aws

Panduan Developer

AWS HealthLake

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

AWS HealthLake: Panduan Developer

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

Merek dagang dan tampilan dagang Amazon tidak boleh digunakan sehubungan dengan produk atau layanan apa pun yang bukan milik Amazon, dengan cara apa pun yang dapat menyebabkan kebingungan di antara pelanggan, atau dengan cara apa pun yang merendahkan atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon merupakan hak milik masing-masing pemiliknya, yang mungkin atau mungkin tidak terafiliasi, terkait dengan, atau disponsori oleh Amazon.

Table of Contents

Apa itu AWS HealthLake? 1
Pemberitahuan penting 2
Fitur 2
Layanan terkait 3
Mengakses 4
HIPAA 4
Harga 5
Memulai 6
Konsep 6
strategi otorisasi 6
NLP terintegrasi 7
Analitik terintegrasi 7
Penyiapan 7
Mendaftar untuk Akun AWS 8
Konfigurasikan pengguna atau peran IAM 8
Menambahkan pengguna atau peran Administrator Data Lake 10
Buat ember S3 11
Buat penyimpanan data 12
Siapkan izin impor 12
Siapkan izin ekspor 15
Instal AWS CLI 19
Tutorial 20
Mengelola penyimpanan data 21
Membuat penyimpanan data 21
Mendapatkan properti penyimpanan data 29
Menyimpan data daftar 32
Menandai penyimpanan data 36
Menandai penyimpanan data 37
Listing tag untuk penyimpanan data 40
Membuka tag penyimpanan data 43
Menghapus penyimpanan data 46
Mengelola Langganan FHIR 51
Cara kerja Langganan FHIR 51
Komponen kunci 51

Topik Berlangganan 52
Langganan 52
Saluran pemberitahuan 53
Muatan pemberitahuan 53
Praktik terbaik 53
Siklus hidup berlangganan 54
Membuat Langganan 56
Contoh muatan Berlangganan 59
Contoh payload notifikasi 63
Mencari Langganan 68
Pemberitahuan penyaringan 71
Mengimpor data FHIR 75
Memulai pekerjaan impor 77
Mendapatkan properti pekerjaan impor 82
Daftar pekerjaan impor 86
Mengelola sumber daya FHIR 91
Membuat sebuah sumber daya 92
Membaca sumber daya 96
Membaca sejarah sumber daya 98
Membaca sejarah khusus versi 101
Memperbarui sumber daya 103
Pembaruan bersyarat 106
Mengkonfigurasi tingkat validasi untuk pembaruan sumber daya 106
Memodifikasi sumber daya 107
Format PATCH yang Didukung 108
Penggunaan 108
Format Patch JSON 109
FHIRPath Format Patch 111
Header Permintaan 113
Contoh Respons 113
Perilaku 114
Penanganan Kesalahan 115
Kesimpulan dari Capabilities 115
Batasan 115
Sumber Daya Tambahan 116
Sumber daya bundling 116

Bundel sebagai entitas independen 120
Bersyarat PUTs 124
Bundel sebagai entitas tunggal 128
Mengkonfigurasi tingkat validasi untuk bundel 131
Dukungan terbatas untuk “pesan” tipe Bundle 132
Transaksi async 134
Menghapus sumber daya 143
Hapus bersyarat untuk FHIR 145
Idempotensi dan Konkurensi 147
Kunci Idempotensi 147
ETag di AWS HealthLake 149
Mencari sumber daya FHIR 151
Mencari dengan GET 151
DAPATKAN contoh pencarian 154
Mencari dengan POST 155
Contoh pencarian POST 158
Tingkat Konsistensi Pencarian 160
Tingkat konsistensi 160
Contoh penggunaan 161
Praktik terbaik 162
Mengekspor data FHIR 163
Memulai pekerjaan ekspor 163
Mendapatkan properti pekerjaan ekspor 168
Daftar pekerjaan ekspor 172
Contoh kode 178
Hal-hal mendasar 178
Tindakan 179
Mengintegrasikan 226
Pemrosesan bahasa alami 226
Pustaka NLP 227
Menggunakan FHIR APIs 229
Parameter pencarian 229
Permintaan contoh 232
Indeks SQL dan kueri 248
Memulai 249
Query dengan SQL 253

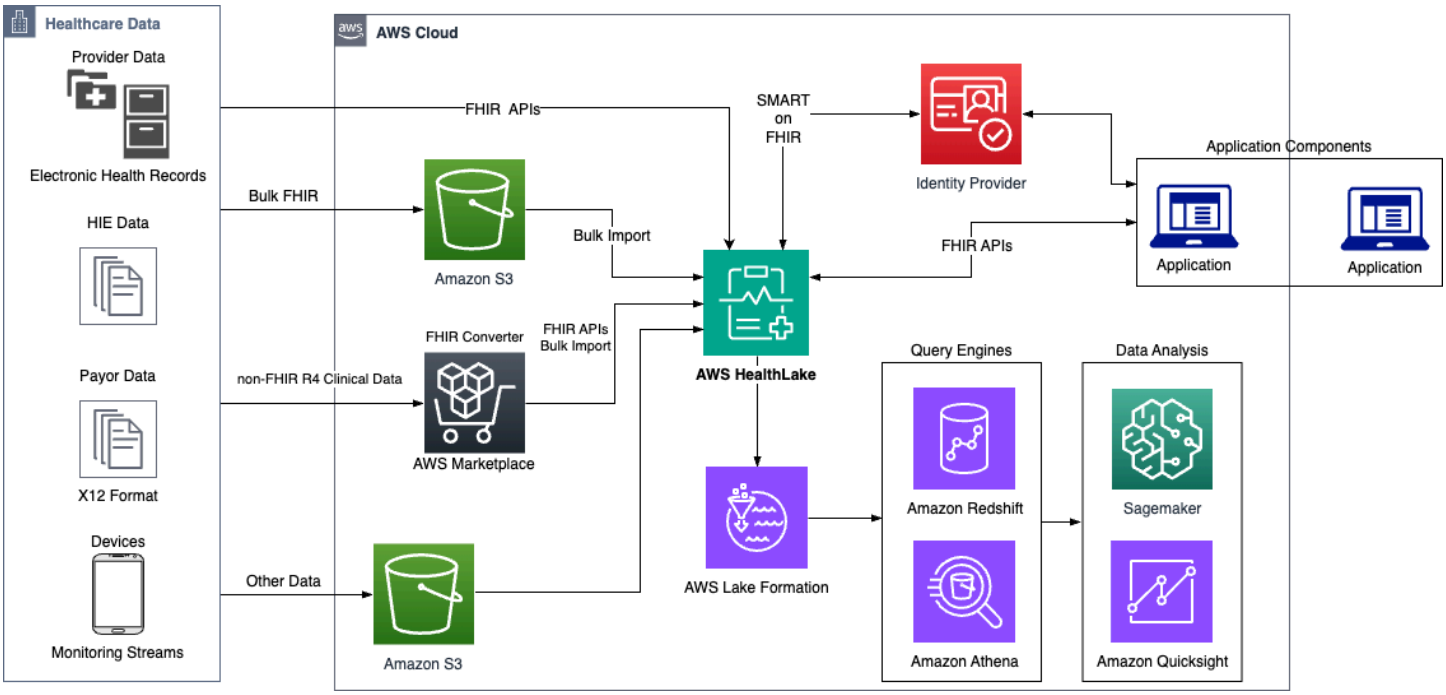
Kueri contoh 260
Memantau 267
CloudTrail (Panggilan API) 268
AWS HealthLake Informasi di CloudTrail 268
Memahami Entri File AWS HealthLake Log 270
CloudWatch (Metrik) 271
Melihat HealthLake metrik 274
Membuat alarm 274
EventBridge (Acara) 275
HealthLake peristiwa dikirim ke EventBridge 276
HealthLake struktur acara 277
Keamanan 291
Perlindungan Data 292
Enkripsi saat diam 293
Kunci KMS milik AWS 293
Kunci KMS yang dikelola pelanggan 293
Buat kunci terkelola pelanggan 294
Izin IAM yang diperlukan untuk menggunakan kunci KMS yang dikelola pelanggan 296
Enkripsi saat bergerak 303
Manajemen identitas dan akses 303
Audiens 303
Mengautentikasi dengan identitas 304
Mengelola akses menggunakan kebijakan 305
Bagaimana AWS HealthLake bekerja dengan IAM 307
Contoh kebijakan berbasis identitas 312
AWS kebijakan terkelola 316
Pemecahan masalah 320
Validasi kepatuhan 322
Keamanan infrastruktur 323
Infrastruktur sebagai kode 323
HealthLake dan CloudFormation template 323
Pelajari lebih lanjut tentang CloudFormation 324
Titik akhir VPC 324
Pertimbangan untuk titik akhir HealthLake VPC 325
Membuat titik akhir VPC antarmuka untuk; HealthLake 325
Membuat kebijakan titik akhir VPC untuk HealthLake 325

Praktik terbaik 326
Ketahanan 327
Referensi 328
SMART di FHIR 328
Memulai 329
Autentikasi 332
OAuth 2.0 cakupan 333
Validasi token 337
Otorisasi berbutir halus 349
Dokumen Penemuan 350
Minta contoh 351
FHIR R4 352
Pernyataan Kemampuan 353
Validasi profil 354
Jenis sumber daya 360
Parameter pencarian 362
$ Operasi 376
Kepatuhan 525
CMS 526
HealthLake 531
Titik akhir dan kuota 532
Tipe data yang dimuat sebelumnya 544
Contoh proyek 545
Pemecahan masalah 545
Bekerja dengan AWS SDKs 554
Rilis 556
..... dlxxxiii

Apa itu AWS HealthLake?

AWS HealthLake adalah layanan HIPAA yang memenuhi syarat untuk menyimpan, menganalisis, dan berbagi data kesehatan di cloud menggunakan spesifikasi Fast Healthcare Interoperability Resources (FHIR) R4. HealthLake kasus penggunaan meliputi:

- Data kesehatan perusahaan — Kelola dan bagikan data kesehatan FHIR R4 secara langsung dari AWS Cloud sambil menjaga kinerja dan ketersediaan tinggi.
- Interoperabilitas layanan kesehatan - Mendukung kesesuaian pelanggan dengan Undang-Undang Penyembuhan Abad ke-21 untuk akses pasien melalui penyimpanan data FHIR yang dikelola sepenuhnya.
- Natural Language Processing (NLP) — Memanfaatkan model NLP terintegrasi untuk mengekstrak informasi medis yang bermakna dari data kesehatan yang tidak terstruktur.
- Analisis multimodal — Menggabungkan HealthLake data dengan AWS HealthImaging data dan AWS HealthOmics data untuk memberikan wawasan untuk pengobatan presisi.



Topik

- Pemberitahuan penting
- Fitur dari AWS HealthLake

- AWS Layanan terkait
- Mengakses AWS HealthLake
- Kelayakan HIPAA dan keamanan data
- Harga

Pemberitahuan penting

AWS HealthLake bukan pengganti nasihat medis profesional, diagnosis, atau pengobatan, dan tidak dimaksudkan untuk menyembuhkan, mengobati, mengurangi, mencegah, atau mendiagnosis penyakit atau kondisi kesehatan apa pun. Anda bertanggung jawab untuk melembagakan tinjauan manusia sebagai bagian dari penggunaan apa pun AWS HealthLake, termasuk terkait dengan produk pihak ketiga yang dimaksudkan untuk menginformasikan pengambilan keputusan klinis. AWS HealthLake hanya boleh digunakan dalam perawatan pasien atau skenario klinis setelah ditinjau oleh profesional medis terlatih yang menerapkan penilaian medis yang baik.

Fitur dari AWS HealthLake

AWS HealthLake menyediakan fitur-fitur berikut.

Impor data kesehatan FHIR R4

> Dengan tindakan impor HealthLake asli, Anda dapat dengan mudah memigrasikan data FHIR dari bucket Amazon S3 ke HealthLake penyimpanan data, termasuk catatan klinis, laporan lab, klaim asuransi, dan banyak lagi. HealthLake mendukung spesifikasi FHIR R4 untuk pertukaran data perawatan kesehatan. Jika diperlukan, Anda dapat bekerja dengan AWS HealthLake Mitra untuk mengonversi data kesehatan Anda ke format FHIR R4.

Menyimpan data kesehatan dengan cara yang aman, patuh, dan dapat diaudit

> Penyimpanan HealthLake data membantu mengindeks data kesehatan sehingga dapat ditanyakan. Penyimpanan data menciptakan tampilan lengkap riwayat medis setiap pasien dalam urutan kronologis dan memfasilitasi pertukaran informasi menggunakan spesifikasi FHIR R4. Dan selalu berjalan untuk menjaga indeks Anda tetap up to date, menawarkan Anda kemampuan untuk mencari informasi kapan saja menggunakan interaksi FHIR R4 standar dengan penyimpanan primer yang tahan lama dan penskalaan indeks.

Memanfaatkan server FHIR transaksional

Manfaatkan FHIR APIs untuk validasi sumber daya standar, SMART pada otorisasi FHIR, dan kemampuan ekspor FHIR API data massal untuk mendukung penyatuan dan analisis data Anda guna mengurangi biaya operasional dan meningkatkan pengambilan keputusan. HealthLake mendukung kesesuaian pelanggan dengan standar peraturan ONC dan CMS terbaru termasuk: HL7 FHIR R4 APIs, Akses Data Massal FHIR, US Core IG STU, HL7 SMART App Launch Framework IG, 2.0, dan OpenID Connect. OAuth

Mengubah data medis tidak terstruktur menggunakan NLP

Pemrosesan bahasa alami medis terintegrasi (NLP) mengubah semua data teks medis mentah di penyimpanan data untuk memahami dan mengekstrak informasi yang bermakna dari HealthLake data perawatan kesehatan yang tidak terstruktur. Dengan NLP medis terintegrasi, Anda dapat secara otomatis mengekstrak entitas, hubungan entitas, sifat entitas, dan informasi kesehatan yang dilindungi (PHI) dari teks medis Anda. Entitas yang diekstrak NLP disimpan sebagai sumber daya FHIR R4 asli dalam penyimpanan HealthLake data dan dapat diakses melalui FHIR R4 atau APIs Amazon Athena (SQL).

AWS Layanan terkait

AWS HealthLake fitur integrasi yang ketat dengan AWS layanan lain. Pengetahuan tentang layanan berikut berguna untuk memanfaatkan sepenuhnya HealthLake.

- AWS Identity and Access Management— Gunakan IAM untuk mengelola identitas dan akses ke sumber daya dengan aman. HealthLake
- Amazon Simple Storage Service — Gunakan Amazon S3 sebagai area pementasan untuk mengimpor data DICOM ke dalamnya. HealthLake
- AWS CloudTrail— Gunakan CloudTrail untuk melacak aktivitas HealthLake pengguna dan penggunaan API.
- Amazon CloudWatch — Gunakan CloudWatch untuk mengamati dan memantau HealthLake sumber daya.
- AWS CloudFormation— Gunakan CloudFormation untuk mengimplementasikan templat infrastruktur sebagai kode (IAc) untuk membuat sumber daya di HealthLake.
- AWS PrivateLink— Gunakan Amazon VPC untuk membangun konektivitas antara HealthLake dan Amazon Virtual Private Cloud tanpa mengekspos data ke internet.

- Amazon EventBridge — Gunakan EventBridge untuk membuat aplikasi yang dapat diskalakan dan didorong oleh peristiwa dengan membuat aturan yang merutekan HealthLake peristiwa ke target.
- AWS Lake Formation— Gunakan Lake Formation untuk mengatur, mengamankan, dan berbagi HealthLake data secara terpusat untuk analitik dan pembelajaran mesin.
- Amazon Athena — Gunakan Athena untuk menanyakan HealthLake data dengan SQL untuk memungkinkan analisis yang lebih dalam.

Mengakses AWS HealthLake

Anda dapat mengakses AWS HealthLake menggunakan Konsol Manajemen AWS, AWS Command Line Interface dan AWS SDKs. Panduan ini memberikan instruksi prosedural untuk contoh Konsol Manajemen AWS dan kode untuk AWS CLI dan AWS SDKs.

AWS Command Line Interface (AWS CLI)

AWS CLI Ini menyediakan perintah untuk serangkaian AWS produk yang luas, dan didukung di Windows, Mac, dan Linux. Untuk informasi selengkapnya, silakan lihat Panduan Pengguna AWS Command Line Interface .

