감지되면 HealthLake 데이터 스토어의 여러 위치에 FHIR 호환 확장으로 추가됩니다.

확장이 추가되는 위치입니다.

- DocumentReference: Amazon Comprehend Medical API 작업의 결과는 DocumentReference 리소스 유형 내에서 찾은 각 문서에 extension 로 추가됩니다. 확장의 결과는 두 그룹으로 나뉩니다. 를 기반으로 결과에서 찾을 수 있습니다URL.
  - http://healthlake.amazonaws.com/system-generated-resources/
    - HealthLake에서 생성하거나에 추가한 리소스 유형입니다.
  - http://healthlake.amazonaws.com/aws-cm/
    - Amazon Comprehend Medical API 작업의 원시 출력이 HealthLake 데이터 스토어에 추가되는 위치입니다.
- Linkage:이 리소스 유형은 통합 NLP의 결과로 추가되거나 생성됩니다. 특정에 대한 GET 요청은 연결된 리소스 목록을 Linkage 반환합니다. HealthLake에서를 추가Linkage했는지 확인하려면 추가된 "tag": [{"display": "SYSTEM_GENERATED"}] 키-값 페어를 찾습니다. Linkage의 FHIR 사양에 대한 자세한 내용은 FHIR R4 설명서Linkage의 섹션을 참조하세요.
- Amazon Comprehend Medical 작업의 결과로 생성된 FHIR 리소스 유형입니다.
  - Observation: 에는 Amazon Comprehend Medical API 작업의 결과와 특성이 SIGN 또는 일 DetectEntities-V2 InferICD10-CM 때의 결과가 포함됩니다SYMPTOM.
  - Condition: Amazon Comprehend Medical API 작업의 결과와 특성이 인 DetectEntities-V2 InferICD10-CM 경우를 포함합니다DIAGNOSIS.
  - MedicationStatement: 에는 Amazon Comprehend Medical API 작업의 결과가 포함됩니다InferRxNorm.

FHIR REST API 상호 작용 사용

기본적으로 Amazon Comprehend Medical API 작업에서 감지된 특성은 GET 요청 시 반환되지 않습니다. 통합 NLP 작업의 결과를 보려면 다음 FHIR 리소스 유형에 ID 대해 알려진를 지정해야 합니다.

- Linkage
- Observation
- Condition
- MedicationStatement

DocumentReference 리소스 유형 외부의 HealthLake 통합 NLP 작업 결과는 지정된에 Amazon Comprehend Medical API 작업의 결과가 포함된 것으로 ID 알려진 GET 요청을 사용하여 사용할 수 있습니다.

HealthLake 통합 NLP에 대한 검색 파라미터

다음 표에는 HealthLake 통합 NLP의 검색 가능한 속성이 나열되어 있습니다.

HealthLake NLP에 대한 검색 파라미터

검색 파라미터	에 대한 일치 항목을 찾습니다.
detectEntities-entity-category	AWS CM 확장 내의 DetectEntities 하위 확장 내의 개체 범주
detectEntities-entity-text	AWS CM 확장 내의 DetectEntities 하위 확장 내의 개체 텍스트
detectEntities-entity-type	AWS CM 확장 내의 DetectEntities 하위 확장명 내의 개체 유형
detectEntities-entity-score	AWS CM 확장 내의 DetectEntities 하위 확장 내의 개체 점수
infer-icd10cm-entity-text	AWS CM 확장 내의 InferICD10CM 하위 확장 내의 개체 텍스트
infer-icd10cm-entity-score	AWS CM 확장 내의 InferICD10CM 하위 확장 내의 개체 점수
infer-icd10cm-entity-concept-code	AWS CM 확장 내의 InferICD10CM 하위 확장 내의 엔터티 개념 코드
infer-icd10cm-entity-concept-description	CM 확장 내의 InferICD10CM 하위 확장 내의 AWS 개체 개념 설명
infer-icd10cm-entity-concept-score	AWS CM 확장 내의 InferICD10CM 하위 확장 내의 개체 개념 점수
infer-rxnorm-entity-score	AWS CM 확장 내의 InferRxNorm 하위 확장 내의 개체 점수

검색 파라미터	에 대한 일치 항목을 찾습니다.
infer-rxnorm-entity-text	AWS CM 확장 내의 InferRxNorm 하위 확장 내의 개체 텍스트
infer-rxnorm-entity-concept-code	AWS CM 확장 내의 InferRxNorm 하위 확장 내의 엔터티 개념 코드
infer-rxnorm-entity-concept-description	CM 확장 내의 InferRxNorm 하위 확장 내의 AWS 개체 개념 설명
infer-rxnorm-entity-concept-score	AWS CM 확장 내의 InferRxNorm 하위 확장 내의 개체 개념 점수

HealthLake는 `EntityText` 및가 동일한 개체의 `EntityCategory` 일부인 기준과 일치하는 특수 검색을 제공합니다. 다음 표에서는 HealthLake에서 지원하는 특수 검색 파라미터를 설명합니다.

검색 파라미터

검색 파라미터	반환된 일치 항목
detectEntities-entity-text-category	DetectEntities 하위 확장에 entityText 및 entityCategory와 모두 일치하는 개체가 하나 이상 있는 경우.
detectEntities-entity-type-score	DetectEntities 하위 확장에 entityType 및 entityScore와 모두 일치하는 개체가 하나 이상 있는 경우.
detectEntities-entity-text-score	DetectEntities 하위 확장에 entityText 및 entityScore와 모두 일치하는 개체가 하나 이상 있는 경우.
detectEntities-entity-text-type	DetectEntities 하위 확장에 entityText 및 entityType과 모두 일치하는 개체가 하나 이상 있는 경우.
detectEntities-entity-category-score	entityCategory 및 entityScore와 모두 일치하는 개체가 하나 이상 있는 경우.
infer-icd10cm-entity-text-concept-code	InferICD10CM 하위 확장에 entityText와 일치하는 엔터티가 하나 이상 있고 해당 엔터티에 대해 코드와 일치하는 conceptCode가 하나 이상 있는 경우.

검색 파라미터	반환된 일치 항목
infer-icd10cm-entity-text-concept-score	InferICD10CM 하위 확장에 entityText와 일치하는 개체가 하나 이상 있고 점수와 일치하는 해당 개체에 대한 conceptScore가 하나 이상 있는 경우.
infer-icd10cm-entity-concept-description-concept-score	InferICD10CM 하위 확장의 개체 내에 개념 설명 및 conceptScore와 일치하는 개념이 하나 이상 있는 경우.
infer-rxnorm-entity-text-concept-code	InferRxNorm 하위 확장에 entityText와 일치하는 엔터티가 하나 이상 있고 코드와 일치하는 해당 엔터티에 대한 conceptCode가 하나 이상 있는 경우.
infer-rxnorm-entity-text-concept-score	InferRxNorm 하위 확장에 entityText와 일치하는 개체가 하나 이상 있고 점수와 일치하는 해당 개체에 대한 conceptScore가 하나 이상 있는 경우.
infer-rxnorm-entity-concept-description-concept-score	InferRxNorm 하위 확장의 개체 내에 개념 설명 및 conceptScore와 일치하는 개념이 하나 이상 있는 경우.

HealthLake 통합 NLP 예제 요청

예제 1: HealthLake 데이터 스토어에 수집된 **`Patient`** 레코드

다음은 의료 전문가와의 `Patient` 접촉을 기반으로 한 임상 기록의 예입니다.

> **합성 데이터**
>
> 다음 예제의 텍스트는 합성 콘텐츠이며 보호 대상 건강 정보(PHI)를 포함하지 않습니다.

```
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)
```

참고로이 정보는 DocumentReference 리소스에서 base64 형식으로 인코딩됩니다. 이 문서가 HealthLake에 수집되고 Amazon Comprehend Medical API 작업이 완료되면 결과를 보기 위해 DocumentReference 리소스 유형에 대한 GET요청으로 시작할 수 있습니다.

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

Amazon Comprehend Medical API 작업이 성공하면 다음과 `extension` 연결된 내에서 이러한 키-값 페어를 찾습니다. `"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."
 }
```

다음 탭은 수집된 의료 기록이 리소스 유형에 따라 HealthLake 데이터 스토어에 보고되는 방법을 보여줍니다.

DocumentReference

에서 단일 `DocumentReference` 리소스 유형에 대한 결과를 보려면 특정 리소스`id`의가 제공되는 `GET` 요청을 하세요.

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

성공하면 `200` HTTP 응답 코드와 다음 JSON 응답(명확성을 위해 잘림)을 가져옵니다.

다음은 `http://healthlake.amazonaws.com/system-generated-resources/` 부분입니다. 새가 추가`Linkage/e366d29f-2c22-4c19-866e-09603937935a`되었음을 확인할 수 있습니다. HealthLake가 특정 `Observation` 및 `Condition` 리소스 유형에 추론 기반 조사 결과를 추가한 위치를 확인할 수도 있습니다.

이러한 리소스 유형이 어떻게 수정되었는지 확인하려면 관련 탭을 선택합니다.

```
{
  "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

에서 단일 Linkage 리소스 유형에 대한 결과를 보려면 특정 리소스ID의가 제공되는 GET 요청을 하세요.

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

성공하면 200 HTTP 응답 코드와 다음과 같은 잘린 JSON 응답을 가져옵니다.

응답에는 item 요소가 포함됩니다. 여기서 키-값 페어는를 수정하는 데 사용되는 특정 DocumentReference 항목을 "type": "source" 나타내며 "type": "alternate" 키-값 페어 아래에 Condition Observations 나열됩니다.

meta 요소 및 해당 키-값 페어*"tag": [{"display": "SYSTEM_GENERATED"}]*인 도 표시됩니다. 이는 이러한 리소스가 HealthLake에서 생성되었음을 나타냅니다.

```
{
  "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

에서 단일 `Observation` 리소스 유형에 대한 결과를 보려면 특정 리소스ID의가 제공되는 GET 요청을 하세요.

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

Amazon Comprehend Medical API 작업의 결과는 `code`, `meta`및 요소로 수정됩니다`modifierExtension`.

`code`

유형의 요소입니다`CodeableConcept`. 자세한 내용은 FHIR R4 설명서`CodeableConcept`의 섹션을 참조하세요.

HealthLake는 다음과 같은 세 가지 키-값 페어를 추가합니다.

- "system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/": URL이 특정 Amazon Comprehend Medical API 작업을 참조하는 경우. 이 경우 InferICD10CM입니다.
- "code": "A52.06": A52.06는 미국 질병 통제 센터의 지식 기반에서 찾을 수 있는 개념을 식별하는 ICD-10-CM 코드입니다.
- "display": "Other syphilitic heart involvement": 여기서 "Other syphilitic heart involvement"는 온톨로지의 ICD-10-CM 코드에 대한 긴 설명입니다.

다음 잘린 JSON 응답에는 code 요소만 포함됩니다.

```
"code": {
  "coding":
  [
    {
   "system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/",
   "code": "A52.06",
   "display": "Other syphilitic heart involvement"
    }
  ],
  "text": "Other syphilitic heart involvement"
 }
```

할당된 ICD-10-CM 코드가 올바르다는 모델의 신뢰도를 이해하려면 modifierExtension 요소를 사용합니다.

meta

meta 요소에는 Amazon Comprehend Medical API 작업에 의해 추가된 세부 정보가 code 요소에 포함되어 있는지 여부를 나타내는 메타데이터가 포함되어 있습니다.

다음 잘린 JSON 응답에는 meta 요소만 포함됩니다.

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

modifierExtension

modifierExtension 요소에는 code 요소에 있는 할당된 코드의 신뢰도 수준에 대한 자세한 내용이 포함되어 있습니다. 또한 결과 및 관련 Linkage 리소스 유형을 생성하는 데 사용되는 원래 DocumentReference에 대한 링크를 다시 제공하는 키-값 페어가 있습니다.

추가된 각 coding 요소에 대해 entity-score 및가 modifierExtension에 entity-Concept-Score 추가됩니다. 키-값 페어의 각 값에 대해 점수가 표시됩니다. 의 경우 entity-score이 점수는 Amazon Comprehend Medical이 탐지 정확도에 대해 갖는 신뢰도 수준입니다. 의 경우 entity-Concept-Score이 점수는 Amazon Comprehend Medical이 엔터티가 ICD-10-CM 개념에 정확하게 연결되어 있다는 확신 수준입니다.

다음 잘린 JSON 응답에는 modifierExtension 요소만 포함됩니다.

```
"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"
   }
  }
 ]
```

전체 JSON 응답

```
{
  "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

에서 단일 Condition 리소스 유형에 대한 결과를 보려면 특정 리소스ID의가 제공되는 GET 요청을 하세요.

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

Amazon Comprehend Medical API 작업의 결과는 code, meta및 요소로 수정됩니다modifierExtension.

code

유형의 요소입니다CodeableConcept. 자세한 내용은 FHIR R4 설명서CodeableConcept의 섹션을 참조하세요.

HealthLake는 다음과 같은 세 가지 키-값 페어를 추가합니다.

- "system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/": URL이 특정 Amazon Comprehend Medical API 작업을 참조하는 경우. 이 경우 InferICD10CM입니다.
- "code": "I70.0": A52.06는 미국 질병 통제 센터의 지식 기반에서 찾을 수 있는 개념을 식별하는 ICD-10-CM 코드입니다.
- "display": "Atherosclerosis of aorta": 여기서 "Other syphilitic heart involvement"는 온톨로지의 ICD-10-CM 코드에 대한 긴 설명입니다.

다음 잘린 JSON 응답에는 code 요소만 포함됩니다.

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

할당된 ICD-10-CM 코드가 올바르다는 모델의 신뢰도를 이해하려면 modifierExtension 요소를 사용합니다.

meta

meta 요소에는 Amazon Comprehend Medical API 작업에 의해 추가된 세부 정보가 code 요소에 포함되어 있는지 여부를 나타내는 메타데이터가 포함되어 있습니다.

다음 잘린 JSON 응답에는 meta 요소만 포함됩니다.

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

modifierExtension

modifierExtension 요소에는 code 요소에 있는 할당된 코드의 신뢰도 수준에 대한 자세한 내용이 포함되어 있습니다. 또한 결과 및 관련 Linkage 리소스 유형을 생성하는 데 사용되는 원래 DocumentReference에 대한 링크를 다시 제공하는 키-값 페어가 있습니다.

추가된 각 coding 요소에 대해 entity-score 및가 modifierExtension에 entity-Concept-Score 추가됩니다. 키-값 페어의 각 값에 대해 점수가 표시됩니다. 의 경우 entity-score이 점수는 Amazon Comprehend Medical이 탐지 정확도에 대해 갖는 신뢰도 수준입니다. 의 경우 entity-Concept-Score이 점수는 Amazon Comprehend Medical이 엔터티가 ICD-10-CM 개념에 정확하게 연결되어 있다는 확신 수준입니다.

다음 잘린 JSON 응답에는 modifierExtension 요소만 포함됩니다.

```
"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"
   }
  }
 ]
```

전체 JSON 응답

```
{
  "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"
 }
```

예제 2: **MedicationStatement** 리소스 유형이 **DocumentReference** 포함된 A

다음은 의료 전문가와의 환자 접촉을 기반으로 한 임상 기록의 예입니다.

> ⚠ 합성 데이터
>
> 이 예제의 텍스트는 합성 콘텐츠이며 보호 대상 건강 정보(PHI)를 포함하지 않습니다.

```
Tom is not prescribed Advil
```

다음 탭은 수집된 의료 기록이 리소스 유형에 따라 HealthLake 데이터 스토어에 보고되는 방법을 보여줍니다.

DocumentReference

에서 단일 DocumentReference 리소스 유형에 대한 결과를 보려면 특정 리소스ID의가 제공되는 GET 요청을 하세요.

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

성공하면 200 HTTP 응답 코드와 다음과 같은 잘린 JSON 응답이 표시됩니다.

키-값 페어인는 Amazon Comprehend Medical API 작업에 의해이 안에 리소스 유형이 추가extension되었음을 "url": "http://healthlake.amazonaws.com/system-generated-resources/"나타냅니다. 새 Linkage 리소스 유형과 여러 MedicationStatement 리소스를 볼 수 있습니다.


```
"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

에서 단일 Linkage 리소스 유형에 대한 결과를 보려면 특정 리소스ID의가 제공되는 GET 요청을 하세요.

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

성공하면 200 HTTP 응답 코드와 다음 JSON 응답을 가져옵니다.

응답에는 item 요소가 포함됩니다. 여기서 키-값 페어는 MedicationStatement 리소스 유형을 수정하는 데 사용되는 특정 DocumentReference 항목을 "type": "source" 나타냅니다.

meta 요소 및 해당 키-값 페어인 도 볼 수 있으며"tag": [{"display": "SYSTEM_GENERATED"}], 이는 이러한 리소스가 HealthLake에서 생성되었음을 나타냅니다.

```
{
  "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

에서 단일 MedicationStatement 리소스 유형에 대한 결과를 보려면 특정 리소스ID의가 제공되는 GET 요청을 하세요.

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

MedicationStatement 리소스 유형은 Amazon Comprehend Medical InferRxNorm API 작업의 결과를 찾을 수 있는 위치입니다. 결과는 medicationCodeableConcept, meta및 요소로 수정됩니다modifierExtension.

medicationCodeableConcept

유형의 요소입니다CodeableConcept. 자세한 내용은 FHIR R4 설명서CodeableConcept의 섹션을 참조하세요.

HealthLake는 다음과 같은 세 가지 키-값 페어를 추가합니다.

- "system": ""http://healthlake.amazonaws.com/aws-cm/infer-rxnorm/: URL이 특정 Amazon Comprehend Medical API 작업을 참조하는 경우. 이 경우 InferRxNorm입니다.
- "code": "731533": 여기서 731533는 RxCUI라고도 하는 RxNorm 개념 ID입니다. RxCUI
- "display": "ibuprofen 200 MG Oral Capsule [Advil]": 여기서 ibuprofen 200 MG Oral Capsule [Advil]는 RxNorm 개념에 대한 설명입니다.

다음 잘린 JSON 응답에는 MedicationStatement 요소만 포함됩니다.

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

meta

meta 요소에는 Amazon Comprehend Medical API 작업에 의해 추가된 세부 정보가 code 요소에 포함되어 있는지 여부를 나타내는 메타데이터가 포함되어 있습니다.

다음 잘린 JSON 응답에는 meta 요소만 포함됩니다.

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

```
   }
  ]
 }
```

modifierExtension

modifierExtension 요소에는 결과 및 관련 Linkage 리소스 유형을 생성하는 데 DocumentReference 사용된 원본에 대한 링크를 다시 제공하는 키-값 페어가 포함되어 있습니다.

```
"modifierExtension": [
   {
    "url": "http://healthlake.amazonaws.com/system-generated-linkage",
    "valueReference": {
     "reference": "Linkage/394bb244-177b-4409-8657-26b20ed56dd7"
    }
   },
   {
    "url": "http://healthlake.amazonaws.com/source-document-reference",
    "valueReference": {
     "reference": "DocumentReference/c549125d-a218-421f-b8bf-23614c5e796c"
    }
   }
 ]
```

Amazon Athena를 사용하여 HealthLake 데이터 쿼리

HealthLake 가져오기 작업 중에 중첩된 FHIR JSON 데이터는 ETL 프로세스를 거치며 Apache Iceberg 오픈 테이블 형식으로 저장됩니다. 여기서 각 FHIR 리소스 유형은 Athena의 개별 테이블로 표시됩니다. 이를 통해 사용자는 먼저 내보낼 필요 없이 SQL을 사용하여 FHIR 데이터를 쿼리할 수 있습니다. 이는 임상의와 과학자가 FHIR 데이터를 쿼리하여 결정을 검증하거나 연구를 진행할 수 있도록 하기 때문에 중요합니다. Athena에서 Apache Iceberg 테이블이 작동하는 방식에 대한 자세한 내용은 Athena 사용 설명서의 Apache Iceberg 테이블 쿼리를 참조하세요.

> **Note**
>
> HealthLake는 Athena의 HealthLake 데이터에 대한 FHIR R4 read 상호 작용을 지원합니다. 자세한 내용은 FHIR 리소스 읽기 단원을 참조하십시오.

이 섹션의 주제에서는 HealthLake 데이터 스토어를 Athena에 연결하는 방법, SQL을 사용하여 쿼리하는 방법, 추가 분석을 위해 결과를 다른 AWS 서비스와 연결하는 방법을 설명합니다.

주제

- Amazon Athena 시작하기
- SQL을 사용하여 HealthLake 데이터 쿼리
- 복잡한 필터링을 사용하는 SQL 쿼리 예제

Amazon Athena 시작하기

HealthLake를 Amazon Athena와 통합하려면 권한을 설정해야 합니다. 이렇게 하려면 Athena 사용자, 그룹 또는 역할을 생성하고 HealthLake 데이터 스토어 내에 있는 FHIR 리소스에 대한 액세스 권한을 부여해야 합니다.

- 사용자, 그룹 또는 역할에 HealthLake 데이터 스토어에 대한 액세스 권한 부여(AWS Lake Formation 콘솔)
- Athena 계정 설정

사용자, 그룹 또는 역할에 HealthLake 데이터 스토어에 대한 액세스 권한 부여(AWS Lake Formation 콘솔)

> 페르소나: HealthLake 관리자
>
> HealthLake 관리자 페르소나는 AWS Lake Formation의 데이터 레이크 관리자입니다. Lake Formation의 HealthLake 데이터 스토어에 대한 액세스 권한을 부여합니다.

생성된 각 데이터 스토어에 대해 AWS Lake Formation 콘솔에 두 개의 항목이 표시됩니다. 한 가지 항목은 리소스 링크입니다. 리소스 링크 이름은 항상 기울임꼴로 표시됩니다. 각 리소스 링크는 연결된 공유 리소스의 이름과 소유자와 함께 표시됩니다. 모든 HealthLake 데이터 스토어의 공유 리소스 소유자는 HealthLake 서비스 계정입니다. 다른 항목은 HealthLake 서비스 계정의 HealthLake 데이터 스토어입니다. 이 절차의 단계에서는 리소스 링크인 데이터 스토어를 사용합니다.

리소스 링크에 대한 자세한 내용은 Lake Formation 개발자 안내서의 Lake Formation에서 리소스 링크가 작동하는 방식을 참조하세요. AWS

사용자, 그룹 또는 역할이 Athena에서 데이터를 쿼리할 수 있으려면 리소스 데이터베이스에 대한 설명 권한을 부여해야 합니다. 그런 다음 테이블에 Select 및 Describe를 부여해야 합니다.

1단계: HealthLake 데이터 스토어 리소스 링크 데이터베이스에 대한 DESCRIBE 권한 부여

1. AWS Lake Formation 콘솔 열기: https://console.aws.amazon.com/lakeformation/
2. 기본 탐색 모음에서 데이터베이스를 선택합니다.
3. 데이터베이스 페이지에서 기울임꼴로 표시된 데이터 스토어 이름 옆에 있는 라디오 버튼을 선택합니다.
4. 작업(")을 선택합니다.
5. 권한 부여를 선택합니다.
6. 데이터 권한 부여 페이지의 보안 주체에서 IAM 사용자 또는 역할을 선택합니다.
7. IAM 사용자 또는 역할에서 아래쪽 화살표(")를 사용하거나 Athena에서 쿼리할 수 있는 IAM 사용자, 역할 또는 그룹을 검색합니다.
8. LF 태그 또는 카탈로그 리소스 카드에서 명명된 데이터 카탈로그 리소스 옵션을 선택합니다.
9. 데이터베이스에서 아래쪽 화살표(")를 사용하여 액세스를 공유할 HealthLake 데이터 스토어 데이터베이스를 선택합니다.
10. 리소스 링크 권한 카드의 리소스 링크 권한에서 설명을 선택합니다.

권한 부여가 성공하면 권한 부여 성공 배너가 나타납니다. 방금 부여한 권한을 보려면 데이터 레이크 권한을 선택합니다. 테이블에서 사용자, 그룹 및 역할을 찾습니다. 권한 열 아래에 나열된 설명이 표시됩니다.

이제 대상에 권한 부여를 사용하여 데이터베이스의 모든 테이블에 대해 선택 및 설명을 부여해야 합니다.

2단계: HealthLake 데이터 스토어 리소스 링크의 모든 테이블에 대한 액세스 권한 부여

1. AWS Lake Formation 콘솔 열기: https://console.aws.amazon.com/lakeformation/
2. 기본 탐색 모음에서 데이터베이스를 선택합니다.
3. 데이터베이스 페이지에서 기울임꼴로 표시된 데이터 스토어 이름 옆에 있는 라디오 버튼을 선택합니다.
4. 작업(")을 선택합니다.
5. 대상에 부여를 선택합니다.

6. 데이터 권한 부여 페이지의 보안 주체에서 IAM 사용자 또는 역할을 선택합니다.
7. IAM 사용자 또는 역할에서 아래쪽 화살표(")를 사용하거나 Athena에서 쿼리할 수 있는 IAM 사용자, 그룹 또는 역할을 검색합니다.
8. LF 태그 또는 카탈로그 리소스 카드에서 명명된 데이터 카탈로그 리소스 옵션을 선택합니다.
9. 데이터베이스에서 아래쪽 화살표(")를 사용하여 액세스 권한을 부여할 HealthLake 데이터 스토어 데이터베이스를 선택합니다.
10. 테이블에서 모든 테이블을 선택하여 모든 테이블을 HealthLake 사용자와 공유합니다.
11. 테이블 권한 카드의 테이블 권한에서 설명 및 선택을 선택합니다.
12. 권한 부여를 선택합니다.

권한 부여를 선택하면 권한 부여 성공 배너가 나타납니다. 이제 지정된 사용자가 Athena의 HealthLake 데이터 스토어에서 쿼리를 수행할 수 있습니다.

Athena 시작하기

> **HealthLake 사용자**
>
> HealthLake 사용자는 Athena 콘솔 AWS CLI또는 AWS SDKs를 사용하여 HealthLake 관리자가 공유한 HealthLake 데이터 스토어를 쿼리합니다.

Athena를 사용하여 데이터 스토어를 쿼리하려면 다음 세 가지 작업을 수행해야 합니다.

- Lake Formation을 통해 IAM 사용자 또는 역할에 HealthLake 데이터 스토어에 대한 액세스 권한을 부여합니다. 자세한 내용은 사용자, 그룹 또는 역할에 HealthLake 데이터 스토어에 대한 액세스 권한 부여(AWS Lake Formation 콘솔)를 참조하세요.
- HealthLake 데이터 스토어에 대한 작업 그룹을 생성합니다.
- 쿼리 결과를 저장할 Amazon S3 버킷을 지정합니다.

Athena를 시작하려면 AmazonAthenaFullAccess 및 AmazonS3FullAccess AWS 관리형 정책을 사용자, 그룹 또는 역할에 추가합니다. AWS 관리형 정책을 사용하는 것은 새 서비스 사용을 시작하는 좋은 방법입니다. AWS 관리형 정책은 모든 AWS 고객이 사용할 수 있으므로 특정 사용 사례에 대해 최소 권한 권한을 부여하지 않을 수 있습니다. IAM 정책을 사용하여 권한을 설정하는 경우 작업을 수행하는 데 필요한 권한만 부여합니다. IAM 및 최소 권한 적용에 대한 자세한 내용은 IAM 사용 설명서의 최소 권한 적용을 참조하세요.

> **Important**
>
> Athena에서 HealthLake 데이터 스토어를 쿼리하려면 Athena 엔진 버전 3을 사용해야 합니다.

작업 그룹은 리소스이므로 IAM 기반 정책을 사용하여 특정 작업 그룹에 대한 액세스를 제어할 수 있습니다. 자세한 내용은 Athena 사용 설명서의 작업 그룹을 사용하여 쿼리 액세스 및 비용 제어를 참조하세요.

작업 그룹 설정에 대한 자세한 내용은 Athena 사용 설명서https://docs.aws.amazon.com/athena/latest/ug/workgroups-procedure.html의 섹션을 참조하세요.

> **Note**
>
> Amazon S3 버킷이 있는 리전이며 Athena 콘솔이 일치해야 합니다.

쿼리를 실행하려면 먼저, Amazon S3에서 쿼리 결과 버킷 위치를 지정하거나, 버킷을 지정했고 구성이 클라이언트 설정을 재정의하는 작업 그룹을 사용해야 합니다. 실행되는 모든 쿼리에 대해 출력 파일은 자동으로 저장됩니다.

Athena 콘솔에서 쿼리 결과 위치를 지정하는 방법에 대한 자세한 내용은 Amazon Athena 사용 설명서의 Athena 콘솔을 사용하여 쿼리 결과 위치 지정을 참조하세요.

Athena에서 HealthLake 데이터 스토어를 쿼리하는 방법의 예를 보려면 섹션을 참조하세요SQL을 사용하여 HealthLake 데이터 쿼리.

SQL을 사용하여 HealthLake 데이터 쿼리

FHIR 데이터를 HealthLake 데이터 스토어로 가져오면 중첩된 JSON FHIR 데이터는 동시에 ETL 프로세스를 거치고 Amazon S3에 Apache Iceberg 오픈 테이블 형식으로 저장됩니다. HealthLake 데이터 스토어의 각 FHIR 리소스 유형은 Amazon Athena를 사용하여 쿼리할 수 있는 테이블로 변환됩니다. SQL 기반 쿼리를 사용하여 테이블을 개별적으로 또는 그룹으로 쿼리할 수 있습니다. 데이터 스토어의 구조로 인해 데이터를 Athena로 여러 데이터 유형으로 가져옵니다. 이러한 데이터 유형에 액세스할 수 있는 SQL 쿼리를 생성하는 방법에 대한 자세한 내용은 Amazon Athena 사용 설명서의 복잡한 유형 및 중첩 구조가 있는 쿼리 배열을 참조하세요.

> **Note**
>
> 이 주제의 모든 예제에서는 Synthea를 사용하여 생성된 가상 데이터를 사용합니다. Synthea 데이터가 사전 로드된 데이터 스토어를 생성하는 방법에 대한 자세한 내용은 섹션을 참조하세요HealthLake 데이터 스토어 생성.

리소스 유형의 각 요소에 대해 FHIR 사양은 카디널리티를 정의합니다. 요소의 카디널리티는이 요소가 나타날 수 있는 횟수의 하한과 상한을 정의합니다. SQL 쿼리를 생성할 때 이를 고려해야 합니다. 예를 들어 리소스 유형: 환자에서 몇 가지 요소를 살펴보겠습니다.

- 요소: 이름 FHIR 사양은 카디널리티를 로 설정합니다`0..*`.

  요소는 배열로 캡처됩니다.

```
[{
 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
}]
```

  Athena에서 리소스 유형이 어떻게 수집되었는지 확인하려면 테이블 및 뷰에서 해당 유형을 검색합니다. 이 배열의 요소에 액세스하려면 점 표기법을 사용할 수 있습니다. 다음은 `given` 및의 값에 액세스하는 간단한 예제입니다`family`.

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

```
FROM Patient
```

- 요소: MaritalStatus FHIR 사양은 카디널리티를 로 설정합니다`0..1`.

  이 요소는 JSON으로 캡처됩니다.

```
{
 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
}
```

  Athena에서 리소스 유형이 어떻게 수집되었는지 확인하려면 테이블 및 뷰에서 해당 유형을 검색합니다. JSON의 키-값 페어에 액세스하려면 점 표기법을 사용할 수 있습니다. 배열이 아니기 때문에 배열 인덱스가 필요하지 않습니다. 다음은의 값에 액세스하는 간단한 예입니다`text`.

```
SELECT
    maritalstatus.text as MaritalStatus
FROM Patient
```

JSON 액세스 및 검색에 대한 자세한 내용은 Athena 사용 설명서의 JSON 쿼리를 참조하세요.

Athena Data Manipulation Language(DML) 쿼리 문은 Trino를 기반으로 합니다. Athena는 Trino의 모든 기능을 지원하지 않으며 상당한 차이가 있습니다. 자세한 내용은 Amazon Athena 사용 설명서의 DML 쿼리, 함수 및 연산자를 참조하세요.

또한 Athena는 HealthLake 데이터 스토어의 쿼리를 생성할 때 발생할 수 있는 여러 데이터 유형을 지원합니다. Athena의 데이터 유형에 대한 자세한 내용은 Amazon Athena 사용 설명서의 Amazon Athena의 데이터 유형을 참조하세요. Amazon Athena

Athena에서 SQL 쿼리가 작동하는 방식에 대한 자세한 내용은 Amazon Athena 사용 설명서의 Amazon Athena용 SQL 참조를 참조하세요. Amazon Athena

각 탭은 Athena를 사용하여 지정된 리소스 유형 및 관련 요소를 검색하는 방법의 예를 보여줍니다.

Element: Extension

요소는 데이터 스토어에서 사용자 지정 필드를 생성하는 데 `extension` 사용됩니다.

이 예제에서는 `Patient` 리소스 유형에 있는 `extension` 요소의 기능에 액세스하는 방법을 보여줍니다.

HealthLake 데이터 스토어를 Athena로 가져오면 리소스 유형의 요소가 다르게 구문 분석됩니다. 의 구조`element`는 가변적이므로 스키마에서 완전히 지정할 수 없습니다. 이러한 변동성을 처리하기 위해 배열 내의 요소는 문자열로 전달됩니다.

의 표 설명에서 로 `extension` 설명된 요소를 볼 수 있습니다. `array<string>`즉`Patient`, 인덱스 값을 사용하여 배열의 요소에 액세스할 수 있습니다. 그러나 문자열의 요소에 액세스하려면를 사용해야 합니다`json_extract`.

다음은 환자 테이블에 있는 `extension` 요소의 단일 항목입니다.

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

유효한 JSON이더라도 Athena는 이를 문자열로 취급합니다.

이 SQL 쿼리 예제에서는 `patient-mothersMaidenName` 및 `patient-birthPlace` 요소가 포함된 테이블을 생성하는 방법을 보여줍니다. 이러한 요소에 액세스하려면 다른 배열 인덱스 및를 사용해야 합니다. `json_extract.`

```
SELECT
    extension[1],
    json_extract(extension[1], '$.valueString') AS MothersMaidenName,
    extension[2],
    json_extract(extension[2], '$.valueAddress.city') AS birthPlace
FROM patient
```

JSON과 관련된 쿼리에 대한 자세한 내용은 Amazon Athena 사용 설명서의 JSON에서 데이터 추출을 참조하세요.

Element: birthDate (Age)

Age는 FHIR에서 환자 리소스 유형의 요소가 아닙니다. 다음은 연령을 기준으로 필터링하는 검색의 두 가지 예입니다.

수명은 요소가 아니므로 SQL 쿼리`birthDate`에를 사용합니다. 요소가 FHIR에 어떻게 수집되었는지 확인하려면 테이블 및 뷰에서 테이블 이름을 검색합니다. 문자열 유형임을 알 수 있습니다.

예제 1: 연령 값 계산

이 샘플 SQL 쿼리에서는 내장 SQL 도구 `current_date` 및를 사용하여 이러한 구성 요소를 추출`year`합니다. 그런 다음 이를 빼서 환자의 실제 연령을 라는 열로 반환합니다`age`.

```
SELECT
 (year(current_date) - year(date(birthdate))) as age
FROM patient
```

예제 2: 이전에 태어났2019-01-01고 인 환자를 필터링합니다male.

SQL 쿼리는 CAST 함수를 사용하여 birthDate 요소를 유형으로 캐스팅하는 방법과 WHERE 절의 두 기준을 기반으로 필터링하는 DATE방법을 보여줍니다. 요소는 기본적으로 유형 문자열로 수집되므로 유형CAST으로 수집해야 합니다DATE. 그런 다음 < 연산자를 사용하여 다른 날짜인와 비교할 수 있습니다2019-01-01. 를 사용하여 WHERE절에 두 번째 기준을 추가할 AND수 있습니다.

```
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

이 예제는 도시 이름이 Attle™인 Location 리소스 유형 내 위치 검색을 보여줍니다.

```
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

리소스 유형 조건은 우려 수준으로 상승한 문제와 관련된 진단 데이터를 저장합니다. HealthLake의 통합 의료 자연어 처리(NLP)는 DocumentReference Condition 리소스 유형에 있는 세부 정보를 기반으로 새 리소스를 생성합니다. 새 리소스가 생성되면 HealthLake는 태그를 SYSTEM_GENERATED meta 요소에 추가합니다. 이 샘플 SQL 쿼리는 조건 테이블을 검색하고 결과가 제거된 SYSTEM_GENERATED 결과를 반환하는 방법을 보여줍니다.

HealthLake의 통합 자연어 처리(NLP)에 대한 자세한 내용은 섹션을 참조하세요HealthLake용 통합 자연어 처리(NLP).

```
SELECT *
FROM condition
```

```
WHERE meta.tag[1] is NULL
```

지정된 문자열 요소 내에서 검색하여 쿼리를 추가로 필터링할 수도 있습니다. `modifierextension` 요소에는 조건 세트를 생성하는 데 사용된 `DocumentReference` 리소스에 대한 세부 정보가 포함되어 있습니다. 다시 말하지만 `json_extract`를 사용하여 Athena로 문자열로 가져온 중첩된 JSON 요소에 액세스해야 합니다.

이 샘플 SQL 쿼리는 특정를 기반으로 생성된 모든를 검색`Condition`하는 방법을 보여줍니다`DocumentReference`. `CAST`를 사용하여를 `LIKE` 비교할 수 있도록 JSON 요소를 문자열로 설정합니다.

```
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

리소스 유형인 관찰은 환자, 디바이스 또는 기타 주제에 대한 측정값과 간단한 어설션을 저장합니다. HealthLake의 통합 자연어 처리(NLP)는 `Observation` 리소스에 있는 세부 정보를 기반으로 새 `DocumentReference` 리소스를 생성합니다. 이 샘플 SQL 쿼리에는 `WHERE meta.tag[1] is NULL` 주석이 추가되어 `SYSTEM_GENERATED` 결과가 포함됩니다.

```
SELECT valueCodeableConcept.coding[1].code
FROM Observation
WHERE  valueCodeableConcept.coding[1].code = '266919005'
-- WHERE meta.tag[1] is NULL
```

이 열은 로 가져왔습니다`struct`. 따라서 점 표기법을 사용하여 내부 요소에 액세스할 수 있습니다.

Resource type: MedicationStatement

MedicationStatement는 환자가 사용했거나 사용 중이거나 향후 사용할 약에 대한 세부 정보를 저장하는 데 사용할 수 있는 FHIR 리소스 유형입니다. HealthLake의 통합 의료 자연어 처리(NLP)는 DocumentReference 리소스 유형에 있는 문서를 기반으로 새 MedicationStatement 리소스를 생성합니다. 새 리소스가 생성되면 HealthLake는 태그를 `SYSTEM_GENERATED` `meta` 요소에 추가합니

다. 이 샘플 SQL 쿼리는 식별자를 사용하여 단일 환자를 기준으로 필터링하고 HealthLake의 통합 NLP에서 추가한 리소스를 찾는 쿼리를 생성하는 방법을 보여줍니다.

```
SELECT *
FROM medicationstatement
WHERE meta.tag[1].display = 'SYSTEM_GENERATED' AND subject.reference =
 'Patient/0679b7b7-937d-488a-b48d-6315b8e7003b';
```

HealthLake의 통합 자연어 처리(NLP)에 대한 자세한 내용은 섹션을 참조하세요HealthLake용 통합 자연어 처리(NLP).

복잡한 필터링을 사용하는 SQL 쿼리 예제

다음 예제에서는 복잡한 필터링과 함께 Amazon Athena SQL 쿼리를 사용하여 HealthLake 데이터 스토어에서 FHIR 데이터를 찾는 방법을 보여줍니다.

Example인구 통계 데이터를 기반으로 필터링 기준 생성

환자 코호트를 생성할 때는 올바른 환자 인구 통계를 식별하는 것이 중요합니다. 이 샘플 쿼리는 Trino 점 표기법 및를 사용하여 HealthLake 데이터 스토어의 데이터를 필터링`json_extract`하는 방법을 보여줍니다.

```
SELECT
    id
    , CONCAT(name[1].family, ' ', name[1].given[1]) as name
    , (year(current_date) - year(date(birthdate))) as age
    , gender as gender
    , json_extract(extension[1], '$.valueString') as MothersMaidenName
    , json_extract(extension[2], '$.valueAddress.city') as birthPlace
    , maritalstatus.coding[1].display as maritalstatus
    , address[1].line[1] as addressline
    , address[1].city as city
    , address[1].district as district
    , address[1].state as state
    , address[1].postalcode as postalcode
    , address[1].country as country
    , json_extract(address[1].extension[1], '$.extension[0].valueDecimal') as latitude
    , json_extract(address[1].extension[1], '$.extension[1].valueDecimal') as longitude
    , telecom[1].value as telNumber
    , deceasedboolean as deceasedIndicator
    , deceaseddatetime
```

```
FROM database.patient;
```

Athena 콘솔을 사용하여 결과를 추가로 정렬하고 다운로드할 수 있습니다.

Example환자 및 관련 조건에 대한 필터 생성

다음 예제 쿼리는 HealthLake 데이터 스토어에서 찾은 환자의 모든 관련 조건을 찾고 정렬하는 방법을 보여줍니다.

```
SELECT
 patient.id as patientId
    , condition.id  as conditionId
    , CONCAT(name[1].family, ' ', name[1].given[1]) as name
    , condition.meta.tag[1].display
    , json_extract(condition.modifierextension[1], '$.valueDecimal') AS confidenceScore
    , category[1].coding[1].code as categoryCode
    , category[1].coding[1].display as categoryDescription
    , code.coding[1].code as diagnosisCode
    , code.coding[1].display as diagnosisDescription
    , onsetdatetime
    , severity.coding[1].code as severityCode
    , severity.coding[1].display as severityDescription
    , verificationstatus.coding[1].display as verificationStatus
    , clinicalstatus.coding[1].display as clinicalStatus
    , encounter.reference as encounterId
    , encounter.type as encountertype
FROM database.patient, condition
WHERE CONCAT('Patient/', patient.id) = condition.subject.reference
ORDER BY name;
```

Athena 콘솔을 사용하여 결과를 추가로 정렬하거나 추가 분석을 위해 다운로드할 수 있습니다.

Example환자 및 관련 관찰에 대한 필터 생성

다음 예제 쿼리는 HealthLake 데이터 스토어에서 발견된 환자의 모든 관련 관찰을 찾고 정렬하는 방법을 보여줍니다.

```
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;
```

Example환자 및 관련 절차에 대한 필터링 조건 생성

환자와 절차를 연결하는 것은 의료의 중요한 측면입니다. 다음 SQL 예제 쿼리는 FHIR `Patient` 및 `Procedure` 리소스 유형을 사용하여 이를 수행하는 방법을 보여줍니다. 다음 SQL 쿼리는 HealthLake 데이터 스토어에 있는 모든 환자와 관련 절차를 반환합니다.

```
SELECT
 patient.id  as patientId
 , PROCEDURE.id as procedureId
 , CONCAT(name[1].family, ' ', name[1].given[1]) as name
```

```
 , status
 , category.coding[1].code as categoryCode
 , category.coding[1].display as categoryDescription
 , code.coding[1].code as procedureCode
 , code.coding[1].display as procedureDescription
 , performeddatetime
 , performer[1]
 , encounter.reference as encounterId
 , encounter.type as encountertype
FROM database.patient, procedure
WHERE CONCAT('Patient/', patient.id) = procedure.subject.reference
ORDER BY name;
```

Athena 콘솔을 사용하여 추가 분석을 위해 결과를 다운로드하거나 결과를 더 잘 이해할 수 있도록 정렬할 수 있습니다.

Example환자 및 관련 처방에 대한 필터링 조건 생성

환자가 현재 사용 중인 약물 목록을 보는 것이 중요합니다. Athena를 사용하면 HealthLake 데이터 스토어에 있는 `Patient` 및 `MedicationRequest` 리소스 유형을 모두 사용하는 SQL 쿼리를 작성할 수 있습니다.

다음 SQL 쿼리는 Athena로 가져온 `Patient` 및 `MedicationRequest` 테이블을 조인합니다. 또한 점 표기법을 사용하여 처방을 개별 항목으로 구성합니다.

```
SELECT
 patient.id  as patientId
 , medicationrequest.id  as medicationrequestid
 , CONCAT(name[1].family, ' ', name[1].given[1]) as name
 , status
 , statusreason.coding[1].code as categoryCode
 , statusreason.coding[1].display as categoryDescription
 , category[1].coding[1].code as categoryCode
 , category[1].coding[1].display as categoryDescription
 , priority
 , donotperform
 , encounter.reference as encounterId
 , encounter.type as encountertype
 , medicationcodeableconcept.coding[1].code as medicationCode
 , medicationcodeableconcept.coding[1].display as medicationDescription
 , dosageinstruction[1].text as dosage
FROM database.patient, medicationrequest
WHERE CONCAT('Patient/', patient.id ) = medicationrequest.subject.reference
```

```
ORDER BY name
```

Athena 콘솔을 사용하여 결과를 정렬하거나 추가 분석을 위해 다운로드할 수 있습니다.

Example **MedicationStatement** 리소스 유형에서 찾은 약물 보기

다음 예제 쿼리는 SQL을 사용하여 Athena로 가져온 중첩된 JSON을 구성하는 방법을 보여줍니다. 쿼리는 FHIR `meta` 요소를 사용하여 HealthLake의 통합 자연어 처리(NLP)에 의해 의약품이 추가된 시기를 나타냅니다. 또한 `json_extract`를 사용하여 JSON 문자열 배열 내의 데이터를 검색합니다. 자세한 내용은 자연어 처리 단원을 참조하십시오.