AWS SDKs

AWS SDKs menyediakan perpustakaan, contoh kode, dan sumber daya lainnya untuk pengembang perangkat lunak. Pustaka ini menyediakan fungsi dasar yang mengotomatiskan tugas seperti menandatangani permintaan Anda secara kriptografis, mencoba ulang permintaan, dan menangani respons kesalahan. Untuk informasi selengkapnya, lihat Alat untuk Dibangun AWS.

Konsol Manajemen AWS

Konsol Manajemen AWS Ini menyediakan antarmuka pengguna berbasis web untuk mengelola HealthLake dan sumber daya yang terkait. Jika Anda telah mendaftar untuk sebuah AWS akun, Anda dapat masuk ke HealthLake Konsol.

Kelayakan HIPAA dan keamanan data

Ini adalah Layanan yang Memenuhi Syarat HIPAA. Untuk informasi lebih lanjut tentang AWS, Undang-Undang Portabilitas dan Akuntabilitas Asuransi Kesehatan AS tahun 1996 (HIPAA), dan menggunakan AWS layanan untuk memproses, menyimpan, dan mengirimkan informasi kesehatan yang dilindungi (PHI), lihat Ikhtisar HIPAA.

Koneksi untuk HealthLake berisi PHI dan informasi identitas pribadi (PII) harus dienkripsi. Secara default, semua koneksi HealthLake menggunakan HTTPS melalui TLS. HealthLake menyimpan konten pelanggan terenkripsi dan beroperasi sesuai dengan Model Tanggung Jawab AWS Bersama.

Harga

Untuk informasi HealthLake harga, lihat AWS HealthLake harga. Untuk memperkirakan biaya, gunakan kalkulator HealthLake harga.

Memulai dengan AWS HealthLake

Untuk mulai menggunakan AWS HealthLake, siapkan AWS akun dan buat AWS Identity and Access Management pengguna. Untuk menggunakan AWS CLIatau AWS SDKs, Anda harus menginstal dan mengkonfigurasinya.

> **Note**
>
> ReferensiBab panduan ini menyediakan konten pendukung untuk SMART di FHIR, FHIR R4, dan. AWS HealthLake Misalnya, Anda dapat menemukan informasi tentang SMART pada konfigurasi FHIR, validasi profil FHIR yang didukung, dan titik akhir. HealthLake

Setelah mempelajari HealthLake konsep dan pengaturan, tutorial singkat dengan contoh kode tersedia untuk membantu Anda memulai.

Topik

- AWS HealthLake konsep
- Menyiapkan AWS HealthLake
- AWS HealthLake tutorial

AWS HealthLake konsep

Terminologi dan konsep berikut sangat penting untuk pemahaman dan penggunaan AWS HealthLake Anda.

Konsep

- Strategi otorisasi penyimpanan data
- NLP terintegrasi
- Analitik terintegrasi

Strategi otorisasi penyimpanan data

Penyimpanan HealthLake data adalah repositori data kesehatan FHIR R4 yang berada dalam satu. AWS Region HealthLake mendukung strategi otorisasi penyimpanan data berikut.

- Otorisasi SIgV4 - mengotorisasi panggilan API FHIR menggunakan HealthLake otorisasi Sigv4 Versi AWS Tanda Tangan 4 (SigV4).
- SMART pada otorisasi FHIR - HealthLake mengotorisasi panggilan API FHIR menggunakan Aplikasi Medis yang Dapat Diganti dan Teknologi yang Dapat Digunakan Kembali (SMART) pada otorisasi FHIR.

Untuk informasi selengkapnya, lihat Membuat penyimpanan HealthLake data.

NLP terintegrasi

AWS HealthLake terintegrasi dengan perpustakaan pemrosesan bahasa alami (NLP) yang memenuhi syarat HIPAA untuk mengekstrak data kesehatan yang bermakna dari teks medis yang tidak terstruktur. Perpustakaan NLP mengidentifikasi entitas medis seperti kondisi, obat, dosis, tes, perawatan, dan prosedur. Mereka mengenali hubungan di antara entitas dan menghubungkannya ke perpustakaan ontologi medis seperti ICD-10-CM dan. RxNorm Untuk informasi selengkapnya, lihat Pemrosesan bahasa alami terintegrasi (NLP) untuk HealthLake.

Analitik terintegrasi

AWS HealthLake melampaui FHIR `search` dan `bundle` APIs menyediakan analitik terintegrasi untuk menanyakan dan menganalisis data kesehatan dalam jumlah besar. Selama impor, HealthLake secara otomatis menghasilkan tabel untuk indeks SQL dan kueri. Ini memungkinkan Anda untuk mendapatkan wawasan yang dapat ditindaklanjuti dari data perawatan kesehatan yang kompleks tanpa memerlukan pekerjaan rekayasa data yang ekstensif. Lihat informasi yang lebih lengkap di Meminta HealthLake data dengan Amazon Athena dan AWS HealthLake proyek sampel.

Menyiapkan AWS HealthLake

Dalam Bab ini, Anda menggunakan Konsol Manajemen AWS untuk mengatur izin yang diperlukan untuk mulai menggunakan AWS HealthLake dan membuat penyimpanan data. Untuk mengatur izin untuk membuat penyimpanan data, Anda membuat pengguna IAM atau peran yang merupakan administrator dan HealthLake administrator data lake. Anda menjadikan pengguna ini administrator danau data di AWS Lake Formation. Administrator data lake memberikan Lake Formation akses ke sumber daya yang diperlukan untuk menggunakan Amazon Athena untuk menanyakan penyimpanan data. Setelah membuat penyimpanan HealthLake data, Anda dapat mengatur izin untuk mengimpor dan mengekspor file.

Topik

- Mendaftar untuk Akun AWS
- Konfigurasikan pengguna IAM atau peran yang akan digunakan HealthLake (Administrator IAM)
- Tambahkan pengguna atau peran sebagai Administrator Data Lake di Lake Formation (IAM Administrator)
- Buat ember S3
- Buat penyimpanan data
- Menyiapkan izin untuk pekerjaan impor
- Menyiapkan izin untuk pekerjaan ekspor
- Instal AWS CLI

Mendaftar untuk Akun AWS

Untuk memulai AWS, Anda membutuhkan Akun AWS. Untuk informasi tentang membuat Akun AWS, lihat Memulai dengan Akun AWS di Panduan AWS Account Management Referensi.

Konfigurasikan pengguna IAM atau peran yang akan digunakan HealthLake (Administrator IAM)

> **Persona: IAM Administrator**
>
> Pengguna yang dapat membuat pengguna dan peran IAM, dan dapat menambahkan administrator data lake.

Langkah-langkah dalam topik ini harus dilakukan oleh administrator IAM.

Untuk menghubungkan penyimpanan HealthLake data Anda ke Athena, Anda perlu membuat pengguna IAM atau peran yang merupakan administrator data lake dan administrator. HealthLake Pengguna atau peran baru ini memberikan akses ke sumber daya yang ditemukan di penyimpanan data melalui AWS Lake Formation, dan kebijakan `AmazonHealthLakeFullAccess` AWS terkelola ditambahkan ke pengguna atau peran mereka.

> **Important**
>
> Pengguna IAM atau peran yang merupakan administrator danau data tidak dapat membuat administrator danau data baru. Untuk menambahkan administrator data lake

tambahan Anda harus menggunakan pengguna IAM atau peran yang telah diberikan `AdministratorAccess` akses.

Untuk membuat administrator

1. Tambahkan kebijakan AWS terkelola **`AmazonHealthlakeFullAccess`** IAM ke pengguna atau peran di organisasi Anda.

   Jika Anda tidak terbiasa membuat pengguna IAM, lihat Membuat Pengguna IAM dan Ikhtisar Kebijakan IAM di Panduan Pengguna AWS IAM.

2. Berikan pengguna IAM atau akses peran ke AWS Lake Formation.

   - Tambahkan kebijakan AWS terkelola IAM berikut ke pengguna atau peran di organisasi Anda: **`AWSLakeFormationDataAdmin`**

     > **Note**
     >
     > `AWSLakeFormationDataAdmin`Kebijakan tersebut memberikan akses ke semua sumber daya AWS Lake Formation. Sebaiknya Anda selalu menggunakan izin minimum yang diperlukan untuk menyelesaikan tugas Anda. Untuk informasi selengkapnya, lihat Praktik Terbaik IAM dalam Panduan Pengguna IAM.

3. Tambahkan kebijakan inline berikut ke pengguna atau peran. Untuk informasi selengkapnya, lihat Kebijakan sebaris di Panduan Pengguna IAM.

   JSON

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

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

Untuk informasi lebih lanjut tentang `AWSLakeFormationDataAdmin` kebijakan ini, lihat Referensi Izin Personas Lake Formation dan IAM di Panduan Pengembang AWS Lake Formation.

Tambahkan pengguna atau peran sebagai Administrator Data Lake di Lake Formation (IAM Administrator)

> **Note**
>
> Langkah ini diperlukan jika Anda mengintegrasikan. Indeks SQL dan kueri

Selanjutnya, administrator IAM harus menambahkan pengguna atau peran yang dibuat pada langkah sebelumnya sebagai administrator danau data di Lake Formation.

Untuk menambahkan pengguna IAM atau peran sebagai administrator danau data

1. Buka konsol AWS Lake Formation: https://console.aws.amazon.com/lakeformation/

> **Note**
>
> Jika ini adalah pertama kalinya Anda mengunjungi Lake Formation, kotak dialog Selamat Datang di Lake Formation muncul meminta Anda untuk menentukan administrator Lake Formation.

2. Tetapkan pengguna atau peran baru untuk menjadi administrator danau data AWS Lake Formation.
   - Opsi 1: Jika Anda menerima kotak dialog Selamat Datang di Lake Formation.
     1. Pilih Tambahkan AWS pengguna atau peran lain.
     2. Pilih panah bawah (▼).
     3. Pilih HealthLake administrator yang Anda inginkan juga menjadi administrator Lake Formation.
     4. Pilih Mulai.
   - Opsi 2: Gunakan panel Navigasi (☰).
     1. Pilih panel Navigasi (☰).
     2. Di bawah Izin, pilih Peran dan tugas administratif.
     3. Di bagian Administrator danau data, pilih Pilih administrator.
     4. Di kotak dialog Kelola data lake administrator, pilih panah bawah (▼).
     5. Selanjutnya, pilih atau cari HealthLake administrator pengguna atau peran yang Anda juga ingin menjadi administrator Lake Formation.
     6. Pilih Simpan.
3. Ubah pengaturan keamanan default untuk dikelola oleh Lake Formation. Sumber daya penyimpanan HealthLake data perlu dikelola oleh Lake Formation bukan IAM. Untuk memperbarui, lihat Mengubah model izin default di Panduan Pengembang AWS Lake Formation.

Buat ember S3

Untuk mengimpor data FHIR R4 ke dalam AWS HealthLake, dua bucket Amazon S3 direkomendasikan. Bucket input Amazon S3 menyimpan data FHIR untuk diimpor dan HealthLake dibaca dari bucket ini. Bucket keluaran Amazon S3 menyimpan hasil pemrosesan pekerjaan impor dan HealthLake menulis (log) ke bucket ini.

> **Note**
>
> Karena kebijakan AWS Identity and Access Management (IAM), nama bucket Amazon S3 Anda harus unik. Untuk informasi selengkapnya, lihat Aturan penamaan bucket di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Untuk tujuan panduan ini, kami menentukan bucket input dan output Amazon S3 berikut saat menyiapkan izin impor nanti di bagian ini.

- Ember masukan: `arn:aws:s3:::amzn-s3-demo-source-bucket`
- Ember keluaran: `arn:aws:s3:::amzn-s3-demo-logging-bucket`

Untuk informasi tambahan, lihat Membuat bucket di Panduan Pengguna Amazon S3.

Buat penyimpanan data

Penyimpanan HealthLake data adalah repositori data FHIR R4 yang berada dalam satu Wilayah. AWS Sebuah AWS akun dapat memiliki nol atau banyak penyimpanan data. HealthLake mendukung dua strategi otorisasi penyimpanan data.

> **Penting:**
>
> Sebelum membuat penyimpanan HealthLake data, tinjau kebijakan kontrol Layanan (SCP) di AWS Organisasi Anda yang mungkin membatasi pembuatan atau pengelolaan sumber daya HealthLake . SCP dapat mencegah keberhasilan pembuatan penyimpanan HealthLake data, bahkan jika izin IAM Anda diatur dengan benar.
> A `datastoreID` dihasilkan saat Anda membuat penyimpanan HealthLake data. Anda harus menggunakan `datastoreID` saat mengatur izin impor nanti di bagian ini.

Untuk membuat penyimpanan HealthLake data, lihatMembuat penyimpanan HealthLake data.

Menyiapkan izin untuk pekerjaan impor

Sebelum mengimpor file ke penyimpanan data, Anda harus memberikan HealthLake izin untuk mengakses bucket input dan output di Amazon S3. Untuk memberikan HealthLake akses, Anda

membuat peran layanan IAM HealthLake, tambahkan kebijakan kepercayaan ke peran untuk memberikan izin peran HealthLake asumsi, dan melampirkan kebijakan izin ke peran yang memberinya akses ke bucket Amazon S3 Anda.

Saat Anda membuat pekerjaan impor, Anda menentukan Nama Sumber Daya Amazon (ARN) peran ini untuk. `DataAccessRoleArn` Untuk informasi selengkapnya tentang peran IAM dan kebijakan kepercayaan, lihat Peran IAM.

Setelah Anda mengatur izin, Anda siap untuk mengimpor file ke penyimpanan data Anda dengan pekerjaan impor. Untuk informasi selengkapnya, lihat Memulai pekerjaan impor FHIR.

Untuk mengatur izin impor

1. Jika belum, buat bucket Amazon S3 tujuan untuk file log keluaran. Bucket Amazon S3 harus berada di AWS Wilayah yang sama dengan layanan, dan Blokir Akses Publik harus diaktifkan untuk semua opsi. Untuk mempelajari selengkapnya, lihat Menggunakan Amazon S3 memblokir akses publik. Kunci KMS Amazon-owned atau milik pelanggan juga harus digunakan untuk enkripsi. Untuk mempelajari selengkapnya tentang menggunakan kunci KMS, lihat Amazon Key Management Service.
2. Buat peran layanan akses data untuk HealthLake dan berikan izin HealthLake layanan untuk menganggapnya dengan kebijakan kepercayaan berikut. HealthLake menggunakan ini untuk menulis output ember Amazon S3.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "healthlake.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "accountID"
                },
```

```
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:healthlake:us-
west-2:111122223333:datastore/fhir/datastoreID"
                }
            }
        }
    ]
}
```

3. Tambahkan kebijakan izin ke peran akses data yang memungkinkannya mengakses bucket Amazon S3. Ganti `amzn-s3-demo-bucket` dengan nama bucket Anda.

   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"
    }]
}
```

Menyiapkan izin untuk pekerjaan ekspor

Sebelum mengekspor file dari penyimpanan data, Anda harus memberikan HealthLake izin untuk mengakses bucket keluaran di Amazon S3. Untuk memberikan HealthLake akses, Anda membuat peran layanan IAMHealthLake, tambahkan kebijakan kepercayaan ke peran untuk memberikan izin peran HealthLake asumsi, dan melampirkan kebijakan izin ke peran yang memberinya akses ke bucket Amazon S3 Anda.

Jika Anda sudah membuat peran HealthLake, Anda dapat menggunakannya kembali dan memberinya izin tambahan untuk bucket ekspor Amazon S3 yang tercantum dalam topik ini. Untuk mempelajari lebih lanjut tentang peran IAM dan kebijakan kepercayaan, lihat Kebijakan dan Izin IAM.

Penting:

HealthLake mendukung permintaan ekspor SDK asli dan operasi FHIR `$export` R4. Tindakan IAM terpisah harus disediakan tergantung pada API ekspor mana yang Anda putuskan untuk digunakan. Ini memungkinkan Anda untuk menangani `allow` dan `deny` mengizinkan secara terpisah. Jika Anda ingin membatasi ekspor HealthLake SDK dan FHIR REST API, Anda harus menerapkan izin penolakan ke tindakan IAM terpisah. Perubahan izin pengguna IAM tidak diperlukan jika Anda memberi pengguna akses HealthLake penuh.

Penggunaan AWS CLI and AWS SDK:

HealthLake Tindakan native berikut tersedia untuk mengekspor data dari penyimpanan data menggunakan AWS CLI dan AWS SDK:

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

Menggunakan API FHIR:

Tindakan IAM berikut tersedia untuk mengekspor data dari penyimpanan HealthLake data dan untuk membatalkan (menghapus) pekerjaan ekspor menggunakan operasi FHIR: `$export`
`POST`:

- `StartFHIRExportJobWithPost`

`GET`:

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

`DELETE`:

- `CancelFHIRExportJobWithDelete`

Pengguna atau peran yang menyiapkan izin harus memiliki izin untuk membuat peran, membuat kebijakan, dan melampirkan kebijakan ke peran. Kebijakan IAM berikut memberikan izin ini.

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"
                }
            }
        }
    ]
}
```

Untuk mengatur izin ekspor

1. Jika belum, buat bucket Amazon S3 tujuan untuk data yang akan Anda ekspor dari penyimpanan data Anda. Bucket Amazon S3 harus berada di Wilayah AWS yang sama dengan layanan, dan Blokir Akses Publik harus diaktifkan untuk semua opsi. Untuk mempelajari selengkapnya, lihat Menggunakan Amazon S3 memblokir akses publik. Kunci KMS Amazon-owned atau milik pelanggan juga harus digunakan untuk enkripsi. Untuk mempelajari selengkapnya tentang menggunakan kunci KMS, lihat Amazon Key Management Service.
2. Jika Anda belum melakukannya, buat peran layanan akses data untuk HealthLake dan berikan izin HealthLake layanan untuk menganggapnya dengan kebijakan kepercayaan berikut. HealthLakemenggunakan ini untuk menulis output ember Amazon S3. Jika Anda sudah membuatnyaMenyiapkan izin untuk pekerjaan impor, Anda dapat menggunakannya kembali dan memberinya izin untuk bucket Amazon S3 Anda di langkah berikutnya.

   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. Tambahkan kebijakan izin ke peran akses data yang memungkinkannya mengakses bucket Amazon S3 keluaran Anda. Ganti `amzn-s3-demo-bucket` dengan nama bucket Anda.

   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"
    }]
}
```

Instal AWS CLI

AWS CLI Diperlukan untuk mendeskripsikan dan mencantumkan properti pekerjaan HealthLake impor dan ekspor. Anda juga dapat meminta informasi ini menggunakan HealthLake SDK.

Untuk mengatur AWS CLI

1. Unduh dan konfigurasikan AWS CLI. Untuk instruksi, lihat topik berikut di AWS Command Line Interface Panduan Pengguna.

   - Menginstal atau memperbarui versi terbaru dari AWS CLI
   - Memulai dengan AWS CLI

2. Dalam AWS CLI `config` file, tambahkan profil bernama untuk administrator. Anda menggunakan profil ini saat menjalankan AWS CLI perintah. Di bawah prinsip keamanan dengan hak istimewa terkecil, kami sarankan Anda membuat peran IAM terpisah dengan hak istimewa khusus untuk tugas yang sedang dilakukan. Untuk informasi selengkapnya tentang profil bernama, lihat Konfigurasi dan pengaturan file kredenal di Panduan AWS Command Line Interface Pengguna.

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

3. Verifikasi pengaturan menggunakan `help` perintah berikut.

   ```
   aws healthlake help
   ```

   Jika AWS CLI dikonfigurasi dengan benar, Anda melihat deskripsi singkat AWS HealthLake dan daftar perintah yang tersedia.

AWS HealthLake tutorial

Objektif

Dalam tutorial ini, Anda akan mengimpor data FHIR R4 ke penyimpanan HealthLake data menggunakan tindakan asli HealthLake. Selanjutnya, Anda akan mengelola (membuat, membaca, memperbarui, menghapus) sumber daya FHIR menggunakan RESTful APIs FHIR. Untuk menyimpulkan tutorial, Anda akan mengekspor data FHIR menggunakan HealthLake tindakan asli.

Prasyarat

Semua prosedur yang Penyiapan tercantum dalam diperlukan untuk menyelesaikan tutorial ini.

Langkah-langkah tutorial

1. Mulai pekerjaan impor FHIR
2. Dapatkan properti pekerjaan impor FHIR
3. Buat sumber daya FHIR
4. Baca sumber daya FHIR
5. Perbarui sumber daya FHIR
6. Hapus sumber daya FHIR
7. Ekspor data FHIR
8. Hapus penyimpanan data

Mengelola penyimpanan data dengan AWS HealthLake

Dengan AWS HealthLake, Anda membuat dan mengelola penyimpanan data untuk sumber daya FHIR R4. Saat Anda membuat penyimpanan HealthLake data, respositori data FHIR tersedia melalui titik akhir API. RESTful Anda dapat memilih untuk mengimpor (preload) Synthea open source data kesehatan FHIR R4 ke penyimpanan data Anda saat Anda membuatnya. Untuk informasi selengkapnya, lihat Tipe data yang dimuat sebelumnya.

> **Penting:**
>
> HealthLake mendukung dua jenis strategi otorisasi penyimpanan data FHIR, AWS SiGv4 atau SMART di FHIR. Anda harus memilih salah satu strategi otorisasi sebelum membuat penyimpanan data HealthLake FHIR. Untuk informasi selengkapnya, lihat Strategi otorisasi penyimpanan data.

Untuk menemukan kemampuan (perilaku) terkait fHIR dari penyimpanan HealthLake data aktif, ambil Pernyataan Kemampuannya.

Topik berikut menjelaskan cara menggunakan tindakan HealthLake cloud native untuk membuat, mendeskripsikan, mencantumkan, menandai, dan menghapus penyimpanan data FHIR menggunakan AWS CLI, AWS SDKs, dan Konsol Manajemen AWS.

Topik

- Membuat penyimpanan HealthLake data
- Mendapatkan properti penyimpanan HealthLake data
- Menyimpan HealthLake data daftar
- Menandai penyimpanan HealthLake data
- Menghapus penyimpanan HealthLake data

Membuat penyimpanan HealthLake data

Gunakan `CreateFHIRDatastore` untuk membuat kesesuaian penyimpanan AWS HealthLake data dengan spesifikasi FHIR R4. HealthLake penyimpanan data digunakan untuk mengimpor, mengelola, mencari, dan mengekspor data FHIR. Anda dapat memilih untuk mengimpor (preload) Synthea

open source data kesehatan FHIR R4 ke penyimpanan data Anda saat Anda membuatnya. Untuk informasi selengkapnya, lihat Tipe data yang dimuat sebelumnya.

> **Penting:**
>
> HealthLake mendukung dua jenis strategi otorisasi penyimpanan data FHIR, AWS SiGv4 atau SMART di FHIR. Anda harus memilih salah satu strategi otorisasi sebelum membuat penyimpanan data HealthLake FHIR. Untuk informasi selengkapnya, lihat Strategi otorisasi penyimpanan data.

Saat Anda membuat penyimpanan HealthLake data, repositori data FHIR tersedia melalui titik akhir API. RESTful Setelah membuat penyimpanan HealthLake data, Anda dapat meminta Pernyataan Kemampuannya untuk menemukan semua kemampuan (perilaku) terkait FHIR terkait.

Menu berikut memberikan contoh untuk AWS CLI dan AWS SDKs dan prosedur untuk. Konsol Manajemen AWS Untuk informasi selengkapnya, lihat `CreateFHIRDatastore` di dalam Referensi API AWS HealthLake .

Untuk membuat penyimpanan HealthLake data

Pilih menu berdasarkan preferensi akses Anda AWS HealthLake.

AWS CLI dan SDKs

CLI

AWS CLI

Contoh 1: Buat penyimpanan data berkemampuan SIGV4 HealthLake

`create-fhir-datastore`Contoh berikut menunjukkan cara membuat penyimpanan data baru di AWS HealthLake.

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

Output:

```
{
```

```
    "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)"
}
```

Contoh 2: Buat SMART di penyimpanan data berkemampuan FHIR HealthLake

`create-fhir-datastore`Contoh berikut menunjukkan cara membuat SMART baru di penyimpanan data berkemampuan FHIR di. AWS HealthLake

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

Isi dari `identity_provider_configuration.json`:

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

```
}
```

Output:

```
{
    "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)"
}
```

- Untuk detail API, lihat Membuat FHIRDatastore di Referensi AWS CLI Perintah.

Python

SDK untuk 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
```

Kode berikut menunjukkan contoh parameter untuk SMART pada penyimpanan data berkemampuan FHIR HealthLake .

```
            sse_configuration = {
                "KmsEncryptionConfig": {"CmkType": "AWS_OWNED_KMS_KEY"}
```

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

- Untuk detail API, lihat Membuat FHIRDatastore di AWS SDK for Python (Boto3) Referensi API.

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

- Untuk detail API, lihat Create FHIRDatastore in AWS SDK untuk referensi SAP ABAP API.

> **Ketersediaan contoh**
>
> Tidak dapat menemukan apa yang Anda butuhkan? Minta contoh kode menggunakan tautan Berikan umpan balik di bilah sisi kanan halaman ini.

AWS Konsol

> **Catatan**
>
> Prosedur berikut membuat penyimpanan HealthLake data dengan otorisasi AWS SiGv4. HealthLake Konsol tidak mendukung pembuatan SMART di penyimpanan data FHIR.

Untuk membuat penyimpanan HealthLake data dengan otorisasi AWS SiGv4

1. Masuk ke halaman Buat penyimpanan data di HealthLake Konsol.
2. Pilih Buat Penyimpanan Data.
3. Di bagian Pengaturan Penyimpanan Data, untuk nama Penyimpanan Data, tentukan nama.
4. (Opsional) Di bagian Pengaturan Penyimpanan Data, untuk Memuat data sampel, pilih kotak centang untuk memuat data Synthea. Synthea data adalah kumpulan data sampel sumber terbuka. Untuk informasi selengkapnya, lihat Synthea tipe data yang dimuat sebelumnya untuk HealthLake.
5. Di bagian enkripsi Penyimpanan Data, pilih Gunakan kunci yang dimiliki AWS (default) atau Pilih kunci AWS KMS yang berbeda (lanjutan).
6. Di bagian Tags - opsional, Anda dapat menambahkan tag ke penyimpanan data Anda. Untuk mempelajari lebih lanjut tentang menandai penyimpanan data Anda, lihatMenandai penyimpanan HealthLake data.
7. Pilih Buat Penyimpanan Data.

   Status penyimpanan data Anda tersedia di halaman Penyimpanan data.

Mendapatkan properti penyimpanan HealthLake data

Gunakan `DescribeFHIRDatastore` untuk mendapatkan properti untuk penyimpanan AWS HealthLake data. Menu berikut memberikan prosedur untuk contoh Konsol Manajemen AWS dan kode untuk AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat `DescribeFHIRDatastore` di dalam Referensi API AWS HealthLake .

Untuk mendapatkan properti untuk penyimpanan HealthLake data

Pilih menu berdasarkan preferensi akses Anda AWS HealthLake.

AWS CLI dan SDKs

CLI

AWS CLI

Untuk menggambarkan penyimpanan data FHIR

`describe-fhir-datastore`Contoh berikut menunjukkan bagaimana menemukan properti penyimpanan data di AWS HealthLake.

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

Output:

```
{
    "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
        }
    }
}
```

- Untuk detail API, lihat Menjelaskan FHIRDatastore di Referensi AWS CLI Perintah.

Python

SDK untuk 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
```

- Untuk detail API, lihat Menjelaskan FHIRDatastore dalam AWS SDK for Python (Boto3) Referensi API.

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

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

- Untuk detail API, lihat Menjelaskan FHIRDatastore di AWS SDK untuk referensi API SAP ABAP.

> **Ketersediaan contoh**
>
> Tidak dapat menemukan apa yang Anda butuhkan? Minta contoh kode menggunakan tautan Berikan umpan balik di bilah sisi kanan halaman ini.

AWS Konsol

1. Masuk ke halaman Penyimpanan data di HealthLake Konsol.
2. Pilih penyimpanan data.

   Halaman detail Penyimpanan Data terbuka dan semua properti penyimpanan HealthLake data tersedia.

Menyimpan HealthLake data daftar

Gunakan `ListFHIRDatastores` untuk mencantumkan semua penyimpanan HealthLake data di akun pengguna, terlepas dari status penyimpanan data. Menu berikut memberikan prosedur untuk contoh Konsol Manajemen AWS dan kode untuk AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat `ListFHIRDatastores` di dalam Referensi API AWS HealthLake .

Untuk daftar semua penyimpanan HealthLake data

Pilih menu berdasarkan preferensi akses Anda AWS HealthLake.

AWS CLI dan SDKs

CLI

AWS CLI

Untuk daftar toko data FHIR

`list-fhir-datastores`Contoh berikut menunjukkan cara menggunakan perintah dan bagaimana pengguna dapat memfilter hasil berdasarkan status penyimpanan data di AWS HealthLake.

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

Output:

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

- Untuk detail API, lihat Daftar FHIRDatastores di Referensi AWS CLI Perintah.

Python

SDK untuk 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
```

- Untuk detail API, lihat Daftar FHIRDatastores di AWS SDK for Python (Boto3) Referensi API.

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    TRY.
        oo_result = lo_hll->listfhirdatastores( ).
        DATA(lt_datastores) = oo_result->get_datastorepropertieslist( ).
        DATA(lv_datastore_count) = lines( lt_datastores ).
        MESSAGE |Found { lv_datastore_count } data store(s).| TYPE 'I'.
```

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

- Untuk detail API, lihat Daftar FHIRDatastores di AWS SDK untuk referensi SAP ABAP API.

> **Ketersediaan contoh**
>
> Tidak dapat menemukan apa yang Anda butuhkan? Minta contoh kode menggunakan tautan Berikan umpan balik di bilah sisi kanan halaman ini.

AWS Konsol

- Masuk ke halaman Penyimpanan data di HealthLake Konsol.

  Semua penyimpanan HealthLake data terdaftar di bawah bagian Penyimpanan data.

Menandai penyimpanan HealthLake data

Anda dapat menetapkan metadata ke penyimpanan HealthLake data dalam bentuk tag. Setiap tag adalah label yang terdiri dari kunci dan nilai yang ditentukan pengguna. Tag membantu Anda mengelola, mengidentifikasi, mengatur, mencari, dan memfilter penyimpanan data.

> **Penting:**
>
> Jangan menyimpan informasi kesehatan yang dilindungi (PHI), informasi identitas pribadi (PII), atau informasi rahasia atau sensitif lainnya dalam tag. Tag tidak dimaksudkan untuk digunakan dalam data sensitif atau privat.

Topik berikut menjelaskan cara menggunakan operasi HealthLake penandaan menggunakan Konsol Manajemen AWS, AWS CLI, dan AWS SDKs. Untuk informasi selengkapnya, lihat Menandai AWS sumber daya Anda di Referensi Umum AWS Panduan.

Topik

- Menandai penyimpanan HealthLake data
- Listing tag untuk penyimpanan HealthLake data
- Membuka tag penyimpanan data HealthLake

Menandai penyimpanan HealthLake data

Gunakan `TagResource` untuk menandai penyimpanan HealthLake data. Menu berikut memberikan prosedur untuk contoh Konsol Manajemen AWS dan kode untuk AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat `TagResource` di dalam Referensi API AWS HealthLake .

Untuk menandai penyimpanan HealthLake data

Pilih menu berdasarkan preferensi akses Anda AWS HealthLake.

AWS CLI dan SDKs

CLI

AWS CLI

Untuk menambahkan tag ke penyimpanan data

`tag-resource`Contoh berikut menunjukkan cara menambahkan tag ke penyimpanan data.

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

Perintah ini tidak menghasilkan output.

.

- Untuk detail API, lihat TagResourcedi Referensi AWS CLI Perintah.