```
SELECT
 medicationcodeableconcept.coding[1].code as medicationCode
 , medicationcodeableconcept.coding[1].display as medicationDescription
 , meta.tag[1].display
 , json_extract(modifierextension[1], '$.valueDecimal') AS confidenceScore
FROM medicationstatement;
```

Athena 콘솔을 사용하여 이러한 결과를 다운로드하거나 정렬할 수 있습니다.

Example특정 질병 유형에 대한 필터

이 예제는 당뇨병 진단을 받은 18~75세의 환자 그룹을 찾는 방법을 보여줍니다.

```
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%');
```

이제 Athena 콘솔을 사용하여 결과를 정렬하거나 추가 분석을 위해 다운로드할 수 있습니다.

모니터링 AWS HealthLake

모니터링 및 로깅은의 보안, 신뢰성, 가용성 및 성능을 유지하는 데 중요한 부분입니다 AWS HealthLake.는 HealthLake를 모니터링하고, 이상이 있을 때 이를 보고하고, 필요한 경우 자동 조치를 취할 수 있도록 다음 서비스를 AWS 제공합니다.

- AWS CloudTrail는 AWS 계정에 의해 또는 계정을 대신하여 수행된 API 호출 및 관련 이벤트를 캡처하고 사용자가 지정한 Amazon S3 버킷에 로그 파일을 전송합니다. 호출한 사용자 및 계정 AWS, 호출이 수행된 소스 IP 주소, 호출이 발생한 시기를 식별할 수 있습니다. 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하십시오.
- Amazon CloudWatch는 AWS 리소스와 AWS 에서 실행하는 애플리케이션을 실시간으로 모니터링합니다. 지표를 수집 및 추적하고, 사용자 지정 대시보드를 생성할 수 있으며, 지정된 지표가 지정한 임곗값에 도달하면 사용자에게 알리거나 조치를 취하도록 경보를 설정할 수 있습니다. 예를 들어 CloudWatch에서 Amazon EC2 인스턴스의 CPU 사용량 또는 기타 지표를 추적하고 필요할 때 자동으로 새 인스턴스를 시작할 수 있습니다. 자세한 내용은 Amazon CloudWatch 사용자 안내서를 참조하세요.
- Amazon EventBridge: 애플리케이션을 다양한 소스의 데이터와 쉽게 연결할 수 있는 서버리스 이벤트 버스 서비스입니다. EventBridge는 자체 애플리케이션, 서비스형 소프트웨어(SaaS) 애플리케이션 및 AWS 서비스의 실시간 데이터 스트림을 제공한 다음, 해당 데이터를 Lambda 등의 대상으로 라우팅합니다. 이를 통해 서비스에서 발생하는 이벤트를 모니터링하고 이벤트 기반 아키텍처를 구축할 수 있습니다. 자세한 내용은 Amazon EventBridge 사용 설명서를 참조하세요.

주제

를 사용하여 HealthLake API 호출 로깅 AWS CloudTrail

AWS HealthLake 는 HealthLake에서 사용자 AWS CloudTrail, 역할 또는 서비스가 수행한 작업에 대한 레코드를 제공하는 AWS 서비스와 통합됩니다. CloudTrail은 HealthLake에 대한 모든 API 호출을 이벤트로 캡처합니다. 캡처되는 호출에는 HealthLake 콘솔의 호출과 HealthLake API 작업에 대한 코드 호출이 포함됩니다. 추적을 생성하면 HealthLake 이벤트를 포함한 CloudTrail 이벤트를 지속적으로 Amazon S3 버킷에 배포할 수 있습니다. 추적을 구성하지 않은 경우에도 이벤트 기록에서 CloudTrail

콘솔의 최신 이벤트를 볼 수 있습니다. CloudTrail에서 수집한 정보를 사용하여 HealthLake에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

CloudTrail에 대한 자세한 설명은 AWS CloudTrail 사용자 가이드를 참조하십시오.

AWS HealthLake CloudTrail의 정보

AWS 계정을 생성할 때 계정에서 CloudTrail이 활성화됩니다. HealthLake에서 활동이 발생하면 해당 활동이 이벤트 기록의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. AWS 계정에서 최신 이벤트를 확인, 검색 및 다운로드할 수 있습니다. 자세한 설명은 CloudTrail 이벤트 기록으로 이벤트 보기를 참조하세요.

HealthLake 이벤트를 포함하여 AWS 계정에 이벤트를 지속적으로 기록하려면 추적을 생성합니다. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 콘솔에서 추적을 생성하면 기본적으로 모든 리전에 추적이 적용됩니다. 추적은 AWS 파티션의 모든 리전에서 이벤트를 로깅하고 지정한 Amazon S3 버킷으로 로그 파일을 전송합니다. 또한 CloudTrail 로그에서 수집된 이벤트 데이터를 추가로 분석하고 조치를 취하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음 자료를 참조하세요.

- 추적 생성 개요
- CloudTrail 지원 서비스 및 통합
- CloudTrail에서 Amazon SNS 알림 구성
- 여러 리전으로부터 CloudTrail 로그 파일 받기 및 여러 계정으로부터 CloudTrail 로그 파일 받기

모든 HealthLake 작업은 CloudTrail에서 로깅되며 FHIR REST API를 사용하여 수행되는 작업에 대한 HealthLake API 참조 및이 개발자 안내서에 설명되어 있습니다. 예를 들어 다음 작업에 대한 호출은 CloudTrail 로그 파일에 항목을 생성합니다.

- `DescribeFHIRImportJob`
- `DescribeFHIRExportJob`
- `StartFHIRImportJob`
- `ListFHIRImportJobs`
- `StartFHIRExportJob`
- `ListFHIRExportJobs`
- `CreateFHIRDatastore`

- `ListFHIRDatastores`
- `DeleteFHIRDatastore`
- `DescribeFHIRDatastore`
- `UpdateResource`
- `CreateResource`
- `DeleteResource`
- `ReadResource`
- `GetCapabilities`
- `SearchWithGet`
- `SearchWithPost`

모든 이벤트 또는 로그 항목에는 요청을 생성했던 사용자에 관한 정보가 포함됩니다. ID 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청이 루트 또는 AWS Identity and Access Management (IAM) 사용자 자격 증명으로 이루어졌는지 여부입니다.
- 역할 또는 페더레이션 사용자의 임시 자격 증명을 사용하여 요청이 생성되었는지 여부.
- 요청이 다른 AWS 서비스에서 이루어졌는지 여부입니다.

자세한 설명은 CloudTrail userIdentity 요소를 참조하세요.

AWS HealthLake 로그 파일 항목 이해

트레일이란 지정한 S3 버킷에 이벤트를 로그 파일로 입력할 수 있게 하는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. 이벤트는 모든 소스로부터의 단일 요청을 나타내며 요청 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보가 들어 있습니다. CloudTrail 로그 파일은 퍼블릭 API 직접 호출의 주문 스택 트레이스가 아니므로 특정 순서로 표시되지 않습니다.

다음은 `CreateFHIRDatastore` 작업을 보여주는 CloudTrail 로그 항목이 나타낸 예시입니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA2B3ZHOADD2OJ4AHJX:git
 full_access_iam_role580074395690222150",
```

```
        "arn": "arn:aws:sts::691207106566:assumed-role/
colossusfrontend_full_access_iam_role/_iam_role580074395690222150",
        "accountId": "AccountID",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROA2B3ZHOADD2OJ4AHJX",
                "arn": "arn:aws:iam::691207106566:role/full_access_iam_role",
                "accountId": "AccountID",
                "userName": "full_access_iam_role"
            },
            "webIdFederationData": {

            },
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-11-20T00:08:15Z"
            }
        }
    },
    "eventTime": "2020-11-20T00:08:16Z",
    "eventSource": "healthlake.amazonaws.com",
    "eventName": "CreateFHIRDatastore",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "3.213.247.1",
    "userAgent": "Coral/Netty4",
    "requestParameters": {
        "datastoreName":
 "testCreateFHIRDatastore_GBYAZFCLLBLSUTOYYFQZRLBLQJNFOYQVRPZBOJAIIUAHICAEAGIWLNVQEYAMSXVWMBLXC
        "datastoreTypeVersion": "R4",
        "clientToken": "d737ffe0-14dd-44cc-9f0a-fdf59b26c66b"
    },
    "responseElements": {
        "datastoreId": "datastoreID",
        "datastoreArn": "arn:aws:healthlake:us-
east-1:691207106566:datastore/55576c487ff4975262b10d1d65eb4509",
        "datastoreStatus": "CREATING",
        "datastoreEndpoint": "datastore_endpoint/"
    },
    "requestID": "68e62bdd-d2d4-44c1-af69-e6f055a69f99",
    "eventID": "7ef483dc-5dca-469e-823a-7d9e3a7fe924",
    "readOnly": false,
    "eventType": "AwsApiCall",
```

```
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "691207106566"
}
```

Amazon CloudWatch를 사용하여 HealthLake 지표 모니터링

원시 데이터를 수집하여 읽기 가능하며 실시간에 가까운 지표로 처리하는 Amazon CloudWatch를 사용하여 HealthLake를 모니터링할 수 있습니다. 이러한 통계는 15개월간 보관되므로 기록 정보를 사용하여 웹 애플리케이션 또는 서비스가 어떻게 실행되고 있는지 전체적으로 더 잘 파악할 수 있습니다. 특정 임계값을 주시하다가 해당 임계값이 충족될 때 알림을 전송하거나 조치를 취하도록 경보를 설정할 수도 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하세요.

> **Note**
>
> 지표는 모든 네이티브 HealthLake 작업에 대해 보고됩니다.

다음 표에는 CloudWatch에 보고된 HealthLake 지표 및 차원이 나열되어 있습니다. 각각은 사용자가 지정한 데이터 범위의 빈도 수로 표시됩니다.

다음 HealthLake 지표는 CloudWatch에 보고됩니다.

CloudWatch에 보고된 HealthLake 지표

지표	설명
호출 개수	API에 대한 호출 횟수. 이는 계정 또는 지정된 데이터 스토어에 대해 보고할 수 있습니다. 단위: 개 유효한 통계: Sum, Count 차원: 작업, 데이터 스토어 ID, 데이터 스토어 유형
성공적인 요청	성공한 API 요청 수입니다. 단위: 개

지표	설명
	유효한 통계: Sum, Average 측정기준: 작업, 데이터 스토어 ID, 데이터 스토어 유형
사용자 오류	사용자 오류로 인해 실패한 요청 수입니다. 단위: 개 유효한 통계: Sum, Average 측정기준: 작업, 데이터 스토어 ID, 데이터 스토어 유형
서버 오류	서버 오류로 인해 실패한 요청 수입니다. 단위: 개 유효한 통계: Sum, Average 측정기준: 작업, 데이터 스토어 ID, 데이터 스토어 유형
조정된(throttle) 요청	병목 현상이 발생한 요청 수입니다. 이 지표는 사용자 또는 서버 오류 수에 포함되지 않습니다. 단위: 개 유효한 통계: Sum, Average 측정기준: 작업, 데이터 스토어 ID, 데이터 스토어 유형

지표	설명
Latency	사용자 요청을 처리하는 데 밀리초 단위로 걸린 시간입니다. 단위: 밀리초 유효 통계: Minimum, Maximum, Average, Sum 측정기준: 작업, 데이터 스토어 ID, 데이터 스토어 유형

다음 HealthLake 차원은 CloudWatch에 보고됩니다.

CloudWatch에 보고된 HealthLake 차원

차원	설명
연산	요청에 사용되는 API 작업
DataStoreID	요청에 사용되는 데이터 스토어 ID입니다.
DataStoreType	요청에 사용되는 데이터 스토어 유형

AWS Management Console AWS CLI, 또는 CloudWatch API를 사용하여 HealthLake에 대한 지표를 가져올 수 있습니다. Amazon AWS 소프트웨어 개발 키트(SDK) 또는 CloudWatch API 도구 중 하나를 통해 CloudWatch API를 사용할 수 있습니다. HealthLake 콘솔에는 CloudWatch API의 원시 데이터를 기반으로 그래프가 표시됩니다.

CloudWatch를 사용하여 HealthLake를 모니터링하려면 적절한 CloudWatch 권한이 있어야 합니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 Amazon CloudWatch에 대한 자격 증명 및 액세스 관리를 참조하세요. Amazon CloudWatch

HealthLake 지표 보기

지표 보기(CloudWatch 콘솔)

1. 에 로그인 AWS Management Console 하고 CloudWatch 콘솔을 엽니다.
2. 지표를 선택하고 모든 지표를 선택한 다음 AWS/HealthLake를 선택합니다.

3. 차원과 지표 이름을 선택한 다음 그래프에 추가를 선택합니다.
4. 날짜 범위 값을 선택합니다. 선택한 날짜 범위에 대한 지표 개수가 그래프에 표시됩니다.

CloudWatch를 사용하여 경보 생성

CloudWatch 경보는 지정된 기간 동안 단일 지표를 감시하고 Amazon Simple Notification Service(SNS) 주제 또는 Auto Scaling 정책에 알림을 보내는 하나 이상의 작업을 수행합니다. 이러한 작업은 지정한 여러 기간 동안 지정된 임계값에 따른 지표의 값을 기반으로 합니다. CloudWatch는 경보 상태가 변경될 때 SNS 메시지를 보낼 수도 있습니다.

> **Note**
>
> CloudWatch 경보는 상태가 변경되어 지정한 기간 동안 지속되는 경우에만 작업을 호출합니다.

지표 보기(CloudWatch 콘솔)

1. CloudWatch 콘솔에 로그인합니다.
2. 알람를 선택한 다음 알람 생성을 선택합니다.
3. AWS/HealthLake를 선택한 다음 지표를 선택합니다.
4. 시간 범위에서 모니터링할 시간 범위를 선택한 후, 다음를 선택합니다.
5. 이름 및 설명을 입력합니다.
6. Whenever에서 >=를 선택하고 최대값을 입력합니다.
7. 경보 상태에 도달했을 때 CloudWatch에서 이메일을 보내도록 하려면 작업 섹션의이 경보가 발생할 때마다 상태가 경보인 경우를 선택합니다. 알림 전송 대상에서 메일링 목록을 선택하거나 새 목록을 선택하고 새 메일링 목록을 생성합니다.
8. 알람 미리보기 섹션에서 경보를 미리 볼 수 있습니다. 경보가 만족스러우면 경보 생성을 선택합니다.

Amazon EventBridge를 사용하여 HealthLake 이벤트 모니터링

Amazon EventBridge는 이벤트를 사용하여 애플리케이션 구성 요소를 서로 연결하는 서버리스 서비스로, 확장 가능한 이벤트 기반 애플리케이션을 더 쉽게 구축할 수 있습니다. EventBridge의 기본은 이벤트를 대상으로 라우팅하는 규칙을 생성하는 것입니다. AWS HealthLake 는 상태 변경을 EventBridge

로 지속적으로 전달합니다. 자세한 내용은 Amazon EventBridge 사용 설명서의 Amazon EventBridge란? 섹션을 참조하세요.

> **Note**
>
> HealthLake 이벤트를 Amazon EventBridge로 보내는 방법을 알아보려면 AWS for Industry 블로그의 용 Amazon EventBridge 통합 AWS HealthLake을 참조하세요.

주제

EventBridge로 전송된 HealthLake 이벤트

다음 표에는 처리를 위해 EventBridge로 전송된 모든 HealthLake 이벤트가 나열되어 있습니다.

HealthLake 이벤트 유형	State
데이터 스토어 이벤트	
데이터 스토어 생성	`CREATING`
데이터 스토어 활성	`ACTIVE`
데이터 스토어 삭제	`DELETING`
데이터 스토어 삭제됨	`DELETED`
데이터 스토어 생성 실패	`CREATE_FAILED`
자세한 내용을 알아보려면 AWS HealthLake API 참조의 *`datastoreStatus`* 섹션을 참조하세요.	
작업 이벤트 가져오기	
가져오기 작업 제출됨	`SUBMITTED`
가져오기 작업 진행 중	`IN_PROGRESS`

HealthLake 이벤트 유형	State
오류로 완료된 가져오기 작업	`COMPLETED_WITH_ERRORS`
가져오기 작업 완료	`COMPLETED`
가져오기 작업 실패	`FAILED`
자세한 내용을 알아보려면 AWS HealthLake API 참조의 *`jobStatus`* 섹션을 참조하세요.	
작업 이벤트 내보내기	
내보내기 작업 제출됨	`SUBMITTED`
내보내기 작업 진행 중	`IN_PROGRESS`
오류로 완료된 내보내기 작업	`COMPLETED_WITH_ERRORS`
내보내기 작업 완료	`COMPLETED`
내보내기 작업 실패	`FAILED`
자세한 내용을 알아보려면 AWS HealthLake API 참조의 *`jobStatus`* 섹션을 참조하세요.	

HealthLake 이벤트 구조

HealthLake 이벤트는 메타데이터 세부 정보도 포함하는 JSON 구조의 객체입니다. 메타데이터를 입력으로 사용하여 이벤트를 다시 생성하거나 자세한 내용을 알아볼 수 있습니다. 연결된 모든 메타데이터 필드는 다음 메뉴의 코드 예제 아래 테이블에 나열됩니다. 자세한 내용은 Amazon EventBridge 사용 설명서의 AWS 서비스 이벤트 메타데이터를 참조하세요.

> **Note**
>
> HealthLake 이벤트를 Amazon EventBridge로 보내는 방법을 알아보려면 for AWS Industry 블로그의 용 Amazon EventBridge 통합 AWS HealthLake을 참조하세요.

데이터 스토어 이벤트

Data Store Creating

상태 - **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

상태 - **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

상태 - **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

상태 - **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/"
    }
}
```

데이터 스토어 이벤트 - 메타데이터 설명

이름	Type	설명
`version`	문자열	EventBridge 이벤트 스키마 버전입니다.
`id`	문자열	모든 이벤트에 대해 생성된 버전 4 UUID입니다.
`detail-type`	문자열	전송 중인 이벤트의 유형입니다.
`source`	문자열	이벤트를 생성한 서비스를 식별합니다.

이름	Type	설명
account	문자열	데이터 스토어 소유자의 12자리 AWS 계정 ID입니다.
time	문자열	이벤트가 발생한 시간입니다.
region	문자열	데이터 스토어의 AWS 리전을 식별합니다.
resources	배열(문자열)	데이터 스토어의 ARN을 포함하는 JSON 배열입니다.
detail	객체	이벤트에 대한 정보를 포함하는 JSON 객체입니다.
detail.datastoreId	문자열	상태 변경 이벤트와 연결된 데이터 스토어 ID입니다.
detail.datastoreName	문자열	데이터 스토어 이름입니다.
detail.datastoreTypeVersion	문자열	데이터 스토어 FHIR 버전입니다.
detail.datastoreEndpoint	문자열	데이터 스토어 엔드포인트입니다.

작업 이벤트 가져오기

Import Job Submitted

상태 - **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

상태 - **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

상태 - **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

상태 - **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

상태 - **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/"
        }
```

```
    }
}
```

작업 이벤트 가져오기 - 메타데이터 설명

이름	Type	설명
version	문자열	EventBridge 이벤트 스키마 버전입니다.
id	문자열	모든 이벤트에 대해 생성된 버전 4 UUID입니다.
detail-type	문자열	전송 중인 이벤트의 유형입니다.
source	문자열	이벤트를 생성한 서비스를 식별합니다.
account	문자열	데이터 스토어 소유자의 12자리 AWS 계정 ID입니다.
time	문자열	이벤트가 발생한 시간입니다.
region	문자열	데이터 스토어의 AWS 리전을 식별합니다.
resources	배열(문자열)	데이터 스토어의 ARN을 포함하는 JSON 배열입니다.
detail	객체	이벤트에 대한 정보를 포함하는 JSON 객체입니다.
detail.jobId	문자열	상태 변경 이벤트와 연결된 가져오기 작업 ID입니다.
detail.submitTime	문자열	가져오기 작업이 제출된 시간입니다.

이름	Type	설명
detail.datastoreId	문자열	상태 변경 이벤트를 생성한 데이터 스토어입니다.
detail.inputDataConfig	문자열	가져올 FHIR 파일이 포함된 Amazon S3 버킷의 입력 접두사 경로입니다.

작업 이벤트 내보내기

Export Job Submitted

상태 - **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

상태 - **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

상태 - **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

상태 - **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

상태 - **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/"
        }
    }
}
```

작업 이벤트 내보내기 - 메타데이터 설명

이름	Type	설명
`version`	문자열	EventBridge 이벤트 스키마 버전입니다.
`id`	문자열	모든 이벤트에 대해 생성된 버전 4 UUID입니다.
`detail-type`	문자열	전송 중인 이벤트의 유형입니다.

이름	Type	설명
source	문자열	이벤트를 생성한 서비스를 식별합니다.
account	문자열	데이터 스토어 소유자의 12자리 AWS 계정 ID입니다.
time	문자열	이벤트가 발생한 시간입니다.
region	문자열	데이터 스토어의 AWS 리전을 식별합니다.
resources	배열(문자열)	데이터 스토어의 ARN을 포함하는 JSON 배열입니다.
detail	객체	이벤트에 대한 정보를 포함하는 JSON 객체입니다.
detail.jobId	문자열	상태 변경 이벤트와 연결된 내보내기 작업 ID입니다.
detail.submitTime	문자열	내보내기 작업이 제출된 시간입니다.
detail.datastoreId	문자열	상태 변경 이벤트를 생성한 데이터 스토어입니다.
detail.outputDataConfig	문자열	내보낼 FHIR 파일이 포함된 Amazon S3 버킷의 출력 접두사 경로입니다.

AWS HealthLake의 보안

의 클라우드 보안 AWS 이 최우선 순위입니다. AWS 고객은 보안에 가장 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 이점을 누릴 수 있습니다.

보안은 AWS 와 사용자 간의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 AWS 은 AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호하는 역할을 합니다. AWS 또한는 안전하게 사용할 수 있는 서비스를 제공합니다. 타사 감사자는 AWS 규정 준수 프로그램 일환으로 보안의 효과를 정기적으로 테스트하고 확인합니다. HealthLake에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 프로그램 제공 범위 내 AWS 서비스규정 준수 프로그램 참조하세요.
- 클라우드의 보안 - 사용자의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 귀하는 귀사의 데이터 민감도, 귀사의 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 HealthLake를 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 주제에서는 보안 및 규정 준수 목표에 맞게 HealthLake를 구성하는 방법을 보여줍니다. 또한 HealthLake 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법을 알아봅니다.

주제

AWS HealthLake의 데이터 보호

AWS 공동 책임 모델 AWS HealthLake의 데이터 보호에 적용됩니다. 이 모델에 설명된 대로 AWS 는 모든를 실행하는 글로벌 인프라를 보호할 책임이 있습니다 AWS 클라우드. 사용자는 이 인프라에 호스팅되는 콘텐츠에 대한 통제 권한을 유지할 책임이 있습니다. 사용하는 AWS 서비스 의 보안 구성과 관리 태스크에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 관한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

데이터 보호를 위해 자격 증명을 보호하고 AWS 계정 AWS IAM Identity Center 또는 AWS Identity and Access Management (IAM)를 사용하여 개별 사용자를 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용합니다.
- SSL/TLS를 사용하여 AWS 리소스와 통신합니다. TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다 AWS CloudTrail. CloudTrail 추적을 사용하여 AWS 활동을 캡처하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 추적 작업을 참조하세요.
- 내부의 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용합니다 AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용합니다.
- 명령줄 인터페이스 또는 API를 AWS 통해에 액세스할 때 FIPS 140-3 검증 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 연방 정보 처리 표준(FIPS) 140-3을 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 형식 텍스트 필드에 입력하지 않는 것이 좋습니다. 여기에는 HealthLake 또는 기타 AWS 서비스 에서 콘솔 AWS CLI, API 또는 AWS SDKs를 사용하여 작업하는 경우가 포함됩니다. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 자격 증명을 URL에 포함해서는 안 됩니다.

REST for AWS HealthLake에서의 암호화

HealthLake는 서비스 소유 AWS Key Management Service(AWS KMS) 키를 사용하여 저장 시 민감한 고객 데이터를 보호하기 위해 기본적으로 암호화를 제공합니다. 고객 관리형 KMS 키도 지원되며 데이

터 스토어에서 파일을 가져오고 내보내는 데 필요합니다. 고객 관리형 KMS 키에 대한 자세한 내용은 Amazon Key Management Service를 참조하세요. 고객은 데이터 스토어를 생성할 때 AWS 소유 KMS 키 또는 고객 관리형 KMS 키를 선택할 수 있습니다. 데이터 스토어가 생성된 후에는 암호화 구성을 변경할 수 없습니다. 데이터 스토어가 AWS 소유 KMS 키를 사용하는 경우 AWS_OWNED_KMS_KEY로 표시되며 저장 시 암호화에 사용되는 특정 키가 표시되지 않습니다.

AWS 소유 KMS 키

HealthLake는 기본적으로 이러한 키를 사용하여 개인 식별 또는 개인 건강 정보(PHI) 저장 데이터와 같은 잠재적으로 민감한 정보를 자동으로 암호화합니다. AWS 소유 KMS 키는 계정에 저장되지 않습니다. 이는 AWS가 여러 AWS 계정에서 사용하기 위해 소유하고 관리하는 KMS 키 모음의 일부입니다. AWS 서비스는 AWS 소유 KMS 키를 사용하여 데이터를 보호할 수 있습니다. AWS 소유 KMS 키를 보거나, 관리하거나, 사용하거나, 사용을 감사할 수 없습니다. 하지만 사용자는 데이터를 암호화하는 키를 보호하기 위해 어떤 작업을 수행하거나 어떤 프로그램을 변경할 필요가 없습니다.

AWS 소유 KMS 키를 사용하는 경우 월별 요금 또는 사용 요금이 부과되지 않으며 계정에 대한 AWS KMS 할당량에 포함되지 않습니다. 자세한 내용은 AWS 소유 키를 참조하세요.

고객 관리형 KMS 키

HealthLake는 사용자가 생성, 소유 및 관리하는 대칭 고객 관리형 KMS 키를 사용하여 기존 AWS 소유 암호화에 두 번째 암호화 계층을 추가할 수 있도록 지원합니다. 사용자가 이 암호화 계층을 완전히 제어할 수 있으므로 다음과 같은 작업을 수행할 수 있습니다.

- 키 정책, IAM 정책 및 권한 부여 설정 및 유지 관리
- 키 암호화 자료 교체
- 키 정책 활성화 및 비활성화
- 태그 추가
- 키 별칭 만들기
- 삭제를 위한 스케줄 키

CloudTrail을 사용하여 HealthLake가 사용자를 대신하여에 보내는 요청을 추적할 수도 AWS KMS 있습니다. 추가 AWS KMS 요금이 적용됩니다. 자세한 내용은 고객 소유 키를 참조하세요.

고객 관리형 키 생성

AWS Management Console 또는 AWS KMS APIs.

AWS Key Management Service 개발자 안내서의 대칭 고객 관리형 키 생성 단계를 따릅니다.

키 정책에서는 고객 관리형 키에 대한 액세스를 제어합니다. 모든 고객 관리형 키에는 키를 사용할 수 있는 사람과 키를 사용하는 방법을 결정하는 문장이 포함된 정확히 하나의 키 정책이 있어야 합니다. 고객 관리형 키를 만들 때 키 정책을 지정할 수 있습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서의 고객 관리형 키에 대한 액세스 관리를 참조하세요.

HealthLake 리소스와 함께 고객 관리형 키를 사용하려면 키 정책에서 kms:CreateGrant 작업을 허용해야 합니다. 이렇게 하면 지정된 KMS 키에 대한 액세스를 제어하는 고객 관리형 키에 권한 부여가 추가되어 HealthLake에서 요구하는 kms:grant 작업에 대한 액세스 권한이 사용자에게 부여됩니다. 자세한 내용은 권한 부여 사용을 참조하세요.

HealthLake 리소스와 함께 고객 관리형 KMS 키를 사용하려면 키 정책에서 다음 API 작업을 허용해야 합니다.

- kms:CreateGrant는 권한 부여 작업에 대한 액세스를 허용하는 특정 고객 관리형 KMS 키에 권한 부여를 추가합니다.
- kms:DescribeKey는 키를 검증하는 데 필요한 고객 관리형 키 세부 정보를 제공합니다. 이것은 모든 작업에 필요합니다.
- kms:GenerateDataKey는 모든 쓰기 작업에 대해 저장된 리소스를 암호화할 수 있는 액세스를 제공합니다.
- kms:Decrypt는 암호화된 리소스에 대한 읽기 또는 검색 작업에 대한 액세스를 제공합니다.

다음은 사용자가 해당 키로 암호화된 AWS HealthLake에서 데이터 스토어를 생성하고 상호 작용할 수 있도록 허용하는 정책 설명 예제입니다.

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

고객 관리형 KMS 키를 사용할 때 필요한 IAM 권한

고객 관리형 KMS 키를 사용하여 AWS KMS 암호화가 활성화된 데이터 스토어를 생성하는 경우 HealthLake 데이터 스토어를 생성하는 사용자 또는 역할에 대한 키 정책 및 IAM 정책 모두에 필요한 권한이 있습니다.

kms:ViaService 조건 키를 사용하여 KMS 키 사용을 HealthLake에서 시작된 요청으로만 제한할 수 있습니다.

키 정책에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 IAM 정책 활성화를 참조하세요.

리포지토리를 생성하는 IAM 사용자, IAM 역할 또는 AWS 계정에는 kms:CreateGrant,kms:GenerateDataKey 및 kms:DescribeKey 권한과 필요한 HealthLake 권한이 있어야 합니다.

HealthLake가 AWS KMS에서 권한 부여를 사용하는 방법

HealthLake에서 고객 관리형 KMS 키를 사용하려면 권한 부여가 필요합니다. 고객 관리형 KMS 키로 암호화된 데이터 스토어를 생성하면 HealthLake는 AWS KMS에 CreateGrant 요청을 전송하여 사용자를 대신하여 권한을 생성합니다. AWS KMS의 권한 부여는 HealthLake에 고객 계정의 KMS 키에 대한 액세스 권한을 부여하는 데 사용됩니다.

HealthLake가 사용자를 대신하여 생성하는 권한 부여는 취소되거나 사용 중지되어서는 안 됩니다. 계정에서 AWS KMS 키를 사용할 수 있는 권한을 HealthLake에 부여하는 권한 부여를 취소하거나 사용 중지하면 HealthLake는이 데이터에 액세스하거나, 데이터 스토어로 푸시된 새 FHIR 리소스를 암호화하거나, 폴링될 때 복호화할 수 없습니다. HealthLake에 대한 권한 부여를 취소하거나 사용 중지하면 변경 사항이 즉시 적용됩니다. 액세스 권한을 취소하려면 권한 부여를 취소하는 대신 데이터 스토어를

삭제해야 합니다. 데이터 스토어가 삭제되면 HealthLake는 사용자를 대신하여 권한 부여를 사용 중지합니다.

HealthLake에 대한 암호화 키 모니터링

CloudTrail을 사용하여 고객 관리형 KMS 키를 사용할 때 HealthLake가 사용자를 대신하여 AWS KMS 에 보내는 요청을 추적할 수 있습니다. CloudTrail 로그의 로그 항목은 userAgent 필드에 healthlake.amazonaws.com 표시하여 HealthLake의 요청을 명확하게 구분합니다.

다음 예제는 고객 관리형 키로 암호화된 데이터에 액세스하기 위해 HealthLake에서 호출한 AWS KMS 작업을 모니터링하기 위한 CreateGrant, GenerateDataKey, Decrypt 및 DescribeKey에 대한 CloudTrail 이벤트입니다.

다음은 CreateGrant를 사용하여 HealthLake가 고객 제공 KMS 키에 액세스하여 HealthLake가 해당 KMS 키를 사용하여 저장된 모든 고객 데이터를 암호화하도록 허용하는 방법을 보여줍니다.

사용자가 자신의 권한 부여를 만들 필요는 없습니다. HealthLake는 AWS KMS에 CreateGrant 요청을 전송하여 사용자를 대신하여 권한 부여를 생성합니다. 의 권한 부여 AWS KMS 는 HealthLake에 고객 계정의 AWS KMS 키에 대한 액세스 권한을 부여하는 데 사용됩니다.

```
         {
"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"
}
```

다음 예제에서는 GenerateDataKey를 사용하여 사용자가 데이터를 저장하기 전에 암호화하는 데 필요한 권한을 갖도록 하는 방법을 보여줍니다.

```
         {
"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"
}
```

다음 예제에서는 HealthLake가 Decrypt 작업을 호출하여 저장된 암호화된 데이터 키를 사용하여 암호화된 데이터에 액세스하는 방법을 보여줍니다.

```
         {
    "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"
}
```

다음 예제에서는 HealthLake가 DescribeKey 작업을 사용하여 AWS KMS 고객 소유 AWS KMS 키가 사용 가능한 상태인지 확인하고 작동하지 않는 경우 사용자가 문제를 해결할 수 있도록 지원하는 방법을 보여줍니다.

```
            {
    "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"
}
```


자세히 알아보기

다음 리소스는 저장 데이터 암호화에 대한 자세한 정보를 제공합니다.

AWS Key Management Service 기본 개념에 대한 자세한 내용은 AWS KMS 설명서를 참조하세요.

AWS KMS 설명서의 보안 모범 사례에 대한 자세한 내용을 참조하세요.

전송 중 암호화 for AWS HealthLake

AWS HealthLake는 TLS 1.2를 사용하여 퍼블릭 엔드포인트와 백엔드 서비스를 통해 전송 중인 데이터를 암호화합니다.

AWS HealthLake의 자격 증명 및 액세스 관리

AWS Identity and Access Management (IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어하는 데 도움이 AWS 서비스 되는 입니다. IAM 관리자는 HealthLake 리소스를 사용할 수 있는 인증(로그인) 및 권한(권한 있음)을 받을 수 있는 사용자를 제어합니다. IAM은 추가 비용 없이 사용할 수 AWS 서비스 있는 입니다.

주제

대상

AWS Identity and Access Management (IAM)를 사용하는 방법은 역할에 따라 다릅니다.

- 서비스 사용자 - 기능에 액세스할 수 없는 경우 관리자에게 권한 요청(참조AWS HealthLake 자격 증명 및 액세스 문제 해결)
- 서비스 관리자 - 사용자 액세스 결정 및 권한 요청 제출(AWS HealthLake와 IAM의 작동 방식 참조)
- IAM 관리자 - 액세스를 관리하기 위한 정책 작성(AWS HealthLake의 자격 증명 기반 정책 예제 참조)

ID를 통한 인증

인증은 자격 증명 자격 증명을 AWS 사용하여에 로그인하는 방법입니다. AWS 계정 루트 사용자, IAM 사용자 또는 IAM 역할을 수임하여 인증되어야 합니다.

AWS IAM Identity Center (IAM Identity Center), Single Sign-On 인증 또는 Google/Facebook 자격 증명과 같은 자격 증명 소스의 자격 증명을 사용하여 페더레이션 자격 증명으로 로그인할 수 있습니다. 로그인하는 방법에 대한 자세한 내용은 AWS Sign-In 사용 설명서의 AWS 계정에 로그인하는 방법 섹션을 참조하세요.

프로그래밍 방식 액세스를 위해는 요청에 암호화 방식으로 서명할 수 있는 SDK 및 CLI를 AWS 제공합니다. 자세한 내용은 IAM 사용 설명서의 API 요청용AWS Signature Version 4 섹션을 참조하세요.

AWS 계정 루트 사용자

를 생성할 때 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 AWS 계정 theroot 사용자라는 하나의 로그인 자격 증명으로 AWS 계정시작합니다. 일상적인 태스크에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 자격 증명이 필요한 작업은 IAM 사용 설명서의 루트 사용자 자격 증명이 필요한 작업 섹션을 참조하세요.

페더레이션 ID

가장 좋은 방법은 인간 사용자에게 자격 증명 공급자와의 페더레이션을 사용하여 임시 자격 증명을 AWS 서비스 사용하여에 액세스하도록 요구하는 것입니다.

페더레이션 자격 증명은 엔터프라이즈 디렉터리, 웹 자격 증명 공급자 또는 자격 증명 소스의 자격 증명을 AWS 서비스 사용하여 Directory Service 에 액세스하는 사용자입니다. 페더레이션 ID는 임시 자격 증명을 제공하는 역할을 수임합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center를 추천합니다. 자세한 정보는 AWS IAM Identity Center 사용 설명서의 What is IAM Identity Center?를 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 단일 개인 또는 애플리케이션에 대한 특정 권한을 가진 ID입니다. 장기 자격 증명이 있는 IAM 사용자 대신 임시 자격 증명을 사용하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 자격 증명 공급자와의 연동을 사용하여 임시 자격 증명을 AWS 사용하여에 액세스하도록 인간 사용자에게 요구하기를 참조하세요.

IAM 그룹은 IAM 사용자 모음을 지정하고 대규모 사용자 집합에 대한 관리 권한을 더 쉽게 만듭니다. 자세한 내용은 IAM 사용 설명서의 IAM 사용자 사용 사례 섹션을 참조하세요.

IAM 역할

IAM 역할은 임시 자격 증명을 제공하는 특정 권한이 있는 자격 증명입니다. 사용자에서 IAM 역할(콘솔)로 전환하거나 또는 API 작업을 호출하여 역할을 수임할 수 있습니다. AWS CLI AWS 자세한 내용은 IAM 사용 설명서의 역할 수임 방법을 참조하세요.

IAM 역할은 페더레이션 사용자 액세스, 임시 IAM 사용자 권한, 교차 계정 액세스, 교차 서비스 액세스 및 Amazon EC2에서 실행되는 애플리케이션에 유용합니다. 자세한 내용은 IAM 사용 설명서의 교차 계정 리소스 액세스를 참조하세요.

정책을 사용하여 액세스 관리

정책을 AWS 생성하고 자격 증명 또는 리소스에 연결하여 AWS 에서 액세스를 제어합니다. 정책은 자격 증명 또는 리소스와 연결될 때 권한을 정의합니다.는 보안 주체가 요청할 때 이러한 정책을 AWS 평가합니다. 대부분의 정책은에 JSON 문서 AWS 로 저장됩니다. JSON 정책 문서에 대한 자세한 내용은 IAM 사용 설명서의 JSON 정책 개요 섹션을 참조하세요.

정책을 사용하여 관리자는 어떤 보안 주체가 어떤 리소스에 대해 어떤 조건에서 작업을 수행할 수 있는지 정의하여 누가 무엇을 액세스할 수 있는지 지정합니다.

기본적으로 사용자 및 역할에는 어떠한 권한도 없습니다. IAM 관리자는 IAM 정책을 생성하고 사용자가 수임할 수 있는 역할에 추가합니다. IAM 정책은 작업을 수행하기 위해 사용하는 방법과 관계없이 작업에 대한 권한을 정의합니다.

ID 기반 정책

ID 기반 정책은 ID(사용자, 사용자 그룹 또는 역할)에 연결하는 JSON 권한 정책 문서입니다. 이러한 정책은 자격 증명이 수행할 수 있는 작업, 대상 리소스 및 이에 관한 조건을 제어합니다. ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

ID 기반 정책은 인라인 정책(단일 ID에 직접 포함) 또는 관리형 정책(여러 ID에 연결된 독립 실행형 정책)일 수 있습니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책 및 인라인 정책 중에서 선택 섹션을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 예를 들어 IAM 역할 신뢰 정책 및 Amazon S3 버킷 정책이 있습니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한

정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책에서는 IAM의 AWS 관리형 정책을 사용할 수 없습니다.

기타 정책 유형

AWS 는 보다 일반적인 정책 유형에서 부여한 최대 권한을 설정할 수 있는 추가 정책 유형을 지원합니다.

- 권한 경계 - ID 기반 정책에서 IAM 엔터티에 부여할 수 있는 최대 권한을 설정합니다. 자세한 정보는 IAM 사용 설명서의 IAM 엔터티의 권한 범위를 참조하세요.
- 서비스 제어 정책(SCP) - AWS Organizations내 조직 또는 조직 단위에 대한 최대 권한을 지정합니다. 자세한 내용은AWS Organizations 사용 설명서의 서비스 제어 정책을 참조하세요.
- 리소스 제어 정책(RCP) – 계정의 리소스에 사용할 수 있는 최대 권한을 설정합니다. 자세한 내용은 AWS Organizations 사용 설명서의 리소스 제어 정책(RCP)을 참조하세요.
- 세션 정책 – 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 자세한 내용은 IAM 사용 설명서의 세션 정책을 참조하세요.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 에서 여러 정책 유형이 관련될 때 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하세요.

AWS HealthLake와 IAM의 작동 방식

IAM을 사용하여 HealthLake에 대한 액세스를 관리하기 전에 HealthLake에서 사용할 수 있는 IAM 기능에 대해 알아봅니다.

AWS HealthLake와 함께 사용할 수 있는 IAM 기능

IAM 특성	HealthLake 지원
자격 증명 기반 정책	예
리소스 기반 정책	아니요

IAM 특성	HealthLake 지원
정책 작업	예
정책 리소스	예
정책 조건 키	예
ACL	아니요
ABAC(정책의 태그)	예
임시 보안 인증	예
엔터티 권한	예
서비스 역할	예
서비스 연결 역할	아니요

HealthLake 및 기타 AWS 서비스가 대부분의 IAM 기능과 작동하는 방법을 개괄적으로 알아보려면 IAM 사용 설명서의 AWS IAM으로 작업하는 서비스를 참조하세요.

AWS HealthLake에 대한 자격 증명 기반 정책

ID 기반 정책 지원: 예

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자 및 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

IAM ID 기반 정책을 사용하면 허용되거나 거부되는 작업과 리소스뿐 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. JSON 정책에서 사용할 수 있는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

AWS HealthLake의 자격 증명 기반 정책 예제

HealthLake 자격 증명 기반 정책의 예를 보려면 섹션을 참조하세요AWS HealthLake의 자격 증명 기반 정책 예제.

AWS HealthLake 내의 리소스 기반 정책

리소스 기반 정책 지원: 아니요

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예제는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는가 포함될 수 있습니다 AWS 서비스.

교차 계정 액세스를 활성화하려는 경우, 전체 계정이나 다른 계정의 IAM 개체를 리소스 기반 정책의 보안 주체로 지정할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM에서 교차 계정 리소스 액세스를 참조하세요.

AWS HealthLake에 대한 정책 작업

정책 작업 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

JSON 정책의 `Action`요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 작업을 설명합니다. 연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하세요.

HealthLake 작업 목록을 보려면 서비스 승인 참조의AWS HealthLake에서 정의한 작업을 참조하세요.

HealthLake의 정책 작업은 작업 앞에 다음 접두사를 사용합니다.

```
healthlake
```

단일 문에서 여러 작업을 지정하려면 각 작업을 쉼표로 구분합니다.

```
"Action": [
      "healthlake:action1",
      "healthlake:action2"
         ]
```

HealthLake 자격 증명 기반 정책의 예를 보려면 섹션을 참조하세요AWS HealthLake의 자격 증명 기반 정책 예제.

AWS HealthLake에 대한 정책 리소스

정책 리소스 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Resource` JSON 정책 요소는 작업이 적용되는 하나 이상의 객체를 지정합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

HealthLake 리소스 유형 및 해당 ARNs 목록을 보려면 서비스 승인 참조의AWS HealthLake에서 정의한 리소스를 참조하세요. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면AWS HealthLake에서 정의한 작업을 참조하세요.

HealthLake 자격 증명 기반 정책의 예를 보려면 섹션을 참조하세요AWS HealthLake의 자격 증명 기반 정책 예제.

AWS HealthLake에 사용되는 정책 조건 키

서비스별 정책 조건 키 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Condition` 요소는 정의된 기준에 따라 문이 실행되는 시기를 지정합니다. 같음(equals) 또는 미만(less than)과 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.

HealthLake 조건 키 목록을 보려면 서비스 승인 참조의 AWS HealthLake에 사용되는 조건 키를 참조하세요. 조건 키를 사용할 수 있는 작업과 리소스를 알아보려면AWS HealthLake에서 정의한 작업을 참조하세요.

HealthLake 자격 증명 기반 정책의 예를 보려면 섹션을 참조하세요AWS HealthLake의 자격 증명 기반 정책 예제.

AWS HealthLake의 액세스 제어 목록(ACLs)

ACL 지원: 아니요

액세스 제어 목록(ACL)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACL은 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

AWS HealthLake를 사용한 속성 기반 액세스 제어(ABAC)

ABAC 지원(정책의 태그): 예

속성 기반 액세스 제어(ABAC)는 태그라고 불리는 속성을 기반으로 권한을 정의하는 권한 부여 전략입니다. IAM 엔터티 및 AWS 리소스에 태그를 연결한 다음 보안 주체의 태그가 리소스의 태그와 일치할 때 작업을 허용하는 ABAC 정책을 설계할 수 있습니다.

태그에 근거하여 액세스를 제어하려면 `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`또는 `aws:TagKeys`조건 키를 사용하여 정책의 조건 요소에 태그 정보를 제공합니다.

서비스가 모든 리소스 유형에 대해 세 가지 조건 키를 모두 지원하는 경우, 값은 서비스에 대해 예입니다. 서비스가 일부 리소스 유형에 대해서만 세 가지 조건 키를 모두 지원하는 경우, 값은 부분적입니다.

ABAC에 대한 자세한 내용은 IAM 사용 설명서의 ABAC 권한 부여를 통한 권한 정의를 참조하세요. ABAC 설정 단계가 포함된 자습서를 보려면 IAM 사용 설명서의 속성 기반 액세스 제어(ABAC) 사용을 참조하세요.

AWS HealthLake에서 임시 자격 증명 사용

임시 자격 증명 지원: 예

임시 자격 증명은 AWS 리소스에 대한 단기 액세스를 제공하며 페더레이션 또는 전환 역할을 사용할 때 자동으로 생성됩니다. 장기 액세스 키를 사용하는 대신 임시 자격 증명을 동적으로 생성하는 것이 AWS 좋습니다. 자세한 내용은 IAM 사용 설명서의 IAM의 임시 보안 자격 증명 및 IAM으로 작업하는 AWS 서비스 섹션을 참조하세요.

AWS HealthLake에 대한 교차 서비스 보안 주체 권한

전달 액세스 세션(FAS) 지원: 예

전달 액세스 세션(FAS)은를 호출하는 보안 주체의 권한을 다운스트림 서비스에 대한 요청 AWS 서비스 과 AWS 서비스함께 사용합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.

AWS HealthLake의 서비스 역할

서비스 역할 지원: 예

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하는 것으로 가정하는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 AWS 서비스 AWS에 권한을 위임할 역할 생성을 참조하세요.

AWS HealthLake에 대한 전체 액세스에 필요한 서비스 역할 및 인라인 정책에 대한 자세한 내용은 섹션을 참조하세요설 AWS HealthLake정.

> **Warning**
>
> 서비스 역할에 대한 권한을 변경하면 HealthLake 기능이 중단될 수 있습니다. HealthLake가 관련 지침을 제공하는 경우에만 서비스 역할을 편집합니다.

AWS HealthLake의 서비스 연결 역할

서비스 연결 역할 지원: 아니요

서비스 연결 역할은에 연결된 서비스 역할의 한 유형입니다 AWS 서비스. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은에 표시 AWS 계정 되며 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

서비스 연결 역할 생성 또는 관리에 대한 자세한 내용은 IAM으로 작업하는AWS 서비스를 참조하세요. 서비스 연결 역할 열에서 `Yes`가 포함된 서비스를 테이블에서 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예(Yes) 링크를 선택합니다.

AWS HealthLake의 자격 증명 기반 정책 예제

기본적으로 사용자 및 역할에는 HealthLake 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성(콘솔)을 참조하세요.

각 리소스 유형에 대한 ARNs 형식을 포함하여 HealthLake에서 정의한 작업 및 리소스 유형에 대한 자세한 내용은 서비스 승인 참조의 AWS HealthLake에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

주제

정책 모범 사례

자격 증명 기반 정책에 따라 계정에서 사용자가 HealthLake 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. ID 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따르세요.

- AWS 관리형 정책을 시작하고 최소 권한으로 전환 - 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반적인 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. 에서 사용할 수 있습니다 AWS 계정. 사용 사례에 맞는 AWS 고객 관리형 정책을 정의하여 권한을 추가로 줄이는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책 또는 AWS 직무에 대한 관리형 정책을 참조하세요.
- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우, 작업을 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명서에 있는 IAM의 정책 및 권한을 참조하세요.
- IAM 정책의 조건을 사용하여 액세스 추가 제한 – 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어, SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 작성할 수 있습니다. AWS 서비스와 같은 특정를 통해 사용되는 경우 조건을 사용하여 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다 CloudFormation. 자세한 내용은 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 확인하여 안전하고 기능적인 권한 보장 - IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 새로운 및 기존 정책을 확인합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 내용은 IAM 사용 설명서의 IAM Access Analyzer에서 정책 검증을 참조하세요.
- 다중 인증(MFA) 필요 -에서 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 MFA를 AWS 계정켭니다. API 작업을 직접적으로 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 내용은 IAM 사용 설명서의 MFA를 통한 보안 API 액세스를 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

AWS HealthLake 콘솔 사용

AWS HealthLake 콘솔에 액세스하려면 최소 권한 집합이 있어야 합니다. 이러한 권한은에서 HealthLake 리소스에 대한 세부 정보를 나열하고 볼 수 있도록 허용해야 합니다 AWS 계정. 최소 필수 권한보다 더 제한적인 ID 기반 정책을 생성하는 경우, 콘솔이 해당 정책에 연결된 엔티티(사용자 또는 역할)에 대해 의도대로 작동하지 않습니다.

AWS CLI 또는 AWS API만 호출하는 사용자에게는 최소 콘솔 권한을 허용할 필요가 없습니다. 대신, 수행하려는 API 작업과 일치하는 작업에만 액세스할 수 있도록 합니다.

HealthLake에 대한 전체 액세스 권한을 얻으려면 IAM 사용자 또는 역할에 `AmazonHealthLakeFullAccess` 및 정책을 연결합니다`AWSLakeFormationDataAdmin`. 또한 서비스 역할인 HealthLake 인라인 정책을 연결해야 합니다. 서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하는 것으로 가정하는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 AWS 서비스 AWS에 권한을 위임할 역할 생성을 참조하세요. 필요한 서비스 역할을 생성하는 인라인 정책에 대한 자세한 내용은 섹션을 참조하세요설 AWS HealthLake정. 또한 AWS Lake Formation 콘솔 또는 CLI를 사용하여 HealthLake 관리자에게 AWS Lake Formation Data Lake 관리자를 할당해야 합니다. 자세한 내용은 설 AWS HealthLake정 단원을 참조하십시오.

에서 AWS HealthLake 데이터 스토어에 액세스 Amazon Athena

사용자 및 역할에의 HealthLake 데이터 스토어에 대한 액세스 권한을 제공하려면 역할 또는 사용자에게 `AmazonAthenaFullAccess` 및와 같은 IAM 정책을 Amazon Athena연결합니다`AmazonS3FullAccess`. `Select` 및 `Describe` 권한은에서 관리하는 테이블에도 필요합니다 AWS Lake Formation. AWS Lake Formation 테이블 권한은 AWS Lake Formation 콘솔의 AWS Lake Formation 관리자가 부여하거나 CLI를 통해 부여됩니다. 자세한 내용은 설 AWS HealthLake정 섹션을 참조하세요.

사용자가 자체 권한을 볼 수 있도록 허용

이 예제는 IAM 사용자가 자신의 사용자 ID에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여 줍니다. 이 정책에는 콘솔에서 또는 AWS CLI 또는 AWS API를 사용하여 프로그래밍 방식으로이 작업을 완료할 수 있는 권한이 포함됩니다.

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

```
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS AWS HealthLake에 대한 관리형 정책

AWS 관리형 정책은에서 생성하고 관리하는 독립 실행형 정책입니다 AWS. AWS 관리형 정책은 사용자, 그룹 및 역할에 권한 할당을 시작할 수 있도록 많은 일반적인 사용 사례에 대한 권한을 제공하도록 설계되었습니다.

AWS 관리형 정책은 모든 AWS 고객이 사용할 수 있으므로 특정 사용 사례에 대해 최소 권한을 부여하지 않을 수 있습니다. 사용 사례에 고유한 고객 관리형 정책을 정의하여 권한을 줄이는 것이 좋습니다.

AWS 관리형 정책에 정의된 권한은 변경할 수 없습니다. 가 관리형 정책에 정의된 권한을 AWS 업데이트하는 AWS 경우 업데이트는 정책이 연결된 모든 보안 주체 자격 증명(사용자, 그룹 및 역할)에 영향

을 미칩니다. AWS AWS 서비스 는 새가 시작되거나 기존 서비스에 새 API 작업을 사용할 수 있게 될 때 AWS 관리형 정책을 업데이트할 가능성이 높습니다.

자세한 내용은 IAM 사용자 가이드의 AWS 관리형 정책을 참조하세요.

AWS 관리형 정책: AmazonHealthLakeFullAccess

이 `AmazonHealthLakeFullAccess` 정책은 HealthLake에 대한 전체 액세스를 제공합니다. 이 정책을 사용자 또는 역할에 연결하면 사용자는 HealthLake를 사용하여 HealthLake에서 데이터에 액세스, 쿼리, 가져오기 및 내보낼 수 있습니다. HealthLake에서 많은 공통 작업을 수행하려면 사용자 또는 역할에 정책을 추가해야 합니다. 자세한 내용은 설 AWS HealthLake정 및 HealthLake 작업 및 권한을 참조하세요.

`AmazonHealthLakeFullAccess` 정책을 IAM ID에 연결할 수 있습니다.

이 정책은 사용자와 역할이 HealthLake를 사용하여 쿼리, 검색, 가져오기 및 내보내기를 수행할 수 있도록 허용하는 관리 및 기여자 권한을 부여하며, HealthLake가 이러한 권한이 있는 사용자 및 역할을 대신하여 작업을 수행할 수 있도록 합니다.

권한 세부 정보

이 정책에는 다음 문이 포함됩니다.

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 관리형 정책: AmazonHealthLakeReadOnlyAccess

`AmazonHealthLakeReadOnlyAccess` 정책은 HealthLake 및 다른 AWS 서비스의 관련 리소스에 대한 읽기 전용 액세스 및 권한을 부여합니다. HealthLake 데이터 스토어를 쿼리하고 볼 수 있는 권한을 부여하려는 사용자에게이 정책을 적용하되, 해당 사용자를 생성하거나 변경할 수는 없습니다.

`AmazonHealthLakeReadOnlyAccess` 정책을 IAM ID에 연결할 수 있습니다.

이 정책은 사용자와 역할이 HealthLake를 쿼리할 수 있는 ## ## 권한을 부여합니다.

권한 세부 정보

이 정책에는 다음 문이 포함됩니다.

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 작업 및 권한

다음 표에는 HealthLake의 일반적인 작업과 이를 수행하는 데 필요한 권한이 나열되어 있습니다.

HealthLake 작업	필수 권한
HealthLake에서 데이터 스토어 생성	`AmazonHealthLakeFullAccess` ,`AmazonLakeFormationDataAdmin` , 인라인 정책 및에서 관리하는 AWS Lake Formation 관리자 권한 AWS Lake Formation
HealthLake에서 데이터 스토어 삭제	`AmazonHealthLakeFullAccess` , `AmazonLakeFormationDataAdmin` , 인라인 정책 및에서 관리하는 AWS Lake Formation 관리자 권한 AWS Lake Formation
HealthLake에서 데이터 스토어 나열, 검색 또는 쿼리	`AmazonHealthLakeReadOnlyAccess`
를 사용하여 데이터 스토어 쿼리 Amazon Athena	`AmazonAthenaFullAccess` 에서 관리하는 테이블에 대한 `AmazonS3FullAccess` ,

HealthLake 작업	필수 권한
	AWS Lake Formation `Select` 및 `Describe` 권한 AWS Lake Formation
HealthLake에서 데이터 가져오기	가져오기 작업에 대한 권한 설정을(를) 참조하세요.
HealthLake에서 데이터 내보내기	내보내기 작업에 대한 권한 설정을(를) 참조하세요.

AWS 관리형 정책에 대한 HealthLake 업데이트

이 서비스가 이러한 변경 사항을 추적하기 시작한 시점부터 HealthLake의 AWS 관리형 정책 업데이트에 대한 세부 정보를 봅니다. 이 페이지의 변경 사항에 대한 자동 알림을 받으려면 HealthLake 문서 기록 페이지에서 RSS 피드를 구독하세요.

변경	설명	Date
AmazonHealthLakeFullAccess	`AmazonHealthLakeFullAccess` HealthLake에 대한 전체 액세스를 허용하는 데 필요한 정책입니다.	2022년 11월 14일
AmazonHealthLakeReadOnlyAccess	`AmazonHealthLakeReadOnlyAccess` HealthLake에 대한 읽기 전용 액세스에 필요한 정책입니다.	2022년 11월 14일
HealthLake에서 변경 사항 추적 시작	HealthLake는 AWS 관리형 정책에 대한 변경 사항 추적을 시작했습니다.	2022년 11월 14일

AWS HealthLake 자격 증명 및 액세스 문제 해결

다음 정보를 사용하여 HealthLake 및 IAM 작업 시 발생할 수 있는 일반적인 문제를 진단하고 수정할 수 있습니다.

주제

- AWS HealthLake에서 작업을 수행할 권한이 없음
- iam:PassRole을 수행하도록 인증되지 않음
- 내 AWS 계정 외부의 사람이 my AWS HealthLake 리소스에 액세스하도록 허용하고 싶습니다.

AWS HealthLake에서 작업을 수행할 권한이 없음

에서 작업을 수행할 권한이 없다는 AWS Management Console 메시지가 표시되면 관리자에게 문의하여 도움을 받아야 합니다. 관리자는 사용자 이름과 암호를 제공한 사람입니다.

다음 예제 오류는 mateojackson IAM 사용자가 콘솔을 사용하여 가상 *my-example-widget* 리소스에 대한 세부 정보를 보려고 하지만 가상 healthlake:*GetWidget* 권한이 없을 때 발생합니다.

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

이 경우, Mateo는 *my-example-widget* 작업을 사용하여 healthlake:*GetWidget* 리소스에 액세스하도록 허용하는 정책을 업데이트하라고 관리자에게 요청합니다.

iam:PassRole을 수행하도록 인증되지 않음

iam:PassRole 작업을 수행할 권한이 없다는 오류가 수신되면 HealthLake에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

일부 AWS 서비스 에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 기존 역할을 해당 서비스에 전달할 수 있습니다. 이렇게 하려면 역할을 서비스에 전달할 권한이 있어야 합니다.

다음 예제 오류는 라는 IAM 사용자가 콘솔을 사용하여 HealthLake에서 작업을 수행하려고 marymajor 할 때 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 권한이 없습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

이 경우, Mary가 iam:PassRole작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

내 AWS 계정 외부의 사람이 my AWS HealthLake 리소스에 액세스하도록 허용하고 싶습니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 지원하는 서비스의 경우, 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세한 내용은 다음을 참조하세요.

- HealthLake가 이러한 기능을 지원하는지 여부를 알아보려면 섹션을 참조하세요AWS HealthLake와 IAM의 작동 방식.
- 소유 AWS 계정 한의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 소유한 다른의 IAM 사용자에게 액세스 권한 제공을 참조 AWS 계정 하세요.
- 타사에 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사가 AWS 계정 소유한에 대한 액세스 권한 제공을 AWS 계정참조하세요.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(ID 페더레이션)을 참조하세요.
- 크로스 계정 액세스에 대한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 크로스 계정 리소스 액세스를 참조하세요.

AWS HealthLake에 대한 규정 준수 검증

타사 감사자는 여러 규정 준수 프로그램의 일환으로 AWS HealthLake의 보안 및 AWS 규정 준수를 평가합니다. HealthLake의 경우 여기에는 HIPAA가 포함됩니다.

AWS 서비스 가 특정 규정 준수 프로그램의 범위 내에 있는지 알아보려면 AWS 서비스 규정 준수 프로그램 범위 내를 참조하고 관심 있는 규정 준수 프로그램을 선택합니다. 일반 정보는 AWS 규정 준수 프로그램.

를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다 AWS Artifact. 자세한 내용은 Downloading Reports inDownloading AWS Artifact을 참조하세요.

사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률과 규정에 따라 AWS 서비스 결정됩니다. 사용 시 규정 준수 책임에 대한 자세한 내용은 AWS 보안 설명서를 AWS 서비스참조하세요.

AWS HealthLake의 인프라 보안

관리형 서비스인 AWS HealthLake는 Amazon Web Services: 보안 프로세스 개요 백서에 설명된 AWS 글로벌 네트워크 보안 절차로 보호됩니다.

AWS 에서 게시한 API 호출을 사용하여 네트워크를 통해 HealthLake에 액세스합니다. 클라이언트가 전송 계층 보안(TLS) 1.0 이상을 지원해야 합니다. TLS 1.2 이상을 권장합니다. 클라이언트는 Ephemeral Diffie-Hellman(DHE) 또는 Elliptic Curve Ephemeral Diffie-Hellman(ECDHE)과 같은 PFS(전달 완전 보안, Perfect Forward Secrecy)가 포함된 암호 제품군도 지원해야 합니다. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 위탁자와 관련된 시크릿 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 보안 자격 증명을 생성하여 요청에 서명할 수 있습니다.

를 사용하여 AWS HealthLake 리소스 생성 AWS CloudFormation

AWS HealthLake는 AWS 리소스 및 인프라를 생성하고 관리하는 데 소요되는 시간을 줄일 수 있도록 리소스를 모델링하고 설정하는 데 도움이 되는 AWS CloudFormation서비스인와 통합됩니다. 원하는 모든 AWS 리소스를 설명하는 템플릿을 생성하고 해당 리소스를 CloudFormation 프로비저닝하고 구성합니다.

를 사용하면 템플릿을 재사용하여 HealthLake 리소스를 일관되고 반복적으로 설정할 CloudFormation 수 있습니다. 리소스를 한 번 설명한 다음 여러 AWS 계정 및 리전에서 동일한 리소스를 반복적으로 프로비저닝합니다.

HealthLake 및 CloudFormation 템플릿

HealthLake 및 관련 서비스에 대한 리소스를 프로비저닝하고 구성하려면 CloudFormation 템플릿을 이해해야 합니다. 템플릿은 JSON 또는 YAML로 서식 지정된 텍스트 파일입니다. 이러한 템플릿은 CloudFormation 스택에서 프로비저닝하려는 리소스를 설명합니다. JSON 또는 YAML에 익숙하지 않은 경우 CloudFormation Designer를 사용하여 CloudFormation 템플릿을 시작할 수 있습니다. 자세한 내용은 AWS CloudFormation 사용자 안내서에서 CloudFormation Designer란?을 참조하세요.

> **Note**
>
> AWS HealthLake는를 사용하여 데이터 스토어 생성을 지원합니다 CloudFormation. HealthLake 데이터 스토어 프로비저닝을 위한 JSON 및 YAML 템플릿의 예를 포함하여 자세한

내용은 AWS CloudFormation 사용 설명서의 AWS HealthLake 리소스 유형 참조를 참조하세요.

에 대해 자세히 알아보기 CloudFormation

에 대해 자세히 알아보려면 다음 리소스를 CloudFormation참조하세요.

- AWS CloudFormation
- AWS CloudFormation 사용 설명서
- CloudFormation API 레퍼런스
- AWS CloudFormation 명령줄 인터페이스 사용 설명서

AWS HealthLake 및 인터페이스 VPC 엔드포인트(AWS PrivateLink)

인터페이스 VPC 엔드포인트를 생성하여 VPC와 AWS HealthLake 간에 프라이빗 연결을 설정할 수 있습니다. 인터페이스 VPC 엔드포인트는 인터넷 게이트웨이AWS PrivateLink, NAT 디바이스, VPN 연결 또는 Direct Connect 연결이 없는 APIs인 HealthLake에 비공개로 액세스하는 데 사용할 수 있는 기술로 구동됩니다. VPC의 인스턴스는 HealthLake; APIs. VPC와 HealthLake 간의 트래픽.는 Amazon 네트워크를 벗어나지 않습니다.

각 인터페이스 엔드포인트는 서브넷에서 하나 이상의 Elastic Network Interfaces로 표현됩니다.

자세한 내용은 Amazon VPC 사용 설명서에서 인터페이스 VPC 종단점(AWS PrivateLink)을 참조하세요.

HealthLake VPC 엔드포인트에 대한 고려 사항

HealthLake용 인터페이스 VPC 엔드포인트를 설정하기 전에 Amazon VPC 사용 설명서의 인터페이스 엔드포인트 속성 및 제한 사항을 검토해야 합니다.

HealthLake는 VPC에서 모든 API 작업을 호출할 수 있도록 지원합니다.

HealthLake용 인터페이스 VPC 엔드포인트 생성

Amazon VPC 콘솔 또는 AWS Command Line Interface ()를 사용하여 HealthLake; 서비스에 대한 VPC 엔드포인트를 생성할 수 있습니다AWS CLI. 자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 엔드포인트 생성을 참조하세요.

다음 서비스 이름을 사용하여 HealthLake용 VPC 엔드포인트를 생성합니다.

- com.amazonaws.*region*.healthlake

엔드포인트에 대해 프라이빗 DNS를 켜면 리전의 기본 DNS 이름을 사용하여 HealthLake에 API 요청을 할 수 있습니다. 예를 들어 *healthlake.us-east-1.amazonaws.com*입니다.

자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 엔드포인트를 통해 서비스 액세스를 참조하세요.

HealthLake에 대한 VPC 엔드포인트 정책 생성

HealthLake에 대한 액세스를 제어하는 VPC 엔드포인트에 엔드포인트 정책을 연결할 수 있습니다. 이 정책은 다음 정보를 지정합니다.

- 작업을 수행할 수 있는 위탁자.
- 수행할 수 있는 작업.
- 작업을 수행할 수 있는 리소스.

자세한 내용은 Amazon VPC 사용 설명서의 VPC 엔드포인트를 통해 서비스에 대한 액세스 제어를 참조하세요.

예: HealthLake 작업에 대한 VPC 엔드포인트 정책

다음은 HealthLake에 대한 엔드포인트 정책의 예입니다. 엔드포인트에 연결되면이 정책은 모든 리소스의 모든 보안 주체에 대해 HealthLake `CreateFHIRDatastore` 작업에 대한 액세스 권한을 부여합니다.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "healthlake:create-fhir-datastore"
         ],
         "Resource":"*"
      }
   ]
```

```
}
```

AWS HealthLake의 보안 모범 사례

AWS HealthLake는 자체 보안 정책을 개발하고 구현할 때 고려해야 할 여러 보안 기능을 제공합니다. 다음 모범 사례는 일반적인 지침이며 완벽한 보안 솔루션을 나타내지는 않습니다. 이러한 모범 사례는 사용자의 환경에 적절하지 않거나 충분하지 않을 수 있으므로 규정이 아닌 참고용으로만 사용하세요.

- 최소 권한 액세스 구현.
- 가능하면 Customer-Managed-Keys(CMKs)를 사용하여 데이터를 암호화합니다. CMKs. https://docs.aws.amazon.com/kms/latest/developerguide/overview.html
- 데이터 스토어에서 PHI 또는 PII를 쿼리할 때 GET으로 검색하지 않고 POST로 검색을 사용합니다.
- 민감하고 중요한 감사 기능에 대한 액세스를 제한합니다.
- 업데이트 또는 대량 가져오기 APIs를 통해 리소스를 생성할 때 데이터 스토어 및 작업의 이름을 포함한 PHI 또는 PII를 표시되는 필드 또는 논리적 FHIR ID(LID)에서 사용하지 마십시오.
- 생성, 읽기, 업데이트, 삭제 또는 검색 요청을 전송할 때 HTTP 헤더에 PHI를 사용하지 마십시오.
- AWS CloudTrail 를 활성화하여 AWS HealthLake 사용을 감사하고 예기치 않은 활동이 없는지 확인합니다.
- Amazon S3 버킷을 안전하게 사용하기 위한 모범 사례를 검토합니다. 자세한 내용은 Amazon S3 사용 설명서의 보안 모범 사례를 참조하세요.

AWS HealthLake의 복원력

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다. AWS 리전은 물리적으로 분리되고 격리된 여러 가용 영역을 제공하며,이 가용 영역은 지연 시간이 짧고 처리량이 높으며 중복성이 높은 네트워킹과 연결됩니다. 가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 극복 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 복수 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

더 먼 지역적 거리를 두고 데이터 또는 애플리케이션을 복제해야 하는 경우 AWS 로컬 지역을 사용하세요. AWS 로컬 리전은 기존 AWS 리전을 보완하도록 설계된 단일 데이터 센터입니다. 모든 AWS 리전과 마찬가지로 AWS 로컬 리전은 다른 AWS 리전과 완전히 격리됩니다.

AWS 리전 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하세요.

AWS HealthLake 참조

다음 지원 참조 자료는 FHIR, FHIR 및의 SMART에 사용할 수 있습니다 AWS HealthLake.

> **Note**
>
> 모든 네이티브 HealthLake 작업 및 데이터 형식은 별도의 참조에 설명되어 있습니다. 자세한 내용은 AWS HealthLake API 참조를 참조하세요.

주제

- 에 대한 FHIR 기반 SMART 지원 AWS HealthLake
- 에 대한 FHIR R4 지원 AWS HealthLake
- 에 대한 규정 준수 참조 AWS HealthLake
- 에 대한 지원 참조 AWS HealthLake

에 대한 FHIR 기반 SMART 지원 AWS HealthLake

FHIR 지원 HealthLake 데이터 스토어의 대체 가능한 의료 애플리케이션 및 재사용 가능한 기술(SMART)을 사용하면 FHIR 준수 애플리케이션의 SMART에 액세스할 수 있습니다. HealthLake 데이터는 타사 권한 부여 서버를 사용하여 요청을 인증하고 권한을 부여하여 액세스합니다. 따라서를 통해 사용자 자격 증명을 관리하는 대신 FHIR 준수 권한 부여 서버의 SMART를 사용하여 AWS Identity and Access Management사용자 자격 증명을 관리합니다.

> **Note**
>
> HealthLake는 FHIR 버전 1.0 및 2.0에서 SMART를 지원합니다. 이러한 프레임워크에 대한 자세한 내용은 FHIR R4 설명서의 SMART 앱 시작을 참조하세요.
> HealthLake 데이터 스토어는 FHIR 요청에서 SMART에 대해 다음과 같은 인증 및 권한 부여 프레임워크를 지원합니다.
>
> - OpenID(AuthN): 개인 또는 클라이언트 애플리케이션을 인증하기 위한 사용자(또는 대상)입니다.

- OAuth 2.0(AuthZ): HealthLake 데이터 스토어의 어떤 FHIR 리소스에 인증된 요청을 읽거나 쓸 수 있는지 승인하는 데 사용됩니다. 이는 권한 부여 서버에 설정된 범위에 의해 정의됩니다.

AWS CLI 또는 AWS SDKs. 자세한 내용은 HealthLake 데이터 스토어 생성 단원을 참조하십시오.

주제

- FHIR에서 SMART 시작하기
- FHIR의 SMART에 대한 HealthLake 인증 요구 사항
- HealthLake에서 지원하는 FHIR OAuth 2.0 범위의 SMART
- 를 사용한 토큰 검증 AWS Lambda
- FHIR 기반 SMART 지원 HealthLake 데이터 스토어에서 세분화된 권한 부여 사용
- FHIR 검색 문서에서 SMART 가져오기
- SMART 지원 HealthLake 데이터 스토어에서 FHIR REST API 요청

FHIR에서 SMART 시작하기

다음 주제에서는 AWS HealthLake에 대한 FHIR 권한 부여에서 SMART를 시작하는 방법을 설명합니다. AWS 여기에는 계정에서 프로비저닝해야 하는 리소스, FHIR 지원 HealthLake 데이터 스토어에서 SMART 생성, FHIR 클라이언트 애플리케이션에서 권한 부여 서버 및 HealthLake 데이터 스토어와 상호 작용하는 방법의 예가 포함됩니다.

주제

- FHIR에서 SMART에 대한 리소스 설정
- FHIR에서 SMART를 위한 클라이언트 애플리케이션 워크플로

FHIR에서 SMART에 대한 리소스 설정

다음 단계에서는 HealthLake에서 FHIR 요청에 대한 SMART를 처리하는 방법과 요청에 필요한 리소스를 정의합니다. 다음 요소는 워크플로에서 함께 작동하여 FHIR에서 SMART 요청을 생성합니다.

- 최종 사용자: 일반적으로 환자 또는 임상의가 FHIR 기반 타사 SMART 애플리케이션을 사용하여 HealthLake 데이터 스토어의 데이터에 액세스합니다.

- FHIR 기반 SMART 애플리케이션(클라이언트 애플리케이션이라고 함): HealthLake 데이터 스토어에 있는 데이터에 액세스하려는 애플리케이션입니다.
- 권한 부여 서버: 사용자를 인증하고 액세스 토큰을 발급할 수 있는 OpenID Connect 호환 서버입니다.
- HealthLake 데이터 스토어: 보유자 토큰을 제공하는 FHIR REST 요청에 응답하기 위해 Lambda 함수를 사용하는 FHIR 지원 HealthLake 데이터 스토어의 SMART입니다.

이러한 요소가 함께 작동하려면 다음 리소스를 생성해야 합니다.

> **Note**
>
> 권한 부여 서버를 설정하고, 권한 부여 서버에 필요한 범위를 정의하고, 토큰 내부 검사를 처리하는 AWS Lambda 함수를 생성한 후 FHIR 지원 HealthLake 데이터 스토어에서 SMART를 생성하는 것이 좋습니다.

1. 권한 부여 서버 엔드포인트 설정

SMART on FHIR 프레임워크를 사용하려면 데이터 스토어에서 이루어진 FHIR REST 요청을 검증할 수 있는 타사 권한 부여 서버를 설정해야 합니다. 자세한 내용은 FHIR의 SMART에 대한 HealthLake 인증 요구 사항 단원을 참조하십시오.

2. 권한 부여 서버의 범위를 정의하여 HealthLake 데이터 스토어 액세스 수준 제어

SMART on FHIR 프레임워크는 OAuth 범위를 사용하여 인증된 요청이 액세스할 수 있는 FHIR 리소스와 범위를 결정합니다. 범위를 정의하는 것은 최소 권한을 설계하는 방법입니다. 자세한 내용은 HealthLake에서 지원하는 FHIR OAuth 2.0 범위의 SMART 단원을 참조하십시오.

3. 토큰 내부 검사를 수행할 수 있는 AWS Lambda 함수 설정

SMART on FHIR 지원 데이터 스토어의 클라이언트 애플리케이션에서 보낸 FHIR REST 요청에는 JSON 웹 토큰(JWT)이 포함됩니다. 자세한 내용은 JWT 디코딩을 참조하세요.

4. FHIR 지원 HealthLake 데이터 스토어에서 SMART 생성

FHIR HealthLake 데이터 스토어에서 SMART를 생성하려면를 제공해야 합니다`IdentityProviderConfiguration`. 자세한 내용은 HealthLake 데이터 스토어 생성 단원을 참조하십시오.

FHIR에서 SMART를 위한 클라이언트 애플리케이션 워크플로

다음 섹션에서는 클라이언트 애플리케이션을 시작하고 FHIR의 SMART 컨텍스트 내에서 HealthLake 데이터 스토어에서 성공적인 FHIR REST 요청을 하는 방법을 설명합니다.

1. 클라이언트 애플리케이션을 사용하여 잘 알려진 Uniform Resource Identifier에 **`GET`** 요청

SMART 지원 클라이언트 애플리케이션은 HealthLake 데이터 스토어의 권한 부여 엔드포인트를 찾기 위해 `GET` 요청해야 합니다. 이는 잘 알려진 URI(Uniform Resource Identifier) 요청을 통해 수행됩니다. 자세한 내용은 FHIR 검색 문서에서 SMART 가져오기 단원을 참조하십시오.

2. 액세스 및 범위 요청

클라이언트 애플리케이션은 사용자가 로그인할 수 있도록 권한 부여 서버의 권한 부여 엔드포인트를 사용합니다. 이 프로세스는 사용자를 인증합니다. 범위는 클라이언트 애플리케이션이 액세스할 수 있는 HealthLake 데이터 스토어의 FHIR 리소스를 정의하는 데 사용됩니다. 자세한 내용은 HealthLake에서 지원하는 FHIR OAuth 2.0 범위의 SMART 단원을 참조하십시오.

3. 액세스 토큰

이제 사용자가 인증되었으므로 클라이언트 애플리케이션은 권한 부여 서버로부터 JWT 액세스 토큰을 받습니다. 이 토큰은 클라이언트 애플리케이션이 FHIR REST 요청을 HealthLake로 전송할 때 제공됩니다. 자세한 내용은 토큰 검증 단원을 참조하십시오.

4. FHIR 지원 HealthLake 데이터 스토어의 SMART에서 FHIR REST API 요청

이제 클라이언트 애플리케이션은 권한 부여 서버에서 제공하는 액세스 토큰을 사용하여 HealthLake 데이터 스토어 엔드포인트에 FHIR REST API 요청을 보낼 수 있습니다. 자세한 내용은 SMART 지원 HealthLake 데이터 스토어에서 FHIR REST API 요청 단원을 참조하십시오.

5. JWT 액세스 토큰 검증

FHIR REST 요청에서 전송된 액세스 토큰을 검증하려면 Lambda 함수를 사용합니다. 자세한 내용은 를 사용한 토큰 검증 AWS Lambda 단원을 참조하십시오.

FHIR의 SMART에 대한 HealthLake 인증 요구 사항

FHIR 지원 HealthLake 데이터 스토어의 SMART에서 FHIR 리소스에 액세스하려면 클라이언트 애플리케이션이 OAuth 2.0 호환 권한 부여 서버의 승인을 받고 FHIR REST API 요청의 일부로 OAuth Bearer 토큰을 제시해야 합니다. 권한 부여 서버의 엔드포인트를 찾으려면 `Well-Known` Uniform Resource

Identifier를 통해 FHIR Discovery Document의 HealthLake SMART를 사용합니다. 이 프로세스에 대한 자세한 내용은 섹션을 참조하세요FHIR 검색 문서에서 SMART 가져오기.

FHIR HealthLake 데이터 스토어에서 SMART를 생성할 때는 `CreateFHIRDatastore` 요청의 `metadata` 요소에 권한 부여 서버의 엔드포인트와 토큰 엔드포인트를 정의해야 합니다. `metadata` 요소 정의에 대한 자세한 내용은 섹션을 참조하세요HealthLake 데이터 스토어 생성.

클라이언트 애플리케이션은 권한 부여 서버 엔드포인트를 사용하여 권한 부여 서비스로 사용자를 인증합니다. 권한 부여 및 인증되면 권한 부여 서비스에서 JSON 웹 토큰(JWT)이 생성되어 클라이언트 애플리케이션으로 전달됩니다. 이 토큰에는 클라이언트 애플리케이션이 사용할 수 있는 FHIR 리소스 범위가 포함되어 있으므로 사용자가 액세스할 수 있는 데이터가 제한됩니다. 선택적으로 시작 범위가 제공된 경우 응답에 이러한 세부 정보가 포함됩니다. HealthLake에서 지원하는 SMART on FHIR 범위에 대한 자세한 내용은 섹션을 참조하세요HealthLake에서 지원하는 FHIR OAuth 2.0 범위의 SMART.

클라이언트 애플리케이션은 권한 부여 서버에서 부여한 JWT를 사용하여 FHIR REST API를 FHIR 지원 HealthLake 데이터 스토어의 SMART로 호출합니다. JWT를 검증하고 디코딩하려면 Lambda 함수를 생성해야 합니다. HealthLake는 FHIR REST API 요청이 수신되면 사용자를 대신하여이 Lambda 함수를 호출합니다. 스타터 Lambda 함수 예제를 보려면 섹션을 참조하세요를 사용한 토큰 검증 AWS Lambda.

FHIR 지원 HealthLake 데이터 스토어에서 SMART를 생성하는 데 필요한 권한 부여 서버 요소

`CreateFHIRDatastore` 요청에서 `IdentityProviderConfiguration` 객체에 있는 `metadata` 요소의 일부로 권한 부여 엔드포인트와 토큰 엔드포인트를 제공해야 합니다. 권한 부여 엔드포인트와 토큰 엔드포인트가 모두 필요합니다. `CreateFHIRDatastore` 요청에 지정되는 방법의 예를 보려면 섹션을 참조하세요HealthLake 데이터 스토어 생성.

FHIR 지원 HealthLake 데이터 스토어의 SMART에서 FHIR REST API 요청을 완료하는 데 필요한 클레임

함수가 SMART on FHIR 지원 HealthLake 데이터 스토어에서 유효한 FHIR REST API 요청이 되려면 AWS Lambda 함수에 다음 클레임이 포함되어야 합니다.

- `nbf`: (이전 아님) 클레임 - "nbf"(이전 아님) 클레임은 JWT가 처리에 수락되지 않아야 하는 시간을 식별합니다. "nbf" 클레임을 처리하려면 현재 날짜/시간이 "nbf" 클레임에 나열된 이전이 아닌 날짜/시간 이후이거나 같아야 합니다. 제공하는 샘플 Lambda 함수는 서버 응답`iat`에서 로 변환됩니다`nbf`.
- `exp`: (만료 시간) 클레임 - "exp"(만료 시간) 클레임은 JWT가 처리에 수락되지 않아야 하는 만료 시간을 식별합니다.

- `isAuthorized`: 로 설정된 부울입니다`True`. 권한 부여 서버에서 요청이 승인되었음을 나타냅니다.
- `aud`: (대상) 클레임 - "aud"(대상) 클레임은 JWT가 의도한 수신자를 식별합니다. 이는 FHIR 지원 HealthLake 데이터 스토어 엔드포인트에서 SMART여야 합니다.
- `scope`: 하나 이상의 FHIR 리소스 관련 범위여야 합니다. 이 범위는 권한 부여 서버에 정의되어 있습니다. HealthLake에서 허용하는 FHIR 리소스 관련 범위에 대한 자세한 내용은 섹션을 참조하세요HealthLake의 FHIR 리소스 범위에 대한 SMART.

HealthLake에서 지원하는 FHIR OAuth 2.0 범위의 SMART

HealthLake는 OAuth 2.0을 권한 부여 프로토콜로 사용합니다. 권한 부여 서버에서이 프로토콜을 사용하면 클라이언트 애플리케이션이 액세스할 수 있는 FHIR 리소스에 대한 HealthLake 데이터 스토어 권한(생성, 읽기, 업데이트, 삭제 및 검색)을 정의할 수 있습니다.

SMART on FHIR 프레임워크는 권한 부여 서버에서 요청할 수 있는 범위 집합을 정의합니다. 예를 들어, 환자가 랩 결과를 보거나 연락처 세부 정보를 볼 수 있도록 설계된 클라이언트 애플리케이션은 `read` 범위를 요청할 수 있는 권한만 부여해야 합니다.

> **Note**
>
> HealthLake는 아래 설명된 대로 FHIR V1 및 V2에서 모두 SMART를 지원합니다. 데이터 스토어가 생성될 때 FHIR의 SMART`AuthorizationStrategy`는 다음 세 가지 값 중 하나로 설정됩니다.
>
> - `SMART_ON_FHIR_V1` - (읽기/검색) 및 `read` (`write`create/update/delete) 권한을 포함하는 FHIR V1의 SMART만 지원합니다.
> - `SMART_ON_FHIR` - , , `create`, 및 `search` 권한을 포함하는 FHIR V1 `delete`및 V2의 SMART`read``update`를 모두 지원합니다.
> - `AWS_AUTH` - default AWS HealthLake 권한 부여 전략으로, FHIR의 SMART와 관련이 없습니다.

독립 실행형 시작 범위

HealthLake는 독립 실행형 시작 모드 범위를 지원합니다`launch/patient`.

독립 실행형 시작 모드에서 클라이언트 애플리케이션은 사용자와 환자가 클라이언트 애플리케이션에 알려지지 않았기 때문에 환자의 임상 데이터에 대한 액세스를 요청합니다. 따라서 클라이언트 애플리

케이션의 권한 부여 요청은 환자 범위를 반환하도록 명시적으로 요청합니다. 인증에 성공하면 권한 부여 서버는 요청된 시작 환자 범위가 포함된 액세스 토큰을 발급합니다. 필요한 환자 컨텍스트는 권한 부여 서버의 응답에서 액세스 토큰과 함께 제공됩니다.

지원되는 시작 모드 범위

Scope	설명
`launch/patient`	OAuth 2.0 권한 부여 요청의 파라미터로, 권한 부여 응답에서 환자 데이터를 반환하도록 요청합니다.

HealthLake의 FHIR 리소스 범위에 대한 SMART

HealthLake는 FHIR 리소스 범위에서 SMART의 세 가지 수준을 정의합니다.

- `patient` 범위는 단일 환자의 특정 데이터에 대한 액세스 권한을 부여합니다.
- `user` 범위는 사용자가 액세스할 수 있는 특정 데이터에 대한 액세스 권한을 부여합니다.
- `system` 범위는 HealthLake 데이터 스토어에 있는 모든 FHIR 리소스에 대한 액세스 권한을 부여합니다.

다음 섹션에서는 FHIR V1의 SMART 또는 FHIR V2의 SMART를 사용하여 FHIR 리소스 범위를 구성하는 구문을 나열합니다.

> **Note**
>
> 데이터 스토어가 생성될 때 SMART on FHIR 권한 부여 전략이 설정됩니다. 자세한 내용은 AWS HealthLake API 참조의 섹션을 참조`AuthorizationStrategy`하세요.

HealthLake에서 지원하는 FHIR V1 범위의 SMART

FHIR V1에서 SMART를 사용하는 경우 HealthLake에 대한 FHIR 리소스 범위를 구성하기 위한 일반적인 구문은 다음과 같습니다. 다음 예제에서 전체 URL 경로를 보려면 복사 버튼을 스크롤합니다.

```
('patient' | 'user' | 'system') '/' (fhir-resource | '*') '.' ('read' | 'write' | '*')
```

FHIR v1에서 지원되는 권한 부여 범위에 대한 SMART

범위 구문	범위 예	결과
`patient/(fhir-resource | '*').('read' | 'write' | '*')`	`patient/AllergyIntolerance.*`	환자 클라이언트 애플리케이션에는 기록된 모든 알러지에 대한 인스턴스 수준의 읽기/쓰기 액세스 권한이 있습니다.
`user/(fhir-resource | '*').('read' | 'write' | '*')`	`user/Observation.read`	사용자 클라이언트 애플리케이션에는 기록된 모든 관측치에 대한 인스턴스 수준의 읽기/쓰기 액세스 권한이 있습니다.
`system/('read' | 'write' | *)`	`system/*.*`	시스템 클라이언트 애플리케이션은 모든 FHIR 리소스 데이터에 대한 읽기/쓰기 액세스 권한이 있습니다.

HealthLake에서 지원하는 FHIR V2 범위의 SMART

FHIR V2에서 SMART를 사용하는 경우 HealthLake에 대한 FHIR 리소스 범위를 구성하기 위한 일반적인 구문은 다음과 같습니다. 다음 예제에서 전체 URL 경로를 보려면 복사 버튼을 스크롤합니다.

```
('patient' | 'user' | 'system') '/' (fhir-resource | '*') '.' ('c' | 'r' | 'u' | 'd' | 's')
```

> **Note**
>
> FHIR V2에서 SMART를 사용하려면 `IdentityProviderConfiguration` 값을 데이터 형식의 멤버인 `permission-v2` 메타데이터 `capabilities` 문자열에 전달해야 합니다.

HealthLake는 세분화된 범위를 지원합니다. 자세한 내용은 FHIR US Core 구현 안내서에서 지원되는 세분화된 범위를 참조하세요.

FHIR V2에서 지원되는 권한 부여 범위에 대한 SMART

범위 구문	예제 V1 범위	결과
`patient/Observation.rs`	`user/Observation.read`	현재 환자의 `Observation` 리소스를 읽고 검색할 수 있는 권한.
`system/*.cruds`	`system/*.*`	시스템 클라이언트 애플리케이션에는 모든 FHIR 리소스 데이터에 대한 전체 create/read/update/삭제/검색 액세스 권한이 있습니다.

를 사용한 토큰 검증 AWS Lambda

FHIR 지원 데이터 스토어에서 HealthLake SMART를 생성할 때는 `CreateFHIRDatastore` 요청에 AWS Lambda 함수의 ARN을 제공해야 합니다. Lambda 함수의 ARN은 `IdpLambdaArn` 파라미터를 사용하여 `IdentityProviderConfiguration` 객체에 지정됩니다.

FHIR 지원 데이터 스토어에서 SMART를 생성하기 전에 Lambda 함수를 생성해야 합니다. 데이터 스토어를 생성한 후에는 Lambda ARN을 변경할 수 없습니다. 데이터 스토어가 생성될 때 지정한 Lambda ARN을 보려면 `DescribeFHIRDatastore` API 작업을 사용합니다.

FHIR REST 요청이 FHIR 지원 데이터 스토어의 SMART에서 성공하려면 Lambda 함수가 다음을 수행해야 합니다.

- HealthLake 데이터 스토어 엔드포인트에 대한 응답을 1초 이내에 반환합니다.
- 클라이언트 애플리케이션에서 보낸 REST API 요청의 권한 부여 헤더에 제공된 액세스 토큰을 디코딩합니다.
- FHIR REST API 요청을 수행할 수 있는 충분한 권한이 있는 IAM 서비스 역할을 할당합니다.

- FHIR REST API 요청을 완료하려면 다음 클레임이 필요합니다. 자세한 내용은 필수 클레임를 참조하세요.
  - `nbf`
  - `exp`
  - `isAuthorized`
  - `aud`
  - `scope`

Lambda로 작업할 때는 Lambda 함수 외에도 실행 역할과 리소스 기반 정책을 생성해야 합니다. Lambda 함수의 실행 역할은 함수에 런타임에 필요한 AWS 서비스 및 리소스에 액세스할 수 있는 권한을 부여하는 IAM 역할입니다. 제공하는 리소스 기반 정책은 HealthLake가 사용자를 대신하여 함수를 호출하도록 허용해야 합니다.

이 주제의 섹션에서는 클라이언트 애플리케이션의 예제 요청과 디코딩된 응답, AWS Lambda 함수를 생성하는 데 필요한 단계, HealthLake가 수임할 수 있는 리소스 기반 정책을 생성하는 방법을 설명합니다.

- 1부: Lambda 함수 생성
- 2부: AWS Lambda 함수에서 사용하는 HealthLake 서비스 역할 생성
- 3부: Lambda 함수의 실행 역할 업데이트
- 4부: Lambda 함수에 리소스 정책 추가
- 5부: Lambda 함수에 대한 동시성 프로비저닝

AWS Lambda 함수 생성

이 주제에서 생성된 Lambda 함수는 HealthLake가 FHIR 지원 데이터 스토어의 SMART에 대한 요청을 수신하면 트리거됩니다. 클라이언트 애플리케이션의 요청에는 REST API 호출과 액세스 토큰이 포함된 권한 부여 헤더가 포함됩니다.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
Authorization: Bearer i8hweunweunweofiwweoijewiwe
```

이 주제의 Lambda 함수 예제에서는 AWS Secrets Manager 를 사용하여 권한 부여 서버와 관련된 자격 증명을 가립니다. Lambda 함수에서 직접 권한 부여 서버 로그인 세부 정보를 제공하지 않는 것이 좋습니다.

Example권한 부여 보유자 토큰이 포함된 FHIR REST 요청 검증

예제 Lambda 함수는 FHIR 지원 데이터 스토어의 SMART로 전송된 FHIR REST 요청을 검증하는 방법을 보여줍니다. 이 Lambda 함수를 구현하는 방법에 대한 step-by-steps 지침은 섹션을 참조하세요를 사용하여 Lambda 함수 생성 AWS Management Console.