Python

SDK untuk 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
```

- Untuk detail API, lihat TagResourcedi AWS SDK for Python (Boto3) Referensi API.

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

- Untuk detail API, lihat TagResourcedi AWS SDK untuk referensi SAP ABAP API.

Ketersediaan contoh

Tidak dapat menemukan apa yang Anda butuhkan? Minta contoh kode menggunakan tautan Berikan umpan balik di bilah sisi kanan halaman ini.

AWS Konsol

1. Masuk ke halaman Penyimpanan data di HealthLake Konsol.
2. Pilih penyimpanan data.

   Halaman detail penyimpanan data terbuka.
3. Di bawah bagian Tag, pilih Kelola tag.

   Halaman Kelola tag terbuka.
4. Pilih Tambahkan tag baru.
5. Masukkan Kunci dan Nilai (opsional).
6. Pilih Simpan.

Listing tag untuk penyimpanan HealthLake data

Gunakan `ListTagsForResource` untuk daftar tag untuk penyimpanan HealthLake data. Menu berikut memberikan prosedur untuk contoh Konsol Manajemen AWS dan kode untuk AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat `ListTagsForResource` di dalam Referensi API AWS HealthLake .

Untuk daftar tag untuk penyimpanan HealthLake data

Pilih menu berdasarkan preferensi akses Anda AWS HealthLake.

AWS CLI dan SDKs

CLI

AWS CLI

Untuk daftar tag untuk penyimpanan data

`list-tags-for-resource`Contoh berikut mencantumkan tag yang terkait dengan penyimpanan data yang ditentukan. :

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

Output:

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

- Untuk detail API, lihat ListTagsForResourcedi Referensi AWS CLI Perintah.

Python

SDK untuk 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
```

- Untuk detail API, lihat ListTagsForResourcedi AWS SDK for Python (Boto3) Referensi API.

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    TRY.
        " iv_resource_arn = 'arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        DATA(lo_result) = lo_hll->listtagsforresource(
          iv_resourcearn = iv_resource_arn
        ).
        ot_tags = lo_result->get_tags( ).
        DATA(lv_tag_count) = lines( ot_tags ).
```

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

- Untuk detail API, lihat ListTagsForResourcedi AWS SDK untuk referensi SAP ABAP API.

> **Ketersediaan contoh**
>
> Tidak dapat menemukan apa yang Anda butuhkan? Minta contoh kode menggunakan tautan Berikan umpan balik di bilah sisi kanan halaman ini.

AWS Konsol

1. Masuk ke halaman Penyimpanan data di HealthLake Konsol.
2. Pilih penyimpanan data.

   Halaman detail penyimpanan data terbuka. Di bawah bagian Tag, semua tag penyimpanan data terdaftar.

Membuka tag penyimpanan data HealthLake

Gunakan `UntagResource` untuk menghapus tag dari penyimpanan HealthLake data. Menu berikut memberikan prosedur untuk contoh Konsol Manajemen AWS dan kode untuk AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat `UntagResource` di dalam Referensi API AWS HealthLake .

Untuk menghapus tag penyimpanan HealthLake data

Pilih menu berdasarkan preferensi akses Anda AWS HealthLake.

AWS CLI dan SDKs

CLI

AWS CLI

Untuk menghapus tag dari penyimpanan data.

`untag-resource`Contoh berikut menunjukkan cara menghapus tag dari penyimpanan data.

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

Perintah ini tidak menghasilkan output.

- Untuk detail API, lihat UntagResourcedi Referensi AWS CLI Perintah.

Python

SDK untuk 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
```

- Untuk detail API, lihat UntagResourcedi AWS SDK for Python (Boto3) Referensi API.

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    TRY.
        " iv_resource_arn = 'arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        lo_hll->untagresource(
          iv_resourcearn = iv_resource_arn
          it_tagkeys = it_tag_keys
        ).
        MESSAGE 'Resource untagged successfully.' TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
```

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

- Untuk detail API, lihat UntagResourcedi AWS SDK untuk referensi SAP ABAP API.

> **Ketersediaan contoh**
>
> Tidak dapat menemukan apa yang Anda butuhkan? Minta contoh kode menggunakan tautan Berikan umpan balik di bilah sisi kanan halaman ini.

AWS Konsol

1. Masuk ke halaman Penyimpanan data di HealthLake Konsol.
2. Pilih penyimpanan data.

   Halaman detail penyimpanan data terbuka.
3. Di bawah bagian Tag, pilih Kelola tag.

   Halaman Kelola tag terbuka.
4. Pilih Hapus di samping tag yang ingin Anda hapus.
5. Pilih Simpan.

Menghapus penyimpanan HealthLake data

Gunakan `DeleteFHIRDatastore` untuk menghapus penyimpanan HealthLake data. Menu berikut memberikan prosedur untuk contoh Konsol Manajemen AWS dan kode untuk AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat `DeleteFHIRDatastore` di dalam Referensi API AWS HealthLake .

Untuk menghapus penyimpanan HealthLake data

Pilih menu berdasarkan preferensi akses Anda AWS HealthLake.

AWS CLI dan SDKs

CLI

AWS CLI

Untuk menghapus penyimpanan data FHIR

`delete-fhir-datastore`Contoh berikut menunjukkan cara menghapus penyimpanan data dan semua isinya di AWS HealthLake.

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

Output:

```
{
    "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)"
}
```

- Untuk detail API, lihat Menghapus FHIRDatastore di Referensi AWS CLI Perintah.

Python

SDK untuk 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
```

- Untuk detail API, lihat Menghapus FHIRDatastore di AWS SDK for Python (Boto3) Referensi API.

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

- Untuk detail API, lihat Menghapus FHIRDatastore di AWS SDK untuk referensi SAP ABAP API.

Ketersediaan contoh

Tidak dapat menemukan apa yang Anda butuhkan? Minta contoh kode menggunakan tautan Berikan umpan balik di bilah sisi kanan halaman ini.

AWS Konsol

1. Masuk ke halaman Penyimpanan data di HealthLake Konsol.
2. Pilih penyimpanan data.

   Halaman detail penyimpanan data terbuka.
3. Pilih Hapus.

   Halaman Hapus penyimpanan data terbuka.
4. Untuk mengonfirmasi penghapusan penyimpanan data, masukkan nama penyimpanan data di bidang input teks.
5. Pilih Hapus.

Mengelola Langganan FHIR di AWS HealthLake

AWS HealthLake mendukung Langganan FHIR, memungkinkan Anda menerima pemberitahuan waktu nyata ketika perubahan data perawatan kesehatan tertentu terjadi. Kemampuan ini mengimplementasikan model berlangganan berbasis topik FHIR R5 Backport, memberikan peningkatan skalabilitas dan fleksibilitas dibandingkan model berlangganan FHIR R4 tradisional.

Dengan Langganan FHIR, Anda dapat membangun aplikasi perawatan kesehatan berbasis acara yang segera merespons perubahan data klinis, memungkinkan intervensi tepat waktu, alur kerja otomatis, dan koordinasi perawatan yang ditingkatkan.

Topik

- Cara kerja Langganan FHIR
- Komponen kunci
- Praktik terbaik
- Siklus hidup Berlangganan FHIR dengan AWS HealthLake
- Membuat Langganan FHIR dengan AWS HealthLake
- Mencari Langganan FHIR dengan AWS HealthLake
- Memfilter notifikasi dengan AWS HealthLake

Cara kerja Langganan FHIR

Langganan FHIR HealthLake beroperasi pada model berbasis topik di mana:

1. Membuat topik untuk menentukan peristiwa: Buat Topik Berlangganan yang menentukan peristiwa yang dapat memicu pemberitahuan
2. Anda berlangganan: Buat Langganan untuk topik ini dengan kriteria penyaringan tertentu
3. HealthLake monitor: Layanan terus memantau acara yang sesuai dengan kriteria Anda
4. Pemberitahuan CWhen dikirimkan: peristiwa yang cocok terjadi, HealthLake mengirimkan pemberitahuan melalui saluran yang Anda pilih

Komponen kunci

Langganan FHIR terdiri dari komponen-komponen berikut.

Topik Berlangganan

Topik Berlangganan adalah dasar dari sistem notifikasi dan menentukan:

- Peristiwa pemicu: Perubahan apa yang memicu pemberitahuan (Misalnya: Pembuatan sumber daya, pembaruan, penghapusan)
- Filter yang tersedia: Opsi penyaringan apa yang tersedia untuk pelanggan
- Konten pemberitahuan: Data apa yang termasuk dalam notifikasi

Tabel berikut mencantumkan jenis topik umum.

Tipe peristiwa	Deskripsi	Kasus penggunaan umum
Pembuatan sumber daya	Dipicu saat sumber daya dibuat	Registrasi pasien baru, observasi baru dicatat
Pembaruan sumber daya	Dipicu saat sumber daya dimodifikasi	Perubahan status, pembaruan klinis
Penghapusan sumber daya	Dipicu saat sumber daya dihapus	Audit dan pelacakan kepatuhan

Langganan

Langganan adalah permintaan Anda untuk menerima pemberitahuan untuk acara tertentu yang ditentukan oleh Topik Langganan. Setiap langganan meliputi:

- Referensi topik: Menentukan Topik Berlangganan mana yang Anda berlangganan
- Filter: Kriteria untuk memilih acara mana yang menghasilkan notifikasi
- Konfigurasi saluran: Di mana dan bagaimana pemberitahuan harus dikirimkan
- Preferensi payload: Tingkat detail apa yang harus disertakan dalam notifikasi

Saluran pemberitahuan

HealthLake mendukung saluran notifikasi berikut:

Tipe saluran	Kasus penggunaan	
EventBridge	Integrasi perusahaan, alur kerja tanpa server, orkestrasi lintas layanan AWS	
REST Hook	Pemberitahuan titik akhir langsung, integrasi sistem pihak ketiga	

Muatan pemberitahuan

Pilih jenis muatan yang sesuai berdasarkan kebutuhan Anda:

Jenis muatan	Deskripsi	Pertimbangan keamanan
Hanya ID	Berisi hanya pengidentifikasi sumber daya	Paparan PHI minimal
Sumber daya penuh	Berisi konten sumber daya lengkap dengan ukuran maksimal 256 KB. Jika ukurannya lebih besar dari 256KB, itu akan kembali ke ID-only	Berisi PHI; Verifikasi penanganan yang aman

Praktik terbaik

Optimalisasi kinerja

- Gunakan filter terfokus: Persempit kriteria Anda untuk hanya menerima pemberitahuan penting

- Pilih jenis muatan yang sesuai: Gunakan muatan khusus ID jika memungkinkan untuk kinerja yang lebih baik
- Menerapkan penerima yang efisien: Pastikan penerima notifikasi memproses pesan dengan cepat

Pertimbangan keamanan

- Titik akhir aman: Menerapkan otentikasi yang tepat untuk titik akhir REST Hook
- Perlindungan PHI: Berhati-hatilah dengan muatan sumber daya penuh karena mengandung PHI
- Kontrol akses: Batasi pembuatan Langganan hanya untuk pengguna yang berwenang

Keunggulan operasional

- Tetapkan tanggal akhir yang sesuai: Gunakan tanggal akhir untuk Langganan sementara
- Pantau status Langganan: Periksa status Langganan Anda secara teratur
- Menerapkan penanganan kesalahan: Rancang aplikasi Anda untuk menangani kegagalan pengiriman notifikasi

Siklus hidup Berlangganan FHIR dengan AWS HealthLake

Ikuti langkah-langkah ini untuk memahami siklus hidup Langganan FHIR:

1. Buat `SubscriptionTopic`
   - Buat `SubscriptionTopic` dengan status `"unknown"`
2. Buat `Subscription`
   - Buat `Subscription` dengan status `"requested"`
   - HealthLake memvalidasi konfigurasi `Subscription`
   - `Subscription`harus mereferensikan topik yang sudah ada (topik harus dalam status`unknown`,`draft`,`active`).
3. Aktivasi
   - Jika valid, HealthLake perbarui status `Subscription` ke `"active"`
   - Saat membuat`Subscription`, jika topik yang diberikan dalam status`"unknown"`, HealthLake perbarui status menjadi `"active"` setelah Langganan juga aktif
   - Langganan biasanya membutuhkan waktu 5-10 menit untuk berhasil dibuat

- Jika `Subscription` tidak berhasil dibuat, status akan berubah ke `error` tempat Anda harus melakukan operasi DELETE, lalu coba lagi pembuatan Langganan. Anda dapat melihat `"error"` bidang di sumber Langganan untuk melihat mengapa Langganan tidak berhasil dibuat.

4. Menelan saat Berlangganan `active`
5. Sementara `Subscription` adalah `active`
   - HealthLake monitor untuk acara yang sesuai dengan kriteria Anda
   - Pemberitahuan dikirim ke titik akhir yang dikonfigurasi saat kecocokan terjadi
6. Penanganan Kesalahan
   - HealthLake mencoba mencoba lagi selama 14 hari dan kemudian berhenti mencoba lagi untuk acara tersebut
7. Penonaktifan
   - A `Subscription` dapat dinonaktifkan dengan:

     Mengatur tanggal akhir (penonaktifan otomatis)

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

Menghapus sumber daya `Subscription`

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

Membuat Langganan FHIR dengan AWS HealthLake

Panduan berikut menunjukkan cara membuat Langganan FHIR menggunakan AWS HealthLake.

Untuk membuat Langganan FHIR

1. Buat a`SubscriptionTopic`.

   Contoh sumber Topik Berlangganan:

```
{
  "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. Siapkan titik akhir notifikasi Anda (saluran khusus). Langkah-langkah berikut adalah langkah-langkah yang diperlukan untuk memastikan titik akhir akan menerima pemberitahuan

   Saat menggunakan REST Hook

   - Percayai `events.amazonaws.com` kebijakan kunci KMS Anda jika menggunakan datastore CM_CMK.
   - Jika menggunakan datastore CM_CMK, Anda harus menambahkan `EventBridgeApiDestinations` tag ke kunci KMS Anda dengan nilai `true`
   - HealthLake digunakan OAuth untuk mengautentikasi titik akhir REST Hook Anda. Oleh karena itu, saat membuat langganan REST hook, Anda harus memasukkan client-id, client-secret, dan oAuth-endpoint-url di saluran. _type.extension [*].

   Contoh kebijakan kunci KMS jika menggunakan datastore CM_CMK:

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

   Saat menggunakan EventBridge

   - Percayai `events.amazonaws.com` kebijakan kunci KMS Anda jika menggunakan datastore CM_CMK.
   - Verifikasi kepercayaan kebijakan EventBridge sumber daya Anda `healthlake.amazonaws.com` sebagai prinsipal layanan.
   - Saat menggunakan CM_CMK dan EventBridge merupakan titik akhir, verifikasi bahwa Anda mengenkripsi EventBridge bus Anda dengan kunci KMS yang sama dengan kunci KMS datastore.
   - Verifikasi bahwa EventBridge Bus Anda memiliki setidaknya 1 aturan yang cocok dengan acara yang dihasilkan oleh HealthLake.

Contoh kebijakan sumber daya untuk bus EventBridge saluran:

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

Contoh EventBridge aturan pola peristiwa untuk menerima peristiwa dari: HealthLake

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

> **Note**
>
> HealthLake mendukung 2 sumber:
>
> - “`healthlake`”: Hanya untuk Langganan.
> - “`aws.healthlake`”: Untuk menerima acara HealthLake layanan.
>
> Gunakan “`healthlake`” sebagai sumber saat membuat aturan untuk bus acara Langganan FHIR.

3. Buat Anda `Subscription`

Kirim sumber Langganan dengan:

- Status: `"requested"`
- Referensi ke `SubscriptionTopic` id yang Anda pilih

- Kriteria filter. Untuk informasi selengkapnya, lihat Memfilter Pemberitahuan untuk filter yang didukung.
- Konfigurasi saluran

Contoh muatan Berlangganan

Contoh kode berikut menunjukkan cara membuat payload Langganan.

EventBridge

Berlangganan dengan jenis `event-bridge` saluran dan `id-only` payload.

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

Berlangganan dengan `event-bridge` tipe endpoint dan `full-resource` payload.

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