FHIR REST API 요청에 유효한 데이터 스토어 엔드포인트, 액세스 토큰 및 REST 작업이 포함되어 있지 않으면 Lambda 함수가 실패합니다. 필요한 권한 부여 서버 요소에 대한 자세한 내용은 섹션을 참조하세요필수 클레임.

```
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()
```

를 사용하여 Lambda 함수 생성 AWS Management Console

다음 절차에서는 SMART on FHIR 지원 데이터 스토어에서 FHIR REST API 요청을 처리할 때 HealthLake가 수임할 서비스 역할을 이미 생성했다고 가정합니다. 서비스 역할을 생성하지 않은 경우에도 Lambda 함수를 생성할 수 있습니다. Lambda 함수가 작동하려면 먼저 서비스 역할의 ARN을 추가해야 합니다. 서비스 역할을 생성하고 Lambda 함수에서 지정하는 방법에 대한 자세한 내용은 섹션을 참조하세요. JWT를 디코딩하는 데 사용되는 AWS Lambda 함수에 사용할 HealthLake 서비스 역할 생성

Lambda 함수를 생성하려면(AWS Management Console)

1. Lambda 콘솔의 함수 페이지를 엽니다.
2. 함수 생성을 선택합니다.
3. 새로 작성을 선택합니다.

4. 기본 정보에서 함수 이름을 입력합니다. 런타임에서 Python 기반 런타임을 선택합니다.
5. Execution role(실행 역할)에서 Create a new role with basic Lambda permissions(기본 Lambda 권한을 가진 새 역할 생성)를 선택합니다.

   Lambda는 Amazon CloudWatch에 대한 로그 업로드 권한을 함수에 부여하는 실행 역할을 생성합니다. Lambda 함수는 함수를 호출할 때 실행 역할을 수임하고 실행 역할을 사용하여 AWS SDK에 대한 자격 증명을 생성합니다.
6. 코드 탭을 선택하고 샘플 Lambda 함수를 추가합니다.

   Lambda 함수가 사용할 서비스 역할을 아직 생성하지 않은 경우 샘플 Lambda 함수가 작동하려면 먼저 생성해야 합니다. Lambda 함수에 대한 서비스 역할 생성에 대한 자세한 내용은 섹션을 참조하세요JWT를 디코딩하는 데 사용되는 AWS Lambda 함수에 사용할 HealthLake 서비스 역할 생성.

```
import base64
import boto3
import logging
import json
import os
from urllib import request, parse

logger = logging.getLogger()
logger.setLevel(logging.INFO)

## Uses Secrets manager to gain access to the access key ID and secret access key
 for the authorization server
client = boto3.client('secretsmanager', region_name="region-of-datastore")
response = client.get_secret_value(SecretId='name-specified-by-customer-in-
secretsmanager')
secret = json.loads(response['SecretString'])
client_id = secret['client_id']
client_secret = secret['client_secret']


unencoded_auth = f'{client_id}:{client_secret}'
headers = {
  'Authorization': f'Basic {base64.b64encode(unencoded_auth.encode()).decode()}',
  'Content-Type': 'application/x-www-form-urlencoded'
}
```

```
auth_endpoint = os.environ['auth-server-base-url'] # Base URL of the Authorization
 server
user_role_arn = os.environ['iam-role-arn'] # The IAM role client application will
 use to complete the HTTP request on the datastore

def lambda_handler(event, context):
    if 'datastoreEndpoint' not in event or 'operationName' not in event or
 'bearerToken' not in event:
    return {}

    datastore_endpoint = event['datastoreEndpoint']
    operation_name = event['operationName']
    bearer_token = event['bearerToken']
    logger.info('Datastore Endpoint [{}], Operation Name:
 [{}]'.format(datastore_endpoint, operation_name))

    ## To validate the token
    auth_response = auth_with_provider(bearer_token)
    logger.info('Auth response: [{}]'.format(auth_response))
    auth_payload = json.loads(auth_response)
    ## Required parameters needed to be sent to the datastore endpoint for the HTTP
 request to go through
    auth_payload["isAuthorized"] = bool(auth_payload["active"])
    auth_payload["nbf"] = auth_payload["iat"]
    return {"authPayload": auth_payload, "iamRoleARN": user_role_arn}

## Access the server
def auth_with_provider(token):
    data = {'token': token, 'token_type_hint': 'access_token'}
    req = request.Request(url=auth_endpoint + '/v1/introspect',
 data=parse.urlencode(data).encode(), headers=headers)
    with request.urlopen(req) as resp:
    return resp.read().decode()
```

Lambda 함수의 실행 역할 수정

Lambda 함수를 생성한 후 Secrets Manager를 호출하는 데 필요한 권한을 포함하도록 실행 역할을 업데이트해야 합니다. Secrets Manager에서 생성하는 각 보안 암호에는 ARN이 있습니다. 최소 권한을 적용하려면 실행 역할에 Lambda 함수가 실행되는 데 필요한 리소스에만 액세스할 수 있어야 합니다.

IAM 콘솔에서 검색하거나 Lambda 콘솔에서 구성을 선택하여 Lambda 함수의 실행 역할을 수정할 수 있습니다. Lambda 함수 실행 역할 관리에 대한 자세한 내용은 섹션을 참조하세요Lambda 실행 역할.

Example에 대한 액세스 권한을 부여하는 Lambda 함수 실행 역할 GetSecretValue

실행 역할에 IAM 작업을 추가`GetSecretValue`하면 샘플 Lambda 함수가 작동하는 데 필요한 권한이 부여됩니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:secret-name-DKodTA"
        }
    ]
}
```

이 시점에서 FHIR 지원 데이터 스토어의 SMART로 전송된 FHIR REST 요청의 일부로 제공된 액세스 토큰을 검증하는 데 사용할 수 있는 Lambda 함수를 생성했습니다.

JWT를 디코딩하는 데 사용되는 AWS Lambda 함수에 사용할 HealthLake 서비스 역할 생성

페르소나: IAM 관리자

IAM 정책을 추가 또는 제거하고 새 IAM 자격 증명을 생성할 수 있는 사용자입니다.

서비스 역할

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하는 것으로 가정하는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 AWS 서비스 AWS에 권한을 위임할 역할 생성을 참조하세요.

JSON 웹 토큰(JWT)이 디코딩된 후 Lambda는 IAM 역할 ARN도 반환해야 합니다. 이 역할에는 REST API 요청을 수행하는 데 필요한 권한이 있어야 합니다. 그렇지 않으면 권한이 부족하여 실패합니다.

IAM을 사용하여 사용자 지정 정책을 설정할 때는 필요한 최소 권한을 부여하는 것이 가장 좋습니다. 자세한 내용은 IAM 사용 설명서의 최소 권한 적용을 참조하세요.

권한 부여 Lambda 함수에서 지정할 HealthLake 서비스 역할을 생성하려면 두 단계가 필요합니다.

- 먼저 IAM 정책을 생성해야 합니다. 정책은 권한 부여 서버에서 범위를 제공한 FHIR 리소스에 대한 액세스를 지정해야 합니다.
- 둘째, 서비스 역할을 생성해야 합니다. 역할을 생성할 때 신뢰 관계를 지정하고 1단계에서 생성한 정책을 연결합니다. 신뢰 관계는 HealthLake를 서비스 보안 주체로 지정합니다. 이 단계에서는 HealthLake 데이터 스토어 ARN과 AWS 계정 ID를 지정해야 합니다.

새 IAM 정책 생성

권한 부여 서버에서 정의한 범위에 따라 인증된 사용자가 HealthLake 데이터 스토어에서 액세스할 수 있는 FHIR 리소스가 결정됩니다.

생성한 IAM 정책은 정의한 범위에 맞게 조정할 수 있습니다.

IAM 정책 설명의 `Action` 요소에 다음 작업을 정의할 수 있습니다. 테이블의 각 `Action`에 대해를 정의할 수 있습니다`Resource types`. HealthLake에서 데이터 스토어는 IAM 권한 정책 문의 `Resource` 요소에 정의할 수 있는 유일하게 지원되는 리소스 유형입니다.

개별 FHIR 리소스는 IAM 권한 정책에서 요소로 정의할 수 있는 리소스가 아닙니다.

HealthLake에서 정의한 작업

작업	설명	액세스 레벨	리소스 유형(필수)
CreateResource	리소스 생성 권한을 부여합니다.	쓰기	데이터 스토어 ARN: arn:aws:healthlake:**your-region**:**111122223333**:datastore/fhir/**your-datastore-id**
DeleteResource	리소스를 삭제할 수 있는 권한을 부여합니다.	쓰기	데이터 스토어 ARN: arn:aws:healthlake:**your-region**:**111122223333**:datastore/fhir/**your-datastore-id**

작업	설명	액세스 레벨	리소스 유형(필수)
ReadResource	리소스를 읽을 수 있는 권한을 부여합니다.	읽기	데이터 스토어 ARN: arn:aws:healthlake:**your-region**:**111122223333**:datastore/fhir/**your-datastore-id**
SearchWithGet	GET 메소드로 리소스를 검색할 수 있는 권한을 부여합니다.	읽기	데이터 스토어 ARN: arn:aws:healthlake:**your-region**:**111122223333**:datastore/fhir/**your-datastore-id**
SearchWithPost	POST 메소드를 사용하여 리소스를 검색할 수 있는 권한을 부여합니다.	읽기	데이터 스토어 ARN: arn:aws:healthlake:**your-region**:**111122223333**:datastore/fhir/**your-datastore-id**
StartFHIRExportJobWithPost	GET을 사용하여 FHIR 내보내기 작업을 시작할 수 있는 권한을 부여합니다.	쓰기	데이터 스토어 ARN: arn:aws:healthlake:**your-region**:**111122223333**:datastore/fhir/**your-datastore-id**
UpdateResource	리소스를 업데이트할 수 있는 권한을 부여합니다.	쓰기	데이터 스토어 ARN: arn:aws:healthlake:**your-region**:**111122223333**:datastore/fhir/**your-datastore-id**

시작하려면를 사용할 수 있습니다`AmazonHealthLakeFullAccess`. 이 정책은 데이터 스토어에 있는 모든 FHIR 리소스에 대한 읽기, 쓰기, 검색 및 내보내기를 부여합니다. 데이터 스토어에 대한 읽기 전용 권한을 부여하려면를 사용합니다`AmazonHealthLakeReadOnlyAccess`.

AWS Management Console AWS CLI또는 IAM SDKs를 사용하여 사용자 지정 정책을 생성하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 IAM 정책 생성을 참조하세요.

HealthLake에 대한 서비스 역할 생성(IAM 콘솔)

이 절차를 사용하여 서비스 역할을 생성합니다. 서비스를 생성할 때 IAM 정책도 지정해야 합니다.

HealthLake에 대한 서비스 역할을 생성하려면(IAM 콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/ IAM 콘솔을 엽니다.
2. IAM 콘솔의 탐색 창에서 역할을 선택합니다.
3. 그런 다음 역할 생성을 선택합니다.
4. 신뢰 엔터티 선택 페이지에서 사용자 지정 신뢰 정책을 선택합니다.
5. 그런 다음 사용자 지정 신뢰 정책에서 다음과 같이 샘플 정책을 업데이트합니다. **your-account-id**를 계정 번호로 바꾸고 가져오기 또는 내보내기 작업에 사용할 데이터 스토어의 ARN을 추가합니다.

   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. 그리고 다음을 선택합니다.
7. 권한 추가 페이지에서 HealthLake 서비스가 수임할 정책을 선택합니다. 정책을 찾으려면 권한 정책에서 해당 정책을 검색합니다.
8. 그런 다음 정책 연결을 선택합니다.

9. 그런 다음 역할 이름 아래의 이름, 검토 및 생성 페이지에서 이름을 입력합니다.
10. (선택 사항) 그런 다음 설명에서 역할에 대한 간단한 설명을 추가합니다.
11. 가능한 경우 이 역할의 목적을 식별하는 데 도움이 되는 역할 이름이나 역할 이름 접미사를 입력합니다. 역할 이름은 AWS 계정내에서 고유해야 합니다. 대/소문자를 구분하지 않습니다. 예를 들어, 이름이 **PRODROLE**과 **prodrole**, 두 가지로 지정된 역할을 만들 수는 없습니다. 다양한 주체가 역할을 참조할 수 있기 때문에 역할이 생성된 후에는 역할 이름을 편집할 수 없습니다.
12. 역할 세부 정보를 검토한 다음 역할 생성을 선택합니다.

샘플 Lambda 함수에서 역할 ARN을 지정하는 방법을 알아보려면 섹션을 참조하세요AWS Lambda 함수 생성.

Lambda 실행 역할

Lambda 함수의 실행 역할은 AWS 서비스 및 리소스에 액세스할 수 있는 권한을 함수에 부여하는 IAM 역할입니다. 이 페이지에서는 Lambda 함수의 실행 역할을 생성하고 보고 관리하는 방법에 대한 정보를 제공합니다.

기본적으로 Lambda는를 사용하여 새 Lambda 함수를 생성할 때 최소한의 권한으로 실행 역할을 생성합니다 AWS Management Console. 실행 역할에 부여된 권한을 관리하려면 Lambda 개발자 안내서의 IAM 콘솔에서 실행 역할 생성을 참조하세요.

이 주제에 제공된 샘플 Lambda 함수는 Secrets Manager를 사용하여 권한 부여 서버의 자격 증명을 가립니다.

생성하는 모든 IAM 역할과 마찬가지로 최소 권한 모범 사례를 따르는 것이 중요합니다. 개발 문구 중에 필요한 것 이상의 권한을 부여하는 경우가 있습니다. 모범 사례로 프로덕션 환경에 함수를 게시하기 전에 필요한 권한만 포함하도록 정책을 조정하는 것이 가장 좋습니다. 자세한 내용은 IAM 사용 설명서의 최소 권한 적용을 참조하세요.

HealthLake가 Lambda 함수를 트리거하도록 허용

HealthLake가 사용자를 대신하여 Lambda 함수를 호출할 수 있도록 다음을 수행해야 합니다.

- HealthLake가 `CreateFHIRDatastore` 요청에서 호출할 Lambda 함수의 ARN과 `IdpLambdaArn` 동일하게를 설정해야 합니다.
- HealthLake가 사용자를 대신하여 Lambda 함수를 호출할 수 있도록 허용하는 리소스 기반 정책이 필요합니다.

HealthLake가 FHIR 지원 데이터 스토어의 SMART에서 FHIR REST API 요청을 수신하면 사용자를 대신하여 데이터 스토어 생성 시 지정된 Lambda 함수를 호출할 권한이 필요합니다. HealthLake 액세스 권한을 부여하려면 리소스 기반 정책을 사용합니다. Lambda 함수에 대한 리소스 기반 정책 생성에 대한 자세한 내용은 AWS Lambda 개발자 안내서의 AWS 서비스에서 Lambda 함수를 호출하도록 허용을 참조하세요.

Lambda 함수에 대한 동시성 프로비저닝

> **Important**
>
> HealthLake를 사용하려면 Lambda 함수의 최대 실행 시간이 1초(1000밀리초) 미만이어야 합니다.
> Lambda 함수가 실행 시간 제한을 초과하면 TimeOut 예외가 발생합니다.

이 예외가 발생하지 않도록 프로비저닝된 동시성을 구성하는 것이 좋습니다. 호출이 증가하기 전에 프로비저닝된 동시성을 할당하면 짧은 지연 시간으로 초기화된 인스턴스에서 모든 요청을 처리하도록 할 수 있습니다. 프로비저닝된 동시성 구성에 대한 자세한 내용은 Lambda 개발자 안내서의 프로비저닝된 동시성 구성을 참조하세요.

Lambda 함수의 평균 실행 시간을 보려면 현재 Lambda 콘솔에서 Lambda 함수의 모니터링 페이지를 사용합니다. 기본적으로 Lambda 콘솔은 함수 코드가 이벤트를 처리하는 데 소요되는 평균, 최소 및 최대 시간을 보여주는 기간 그래프를 제공합니다. Lambda 함수 모니터링에 대한 자세한 내용은 Lambda 개발자 안내서의 Lambda 콘솔에서 함수 모니터링을 참조하세요.

Lambda 함수에 대한 동시성을 이미 프로비저닝하고 이를 모니터링하려면 Lambda 개발자 안내서의 동시성 모니터링을 참조하세요.

FHIR 기반 SMART 지원 HealthLake 데이터 스토어에서 세분화된 권한 부여 사용

범위만으로는 요청자가 데이터 스토어에서 액세스할 권한이 있는 데이터에 대해 필요한 특정성을 제공하지 않습니다. 세분화된 권한 부여를 사용하면 FHIR 지원 HealthLake 데이터 스토어의 SMART에 대한 액세스 권한을 부여할 때 더 높은 수준의 특이도를 확보할 수 있습니다. 세분화된 권한 부여를 사용하려면 `CreateFHIRDatastore` 요청의 `IdentityProviderConfiguration` 파라미터`True`에서를 `FineGrainedAuthorizationEnabled`로 설정합니다.

세분화된 권한 부여를 활성화한 경우 권한 부여 서버는 액세스 토큰과 `id_token` 함께의 `fhirUser` 범위를 반환합니다. 이렇게 하면 클라이언트 애플리케이션에서 사용자에 대한 정보를 검색할 수 있습

니다. 클라이언트 애플리케이션은 fhirUser 클레임을 현재 사용자를 나타내는 FHIR 리소스의 URI로 취급해야 합니다. Patient, Practitioner 또는 RelatedPerson 유형을 지정할 수 있습니다. 권한 부여 서버의 응답에는 사용자가 액세스할 수 있는 데이터를 정의하는 user/ 범위도 포함됩니다. FHIR 리소스별 범위와 관련된 범위에 대해 정의된 구문을 사용합니다.

```
user/(fhir-resource | '*').('read' | 'write' | '*')
```

다음은 세분화된 권한 부여를 사용하여 데이터 액세스 관련 FHIR 리소스 유형을 추가로 지정하는 방법의 예입니다.

- fhirUser가 인 경우 Practitioner세분화된 권한 부여에 따라 사용자가 액세스할 수 있는 환자 모음이 결정됩니다. 에 대한 액세스fhirUser는 환자가 일반의fhirUser로서를 참조하는 경우에만 허용됩니다.

  ```
  Patient.generalPractitioner : [{Reference(Practitioner)}]
  ```

- fhirUser가 Patient 또는 이고 요청에서 참조되는 환자가 RelatedPerson와 다른 경우 fhirUser세분화된 권한 부여에 따라 요청된 환자의에 fhirUser 대한 액세스가 결정됩니다. 요청된 Patient 리소스에 지정된 관계가 있는 경우 액세스가 허용됩니다.

  ```
  Patient.link.other : {Reference(Patient|RelatedPerson)}
  ```

FHIR 검색 문서에서 SMART 가져오기

SMART는 클라이언트가 HealthLake 데이터 스토어가 지원하는 권한 부여 엔드포인트 URLs 및 기능을 학습할 수 있도록 하는 검색 문서를 정의합니다. 이 정보는 클라이언트가 올바른 엔드포인트로 권한 부여 요청을 전달하고 HealthLake 데이터 스토어가 지원하는 권한 부여 요청을 구성하는 데 도움이 됩니다.

클라이언트 애플리케이션이 HealthLake에 FHIR REST 요청을 성공적으로 수행하려면 HealthLake 데이터 스토어에서 정의한 권한 부여 요구 사항을 수집해야 합니다. 이 요청이 성공하려면 보유자 토큰(권한 부여)이 필요하지 않습니다.

HealthLake 데이터 스토어에 대한 검색 문서를 요청하려면

1. HealthLake region 및 datastoreId 값을 수집합니다. 자세한 내용은 데이터 스토어 속성 가져오기 단원을 참조하십시오.

2. HealthLake region 및에 대해 수집된 값을 사용하여 요청에 대한 URL을 구성합니다datastoreId. URL의 엔드포인트/.well-known/smart-configuration에를 추가합니다. 다음 예제에서 전체 URL 경로를 보려면 복사 버튼을 스크롤합니다.

```
https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/.well-known/smart-configuration
```

3. 서명 버전 4 서명 프로토콜과 GET 함께를 사용하여 요청을 보냅니다. AWS 전체 예제를 보려면 복사 버튼을 스크롤합니다.

   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 데이터 스토어의 검색 문서는 데이터 스토어의 사양 및 정의된 기능과 token_endpoint함께 authorization_endpoint 및를 찾을 수 있는 JSON BLOB으로 반환됩니다.

   ```
   {
       "authorization_endpoint": "https://oidc.example.com/authorize",
       "token_endpoint": "https://oidc.example.com/oauth/token",
       "capabilities": [
           "launch-ehr",
           "client-public"
       ]
   }
   ```

   클라이언트 애플리케이션을 시작하려면 authorization_endpoint 및 token_endpoint가 모두 필요합니다.

   - 권한 부여 엔드포인트 - 클라이언트 애플리케이션 또는 사용자에게 권한을 부여하는 데 필요한 URL입니다.

- 토큰 엔드포인트 - 클라이언트 애플리케이션이 통신하는 데 사용하는 권한 부여 서버의 엔드포인트입니다.

SMART 지원 HealthLake 데이터 스토어에서 FHIR REST API 요청

FHIR 지원 HealthLake 데이터 스토어의 SMART에서 FHIR REST API 요청을 수행할 수 있습니다. 다음 예제에서는 권한 부여 헤더에 JWT가 포함된 클라이언트 애플리케이션의 요청과 Lambda가 응답을 디코딩하는 방법을 보여줍니다. 클라이언트 애플리케이션 요청이 승인 및 인증된 후에는 권한 부여 서버로부터 보유자 토큰을 받아야 합니다. FHIR 지원 HealthLake 데이터 스토어의 SMART에서 FHIR REST API 요청을 보낼 때 권한 부여 헤더의 보유자 토큰을 사용합니다.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/[ID]
Authorization: Bearer auth-server-provided-bearer-token
```

권한 부여 헤더에서 보유자 토큰이 발견되고 AWS IAM 자격 증명이 감지되지 않았기 때문에 HealthLake는 FHIR에서 SMART를 활성화한 HealthLake 데이터 스토어가 생성될 때 지정된 Lambda 함수를 호출합니다. Lambda 함수에 의해 토큰이 성공적으로 디코딩되면 다음 예제 응답이 HealthLake로 전송됩니다.

```
{
  "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
}
```

에 대한 FHIR R4 지원 AWS HealthLake

AWS HealthLake 는 상태 데이터 교환을 위한 FHIR R4 사양을 지원합니다. 다음 섹션에서는 HealthLake가 FHIR R4 사양을 활용하여 FHIR R4 RESTful APIs를 사용하여 HealthLake 데이터 스토어에서 FHIR 리소스를 관리하고 검색하는 방법에 대한 지원 정보를 제공합니다.

주제

- 에 대한 FHIR R4 기능 설명 AWS HealthLake
- HealthLake에 대한 FHIR 프로파일 검증
- HealthLake에 지원되는 FHIR R4 리소스 유형
- HealthLake에 대한 FHIR R4 검색 파라미터
- HealthLake$operations용 FHIR R4

에 대한 FHIR R4 기능 설명 AWS HealthLake

활성 HealthLake 데이터 스토어의 FHIR 관련 기능(동작)을 찾으려면 해당 기능 설명을 검색해야 합니다. 기능 설명은 실제 서버 기능에 대한 설명 또는 필수 또는 원하는 서버 구현에 대한 설명으로 사용됩니다. FHIR `capabilities` 상호 작용은 HealthLake 데이터 스토어 기능 및 지원하는 FHIR 사양의 일부에 대한 정보를 검색합니다. HealthLake는 FHIR R4 리소스에 따라 FHIR `StructureDefinition` 리소스 유형을 검증합니다.

HealthLake 데이터 스토어에 대한 기능 설명을 가져오는 방법

1. HealthLake `region` 및 `datastoreId` 값을 수집합니다. 자세한 내용은 데이터 스토어 속성 가져오기 단원을 참조하십시오.
2. HealthLake `region` 및에 대해 수집된 값을 사용하여 요청에 대한 URL을 구성합니다`datastoreId`. 또한 URL에 FHIR `metadata` 요소를 포함합니다. 다음 예제에서 전체 URL 경로를 보려면 복사 버튼을 스크롤합니다.

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

3. 요청을 보냅니다. FHIR `capabilities` 상호 작용은 AWS 서명 버전 4 서명 프로토콜과 함께 `GET` 요청을 사용합니다. 다음 `curl` 예제에서는에서 지정한 HealthLake 데이터 스토어에 대한 기능 설명을 가져옵니다`datastoreId`. 전체 예제를 보려면 복사 버튼을 스크롤합니다.

curl

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

HealthLake 데이터 스토어에 대한 200 HTTP 응답 코드와 기능 설명을 받게 됩니다. 자세한 내용은 FHIR R4 설명서CapabilityStatement의 섹션을 참조하세요.

HealthLake에 대한 FHIR 프로파일 검증

AWS HealthLake 는 기본 FHIR R4 사양을 지원합니다. 기본 FHIR R4 사양에는 FHIR 프로파일이 포함되어 있습니다. 프로필은 FHIR 리소스 유형에서 기본 리소스 유형에 대한 제약 조건 및/또는 확장을 사용하여 보다 구체적인 리소스 유형 정의를 정의하는 데 사용됩니다. 예를 들어 FHIR 프로필은 확장 및 값 세트와 같은 필수 필드를 식별할 수 있습니다. 리소스는 여러 프로파일을 지원할 수 있습니다. 모든 HealthLake 데이터 스토어는 FHIR 프로파일 사용을 지원합니다.

> **Note**
>
> HealthLake 데이터 스토어에 데이터를 추가할 때는 FHIR 프로파일을 추가할 필요가 없습니다. 리소스를 추가하거나 업데이트할 때 FHIR 프로파일을 지정하지 않으면 리소스는 기본 FHIR R4 스키마에 대해서만 검증됩니다.
> FHIR 리소스가 준수하는 FHIR 프로파일은 HealthLake로 가져오기 전에 리소스에 포함됩니다. 따라서 FHIR 프로파일은 가져오는 동안 HealthLake에서 검증됩니다.

FHIR 프로필은 구현 가이드에 지정되어 있습니다. FHIR 구현 가이드(IG)는 특정 용도로 FHIR 표준을 사용하는 방법을 설명하는 일련의 지침입니다. HealthLake는 다음 구현 가이드에 정의된 FHIR 프로파일을 검증합니다.

에서 지원하는 FHIR 프로필 AWS HealthLake

이름	버전	구현 안내서	기능	미국 동부(오하이오)	미국 동부(버지니아 북부)	미국 서부(오리건)	아시아 태평양(뭄바이)	아시아 태평양(시드니)	Ca(Ce	유럽(아일랜드)	유럽(런던)
미국 코어	3.1	http://hl7.org/fhir/us/core/STU3.1.1/	기본값	X	X	X	X	X	X	X	X
미국 코어	4.0	https://hl7.org/fhir/us/core/STU4/index.html	지원됨	X	X	X	X	X	X	X	X
미국 코어	5.0	https://hl7.org/fhir/us/core/STU5.0.1/index.html	지원됨	X	X	X	X	X	X	X	X
미국 코어	6.1	https://hl7.org/fhir/us/core/STU6.1/index.html	지원됨	X	X	X	X	X	X	X	X
미국 코어	7.0	https://hl7.org/fhir/us/core/STU7/	지원됨	X	X	X	X	X	X	X	X
영국 코어	2.0	https://simplifier.net/guide/uk-core-implementation-guide-stu2/Home/ProfilesandExtensions/ProfilesIndex?version=2.0.1	지원됨	X	X	X	X			X	X

이름	버전	구현 안내서	기능	미국 동부(오하이오)	미국 동부(버지니아 북부)	미국 서부(오리건)	아시아 태평양(뭄바이)	아시아 태평양(시드니)	Ca(Ce	유럽(아일랜드)	유럽(런던)
카린 블루 버튼	1.1	http://hl7.org/fhir/us/carin-bb/STU1.1/	기본값	X	X	X	X	X	X	X	X
카린 블루 버튼	1.0 2.0 2.1	https://hl7.org/fhir/us/carin-bb/history.html	지원됨	X	X	X	X	X	X	X	X
Da Vinci 지급인 데이터 교환	1.0	https://hl7.org/fhir/us/davinci-pdex/	기본값	X	X	X	X	X	X	X	X
Da Vinci 지급인 데이터 교환	2.0 2.1	https://hl7.org/fhir/us/davinci-pdex/history.html	지원됨	X	X	X	X	X	X	X	X
Da Vinci Health Record Exchange(HRex)	0.2	https://hl7.org/fhir/us/davinci-hrex/2020Sep/	기본값	X	X	X	X	X	X	X	X
DaVinci PDEX 플랜 넷	1.1	https://hl7.org/fhir/us/davinci-pdex-plan-net/STU1.1/	기본값	X	X	X	X	X	X	X	X

이름	버전	구현 안내서	기능	미국 동부(오하이오)	미국 동부(버지니아 북부)	미국 서부(오리건)	아시아 태평양(뭄바이)	아시아 태평양(시드니)	Ca(Ce	유럽(아일랜드)	유럽(런던)
DaVinci PDEX 플랜넷	1.0	https://hl7.org/fhir/us/davinci-pdex-plan-net/STU1/	지원됨	X	X	X	X	X	X	X	X
DaVinci 지급인 데이터 교환(PDex) 미국 의약품 집	1.1	https://hl7.org/fhir/us/davinci-drug-formulary/STU1.1/	기본값	X	X	X	X	X	X	X	X
DaVinci 지급인 데이터 교환(PDex) 미국 의약품 집	1.0 2.0 2.1	https://hl7.org/fhir/us/davinci-drug-formulary/history.html	지원됨	X	X	X	X	X	X	X	X
Da Vinci 임상 데이터 교환(CDex)	2.1	https://build.fhir.org/ig/HL7/davinci-ecdx/index.html	기본값	X	X	X	X	X	X	X	X
Da Vinci 사전 승인 지원(PAS) FHIR IG	2.1	https://hl7.org/fhir/us/davinci-pas/	기본값	X	X	X	X	X	X	X	X

이름	버전	구현 안내서	기능	미국 동부(오하이오)	미국 동부(버지니아 북부)	미국 서부(오리건)	아시아 태평양(뭄바이)	아시아 태평양(시드니)	Ca(Ce	유럽(아일랜드)	유럽(런던)
NCQA HEDIS® 구현 가이드	0.3	https://www.ncqa.org/resources/hedis-ig-resource-page/	기본값	X	X	X	X			X	X
국제 환자 요약(IPS)	2.0 ballot	https://hl7.org/fhir/uv/ips/2024Sep/	기본값	X	X	X	X	X	X	X	X
품질 측정	5.0	https://registry.fhir.org/package/hl7.fhir.us.cqfmeasures%7C5.0.0	기본값	X	X	X	X			X	X
유전체학 보고	3.0	https://build.fhir.org/ig/HL7/genomics-reporting/index.html	기본값	X	X	X	X			X	X
National Health Authority's Ayushman Bharat Digital Mission(ABDM)	2.0	https://www.nrces.in/ndhm/fhir/r4/index.html	기본값	X	X	X	X			X	X

이름	버전	구현 안내서	기능	미국 동부 (오하이오)	미국 동부 (버지니아 북부)	미국 서부 (오리건)	아시아 태평양 (뭄바이)	아시아 태평양 (시드니)	Ca (Ce	유럽 (아일랜드)	유럽 (런던)
CA Core+	1.1	https://simplifier.net/ca-core	지원됨						X		
CA:eReC Pan-캐나다 eReferral-eConsult	1.1	https://simplifier.net/CA-eReC/~introduction	지원됨						X		
환자 요약 캐나다 버전 - (PS-CA)	2.1	https://simplifier.net/PS-CA-R1/~introduction	지원됨						X		
온타리오 디지털 건강 약물 리포지토리	4.0	https://simplifier.net/ca-on-dhdr-r4/~introduction	지원됨						X		
AU 코어	1.0	https://hl7.org.au/fhir/core/	지원됨					X			
Magentus 연습 관리	1.2	https://fhir-versions.dev.geniesolutions.io/1.2.16/downloads.html	지원됨					X			

리소스에 지정된 FHIR 프로파일 검증

FHIR 프로파일을 검증하려면 구현 가이드에 지정된 프로파일 URL을 사용하여 개별 리소스의 `profile` 요소에 추가합니다.

FHIR 프로필은 데이터 스토어에 새 리소스를 추가할 때 검증됩니다. 새 리소스를 추가하려면 `StartFHIRImportJob` API 작업을 사용하거나, 새 리소스를 추가하도록 `POST` 요청하거나, 기존 리소스를 업데이트 `PUT` 하도록 할 수 있습니다.

Example- 리소스에서 참조되는 FHIR 프로파일을 확인하려면

프로필 URL은 `"meta" : "profile"` 키-값 페어의 `profile` 요소에 추가됩니다. 이 리소스는 명확성을 위해 잘렸습니다.

```
{
    "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- 기본값이 아닌 지원되는 FHIR 프로파일을 참조하는 방법

지원되는 기본이 아닌 프로필(예: CarinBB 1.0.0)에 대해 검증하려면 버전('|'로 구분)이 있는 프로필 URL과 `meta.profile` 요소에 기본 프로필 URL을 추가합니다. 이 예제 리소스는 명확성을 위해 잘렸습니다.

```
{
    "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“
```

```
        ]
    }
}
```

HealthLake에 지원되는 FHIR R4 리소스 유형

다음 표에는에서 지원하는 FHIR R4 리소스 유형이 나열되어 있습니다 AWS HealthLake. 자세한 내용은 FHIR R4 설명서의 리소스 인덱스를 참조하세요.

HealthLake에서 지원하는 FHIR R4 리소스 유형

Account	DetectedIssue	Invoice	프랙티셔너
ActivityDefinition	장치	라이브러리	PractitionerRole
AdverseEvent	DeviceDefinition	연결	절차
AllergyIntolerance	DeviceMetric	List	증명
예약	DeviceUseStatement	Location	설문 조사
AppointmentResponse	DeviceRequest	치수	QuestionnaireResponse
AuditEvent - 노트 참조	DiagnosticReport	MeasureReport	RelatedPerson
바이너리	DocumentManifest	미디어	RequestGroup
BodyStructure	DocumentReference	약물	ResearchStudy
번들 - 노트 참조	EffectEvidenceSynthesis	MedicationAdministration	ResearchSubject
CapabilityStatement	상황	MedicationDispense	RiskAssessment
CarePlan	엔드포인트	MedicationKnowledge	RiskEvidenceSynthesis
CareTeam	EpisodeOfCare	MedicationRequest	일정
ChargeItem	EnrollmentRequest	MedicationStatement	ServiceRequest

ChargeItemDefinition	EnrollmentResponse	MessageHeader	슬롯
Claim	ExplanationOfBenefit	MolecularSequence	샘플
ClaimResponse	FamilyMemberHistory	NutritionOrder	StructureDefinition
통신	플래그	관측치	StructureMap
CommunicationReque st	Goal	OperationOutcome - 노트 참조	Substance
구성	Group	Organization	SupplyDelivery
ConceptMap	GuidanceResponse	OrganizationAffiliation	SupplyRequest
조건	HealthcareService	파라미터 - 노트 참조	Task
동의	ImagingStudy	환자	ValueSet
계약	면역화	PaymentNotice	VisionPrescription
적용 범위	ImmunizationEvalua tion	PaymentReconciliat ion	VerificationResult - 노트 참조
CoverageEligibilit yRequest	ImmunizationRecomm endation	Person	
CoverageEligibilit yResponse	InsurancePlan	PlanDefinition	

⚠ FHIR 사양 및 HealthLake

- FHIR `OperationOutcome` 및 `Parameters` 리소스 유형을 사용하여 `GET` 또는 `POST` 요청을 할 수 없습니다.
- AuditEvent - AuditEvent 리소스를 생성하거나 읽을 수 있지만 업데이트하거나 삭제할 수는 없습니다.
- 번들 - HealthLake가 번들 요청을 관리하는 방법에는 여러 가지가 있습니다. 자세한 내용은 FHIR 리소스 번들링 섹션을 참조하세요.

- VerificationResult -이 리소스 유형은 2023년 12월 9일 이후에 생성된 데이터 스토어에만 지원됩니다.

HealthLake에 대한 FHIR R4 검색 파라미터

FHIR `search` 상호 작용을 사용하여 일부 필터 기준에 따라 HealthLake 데이터 스토어에서 FHIR 리소스 세트를 검색합니다. 상호 `search` 작용은 `GET` 또는 `POST` 요청을 사용하여 수행할 수 있습니다. 개인 식별 정보(PII) 또는 보호 대상 건강 정보(PHI)가 포함된 검색의 경우 PII 및 PHI가 `POST` 요청 본문의 일부로 추가되고 전송 중에 암호화되므로 요청을 사용하는 것이 좋습니다.

> **Note**
>
> 이 장에 설명된 FHIR `search` 상호 작용은 의료 데이터 교환을 위한 HL7 FHIR R4 표준을 준수하도록 구축되었습니다. HL7 FHIR 서비스의 표현이므로 AWS CLI 및 AWS SDKs 통해 제공되지 않습니다. 자세한 내용은 FHIR R4 RESTful API 설명서`search`의 섹션을 참조하세요. Amazon Athena를 사용하여 SQL을 사용하여 HealthLake 데이터 스토어를 쿼리할 수도 있습니다. 자세한 내용은 통합을 참조하세요.

HealthLake는 다음과 같은 FHIR R4 검색 파라미터의 하위 집합을 지원합니다. 자세한 내용은 HealthLake에 대한 FHIR R4 검색 파라미터 단원을 참조하십시오.

지원되는 검색 파라미터 유형

다음 표에는 HealthLake에서 지원되는 검색 파라미터 유형이 나와 있습니다.

지원되는 검색 파라미터 유형

검색 파라미터	설명
_id	리소스 ID(전체 URL 아님)
_lastUpdated	마지막으로 업데이트된 날짜입니다. 서버는 경계 정밀도에 대한 재량이 있습니다.
_태그	리소스 태그로 검색합니다.
_profile	프로필로 태그가 지정된 모든 리소스를 검색합니다.

검색 파라미터	설명
_보안	이 리소스에 적용된 보안 레이블을 검색합니다.
_소스	리소스의 출처를 검색합니다.
_텍스트	리소스의 서술을 검색합니다.
createdAt	createdAt 사용자 지정 확장을 검색합니다.

> **Note**
>
> 다음 검색 파라미터는 2023년 12월 9일 이후에 생성된 데이터 스토어에만 지원됩니다. _security, _source, _text, createdAt.

다음 표에는 지정된 리소스 유형에 대해 지정된 데이터 유형을 기반으로 쿼리 문자열을 수정하는 방법의 예가 나와 있습니다. 명확성을 위해 예제 열의 특수 문자는 인코딩되지 않았습니다. 쿼리에 성공하려면 쿼리 문자열이 제대로 인코딩되었는지 확인합니다.

검색 파라미터 예제

파라미터 유형 검색	세부 정보	예제

숫자	지정된 리소스에서 숫자 값을 검색합니다. 중요한 수치가 관찰됩니다. 유효 자릿수는 선행 0을 제외하고 검색 파라미터 값에 따라에 고유합니다. 비교 접두사는 허용됩니다.	`[parameter]=100`
		
		`[parameter]=1e2`
		
		`[parameter]=lt100`

날짜/DateTime	특정 날짜 또는 시간을 검색합니다. 예상 형식은 `yyyy-mm-ddThh:mm:ss[Z|(+|-)hh:mm]` 이지만 다를 수 있습니다.	`[parameter]=eq2013-01-14`
		
		`[parameter]=gt2013-01-14T10:00`
		
		`[parameter]=ne2013-01-14`

파라미터 유형 검색	세부 정보	예제
	date, , dateTime, 및 데이터 형식을 허용합니다instantPeriodTiming. 검색에서 이러한 데이터 유형을 사용하는 자세한 내용은 FHIR R4 RESTful API 설명서의 날짜를 참조하세요. 비교 접두사는 허용됩니다.	
문자열	대소문자를 구분하여 문자 시퀀스를 검색합니다. HumanName 및 Address 유형을 모두 지원합니다. 자세한 내용은 FHIR R4 설명서의 HumanName 데이터 유형 항목과 Address 데이터 유형 항목을 참조하세요. 고급 검색은 :text 한정자를 사용하여 지원됩니다.	[base]/Patient?given=eve [base]/Patient?given:contains=eve

파라미터 유형 검색	세부 정보	예제
토큰	종종 한 쌍의 의료 코드 값과 비교하여 문자열에 대해 close-to-exact 일치하는 항목을 검색합니다. 대소문자 구분은 쿼리를 생성할 때 사용되는 코드 시스템에 연결됩니다.소비 기반 쿼리는 대소문자 구분과 관련된 문제를 줄이는 데 도움이 될 수 있습니다. 명확성을 위해 |는 인코딩되지 않았습니다.	`[parameter]=[system]|[code]` : 여기서 `[system]`는 코딩 시스템을 참조하고,는 해당 특정 시스템 내에 있는 코드 값을 `[code]` 참조합니다. `[parameter]=[code]` : 여기서 입력은 코드 또는 시스템과 일치합니다. `[parameter]=|[code]` : 여기서 입력은 코드와 일치하며 시스템 속성에는 식별자가 없습니다.

복합	한정자$ 및 , 작업을 사용하여 단일 리소스 유형 내에서 여러 파라미터를 검색합니다.
	
	비교 접두사는 허용됩니다.
		`/Patient?language=FR,NL&language=EN`
		
		`Observation?component-code-value-quantity=http://loinc.org|8480-6$lt60`
		
		`[base]/Group?characteristic-value=gender$mixed`

파라미터 유형 검색	세부 정보	예제

수량	숫자, 시스템 및 코드를 값으로 검색합니다. 숫자는 필수이지만 시스템 및 코드는 선택 사항입니다. 수량 데이터 유형에 따라 다릅니다. 자세한 내용은 FHIR R4 설명서의 수량을 참조하세요.
	
	다음과 같이 가정된 구문을 사용합니다. `[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`

레퍼런스	다른 리소스에 대한 참조를 검색합니다.	`[base]/Observation?subject=Patient/23`
		
		`test`

URI	특정 리소스를 명확하게 식별하는 문자열을 검색합니다.	`[base]/ValueSet?url=http://acme.org/fhir/ValueSet/123`
특별	통합된 의료 NLP 확장을 기반으로 검색합니다.	

HealthLake에서 지원하는 고급 검색 파라미터

HealthLake는 다음과 같은 고급 검색 파라미터를 지원합니다.

이름	설명	예제	기능
_includ	검색 요청에서 추가 리소스를 반환하도록 요청하는 데 사용됩니다. 대상 리소스 인스턴스에서 참조하는 리소스를 반환합니다.	`Encounter?_include=Encounter:subject`	
_revinc de	검색 요청에서 추가 리소스를 반환하도록 요청하는 데 사용됩니다. 기본 리소스 인스턴스를 참조하는 리소스를 반환합니다.	`Patient?_id=` **patient-identifier** `&_revinclude=Encounter:patient`	
_summar	요약을 사용하여 리소스의 하위 집합을 요청할 수 있습니다.	`Patient?_summary=text`	지원되는 요약 파라미터는 `_summary=true` , `_summary=false` , `_summary=text` , 입니다`_summary=data` .
_elemen	검색 결과에 리소스의 일부로 반환할 특정 요소 세트를 요청합니다.	`Patient?_elements=identifier,active,link`	
_total	검색 파라미터와 일치하는 리소스 수를 반환합니다.	`Patient?_total=accurate`	`_total=accurate` ,를 지원합니다`_total=none` .
_sort	쉼표로 구분된 목록을 사용하여 반환된 검색 결과의 정렬 순서를 지정합니다. - 접두사는 쉼표로 구분된 목록의 모든 정렬 규칙에 사용하여 내림차순을 나타낼 수 있습니다.	`Observation?_sort=status,-date`	유형별 필드 정렬을 지원합니다`Number, String, Quantity, Token, URI, Reference` . 정렬 기준 `Date`는 2023년 12월 9일 이후에 생성된 데이터 스토어에만

이름	설명	예제	기능
			지원됩니다. 최대 5개의 정렬 규칙을 지원합니다.
_count	검색 번들의 페이지당 반환되는 리소스 수를 제어합니다.	Patient?_count=100	최대 페이지 크기는 100입니다.
chainin	참조된 리소스의 요소를 검색합니다. 는 연결된 검색을 참조된 리소스 내의 요소로 . 전달합니다.	DiagnosticReport?subject:Patient.name=peter	
reverse chainin (_has)	리소스를 참조하는 리소스 요소를 기반으로 리소스를 검색합니다.	Patient?_has:Observation:patient:code=1234-5	

_include

검색 쿼리_include에서를 사용하면 지정된 추가 FHIR 리소스도 반환할 수 있습니다. _include를 사용하여 앞으로 연결된 리소스를 포함합니다.

Example- _include를 사용하여 환자 또는 환자의 그룹을 찾으려면

진단 코드를 지정하는 Condition 리소스 유형을 검색한 다음 해당 진단subject의 도 반환하도록 _include 지정합니다. Condition 리소스 유형에서 subject는 환자 리소스 유형 또는 그룹 리소스 유형을 나타냅니다.

명확성을 위해 예제의 특수 문자는 인코딩되지 않았습니다. 쿼리를 성공적으로 수행하려면 쿼리 문자열이 제대로 인코딩되었는지 확인합니다.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
Condition?code=49727002&_include=Condition:subject
```

_include:iterate 수정자

_include:iterate 한정자를 사용하면 두 수준에서 참조된 리소스를 재귀적으로 포함할 수 있습니다. 예:

```
GET /ServiceRequest?
identifier=025C0931195&_include=ServiceRequest:requester&_include:iterate=PractitionerRole:prac
```

는 ServiceRequest, 연결된 PractitionerRole(요청자 참조를 통해)을 반환한 다음 해당 PractitionerRole 재귀적으로 포함합니다. 이 한정자는 HealthLake의 모든 리소스 유형에 사용할 수 있습니다.

_include=* 수정자

_include=* 한정자는 검색 결과에서 직접 참조하는 모든 리소스를 자동으로 포함하는 와일드카드입니다. 예:

```
GET /ServiceRequest?specimen.accession=12345&_include=*
```

는 각 참조 경로를 개별적으로 지정할 필요 없이 참조하는 모든 리소스(예: 환자, 의사, 표본 등)와 함께 일치하는 ServiceRequest를 반환합니다. 이 한정자는 HealthLake의 모든 리소스 유형에 사용할 수 있습니다.

_revinclude

검색 쿼리_revinclude에서를 사용하면 지정된 추가 FHIR 리소스도 반환할 수 있습니다. _revinclude를 사용하여 역방향으로 연결된 리소스를 포함합니다.

Example- _revinclude를 사용하여 특정 환자와 연결된 관련 상황 및 관찰 리소스 유형을 포함하려면

이 검색을 수행하려면 먼저 _id 검색 파라미터에 식별자를 지정Patient하여 개인을 정의해야 합니다. 그런 다음 구조 Encounter:patient 및를 사용하여 추가 FHIR 리소스를 지정합니다Observation:patient.

명확성을 위해 예제의 특수 문자는 인코딩되지 않았습니다. 쿼리를 성공적으로 수행하려면 쿼리 문자열이 제대로 인코딩되었는지 확인합니다.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
Patient?_id=patient-
identifier&_revinclude=Encounter:patient&_revinclude=Observation:patient
```

_summary

검색 쿼리_summary에서를 사용하면 사용자가 FHIR 리소스의 하위 집합을 요청할 수 있습니다. 다음 값 중 하나를 포함할 수 있습니다`true, text, data, false`. 다른 모든 값은 유효하지 않은 것으로 처리됩니다. 반환된 리소스는 meta.tag `'SUBSETTED'`에 로 표시되어 리소스가 불완전함을 나타냅니다.

- `true`: 리소스의 기본 정의에서 'summary(요약)'로 표시된 지원되는 모든 요소를 반환합니다.
- `text`: 'text', 'id', 'meta' 요소만 반환하고 최상위 필수 요소만 반환합니다.
- `data`: 'text' 요소를 제외한 모든 부분을 반환합니다.
- `false`: 리소스(들)의 모든 부분 반환

단일 검색 요청에서는 _include 또는 _revinclude 검색 파라미터와 결합할 수 `_summary=text` 없습니다.

Example- 데이터 스토어에서 환자 리소스의 “텍스트” 요소를 가져옵니다.

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

_elements

검색 쿼리`_elements`에서를 사용하면 특정 FHIR 리소스 요소를 요청할 수 있습니다. 반환된 리소스는 meta.tag `'SUBSETTED'`에 로 표시되어 리소스가 불완전함을 나타냅니다.

`_elements` 파라미터는 리소스의 루트 수준에서 정의된 요소와 같이 쉼표로 구분된 기본 요소 이름 목록으로 구성됩니다. 나열된 요소만 반환됩니다. `_elements` 파라미터 값에 잘못된 요소가 포함된 경우 서버는 해당 요소를 무시하고 필수 요소와 유효한 요소를 반환합니다.

`_elements`는 포함된 리소스(검색 모드가 인 반환된 리소스)에는 적용되지 않습니다`include`.

단일 검색 요청에서는를 _summary 검색 파라미터와 결합할 수 `_elements` 없습니다.

Example- HealthLake 데이터 스토어에서 환자 리소스의 “식별자”, “활성”, “링크” 요소를 가져옵니다.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?
_elements=identifier,active,link
```

_total

검색 쿼리_total에서를 사용하면 요청된 검색 파라미터와 일치하는 리소스 수가 반환됩니다. HealthLake는 검색 응답의에서 일치하는 리소스(검색 모드가 인 반환된 리소스match)Bundle.total의 총 수를 반환합니다.

_total는 accurate, none 파라미터 값을 지원합니다. _total=estimate는 지원되지 않습니다. 다른 모든 값은 유효하지 않은 것으로 처리됩니다. _total는 포함된 리소스(검색 모드가 인 반환된 리소스)에는 적용되지 않습니다include.

Example- 데이터 스토어의 총 환자 리소스 수를 가져옵니다.

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

_sort

검색 쿼리_sort에서를 사용하면 결과가 특정 순서로 정렬됩니다. 결과는 쉼표로 구분된 정렬 규칙 목록을 기준으로 우선 순위에 따라 정렬됩니다. 정렬 규칙은 유효한 검색 파라미터여야 합니다. 다른 모든 값은 유효하지 않은 것으로 처리됩니다.

단일 검색 요청에서 최대 5개의 정렬 검색 파라미터를 사용할 수 있습니다. 선택적으로 - 접두사를 사용하여 내림차순을 나타낼 수 있습니다. 서버는 기본적으로 오름차순으로 정렬됩니다.

지원되는 정렬 검색 파라미터 유형은 입니다Number, String, Date, Quantity, Token, URI, Reference. 검색 파라미터가 중첩된 요소를 참조하는 경우이 검색 파라미터는 정렬에 지원되지 않습니다. 예를 들어, 리소스 유형의 '이름'에 대한 검색 환자란 HumanName 데이터 형식이 중첩된 것으로 간주되는 Patient.name 요소를 말합니다. 따라서 '이름'을 기준으로 환자 리소스를 정렬하는 것은 지원되지 않습니다.

Example- 데이터 스토어에서 환자 리소스를 가져오고 생년월일을 기준으로 오름차순으로 정렬합니다.

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

_count

파라미터_count는 단일 페이지에서 반환해야 하는 리소스 수에 대한 서버 지침으로 정의됩니다.

최대 페이지 크기는 100입니다. 100보다 큰 값은 유효하지 않습니다. _count=0는 지원되지 않습니다.

Example- 환자 리소스를 검색하고 검색 페이지 크기를 25로 설정합니다.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?_count=25
```

Chaining and Reverse Chaining(_has)

FHIR에서 연결 및 역방향 연결은 상호 연결된 데이터를 얻는 보다 효율적이고 컴팩트한 방법을 제공하여 여러 개의 개별 쿼리에 대한 필요성을 줄이고 개발자와 사용자에게 데이터 검색을 더 편리하게 만듭니다.

재귀 수준이 100개 이상의 결과를 반환하는 경우 HealthLake는 4xx를 반환하여 데이터 스토어가 오버로드되어 여러 페이지 매김이 발생하지 않도록 보호합니다.

Example- 연결 - 환자 이름이 피터인 환자를 참조하는 모든 DiagnosticReport를 가져옵니다.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/DiagnosticReport?
subject:Patient.name=peter
```

Example- 역방향 연결 - 환자 리소스를 가져옵니다. 여기서 환자 리소스는 관찰의 코드가 1234인 하나 이상의 관찰에 의해 참조되고 관찰은 환자 검색 파라미터의 환자 리소스를 참조합니다.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?
_has:Observation:patient:code=1234
```

지원되는 검색 한정자

검색 한정자는 문자열 기반 필드에 사용됩니다. HealthLake의 모든 검색 한정자는 부울 기반 로직을 사용합니다. 예를 들어, 검색 결과에 포함할 수 `:contains` 있도록 더 큰 문자열 필드에 작은 문자열을 포함하도록 지정할 수 있습니다.

지원되는 검색 한정자

검색 한정자	Type
:missing	를 제외한 모든 파라미터 `Composite`
:exact	문자열
:포함	문자열

검색 한정자	Type
:not	토큰
:text	토큰
:identifier	레퍼런스
:아래	URI

지원되는 검색 비교기

검색 비교기를 사용하여 검색에서 일치하는의 특성을 제어할 수 있습니다. 번호, 날짜 및 수량 필드를 검색할 때 비교기를 사용할 수 있습니다. 다음 표에는 HealthLake에서 지원하는 검색 비교기 및 해당 정의가 나열되어 있습니다.

지원되는 검색 비교기

검색 비교기	설명
eq	리소스의 파라미터 값이 제공된 값과 같습니다.
ne	리소스의 파라미터 값이 제공된 값과 같지 않습니다.
gt	리소스의 파라미터 값이 제공된 값보다 큽니다.
lt	리소스의 파라미터 값이 제공된 값보다 작습니다.
ge	리소스의 파라미터 값이 제공된 값보다 크거나 같습니다.
le	리소스의 파라미터 값이 제공된 값보다 작거나 같습니다.
sa	리소스의 파라미터 값은 제공된 값 이후에 시작됩니다.

검색 비교기	설명
eb	리소스의 파라미터 값은 제공된 값보다 먼저 종료됩니다.

HealthLake에서 지원하지 않는 FHIR 검색 파라미터

HealthLake는 다음 표에 나열된 파라미터를 제외한 모든 FHIR 검색 파라미터를 지원합니다. FHIR 검색 파라미터의 전체 목록은 FHIR 검색 파라미터 레지스트리를 참조하세요.

지원되지 않는 검색 파라미터

번들 구성	위치-근접
번들 식별자	Consent-source-reference
번들 메시지	계약 환자
번들 유형	Resource-content
번들 타임스탬프	리소스 쿼리

HealthLake**$operations**용 FHIR R4

FHIR $ 작업('달러 작업'이라고도 함)은 FHIR 사양의 표준 CRUD(`Create`, `Read`, `Update`, `Delete`) 작업 이상으로 확장되는 특수 서버 측 함수입니다. 이러한 작업은 "$" 접두사로 식별되며 표준 REST API 호출을 사용하여 수행하기 어렵거나 비효율적인 복잡한 처리, 데이터 변환 및 대량 작업을 가능하게 합니다. 시스템 수준, 리소스 유형 수준 또는 특정 리소스 인스턴스에서 $ 작업을 호출하여 FHIR 데이터와 상호 작용하는 유연한 방법을 제공할 수 있습니다.는 여러 FHIR를 AWS HealthLake 지원합니다`$operations`. 자세한 내용은 아래의 각 개별 페이지를 참조하세요.

주제

- HealthLake에 대한 FHIR R4 $attribution-status 작업
- 를 사용하여 리소스 유형 삭제 $bulk-delete
- $bulk-member-match HealthLake에 대한 작업
- HealthLake에 대한 FHIR R4 $confirm-attribution-list 작업
- HealthLake에 대한 FHIR R4 $davinci-data-export 작업

- 를 사용하여 임상 문서 생성 $document
- 를 사용하여 리소스 영구 제거 $erase
- 를 사용하여 환자 데이터 가져오기 Patient/$everything
- 를 사용하여 ValueSet 코드 검색 $expand
- FHIR을 사용하여 HealthLake 데이터 내보내기 $export
- HealthLake에 대한 FHIR $inquire 작업
- 를 사용하여 개념 세부 정보 검색 $lookup
- $member-add HealthLake에 대한 작업
- $member-match HealthLake에 대한 작업
- $member-remove HealthLake에 대한 작업
- 를 사용하여 환자 구획 리소스 제거 $purge
- HealthLake에 대한 FHIR $questionnaire-package 작업
- HealthLake에 대한 FHIR $submit 작업
- 를 사용하여 FHIR 리소스 검증 $validate

HealthLake에 대한 FHIR R4 **$attribution-status** 작업

특정 멤버의 어트리뷰션 상태를 검색하여 환자와 관련된 모든 어트리뷰션 리소스가 포함된 번들을 반환합니다. 이 작업은 FHIR 멤버 속성(ATR) 목록 IG 2.1.0 구현의 일부입니다.

엔드포인트

```
POST [base]/Group/[id]/$attribution-status
```

요청 파라미터

작업은 다음과 같은 선택적 파라미터를 허용합니다.

파라미터	유형	설명
memberId	식별자	어트리뷰션 상태가 요청된 멤버의 MemberId
patientReference	레퍼런스	생산자 시스템의 환자 리소스에 대한 참조

> **Note**
>
> memberId 검증을 위해 또는를 제공하거나 둘 다 제공할 patientReference 수 있습니다.

샘플 요청

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "memberId",
      "valueIdentifier": {
        "system": "http://example.org",
        "value": "MBR123456789"
      }
    },
    {
      "name": "patientReference",
      "valueReference": {
        "reference": "Patient/patient-123",
        "display": "John Doe"
      }
    }
  ]
}
```

응답

환자와 관련된 어트리뷰션 리소스가 포함된 번들을 반환합니다.

Resource	카디널리티	Location
환자	1..1	Group.member.entity
적용 범위	0..1	Group.member.extension:coverageReference
Organization/Practitioner/PractitionerRole	0..1	Group.member.extension:attributedProvider

Resource	카디널리티	Location
모든 리소스	0..1	Group.member.extension:associatedData

샘플 응답

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

오류 처리

작업은 다음 오류 조건을 처리합니다.

오류	HTTP 상태	설명
잘못된 작업 요청	400	규정 미준수 요청 파라미터 또는 구조
그룹 리소스를 찾을 수 없음	404	지정된 그룹 ID가 존재하지 않습니다.
환자 리소스를 찾을 수 없음	404	지정된 환자 참조가 존재하지 않습니다.

권한 부여 및 보안

SMART 범위 요구 사항

클라이언트는 그룹 리소스 및 관련 어트리뷰션 리소스를 읽을 수 있는 적절한 권한이 있어야 합니다.

표준 FHIR 권한 부여 메커니즘은 모든 작업에 적용됩니다.

를 사용하여 리소스 유형 삭제 **$bulk-delete**

AWS HealthLake 는 데이터 스토어 내에서 특정 유형의 모든 리소스를 삭제할 수 있도록 `$bulk-delete` 작업을 지원합니다. 이 작업은 다음이 필요한 경우에 특히 유용합니다.

- 계절별 감사 및 정리 수행
- 대규모 데이터 수명 주기 관리
- 특정 리소스 유형 제거
- 데이터 보존 정책 준수

사용법

POST 메서드를 사용하여 `$bulk-delete` 작업을 호출할 수 있습니다.

```
POST [base]/[ResourceType]/$bulk-delete?isHardDelete=false&deleteAuditEvent=true
```

Parameters

파라미터	Type	필수	기본값	설명
`isHardDelete`	부울	아니요	false	true인 경우는 스토리지에서 리소스를 영구적으로 제거합니다.
`deleteAuditEvent`	부울	아니요	true	true인 경우 연결된 감사 이벤트를 삭제합니다.
`_since`	문자열	No	데이터 스토어 생성 시간	입력하면는 lastModified 시간을 기반으로 리소스를 찾을 시작 마감 시간을 선택합니다. 시작 또는 종료에 사용할 수 없음
`start`	문자열	No	데이터 스토어 생성 시간	입력 시는 마감 시간을 선택하여 lastModified 시간을 기준으로 리소스를 찾습니다. 끝과 함께 사용할 수 있습니다.
`end`	문자열	No	작업 제출 시간	입력 시 종료 마감 시간을 선택하여 lastModified 시간을 기준으로 리소스를 찾습니다.

예제

요청 예시

```
POST [base]/Observation/$bulk-delete?isHardDelete=false
```

응답의 예

```
{
      "jobId": "jobId",
      "jobStatus": "SUBMITTED"
    }
```

작업 상태

대량 삭제 작업의 상태를 확인하려면:

```
GET [base]/$bulk-delete/[jobId]
```

작업은 작업 상태 정보를 반환합니다.

```
{
      "datastoreId": "datastoreId",
      "jobId": "jobId",
      "status": "COMPLETED",
      "submittedTime": "2025-10-09T15:09:51.336Z"
    }
```

동작

`$bulk-delete` 작업:

1. 대량의 리소스를 처리하기 위해 비동기적으로 처리
2. 데이터 무결성을 위해 ACID 트랜잭션 유지
3. 리소스 삭제 수와 함께 작업 상태 추적을 제공합니다.
4. 소프트 삭제 모드와 하드 삭제 모드 모두 지원
5. 삭제 활동에 대한 포괄적인 감사 로깅 포함
6. 기록 버전 및 감사 이벤트를 선택적으로 삭제할 수 있습니다.

감사 로깅

작업은 자세한 $bulk-delete 작업 정보와 함께 StartFHIRBulkDeleteJob 및 DescribeFHIRBulkDeleteJob으로 로깅합니다.

제한 사항

- isHardDelete를 true로 설정하면 하드 삭제된 리소스가 검색 결과 또는 _history 쿼리에 표시되지 않습니다.
- 이 작업을 통해 삭제되는 리소스는 처리 중에 일시적으로 액세스하지 못할 수 있습니다.
- 스토리지 측정은 기록 버전에서만 조정됩니다. deleteVersionHistory=false는 데이터 스토어 스토리지를 조정하지 않습니다.

$bulk-member-match HealthLake에 대한 작업

AWS HealthLake 는 여러 멤버 일치 요청을 비동기적으로 처리하기 위한 $bulk-member-match 작업을 지원합니다. 이 작업을 통해 의료 기관은 단일 대량 요청으로 인구 통계 및 적용 범위 정보를 사용하여 여러 의료 시스템에서 수백 명의 멤버의 고유 식별자를 효율적으로 일치시킬 수 있습니다. 이 기능은 대규모 payer-to-payer 데이터 교환, 멤버 전환 및 CMS 규정 준수 요구 사항에 필수적이며 FHIR 사양을 따릅니다.

> **Note**
>
> $bulk-member-match 작업은 현재 실험적이고 변경될 수 있는 기본 FHIR 사양을 기반으로 합니다. 사양이 발전함에 따라이 API의 동작과 인터페이스가 그에 따라 업데이트됩니다. 개발자는 AWS HealthLake 릴리스 정보와 관련 FHIR 사양 업데이트를 모니터링하여 통합에 영향을 미칠 수 있는 변경 사항을 최신 상태로 유지하는 것이 좋습니다.

이 작업은 다음이 필요한 경우에 특히 유용합니다.

- 공개 등록 기간 동안 대규모로 멤버 매칭 처리
- 지급인 간 대량 멤버 전환 촉진
- 대규모 CMS 규정 준수 데이터 교환 요구 사항 지원
- 의료 네트워크 전반의 구성원 코호트를 효율적으로 일치시킵니다.
- 대량 매칭 시나리오에서 API 호출을 최소화하고 운영 효율성 개선

사용법

$bulk-member-match 작업은 POST 메서드를 사용하여 그룹 리소스에서 호출되는 비동기 작업입니다.

```
POST [base]/Group/$bulk-member-match
```

대량 일치 요청을 제출한 후 다음을 사용하여 작업 상태를 폴링할 수 있습니다.

```
GET [base]/$bulk-member-match-status/{jobId}
```

지원되는 파라미터

HealthLake는 다음 FHIR $bulk-member-match 파라미터를 지원합니다.

파라미터	Type	필수	설명
MemberPatient	환자	예	일치시킬 구성원의 인구통계 정보가 포함된 환자 리소스입니다.
CoverageToMatch	적용 범위	예	기존 레코드와 일치시키는 데 사용할 적용 범위 리소스입니다.
CoverageToLink	적용 범위	아니요	매칭 프로세스 중에 연결할 적용 범위 리소스입니다.
Consent	동의	예	권한 부여를 위한 동의 리소스도 저장됩니다. 이는 동의가 필요하지 않은 개별 $member-match 작업과 다릅니다.

대량 멤버 일치 작업 제출을 위한 POST 요청

다음 예제는 대량 멤버 일치 작업을 제출하기 위한 POST 요청을 보여줍니다. 각 멤버는 필수 MemberPatient, CoverageToMatch및 Consent 리소스를 포함하는 MemberBundle 파라미터와 선택적 로 래핑됩니다CoverageToLink.

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

출력이 포함된 완료된 작업 응답

작업이 완료되면 응답에는 작업 메타데이터와 일치 결과를 분류하는 세 개의 그룹 리소스가 포함된 FHIR 파라미터 리소스가 포함됩니다.

```
{
  "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"
```

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

출력 그룹 리소스

완료된 작업은 세 개의 그룹 리소스가 포함된 파라미터 리소스를 반환합니다.

MatchedMembers 그룹

성공적으로 일치하는 모든 구성원에 대한 환자 참조를 포함하며 동의 제약으로 분류되지 않습니다.

`Group.quantity` 필드에는 각 그룹의 총 멤버 수가 포함됩니다.

NonMatchedMembers 그룹

일치 항목이 없거나 여러 일치 항목이 식별된 멤버에 대한 참조가 포함되어 있습니다.

ConsentConstrainedMembers 그룹

동의 제약 조건이 데이터 공유를 방해하는 성공적으로 일치하는 모든 멤버에 대한 환자 참조가 포함되어 있습니다. 동의 리소스는 다음 두 조건이 모두 있는 경우 제한된 것으로 간주됩니다.

- 정책 URI는 로 끝**#sensitive**납니다. 가장 명확하고 직접적인 신호입니다. 제출 지급인은 "이 구성원의 데이터는 민감합니다. 주의해서 처리합니다."라고 명시적으로 말합니다.

  ```
  "policy": [{ "uri": "...hrex-consent.html#sensitive" }]
  ```

- 최상위 프로비저닝 유형은 입니다. **deny** 멤버가 데이터 공유를 광범위하게 거부했습니다.

  ```
  "provision": { "type": "deny" }
  ```

$davinci-data-export와 통합

에서 반환한 MatchedMembers 그룹 리소스를 `$davinci-data-export` 작업과 함께 직접 사용하여 대량 멤버 데이터를 검색할 `$bulk-member-match` 수 있습니다.

```
POST [base]/Group/{matched-group-id}/$davinci-data-export
GET [base]/Group/{matched-group-id}
```

이 통합을 통해 먼저 일치하는 구성원을 대량으로 식별한 다음 결과 그룹 리소스를 사용하여 전체 상태 레코드를 내보내는 효율적인 워크플로를 사용할 수 있습니다.

성능 특성

이 `$bulk-member-match` 작업은 대량 처리를 위해 설계되었으며 비동기적으로 실행됩니다.

- 동시성: 데이터 스토어당 최대 5개의 동시 작업.
- 확장성: 요청당 최대 500명의 멤버를 처리합니다(5MB 페이로드 제한).
- 병렬 작업: 동시 가져오기, 내보내기 또는 대량 삭제 작업과 호환됩니다.

권한 부여

API는 다음 필수 범위와 함께 FHIR 인증 프로토콜에서 SMART를 사용합니다.

- `system/Patient.read` - 환자 리소스를 검색하고 일치시키는 데 필요합니다.
- `system/Coverage.read` - 적용 범위 정보를 검증하는 데 필요합니다.
- `system/Group.write` - 결과 그룹 리소스를 생성하는 데 필요합니다.
- `system/Organization.read` - 조건부, 적용 범위가 조직을 참조하는 경우 필요합니다.
- `system/Practitioner.read` - 조건부, 적용 범위가 실무자를 참조하는 경우 필수입니다.
- `system/PractitionerRole.read` - 조건부, 적용 범위가 실무자 역할을 참조하는 경우 필수입니다.
- `system/Consent.write` - 조건부, 동의 리소스가 제공된 경우 필수입니다.

또한이 작업은 프로그래밍 방식 액세스를 위한 AWS IAM 서명 버전 4(SigV4) 권한 부여를 지원합니다.

오류 처리

작업은 다음 오류 조건을 처리합니다.

- 400 잘못된 요청: 잘못된 요청 형식, 필수 파라미터 누락 또는 페이로드가 크기 제한(멤버 500명 또는 5MB)을 초과합니다.
- 422 처리 불가능한 엔터티: 작업 실행 중 처리 오류.

- 개별 멤버 오류: 특정 멤버가 처리에 실패하면 작업은 나머지 멤버로 계속되며 적절한 사유 코드와 함께 NonMatchedMembers 그룹에 오류 세부 정보가 포함됩니다. 예를 들어, 환자가 `birthDate` 파라미터가 누락된 `MemberBundle`는 다음 오류를 반환합니다.

```
"errors": [
  {
    "memberIndex": 1,
    "jsonBlob": {
      "resourceType": "OperationOutcome",
      "issue": [
        {
          "severity": "error",
          "code": "invalid",
          "diagnostics": "MemberPatient.birthDate is required"
        }
      ],
      "statusCode": 400
    }
  }
]
```

오류 세부 정보는 상태 폴링 엔드포인트를 통해 사용할 수 있으며 다음을 포함합니다.

- `numberOfMembersWithCustomerError`: 검증 또는 입력 오류가 있는 멤버 수입니다.
- `numberOfMembersWithServerError`: 서버 측 처리 오류가 있는 멤버 수입니다.

관련 작업

- the section called “$member-match” - 개별 멤버 매칭 작업입니다.
- the section called “$davinci-data-export” - 그룹 리소스를 사용하여 대량 데이터 내보내기.
- the section called “$Operations” - 지원되는 작업의 전체 목록입니다.

HealthLake에 대한 FHIR R4 **`$confirm-attribution-list`** 작업

생산자에게 소비자가 더 이상 속성 목록을 변경할 필요가 없음을 나타냅니다. 이 작업은 목록에서 비활성 멤버를 제거하고, 상태를 "최종"으로 변경하고, 작업을 승인하여 어트리뷰션 목록을 마무리합니다. 이 작업은 FHIR 멤버 속성(ATR) 목록 IG 2.1.0 구현의 일부입니다.

엔드포인트

```
POST [base]/Group/[id]/$confirm-attribution-list
```

요청

파라미터가 필요하지 않습니다.

```
POST [base]/Group/[id]/$confirm-attribution-list
```

응답

확인 메시지와 함께 HTTP 201을 반환합니다.

샘플 응답

```
HTTP Status Code: 201

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "message",
      "valueString": "Accepted."
    }
  ]
}
```

확인 후 그룹 상태

확인에 성공하면 그룹 리소스의 어트리뷰션 목록 상태가 "최종"으로 설정됩니다.

```
{
  "resourceType": "Group",
  "id": "fullexample",
  "extension": [
    {
      "url": "http://hl7.org/fhir/us/davinci-atr/StructureDefinition/ext-
attributionListStatus",
      "valueCode": "final"
    }
```

```
  ]
  // ... other Group properties
}
```

오류 처리

작업은 다음 오류 조건을 처리합니다.

오류	HTTP 상태	설명
잘못된 작업 요청	400	규정 미준수 요청 파라미터 또는 구조
그룹 리소스를 찾을 수 없음	404	지정된 그룹 ID가 존재하지 않습니다.

권한 부여 및 보안

SMART 범위 요구 사항

클라이언트는 그룹 리소스 및 관련 어트리뷰션 리소스를 읽을 수 있는 적절한 권한이 있어야 합니다.

의 경우 `$confirm-attribution-list`클라이언트에 그룹 리소스를 수정할 수 있는 쓰기 권한도 있어야 합니다.

표준 FHIR 권한 부여 메커니즘은 모든 작업에 적용됩니다.

HealthLake에 대한 FHIR R4 **`$davinci-data-export`** 작업

`$davinci-data-export` 작업은 의료 데이터를 내보내는 데 사용할 수 있는 비동기식 FHIR 작업입니다 AWS HealthLake. 이 작업은 멤버 속성(ATR), PDex 공급자 액세스, Payer-to-Payer, 멤버 액세스 APIs 등 여러 내보내기 유형을 지원합니다. DaVinci 구현 가이드의 요구 사항을 충족하도록 설계된 표준 FHIR `$export` 작업의 특수 버전입니다.

주요 기능

- 비동기 처리: 표준 FHIR 비동기 요청 패턴을 따릅니다.
- 그룹 수준 내보내기: 특정 그룹 리소스 내의 멤버에 대한 데이터를 내보냅니다.

- 여러 내보내기 유형: ATR(멤버 속성), PDex 공급자 액세스, Payer-to-Payer 및 멤버 액세스 APIs 지원
- 포괄적인 프로필 지원: US Core, CARIN 블루 버튼 및 PDex 프로필 포함
- 유연한 필터링: 환자, 리소스 유형 및 시간 범위별 필터링 지원
- NDJSON 출력: 줄 바꿈으로 구분된 JSON 형식으로 데이터를 제공합니다.

작업 엔드포인트

```
GET [base]/Group/[id]/$davinci-data-export
POST [base]/Group/[id]/$davinci-data-export
```

요청 파라미터

파라미터	카디널리티	설명
`patient`	0..*	내보낼 데이터가 있는 특정 멤버입니다. 생략하면 그룹의 모든 멤버가 내보내집니다.
`_type`	0..1	내보낼 FHIR 리소스 유형의 쉼표로 구분된 목록입니다.
`_since`	0..1	이 날짜 및 시간 이후에 업데이트된 리소스만 포함합니다.
`_until`	0..1	이 날짜 및 시간 이전에 업데이트된 리소스만 포함합니다.
`exportType`	0..1	수행할 내보내기 유형입니다. 유효한 값: `hl7.fhir.us.davinci-atr` , `hl7.fhir.us.davinci-pdex` , `hl7.fhir.us.davinci-pdex.p2p` , `hl7.fhir.us.davinci-pdex.member` . 기본값: `hl7.fhir.us.davinci-atr` .
`_includeEOB2xWoFinancial`	0..1	금융 데이터가 제거된 상태에서 CARIN BB 2.x ExplanationOfBenefit 리소스를 포함할지 여부를 지정합니다. 기본값: `false`.

지원되는 리소스 유형

지원되는 리소스 유형은 지정한 내보내기 유형에 따라 다릅니다. ATR 내보내기의 경우 다음 리소스 유형이 지원됩니다.

- `Group`
- `Patient`
- `Coverage`
- `RelatedPerson`
- `Practitioner`
- `PractitionerRole`
- `Organization`
- `Location`

PDex 내보내기(Provider Access, Payer-to-Payer, Member Access)의 경우 이전 유형 외에도 모든 임상 및 클레임 리소스 유형이 지원됩니다. 지원되는 리소스 유형의 전체 목록은 US Core 구현 가이드 (STU 6.1), CARIN 블루 버튼 구현 가이드 및 Da Vinci 사전 승인 지원 구현 가이드를 참조하세요.

내보내기 유형

`$davinci-data-export` 작업은 다음 내보내기 유형을 지원합니다. `exportType` 파라미터를 사용하여 내보내기 유형을 지정합니다.

내보내기 유형	용도	데이터 범위	시간 제한
`hl7.fhir.us.davinci-atr`	멤버 속성 목록	속성 관련 리소스	없음
`hl7.fhir.us.davinci-pdex`	공급자 액세스 API	귀속된 환자의 임상 및 클레임 데이터	5년
`hl7.fhir.us.davinci-pdex.p2p`	Payer-to-Payer 교환	보험 전환을 위한 과거 멤버 데이터	5년
`hl7.fhir.us.davinci-pdex.member`	멤버 액세스 API	멤버의 자체 상태 데이터	5년

> **Note**
>
> PDex 내보내기의 경우 ATR 리소스 유형(, , Group, Patient, Coverage, RelatedPerson, PractitionerPractitionerRole, Organization)에는 5년 시간 제한이 적용되지 않습니다Location. 이러한 리소스는 연령에 관계없이 항상 포함됩니다.

ATR(hl7.fhir.us.davinci-atr)

ATR 내보내기 유형을 사용하면 멤버 속성 목록 데이터를 내보낼 수 있습니다. 이 내보내기 유형을 사용하여 그룹 내 멤버의 어트리뷰션 관련 리소스를 검색합니다. 자세한 내용은 Da Vinci ATR 내보내기 작업을 참조하세요.

지원되는 리소스 유형

Group, Patient, Coverage, RelatedPerson, Practitioner, PractitionerRole, Organization, Location

임시 필터링

시간 필터링은 적용되지 않습니다. 일치하는 모든 리소스는 날짜와 관계없이 내보내집니다.

PDex 내보내기 유형

모든 PDex 내보내기 유형은 지원되는 동일한 프로필과 필터링 로직을 공유합니다. 자세한 내용은 Da Vinci PDex Provider Access API를 참조하세요. 지원되는 프로필은 다음과 같습니다.

- US Core 3.1.1, 6.1.0 및 7.0.0
- PDex 사전 승인(멤버 액세스에는 지원되지 않음)
- CARIN BB 2.x 기본 프로필: 입원 환자 기관, 외래 환자 기관, 전문 NonClinician, 구두, 기관

공급자 액세스(hl7.fhir.us.davinci-pdex)

네트워크 내 공급자가 귀속된 환자의 환자 데이터를 검색할 수 있습니다.

Payer-to-Payer(hl7.fhir.us.davinci-pdex.p2p)

환자가 보험을 변경할 때 지급인 간의 데이터 교환을 활성화합니다.

멤버 액세스(`hl7.fhir.us.davinci-pdex.member`)

멤버가 자신의 상태 데이터에 액세스할 수 있습니다. 이 내보내기 유형에는 클레임 리소스의 금융 데이터가 포함될 수 있습니다.

프로필 지원 및 포함 로직

PDex 내보내기의 경우 `$davinci-data-export` 작업은 `meta.profile` 요소의 프로파일 선언을 사용하여 내보내기에 포함할 리소스를 결정합니다.

ExplanationOfBenefit 리소스 처리

`ExplanationOfBenefit` (EOB) 리소스는 `meta.profile` 선언에 따라 PDex 내보내기에 포함되거나 제외됩니다.

- CARIN BB 1.x 프로파일이 있는 ExplanationOfBenefit 리소스는 내보내기에서 제외됩니다.
- `meta.profile` 집합이 없는 ExplanationOfBenefit 리소스는 내보내기에서 제외됩니다.
- CARIN BB 2.x 기본 프로파일이 있는 ExplanationOfBenefit 리소스는 항상 포함됩니다.
- 재무 데이터가 포함된 CARIN BB 2.x 프로필이 있는 ExplanationOfBenefit 리소스는 기본적으로 제외됩니다. `_includeEOB2xWoFinancial=true`이 설정되면 재무 데이터가 제거되고 리소스가 해당 기본 프로필로 변환됩니다.
- PDex 사전 승인 프로필이 있는 ExplanationOfBenefit 리소스는 항상 포함됩니다.

재무 데이터 변환

`_includeEOB2xWoFinancial=true`를 설정하면 작업은 금융 데이터를 제거하여 CARIN BB 2.x ExplanationOfBenefit 리소스를 해당 기본 프로필로 변환합니다. 예를 들어 `C4BB ExplanationOfBenefit Oral` 리소스는 FHIR 사양에 따라 레코드에서 금융 데이터를 `C4BB ExplanationOfBenefit Oral Basis`제거하는 로 변환됩니다.

변환 중에 다음 금융 데이터 요소가 제거됩니다.

- `total` 요소의 모든 조각화
- `amounttype` 조각이 있는 모든 `adjudication` 요소
- 금액 정보가 있는 모든 `item.adjudication` 요소

작업은 변환 중에 프로필 메타데이터도 업데이트합니다.

- meta.profile가 기본 프로필 정식 URL로 업데이트됨
- 버전이 CARIN BB 2.x 기본 버전으로 업데이트됨
- 데이터 스토어의 기존 리소스는 수정되지 않습니다.
- 내보낸 리소스는 데이터 스토어로 다시 유지되지 않습니다.

프로필 감지 규칙

작업은 다음 규칙을 사용하여 프로파일을 감지하고 검증합니다.

- 버전 감지는 meta.profile 정식 URLs.
- 선언된 프로파일 중 하나라도 내보내기 기준과 일치하는 경우 리소스가 포함됩니다.
- 프로파일 검증은 내보내기 처리 중에 발생합니다.

PDex 내보내기에 대한 5년 임시 필터링

모든 PDex 내보내기 유형에 대해 HealthLake는 리소스가 마지막으로 업데이트된 시간을 기준으로 5년 시간 필터를 적용합니다. 시간 필터는 연령에 관계없이 항상 내보내지는 다음 코어 속성 리소스 유형을 제외한 모든 리소스에 적용됩니다.

- Patient
- Coverage
- Organization
- Practitioner
- PractitionerRole
- RelatedPerson
- Location
- Group

이러한 관리 및 인구 통계 리소스는 내보낸 데이터에 대한 필수 컨텍스트를 제공하므로 제외됩니다. ATR 내보내기에는 시간 필터링이 적용되지 않습니다.

예제 요청

다음 예제에서는 다양한 내보내기 유형에 대해 내보내기 작업을 시작하는 방법을 보여줍니다.

ATR 내보내기

```
GET https://healthlake.{region}.amazonaws.com/datastore/
{datastoreId}/r4/Group/example-group/$davinci-data-export?
_type=Group,Patient,Coverage,Practitioner,Organization&exportType=hl7.fhir.us.davinci-
atr

POST https://healthlake.{region}.amazonaws.com/datastore/
{datastoreId}/r4/Group/example-group/$davinci-data-export?
_type=Group,Patient,Coverage,Practitioner,Organization&exportType=hl7.fhir.us.davinci-
atr
Content-Type: application/json

{
  "DataAccessRoleArn": "arn:aws:iam::444455556666:role/your-healthlake-service-role",
  "JobName": "attribution-export-job",
  "OutputDataConfig": {
    "S3Configuration": {
      "S3Uri": "s3://your-export-bucket/EXPORT-JOB",
      "KmsKeyId":
 "arn:aws:kms:region:444455556666:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    }
  }
}
```

ExplanationOfBenefit 재무 데이터 제거를 통한 공급자 액세스 내보내기

```
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 내보내기

```
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
```

특정 환자의 멤버 액세스 내보내기

```
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
```

샘플 응답

```
{
  "datastoreId": "eaee622d8406b41eb86c0f4741201ff9",
  "jobStatus": "SUBMITTED",
  "jobId": "48d7b91dae4a64d00d54b70862f33f61"
}
```

리소스 관계

작업은 멤버 속성 목록 내의 관계를 기반으로 리소스를 내보냅니다.

```
Group (Attribution List)
### Patient (Members)
### Coverage # RelatedPerson (Subscribers)
### Practitioner (Attributed Providers)
### PractitionerRole # Location
### Organization (Attributed Providers)
```

리소스 소스

Resource	소스 위치	설명
Patient	Group.member.entity	어트리뷰션 목록의 멤버인 환자
Coverage	Group.member.extension:coverageReference	환자 멤버십을 초래한 적용 범위
Organization	Group.member.extension:attributedProvider	환자가 속한 조직
Practitioner	Group.member.extension:attributedProvider	환자가 속한 개별 실무자
PractitionerRole	Group.member.extension:attributedProvider	환자가 속한 프랙티셔너 역할

Resource	소스 위치	설명
RelatedPerson	Coverage.subscriber	적용 범위 구독자
Location	PractitionerRole.location	실무자 역할과 연결된 위치
Group	입력 엔드포인트	어트리뷰션 목록 자체

작업 관리

작업 상태 확인

```
GET [base]/export/[job-id]
```

작업 취소

```
DELETE [base]/export/[job-id]
```

작업 수명 주기

- SUBMITTED - 작업이 수신되고 대기열에 있음
- IN_PROGRESS - 작업이 적극적으로 처리 중입니다.
- COMPLETED - 작업이 성공적으로 완료되었으며 파일을 다운로드할 수 있음
- FAILED - 작업에 오류가 발생했습니다.

출력 형식

- 파일 형식: NDJSON(줄 바꿈으로 구분된 JSON)
- 파일 조직: 각 리소스 유형에 대해 파일을 구분합니다.
- 파일 확장명: .ndjson
- 위치: 지정된 S3 버킷 및 경로

오류 처리

작업은 다음 조건에 대해 OperationOutcome과 함께 HTTP 400 잘못된 요청을 반환합니다.

권한 부여 오류

에 지정된 IAM 역할에 내보내기 작업을 수행할 수 있는 충분한 권한이 DataAccessRoleArn 없습니다. 필요한 S3 및 KMS 권한의 전체 목록은 내보내기 작업에 대한 권한 설정을 참조하세요.

파라미터 검증 오류

- patient 파라미터의 형식은가 아닙니다. Patient/id,Patient/id,...
- 하나 이상의 환자 참조가 유효하지 않거나 지정된 그룹에 속하지 않습니다.
- exportType 파라미터 값이 지원되는 내보내기 유형이 아닙니다.
- _type 파라미터에는 지정된 내보내기 유형에 대해 지원되지 않는 리소스 유형이 포함되어 있습니다.
- _type 파라미터에 hl7.fhir.us.davinci-atr 내보내기 유형에 필요한 리소스 유형(Group, Patient, Coverage)이 없습니다.
- _includeEOB2xWoFinancial 파라미터 값이 유효한 부울이 아닙니다.

리소스 검증 오류

- 지정된 그룹 리소스가 데이터 스토어에 존재하지 않습니다.
- 지정된 그룹 리소스에 멤버가 없음
- 하나 이상의 그룹 멤버가 데이터 스토어에 없는 환자 리소스를 참조합니다.

보안 및 권한 부여

- 표준 FHIR 권한 부여 메커니즘 적용
- 데이터 액세스 역할에는 S3 및 KMS 작업에 필요한 IAM 권한이 있어야 합니다. 필요한 권한의 전체 목록은 내보내기 작업에 대한 권한 설정을 참조하세요.

모범 사례

- 리소스 유형 선택: 내보내기 크기 및 처리 시간을 최소화하는 데 필요한 리소스 유형만 요청합니다.
- 시간 기반 필터링: 증분 내보내기에 _since 파라미터 사용
- 환자 필터링: 특정 멤버에 대한 데이터만 필요한 경우 patient 파라미터를 사용합니다.
- 작업 모니터링: 대규모 내보내기에 대한 작업 상태를 정기적으로 확인
- 오류 처리: 실패한 작업에 대한 적절한 재시도 로직 구현
- 임시 필터 인식: PDex 내보내기의 경우 리소스 유형을 선택할 때 5년 임시 필터를 고려하세요.

- 재무 데이터 제거: 재무 정보 없이 클레임 데이터가 필요한 `_includeEOB2xWoFinancial=true` 경우 사용
- 프로파일 관리: 리소스에 적절한 프로파일 선언이 있는지 확인하고, 수집 전에 대상 프로파일에 대해 검증하고, 프로파일 버전 관리를 사용하여 내보내기 동작을 제어합니다.

제한 사항

- `patient` 파라미터에 최대 500명의 환자를 지정할 수 있습니다.
- 내보내기는 그룹 수준 작업으로만 제한됩니다.
- 각 내보내기 유형에 대해 미리 정의된 리소스 유형 집합만 지원
- 출력은 항상 NDJSON 형식입니다.
- PDex 내보내기는 5년의 임상 및 클레임 데이터로 제한됩니다.
- 금융 데이터 변환은 CARIN BB 2.x ExplanationOfBenefit 프로필에만 적용됩니다.

추가 리소스

- Da Vinci 멤버 속성 목록 IG
- Da Vinci 지급인 데이터 교환 IG
- CARIN 소비자 주도 지급인 데이터 교환 IG
- 미국 코어 구현 가이드
- FHIR 대량 데이터 액세스 사양

를 사용하여 임상 문서 생성 **`$document`**

AWS HealthLake 는 이제 구성 리소스에 대한 `$document` 작업을 지원하므로 참조된 모든 리소스와 구성을 단일 응집력 있는 패키지로 번들링하여 전체 임상 문서를 생성할 수 있습니다. 이 작업은 다음을 수행해야 하는 의료 애플리케이션에 필수적입니다.

- 표준화된 임상 문서 생성
- 전체 환자 레코드 교환
- 포괄적인 임상 문서 저장
- 모든 관련 컨텍스트를 포함하는 보고서 생성

사용법

GET 및 POST 메서드를 모두 사용하여 구성 리소스에서 $document 작업을 호출할 수 있습니다.

지원되는 작업

```
GET/POST [base]/Composition/[id]/$document
```

지원되는 파라미터

HealthLake는 다음 FHIR $document 파라미터를 지원합니다.

파라미터	Type	필수	기본값	설명
persist	부울	아니요	false	서버가 생성된 문서 번들을 저장해야 하는지 여부를 나타내는 부울

예제

GET 요청

```
GET [base]/Composition/180f219f-97a8-486d-99d9-ed631fe4fc57/$document?persist=true
```

파라미터를 사용한 POST 요청

```
POST [base]/Composition/180f219f-97a8-486d-99d9-ed631fe4fc57/$document
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "persist",
      "valueBoolean": true
    }
  ]
}
```

샘플 응답

작업은 구성 및 참조된 모든 리소스가 포함된 "문서" 유형의 번들 리소스를 반환합니다.


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

동작

$document 작업:

1. 지정된 구성 리소스를 문서의 기반으로 사용합니다.
2. 구성에서 직접 참조하는 모든 리소스를 식별하고 검색합니다.
3. 구성 및 참조된 모든 리소스를 "문서" 유형의 번들로 패키징합니다.

4. 지속 파라미터가 true로 설정된 경우 생성된 문서 번들을 데이터 스토어에 저장합니다.
5. 포괄적인 문서 생성을 위해 구성에서 간접적으로 참조하는 리소스를 식별하고 검색합니다.

$document 작업은 현재 다음 형식의 리소스 참조 검색을 지원합니다.

1. ```
   GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Resource/id
   ```
2. 리소스/id

구성 리소스 내에서 지원되지 않는 리소스 참조는 생성된 문서에서 필터링됩니다.

오류 처리

작업은 다음 오류 조건을 처리합니다.

- 400 잘못된 요청: 잘못된 $document 작업(부적합 요청) 또는 지속이 true로 설정된 경우 필터링된 참조로 인해 결과 문서가 FHIR 검증에 실패하는 경우
- 404 찾을 수 없음: 구성 리소스를 찾을 수 없음

$document 작업 사양에 대한 자세한 내용은 FHIR R4 구성 $document 설명서를 참조하세요.

를 사용하여 리소스 영구 제거 **$erase**

AWS HealthLake 는 $erase 작업을 지원하여 특정 리소스와 해당 기록 버전을 영구적으로 삭제할 수 있습니다. 이 작업은 다음이 필요한 경우에 특히 유용합니다.