```
        }
      ]
    }
  }
}
```

Kait Istirahat

Berlangganan dengan `rest-hook` tipe endpoint dan `id-only` payload.

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

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

Berlangganan dengan jenis `rest-hook` saluran dan `full-resource` payload.

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

Contoh payload notifikasi

Saat Langganan sedang dibuat, HealthLake periksa penyiapan langganan yang berhasil dengan mengirimkan bundel jabat tangan ke saluran yang dikonfigurasi. Payload berikut adalah contoh bundel jabat tangan.

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

Contoh bundel notifikasi id-only.

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

Contoh bundel notifikasi sumber daya lengkap.

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

Pembuatan versi acara

HealthLake mendukung sejarah FHIR secara default.

Untuk mengetahui versi sumber daya apa yang Anda terima di bundel notifikasi:

- sumber daya penuh: Karena bundel sumber daya lengkap mencakup seluruh sumber daya, versi akan disertakan dalam `entry[*]` untuk setiap sumber daya yang ada di bundel.
- id-only: Bundel tidak akan menyertakan informasi sumber daya apa pun. HealthLaketermasuk versi yang cocok dan disertakan dalam bundel melalui `entry[0].notificationEvent[*].additionalContext` bidang. Bidang ini dalam format`<ResourceType>/<ResourceId>/_history/<Version Id>`. Untuk informasi selengkapnya, lihat bidang AdditionalContext dalam contoh payload khusus id.

Deteksi duplikasi peristiwa

HealthLakeFitur Langganan FHIR menjamin setidaknya satu pengiriman. Ini berarti Anda mungkin menerima acara yang sama beberapa kali, baik dalam bundel yang sama atau dalam bundel yang berbeda. Untuk mengidentifikasi duplikat, HealthLake berikan id unik untuk setiap peristiwa dalam bundel notifikasi di`entry[0].notificationEvent[*].id`.

Id ini unik untuk versi spesifik dari acara yang dicocokkan dan dikirimkan. Misalnya, jika Encounter yang sama diperbarui dua kali dan kedua pembaruan cocok dengan kriteria filter, Anda akan menerima dua peristiwa terpisah dengan referensi Encounter yang sama. Mereka akan memiliki hal yang sama`notificationEvent[*].focus`, tetapi akan memiliki yang unik`notificationEvent[*].id`. Selain itu, peristiwa ini dapat dikirim dalam bundel terpisah atau dalam bundel pemberitahuan yang sama.

Mencari Langganan FHIR dengan AWS HealthLake

`Subscription`dan `SubscriptionTopic` sumber daya juga dapat dicari. HealthLake mendukung semua parameter pencarian umum untuk Langganan dan SubscriptionTopic sumber daya.

Selain itu, kami mendukung kemampuan pencarian tambahan melalui parameter berikut:

Berlangganan

Parameter pencarian	Deskripsi	Contoh
kontak	Cari di bidang berlangganan. Kontak dalam spesifikasi inti R4	`Subscription?contact=phone`

Parameter pencarian	Deskripsi	Contoh
kriteria	Cari di bidang Subscription.Criteria dalam spesifikasi inti R4	`Subscription?criteria=[baseUrl]/datastore/[datastoreId]/r4/SubscriptionTopic/[topicId]`
payload	Cari di bidang Subscription.channel.Payload dalam spesifikasi inti R4	`Subscription?payload=application/fhir+json`
status	Cari di bidang Subscription.status dalam spesifikasi inti R4	`Subscription?status=error`
jenis	Cari di bidang Subscription.channel.type dalam spesifikasi inti R4	`Subscription?topic=event-bridge`
url	Cari di bidang subscription.channel.endpoint dalam spesifikasi inti R4	`Subscription?url=[Subscription.channel.endpoint]`
kriteria filter	Cari di bidang ekstensi kriteria dengan url "http://hl7. org/fhir/uv/subscriptions-backport/StructureDefinition/backport-filter-criteria” sebagai string	`Subscription?filter-criteria=Encounter?`
kanal kustom	Cari di bidang ekstensi jenis saluran khusus dengan url "http://hl7. org/fhir/uv/subscriptions-backport/StructureDefinition/backport-channel-type” sebagai Coding Contoh payload Berlangganan	`Subscription?custom-channel=[System]|[code]`

Parameter pencarian	Deskripsi	Contoh
jenis muatan	Cari di bidang ekstensi jenis payload tempat Anda menentukan bagaimana payload diformat (salah satu atau) `id-only` `full-resource`	`Subscription?payload-type=full-resource`
topik	Cari di bidang Langganan .kriteria, tempat topik ditambahkan	`Subscription?topic=[topicId]`

SubscriptionTopic

Parameter pencarian	Deskripsi	Contoh
date	Cari di bidang tanggal di SubscriptionTopic sumber daya	`SubscriptionTopic?date=[SubscriptionTopic.date]`
derived-or-self	Cari di salah satu `url` atau `derivedFrom` bidang di SubscriptionTopic sumber daya	`SubscriptionTopic?derived-or-self=[SubscriptionTopic.url | SubscriptionTopic.derivedFrom]`
pengenal	Cari di bidang SubscriptoinTopic .identifier	`SubscriptionTopic?identifier=[SubscriptionTopic.identifier]`
sumber daya	Cari di bidang SubscriptionTopic .resourceTrigger.resource	`SubscriptionTopic?resource=Encounter`

Parameter pencarian	Deskripsi	Contoh
status	Cari pada status SubscriptionTopic	`SubscriptionTopic?status=unknown`
title	Cari pada judul untuk SubscriptionTopic	`SubscriptionTopic?title=admission`
pemicu-deskripsi	Cari di SubscriptionTopic .resourceTrigger.description	`SubscriptionTopic.trigger-description=resource moving to state 'in-progress'`
url	Cari di url untuk SubscriptionTopic	`SubscriptionTopic?url=[SubscriptionTopic.url]`
versi	Cari di versi untuk SubscriptionTopic	`SubscriptionTopic?version=1`

Memfilter notifikasi dengan AWS HealthLake

Perbaiki notifikasi Anda menggunakan parameter pencarian FHIR standar dalam kriteria Anda`Subscription`.

Filter yang didukung

Tabel berikut menunjukkan daftar kriteria filter yang didukung yang HealthLake mendukung untuk Langganan.

Cari jenis parameter	Tipe data FHIR	Pengubah	Awalan yang didukung
String	string HumanName Alamat	tepat, berisi hilang	

Cari jenis parameter	Tipe data FHIR	Pengubah	Awalan yang didukung
Token	boolean code string Pengkodean CodeableConcept Pengidentifikasi ContactPoint id	tidak, teks, hilang	
Bilangan	integer desimal PositiveInt Tidak ditandata nganinya	hilang	“eq”, “ne”, “gt”, “lt”, “ge”, “le”, “sa”, “eb”, “ap”
Date	date dateTime instan Periode Pengaturan waktu		“eq”, “ne”, “gt”, “lt”, “ge”, “le”, “sa”, “eb”, “ap”

Cari jenis parameter	Tipe data FHIR	Pengubah	Awalan yang didukung
Kuantitas	Kuantitas Uang Kisaran SimpleQuantity		“eq”, “ne”, “gt”, “lt”, “ge”, “le”, “sa”, “eb”, “ap”
Referensi	Referensi	hilang, pengidentifikasi, jenis	
URI	uri url canonical uid oid	hilang	

Contoh filter yang didukung

Tabel berikut menunjukkan contoh kriteria filter yang didukung yang HealthLake mendukung Langganan:

Tujuan	Kriteria filter	Deskripsi
Pengamatan khusus pasien	`Observation?patient=Patient/[id]&status=final`	Dapatkan pemberitahuan saat observasi untuk pasien tertentu diselesaikan
Pengamatan khusus pasien	`Patient?birthdate=gt2021`	Dapatkan notifikasi saat pasien yang lahir setelah tahun 2021 terdaftar atau diperbarui

Tujuan	Kriteria filter	Deskripsi
Pengamatan khusus pasien	`Condition?code=http://snomed.info/sct|39065001`	Dapatkan pemberitahuan untuk kondisi dengan kode SNOMED tertentu
Pengamatan khusus pasien	`Observation?code=http://loinc.org|8480-6&value-quantity=gt160`	Dapatkan pemberitahuan untuk pembacaan tekanan darah sistolik tinggi

Mengimpor data FHIR dengan AWS HealthLake

Setelah membuat penyimpanan HealthLake data, langkah selanjutnya adalah mengimpor file dari bucket Amazon Simple Storage Service (S3). Anda dapat memulai pekerjaan impor FHIR menggunakan Konsol Manajemen AWS, AWS CLI, atau AWS SDKs. Gunakan AWS HealthLake tindakan asli untuk memulai, mendeskripsikan, dan mencantumkan pekerjaan impor FHIR.

> **Penting:**
>
> HealthLake mendukung spesifikasi FHIR R4 untuk pertukaran data perawatan kesehatan. Jika diperlukan, Anda dapat bekerja dengan AWS HealthLake Mitra untuk mengonversi data kesehatan Anda ke format FHIR R4 sebelum mengimpor.

Saat memulai pekerjaan impor FHIR, Anda menentukan lokasi input bucket Amazon S3, lokasi keluaran bucket Amazon S3 (untuk hasil pemrosesan pekerjaan), peran IAM yang memberikan HealthLake akses ke bucket Amazon S3, dan kunci yang dimiliki atau dimiliki pelanggan. AWS AWS Key Management Service Untuk informasi selengkapnya, lihat Menyiapkan izin untuk pekerjaan impor.

> **Note**
>
> Anda dapat mengantri pekerjaan impor. Pekerjaan impor asinkron diproses dengan cara FIFO (First In First Out). Anda dapat mengantri pekerjaan dengan cara yang sama seperti Anda memulai pekerjaan impor. Jika sedang berlangsung, itu hanya akan mengantri. Anda dapat membuat, membaca, memperbarui, atau menghapus sumber daya FHIR saat pekerjaan impor sedang berlangsung.

HealthLake menghasilkan `manifest.json` file untuk setiap pekerjaan impor FHIR. File tersebut menjelaskan keberhasilan dan kegagalan pekerjaan impor FHIR. HealthLake mengeluarkan `manifest.json` file ke bucket Amazon S3 yang ditentukan saat memulai pekerjaan impor FHIR. File log disusun menjadi dua folder, bernama `SUCCESS` dan`FAILURE`. Gunakan `manifest.json` file sebagai langkah pertama dalam memecahkan masalah pekerjaan impor yang gagal, karena memberikan detail pada setiap file.

```
{
```

```
    "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
}
```

Mengkonfigurasi tingkat validasi untuk impor

Saat memulai pekerjaan impor FHIR, Anda dapat secara opsional menentukan `ValidationLevel` untuk diterapkan ke setiap sumber daya. AWS HealthLake saat ini mendukung tingkat validasi berikut:

- `strict`: Sumber daya divalidasi sesuai dengan elemen profil sumber daya, atau spesifikasi R4 jika tidak ada profil. Ini adalah tingkat validasi default untuk AWS HealthLake.
- `structure-only`: Sumber daya divalidasi terhadap R4, mengabaikan profil yang direferensikan.
- `minimal`: Sumber daya divalidasi minimal, mengabaikan aturan R4 tertentu. Sumber daya yang gagal dalam pemeriksaan struktur yang diperlukan search/analytics akan diperbarui untuk menyertakan peringatan untuk audit.

Saat mengimpor menggunakan tingkat `minimal` validasi, file log tambahan dapat dihasilkan dalam folder bernama. `SUCCESS_WITH_SEARCH_VALIDATION_FAILURES` Sumber daya dalam file log folder ini dicerna ke dalam datastore Anda meskipun pemeriksaan validasi terkait pencarian gagal. Ini menyiratkan aspek-aspek tertentu dari sumber daya FHIR Anda tidak valid menurut FHIR, dan bidang yang salah bentuk mungkin tidak dapat dicari. Sumber daya ini akan `extension` ditambahkan pada mereka yang menggambarkan kegagalan tersebut.

Topik

- Memulai pekerjaan impor FHIR
- Mendapatkan properti pekerjaan impor FHIR
- Pencatatan pekerjaan impor FHIR

Memulai pekerjaan impor FHIR

Gunakan `StartFHIRImportJob` untuk memulai pekerjaan impor FHIR ke penyimpanan HealthLake data. Menu berikut memberikan prosedur untuk contoh Konsol Manajemen AWS dan kode untuk AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat `StartFHIRImportJob` di dalam Referensi API AWS HealthLake .

> **Penting:**
>
> HealthLake mendukung spesifikasi FHIR R4 untuk pertukaran data perawatan kesehatan. Jika diperlukan, Anda dapat bekerja dengan AWS HealthLake Mitra untuk mengonversi data kesehatan Anda ke format FHIR R4 sebelum mengimpor.

Untuk memulai pekerjaan impor FHIR

Pilih menu berdasarkan preferensi akses Anda AWS HealthLake.

AWS CLI dan SDKs

CLI

AWS CLI

Untuk memulai pekerjaan impor FHIR

`start-fhir-import-job`Contoh berikut menunjukkan bagaimana memulai pekerjaan impor FHIR menggunakan AWS HealthLake.

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

Output:

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

- Untuk detail API, lihat Memulai FHIRImport Job di Referensi AWS CLI Perintah.

Python

SDK untuk 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
```

- Untuk detail API, lihat Memulai FHIRImport Job di AWS SDK for Python (Boto3) Referensi API.

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

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

- Untuk detail API, lihat Memulai FHIRImport Job di AWS SDK untuk referensi API SAP ABAP.

> **Ketersediaan contoh**
>
> Tidak dapat menemukan apa yang Anda butuhkan? Minta contoh kode menggunakan tautan Berikan umpan balik di bilah sisi kanan halaman ini.

AWS Konsol

1. Masuk ke halaman Penyimpanan data di HealthLake Konsol.
2. Pilih penyimpanan data.
3. Pilih Impor.

   Halaman Impor terbuka.
4. Di bawah bagian Input data, masukkan informasi berikut:
   - Masukan lokasi data di Amazon S3
5. Di bawah bagian Impor file keluaran, masukkan informasi berikut:
   - Impor lokasi file keluaran di Amazon S3

- Impor enkripsi file keluaran

6. Di bawah bagian Izin akses, pilih Gunakan peran layanan IAM yang ada dan pilih peran dari menu Nama peran layanan atau pilih Buat peran IAM.
7. Pilih Impor data.

> **Note**
>
> Selama impor, pilih Salin ID pekerjaan pada spanduk di bagian atas halaman. Anda dapat menggunakan `JobID`untuk meminta properti pekerjaan impor menggunakan file AWS CLI. Lihat informasi yang lebih lengkap di Mendapatkan properti pekerjaan impor FHIR.

Mendapatkan properti pekerjaan impor FHIR

Gunakan `DescribeFHIRImportJob` untuk mendapatkan properti pekerjaan impor FHIR. Menu berikut memberikan prosedur untuk contoh Konsol Manajemen AWS dan kode untuk AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat `DescribeFHIRImportJob` di dalam Referensi API AWS HealthLake .

Untuk mendapatkan properti pekerjaan impor FHIR

Pilih menu berdasarkan preferensi akses Anda AWS HealthLake.

AWS CLI dan SDKs

CLI

AWS CLI

Untuk menggambarkan pekerjaan impor FHIR

`describe-fhir-import-job`Contoh berikut menunjukkan bagaimana mempelajari properti pekerjaan impor FHIR menggunakan AWS HealthLake.

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

```
    --job-id c145fbb27b192af392f8ce6e7838e34f
```

Output:

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

- Untuk detail API, lihat Menjelaskan FHIRImport Pekerjaan dalam Referensi AWS CLI Perintah.

Python

SDK untuk 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
```

- Untuk detail API, lihat Menjelaskan FHIRImport Pekerjaan dalam AWS SDK untuk Referensi API Python (Boto3).

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    TRY.
```

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

- Untuk detail API, lihat Menjelaskan FHIRImport Job in AWS SDK untuk referensi SAP ABAP API.

Ketersediaan contoh

Tidak dapat menemukan apa yang Anda butuhkan? Minta contoh kode menggunakan tautan Berikan umpan balik di bilah sisi kanan halaman ini.

AWS Konsol

Note

Informasi pekerjaan impor FHIR tidak tersedia di HealthLake Konsol. Sebagai gantinya, gunakan AWS CLI with `DescribeFHIRImportJob` untuk meminta properti pekerjaan impor seperti `JobStatus`. Untuk informasi lebih lanjut, lihat AWS CLI contoh di halaman ini.

Pencatatan pekerjaan impor FHIR

Gunakan `ListFHIRImportJobs` untuk daftar pekerjaan impor FHIR untuk penyimpanan HealthLake data aktif. Menu berikut memberikan prosedur untuk contoh Konsol Manajemen AWS dan kode untuk AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat `ListFHIRImportJobs` di dalam Referensi API AWS HealthLake .

Untuk daftar pekerjaan impor FHIR

Pilih menu berdasarkan preferensi akses Anda AWS HealthLake.

AWS CLI dan SDKs

CLI

AWS CLI

Untuk mencantumkan semua pekerjaan impor FHIR

`list-fhir-import-jobs`Contoh berikut menunjukkan cara menggunakan perintah untuk melihat daftar semua pekerjaan impor yang terkait dengan akun.

```
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)
```

Output:

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

- Untuk detail API, lihat Daftar FHIRImport Pekerjaan di Referensi AWS CLI Perintah.

Python

SDK untuk 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
```

- Untuk detail API, lihat Daftar FHIRImport Lowongan di AWS SDK for Python (Boto3) Referensi API.

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

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

- Untuk detail API, lihat Daftar FHIRImport Lowongan di AWS SDK untuk referensi SAP ABAP API.

> **Ketersediaan contoh**
>
> Tidak dapat menemukan apa yang Anda butuhkan? Minta contoh kode menggunakan tautan Berikan umpan balik di bilah sisi kanan halaman ini.

AWS Konsol

> **Note**
>
> Informasi pekerjaan impor FHIR tidak tersedia di HealthLake Konsol. Sebagai gantinya, gunakan AWS CLI with `ListFHIRImportJobs` untuk mencantumkan semua pekerjaan impor FHIR. Untuk informasi lebih lanjut, lihat AWS CLI contoh di halaman ini.

Mengelola sumber daya FHIR di AWS HealthLake

Gunakan interaksi RESTful API FHIR R4 untuk mengelola sumber daya FHIR di penyimpanan data. HealthLake Bagian berikut menjelaskan semua interaksi RESTful API FHIR R4 yang HealthLake didukung yang tersedia untuk manajemen sumber daya FHIR. Untuk informasi tentang kemampuan penyimpanan HealthLake data dan bagian mana dari spesifikasi FHIR yang didukungnya, lihatPernyataan Kemampuan FHIR R4 untuk AWS HealthLake.

> **Note**
>
> Interaksi FHIR yang tercantum dalam Bab ini dibangun sesuai dengan standar HL7 FHIR R4 untuk pertukaran data perawatan kesehatan. Karena mereka adalah representasi dari layanan HL7 FHIR, mereka tidak ditawarkan melalui AWS CLI dan. AWS SDKs Untuk informasi selengkapnya, lihat RESTful API dalam dokumentasi FHIR R4 RESTful API.

Tabel berikut mencantumkan interaksi FHIR R4 yang didukung oleh. AWS HealthLake Untuk informasi tentang jenis sumber daya FHIR yang didukung oleh HealthLake, lihatJenis sumber daya.

Interaksi FHIR R4 didukung oleh AWS HealthLake

Interaksi	Deskripsi
Interaksi seluruh sistem	
`capabilities`	Dapatkan pernyataan kemampuan untuk sistem. Lihat Pernyataan Kemampuan FHIR R4 untuk AWS HealthLake.
`batch`	Perbarui, buat, atau hapus sekumpulan sumber daya dalam satu interaksi. Lihat Bundling sumber daya FHIR.
Jenis interaksi tingkat	
`create`	Buat sumber daya baru dengan ID yang ditetapkan server. Lihat Membuat sumber daya FHIR.
`search`	Cari jenis sumber daya berdasarkan beberapa kriteria filter. Lihat Mencari sumber daya FHIR.

Interaksi	Deskripsi
`history`	Ambil riwayat perubahan untuk jenis sumber daya tertentu. Lihat Membaca sejarah sumber daya FHIR.
Interaksi tingkat instans	
`read`	Baca status sumber daya saat ini. Lihat Membaca sumber daya FHIR.
`history`	Baca riwayat perubahan untuk sumber daya tertentu. Lihat Membaca sejarah sumber daya FHIR.
`vread`	Baca status versi sumber daya tertentu. Lihat Membaca riwayat sumber daya FHIR khusus versi.
`update`	Perbarui sumber daya dengan ID-nya (atau buat jika baru). Lihat Memperbarui sumber daya FHIR.
`delete`	Hapus sumber daya. Lihat Menghapus sumber daya FHIR.

Topik

- Membuat sumber daya FHIR
- Membaca sumber daya FHIR
- Membaca sejarah sumber daya FHIR
- Memperbarui sumber daya FHIR
- Memodifikasi Sumber Daya dengan Operasi PATCH
- Bundling sumber daya FHIR
- Menghapus sumber daya FHIR
- Idempotensi dan Konkurensi

Membuat sumber daya FHIR

`create`Interaksi FHIR menciptakan sumber daya FHIR baru di penyimpanan HealthLake data. Untuk informasi tambahan, lihat `create`di dokumentasi FHIR R4 RESTful API.

Untuk membuat sumber daya FHIR

1. Kumpulkan HealthLake `region` dan `datastoreId` nilai. Untuk informasi selengkapnya, lihat Mendapatkan properti penyimpanan data.
2. Tentukan jenis FHIR yang `Resource` akan dibuat. Untuk informasi selengkapnya, lihat Jenis sumber daya.
3. Buat URL untuk permintaan menggunakan nilai yang dikumpulkan untuk HealthLake `region` dan`datastoreId`. Juga termasuk `Resource` jenis FHIR untuk membuat. Untuk melihat seluruh jalur URL dalam contoh berikut, gulir ke atas tombol Salin.

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

4. Membangun badan JSON untuk permintaan, menentukan data FHIR untuk sumber daya baru. Untuk tujuan prosedur ini, kami menggunakan `Patient` sumber daya FHIR, jadi simpan file sebagai`create-patient.json`.

```
{
    "resourceType": "Patient",
    "identifier": [
        {
            "system": "urn:oid:1.2.36.146.595.217.0.1",
            "value": "12345"
        }
    ],
    "name": [
        {
            "family": "Silva",
            "given": [
                "Ana",
                "Carolina"
            ]
        }
    ],
    "gender": "female",
    "birthDate": "1992-02-10"
}
```

5. Kirim permintaan . `create`Interaksi FHIR menggunakan `POST` permintaan dengan AWS Signature Version 4 atau SMART pada otorisasi FHIR. Contoh berikut membuat `Patient` resource FHIR dalam HealthLake menggunakan curl atau Console. HealthLake Untuk melihat seluruh contoh, gulir ke atas tombol Salin.

SigV4

Otorisasi SiGv4

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

SMART on FHIR

SMART pada contoh otorisasi FHIR untuk tipe `IdentityProviderConfiguration`data.

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

Penelepon dapat menetapkan izin di lambda otorisasi. Untuk informasi selengkapnya, lihat OAuth 2.0 cakupan.

AWS Console

> **Note**
>
> HealthLake Konsol hanya mendukung otorisasi AWS SiGv4.

1. Masuk ke halaman Jalankan kueri di HealthLake Konsol.
2. Di bawah bagian Pengaturan kueri, buat pilihan berikut.
   - ID Penyimpanan Data — pilih ID penyimpanan data untuk menghasilkan string kueri.
   - Jenis kueri - pilih`Create`.
   - Jenis sumber daya - pilih jenis sumber daya FHIR untuk dibuat.
   - Badan permintaan - membangun badan JSON untuk permintaan, menentukan data FHIR untuk sumber daya baru.
3. Pilih Run query (Jalankan kueri).

Mengkonfigurasi tingkat validasi untuk pembuatan sumber daya

Saat membuat sumber daya FHIR, Anda dapat menentukan header `x-amzn-healthlake-fhir-validation-level` HTTP secara opsional untuk mengonfigurasi tingkat validasi sumber daya. AWS HealthLake saat ini mendukung tingkat validasi berikut:

- `strict`: Sumber daya divalidasi sesuai dengan elemen profil sumber daya, atau spesifikasi R4 jika tidak ada profil. Ini adalah tingkat validasi default untuk AWS HealthLake.
- `structure-only`: Sumber daya divalidasi terhadap R4, mengabaikan profil yang direferensikan.
- `minimal`: Sumber daya divalidasi minimal, mengabaikan aturan R4 tertentu. Sumber daya yang gagal dalam pemeriksaan struktur yang diperlukan search/analytics akan diperbarui untuk menyertakan peringatan untuk audit.

Sumber daya yang dibuat dengan tingkat validasi minimal dapat dicerna ke dalam Datastore meskipun validasi gagal diperlukan untuk pengindeksan pencarian. Dalam hal ini, sumber daya akan

diperbarui untuk menyertakan ekstensi khusus Healthlake untuk mendokumentasikan kegagalan tersebut:

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

Selain itu, header respons HTTP berikut akan disertakan dengan nilai “true”:

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

> **Note**
>
> Data yang dicerna yang salah bentuk sesuai spesifikasi R4 mungkin tidak dapat dicari seperti yang diharapkan jika kesalahan ini ada.

Membaca sumber daya FHIR

`read`Interaksi FHIR membaca keadaan sumber daya saat ini di penyimpanan HealthLake data. Untuk informasi tambahan, lihat `read`di dokumentasi FHIR R4 RESTful API.

Untuk membaca sumber daya FHIR

1. Kumpulkan HealthLake `region` dan `datastoreId` nilai. Untuk informasi selengkapnya, lihat Mendapatkan properti penyimpanan data.
2. Tentukan jenis FHIR `Resource` untuk membaca dan mengumpulkan `id` nilai terkait. Untuk informasi selengkapnya, lihat Jenis sumber daya.
3. Buat URL untuk permintaan menggunakan nilai yang dikumpulkan untuk HealthLake `region` dan`datastoreId`. Juga termasuk `Resource` jenis FHIR dan yang terkait`id`. Untuk melihat seluruh jalur URL dalam contoh berikut, gulir ke atas tombol Salin.

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

4. Kirim permintaan . `read`Interaksi FHIR menggunakan `GET` permintaan dengan AWS Signature Version 4 atau SMART pada otorisasi FHIR. `curl`Contoh berikut membaca status saat ini dari `Patient` sumber daya FHIR di HealthLake. Untuk melihat seluruh contoh, gulir ke atas tombol Salin.

   SigV4

   Otorisasi SiGv4

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

   SMART on FHIR

   SMART pada contoh otorisasi FHIR untuk tipe `IdentityProviderConfiguration`data.

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

Penelepon dapat menetapkan izin di lambda otorisasi. Untuk informasi selengkapnya, lihat OAuth 2.0 cakupan.

AWS Console

1. Masuk ke halaman Jalankan kueri di HealthLake Konsol.
2. Di bawah bagian Pengaturan kueri, buat pilihan berikut.
   - ID Penyimpanan Data — pilih ID penyimpanan data untuk menghasilkan string kueri.
   - Jenis kueri - pilih`Read`.
   - Jenis sumber daya - pilih jenis sumber daya FHIR untuk dibaca.
   - ID Sumber Daya — masukkan ID sumber daya FHIR.
3. Pilih Run query (Jalankan kueri).

Membaca sejarah sumber daya FHIR

`history`Interaksi FHIR mengambil sejarah sumber daya FHIR tertentu di penyimpanan data. HealthLake Dengan menggunakan interaksi ini, Anda dapat menentukan bagaimana isi sumber daya FHIR telah berubah dari waktu ke waktu. Ini juga berguna dalam koordinasi dengan log audit untuk melihat keadaan sumber daya sebelum dan sesudah modifikasi. Interaksi FHIR`create`,`update`, dan `delete` menghasilkan versi historis dari sumber daya yang akan disimpan. Untuk informasi tambahan, lihat `history`di dokumentasi FHIR R4 RESTful API.

> **Note**
>
> Anda dapat memilih keluar dari `history` jenis sumber daya FHIR tertentu. Untuk memilih keluar, buat kasus menggunakan AWS Support Center Console. Untuk membuat kasus Anda, masuk ke kasing Anda Akun AWS dan pilih Buat kasus.

Untuk membaca sejarah sumber daya FHIR

1. Kumpulkan HealthLake `region` dan `datastoreId` nilai. Untuk informasi selengkapnya, lihat Mendapatkan properti penyimpanan data.

2. Tentukan jenis FHIR `Resource` untuk membaca dan mengumpulkan `id` nilai terkait. Untuk informasi selengkapnya, lihat Jenis sumber daya.

3. Buat URL untuk permintaan menggunakan nilai yang dikumpulkan untuk HealthLake `region` dan`datastoreId`. Juga termasuk `Resource` jenis FHIR, yang terkait`id`, dan parameter pencarian opsional. Untuk melihat seluruh jalur URL dalam contoh berikut, gulir ke atas tombol Salin.

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

HealthLake parameter pencarian yang didukung untuk interaksi FHIR **`history`**

Parameter	Deskripsi
`_count : integer`	Jumlah maksimum hasil pencarian pada halaman. Server akan mengembalikan nomor yang diminta atau jumlah maksimum hasil pencarian yang diizinkan secara default untuk penyimpanan data, mana yang lebih rendah.
`_since : instant`	Hanya sertakan versi sumber daya yang dibuat pada atau setelah waktu instan yang diberikan.
`_at : date(Time)`	Hanya sertakan versi sumber daya yang saat ini di beberapa titik selama periode waktu yang ditentukan dalam nilai waktu tanggal. Untuk informasi selengkapnya, lihat `date`di dokumentasi HL7 FHIR RESTful API.

4. Kirim permintaan . `history`Interaksi FHIR menggunakan `GET` permintaan dengan AWS Signature Version 4 atau SMART pada otorisasi FHIR. `curl`Contoh berikut menggunakan parameter `_count` pencarian untuk mengembalikan 100 hasil pencarian historis per halaman untuk `Patient` sumber daya FHIR di HealthLake. Untuk melihat seluruh contoh, gulir ke atas tombol Salin.

SigV4

Otorisasi SiGv4

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

SMART on FHIR

SMART pada contoh otorisasi FHIR untuk tipe `IdentityProviderConfiguration`data.

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

Penelepon dapat menetapkan izin di lambda otorisasi. Untuk informasi selengkapnya, lihat OAuth 2.0 cakupan.

Isi pengembalian `history` interaksi terkandung dalam sumber daya FHIR`Bundle`, dengan tipe yang disetel ke`history`. Ini berisi riwayat versi yang ditentukan, diurutkan dengan versi tertua yang terakhir, dan termasuk sumber daya yang dihapus. Untuk informasi lebih lanjut, lihat `Resource Bundle`di dokumentasi FHIR R4.

Membaca riwayat sumber daya FHIR khusus versi

`vread`Interaksi FHIR melakukan pembacaan sumber daya khusus versi di penyimpanan data. HealthLake Dengan menggunakan interaksi ini, Anda dapat melihat konten sumber daya FHIR seperti pada waktu tertentu di masa lalu.

> **Note**
>
> Jika Anda menggunakan `history` interaksi FHIR tanpa`vread`, HealthLake selalu mengembalikan versi terbaru dari metadata sumber daya.

HealthLake mendeklarasikannya mendukung pembuatan versi `CapabilityStatement.rest.resource.versioning`untuk setiap sumber daya yang didukung. Semua penyimpanan HealthLake data termasuk `Resource.meta.versionId` (`vid`) pada semua sumber daya.

Ketika `history` interaksi FHIR diaktifkan (secara default untuk penyimpanan data yang dibuat setelah 10/25/2024 atau berdasarkan permintaan untuk penyimpanan data yang lebih lama), `Bundle` respons menyertakan `vid` sebagai bagian dari elemen. `location` Dalam contoh berikut, `vid` ditampilkan sebagai angka`1`. Untuk melihat contoh selengkapnya, lihat Contoh Bundle/Bundle-response (JSON).

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

Untuk membaca riwayat sumber daya FHIR khusus versi

1. Kumpulkan HealthLake `region` dan `datastoreId` nilai. Untuk informasi selengkapnya, lihat Mendapatkan properti penyimpanan data.
2. Tentukan `Resource` jenis FHIR untuk membaca dan mengumpulkan terkait `id` dan `vid` nilai. Untuk informasi selengkapnya, lihat Jenis sumber daya.
3. Buat URL untuk permintaan menggunakan nilai yang dikumpulkan untuk HealthLake dan FHIR. Untuk melihat seluruh jalur URL dalam contoh berikut, gulir ke atas tombol Salin.

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

4. Kirim permintaan . `history`Interaksi FHIR menggunakan `GET` permintaan dengan AWS Signature Version 4 atau SMART pada otorisasi FHIR. `vread`Interaksi berikut mengembalikan satu contoh dengan konten yang ditentukan untuk `Patient` sumber daya FHIR untuk versi metadata sumber daya yang ditentukan oleh. `vid` Untuk melihat seluruh jalur URL dalam contoh berikut, gulir ke atas tombol Salin.

   SigV4

   Otorisasi SiGv4

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

   SMART on FHIR

   SMART pada contoh otorisasi FHIR untuk tipe `IdentityProviderConfiguration`data.

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

Penelepon dapat menetapkan izin di lambda otorisasi. Lihat informasi yang lebih lengkap di OAuth 2.0 cakupan.

Memperbarui sumber daya FHIR

`update`Interaksi FHIR membuat versi baru saat ini untuk sumber daya yang ada atau membuat versi awal jika tidak ada sumber daya yang sudah ada untuk yang diberikan`id`. Untuk informasi tambahan, lihat `update`di dokumentasi FHIR R4 RESTful API.

Untuk memperbarui sumber daya FHIR

1. Kumpulkan HealthLake `region` dan `datastoreId` nilai. Untuk informasi selengkapnya, lihat Mendapatkan properti penyimpanan data.
2. Tentukan jenis FHIR `Resource` untuk memperbarui dan mengumpulkan `id` nilai terkait. Untuk informasi selengkapnya, lihat Jenis sumber daya.
3. Buat URL untuk permintaan menggunakan nilai yang dikumpulkan untuk HealthLake `region` dan`datastoreId`. Juga termasuk `Resource` jenis FHIR dan yang terkait`id`. Untuk melihat seluruh jalur URL dalam contoh berikut, gulir ke atas tombol Salin.

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

4. Membangun `JSON` badan untuk permintaan, menentukan update data FHIR yang akan dibuat. Untuk tujuan prosedur ini, simpan file sebagai`update-patient.json`.

   ```
   {
       "id": "2de04858-ba65-44c1-8af1-f2fe69a977d9",
       "resourceType": "Patient",
   ```