- 개별 리소스 영구 제거
- 특정 버전 기록 삭제
- 개별 리소스 수명 주기 관리
- 특정 데이터 제거 요구 사항 준수

사용법

$erase 작업은 두 가지 수준에서 호출할 수 있습니다.

리소스 인스턴스 수준
   ```

```
POST [base]/[ResourceType]/[ID]/$erase?deleteAuditEvent=true
```

버전별 수준

```
POST [base]/[ResourceType]/[ID]/_history/[VersionID]/$erase
```

Parameters

파라미터	Type	필수	기본값	설명
deleteAuditEvent	부울	아니요	false	true인 경우 연결된 감사 이벤트를 삭제합니다.

예제

요청 예시

```
POST [base]/Patient/example-patient/$erase
```

응답의 예

```
{
      "jobId": "5df47e2f51ff3c731847678cb8cad48e",
      "jobStatus": "SUBMITTED"
    }
```

작업 상태

지우기 작업의 상태를 확인하려면:

```
GET [base]/$erase/[jobId]
```

작업은 작업 상태 정보를 반환합니다.

```
{
      "datastoreId": "36622996b1fcecb7e12ee2ee085308d3",
      "jobId": "5df47e2f51ff3c731847678cb8cad48e",
```

```
    "status": "COMPLETED",
    "submittedTime": "2025-10-30T16:39:24.160Z"
}
```

동작

$erase 작업:

1. 데이터 무결성을 보장하기 위해 비동기적으로 처리
2. ACID 트랜잭션 유지
3. 작업 상태 추적 제공
4. 지정된 리소스와 해당 버전을 영구적으로 제거합니다.
5. 삭제 활동에 대한 포괄적인 감사 로깅 포함
6. 감사 이벤트의 선택적 삭제 지원

감사 로깅

$erase 작업은 사용자 ID, 타임스탬프 및 리소스 세부 정보가 포함된 DeleteResource로 로깅됩니다.

제한 사항

- $erased 리소스는 검색 결과 또는 _history 쿼리에 표시되지 않습니다.
- 삭제되는 리소스는 처리 중에 일시적으로 액세스하지 못할 수 있습니다.
- 리소스가 영구적으로 제거되면 스토리지 측정이 즉시 조정됩니다.

를 사용하여 환자 데이터 가져오기 **Patient/$everything**

Patient/$everything 작업은 해당와 관련된 다른 Patient 리소스와 함께 FHIR 리소스를 쿼리하는 데 사용됩니다Patient. 작업은 환자에게 전체 레코드에 대한 액세스 권한을 제공하거나 공급자가 환자와 관련된 대량 데이터 다운로드를 수행하는 데 사용할 수 있습니다. HealthLake는 특정 환자에 Patient/$everything 대해를 지원합니다id.

Patient/$everything는 아래 예제와 같이 호출할 수 있는 FHIR REST API 작업입니다.

GET request

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything
```

> **Note**
>
> 응답 리소스는 리소스 유형 및 리소스 별로 정렬됩니다`id`.
> 응답은 항상 로 채워집니다`Bundle.total`.

`Patient/$everything` 파라미터

HealthLake는 다음 쿼리 파라미터를 지원합니다.

파라미터	세부 정보
시작	지정된 시작 날짜 이후의 모든 `Patient` 데이터를 가져옵니다.
최종	지정된 종료 날짜 이전의 모든 `Patient` 데이터를 가져옵니다.
since	지정된 날짜 이후에 모든 `Patient` 데이터를 업데이트합니다.
_유형	특정 리소스 유형에 대한 `Patient` 데이터를 가져옵니다.
_개	`Patient` 데이터를 가져오고 페이지 크기를 지정합니다.

Example- 지정된 시작 날짜 이후의 모든 환자 데이터 가져오기

`Patient/$everything`는 `start` 필터를 사용하여 특정 날짜 이후의 데이터만 쿼리할 수 있습니다.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?start=2024-03-15T00:00:00.000Z
```

Example- 지정된 종료 날짜 이전의 모든 Patient 데이터 가져오기

환자 $everything은 end 필터를 사용하여 특정 날짜 이전의 데이터만 쿼리할 수 있습니다.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?end=2024-03-15T00:00:00.000Z
```

Example- 지정된 날짜 이후에 업데이트된 모든 Patient 데이터 가져오기

`Patient/$everything`는 `since` 필터를 사용하여 특정 날짜 이후에 업데이트된 데이터만 쿼리할 수 있습니다.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?since=2024-03-15T00:00:00.000Z
```

Example- 특정 리소스 유형에 대한 Patient 데이터 가져오기

환자 $everything은 _type 필터를 사용하여 응답에 포함할 특정 리소스 유형을 지정할 수 있습니다. 쉼표로 구분된 목록에서 여러 리소스 유형을 지정할 수 있습니다.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?_type=Observation,Condition
```

Example- Patient 데이터 가져오기 및 페이지 크기 지정

환자 $everything은 _count를 사용하여 페이지 크기를 설정할 수 있습니다.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?_count=15
```

Patient/$everything start 및 **end** 속성

HealthLake는 Patient/ $everything start 및 end 쿼리 파라미터에 대해 다음과 같은 리소스 속성을 지원합니다.

Resource	리소스 요소
Account	Account.servicePeriod.start
AdverseEvent	AdverseEvent.date
AllergyIntolerance	AllergyIntolerance.recordedDate
예약	Appointment.start
AppointmentResponse	AppointmentResponse.start
AuditEvent	AuditEvent.period.start

Resource	리소스 요소
기본	Basic.created
BodyStructure	NO_DATE
CarePlan	CarePlan.period.start
CareTeam	CareTeam.period.start
ChargeItem	ChargeItem.occurrenceDateTime, ChargeItem.occurrencePeriod.start, ChargeItem.occurrenceTiming.event
Claim	Claim.billablePeriod.start
ClaimResponse	ClaimResponse.created
ClinicalImpression	ClinicalImpression.date
통신	Communication.sent
CommunicationRequest	CommunicationRequest.occurrenceDateTime, CommunicationRequest.occurrencePeriod.start
구성	Composition.date
조건	Condition.recordedDate
동의	consent.dateTime
적용 범위	Coverage.period.start
CoverageEligibilityRequest	CoverageEligibilityRequest.created

Resource	리소스 요소
CoverageEligibilityResponse	CoverageEligibilityResponse.created
DetectedIssue	DetectedIssue.identified
DeviceRequest	DeviceRequest.authoredOn
DeviceUseStatement	DeviceUseStatement.recordedOn
DiagnosticReport	DiagnosticReport.effective
DocumentManifest	DocumentManifest.created
DocumentReference	DocumentReference.context.period.start
상황	Encounter.period.start
EnrollmentRequest	EnrollmentRequest.created
EpisodeOfCare	EpisodeOfCare.period.start
ExplanationOfBenefit	ExplanationOfBenefit.billablePeriod.start
FamilyMemberHistory	NO_DATE
플래그	Flag.period.start

Resource	리소스 요소
Goal	Goal.statusDate
Group	NO_DATE
ImagingStudy	ImagingStudy.started
면역화	Immunization.recorded
ImmunizationEvaluation	ImmunizationEvaluation.date
ImmunizationRecommendation	ImmunizationRecommendation.date
Invoice	Invoice.date
List	List.date
MeasureReport	MeasureReport.period.start
미디어	Media.issued
MedicationAdministration	MedicationAdministration.effective
MedicationDispense	MedicationDispense.whenPrepared
MedicationRequest	MedicationRequest.authoredOn
MedicationStatement	MedicationStatement.dateAsserted

Resource	리소스 요소
Molecular Sequence	NO_DATE
Nutrition Order	NutritionOrder.dateTime
관측치	Observation.effective
환자	NO_DATE
Person	NO_DATE
절차	Procedure.performed
증명	Provenance.occurredPeriod.start, Provenance.occurredDateTime
QuestionnaireResponse	QuestionnaireResponse.authored
RelatedPerson	NO_DATE
RequestGroup	RequestGroup.authoredOn
ResearchSubject	ResearchSubject.period
RiskAssessment	RiskAssessment.occurrenceDateTime, RiskAssessment.occurrencePeriod.start
일정	Schedule.planningHorizon
ServiceRequest	ServiceRequest.authoredOn
샘플	©.receivedTime

Resource	리소스 요소
SupplyDelivery	SupplyDelivery.occurrenceDateTime, SupplyDelivery.occurrencePeriod.start, SupplyDelivery.occurrenceTiming.event
SupplyRequest	SupplyRequest.authoredOn
VisionPrescription	VisionPrescription.dateWritten

를 사용하여 ValueSet 코드 검색 **$expand**

AWS HealthLake 는 이제 고객이 수집한 ValueSets에 대한 $expand 작업을 지원하므로 해당 ValueSet 리소스(들)에 포함된 코드의 전체 목록을 검색할 수 있습니다. 이 작업은 다음이 필요한 경우에 특히 유용합니다.

- 검증을 위해 가능한 모든 코드 검색
- 사용자 인터페이스에 사용 가능한 옵션 표시
- 특정 용어 컨텍스트 내에서 포괄적인 코드 조회 수행

사용법

$expand 작업은 GET 및 POST 메서드를 모두 사용하여 ValueSet 리소스에서 호출할 수 있습니다.

지원되는 작업

```
GET/POST [base]/ValueSet/[id]/$expand
GET [base]/ValueSet/$expand?url=http://example.com
POST [base]/ValueSet/$expand
```

지원되는 파라미터

HealthLake는 FHIR R4 $expand 파라미터의 하위 집합을 지원합니다.

파라미터	Type	필수	설명
`url`	uri	아니요	확장할 ValueSet의 정식 URL
`id`	id	아니요	확장할 ValueSet 리소스 ID(GET 또는 POST 작업의 경우)
`filter`	문자열	No	코드 확장 결과 필터링
`count`	정수	아니요	반환할 코드 수
`offset`	정수	아니요	반환하기 전에 건너뛸 일치하는 코드 수입니다. 필터링 후 일치 코드에만 적용되며 원래 ValueSet의 필터링되지 않은 전체 콘텐츠에는 적용되지 않습니다.

예제

ID별 GET 요청

```
GET [base]/ValueSet/example-valueset/$expand
```

필터를 사용한 URL별 GET 요청

```
GET [base]/ValueSet/$expand?url=http://example.com/ValueSet/my-
valueset&filter=male&count=5
```

파라미터를 사용한 POST 요청(ID 기준)

```
POST [base]/ValueSet/example-valueset/$expand
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "count",
      "valueInteger": 10
```

```
    },
    {
      "name": "filter",
      "valueString": "admin"
    }
  ]
}
```

파라미터를 사용한 POST 요청(URL 기준)

```
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
    }
  ]
}
```

샘플 응답

작업은 확장된 코드가 포함된 expansion 요소와 함께 ValueSet 리소스를 반환합니다.

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

응답에는 다음이 포함됩니다.

- expansion.total: 확장된 ValueSet의 총 코드 수
- expansion.contains: 확장된 코드와 시스템, 코드 및 표시 값의 배열
- expansion.parameter: 확장 요청에 사용되는 파라미터

`$expand` 작업 사양에 대한 자세한 내용은 FHIR R4 ValueSet `$expand` 설명서를 참조하세요.

FHIR을 사용하여 HealthLake 데이터 내보내기 **`$export`**

FHIR $export 작업을 사용하여 HealthLake 데이터 스토어에서 대량으로 데이터를 내보낼 수 있습니다. HealthLake는 `POST` 및 `GET` 요청을 `$export` 사용하여 FHIR을 지원합니다. 를 사용하여 내보내기

요청을 수행하려면 필요한 권한이 있는 IAM 사용자, 그룹 또는 역할이 있어야 하며 요청의 $export 일부로를 지정하고 요청 본문에 원하는 파라미터를 포함해야 POST합니다.

> **Note**
>
> FHIR을 사용하여 이루어진 모든 HealthLake 내보내기 요청은 ndjson 형식으로 반환되고 Amazon S3 버킷으로 내보내$export집니다. 여기서 각 Amazon S3 객체에는 단일 FHIR 리소스 유형만 포함됩니다.
> AWS 계정 서비스 할당량당 내보내기 요청을 대기열에 넣을 수 있습니다. 자세한 내용은 Service Quotas 단원을 참조하십시오.

HealthLake는 다음과 같은 세 가지 유형의 대량 내보내기 엔드포인트 요청을 지원합니다.

HealthLake 대량 **$export** 유형

내보내기 유형	설명	구문
시스템	HealthLake FHIR 서버에서 모든 데이터를 내보냅니다.	`POST https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/$export`

모든 환자	환자 리소스 유형과 연결된 리소스 유형을 포함하여 모든 환자와 관련된 모든 데이터를 내보냅니다.	`POST https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/Patient/$export`
		
		`GET https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/Patient/$export`

환자 그룹	그룹 ID로 지정된 환자 그룹과 관련된 모든 데이터를 내보냅니다.	`POST https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/Group/ id/$export`

내보내기 유형	설명	구문
		`GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Group/id/$export`

시작하기 전 준비 사항

HealthLake용 FHIR REST API를 사용하여 내보내기 요청을 하려면 다음 요구 사항을 충족해야 합니다.

- 내보내기 요청을 수행하는 데 필요한 권한이 있는 사용자, 그룹 또는 역할을 설정해야 합니다. 자세한 내용은 $export 요청 권한 부여를 참조하세요.
- 데이터를 내보낼 Amazon S3 버킷에 대한 HealthLake 액세스 권한을 부여하는 서비스 역할을 생성해야 합니다. 또한 서비스 역할은 HealthLake를 서비스 보안 주체로 지정해야 합니다. 권한 설정에 대한 자세한 내용은 섹션을 참조하세요내보내기 작업에 대한 권한 설정.

$export 요청 권한 부여

FHIR REST API를 사용하여 내보내기 요청을 성공적으로 수행하려면 IAM 또는 OAuth2.0. 서비스 역할도 있어야 합니다.

IAM을 사용하여 요청 권한 부여

$export 요청을 할 때 사용자, 그룹 또는 역할에 IAM 작업이 정책에 포함되어야 합니다. 자세한 내용은 내보내기 작업에 대한 권한 설정 단원을 참조하십시오.

FHIR(OAuth 2.0)에서 SMART를 사용하여 요청 권한 부여

FHIR 지원 HealthLake 데이터 스토어의 SMART에서 $export 요청할 때는 적절한 범위가 할당되어 있어야 합니다. 자세한 내용은 HealthLake의 FHIR 리소스 범위에 대한 SMART 단원을 참조하십시오.

> **Note**
>
> GET 요청이 $export 있는 FHIR은 내보내기 및 검색 파일을 요청하기 위해 동일한 인증 방법 또는 보유자 토큰(FHIR에서 SMART인 경우)이 필요합니다. $export에서 FHIR을 사용하여 내보낸 파일은 48시간 동안 다운로드할 GET 수 있습니다.

$export 요청하기

이 섹션에서는 FHIR REST API를 사용하여 내보내기 요청을 할 때 수행해야 하는 필수 단계를 설명합니다.

AWS 계정에 실수로 요금이 부과되지 않도록 $export 구문을 제공하지 않고 POST 요청하여 요청을 테스트하는 것이 좋습니다.

요청을 하려면 다음을 수행해야 합니다.

1. 지원되는 엔드포인트에 대한 POST 요청 URL$export에서를 지정합니다.
2. 필요한 헤더 파라미터를 지정합니다.
3. 필요한 파라미터를 정의하는 요청 본문을 지정합니다.

1단계: 지원되는 엔드포인트의 **POST** 요청 URL**$export**에서를 지정합니다.

HealthLake는 세 가지 유형의 대량 내보내기 엔드포인트 요청을 지원합니다. 대량 내보내기 요청을 하려면 지원되는 세 엔드포인트 중 하나에 대해 POST기반 요청을 해야 합니다. 다음 예제에서는 요청 URL$export에서를 지정할 위치를 보여줍니다.

- `POST https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/$export`
- `POST https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/$export`
- `POST https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Group/id/$export`

POST 요청 문자열에서 다음과 같이 지원되는 검색 파라미터를 사용할 수 있습니다.

지원되는 검색 파라미터

HealthLake는 대량 내보내기 요청에서 다음과 같은 검색 한정자를 지원합니다.

다음 예제에는 요청을 제출하기 전에 인코딩해야 하는 특수 문자가 포함되어 있습니다.

이름	필수?	설명	예제
`_outputFormat`	아니요	생성할 요청된 대량 데이터 파일의 형식입니다. 허용되는 값은 `application/fhir+ndjson` , `application/ndjson` , 입니다`ndjson`.	
`_type`	아니요	내보내기 작업에 포함할 쉼표로 구분된 FHIR 리소스 유형의 문자열입니다. 모든 리소스를 내보낼 때 비용에 영향을 미칠 수 `_type` 있으므로를 포함하는 것이 좋습니다.	`&_type=MedicationStatement,Observation`
`_since`	아니요	날짜 타임스탬프 당일 또는 이후에 수정된 리소스 유형입니다. 리소스 유형에 마지막으로 업데이트된 시간이 없는 경우 응답에 포함됩니다.	`&_since=2024-05-09T00%3A00%3A00Z`
`_until`	아니요	날짜 타임스탬프 또는 그 이전에 수정된 리소스 유형입니다. 내보내기를 위한 특정 시간	`&_until=2024-12-31T23%3A59%3A59Z`

이름	필수?	설명	예제
		범위를 정의_since하기 위해와 함께 사용됩니다. 리소스 유형에 마지막으로 업데이트된 시간이 없는 경우 응답에서 제외됩니다.	

2단계: 필수 헤더 파라미터 지정

FHIR REST API를 사용하여 내보내기 요청을 하려면 다음 헤더 파라미터를 지정해야 합니다.

- 콘텐츠 타입: `application/fhir+json`
- 선호 사항: `respond-async`

다음으로 요청 본문에 필수 요소를 지정해야 합니다.

3단계: 필요한 파라미터를 정의하는 요청 본문을 지정합니다.

내보내기 요청에는 JSON 형식의 본문도 필요합니다. 본문에는 다음 파라미터가 포함될 수 있습니다.

Key(키)	필수?	설명	값
DataAccessRoleArn	예	HealthLake 서비스 역할의 ARN입니다. 사용되는 서비스 역할은 HealthLake를 서비스 보안 주체로 지정해야 합니다.	`arn:aws:iam:: 444455556666 :role/your-healthlake-service-role`
JobName	아니요	내보내기 요청의 이름입니다.	**your-export-job-name**
S3Uri	예	OutputDataConfig 키의 일부입니다. 내보낸 데이터를 다운로드할	`s3://amzn-s3-demo-bucket/ EXPORT-JOB /`

Key(키)	필수?	설명	값
		대상 버킷의 S3 URI입니다.	
KmsKeyId	예	OutputDataConfig 키의 일부입니다. Amazon S3 버킷을 보호하는 데 사용되는 AWS KMS 키의 ARN입니다.	arn:aws:kms: **region-of-bucket:123456789012** :key/**1234abcd-12ab-34cd-56ef-1234567890ab**

Example FHIR REST API를 사용하여 수행된 내보내기 요청의 본문

FHIR REST API를 사용하여 내보내기 요청을 하려면 다음과 같이 본문을 지정해야 합니다.

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

요청이 성공하면 다음과 같은 응답을 받게 됩니다.

응답 헤더

```
content-location: https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
export/your-export-request-job-id
```

응답 본문

```
{
```

```
  "datastoreId": "your-data-store-id",
  "jobStatus": "SUBMITTED",
  "jobId": "your-export-request-job-id"
}
```

내보내기 요청 관리

내보내기 요청이 성공하면를 사용하여 요청을 관리$export하여 현재 내보내기 요청의 상태를 설명하고 현재 내보내기 요청을 $export 취소할 수 있습니다.

REST API를 사용하여 내보내기 요청을 취소하면 취소 요청을 제출한 시점까지 내보낸 데이터 부분에 대해서만 요금이 청구됩니다.

다음 주제에서는 현재 내보내기 요청의 상태를 가져오거나 취소할 수 있는 방법을 설명합니다.

내보내기 요청 취소

내보내기 요청을 취소하려면 DELETE 요청을 하고 요청 URL에 작업 ID를 입력합니다.

```
DELETE https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/export/your-export-request-job-id
```

요청이 성공하면 다음을 수신합니다.

```
{
  "exportJobProperties": {
    "jobId": "your-original-export-request-job-id",
    "jobStatus": "CANCEL_SUBMITTED",
    "datastoreId": "your-data-store-id"
  }
}
```

요청이 성공하지 못하면 다음을 수신합니다.

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "not-supported",
```

```
      "diagnostics": "Interaction not supported."
    }
  ]
}
```

내보내기 요청 설명

내보내기 요청의 상태를 가져오려면 `export` 및를 사용하여 GET 요청합니다**`export-request-job-id`**.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/export/your-export-request-id
```

JSON 응답에는 `ExportJobProperties` 객체가 포함됩니다. 여기에는 다음과 같은 키:값 페어가 포함될 수 있습니다.

이름	필수?	설명	값
DataAccessRoleArn	아니요	HealthLake 서비스 역할의 ARN입니다. 사용되는 서비스 역할은 HealthLake를 서비스 보안 주체로 지정해야 합니다.	`arn:aws:iam::` **`444455556666`** `:role/`**`your-healthlake-service-role`**
SubmitTime	아니요	내보내기 작업이 제출된 날짜입니다.	`Apr 21, 2023 5:58:02`
EndTime	아니요	내보내기 작업이 완료된 시간입니다.	`Apr 21, 2023 6:00:08 PM`
JobName	아니요	내보내기 요청의 이름입니다.	**`your-export-job-name`**
JobStatus	아니요		유효값은 다음과 같습니다. `SUBMITTED | IN_PROGRESS | COMPLETED`

이름	필수?	설명	값
			_WITH_ERRORS | COMPLETED | FAILED
S3Uri	예	OutputDataConfig 객체의 일부입니다. 내보낸 데이터를 다운로드할 대상 버킷의 Amazon S3 URI입니다.	s3://amzn-s3-demo-bucket/ **EXPORT-JOB** /
KmsKeyId	예	OutputDataConfig 객체의 일부입니다. Amazon S3 버킷을 보호하는 데 사용되는 AWS KMS 키의 ARN입니다.	arn:aws:kms: **region-of-bucket:123456789012** :key/**1234abcd-12ab-34cd-56ef-1234567890ab**

Example: FHIR REST API를 사용하여 수행된 설명 내보내기 요청의 본문

성공하면 다음과 같은 JSON 응답을 받게 됩니다.

```
{
  "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",
  }
}
```

HealthLake에 대한 FHIR **$inquire** 작업

$inquire 작업을 통해 이전에 제출한 사전 권한 부여 요청의 상태를 확인할 수 있습니다. 이 작업은 Da Vinci 사전 승인 지원(PAS) 구현 가이드를 구현하여 현재 권한 부여 결정을 검색하는 표준화된 FHIR 기반 워크플로를 제공합니다.

작동 방식

- 쿼리 제출: 확인하려는 클레임과 지원 정보가 포함된 FHIR 번들을 보냅니다.
- 검색: HealthLake는 데이터 스토어에서 해당 ClaimResponse를 검색합니다.
- 검색: 가장 최근 권한 부여 상태가 검색됩니다.
- 응답: 현재 권한 부여 상태(대기열 있음, 승인됨, 거부됨 등)의 즉각적인 응답을 받습니다.

> **Note**
>
> $inquire는 기존 권한 부여 상태를 검색하는 읽기 전용 작업입니다. 데이터 스토어의 리소스는 수정하거나 업데이트하지 않습니다.

API 엔드포인트

```
POST /datastore/{datastoreId}/r4/Claim/$inquire
Content-Type: application/fhir+json
```

요청 구조

번들 요구 사항

요청은 다음을 포함하는 FHIR 번들 리소스여야 합니다.

- Bundle.type: 이어야 함 "collection"
- Bundle.entry: 다음을 사용하여 정확히 하나의 클레임 리소스를 포함해야 합니다.
  - use = "preauthorization"

  - `status = "active"`
- 참조 리소스: 클레임에서 참조하는 모든 리소스가 번들에 포함되어야 합니다.

> **Query-by-Example**
>
> 입력 번들의 리소스는 검색 템플릿 역할을 합니다. HealthLake는 제공된 정보를 사용하여 해당 ClaimResponse를 찾습니다.

필수 리소스

Resource	카디널리티	프로필	설명
클레임	1	PAS 클레임 문의	문의하려는 사전 권한 부여
환자	1	PAS 수혜자 환자	환자 인구통계 정보
조직(보험사)	1	PAS 보험 회사 조직	보험 회사
조직(제공자)	1	PAS 요청 조직	요청을 제출한 의료 서비스 제공자

중요 검색 기준

HealthLake는 다음을 사용하여 ClaimResponse를 검색합니다.

- 클레임의 환자 참조
- 클레임의 보험 회사 참조
- 클레임의 공급자 참조
- 클레임에서 생성된 날짜(시간 필터)

> **환자별 쿼리만**
>
> 모든 문의는 특정 환자와 연결되어야 합니다. 환자 식별이 없는 시스템 전체 쿼리는 허용되지 않습니다.

요청 예제

```
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"
      }
```

```
    }
  ]
}
```

응답 형식

성공 응답(200 OK)

다음이 포함된 PAS 쿼리 응답 번들을 받게 됩니다.

- 현재 권한 부여 상태의 ClaimResponse, 검색 기준과 일치하는 경우 여러 ClaimResponse
- 요청의 모든 원본 리소스(되돌아옴)
- 응답이 어셈블된 시간의 타임스탬프

가능한 ClaimResponse 결과

결과	설명
queued	권한 부여 요청이 아직 검토 보류 중입니다.
complete	권한 부여 결정(disposition 승인/거부 확인)
error	처리 중에 오류가 발생했습니다.
partial	부분 권한 부여

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

오류 응답

400 잘못된 요청

요청 형식이 유효하지 않거나 검증이 실패할 때 반환됩니다.

```
{
    "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 권한이 없음

인증 자격 증명이 누락되거나 유효하지 않을 때 반환됩니다.

```
{
    "resourceType": "OperationOutcome",
    "issue": [
        {
            "severity": "error",
            "code": "forbidden",
            "diagnostics": "Invalid authorization header"
        }
    ]
```

```
}
```

403 금지됨

인증된 사용자에게 요청된 리소스에 액세스할 수 있는 권한이 없는 경우 반환됩니다.

```
{
    "resourceType": "OperationOutcome",
    "issue": [
        {
            "severity": "error",
            "code": "exception",
            "diagnostics": "Insufficient SMART scope permissions."
        }
    ]
}
```

400 찾을 수 없는 경우

쿼리에 대해 일치하는 ClaimResponse를 찾을 수 없는 경우 반환됩니다.

```
{
  "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 지원되지 않는 미디어 유형

Content-Type 헤더가 application/fhir+json이 아닐 때 반환됩니다.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "value",
    "diagnostics": "Incorrect MIME-type. Update request Content-Type header."
```

```
  }]
}
```

429 요청이 너무 많음

속도 제한을 초과하면 반환됩니다.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "throttled",
    "diagnostics": "Rate limit exceeded. Please retry after some time."
  }]
}
```

검증 규칙

HealthLake는 문의에 대해 포괄적인 검증을 수행합니다.

번들 검증

- PAS 문의 요청 번들 프로필을 준수해야 합니다.
- `Bundle.type` 여야 합니다. `"collection"`
- 정확히 하나의 클레임 리소스를 포함해야 합니다.
- 참조된 모든 리소스가 번들에 포함되어야 합니다.

클레임 검증

- PAS 클레임 쿼리 프로필을 준수해야 합니다.
- `Claim.use` 여야 합니다. `"preauthorization"`
- `Claim.status` 여야 합니다. `"active"`
- 필수 필드: `patient`, `insurer`, `provider`, `created`

리소스 검증

- 모든 리소스는 해당 PAS 쿼리 프로필을 준수해야 합니다.
- 필요한 지원 리소스가 있어야 합니다(환자, 보험 조직, 공급자 조직).

- 교차 참조는 번들 내에서 유효하고 해석 가능해야 합니다.

성능 사양

지표	사양
리소스 수 제한	번들당 리소스 500개
번들 크기 제한	최대 5MB

필수 권한

`$inquire` 작업을 사용하려면 IAM 역할에 다음이 있는지 확인합니다.

- `healthlake:InquirePreAuthClaim` - 작업을 호출하려면

FHIR 범위의 SMART

최소 필수 범위:

- SMART v1: `user/ClaimResponse.read`
- SMART v2: `user/ClaimResponse.s`

중요한 구현 참고 사항

검색 동작

문의를 제출하면 HealthLake는 다음을 사용하여 ClaimResponse를 검색합니다.

- 입력 클레임의 환자 참조
- 입력 클레임의 보험 회사 참조
- 입력 클레임의 공급자 참조
- 입력 클레임에서 생성된 날짜(시간 필터)

다중 일치: 여러 ClaimResponses 검색 기준과 일치하는 경우 HealthLake는 일치하는 모든 결과를 반환합니다. 최신 `ClaimResponse.created` 타임스탬프를 사용하여 최신 상태를 식별해야 합니다.

업데이트된 클레임

동일한 사전 권한 부여(예: 클레임 v1.1, v1.2, v1.3)에 대한 여러 업데이트를 제출한 경우 `$inquire` 작업은 제공된 검색 기준에 따라 최신 버전과 연결된 ClaimResponse를 검색합니다.

읽기 전용 작업

`$inquire` 작업:

- 기존 권한 부여 상태를 검색합니다.
- 최신 ClaimResponse를 반환합니다.
- 리소스를 수정하거나 업데이트하지 않음
- 새 리소스를 생성하지 않음
- 새 권한 부여 처리를 트리거하지 않음

워크플로 예시

일반적인 사전 승인 문의 워크플로

```
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"
```

관련 작업

- `Claim/$submit` - 새 사전 권한 부여 요청을 제출하거나 기존 권한 부여 요청을 업데이트합니다.
- `Patient/$everything` - 사전 권한 부여 컨텍스트에 대한 포괄적인 환자 데이터 검색

를 사용하여 개념 세부 정보 검색 **$lookup**

AWS HealthLake 는 이제 CodeSystem 리소스에 대한 `$lookup` 작업을 지원하므로 코드와 같은 식별 정보를 제공하여 코드 시스템의 특정 개념에 대한 세부 정보를 검색할 수 있습니다. 이 작업은 다음이 필요한 경우에 특히 유용합니다.

- 특정 의료 코드에 대한 세부 정보 검색
- 코드 의미 및 속성 검증
- 액세스 개념 정의 및 관계
- 정확한 용어 데이터로 임상 의사 결정 지원

사용법

`$lookup` 작업은 GET 및 POST 메서드를 모두 사용하여 CodeSystem 리소스에서 호출할 수 있습니다.

지원되는 작업

```
GET [base]/CodeSystem/$lookup?system=http://snomed.info/
sct&code=73211009&version=20230901
POST [base]/CodeSystem/$lookup
```

지원되는 파라미터

HealthLake는 FHIR R4 `$lookup` 파라미터의 하위 집합을 지원합니다.

파라미터	Type	필수	설명
`code`	code	예	찾고 있는 개념 코드(예: SNOMED CT의 "71620000")
`system`	uri	예	코드 시스템의 정식 URL(예: "http://snomed.info/sct")
`version`	문자열	No	코드 시스템의 특정 버전

예제

GET 요청

```
GET [base]/CodeSystem/$lookup?system=http://snomed.info/
sct&code=71620000&version=2023-09
```

POST 요청

```
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"
    }
  ]
}
```

샘플 응답

작업은 개념 세부 정보가 포함된 파라미터 리소스를 반환합니다.

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

응답 파라미터

응답에는 사용 가능한 경우 다음 파라미터가 포함됩니다.

파라미터	유형	설명
name	문자열	코드 시스템 이름
version	문자열	코드 시스템 버전
display	문자열	개념의 표시 이름
designation	BackboneElement	이 개념에 대한 추가 표현입니다.
property	BackboneElement	개념의 추가 속성(정의, 관계 등)

동작

$lookup 작업:

1. 필수 파라미터(code 및 system)를 검증합니다.
2. 데이터 스토어에 저장된 지정된 코드 시스템 내에서 개념을 검색합니다.
3. 표시 이름, 지정 및 속성을 포함한 자세한 개념 정보를 반환합니다.
4. version 파라미터가 제공될 때 버전별 조회 지원
5. HealthLake 데이터 스토어에 명시적으로 저장된 코드 시스템에서만 작동합니다.

오류 처리

작업은 다음 오류 조건을 처리합니다.

- 400 잘못된 요청: 잘못된 $lookup 작업(부적합 요청 또는 필수 파라미터 누락)
- 404 찾을 수 없음: 코드 시스템을 찾을 수 없거나 지정된 코드 시스템에서 코드를 찾을 수 없음

경고

이 릴리스에서는 다음이 지원되지 않습니다.

- $lookup 외부 용어 서버를 호출하여 작업

- `$lookup` HealthLake에서 관리하지만 데이터 스토어에 명시적으로 저장되지 않은 CodeSystems에서의 작업

`$lookup` 작업 사양에 대한 자세한 내용은 FHIR R4 CodeSystem `$lookup` 설명서를 참조하세요.

`$member-add` HealthLake에 대한 작업

FHIR `$member-add` 작업은 그룹 리소스, 특히 멤버 속성 목록에 멤버(환자)를 추가합니다. 이 작업은 DaVinci 멤버 속성 구현 가이드의 일부이며 멤버 속성을 관리하기 위한 조정 프로세스를 지원합니다.

작업 엔드포인트

```
POST [base]/datastore/{datastoreId}/r4/Group/{groupId}/$member-add
Content-Type: application/json
```

Parameters

작업은 다음 파라미터 조합이 있는 FHIR 파라미터 리소스를 수락합니다.

파라미터 옵션

다음 파라미터 조합 중 하나를 사용할 수 있습니다.

옵션 1: 멤버 ID + 공급자 NPI

`memberId` + `providerNpi`

`memberId` + `providerNpi` + `attributionPeriod`

옵션 2: 환자 참조 + 공급자 참조

`patientReference` + `providerReference`

`patientReference` + `providerReference` + `attributionPeriod`

파라미터 세부 정보

memberId(선택 사항)

그룹에 추가할 멤버의 식별자입니다.

유형: 식별자

시스템: 환자 식별자 시스템

```
{
  "name": "memberId",
  "valueIdentifier": {
    "system": "http://example.org/patient-id",
    "value": "patient-new"
  }
}
```

providerNpi(선택 사항)

귀속된 공급자의 국가 공급자 식별자(NPI)입니다.

유형: 식별자

시스템: http://terminology.hl7.org/CodeSystem/NPI

```
{
  "name": "providerNpi",
  "valueIdentifier": {
    "system": "http://terminology.hl7.org/CodeSystem/NPI",
    "value": "1234567890"
  }
}
```

patientReference(선택 사항)

추가할 환자 리소스에 대한 직접 참조입니다.

유형: 참조

```
{
  "name": "patientReference",
  "valueReference": {
    "reference": "Patient/patient-123"
  }
}
```

providerReference(선택 사항)

공급자 리소스에 대한 직접 참조입니다.

유형: 참조

```
{
  "name": "providerReference",
  "valueReference": {
    "reference": "Practitioner/provider-456"
  }
}
```

attributionPeriod(선택 사항)

환자가 공급자에 귀속되는 기간입니다.

유형: 기간

```
{
  "name": "attributionPeriod",
  "valuePeriod": {
    "start": "2024-07-15",
    "end": "2025-07-14"
  }
}
```

요청 예제

멤버 ID 및 공급자 NPI 사용

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

환자 및 공급자 참조 사용

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

응답 형식

성공적인 추가 응답

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

오류 응답

잘못된 요청 구문

HTTP 상태: 400 잘못된 요청

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "invalid",
      "details": {
        "text": "Invalid parameter combination provided"
      }
    }
  ]
}
```

리소스를 찾을 수 없음

HTTP 상태: 404 찾을 수 없음

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

```
      "code": "not-found",
      "details": {
        "text": "Resource not found."
      }
    }
  ]
}
```

버전 충돌

HTTP 상태: 409 충돌

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "conflict",
      "details": {
        "text": "Resource version conflict detected"
      }
    }
  ]
}
```

잘못된 속성 상태

HTTP 상태: 422 처리 불가능한 엔터티

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

비즈니스 규칙

속성 상태 검증

그룹의 속성 상태가 다음과 같은 경우에만 작업을 수행할 수 있습니다.

- `draft`
- `open`

상태가 인 경우 작업이 허용되지 않습니다`final`.

중복 멤버 방지

시스템은 다음과 같은 고유한 조합을 기반으로 중복 멤버를 추가하는 것을 방지합니다.

- 멤버 식별자
- 지급인 식별자
- 적용 범위 식별자

적용 기간 검증

`attributionPeriod`가 제공되면 멤버의 적용 범위 기간 범위 내에 속해야 합니다. 시스템은 다음을 수행합니다.

- 멤버의 적용 범위 리소스 검색
- 여러 개가 있는 경우 최신 적용 범위(가장 높은 versionId) 사용
- 어트리뷰션 기간이 적용 범위를 초과하지 않는지 확인

참조 검증

동일한 리소스(환자 또는 공급자)에 대해 ID와 참조가 모두 제공되면 시스템은 동일한 리소스에 해당하는지 확인합니다.

동일한 리소스(환자 또는 공급자)에 대해 ID 및 reference.identifier 필드가 모두 제공되면 오류가 발생합니다.

인증 및 권한 부여

작업을 수행하려면 다음에 대한 FHIR 인증에서 SMART가 필요합니다.

- 읽기 권한 - 환자, 공급자 및 그룹 리소스를 검증하려면
- 검색 권한 - 식별자별로 리소스를 찾으려면
- 권한 업데이트 - 그룹 리소스를 수정하려면

운영 동작

리소스 업데이트

- 그룹 리소스 버전 ID를 업데이트합니다.
- 작업 전에 원래 리소스 상태로 기록 항목을 생성합니다.
- 다음을 사용하여 Group.member 배열에 멤버 정보를 추가합니다.
  - entity.reference의 환자 참조
  - 기간의 속성 기간
  - 확장 필드의 적용 범위 및 공급자 정보

검증 단계

- 파라미터 검증 - 유효한 파라미터 조합 확인
- 리소스 존재 - 환자, 공급자 및 그룹 리소스가 존재하는지 검증합니다.
- 속성 상태 - 그룹 상태가 수정을 허용하는지 확인합니다.
- 중복 검사 - 기존 멤버 추가 방지
- 적용 범위 검증 - 어트리뷰션 기간이 적용 범위 범위 범위 내에 있는지 확인합니다.

제한 사항

- 참조된 모든 리소스는 동일한 데이터 스토어 내에 있어야 합니다.
- 작업은 멤버 속성 목록 그룹 리소스에서만 작동합니다.
- 속성 기간은 적용 범위 범위 내에 있어야 합니다.
- "최종" 상태의 그룹은 수정할 수 없습니다.

$member-match HealthLake에 대한 작업

AWS HealthLake 는 이제 환자 리소스에 대한 `$member-match` 작업을 지원하므로 의료 기관은 인구 통계 및 적용 범위 정보를 사용하여 다양한 의료 시스템에서 멤버의 고유 식별자를 찾을 수 있습니다. 이 작업은 환자의 개인 정보를 유지하면서 CMS 규정 준수를 달성하고 안전한 payer-to-payer 데이터 교환을 용이하게 하는 데 필수적입니다.

이 작업은 다음이 필요한 경우에 특히 유용합니다.

- 조직 간의 안전한 의료 데이터 교환 활성화
- 다양한 시스템에서 환자의 치료 연속성 유지

- CMS 규정 준수 요구 사항 지원
- 의료 네트워크에서 정확한 구성원 식별 촉진

사용법

POST 메서드를 사용하여 환자 리소스에서 $member-match 작업을 호출할 수 있습니다.

```
POST [base]/Patient/$member-match
```

지원되는 파라미터

HealthLake는 다음 FHIR $member-match 파라미터를 지원합니다.

파라미터	Type	필수	기본값	설명
MemberPatient	환자	예	—	일치시킬 구성원의 인구통계 정보가 포함된 환자 리소스
CoverageToMatch	적용 범위	예	—	기존 레코드와 일치시키는 데 사용할 적용 범위 리소스
CoverageToLink	적용 범위	아니요	—	매칭 프로세스 중에 연결할 적용 범위 리소스
동의	동의	아니요	—	권한 부여를 위한 동의 리소스

예제

파라미터를 사용한 POST 요청

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

샘플 응답

작업은 일치하는 결과가 포함된 파라미터 리소스를 반환합니다.

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

응답 파라미터

응답에는 일치 항목이 발견되면 다음 파라미터가 포함됩니다.

파라미터	유형	설명
MemberIdentifier	식별자	일치하는 멤버의 고유 식별자입니다.
MemberId	레퍼런스	환자 리소스에 대한 참조
matchAlgorithm	문자열	사용된 일치 알고리즘 유형(EXACT_MATCH, STRONG_MATCH 또는 DEMOGRAPH IC_MATCH)
matchDetails	문자열	매칭 프로세스에 대한 세부 정보

파라미터	유형	설명
matchedFields	문자열	성공적으로 일치하는 특정 필드 목록

일치 알고리즘

$member-match API는 다계층 매칭 접근 방식을 사용하여 정확한 멤버 식별을 보장합니다.

EXACT_MATCH

Coverage SubscriberId와 결합된 환자 식별자 사용

멤버 일치에 대한 가장 높은 신뢰도 수준을 제공합니다.

STRONG_MATCH

최소 적용 범위 정보와 함께 환자 식별자를 사용합니다.

정확한 일치 기준이 충족되지 않을 때 높은 신뢰도를 제공합니다.

데모그래픽_일치

기본 인구 통계 정보에 의존합니다.

식별자 기반 일치가 불가능한 경우에 사용됩니다.

동작

$member-match 작업:

- 환자 인구 통계, 적용 범위 세부 정보 및 선택적 동의 정보를 입력으로 수락합니다.
- 후속 상호 작용에 사용할 수 있는 고유한 멤버 식별자를 반환합니다.
- 여러 의료 시스템에서 정확한 구성원 식별을 보장하기 위해 다계층 일치(정확, 강력, 인구 통계)를 구현합니다.
- 향후 권한 부여를 위해 제공된 동의 정보를 저장합니다.
- 환자 개인 정보를 유지하면서 안전한 payer-to-payer 데이터 교환 지원
- 의료 데이터 교환에 대한 CMS 요구 사항 준수

권한 부여

API는 다음과 같은 필수 범위와 함께 FHIR 인증 프로토콜에서 SMART를 사용합니다.

- system/Patient.read
- system/Coverage.read
- system/Organization.read (조건부)
- system/Practitioner.read (조건부)
- system/PractitionerRole.read (조건부)
- system/Consent.write (조건부)

오류 처리

작업은 다음 오류 조건을 처리합니다.

- 400 Bad Request: 잘못된 $member-match 작업(부적합 요청 또는 필수 파라미터 누락)
- 422 Unprocessable Entity: 일치하는 항목 또는 여러 항목을 찾을 수 없음

$member-remove HealthLake에 대한 작업

$member-remove 작업을 통해의 FHIR 멤버 속성 목록(그룹 리소스)에서 멤버를 제거할 수 있습니다 AWS HealthLake. 이 작업은 DaVinci 멤버 속성 구현 가이드의 일부이며 멤버 속성을 관리하기 위한 조정 프로세스를 지원합니다.

사전 조건

- AWS HealthLake FHIR 데이터 스토어
- HealthLake 작업에 대한 적절한 IAM 권한
- 초안 또는 미결 상태의 멤버 속성 목록(그룹 리소스)

작업 세부 정보

엔드포인트

```
POST /Group/{id}/$member-remove
```

콘텐츠 유형

```
application/fhir+json
```

Parameters

작업은 다음과 같은 선택적 파라미터가 있는 FHIR 파라미터 리소스를 수락합니다.

파라미터	카디널리티	Type	설명
memberId	0..1	식별자	제거할 멤버의 비즈니스 식별자
providerNpi	0..1	식별자	귀속된 공급자의 NPI
patientReference	0..1	레퍼런스	환자 리소스에 대한 직접 참조
providerReference	0..1	레퍼런스	공급자 리소스(Practitioner, PractitionerRole 또는 Organization)에 대한 직접 참조
coverageReference	0..1	레퍼런스	적용 범위 리소스에 대한 참조

지원되는 파라미터 조합

지원되는 파라미터 조합은 다음과 같습니다.

- `memberId` 전용 - 지정된 멤버에 대한 모든 어트리뷰션을 제거합니다.
- `memberId` + `providerNpi` - 특정 멤버-공급자 조합에 대한 어트리뷰션을 제거합니다.
- `patientReference` 전용 - 지정된 환자의 모든 속성을 제거합니다.
- `patientReference` + `providerReference` - 특정 환자-제공자 조합에 대한 어트리뷰션을 제거합니다.
- `patientReference` + `providerReference` + `coverageReference` - 환자, 공급자 및 적용 범위에 따라 특정 속성을 제거합니다.

요청 예제

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "patientReference",
```

```
      "valueReference": {
        "reference": "Patient/12345"
      }
    },
    {
      "name": "providerReference",
      "valueReference": {
        "reference": "Practitioner/67890"
      }
    }
  ]
}
```

응답

응답 성공

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

동작

상태 요구 사항

작업은 draft 또는 상태의 어트리뷰션 목록에서만 작동합니다. open

final 상태가 인 목록은 422 오류와 함께 작업을 거부합니다.

멤버 제거 프로세스

초안 상태 목록: 멤버가 비활성(inactive: true)으로 표시되고 changeType 확장이 로 업데이트됩니다. changed

미결 상태 목록: 초안 상태와 유사한 동작

최종 상태 목록: 작업이 거부됨

검증

참조가 HealthLake 데이터 스토어에 존재하는지 검증됩니다.