```
    "active": true,
    "name": [
        {
            "use": "official",
            "family": "Doe",
            "given": [
                "Jane"
            ]
        },
        {
            "use": "usual",
            "given": [
                "Jane"
            ]
        }
    ],
    "gender": "female",
    "birthDate": "1985-12-31"
}
```

5. Kirim permintaan . `update`Interaksi FHIR menggunakan `PUT` permintaan dengan AWS Signature Version 4 atau SMART pada otorisasi FHIR. `curl`Contoh berikut memperbarui sumber `Patient` daya di HealthLake. Untuk melihat seluruh contoh, gulir ke atas tombol Salin.

   SigV4

   Otorisasi SiGv4

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

   Permintaan Anda akan mengembalikan kode status `200` HTTP jika sumber daya yang ada diperbarui atau kode status `201` HTTP jika sumber daya baru dibuat.

   SMART on FHIR

   SMART pada contoh otorisasi FHIR untuk tipe `IdentityProviderConfiguration`data.

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

Penelepon dapat menetapkan izin di lambda otorisasi. Untuk informasi selengkapnya, lihat OAuth 2.0 cakupan.

AWS Console

1. Masuk ke halaman Jalankan kueri di HealthLake Konsol.

2. Di bawah bagian Pengaturan kueri, buat pilihan berikut.

- ID Penyimpanan Data — pilih ID penyimpanan data untuk menghasilkan string kueri.
- Jenis kueri - pilih`Update (PUT)`.
- Jenis sumber daya - pilih jenis sumber daya FHIR untuk memperbarui atau membuat.
- Badan permintaan - buat badan JSON untuk permintaan, menentukan data FHIR untuk memperbarui sumber daya dengan.

3. Pilih Run query (Jalankan kueri).

Memperbarui sumber daya FHIR berdasarkan kondisi

Pembaruan bersyarat memungkinkan Anda memperbarui sumber daya yang ada berdasarkan beberapa kriteria pencarian identifikasi, bukan oleh `id` FHIR logis. Ketika server memproses pembaruan, ia melakukan pencarian menggunakan kemampuan pencarian standar untuk jenis sumber daya, dengan tujuan menyelesaikan satu logis `id` untuk permintaan tersebut.

Tindakan yang diambil server tergantung pada berapa banyak kecocokan yang ditemukannya:

- Tidak ada kecocokan, tidak **`id`** disediakan di badan permintaan: Server membuat sumber daya FHIR.
- Tidak ada kecocokan, **`id`** disediakan, dan sumber daya yang belum ada dengan **`id`**: Server memperlakukan interaksi sebagai Pembaruan sebagai interaksi Buat.
- Tidak ada kecocokan, **`id`** disediakan dan sudah ada: Server menolak pembaruan dengan `409 Conflict` kesalahan.
- One Match, tidak ada sumber daya yang **`id`** **`id`** disediakan ATAU (sumber daya yang disediakan dan cocok dengan sumber daya yang ditemukan): Server melakukan pembaruan terhadap sumber daya yang cocok seperti di atas di mana, jika sumber daya diperbarui, server HARUS mengembalikan file`200 OK`.
- One Match, sumber daya yang **`id`** disediakan tetapi tidak cocok dengan sumber daya yang ditemukan: Server mengembalikan `409 Conflict` kesalahan yang menunjukkan spesifikasi id klien adalah masalah yang lebih disukai dengan `OperationOutcome`
- Beberapa kecocokan: Server mengembalikan `412 Precondition Failed` kesalahan yang menunjukkan kriteria klien tidak cukup selektif lebih disukai dengan OperationOutcome

Contoh berikut memperbarui `Patient` sumber daya yang namanya peter, tanggal lahir 1 Jan 2000, dan nomor telepon 1234567890.

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

Mengkonfigurasi tingkat validasi untuk pembaruan sumber daya

Saat memperbarui sumber daya FHIR, Anda dapat menentukan header `x-amzn-healthlake-fhir-validation-level` HTTP secara opsional untuk mengonfigurasi tingkat validasi sumber daya. AWS HealthLake saat ini mendukung tingkat validasi berikut:

- `strict`: Sumber daya divalidasi sesuai dengan elemen profil sumber daya, atau spesifikasi R4 jika tidak ada profil. Ini adalah tingkat validasi default untuk AWS HealthLake.
- `structure-only`: Sumber daya divalidasi terhadap R4, mengabaikan profil yang direferensikan.
- `minimal`: Sumber daya divalidasi minimal, mengabaikan aturan R4 tertentu. Sumber daya yang gagal dalam pemeriksaan struktur yang diperlukan search/analytics akan diperbarui untuk menyertakan peringatan untuk audit.

Sumber daya yang diperbarui dengan tingkat validasi minimal dapat dicerna ke dalam Datastore meskipun validasi gagal diperlukan untuk pengindeksan pencarian. Dalam hal ini, sumber daya akan diperbarui untuk menyertakan ekstensi khusus Healthlake untuk mendokumentasikan kegagalan tersebut:

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

Selain itu, header respons HTTP berikut akan disertakan dengan nilai “true”:

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

> **Note**
>
> Perhatikan bahwa data yang dicerna yang cacat menurut spesifikasi R4 mungkin tidak dapat dicari seperti yang diharapkan jika kesalahan ini ada.

Memodifikasi Sumber Daya dengan Operasi PATCH

AWS HealthLake mendukung operasi PATCH untuk sumber daya FHIR, memungkinkan Anda untuk memodifikasi sumber daya dengan menargetkan elemen tertentu untuk menambah, mengganti, atau menghapus tanpa memperbarui seluruh sumber daya. Operasi ini sangat berguna ketika Anda perlu:

- Buat pembaruan yang ditargetkan ke sumber daya besar

- Mengurangi penggunaan bandwidth jaringan
- Lakukan modifikasi atom pada elemen sumber daya tertentu
- Meminimalkan risiko penimpaan perubahan bersamaan
- Perbarui sumber daya sebagai bagian dari alur kerja batch dan transaksi

Format PATCH yang Didukung

AWS HealthLake mendukung dua format PATCH standar:

Tambalan JSON (RFC 6902)

Menggunakan sintaks JSON Pointer untuk menargetkan elemen dengan posisi mereka dalam struktur sumber daya.

Tipe Konten: `application/json-patch+json`

FHIRPath Patch (Spesifikasi FHIR R4)

Menggunakan FHIRPath ekspresi untuk menargetkan elemen berdasarkan konten dan hubungannya, memberikan pendekatan asli FHIR untuk menambal.

Tipe Konten: `application/fhir+json`

Penggunaan

Operasi PATCH Langsung

Operasi PATCH dapat dipanggil langsung pada sumber daya FHIR menggunakan metode PATCH HTTP:

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

PATCH dalam Bundel

Operasi PATCH dapat dimasukkan sebagai entri dalam FHIR Bundle jenis `batch` atau`transaction`, memungkinkan Anda untuk menggabungkan operasi patch dengan interaksi FHIR lainnya (membuat, membaca, memperbarui, menghapus) dalam satu permintaan.

- Bundel transaksi: Semua entri berhasil atau gagal secara atom
- Bundel Batch: Setiap entri diproses secara independen

Format Patch JSON

Operasi yang Didukung

Operasi	Deskripsi
`add`	Tambahkan nilai baru ke sumber daya
`remove`	Hapus nilai dari sumber daya
`replace`	Ganti nilai yang ada di sumber daya
`move`	Hapus nilai dari satu lokasi dan tambahkan ke lokasi lain
`copy`	Salin nilai dari satu lokasi ke lokasi lain
`test`	Uji bahwa nilai di lokasi target sama dengan nilai yang ditentukan

Sintaks Path

JSON Patch menggunakan sintaks JSON Pointer (RFC 6901):

Contoh Jalur	Deskripsi
`/name/0/family`	Elemen keluarga nama depan
`/telecom/-`	Tambahkan ke array telekomunikasi
`/active`	Elemen aktif tingkat akar
`/address/0/line/1`	Baris kedua dari alamat pertama

Contoh

Permintaan Patch JSON Langsung dengan Beberapa Operasi

```
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"
  }
]
```

Permintaan Patch JSON Langsung dengan Operasi Tunggal

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

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

JSON Patch dalam Bundel

Gunakan sumber daya Biner yang berisi payload JSON Patch yang disandikan base64:

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

FHIRPath Format Patch

Operasi yang Didukung

Operasi	Deskripsi
`add`	Tambahkan elemen baru ke sumber daya
`insert`	Menyisipkan elemen pada posisi tertentu dalam daftar
`delete`	Hapus elemen dari sumber daya

Operasi	Deskripsi
`replace`	Ganti nilai elemen yang ada
`move`	Menyusun ulang elemen dalam daftar

Sintaks Path

FHIRPath Patch menggunakan FHIRPath ekspresi, mendukung:

- Akses berbasis indeks: `Patient.name[0]`
- Penyaringan dengan **`where()`**: `Patient.name.where(use = 'official')`
- Logika Boolean: `Patient.telecom.where(system = 'phone' and use = 'work')`
- Fungsi subsetting:`first()`, `last()`
- Pemeriksaan keberadaan:`exists()`, `count()`
- Navigasi polimorfik: `Observation.value`

Contoh

Permintaan FHIRPath Patch Langsung

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

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

FHIRPath Patch dalam Bundel

Gunakan sumber daya Parameter sebagai sumber daya entri dengan`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"
    }
  }]
}
```

Header Permintaan

Header	Diperlukan	Deskripsi
`Content-Type`	Ya	`application/json-patch+json` untuk JSON Patch atau `application/fhir+json` untuk FHIRPath Patch
`If-Match`	Tidak	Pembaruan bersyarat khusus versi menggunakan ETag

Contoh Respons

Operasi mengembalikan sumber daya yang diperbarui dengan informasi versi baru:

```
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"
  }
}
```

Perilaku

Operasi PATCH:

- Memvalidasi sintaks patch sesuai dengan spesifikasi yang sesuai (RFC 6902 untuk JSON Patch, FHIR R4 untuk Patch) FHIRPath
- Menerapkan operasi secara atom - semua operasi berhasil atau semuanya gagal
- Memperbarui ID versi sumber daya dan membuat entri riwayat baru
- Mempertahankan sumber daya asli dalam sejarah sebelum menerapkan perubahan
- Memvalidasi kendala sumber daya FHIR setelah menerapkan tambalan
- Mendukung pembaruan bersyarat menggunakan header If-Match dengan ETag

Penanganan Kesalahan

Operasi menangani kondisi kesalahan berikut:

- 400 Permintaan Buruk: Sintaks tambalan tidak valid (permintaan tidak sesuai atau dokumen tambalan yang salah)
- 404 Tidak Ditemukan: Sumber daya tidak ditemukan (ID tertentu tidak ada)
- 409 Konflik: Konflik versi (pembaruan bersamaan atau ID versi tidak saat ini disediakan)
- 422 Entitas yang Tidak Dapat Diproses: Operasi tambalan tidak dapat diterapkan ke elemen sumber daya yang ditentukan

Kesimpulan dari Capabilities

Kemampuan	Tambalan JSON	FHIRPath Patch
Jenis Konten	`application/json-patch+json`	`application/fhir+json`
Format Jalur	Penunjuk JSON (RFC 6901)	FHIRPath ekspresi
API PATCH langsung	Didukung	Didukung
Bundel Batch	Didukung (melalui Biner)	Didukung (melalui Parameter)
Transaksi Bundel	Didukung (melalui Biner)	Didukung (melalui Parameter)
Operasi	tambahkan, hapus, ganti, pindahkan, salin, uji	tambahkan, masukkan, hapus, ganti, pindahkan

Batasan

- Operasi PATCH bersyarat menggunakan kondisi pencarian tidak didukung
- JSON Patch dalam bundel harus menggunakan sumber daya Biner dengan konten yang disandikan base64
- FHIRPath Patch dalam bundel harus menggunakan sumber daya Parameter

Sumber Daya Tambahan

Untuk informasi selengkapnya tentang operasi PATCH, lihat:

- Dokumentasi PATCH FHIR R4
- Spesifikasi Patch FHIR R4 FHIRPath
- RFC 6902 - Patch JSON
- RFC 6901 - Penunjuk JSON

Bundling sumber daya FHIR

FHIR `Bundle` adalah wadah untuk koleksi sumber daya FHIR di. AWS HealthLake AWS HealthLake mendukung dua jenis bundel dengan perilaku pemrosesan yang berbeda.

`Batch`bundel memproses setiap sumber daya secara independen. Jika satu sumber daya gagal, sumber daya yang tersisa masih bisa berhasil. Setiap operasi diproses secara individual, dan pemrosesan berlanjut bahkan ketika beberapa operasi gagal. Gunakan bundel batch untuk operasi massal di mana keberhasilan sebagian dapat diterima, seperti mengunggah beberapa catatan pasien yang tidak terkait.

`Transaction`bundel memproses semua sumber daya secara atom sebagai satu unit. Entah semua operasi sumber daya berhasil, atau tidak AWS HealthLake melakukan satu pun dari mereka. Gunakan bundel transaksi saat Anda membutuhkan jaminan integritas referensial di seluruh sumber daya terkait, seperti membuat pasien dengan pengamatan dan kondisi terkait di mana semua data harus direkam bersama.

Perbedaan antara batch dan bundel transaksi

Fitur	Batch	Transaksi
Model pengolahan	Setiap operasi berhasil atau gagal secara independen.	Semua operasi berhasil atau gagal sebagai satu unit atom.
Penanganan kegagalan	Pemrosesan berlanjut bahkan jika operasi individu gagal.	Seluruh bundel gagal jika ada operasi tunggal yang gagal.
Perintah eksekusi	Perintah eksekusi tidak dijamin.	Operasi diproses dalam urutan yang ditentukan.

Fitur	Batch	Transaksi
Integritas referensial	Tidak diberlakukan di seluruh operasi.	Ditegakkan untuk sumber daya yang direferensikan secara lokal dalam bundel.
Paling baik digunakan untuk	Operasi massal di mana keberhasilan sebagian dapat diterima.	Sumber daya terkait yang harus dibuat atau diperbarui bersama.

Anda dapat menggabungkan sumber daya FHIR dari jenis yang sama atau berbeda, dan mereka dapat menyertakan campuran operasi FHIR, seperti,`create`,, `read` `update`, `delete`, dan. `patch` Untuk informasi tambahan, lihat Resource Bundle dalam dokumentasi FHIR R4.

Berikut ini adalah contoh kasus penggunaan untuk setiap jenis bundel.

Bundel Batch

- Unggah beberapa catatan pasien yang tidak terkait dari fasilitas yang berbeda selama sinkronisasi data malam hari.
- Unggah massal catatan pengobatan historis di mana beberapa catatan mungkin memiliki masalah validasi.
- Muat data referensi, seperti organisasi dan praktisi, di mana kegagalan individu tidak memengaruhi entri lain.

Bundel transaksi

- Buat pasien dengan pengamatan dan kondisi terkait selama penerimaan departemen gawat darurat di mana semua data harus direkam bersama.
- Perbarui daftar obat pasien dan informasi alergi terkait yang harus tetap konsisten.
- Catat pertemuan lengkap dengan pasien, pengamatan, prosedur, dan informasi penagihan sebagai satu unit atom.

Important

Bundel batch dan transaksi menggunakan struktur `Bundle` sumber daya yang sama. Satu-satunya perbedaan adalah nilai `type` bidang.