동일한 리소스 유형에 대해 식별자와 참조가 모두 제공되는 경우 동일한 리소스에 해당해야 합니다.

파라미터 조합은 지원되는 패턴에 따라 검증됩니다.

오류 처리

일반적인 오류 응답

리소스를 찾을 수 없음(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"]
    }
  ]
}
```

속성 목록 최종 상태(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"]
    }
  ]
}
```

잘못된 작업(400)

파라미터 조합이 유효하지 않거나 잘못된 형식일 때 반환됩니다.

여러 일치 항목 찾음(412)

제공된 파라미터가 어트리뷰션 목록의 여러 멤버와 일치할 때 반환됩니다.

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

```
      "diagnostics": "Multiple members found matching the criteria"
    }
  ]
}
```

모범 사례

- 특정 파라미터 사용: 가능하면 가장 구체적인 파라미터 조합을 사용하여 의도하지 않은 제거를 방지합니다.
- 확인 목록 상태: 제거를 시도하기 전에 어트리뷰션 목록 상태 확인
- 오류 정상 처리: 가능한 모든 오류 조건에 적절한 오류 처리 구현
- 참조 검증: 요청하기 전에 참조된 모든 리소스가 존재하는지 확인합니다.

를 사용하여 환자 구획 리소스 제거 **$purge**

AWS HealthLake 는 환자 구획 내의 모든 리소스를 영구적으로 삭제할 수 있도록 $purge 작업을 지원합니다. 이 작업은 다음이 필요한 경우에 특히 유용합니다.

- 환자와 연결된 모든 데이터 제거
- 환자 데이터 제거 요청 준수
- 환자 데이터 수명 주기 관리
- 포괄적인 환자 레코드 정리 실행

사용법

$purge 작업은 환자 리소스에서 호출할 수 있습니다.

```
POST [base]/Patient/[ID]/$purge?deleteAuditEvent=true
```

Parameters

파라미터	Type	필수	기본값	설명
deleteAuditEvent	부울	아니요	false	true인 경우 연결된 감사 이벤트를 삭제합니다.

파라미터	Type	필수	기본값	설명
_since	문자열	No	데이터 스토어 생성 시간	입력하면는 lastModified 시간을 기반으로 리소스를 찾을 시작 마감 시간을 선택합니다. 시작 또는 종료와 함께 사용할 수 없음
start	문자열	No	데이터 스토어 생성 시간	입력 시는 마감 시간을 선택하여 lastModified 시간을 기준으로 리소스를 찾습니다. 끝과 함께 사용할 수 있습니다.
end	문자열	No	작업 제출 시간	입력 시 종료 마감 시간을 선택하여 lastModified 시간을 기반으로 리소스를 찾습니다.

예제

요청 예시

```
POST [base]/Patient/example-patient/$purge?deleteAuditEvent=true
```

응답의 예

```
{
  "resourceType": "OperationOutcome",
  "id": "purge-job",
  "issue": [
    {
      "severity": "information",
      "code": "informational",
      "diagnostics": "Purge job started successfully. Job ID:
 12345678-1234-1234-1234-123456789012"
    }
  ]
}
```

작업 상태

제거 작업의 상태를 확인하려면:

```
GET [base]/$purge/[jobId]
```

작업은 작업 상태 정보를 반환합니다.

```
{
      "datastoreId": "36622996b1fcecb7e12ee2ee085308d3",
      "jobId": "3dd1c7a5b6c0ef8c110f566eb87e2ef9",
      "status": "COMPLETED",
      "submittedTime": "2025-10-31T18:43:21.822Z"
    }
```

동작

$purge 작업:

1. 여러 리소스를 처리하기 위해 비동기적으로 처리
2. 데이터 무결성을 위해 ACID 트랜잭션 유지
3. 리소스 삭제 수와 함께 작업 상태 추적을 제공합니다.
4. 환자 구획의 모든 리소스를 영구적으로 제거합니다.
5. 삭제 활동에 대한 포괄적인 감사 로깅 포함
6. 감사 이벤트의 선택적 삭제 지원

감사 로깅

작업은 자세한 $purge 작업 정보와 함께 StartFHIRBulkDeleteJob 및 DescribeFHIRBulkDeleteJob으로 로깅합니다.

제한 사항

- 제거된 리소스는 검색 응답에 표시되지 않습니다.
- 제거 중인 리소스는 처리 중에 일시적으로 액세스하지 못할 수 있습니다.
- 환자 구획의 모든 리소스가 영구적으로 제거됩니다.

HealthLake에 대한 FHIR **$questionnaire-package** 작업

$questionnaire-package 작업은 FHIR 설문과 설문을 렌더링하고 처리하는 데 필요한 모든 종속성이 포함된 포괄적인 번들을 검색합니다. 이 작업은 Da Vinci 설명서 템플릿 및 규칙(DTR) 구현 가이드를 구현하여 의료 워크플로의 설명서 요구 사항에 맞게 동적 양식 렌더링을 활성화합니다.

작동 방식

- 요청: 적용 범위 및 주문 컨텍스트와 함께 필요한 설문지(들)를 식별하는 파라미터를 전송합니다.
- 검색: HealthLake는 설문지와 모든 종속성(ValueSets, CQL 라이브러리 등)을 수집합니다.
- 패키지: 모든 리소스가 표준화된 형식으로 함께 번들링됩니다.
- 응답: 렌더링 및 데이터 수집을 위한 전체 패키지를 받습니다.

사용 사례

- 사전 승인 설명서: 사전 승인 요청에 필요한 임상 정보 수집
- 적용 범위 요구 사항: 지급인 적용 범위 요구 사항을 충족하는 데 필요한 문서 수집
- 임상 데이터 교환: 지급자에게 제출할 임상 데이터 구조화
- 동적 양식: 환자 데이터 및 조건부 로직이 미리 채워진 설문지 렌더링

API 엔드포인트

```
POST /datastore/{datastoreId}/r4/Questionnaire/$questionnaire-package
Content-Type: application/fhir+json
```

요청 파라미터

입력 파라미터

요청 본문에는 다음 파라미터와 함께 FHIR 파라미터 리소스가 포함되어야 합니다.

파라미터	Type	카디널리티	설명
`coverage`	적용 범위	1..* (필수)	문서에 대한 멤버 및 적용 범위를 설정하기 위한 적용 범위 리소스(들)
`questionnaire`	canonical	0..*	반환할 특정 설문 조사(들)의 정식 URL(들)(버전을 포함할 수 있음)
`order`	Resource	0..*	리소스(DeviceRequest, ServiceRequest, MedicationRequest, Encounter, Appointment)를 주문하여 컨텍스트 설정

파라미터	Type	카디널리티	설명
changedSince	dateTime	0..1	있는 경우이 타임스탬프 이후에 변경된 리소스만 반환합니다.

파라미터 검증 규칙

다음 중 하나 이상을 제공해야 합니다(필수 외에도 coverage).

- 하나 이상의 questionnaire 정식 URLs
- 하나 이상의 order 리소스

유효한 요청 조합:

- coverage + questionnaire
- coverage + order
- coverage + questionnaire + order

요청 예제

```
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"
    }
  ]
}
```

응답 형식

성공 응답(200 OK)

작업은 하나 이상의 패키지 번들이 포함된 FHIR 파라미터 리소스를 반환합니다. 각 패키지 번들에는 다음이 포함됩니다.

입력 유형	카디널리티	설명
설문 조사	1	렌더링할 설문지
QuestionnaireResponse	0..1	미리 채워지거나 부분적으로 완료된 응답(해당하는 경우)
라이브러리	0..*	사전 모집단 및 조건부 로직이 포함된 CQL 라이브러리
ValueSet	0..*	확장된 ValueSets(확장이 <40인 응답 선택의 경우)

Example응답의 예

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

작업 워크플로

HealthLake가 요청을 처리하는 방법

를 호출하면 $questionnaire-package HealthLake는 다음 단계를 수행합니다.

1. 환자 및 지급인 식별: coverage 파라미터에서 환자 및 보험 조직을 추출합니다.
2. 올바른 설문 조사 찾기:
   - 파라미터 사용questionnaire: 제공한 정식 URL을 사용합니다.
   - 파라미터 사용order: 주문 코드(CPT/HCPCS/LOINC) 및 지급인과 일치하여 적절한 설문지를 찾습니다.
3. 종속성 수집: 설문지를 렌더링하는 데 필요한 모든 것을 자동으로 검색합니다.
   - CQL 라이브러리 - 사전 모집단 및 조건부 질문을 위한 로직
   - ValueSets - 응답 선택 사항(<40 옵션인 경우 자동으로 확장됨)
   - QuestionnaireResponse - 진행 중이거나 완료된 기존 응답
4. 모든 것을 함께 패키징합니다.
   - 모든 리소스를 번들링합니다(각 리소스는 한 번만 포함됨).
   - 제공된 경우 changedSince 타임스탬프를 기준으로 필터링
   - 리소스가 누락된 Outcome 경우에 경고를 추가합니다.

결과: 렌더링 준비가 완료된 독립형 패키지입니다.

오류 응답

400 잘못된 요청

요청 검증에 실패하면 반환됩니다.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "required",
    "details": {
      "text": "At least one of 'questionnaire' or 'order' must be provided along with
 'coverage'"
    }
  }]
}
```

424 종속성 실패

종속 리소스를 검색할 수 없을 때 반환됩니다.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "warning",
    "code": "not-found",
    "details": {
      "text": "Referenced Library 'http://example.org/fhir/Library/missing-library'
 could not be retrieved"
    }
  }]
}
```

401 권한이 없음

인증 자격 증명이 누락되거나 유효하지 않을 때 반환됩니다.

403 금지됨

인증된 사용자에게 요청된 리소스에 액세스할 수 있는 권한이 없는 경우 반환됩니다.

406 허용되지 않음

요청된 콘텐츠 유형을 제공할 수 없을 때 반환됩니다.

409 충돌

버전 또는 동시성 충돌이 있을 때 반환됩니다.

410 사라짐

요청된 리소스가 영구적으로 삭제되면 반환됩니다.

429 요청이 너무 많음

속도 제한을 초과하면 반환됩니다.

500 Internal Server Error

예기치 않은 서버 오류가 발생할 때 반환됩니다.

501 구현되지 않음

요청된 작업이 아직 구현되지 않은 경우 반환됩니다.

검증 규칙

입력 검증

- `coverage` 파라미터 필요(1..* 카디널리티)
- `questionnaire` 또는 중 하나 이상을 제공해야 `order` 합니다.
- 모든 적용 범위 리소스는 유효한 FHIR 리소스여야 합니다.
- 모든 주문 리소스는 유효한 FHIR 리소스여야 합니다.
- 정식 URLs 올바른 형식이어야 합니다.
- `changedSince`는 유효한 ISO 8601 dateTime이어야 합니다.

QuestionnaireResponse 검증

- `status`가 적절해야 함(`in-progress`, `completed`, `amended`)
- 구조는 참조된 설문 조사와 일치해야 합니다.
- `basedOn`는 유효한 주문 리소스를 참조해야 합니다.
- `subject`는 유효한 환자 리소스를 참조해야 합니다.

리소스 중복 제거

- 각 리소스는 번들에 한 번만 표시됩니다.
- 예외: 동일한 리소스의 다른 버전이 모두 포함될 수 있습니다.
- 리소스는 표준 URL 및 버전으로 식별됩니다.

성능 사양

지표	사양
리소스 수 제한	번들당 리소스 500개
번들 크기 제한	최대 5MB

필수 권한

`$questionnaire-package` 작업을 사용하려면 IAM 역할에 다음이 있는지 확인합니다.

- `healthlake:QuestionnairePackage` - 작업을 호출하려면
- `healthlake:ReadResource` - 설문 조사 및 종속 리소스를 검색하려면
- `healthlake:SearchWithPost` - QuestionnaireResponse 및 관련 리소스를 검색하려면

FHIR 범위에 대한 SMART

최소 필수 범위:

- 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`

중요한 구현 참고 사항

리소스 검색 전략

설문 조사 식별 우선 순위:

- 표준 URL(`questionnaire`파라미터가 제공된 경우) - 최고 우선 순위
- 주문 분석(`order`파라미터가 제공된 경우):
  - 주문 코드(CPT, HCPCS, LOINC)를 지급인 의료 정책과 일치시킵니다.
  - 담당률 지급인을 사용하여 지급인별 설문지 필터링
  - 추가 컨텍스트에 대한 사유 코드 고려

종속성 해결

CQL 라이브러리:

- 설문지 리소스의 `cqf-library` 확장을 통해 검색됨
- 유형을 `Library.relatedArtifact` 사용하여를 통해 종속 라이브러리를 재귀적으로 가져옵니다. `depends-on`
- 모든 라이브러리 종속성이 패키지에 포함됨

ValueSets:

- 개념이 40개 미만인 경우 자동으로 확장됨
- 더 큰 ValueSets 확장 없이 포함됩니다.
- 설문 조사 및 라이브러리 리소스 모두에서 참조되는 ValueSets가 포함됩니다.

QuestionnaireResponse 사전 모집단

작업은 다음과 같은 경우 미리 채워진 데이터가 포함된 QuestionnaireResponse를 반환할 수 있습니다.

- 진행 중이거나 완료된 기존 응답이 발견됨
- 연결된 라이브러리의 CQL 로직은 환자 레코드에서 데이터를 추출할 수 있습니다.
- 응답은 관련 순서 및 적용 범위와 연결됩니다.

QuestionnaireResponse 검색 기준:

검색 파라미터	FHIR 경로	설명
based-on	QuestionnaireResponse.basedOn	ServiceRequest 또는 CarePlan에 대한 링크
patient	QuestionnaireResponse.subject	주제인 환자
questionnaire	QuestionnaireResponse.questionnaire	답변 중인 설문

리소스 필터링 변경

changedSince 파라미터가 제공되는 경우:

- 지정된 타임스탬프 이후에 수정된 리소스만 포함됩니다.
- 리소스가 변경되지 않은 경우는 빈 패키지200 OK와 함께를 반환합니다.
- 증분 업데이트 및 캐싱 전략에 유용합니다.
- 타임스탬프 비교는 리소스 meta.lastUpdated 필드를 사용합니다.

여러 패키지 번들

작업은 다음과 같은 경우 여러 패키지 번들을 반환할 수 있습니다.

- 정식 URLs 통해 여러 설문이 요청됩니다.
- 여러 주문에 서로 다른 설문지 필요
- 동일한 설문 조사의 다른 버전이 적용됩니다.

각 패키지 번들에는 필요한 모든 종속성이 포함되어 있습니다.

일반 사용 사례

사용 사례 1: 사전 승인 설명서

시나리오: 공급자는 사전 승인을 받기 전에 가정용 항기 치료용 설명서를 수집해야 합니다.

```
{
```

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

결과: 환자 바이탈과 EHR의 진단 코드로 미리 채워진, 항산기 치료 설문지가 포함된 패키지를 반환합니다.

사용 사례 2: 특정 설문 조사 버전 검색

시나리오: 규정 준수를 위해 공급자에게 특정 버전의 설문지가 필요합니다.

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "coverage",
      "resource": { /* Coverage resource */ }
    },
    {
      "name": "questionnaire",
      "valueCanonical": "http://example.org/fhir/Questionnaire/dme-request|3.1.0"
    }
  ]
}
```

결과: 모든 종속성과 함께 정확히 버전 3.1.0의 Dell 요청 설문지를 반환합니다.

사용 사례 3: 업데이트 확인

시나리오: 공급자는 마지막 검색 이후 업데이트된 설문지 리소스가 있는지 확인하려고 합니다.

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

결과: 2024년 3월 1일 이후에 수정된 리소스만 반환하거나 변경되지 않은 경우 빈 패키지를 반환합니다.

사용 사례 4: 여러 주문

시나리오: 공급자가 서로 다른 설문지가 필요할 수 있는 여러 서비스 요청을 제출합니다.

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "coverage",
      "resource": { /* Coverage resource */ }
    },
    {
      "name": "order",
      "resource": { /* ServiceRequest for imaging */ }
    },
    {
      "name": "order",
      "resource": { /* ServiceRequest for DME */ }
```

```
    }
  ]
}
```

결과: 해당하는 각 설문에 대해 하나씩 여러 패키지 번들을 반환합니다.

다른 Da Vinci IGs와 통합

적용 범위 요구 사항 검색(CRD)

워크플로 통합:

- 공급자가 EHR에서 서비스 주문
- CRD 후크 발사, 적용 범위 요구 사항 확인
- 지급인이 문서가 필요함을 나타내는 응답
- 공급자가를 호출`$questionnaire-package`하여 설명서 양식을 검색합니다.
- 공급자가 설문지를 작성합니다.
- 설명서는 PAS 또는 CDex를 통해 제출됩니다.

사전 승인 지원(PAS)

워크플로 통합:

- `$questionnaire-package`를 사용하여 설명서 요구 사항 검색
- 필수 임상 데이터로 QuestionnaireResponse 작성
- 완료된 QuestionnaireResponse와 `Claim/$submit` 함께를 사용하여 사전 권한 부여 제출
- 를 사용하여 상태 확인 `Claim/$inquire`

임상 데이터 교환(CDex)

워크플로 통합:

- 지급인이 클레임에 대한 추가 문서를 요청합니다.
- 공급자는 `$questionnaire-package`를 사용하여 구조화된 데이터 수집 양식을 검색합니다.

- 공급자가 QuestionnaireResponse를 완료합니다.
- CDex 연결 워크플로를 통해 지급자에게 문서가 제출됩니다.

문제 해결 가이드

문제: 반환된 설문지 없음

가능한 원인:

- 정식 URL이 데이터 스토어의 설문과 일치하지 않습니다.
- 주문 코드가 지급인의 의료 정책에 있는 설문에 매핑되지 않음
- 담당 지급인에 연결된 설문지가 없음

솔루션

- 표준 URL이 올바르고 설문 조사가 존재하는지 확인합니다.
- 주문 코드(CPT/HCPCS)가 올바르게 지정되었는지 확인
- 지급인 조직에 구성된 설문지가 있는지 확인

문제: 패키지의 종속성 누락

가능한 원인:

- 참조된 라이브러리 또는 ValueSet가 데이터 스토어에 존재하지 않음
- 라이브러리 참조가 손상되었거나 잘못되었습니다.
- ValueSet 확장 실패

솔루션

- 누락된 리소스에 대한 경고가 있는지 `Outcome` 파라미터 확인
- 참조된 모든 리소스가 데이터 스토어에 있는지 확인
- ValueSet URLs 올바르고 확인 가능한지 확인

문제: changedSince가 있는 빈 패키지

가능한 원인:

- 이는 예상되는 동작입니다. 지정된 타임스탬프 이후 리소스가 수정되지 않았습니다.

솔루션

- 패키지의 캐시된 버전 사용
- changedSince 파라미터를 제거하여 전체 패키지 검색

문제: QuestionnaireResponse가 미리 채워지지 않음

가능한 원인:

- 기존 QuestionnaireResponse를 찾을 수 없음
- CQL 라이브러리 로직이 필요한 데이터를 추출할 수 없음
- 환자 데이터가 누락되었거나 불완전함

솔루션

- 이는 예상할 수 있습니다. 모든 설문지에 사전 모집단 로직이 있는 것은 아닙니다.
- 환자 데이터가 데이터 스토어에 존재하는지 확인
- 데이터 추출 요구 사항에 대한 CQL 라이브러리 로직 검토

모범 사례

1. 버전URLs 사용

특정 설문지를 요청할 때는 항상 버전 번호를 지정하십시오.

```
{
  "name": "questionnaire",
  "valueCanonical": "http://example.org/fhir/Questionnaire/dme|2.1.0"
}
```

이유: 일관성을 보장하고 설문이 업데이트될 때 예기치 않은 변경을 방지합니다.

2. 성능 측면에서 changedSince 활용

자주 액세스하는 설문지의 경우 changedSince를 사용하여 데이터 전송을 최소화합니다.

```
{
  "name": "changedSince",
  "valueDateTime": "2024-03-10T15:30:00Z"
}
```

이유: 업데이트된 리소스만 검색하여 지연 시간과 대역폭 사용량을 줄입니다.

3. 전체 적용 범위 정보 포함

올바른 설문 선택을 위해 포괄적인 적용 범위 세부 정보를 제공합니다.

```
{
  "name": "coverage",
  "resource": {
    "resourceType": "Coverage",
    "beneficiary": { "reference": "Patient/123" },
    "payor": [{ "reference": "Organization/payer-abc" }],
    "class": [{ /* Group/plan information */ }]
  }
}
```

이유: HealthLake가 지급인별 설문지 및 요구 사항을 식별하는 데 도움이 됩니다.

관련 작업

- `Claim/$submit` - 완료된 설명서와 함께 사전 권한 부여 요청 제출
- `Claim/$inquire` - 제출된 사전 권한 부여의 상태 확인
- `ValueSet/$expand` - 답변 선택에 대한 ValueSets 확장

HealthLake에 대한 FHIR **`$submit`** 작업

`$submit` 작업을 통해 승인을 위해 지급자에게 사전 권한 부여 요청을 전자적으로 제출할 수 있습니다. 이 작업은 Da Vinci 사전 승인 지원(PAS) 구현 가이드를 구현하여 사전 승인 제출을 위한 표준화된 FHIR 기반 워크플로를 제공합니다.

작동 방식

- 제출: 사전 승인 요청과 지원 임상 데이터가 포함된 FHIR 번들을 보냅니다.
- 검증: HealthLake는 PAS 요구 사항을 기준으로 제출을 검증합니다.
- 지속성: 모든 리소스가 HealthLake 데이터 스토어에 저장됩니다.
- 응답: "대기열" 상태의 즉각적인 응답을 받습니다.
- 프로세스: 권한 부여 결정은 지급인에 의해 비동기적으로 처리됩니다.

API 엔드포인트

```
POST /datastore/{datastoreId}/r4/Claim/$submit
Content-Type: application/fhir+json
```

요청 구조

번들 요구 사항

요청은 다음을 포함하는 FHIR 번들 리소스여야 합니다.

- Bundle.type: 이어야 함 `"collection"`
- Bundle.entry:를 사용하여 정확히 하나의 클레임 리소스를 포함해야 합니다. `use = "preauthorization"`
- 참조 리소스: 클레임에서 참조하는 모든 리소스가 번들에 포함되어야 합니다.

필수 리소스

Resource	카디널리티	프로필	설명
클레임	1	PAS 클레임	이전 권한 부여 요청
환자	1	PAS 환자	환자 인구통계 정보
조직(보험사)	1	PAS 보험 회사	보험 회사
조직(제공자)	1	PAS 요청자	요청을 제출하는 의료 서비스 제공자
적용 범위	1개 이상	PAS 적용 범위	보험 적용 범위 세부 정보

선택적 리소스

Resource	카디널리티	프로필	설명
프랙티셔너	0개 이상	PAS 프랙티셔너	의료 종사자
PractitionerRole	0개 이상	PAS PractitionerRole	프랙티셔너 역할
ServiceRequest	0개 이상	PAS ServiceRequest	요청된 의료 서비스
DeviceRequest	0개 이상	PAS DeviceRequest	요청된 의료 디바이스
MedicationRequest	0개 이상	PAS MedicationRequest	요청된 약물
DocumentReference	0개 이상	PAS DocumentReference	임상 문서 지원

요청 예제

```
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"
    }
  }]
```

```
}
```

응답 형식

성공 응답(200 OK)

다음이 포함된 PAS 응답 번들을 받게 됩니다.

- `outcome: "queued"` 및를 사용한 ClaimResponse `status: "active"`
- 요청의 모든 원본 리소스
- 수신 확인 타임스탬프

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

오류 응답

400 잘못된 요청

요청 형식이 유효하지 않거나 잘못된 형식일 때 반환됩니다.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "invalid",
    "diagnostics": "The provided payload was invalid and could not be parsed
 correctly."
  }]
}
```

412 사전 조건 실패

동일한 사전 권한 부여 요청이 이미 제출되었을 때 반환됩니다(중복 제출이 감지됨).

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "processing",
    "diagnostics": "PreAuth Claim already exists"
  }]
}
```

> **멱등성**
>
> `$submit` 작업은 멱등성입니다. 동일한 요청을 여러 번 제출해도 사전 권한 부여 요청이 중복되지 않습니다. 대신를 사용하여 원래 제출의 상태를 `$inquire` 확인하라는 412 오류가 표시됩니다.

422 처리할 수 없는 개체

FHIR 검증에 실패할 때 반환됩니다.

```
{
```

```
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "required",
    "diagnostics": "Bundle contains more than one preauthorization claim"
  }]
}
```

검증 규칙

HealthLake는 제출에 대해 포괄적인 검증을 수행합니다.

번들 검증

- PAS 요청 번들 프로필을 준수해야 합니다.
- `Bundle.type` 여야 합니다. `"collection"`
- 여러 클레임 리소스를 포함할 수 있음
- 그러나는 사전 권한 부여를 사용하는 정확히 하나의 클레임 리소스를 포함해야 합니다.
  - 그리고이 클레임 리소스는 번들의 첫 번째 항목이어야 합니다.
- 참조된 모든 리소스가 번들에 포함되어야 합니다.

클레임 검증

- PAS 클레임 프로파일을 준수해야 합니다.
- `Claim.use` 여야 합니다. `"preauthorization"`
- 필수 필드: `patient`, `insurer`, `provider`, `created`, `priority`
- 비즈니스 식별자가 존재하고 유효해야 합니다.

리소스 검증

- 모든 리소스는 해당 PAS 프로파일을 준수해야 합니다.
- 필요한 지원 리소스가 있어야 합니다(환자, 담당 범위, 조직).
- 교차 참조는 번들 내에서 유효하고 해석 가능해야 합니다.

성능 사양

지표	사양
번들 크기 제한	최대 5MB
리소스 수 제한	번들당 리소스 500개

필수 권한

`$submit` 작업을 사용하려면 FHIR에서 AWS Sigv4 또는 SMART를 사용할 수 있습니다.

- IAM 역할에 다음이 있는지 확인합니다. `healthlake:SubmitPreAuthClaim` - 작업을 호출하려면

FHIR 범위에 대한 SMART

최소 필수 범위:

- SMART v1: `user/Claim.write & <all_resourceTypes_in_Bundle>.write`
- SMART v2: `user/Claim.c & <all_resourceTypes_in_Bundle>.c or system/*.*`

중요한 구현 참고 사항

리소스 지속성

- 모든 번들 항목은 데이터 스토어에서 개별 FHIR 리소스로 유지됩니다.
- 고객 제공 IDs 제공 시 보존됩니다.
- 버전 기록은 감사 목적으로 유지 관리됩니다.
- 중복 탐지는 리소스 충돌을 방지합니다.

처리 동작

- 유효한 각 제출은 `"queued"` 결과가 있는 정확히 하나의 ClaimResponse를 생성합니다.
- 잘못된 제출은 자세한 오류 정보와 함께 400 또는 422 상태 코드를 반환합니다.
- 시스템 오류는 적절한 5xx 상태 코드를 반환합니다.

- 모든 성공적인 제출은 호출된 ClaimResponse와 함께 200 상태를 반환합니다.

번들 요구 사항

- `Bundle.entry.fullUrl` 값은 REST URLs 또는 `"urn:uuid:[guid]"` 형식이어야 합니다.
- 모든 GUIDs 제출 간에 고유해야 합니다(동일한 리소스 인스턴스 제외).
- 참조된 리소스는 번들 내에 존재하거나 해결 가능해야 합니다.

관련 작업

- `Claim/$inquire` - 제출된 사전 권한 부여 요청의 상태 쿼리
- `Patient/$everything` - 사전 권한 부여 컨텍스트에 대한 포괄적인 환자 데이터 검색

를 사용하여 FHIR 리소스 검증 **$validate**

AWS HealthLake 는 이제 FHIR 리소스에 대한 `$validate` 작업을 지원하므로 스토리지 작업을 수행하지 않고도 FHIR 사양에 따라 리소스를 검증하고 지정된 프로필 또는 기본 리소스 정의에 대한 적합성을 확인할 수 있습니다. 이 작업은 다음이 필요한 경우에 특히 유용합니다.

- FHIR CMS 규정 준수 요구 사항 검증
- 프로덕션에서 사용하기 전에 리소스 테스트
- 사용자가 임상 데이터를 편집할 때 실시간 검증 피드백 제공
- 잘못된 데이터 제출을 줄여 APIs 생성 및 업데이트

사용법

POST 메서드를 사용하여 FHIR 리소스에서 `$validate` 작업을 호출할 수 있습니다.

지원되는 작업

```
POST [base]/[type]/[id]/$validate
POST [base]/[type]/$validate
```

지원되는 페이로드

파라미터 리소스

HealthLake는 다음 FHIR `$validate` 파라미터를 지원합니다.

파라미터	Type	필수	설명
`resource`	Resource	예	검증할 리소스
`profile`	canonical	아니요	검증할 프로필의 정식 URL
`mode`	code	아니요	검증 모드: `create`또는 `update`

쿼리 파라미터가 있는 다이렉트 리소스

파라미터	Type	필수	설명
`profile`	canonical	아니요	검증할 프로필의 정식 URL
`mode`	code	아니요	검증 모드: `create`또는 `update`

예제

ID 및 파라미터 페이로드를 사용한 리소스에 대한 POST 요청

```
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"
    }
  ]
}
```

리소스 유형 및 파라미터 페이로드에 대한 POST 요청

```
POST [base]/Patient/$validate
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "resource",
      "resource": {
        "resourceType": "Patient",
        "name": [
          {
            "family": "Doe",
            "given": ["Jane"]
          }
        ],
        "gender": "female",
        "birthDate": "1985-05-15"
      }
    },
    {
      "name": "profile",
      "valueCanonical": "http://hl7.org/fhir/us/core/StructureDefinition/us-core-
patient"
```

```
    },
    {
      "name": "mode",
      "valueString": "update"
    }
  ]
}
```

ID 및 직접 리소스 페이로드가 있는 리소스에 대한 POST 요청

```
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"
}
```

리소스 유형 및 직접 리소스 페이로드에 대한 POST 요청

```
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"
}
```

샘플 응답

작업은 검증 결과와 함께 OperationOutcome 리소스를 반환합니다.

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "information",
      "code": "informational",
      "diagnostics": "Validation successful"
    }
  ]
}
```

검증 오류가 있는 샘플 응답

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

동작

`$validate` 작업:

1. FHIR 사양 및 기본 리소스 정의를 기준으로 리소스를 검증합니다.
2. `profile` 파라미터가 제공될 때 지정된 프로필에 대한 적합성을 확인합니다.
3. 지정된 모드(`create` 또는 `update`)를 기반으로 검증합니다.
4. 오류, 경고 및 정보 메시지를 포함한 자세한 검증 결과를 반환합니다.
5. 스토리지 작업을 수행하지 않음 - 검증만 수행
6. 검증 문제가 있는지 여부에 관계없이 검증을 수행할 수 있을 때 HTTP 200 OK를 반환합니다.

검증 모드

- create: 리소스를 생성 중인 것처럼 검증합니다(새 리소스).
- update: 리소스가 업데이트되는 것처럼 확인합니다(기존 리소스).

오류 처리

작업은 다음을 반환합니다.

- 200 OK: 검증이 성공적으로 수행되었습니다(검증 결과와 무관).
- 400 잘못된 요청: 잘못된 요청 형식 또는 파라미터
- 404 찾을 수 없음: 리소스 유형 또는 프로필을 찾을 수 없음

`$validate` 작업 사양에 대한 자세한 내용은 FHIR R4 리소스 `$validate` 설명서를 참조하세요.

에 대한 규정 준수 참조 AWS HealthLake

AWS HealthLake 는 CMS(Centers for Medicare & MasterCard Services) 상호 운용성 요구 사항에 따라 API 사용량을 추적하고 보고하는 데 도움이 되도록 설계된 기능을 제공합니다. 이러한 기능을 사용

하면 API 트랜잭션을 CMS 필수 범주별로 분류하고 규정 준수 보고를 위해 사용량 지표를 자동으로 캡처할 수 있습니다.

⚠ 규정 준수 책임 이해

AWS HealthLake 및 해당 CMS 상호 운용성 엔드포인트를 사용하는 것만으로는 CMS 규정 준수를 달성하기에 충분하지 않습니다. 다음과 같은 작업을 담당합니다.

- 특정 사용 사례 및 규제 의무에 따라 API 워크플로를 적절한 CMS 범주 엔드포인트에 올바르게 매핑
- CMS 요구 사항을 충족하는 적절한 인증 및 권한 부여 제어 구현
- FHIR 리소스 및 데이터 교환이 해당 CMS 규정 및 구현 가이드를 준수하는지 확인
- 규정 준수 보고 요구 사항을 지원하기 위해 CloudWatch 지표 구성 및 모니터링
- 조직에 적용되는 CMS 규칙 이해 및 적절한 기술 및 운영 제어 구현

AWS HealthLake 는 규정 준수 노력을 지원하는 인프라와 도구를 제공하지만 특정 규제 요구 사항에 따라 이러한 기능을 적절하게 사용해야 합니다. 이러한 엔드포인트를 통해 API 호출을 라우팅하는 것만으로는 애플리케이션이 CMS 규정을 자동으로 준수하지 않습니다.

주제

CMS 규정 준수 기능

AWS HealthLake 는 CMS(Centers for Medicare & MasterCard Services) 상호 운용성 및 규정 준수 요구 사항을 충족하는 데 도움이 되는 기능을 제공합니다. 이러한 기능을 사용하면 CMS 범주별로 API 사용량을 추적한 다음 규정 준수를 위해 사용량 지표를 보고할 수 있습니다.

주제

CMS 상호 운용성 엔드포인트

개요

HealthLake는 CMS 필수 API 범주에 해당하는 4개의 CMS 상호 운용성 엔드포인트를 제공합니다. HealthLake 데이터 스토어의 기본 기본 URL은 변경되지 않습니다. 이러한 엔드포인트는 CMS 보고 목적으로 API 호출을 분류하고 추적하는 방법을 제공합니다.

용도

이러한 상호 운용성 엔드포인트의 주요 목적은 고객이 다음을 수행할 수 있도록 하는 것입니다.

- CMS 범주별로 API 트랜잭션을 쉽게 추적
- CMS 규정 준수를 위한 사용량 지표 자동 보고
- 최소한의 변경으로 기존 FHIR 워크플로 유지 관리

상호 운용성 엔드포인트를 사용하든 표준 FHIR 엔드포인트를 사용하든 모든 API 호출은 동일하게 작동합니다. 유일한 차이점은 트랜잭션이 CloudWatch 지표로 분류되는 방식입니다.

지원되는 CMS 상호 운용성 엔드포인트

CMS 범주	상호 운용성 엔드포인트	사용 예
환자 액세스	`/patientaccess/v2/r4`	`baseURL/patientaccess/v2/r4/Patient/123`
공급자 액세스	`/provideraccess/v2/r4`	`baseURL/provideraccess/v2/r4/Observation?patient=123`
지급인에서 지급인으로	`/payertopayerdx/v2/r4`	`baseURL/payertopayerdx/v2/r4/Practitioner/456`
이전 인증 서비스	`/priorauthservice/v2/r4`	`baseURL/priorauthservice/v2/r4/ExplanationOfBenefit?patient=789`

상호 운용성 엔드포인트 작동 방식

표준 HealthLake API 호출:

```
baseURL/resourceType/[id]
baseURL/resourceType?[parameters]
```

CMS 상호 운용성 엔드포인트 사용:

```
baseURL/interoperability-endpoint/resourceType/[id]
baseURL/interoperability-endpoint/resourceType?[parameters]
```

상호 운용성 엔드포인트 경로는 기본 URL과 리소스 유형 사이에 간단히 삽입됩니다. 상호 운용성 엔드포인트 경로 이후의 모든 것은 현재 API 호출과 정확히 동일하게 유지됩니다.

사용 예제

예제 1: 환자 액세스 API

현재 API 직접 호출(아직 작동):

```
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"
```

환자 액세스 상호 운용성 엔드포인트(CMS 추적용):

```
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"
```

요점:

- 기본 URL은 다음과 같이 유지됩니다. `https://healthlake.us-east-1.amazonaws.com/datastore/abc123`
- 상호 운용성 엔드포인트 삽입: `/patientaccess/v2/r4`
- 리소스 경로 변경되지 않음: `/Patient/123`
- 두 호출 모두 동일한 응답을 반환합니다.
- 상호 운용성 엔드포인트 호출은 CloudWatchURIType=patient-access의에서 자동으로 추적됩니다.

- POST, PUT, PATCH, DELETE 작업은 동일하게 작동합니다.
- 요청 본문은 변경되지 않습니다.

엔드포인트 번역 참조

상호 운용성 엔드포인트	변환 대상	CMS 범주
`baseURL/patientaccess/v2/r4/Patient`	`baseURL/r4/Patient`	환자 액세스
`baseURL/provideraccess/v2/r4/Observation`	`baseURL/r4/Observation`	공급자 액세스
`baseURL/payertopayerdx/v2/r4/Practitioner/456`	`baseURL/r4/Practitioner/456`	지급인 간 데이터 교환
`baseURL/priorauthservice/v2/r4/ExplanationOfBenefit?patient=789`	`baseURL/r4/ExplanationOfBenefit?patient=789`	사전 승인

중요 정보

- 기능적 차이 없음: 상호 운용성 엔드포인트와 표준 FHIR 엔드포인트는 동일한 응답을 반환하고 동일한 작업을 지원합니다.
- 기본 URL 변경되지 않음: HealthLake 데이터 스토어 엔드포인트는 동일하게 유지됩니다.
- 단순 통합: 기본 URL과 리소스 유형 사이에 상호 운용성 엔드포인트 경로 삽입
- 자동 추적: CloudWatch 지표는 사용된 상호 운용성 엔드포인트별로 호출을 자동으로 분류합니다.
- 역호환성: 상호 운용성 엔드포인트가 없는 기존 API 호출은 정상적으로 계속 작동합니다.

CMS 규정 준수를 위한 향상된 CloudWatch 지표

개요

CMS 상호 운용성 엔드포인트를 사용하면 HealthLake는 CMS 보고 요구 사항을 지원하기 위해 추가 차원과 함께 향상된 CloudWatch 지표를 자동으로 내보냅니다. 이러한 지표는 추가 구성 없이 호출자 ID, 애플리케이션 및 CMS별 URI 유형별로 API 사용량을 추적합니다.

작동 방식

상호 운용성 엔드포인트를 사용하여 API를 호출하는 경우:

```
# 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.)
```

추가 코드나 구성은 필요하지 않습니다. 상호 운용성 엔드포인트를 사용하면 향상된 지표가 자동으로 캡처됩니다.

> **Note**
>
> FHIR 데이터 스토어의 비 SMART의 경우 URIType차원이 계속 캡처되므로 CMS 범주별로 API 사용량을 추적할 수 있습니다. Sub 및 ClientId차원은 이러한 클레임이 포함된 보유자 토큰과 함께 FHIR 인증에서 SMART를 사용할 때만 사용할 수 있습니다.

새로운 지표 차원

기존 차원(DatastoreId, DatastoreType, Operation) 외에도 상호 운용성 엔드포인트를 사용할 때 다음 차원이 자동으로 추가됩니다.

차원	설명	예제 값	소스
URIType	CMS 규정 준수 범주	patient-access , provider-access ,	상호 운용성 엔드포인트 경로에서 자동으로 결정됨

차원	설명	예제 값	소스
		`payer-to-payer` , `prior-authorization`	
하위	발신자 자격 증명	사용자/엔터티 식별자	보유자 토큰 `sub` 클레임에서 추출됨
ClientId	애플리케이션 식별자	`portal_app` , `ehr_system`	보유자 토큰 `client_id` 클레임에서 추출됨

사용 가능한 지표

이제 모든 기존 HealthLake 지표에 상호 운용성 엔드포인트를 사용할 때 추가 차원이 포함됩니다.

- CallCount - 총 API 호출 수
- 지연 시간 - 밀리초 단위의 API 응답 시간
- UserErrors - 4xx 클라이언트 오류 수
- SystemErrors - 5xx 서버 오류 수
- 제한 - 제한된 요청 수
- SuccessfulRequests - 성공한 API 호출 수

CloudWatch에서 지표 쿼리

CloudWatch Insights 쿼리 예제

애플리케이션별로 모든 환자 액세스 API 호출을 쿼리합니다.

```
SELECT SUM(CallCount)
FROM "AWS/HealthLake"
WHERE DatastoreId = '75c1cf9b0d71cd38fec8f7fb317c4c1a'
    AND URIType = 'patient-access'
GROUP BY ClientId
```

에 대한 지원 참조 AWS HealthLake

다음 지원 참조 자료를 사용할 수 있습니다 AWS HealthLake.

> **Note**
>
> 모든 네이티브 HealthLake 작업 및 데이터 형식은 별도의 참조에 설명되어 있습니다. 자세한 내용은 AWS HealthLake API 참조를 참조하세요.

주제

AWS HealthLake 엔드포인트 및 할당량

다음 주제에는 AWS HealthLake 서비스 엔드포인트 및 할당량에 대한 정보가 포함되어 있습니다.

주제

서비스 엔드포인트

서비스 엔드포인트는 호스트 및 포트를 웹 서비스의 진입점으로 식별하는 URL입니다. 모든 웹 서비스 요청에는 진입점이 포함되어 있습니다. 대부분의 AWS 서비스는 특정 리전에 대한 엔드포인트를 제공하여 더 빠른 연결을 지원합니다. 다음 표에는 서비스 엔드포인트가 나열되어 있습니다 AWS HealthLake.

리전 이름	리전	엔드포인트	프로토콜	
미국 동부(오하이오)	us-east-2	healthlake.us-east-2.amazonaws.com	HTTPS	
		healthlake-fips.us-east-2.amazonaws.com	HTTPS	

리전 이름	리전	엔드포인트	프로토콜
미국 동부(버지니아 북부)	us-east-1	healthlake.us-east-1.amazonaws.com healthlake-fips.us-east-1.amazonaws.com	HTTPS HTTPS
미국 서부(오리곤)	us-west-2	healthlake.us-west-2.amazonaws.com healthlake-fips.us-west-2.amazonaws.com	HTTPS HTTPS
아시아 태평양(뭄바이)	ap-south-1	healthlake.ap-south-1.amazonaws.com	HTTPS
아시아 태평양(시드니)	ap-southeast-2	healthlake.ap-southeast-2.amazonaws.com	HTTPS
캐나다(중부)	ca-central-1	healthlake.ca-central-1.amazonaws.com	HTTPS
유럽(아일랜드)	eu-west-1	healthlake.eu-west-1.amazonaws.com	HTTPS
유럽(런던)	eu-west-2	healthlake.eu-west-2.amazonaws.com	HTTPS

Service Quotas

서비스 할당량은 AWS 계정의 리소스, 작업 및 항목에 대한 최대값으로 정의됩니다.

> **Note**
>
> 조정 가능한 할당량의 경우 Service Quotas 콘솔을 사용하여 할당량 증가를 요청할 수 있습니다. 자세한 내용은 Service Quotas 사용 설명서의 할당량 증가 요청을 참조하세요.
> Sync Write API 속도는 페이로드 크기에 비례하여 증가하며, 각 1KB 증분은 추가 용량을 소비합니다(예: 4KB 페이로드는 4x 쓰기 용량을 사용함). 선택적 `x-amz-fhir-history-consistency-level` 헤더를 로 설정하면 리소스당 쓰기 용량 소비가 두 `strong`배로 늘어납니다.

번들 내의 리소스는 1KB 페이로드 크기에 따른 표준 읽기/쓰기 제한을 따릅니다. 번들 유형 트랜잭션은 배치 유형보다 쓰기 용량을 두 배 더 소비합니다. 즉, 배치 번들은 트랜잭션 번들보다 초당 두 배 많은 리소스를 처리할 수 있습니다.

다음 표에는에 대한 기본 할당량이 나열되어 있습니다 AWS HealthLake.

이름	기본값	조정 가능	설명
계정당 활성 구독 수	지원되는 각 리전: 100	예	계정당 최대 활성 구독 리소스 수입니다.
데이터 스토어당 활성 구독 리소스 수	지원되는 각 지역: 50	예	데이터 스토어당 최대 활성 구독 리소스 수입니다.
계정당 활성 SubscriptionTopic 수	지원되는 각 리전: 100	예	계정당 활성 SubscriptionTopic 리소스의 최대 수입니다.
데이터 스토어당 활성 SubscriptionTopic 리소스 수	지원되는 각 지역: 50	예	데이터 스토어당 활성 SubscriptionTopic 리소스의 최대 수입니다.
음성 메시지의 문자 수.	각 지원되는 리전: 10,000개	아니요	문서 참조 리소스 유형 (POST/PUT 요청) 내 개별 의료 기록의 최대 문자 수.
동시 StartFHIRExportJob 작업 수	지원되는 각 리전: 1	아니요	동시 StartFHIRExportJob 작업 최대 수.
동시 StartFHIRImportJob 작업 수	각 지원되는 리전: 1개	아니요	동시 StartFHIRImportJob 작업 최대 수.

이름	기본값	조정 가능	설명
계정당 데이터 저장소 수	지원되는 각 지역: 10개	예	계정당 기본 활성 데이터 저장소 최대 수.
StartFHIRImportJob에 포함된 파일 수	지원되는 각 리전: 1,000,000	예	StartFHIRImportJob에 포함된 파일 최대 수
번들당 리소스 수	지원되는 각 리전: 500개	아니요	번들 요청에 허용되는 리소스 최대 수.
DELETE를 사용한 계정당 CancelFHIRExportJob 요청 속도	지원되는 각 리전: 1	아니요	DELETE를 사용하여 계정당 초당 수행할 수 있는 CancelFHIRExportJob 요청 최대 수
계정당 CreateFHIRDatastore 요청 속도	지원되는 각 리전: 1	아니요	계정당 분당 수행할 수 있는 CreateFHIRDatastore 요청 최대 수.
계정당 DeleteFHIRDatastore 요청 속도	지원되는 각 리전: 1	아니요	계정당 분당 수행할 수 있는 DeleteFHIRDatastore 요청 최대 수.
계정당 DescribeFHIRBulkDeleteJob 요청 속도	지원되는 각 리전: 10개	예	계정당 초당 수행할 수 있는 DescribeFHIRBulkDeleteJob 요청 수입니다.
데이터 스토어당 DescribeFHIRBulkDeleteJob 요청 속도	지원되는 각 리전: 10개	예	데이터 스토어당 초당 수행할 수 있는 DescribeFHIRBulkDeleteJob 요청 수입니다.

이름	기본값	조정 가능	설명
계정당 DescribeFHIRDatastore 요청 속도	지원되는 각 리전: 10	아니요	계정당 초당 수행할 수 있는 DescribeFHIRDatastore 요청 최대 수.
계정당 DescribeFHIRExportJob 요청 속도	지원되는 각 리전: 10	아니요	계정당 초당 수행할 수 있는 DescribeFHIRExportJob 요청 최대 수.
GET을 사용한 계정당 DescribeFHIRExportJob 요청 속도	지원되는 각 리전: 10	아니요	GET을 사용한 계정당 초당 수행할 수 있는 DescribeFHIRExportJob 요청 최대 수.
계정당 DescribeFHIRImportJob 요청 속도	지원되는 각 리전: 10	아니요	계정당 초당 수행할 수 있는 DescribeFHIRImportJob 요청 최대 수.
계정당 검색 요청 속도	지원되는 각 리전: 10	아니요	계정당 초당 수행할 수 있는 Discovery 요청 최대 수입니다.
계정당 GET 요청 속도	지원되는 각 리전: 6,000개	예	계정당 초당 수행할 수 있는 GET 요청 최대 수.
데이터 저장소당 GET 요청 속도	각각 지원되는 리전: 3,000개	예	데이터 저장소당 초당 수행할 수 있는 GET 요청 최대 수. 2023년 8월 21일 이전에 생성된 데이터 저장소는 초당 요청 1개로 제한됩니다.

이름	기본값	조정 가능	설명
계정당 GetCapabilities 요청 속도	지원되는 각 리전: 10	아니요	계정당 초당 수행할 수 있는 GetCapabilities 요청 최대 수.
데이터 저장소당 GetExportedFile 요청 속도	지원되는 각 리전: 10	아니요	데이터 저장소당 초당 수행할 수 있는 GetExportedFile 요청 최대 수
계정당 ListFHIRDatastores 요청 속도	지원되는 각 리전: 10	아니요	계정당 초당 수행할 수 있는 ListFHIRDatastores 요청 최대 수.
계정당 ListFHIRExportJobs 요청 속도	지원되는 각 리전: 10	아니요	계정당 초당 수행할 수 있는 ListFHIRExportJobs 요청 최대 수.
계정당 ListFHIRImportJobs 요청 속도	지원되는 각 리전: 10	아니요	계정당 초당 수행할 수 있는 ListFHIRImportJobs 요청 최대 수.
계정당 ListTagsforResource 요청 속도	지원되는 각 리전: 10	아니요	계정당 초당 수행할 수 있는 ListTagsforResource 요청 최대 수.
계정당 SearchEverything 요청 속도	지원되는 각 리전: 10개	예	계정당 초당 수행할 수 있는 SearchEverything 요청의 최대 수입니다.
데이터 스토어당 SearchEverything 요청 속도	지원되는 각 리전: 10개	예	데이터 스토어당 초당 수행할 수 있는 SearchEverything 요청의 최대 수입니다.

이름	기본값	조정 가능	설명
계정당 StartFHIRBulkDeleteJob 요청 속도	지원되는 각 리전: 10개	예	계정당 초당 수행할 수 있는 StartFHIRBulkDeleteJob 요청 수입니다.
데이터 스토어당 StartFHIRBulkDeleteJob 요청 속도	지원되는 각 리전: 10개	예	데이터 스토어당 초당 수행할 수 있는 StartFHIRBulkDeleteJob 요청의 최대 수입니다.
계정당 StartFHIRExportJob 요청 속도	지원되는 각 리전: 1	아니요	계정당 초당 수행할 수 있는 StartFHIRExportJob 요청 최대 수
계정당 GET을 사용한 StartFHIRExportJob 요청 속도	지원되는 각 리전: 1	아니요	계정당 초당 수행할 수 있는 GET을 사용한 StartFHIRExportJob 요청 최대 수
계정당 POST를 사용한 StartFHIRExportJob 요청 속도	지원되는 각 리전: 1	아니요	계정당 초당 수행할 수 있는 POST를 사용한 StartFHIRExportJob 요청 최대 수
계정당 StartFHIRImportJob 요청 속도	지원되는 각 리전: 25	아니요	계정당 초당 수행할 수 있는 StartFHIRImportJob 요청 최대 수
계정당 TagResource 요청 속도	지원되는 각 리전: 10	아니요	계정당 초당 수행할 수 있는 TagResource 요청 최대 수.

이름	기본값	조정 가능	설명
계정당 UntagResource 요청 속도	지원되는 각 리전: 10	아니요	계정당 초당 수행할 수 있는 UntagResource 요청 최대 수.
계정당 ValidateResource 요청 속도	지원되는 각 리전: 2,000	예	계정당 초당 수행할 수 있는 최대 ValidateResource 요청 수입니다. 2023년 8월 21일 이전에 생성된 데이터 저장소는 초당 요청 300개로 제한됩니다.
데이터 스토어당 ValidateResource 요청 속도	지원되는 각 리전: 1,000	예	데이터 스토어당 초당 수행할 수 있는 최대 ValidateResource 요청 수입니다. 2023년 8월 21일 이전에 생성된 데이터 저장소는 초당 요청 300개로 제한됩니다.
계정당 쓰기 요청 속도	지원되는 각 리전: 6,000개	예	계정당 초당 수행할 수 있는 CREATE|UPDATE|DELETE 요청 최대 수
데이터 저장소당 쓰기 요청 속도	지원되는 각 리전: 3,000	예	데이터 저장소당 초당 수행할 수 있는 CREATE|UPDATE|DELETE 요청 최대 수. 2023년 8월 21일 이전에 생성된 데이터 저장소는 초당 요청 300개로 제한됩니다.

이름	기본값	조정 가능	설명
GET을 사용한 계정당 검색 요청 속도	지원되는 각 리전: 200회	예	계정당 GET을 사용하여 초당 수행할 수 있는 검색 요청의 최대 수.
데이터 저장소당 GET을 사용한 검색 요청 속도	지원되는 지역: 100개	예	데이터 저장소당 GET을 사용하여 초당 수행할 수 있는 검색 요청 최대 수.
POST를 사용한 계정당 검색 요청 속도	지원되는 각 리전: 200개	예	계정당 POST를 사용하여 초당 수행할 수 있는 검색 요청의 최대 수.
데이터 저장소당 POST를 사용한 검색 요청 속도	지원되는 각 리전: 100	예	데이터 저장소당 POST를 사용하여 초당 수행할 수 있는 검색 요청 최대 수.
가져온 개별 파일의 크기	지원되는 각 리전: 50GB	아니요	StartFHIRImportJob에 포함된 개별 파일의 최대 크기(GB).
데이터 스토어당 대기 중인 총 비동기 번들 트랜잭션 수	지원되는 각 리전: 500	예	지정된 시간에 데이터 스토어당 대기 중인 비동기 번들 트랜잭션의 최대 수입니다.
데이터 저장소당 대기 중인 대량 내보내기 작업의 총 수	지원되는 각 지역: 25	예	지정된 시간에 데이터 저장소당 대기 중인 대량 내보내기 작업의 최대 수
데이터 저장소당 대기 중인 대량 가져오기 작업의 총 수	지원되는 각 지역: 25	예	지정된 시간에 데이터 저장소당 대기 중인 대량 가져오기 작업의 최대 수

이름	기본값	조정 가능	설명
가져오기 작업 전체 크기	지원되는 각 리전: 5,000기가바이트	예	가져오기 작업에 포함된 모든 파일의 최대 크기(GB)

HealthLake용 Synthea 사전 로드된 데이터 형식

HealthLake는 SYNTHEA만 사전 로드된 데이터 유형으로 지원합니다. Synthea는 `Patient` 의료 기록을 모델링하는 합성 환자 생성기입니다. HealthLake가 FHIR R4-compliant 리소스를 생성할 수 있도록 하는 오픈 소스 Git 리포지토리에서 호스팅`Bundle`되므로 사용자는 실제 환자 데이터를 사용하지 않고도 모델을 테스트할 수 있습니다.

사전 로드된 HealthLake 데이터 스토어에서 사용할 수 있는 리소스 유형은 다음과 같습니다. HealthLake 데이터 스토어를 Synthea 데이터로 사전 로드하는 방법에 대한 자세한 내용은 섹션을 참조하세요HealthLake 데이터 스토어 생성.

> **Note**
>
> HealthLake 지원 FHIR R4 리소스의 전체 목록을 보려면 섹션을 참조하세요HealthLake에 지원되는 FHIR R4 리소스 유형.

HealthLake에서 지원하는 Synthea FHIR 리소스 유형

AllergyIntolerance	Location
CarePlan	MedicationAdministration
CareTeam	MedicationRequest
Claim	관측치
조건	Organization

장치	환자
DiagnosticReport	프랙티셔너
상황	PractitionerRole
ExplanationofBenefit	절차
ImagingStudy	증명
면역화	

AWS HealthLake 샘플 프로젝트

다음 블로그 게시물 예제와 같이 HealthLake를 다른 AWS 서비스와 함께 사용하여 분석을 강화할 수 있습니다.

HealthLake 통합 분석

- 의 모집단 상태 애플리케이션 AWS HealthLake - 1부: Amazon Quick을 사용한 분석 및 모니터링.
- AWS HealthLake 정규화된 데이터로 Amazon SageMaker AI를 사용하여 예측적 질병 모델을 구축합니다.
- AWS AI 서비스를 사용하여 인지 검색 및 건강 지식 그래프를 구축합니다.

HealthLake 이벤트 모니터링

- Amazon EventBridge와 통합 AWS HealthLake.

문제 해결 AWS HealthLake

다음 주제에서는 AWS CLI, AWS SDKs 또는 HealthLake 콘솔을 사용할 때 발생할 수 있는 오류 및 문제에 대한 문제 해결 조언을 제공합니다. 이 섹션에 나열되지 않은 문제가 발견되면이 페이지의 오른쪽 사이드바에 있는 피드백 제공 버튼을 사용하여 보고합니다.

주제

- 데이터 스토어 작업
- 가져오기 작업

- FHIR APIs
- NLP 통합
- SQL 통합

데이터 스토어 작업

문제: HealthLake 데이터 스토어를 생성하려고 하면 다음 오류가 발생합니다.

```
AccessDeniedException: Insufficient Lake Formation permission(s): Required Database on
 Catalog
```

2022년 11월 14일에 HealthLake는 새 데이터 스토어를 생성하는 데 필요한 IAM 권한을 업데이트했습니다. 자세한 내용은 HealthLake를 사용하도록 IAM 사용자 또는 역할 구성(IAM 관리자) 단원을 참조하십시오.

문제: AWS SDKs를 사용하여 HealthLake 데이터 스토어를 생성할 때 데이터 스토어 생성 상태는 예외 또는 알 수 없음 상태를 반환합니다.

`DescribeFHIRDatastore` 또는 `ListFHIRDatastores` API 호출에서 예외 또는 알 수 없는 데이터 스토어 상태가 반환되는 경우 AWS SDK를 최신 버전으로 업데이트합니다.

가져오기 작업

문제: 데이터가 FHIR R4 형식이 아닌 경우에도 HealthLake를 계속 사용할 수 있나요?

FHIR R4 형식의 데이터만 HealthLake 데이터 스토어로 가져올 수 있습니다. 기존 상태 데이터를 FHIR R4 형식으로 변환하는 데 도움이 될 수 있는 파트너 목록은 AWS HealthLake 파트너를 참조하세요.

문제: FHIR 가져오기 작업이 실패한 이유는 무엇입니까?

가져오기 작업이 성공하면 `.ndjson` 형식의 결과(출력 로그)가 있는 폴더가 생성되지만 개별 레코드를 가져오지 못할 수 있습니다. 이 경우 가져오기에 실패한 레코드 매니페스트와 함께 두 번째 `FAILURE` 폴더가 생성됩니다. 자세한 내용은 를 사용하여 FHIR 데이터 가져오기 AWS HealthLake 단원을 참조하십시오.

가져오기 작업이 실패한 이유를 분석하려면 `DescribeFHIRImportJob` API를 사용하여 JobProperties를 분석합니다. 다음은 권장됩니다.

- 상태가 `FAILED` 이고 메시지가 있는 경우 실패는 입력 데이터 크기 또는 HealthLake 할당량을 초과하는 입력 파일 수와 같은 작업 파라미터와 관련이 있습니다.

- 가져오기 작업 상태가 인 경우 매니페스트 파일에서 성공적으로 가져오지 못한 파일에 manifest.json대한 정보를 COMPLETED_WITH_ERRORS확인합니다.
- 가져오기 작업 상태가 FAILED 이고 메시지가 없는 경우 작업 출력 위치로 이동하여 매니페스트 파일에 액세스합니다manifest.json.

각 입력 파일에 대해 가져오기에 실패한 리소스에 대한 입력 파일 이름이 있는 실패 출력 파일이 있습니다. 응답에는 입력 데이터의 위치에 해당하는 줄 번호(lineId), FHIR 응답 객체(UpdateResourceResponse) 및 응답의 상태 코드(statusCode)가 포함됩니다.

샘플 출력 파일은 다음과 유사할 수 있습니다.

```
{"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}
```

위 예제는 입력 파일의 해당 입력 행에서 3, 4, 7, 9, 15행에 오류가 있음을 보여줍니다. 이러한 각 줄에 대한 설명은 다음과 같습니다.

- 3행에서 응답은 입력 파일의 3행에 `resourceType` 제공된가 유효하지 않다고 설명합니다.
- 5행에서 응답은 입력 파일의 5행에 FHIR 검증 오류가 있음을 설명합니다.
- 7행에서 응답은 입력으로 `resourceId` 제공된에 검증 문제가 있음을 설명합니다.
- 9행에서 응답은 입력 파일에 유효한 리소스 ID가 포함되어야 한다고 설명합니다.

- 15행에서 입력 파일의 응답은 파일이 유효한 JSON 형식이 아니라는 것입니다.

FHIR APIs

문제: FHIR RESTful APIs에 대한 권한 부여를 구현하려면 어떻게 해야 합니까?

사용할 데이터 스토어 권한 부여 전략를 결정합니다.

를 사용하여 SigV4 권한 부여를 생성하려면 다음 예제와 유사한 스크립트를 AWS SDK for Python (Boto3)생성합니다.

```
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())
```

문제: 고객 관리형 KMS 키로 암호화된 데이터 스토어에 FHIR RESTful APIs를 사용할 때 *AccessDenied* 오류가 발생하는 이유는 무엇입니까?

사용자 또는 역할이 데이터 스토어에 액세스하려면 고객 관리형 키와 IAM 정책 모두에 대한 권한이 필요합니다. 사용자는 고객 관리형 키를 사용하는 데 필요한 IAM 권한이 있어야 합니다. 사용자가 HealthLake에 고객 관리형 KMS 키를 사용할 수 있는 권한을 부여하는 권한을 취소하거나 사용 중지한 경우 HealthLake는 `AccessDenied` 오류를 반환합니다.

HealthLake에는 고객 데이터에 액세스하고, 데이터 스토어로 가져온 새 FHIR 리소스를 암호화하고, 요청 시 FHIR 리소스를 복호화할 수 있는 권한이 있어야 합니다. 자세한 내용은 권한 문제 해결을 참조하세요 AWS KMS.

문제: 10MB 문서를 사용하여 HealthLake에 대한 FHIR *POST* API 작업이 *413 Request Entity Too Large* 오류를 반환합니다.

AWS HealthLake 에는 지연 시간 및 제한 시간 증가를 방지하기 위해 5MB의 동기식 생성 및 업데이트 API 제한이 있습니다. 대량 가져오기 API를 사용하여 `Binary` 리소스 유형을 사용하여 최대 164MB의 대용량 문서를 수집할 수 있습니다.

NLP 통합

문제: HealthLake의 통합 자연어 처리 기능을 켜려면 어떻게 해야 합니까?

2022년 11월 14일부터 HealthLake 데이터 스토어의 기본 동작이 변경되었습니다.

현재 데이터 스토어: 현재 모든 HealthLake 데이터 스토어는 base64 인코딩 `DocumentReference` 리소스에서 자연어 처리(NLP) 사용을 중단합니다. 즉, 새 `DocumentReference` 리소스는 NLP를 사용하여 분석되지 않으며 리소스 유형의 텍스트를 기반으로 새 `DocumentReference` 리소스가 생성되지 않습니다. 기존 `DocumentReference` 리소스의 경우 NLP를 통해 생성된 데이터 및 리소스는 남아 있지만 2023년 2월 20일 이후에는 업데이트되지 않습니다.

새 데이터 스토어: 2023년 2월 20일 이후에 생성된 HealthLake 데이터 스토어는 base64 인코딩 `DocumentReference` 리소스에서 자연어 처리(NLP)를 수행하지 않습니다.

HealthLake NLP 통합을 켜려면를 사용하여 지원 사례를 생성합니다AWS Support Center Console. 사례를 생성하려면에 로그인 AWS 계정한 다음 사례 생성을 선택합니다. 사례 및 사례 관리 생성에 대한 자세한 내용은 지원 사용 설명서의 지원 사례 및 사례 관리 생성을 참조하세요.

문제: >통합 NLP에서 처리할 수 없는 *DocumentReference* 리소스를 찾으려면 어떻게 해야 합니까?

`DocumentReference` 리소스가 유효하지 않은 경우 HealthLake는 통합 의료 NLP 출력에 제공하는 대신 검증 오류를 나타내는 확장을 제공합니다. NLP 처리 중에 검증 오류가 발생한 `DocumentReference` 리소스를 찾으려면 검색 키 cm-decoration-status 및 검색 값

VALIDATION_ERROR과 함께 HealthLake의 FHIR `search` 함수를 사용할 수 있습니다. 이 검색에는 오류의 특성을 설명하는 오류 메시지와 함께 검증 오류가 발생한 모든 `DocumentReference` 리소스가 나열됩니다. 검증 오류가 있는 `DocumentReference` 리소스의 확장 필드 구조는 다음 예제와 유사합니다.

```
"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**
>
> NLP 데코레이션이 10,000개 이상의 중첩 객체를 생성하는 경우에도가 발생할 VALIDATION_ERROR 수 있습니다. 이 경우 처리 전에 문서를 더 작은 문서로 분할해야 합니다.

SQL 통합

문제: 새 데이터 레이크 관리자를 추가할 *permissions error: lakeformation:PutDataLakeSettings* 때 Lake Formation을 받는 이유는 무엇입니까?

IAM 사용자 또는 역할에 `AWSLakeFormationDataAdmin` AWS 관리형 정책이 포함된 경우 새 데이터 레이크 관리자를 추가할 수 없습니다. 다음을 포함하는 오류가 발생합니다.

```
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
```

AWS 관리형 정책은 IAM 사용자 또는 역할을 AWS Lake Formation 데이터 레이크 관리자로 추가하는 데 `AdministratorAccess` 필요합니다. IAM 사용자 또는 역할에도 `AWSLakeFormationDataAdmin` 작업이 포함된 경우 작업이 실패합니다. `AWSLakeFormationDataAdmin` AWS 관리형 정책에는 AWS Lake Formation API 작업에 대한 명시적 거부가 포함되어 있습니다`PutDataLakeSetting`. `AdministratorAccess` 관리형 정책을 AWS 사용하여에 대한 전체 액세스 권한이 있는 관리자도 `AWSLakeFormationDataAdmin` 정책에 의해 제한될 수 있습니다.

문제: Amazon Athena SQL 통합을 사용하도록 기존 HealthLake 데이터 스토어를 마이그레이션하려면 어떻게 해야 합니까?

2022년 11월 14일 이전에 생성된 HealthLake 데이터 스토어는 작동하지만 SQL을 사용하여 Athena에서 쿼리할 수 없습니다. Athena를 사용하여 기존 데이터 스토어를 쿼리하려면 먼저 새 데이터 스토어로 마이그레이션해야 합니다.

HealthLake 데이터를 새 데이터 스토어로 마이그레이션하려면

1. 새 데이터 스토어를 생성합니다.
2. 기존의 데이터를 Amazon S3 버킷으로 내보냅니다.
3. Amazon S3 버킷에서 새 데이터 스토어로 데이터를 가져옵니다.

> **Note**
>
> Amazon S3 버킷으로 데이터를 내보내면 추가 요금이 발생합니다. 추가 요금은 내보내는 데이터의 크기에 따라 달라집니다.

문제: SQL 통합을 위한 새 HealthLake 데이터 스토어를 생성할 때 데이터 스토어 상태가에서 변경되지 않습니다*`Creating`*.

새 HealthLake 데이터 스토어를 생성하려고 하는데 데이터 스토어 상태가 생성에서 변경되지 않는 경우를 사용하도록 Athena를 업데이트해야 합니다 AWS Glue Data Catalog. 자세한 내용은 Amazon Athena 사용 설명서의 AWS Glue 데이터 카탈로그로 step-by-step 업그레이드를 참조하세요.

를 성공적으로 업그레이드 AWS Glue Data Catalog한 후 HealthLake 데이터 스토어를 생성할 수 있습니다.

이전 HealthLake 데이터 스토어를 제거하려면를 사용하여 지원 사례를 생성합니다AWS Support Center Console. 사례를 생성하려면에 로그인한 AWS 계정다음 사례 생성을 선택합니다. 자세한 내용은 지원 사용 설명서의 지원 사례 및 사례 관리 생성을 참조하세요.

문제: 데이터를 새 HealthLake 데이터 스토어로 가져온 후 Athena 콘솔이 작동하지 않음

데이터를 새 HealthLake 데이터 스토어로 가져온 후에는 데이터를 즉시 사용하지 못할 수 있습니다. 이는 데이터가 Apache Iceberg 테이블로 수집될 시간을 허용하기 위한 것입니다. 나중에 다시 시도하세요.

문제: Athena의 검색 결과를 다른 AWS 서비스에 연결하려면 어떻게 해야 하나요?

Athena의 검색 결과를 다른 AWS 서비스와 공유할 때 SQL 검색 쿼리의 `json_extract[1]` 일부로를 사용할 때 문제가 발생할 수 있습니다. 이 문제를 해결하려면를 로 업데이트해야 합니다`CATVAR`.

저장 결과, 테이블(정적) 또는 보기(동적)를 생성하려고 할 때이 문제가 발생할 수 있습니다.

AWS SDK에서 HealthLake 사용

AWS 소프트웨어 개발 키트(SDKs)는 널리 사용되는 여러 프로그래밍 언어에 사용할 수 있습니다. 각 SDK는 개발자가 선호하는 언어로 애플리케이션을 쉽게 구축할 수 있도록 하는 API, 코드 예제 및 설명서를 제공합니다.

SDK 설명서	코드 예제
AWS SDK for C++	AWS SDK for C++ 코드 예제
AWS CLI	AWS CLI 코드 예제
AWS SDK for Go	AWS SDK for Go 코드 예제
AWS SDK for Java	AWS SDK for Java 코드 예제
AWS SDK for JavaScript	AWS SDK for JavaScript 코드 예제
AWS SDK for Kotlin	AWS SDK for Kotlin 코드 예제
AWS SDK for .NET	AWS SDK for .NET 코드 예제
AWS SDK for PHP	AWS SDK for PHP 코드 예제

SDK 설명서	코드 예제
AWS Tools for PowerShell	AWS Tools for PowerShell 코드 예제
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) 코드 예제
AWS SDK for Ruby	AWS SDK for Ruby 코드 예제
AWS SDK for Rust	AWS SDK for Rust 코드 예제
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP 코드 예제
AWS SDK for Swift	AWS SDK for Swift 코드 예제

예제 가용성

필요한 예제를 찾을 수 없습니까? 이 페이지 하단의 피드백 제공 링크를 사용하여 코드 예시를 요청하세요.

AWS HealthLake 릴리스

다음 표에는 기능 및 업데이트가 릴리스된 시기가 나와 있습니다 AWS HealthLake. 릴리스에 대한 자세한 내용은 연결된 주제를 참조하세요.

변경 사항	설명	날짜
$bulk-member-match 작업	AWS HealthLake 는 이제 여러 멤버 일치 요청을 비동기적으로 처리하는 `$bulk-member-match` 작업을 지원합니다. 이 작업을 통해 의료 기관은 단일 대량 요청으로 인구 통계 및 적용 범위 정보를 사용하여 여러 의료 시스템에서 수백 명의 멤버의 고유 식별자를 효율적으로 일치시킬 수 있습니다. • 데이터 스토어당 최대 5개의 동시 작업으로 요청당 최대 500명의 멤버 처리 • MatchedMembers, NonMatchedMembers 및 ConsentConstrained Members 그룹으로 분류된 결과 • end-to-end 대량 데이터 워크플로를 `$davinci-data-export` 위해와 통합 자세한 내용은 the section called “$bulk-member-match” 단원을 참조하십시오.	2026년 4월 1일

비동기 번들 트랜잭션	AWS HealthLake 는 이제 비동기 `Bundle` 유형를 `transaction` 지원하므로 최대 500개의 리소스가 있는 트랜잭션을 제출할 수 있습니다. HealthLake는 처리를 위해 트랜잭션을 대기열에 넣고 즉시 폴링 URL을 반환하여 상태를 확인하고 결과를 검색합니다. 자세한 내용은 비동기 번들 트랜잭션을 참조하세요.	2026년 3월 24일
$davinci-data-export에 대한 DaVinci PDex 내보내기 유형 davinci-data-export	이제이 `$davinci-data-export` 작업은 공급자 액세스, Payer-to-Payer 및 멤버 액세스 APIs에 대한 PDex 내보내기 유형을 지원합니다. • ExplanationOfBenefit 리소스에 대한 프로필 기반 포함 로직 • `_includeEOB2xWoFinancial` 파라미터를 사용한 금융 데이터 변환 • 임상 및 클레임 데이터에 대한 5년 시간 필터	2026년 3월 20일
$export 및 $davinci-data-export의 _until 파라미터	내보내기 작업에 대한 임시 필터링 파라미터	2026년 2월 26일
_include 검색 파라미터	HealthLake는 이제 다음을 지원합니다.* 및 include:iterate	2026년 2월 26일
CMS 상호 운용성 엔드포인트	이 기능을 사용하면 CMS 범주별로 API 사용량을 추적한 다음 규정 준수를 위해 사용량 지표를 보고할 수 있습니다.	2026년 2월 26일

번들 메시지 유형 지원	메시지 유형이 있는 FHIR 번들 리소스에 대한 제한된 지원	2026년 2월 26일
새 IGs에 대한 지원 추가	AWS HealthLake 는 CMS 0057F에 대한 FHIR 구현 가이드(IGs) 지원을 확장했습니다. • CARIN 블루 버튼 2.0.0 및 2.1.0 지원 • Da Vinci Payer Data Exchange 2.0.0 및 2.1.0 지원 • DaVinci 지급인 데이터 교환(PDex) 미국 의약품집 2.0.1 및 2.1.0 지원 • Da Vinci 임상 데이터 교환(CDex) 2.1.0 지원 • Da Vinci 사전 승인 지원(PAS) FHIR IG 2.1.0 지원	2026년 2월 26일
$submit 작업	`$submit` 작업을 사용하면 승인을 위해 지급자에게 사전 권한 부여 요청을 전자적으로 제출할 수 있습니다.	2026년 2월 26일
$questionnaire-package 작업	`$questionnaire-package` 작업은 FHIR 설문과 설문을 렌더링하고 처리하는 데 필요한 모든 종속성이 포함된 포괄적인 번들을 검색합니다.	2026년 2월 26일
$inquire 작업	`$inquire` 작업을 통해 이전에 제출한 사전 권한 부여 요청의 상태를 확인할 수 있습니다.	2026년 2월 26일

$member-remove 작업	$member-remove 작업을 사용하면의 FHIR 멤버 속성 목록(그룹 리소스)에서 멤버를 제거할 수 있습니다 AWS HealthLake.	2025년 11월 12일
$member-match 작업	AWS HealthLake 는 이제 환자 리소스에 대한 $member-match 작업을 지원하므로 의료 기관은 인구 통계 및 적용 범위 정보를 사용하여 다양한 의료 시스템에서 멤버의 고유 식별자를 찾을 수 있습니다.	2025년 11월 12일
$member-add 작업	FHIR $member-add 작업은 그룹 리소스, 특히 멤버 속성 목록에 멤버(환자)를 추가합니다.	2025년 11월 12일
$davinci-data-export 작업	$davinci-data-export 작업은 멤버 속성 목록 데이터를 내보낼 수 있는 비동기 FHIR 작업입니다 AWS HealthLake.	2025년 11월 12일
$confirm-attribution-list 작업	생산자에게 소비자가 더 이상 속성 목록을 변경할 필요가 없음을 나타내며, 비활성 멤버를 제거하고 상태를 "최종"으로 변경하여 속성 목록을 마무리합니다.	2025년 11월 12일
$attribution-status 작업	특정 멤버의 어트리뷰션 상태를 검색하여 환자와 관련된 모든 어트리뷰션 리소스가 포함된 번들을 반환합니다.	2025년 11월 12일

FHIR 구독	HealthLake는 FHIR 구독을 지원하므로 특정 의료 데이터 변경이 발생할 때 실시간 알림을 수신하고 이벤트 기반 워크플로를 구축할 수 있습니다.	2025년 10월 30일
캐나다 몬트리올로 리전 확장	HealthLake는 캐나다(몬트리올) 리전에서 사용할 수 있습니다. 자세한 내용은 서비스 엔드포인트를 참조하세요.	2025년 10월 17일
새 IGs에 대한 지원 추가	AWS HealthLake 는 다음 캐나다 시장에 대한 FHIR 구현 가이드(IGs) 지원을 확장했습니다. • 캐나다 의료 시스템 간 상호 운용성을 위한 핵심 데이터 요소와 제약 조건을 정의하는 CA Core+캐나다 기준 FHIR 프로파일입니다. • CA:eReC 캐나다 전역의 의료 서비스 공급자 간 전자 추천 및 상담을 위한 Pan-Canadian eReferral-eConsultStandardized FHIR 사양입니다. • 환자 요약 캐나다 에디션(PS-CA) 의료 환경 전반에서 필수 환자 건강 정보를 공유하기 위한 국제 환자 요약(IPS)의 캐나다 적응. • 온타리오 디지털 건강 약물 RepositoryOntario별 FHIR 프로파일입니다.	2025년 10월 17일

리전별 IG 지원	HealthLake는 이제 리전별 IGs 지원합니다. 자세한 내용은 프로파일 검증을 참조하세요.	2025년 10월 8일
패치 작업	HealthLake를 사용하면 전체 리소스를 업데이트하지 않고도 JSON 패치 작업을 사용하여 FHIR 리소스의 특정 요소를 수정할 수 있습니다.	2025년 8월 18일
$validate 작업	HealthLake를 사용하면 스토리지 작업을 수행하지 않고도 사양 및 프로필에 대해 FHIR 리소스를 검증하여 자세한 검증 결과를 반환할 수 있습니다.	2025년 8월 18일
$purge 작업	HealthLake에는 데이터 스토어에서 환자 구획 내의 모든 리소스를 영구적으로 삭제하는 기능이 포함되어 있습니다.	2025년 8월 18일
$lookup 작업	HealthLake는 코드 및 시스템 식별자를 제공하여 CodeSystem의 특정 개념에 대한 세부 정보를 검색할 수 있는 기능을 제공합니다.	2025년 8월 18일
$expand 작업	HealthLake를 사용하면 이제 ValueSet 리소스를 확장하여 고객 수집 ValueSets.	2025년 8월 18일
$erase 작업	HealthLake는 이제 데이터 스토어에서 특정 리소스와 모든 기록 버전을 영구적으로 삭제할 수 있습니다.	2025년 8월 18일

$document 작업	HealthLake는 구성 리소스를 참조된 모든 리소스와 번들링하여 전체 임상 문서를 단일 문서 번들로 생성할 수 있도록 지원합니다.	2025년 8월 18일
:검색 한정자 아래	HealthLake는 용어 시스템에서 지정된 URI보다 계층적으로 낮은 URI 값을 검색하는 기능을 도입합니다.	2025년 8월 8일
조건부 삭제	HealthLake는 이제 FHIR 조건부 삭제를 지원하므로 의료 기관은 논리적 FHIR ID가 아닌 검색 기준에 따라 기존 리소스를 삭제할 수 있습니다. 자세한 내용은 조건에 따라 FHIR 리소스 삭제를 참조하세요.	2025년 7월 7일
새 IGs에 대한 지원 추가	AWS HealthLake 는 다음에 대한 FHIR 구현 가이드(IGs • FHIR을 사용하여 USCDI 4.0 표준을 구현하는 방법을 지정하는 US Core 7.0.0 • 영국 전역 FHIR 구현 지침을 제공하는 UK Core 2.0.1 구현 가이드	2025년 7월 7일
더블린 아일랜드로 리전 확장	HealthLake는 EU(더블린) 리전에서 사용할 수 있습니다. 자세한 내용은 서비스 엔드포인트를 참조하세요.	2025년 6월 9일

번들 유형 트랜잭션	HealthLake는 이제 FHIR 번들 유형 '트랜잭션'을 지원하므로 의료 기관은 여러 리소스를 단일 원자성 작업으로 제출할 수 있습니다. 이를 통해 데이터 교환 및 통합 워크플로의 효율성을 높일 수 있습니다. 예를 들어 의료 공급자는 이제 단일 트랜잭션에서 환자 레코드, 약물 목록 및 예약을 업데이트하여 복잡성과 잠재적 오류를 줄일 수 있습니다. 자세한 내용은 FHIR 리소스 번들링을 참조하세요.	2025년 4월 28일
새 IGs에 대한 지원 추가	AWS HealthLake AWS HealthLake는 다음에 대한 FHIR 구현 가이드(IGs) 지원을 확장했습니다. • NCQA HEDIS® 구현 가이드(0.3.1): 의료 효과 데이터 및 정보 세트(HEDIS)에 대한 품질 측정 및 보고를 지원합니다. • 국제 환자 요약(IPS)(2.0.0): 환자의 치료 연속성을 지원하기 위해 필수 건강 정보를 교환할 수 있습니다. • 품질 측정(5.0.0): 품질 측정 정의 및 데이터의 표현 및 교환을 지원합니다. • Genomics Reporting(3.0.0): 게놈 데이터와 보고서의 교환을 용이하게 합니다.	2025년 4월 28일

자격 증명 키	HealthLake는 이제 FHIR POST 작업에 대한 멱등성 키를 지원하여 리소스 생성 중에 데이터 무결성을 보장하는 강력한 메커니즘을 제공합니다. 자세한 내용은 Idempotency and Concurrency를 참조하세요.	2025년 4월 18일
FHIR 기록 일관성	HealthLake는 이제 새 `x-amz-fhir-history-consistency-level` 헤더를 통해 기록 지원 데이터 스토어에 대한 강력한 일관성을 지원합니다. 'strong'으로 설정하면 FHIR 검색 결과에는 업데이트 상태에 관계없이 인덱싱된 모든 레코드가 포함됩니다. 자세한 내용은 FHIR 검색 일관성 수준을 참조하세요.	2025년 4월 18일
태그 및 'if-match'	HealthLake는 이제 eTag 지원을 제공하므로 클라이언트가 'If-Match' 헤더를 사용하여 멱등성 업데이트를 보장할 수 있습니다. 이렇게 하면 동시 업데이트 중에 실수로 덮어쓰는 것을 방지하여 데이터 무결성을 유지할 수 있습니다. 이는 여러 시스템이 동일한 레코드를 동시에 업데이트하려고 시도할 수 있는 대용량 의료 환경에서 특히 유용합니다. 자세한 내용은 AWS HealthLake의 ETag를 참조하세요.	2025년 4월 18일

번들의 조건부 PUTs	HealthLake는 이제 FHIR 번들에 대한 조건부 업데이트를 지원하므로 의료 기관에서 데이터를 보다 유연하게 관리하고 업데이트할 수 있습니다. 이제 클라이언트는 번들 트랜잭션의 일부로 리소스를 조건부로 생성, 업데이트 또는 삭제하는 기준을 지정할 수 있습니다. 이렇게 하면 시스템 간 데이터 동기화 프로세스가 간소화되고 복잡한 클라이언트 측 로직이 필요하지 않습니다. 자세한 내용은 번들의 조건부 PUTs를 참조하세요.	2025년 4월 18일

FHIR V2 범위의 SMART	HealthLake는 FHIR 리소스를 생성, 읽기, 업데이트, 삭제 및 검색하기 위한 FHIR V2 범위의 SMART를 지원합니다. 자세한 내용은 HealthLake의 FHIR 리소스 범위에 대한 SMART를 참조하세요. • FHIR V2의 SMART 범위는 01/22/2025 이후에 생성된 모든 HealthLake 데이터 스토어에서 사용할 수 있습니다. 이 날짜 이전에 데이터 스토어를 생성한 경우 FHIR V2 범위에서 SMART를 활성화하도록 지원 티켓을 제출할 수 있습니다. 를 사용하여 사례를 생성합니다AWS Support Center Console. 사례를 생성하려면에 로그인 AWS 계정 하고 사례 생성을 선택합니다.	2025년 1월 22일
FHIR US Core Profile, 버전 6.1.0	HealthLake는 FHIR US Core Profile 버전 6.1.0을 지원합니다. 자세한 내용은 HealthLake에 대한 FHIR 프로파일 검증을 참조하세요.	2025년 1월 22일
GET을 사용한 FHIR $export	HealthLake는 `$export`에서 FHIR을 지원합니다`GET`. 자세한 내용은 FHIR을 사용하여 HealthLake 데이터 내보내기를 참조하세요`$export`.	2025년 1월 22일

테스트된 코드 예제가 포함된 리팩터링된 개발자 안내서	HealthLake는 네이티브 AWS CLI 및 AWS SDK 작업에 대해 테스트된 코드 예제가 포함된 리팩터링된 개발자 안내서를 소개합니다. 또한 이제 지원되는 모든 FHIR API 상호 작용에 프로시저를 사용할 수 있습니다. 자세한 내용은 코드 예제 및 FHIR 리소스 관리를 참조하세요.	2024년 12월 18일
FHIR `history` 및 `vread` 상호 작용	HealthLake는 특정 리소스의 기록을 검색하기 위한 FHIR `history` 상호 작용과 리소스의 버전별 읽기를 수행하기 위한 `vread` 상호 작용을 지원합니다. 자세한 내용은 FHIR 리소스 기록 읽기를 참조하세요. • FHIR 리소스`history`는 기본적으로 10/25/2024 이후에 생성된 모든 HealthLake 데이터 스토어에서 활성화됩니다. 이 날짜 이전에 데이터 스토어가 생성된 경우 지원 티켓을 제출하여 FHIR `history` 상호 작용을 활성화할 수 있습니다. 를 사용하여 사례를 생성합니다AWS Support Center Console. 사례를 생성하려면에 AWS 계정 로그인하고 사례 생성을 선택합니다.	2024년 10월 25일

FHIR Patient/$everything 작업	HealthLake는 Patient 리소스 및 모든 관련 리소스를 검색하기 위한 FHIR Patient/$everything 작업을 지원합니다. 이 작업을 사용하면 환자의 전체 레코드에 액세스하거나 Patient 데이터를 대량으로 다운로드할 수 있습니다. 자세한 내용은 를 사용하여 환자 데이터 가져오기Patient/$everything 를 참조하세요. • FHIRPatient/$everything 은 기본적으로 02/27/2024 이후에 생성된 모든 HealthLake 데이터 스토어에서 활성화됩니다. 이 날짜 이전에 데이터 스토어가 생성된 경우 지원 티켓을 제출하여 Patient/$everything 작업을 활성화할 수 있습니다. 를 사용하여 사례를 생성합니다AWS Support Center Console. 사례를 생성하려면에 AWS 계정 로그인하고 사례 생성을 선택합니다.	2024년 2월 27일

FHIR VerificationResult 리소스	HealthLake는 하나 이상의 요소에 대한 검증 요구 사항, 소스, 상태 및 날짜를 설명하기 위한 FHIR VerificationResult 리소스 유형을 지원합니다. 자세한 내용은 HealthLake에 대한 FHIR R4 리소스 유형을 참조하세요.	2023년 12월 9일

FHIR $export 작업	HealthLake는 HealthLake 데이터 스토어에서 상태 데이터를 대량으로 내보내기 위한 FHIR $export 작업을 지원합니다. 자세한 내용은 FHIR을 사용하여 HealthLake 데이터 내보내기를 참조하세요$export. • FHIR$export은 2023년 6월 1일 이후에 생성된 모든 HealthLake 데이터 스토어에서 기본적으로 활성화됩니다. 이 날짜 이전에 데이터 스토어가 생성된 경우 지원 티켓을 제출하여 $export 작업을 활성화할 수 있습니다. 를 사용하여 사례를 생성합니다AWS Support Center Console. 사례를 생성하려면에 로그인 AWS 계정 하고 사례 생성을 선택합니다. • 06/01/23 이전에 생성된 HealthLake 데이터 스토어는 시스템 전체 내보내기에 대한 $export 작업 요청만 지원합니다. • 06/01/23 이전에 생성된 HealthLake 데이터 스토어는 데이터 스토어의 엔드포인트에서 GET 요청을 $export 사용하여 FHIR 상태를 가져오는 것을 지원하지 않습니다.	2023년 6월 1일

FHIR에서 SMART 지원	HealthLake는 FHIR 권한 부여에서 SMART에 대한 지원을 추가합니다. 자세한 내용은 FHIR 지원에 대한 SMART를 참조하세요 AWS HealthLake.	2023년 5월 31일
FHIR 프로파일 검증	HealthLake는 기본 리소스 유형에 대한 제약 조건 및/또는 확장을 사용하여 특정 리소스 유형 정의를 정의하기 위한 FHIR 프로파일 검증을 지원합니다. 자세한 내용은 프로파일 검증을 참조하세요.	2023년 5월 31일
아시아 태평양(뭄바이) 리전	HealthLake는 아시아 태평양(뭄바이) 리전에서 사용할 수 있습니다. 자세한 내용은 서비스 엔드포인트를 참조하세요.	2023년 4월 4일
자연어 처리가 기본적으로 꺼져 있음	HealthLake는 2023년 2월 20일부터 모든 데이터 스토어에서 통합 자연어 처리(NLP)를 해제했습니다. 지원 티켓을 제출하여 통합 NLP 기능을 활성화할 수 있습니다. 를 사용하여 사례를 생성합니다AWS Support Center Console. 사례를 생성하려면에 로그인 AWS 계정 하고 사례 생성을 선택합니다. 통합 NLP에 대한 자세한 내용은 NLP와 HealthLake 통합을 참조하세요. •	2023년 2월 20일

Amazon Athena를 사용한 SQL 인덱스 및 쿼리	HealthLake는 Amazon Athena를 사용하여 SQL로 FHIR 데이터 쿼리를 지원합니다. 자세한 내용은 Amazon Athena를 사용하여 HealthLake 데이터 쿼리를 참조하세요. • SQL 쿼리 기능은 기본적으로 11/14/2022 이후에 생성된 모든 HealthLake 데이터 스토어에서 활성화됩니다. 이 날짜 이전에 데이터 스토어가 생성된 경우 지원 티켓을 제출하여 SQL 쿼리 기능을 활성화할 수 있습니다. 를 사용하여 사례를 생성합니다AWS Support Center Console. 사례를 생성하려면 에 로그인 AWS 계정 하고 사례 생성을 선택합니다. • SQL 쿼리 기능을 사용하면 HealthLake에 액세스하기 위한 IAM 설정을 업데이트해야 합니다. Athena에서 HealthLake 데이터 스토어를 생성하고 액세스 권한을 부여하려면 `AWSLakeFormationDataAdmin` 관리형 정책을 IAM 사용자, 그룹 또는 역할에 추가해야 합니다. `AWSLakeFormationDataAdmin` 정책을 사용하여 데이터 레이크 관리자를 생성하고 Athena의 데이터 스토어에 대한 액세스 권한을 부여할	2022년 11월 14일

	수 있습니다. 자세한 내용은 IAM 사용자 또는 역할 구성을 참조하세요.	
총 가져오기 작업 크기 증가	HealthLake는 `StartFHIRImportJob` 요청에 `Total import job size` 대한를 500GB로 업데이트합니다. 자세한 내용은 Service Quotas을 참조하세요.	2022년 10월 3일
FHIR `Bundle` 리소스	HealthLake는 여러 FHIR `Bundle` 리소스를 동시에 처리하기 위해 FHIR 리소스 유형을 지원합니다. 자세한 내용은 FHIR 리소스 번들링을 참조하세요.	2022년 8월 5일
FHIR 상호 작용에 대한 할당량 업데이트	HealthLake는 FHIR 리소스 관리 상호 작용에 대한 할당량을 업데이트합니다. 자세한 내용은 Service Quotas을 참조하세요.	2022년 7월 16일
FHIR `_include` 검색 파라미터	HealthLake는 `search` 요청에 추가 리소스를 반환하기 위해 FHIR `_include` 검색 파라미터에 대한 지원을 추가합니다. 자세한 내용은 고급 검색 파라미터를 참조하세요.	2022년 7월 16일
AWS HealthLake 는 일반적으로 사용할 수 있습니다.	HealthLake는 일반적으로 지원되는 모든 리전에서 사용할 수 있습니다. 자세한 내용은 서비스 엔드포인트를 참조하세요.	2021년 7월 15일