Contoh berikut menunjukkan bundel transaksi dengan beberapa jenis sumber daya dan operasi.

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

Menggabungkan sumber daya FHIR sebagai entitas independen

Untuk menggabungkan sumber daya FHIR sebagai entitas independen

1. Kumpulkan HealthLake `region` dan `datastoreId` nilai. Untuk informasi selengkapnya, lihat Mendapatkan properti penyimpanan data.
2. Buat URL untuk permintaan menggunakan nilai yang dikumpulkan untuk HealthLake `region` dan`datastoreId`. Jangan tentukan jenis sumber daya FHIR di URL. Untuk melihat seluruh jalur URL dalam contoh berikut, gulir ke atas tombol Salin.

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

3. Membangun badan JSON untuk permintaan, menentukan setiap kata kerja HTTP sebagai bagian dari elemen. `method` Contoh berikut menggunakan interaksi `batch` tipe dengan `Bundle` sumber daya untuk membuat baru `Patient` dan `Medication` sumber daya. Semua bagian yang diperlukan dikomentari sesuai. Untuk tujuan prosedur ini, simpan file sebagai`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. Kirim permintaan . Jenis `Bundle` batch FHIR menggunakan `POST` permintaan dengan AWS Signature Version 4 atau SMART pada otorisasi FHIR. Contoh kode berikut menggunakan alat baris `curl` perintah untuk tujuan demonstrasi.

   SigV4

   Otorisasi SiGv4

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

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

SMART on FHIR

SMART pada contoh otorisasi FHIR untuk tipe `IdentityProviderConfiguration`data.

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

Penelepon dapat menetapkan izin di lambda otorisasi. Untuk informasi selengkapnya, lihat OAuth 2.0 cakupan.

Server mengembalikan respons yang menunjukkan `Patient` dan `Medication` sumber daya yang dibuat sebagai hasil dari permintaan jenis `Bundle` batch.

Bersyarat PUTs dalam bundel

AWS HealthLake mendukung pembaruan bersyarat dalam bundel menggunakan parameter kueri berikut:

> **Note**
>
> PUTs Kondisional hanya didukung dalam `batch` bundel. `Transaction`bundel tidak mendukung kondisional PUTs.

- `_id`(mandiri)
- `_id`dalam kombinasi dengan salah satu dari berikut ini:
  - `_tag`
  - `_createdAt`
  - `_lastUpdated`

Saat Anda menggunakan kondisional PUTs dalam bundel, AWS HealthLake mengevaluasi parameter kueri terhadap sumber daya yang ada dan mengambil tindakan berdasarkan hasil pencocokan.

Perilaku pembaruan bersyarat

Skenario	Status HTTP	Tindakan yang diambil
Sumber daya tanpa ID yang disediakan	201 Dibuat	Selalu menciptakan sumber daya baru.
Sumber daya dengan ID baru (tidak cocok)	201 Dibuat	Membuat sumber daya baru dengan ID yang ditentukan.
Sumber daya dengan ID yang ada (kecocokan tunggal)	200 OK	Memperbarui sumber daya yang cocok.
Sumber daya dengan ID yang ada (konflik terdeteksi)	409 Konflik	Mengembalikan kesalahan. Tidak ada perubahan yang dibuat.
Sumber daya dengan ID yang ada (ketidakcocokan ID)	400 Permintaan Buruk	Mengembalikan kesalahan. Tidak ada perubahan yang dibuat.
Beberapa kondisi pencocokan sumber daya	412 Prasyarat Gagal	Mengembalikan kesalahan. Tidak ada perubahan yang dibuat.

Dalam contoh bundel berikut dengan pembaruan bersyarat, `Patient` sumber daya dengan ID FHIR `476` diperbarui hanya jika kondisi `_lastUpdated=lt2025-04-20` terpenuhi.

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

Menggabungkan sumber daya FHIR sebagai satu kesatuan

Untuk menggabungkan sumber daya FHIR sebagai satu kesatuan

1. Kumpulkan HealthLake `region` dan `datastoreId` nilai. Untuk informasi selengkapnya, lihat Mendapatkan properti penyimpanan data.
2. Buat URL untuk permintaan menggunakan nilai yang dikumpulkan untuk HealthLake `region` dan`datastoreId`. Sertakan jenis sumber daya FHIR `Bundle` sebagai bagian dari URL. Untuk melihat seluruh jalur URL dalam contoh berikut, gulir ke atas tombol Salin.

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

3. Membangun badan JSON untuk permintaan, menentukan sumber daya FHIR untuk dikelompokkan bersama. Contoh berikut mengelompokkan dua `Patient` sumber daya di HealthLake. Untuk tujuan prosedur ini, simpan file sebagai`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. Kirim permintaan . Jenis `Bundle` dokumen FHIR menggunakan `POST` permintaan dengan protokol penandatanganan AWS Signature Version 4. Contoh kode berikut menggunakan alat baris `curl` perintah untuk tujuan demonstrasi.

   SigV4

   Otorisasi SiGv4

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

   SMART on FHIR

   SMART pada contoh otorisasi FHIR untuk tipe `IdentityProviderConfiguration`data.

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

Penelepon dapat menetapkan izin di lambda otorisasi. Untuk informasi selengkapnya, lihat OAuth 2.0 cakupan.

Server mengembalikan respons yang menunjukkan dua `Patient` sumber daya yang dibuat sebagai hasil dari permintaan jenis `Bundle` dokumen.

Mengkonfigurasi tingkat validasi untuk bundel

Saat menggabungkan sumber daya FHIR, Anda dapat menentukan header `x-amzn-healthlake-fhir-validation-level` HTTP secara opsional untuk mengonfigurasi tingkat validasi sumber daya. Tingkat validasi ini akan ditetapkan untuk semua permintaan buat dan perbarui dalam bundel. AWS HealthLake saat ini mendukung tingkat validasi berikut:

- `strict`: Sumber daya divalidasi sesuai dengan elemen profil sumber daya, atau spesifikasi R4 jika tidak ada profil. Ini adalah tingkat validasi default untuk AWS HealthLake.
- `structure-only`: Sumber daya divalidasi terhadap R4, mengabaikan profil yang direferensikan.
- `minimal`: Sumber daya divalidasi minimal, mengabaikan aturan R4 tertentu. Sumber daya yang gagal dalam pemeriksaan struktur yang diperlukan search/analytics akan diperbarui untuk menyertakan peringatan untuk audit.

Sumber daya yang dibundel dengan tingkat validasi minimal dapat dicerna ke dalam Datastore meskipun validasi gagal diperlukan untuk pengindeksan pencarian. Dalam hal ini, sumber daya akan diperbarui untuk menyertakan ekstensi khusus Healthlake untuk mendokumentasikan kegagalan tersebut, dan entri dalam respons Bundel akan mencakup OperationOutcome sumber daya sebagai berikut:


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

Selain itu, header respons HTTP berikut akan disertakan dengan nilai “true”:

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

> **Note**
>
> Perhatikan bahwa data yang dicerna yang cacat menurut spesifikasi R4 mungkin tidak dapat dicari seperti yang diharapkan jika kesalahan ini ada.

Dukungan terbatas untuk “pesan” tipe Bundle

HealthLake memberikan dukungan terbatas untuk jenis FHIR Bundle `message` melalui proses konversi internal. Dukungan ini dirancang untuk skenario di mana bundel pesan tidak dapat diformat ulang di sumbernya, seperti menelan umpan ADT (Admission, Discharge, Transfer) dari sistem rumah sakit lama.

> **Warning**
>
> Fitur ini memerlukan daftar izin AWS akun eksplisit dan tidak memberlakukan semantik pesan FHIR R4 atau integritas referensial. Hubungi AWS Support untuk meminta pengaktifan akun Anda sebelum menggunakan bundel pesan.

Perbedaan utama dari pemrosesan pesan standar

- Bundel Pesan (spesifikasi FHIR): Entri pertama harus berupa referensi sumber `MessageHeader` daya lain. Sumber daya tidak memiliki `request` objek individu, dan MessageHeader acara menentukan tindakan pemrosesan.
- HealthLake Pemrosesan: Mengonversi Bundel pesan ke Bundel batch dengan secara otomatis menetapkan operasi PUT ke setiap entri sumber daya. Sumber daya diproses secara independen tanpa menegakkan semantik pesan atau integritas referensial.

Keterbatasan penting

- Aturan pemrosesan khusus pesan FHIR R4 tidak diberlakukan
- Tidak ada integritas transaksional di seluruh sumber daya
- Referensi antar sumber daya tidak divalidasi
- Memerlukan daftar izin akun eksplisit

Contoh struktur Bundle pesan

```
{
              "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**
>
> Setiap sumber daya disimpan secara independen seolah-olah diserahkan melalui operasi PUT individu. Jika semantik pesan FHIR lengkap atau validasi integritas referensial diperlukan, pra-proses Bundel pesan atau terapkan validasi tingkat aplikasi sebelum pengiriman.

Transaksi bundel asinkron

AWS HealthLake mendukung `Bundle` tipe asinkron `transaction` yang memungkinkan Anda mengirimkan transaksi dengan hingga 500 sumber daya. Saat Anda mengirimkan transaksi asinkron, HealthLake antrean untuk diproses dan segera mengembalikan URL polling. Anda dapat menggunakan URL ini untuk memeriksa status dan mengambil respons. Ini mengikuti pola bundel async FHIR.

Kapan menggunakan transaksi asinkron

- Anda perlu mengirimkan lebih dari 100 sumber daya (batas sinkron) dalam satu transaksi.
- Anda ingin menghindari pemblokiran aplikasi Anda sambil menunggu pemrosesan transaksi selesai.
- Anda perlu memproses volume tinggi sumber daya terkait dengan throughput yang lebih baik.

Important

Hasil polling tersedia selama 90 hari setelah transaksi selesai. Setelah periode 90 hari ini, URL polling tidak lagi mengembalikan hasil. Rancang integrasi Anda untuk mengambil dan menyimpan hasil dalam jendela ini.

Note

BundleTipe sinkron `transaction` terus mendukung hingga 100 sumber daya dan merupakan mode pemrosesan default. Jika Anda mengirimkan `Bundle` jenis `transaction` dengan lebih dari 100 sumber daya tanpa `Prefer: respond-async` header, HealthLake mengembalikan `422 Unprocessable Entity` kesalahan. Bundel dengan tipe tidak `batch` didukung untuk pemrosesan asinkron—hanya `Bundle` jenis yang `transaction` dapat dikirimkan secara asinkron (dengan hingga 500 operasi).

Note

PATCHoperasi dan kondisional tidak PUTs didukung dalam transaksi bundel asinkron.

Mengirimkan transaksi asinkron

Untuk mengirimkan transaksi asinkron, kirim POST permintaan ke titik akhir penyimpanan data dengan header. `Prefer: respond-async` Bundel harus memiliki tipe`transaction`. Bundel dengan tipe tidak `batch` didukung untuk pemrosesan bundel asinkron.

HealthLake melakukan validasi awal untuk bundel pada waktu pengiriman. Jika validasi berhasil, HealthLake mengembalikan HTTP 202 Diterima dengan header `content-location` respons yang berisi URL polling.

Untuk mengirimkan tipe asinkron **`Bundle transaction`**

1. Kirim POST permintaan ke titik akhir penyimpanan HealthLake data.

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

2. Buat badan JSON untuk permintaan dengan tipe bundel. `transaction` Untuk tujuan prosedur ini, simpan file sebagai`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. Kirim permintaan dengan `Prefer: respond-async` header. Jenis `Bundle` transaksi FHIR menggunakan `POST` permintaan dengan AWS Signature Version 4 atau SMART pada otorisasi FHIR. Contoh kode berikut menggunakan alat baris `curl` perintah untuk tujuan demonstrasi.

   SigV4

   Otorisasi SiGv4

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

   SMART on FHIR

   SMART pada contoh otorisasi FHIR untuk tipe `IdentityProviderConfiguration`data.

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

Penelepon dapat menetapkan izin di lambda otorisasi. Untuk informasi selengkapnya, lihat OAuth 2.0 cakupan.

4. Pada pengiriman yang berhasil, server mengembalikan HTTP 202 Diterima. Header `content-location` respon berisi URL polling. Badan respons adalah sumber `OperationOutcome` daya.

```
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"
            ]
        }
    ]
}
```

Polling untuk status transaksi

Setelah Anda mengirimkan transaksi asinkron, gunakan URL polling dari header `content-location` respons untuk memeriksa status transaksi. Kirim `GET` permintaan ke URL polling.

> **Note**
>
> Untuk SMART pada penyimpanan data yang diaktifkan FHIR, token otorisasi harus menyertakan `read` izin pada jenis `Transaction` sumber daya untuk melakukan polling untuk status transaksi. Untuk informasi lebih lanjut tentang SMART pada cakupan FHIR, lihat. SMART pada cakupan FHIR OAuth 2.0 didukung oleh HealthLake

Kirim `GET` permintaan ke URL polling. Contoh berikut menggunakan alat baris `curl` perintah.

SigV4

Otorisasi SiGv4

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

SMART on FHIR

SMART pada otorisasi FHIR. Token otorisasi harus menyertakan `read` izin pada jenis `Transaction` sumber daya.

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

Tabel berikut menjelaskan kemungkinan tanggapan.

Kode respons polling

Status HTTP	Arti	Isi respons
202 Diterima	Transaksi antri	`OperationOutcome` dengan diagnostik "DISERAHKAN"
202 Diterima	Transaksi sedang diproses	`OperationOutcome` dengan diagnostik "IN_PROGRESS"
200 OK	Transaksi berhasil diselesaikan	`Bundle`dengan tipe `transaction-response`
4xx/5xx	Transaksi gagal	`OperationOutcome` dengan detail kesalahan

Contoh berikut menunjukkan setiap jenis respons.

Transaksi antri (202)

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

Pemrosesan transaksi (202)

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

```
            "diagnostics": "IN_PROGRESS"
        }
    ]
}
```

Transaksi selesai (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"
            }
        }
    ]
}
```

Transaksi gagal (4xx/5xx)

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

```
    ]
}
```

Memproses pesanan

Jenis bundel asinkron diantrian tetapi tidak `transaction` diproses dalam urutan pengiriman yang ketat. HealthLake mengoptimalkan pemrosesan berdasarkan kapasitas yang tersedia dan beban sistem.

> **Important**
>
> Jangan bergantung pada transaksi yang diproses sesuai urutan yang diajukan. Misalnya, jika Anda mengirimkan Transaksi A pada pukul 10:00 dan Transaksi B pukul 10:01, Transaksi B mungkin selesai sebelum Transaksi A. Rancang aplikasi Anda ke:
>
> - Menangani out-of-order penyelesaian.
> - Gunakan URL polling untuk melacak setiap transaksi secara independen.
> - Terapkan pengurutan tingkat aplikasi jika urutan penting untuk kasus penggunaan Anda.

Kuota dan pelambatan

Kuota dan batas tarif berikut berlaku untuk transaksi asinkron.

Kuota transaksi asinkron

Kuota	Nilai	Dapat Disesuaikan
Operasi maksimum per transaksi asinkron	500	Tidak
Transaksi tertunda maksimum per penyimpanan data	500	Ya

- Transaksi asinkron memiliki batas tarif API yang sama yang ditentukan di bawah. Kuota layanan
- Polling untuk status transaksi memiliki batas tarif API yang sama dengan operasi read (`GET`) pada sumber daya FHIR.
- Jika batas transaksi yang tertunda tercapai, pengiriman berikutnya mengembalikan kesalahan hingga transaksi yang ada selesai.

Penanganan kesalahan

Untuk bundel 'transaksi', semua sumber daya FHIR yang terkandung dalam bundel diproses sebagai operasi atom. Semua sumber daya dalam operasi harus berhasil, atau tidak ada operasi dalam bundel yang diproses.

Kesalahan terbagi dalam dua kategori: kesalahan pengiriman yang HealthLake kembali secara sinkron, dan kesalahan pemrosesan yang Anda ambil melalui polling.

Kesalahan pengiriman

HealthLake memvalidasi bundel pada waktu pengiriman dan mengembalikan kesalahan secara serempak sebelum transaksi antri. Kesalahan pengiriman mencakup kesalahan validasi sumber daya FHIR yang tidak valid, jenis sumber daya yang tidak didukung, melebihi batas operasi 500, dan menggunakan header dengan bundel batch. `Prefer: respond-async` Jika batas transaksi yang tertunda untuk penyimpanan data telah tercapai, HealthLake kembalikan `aThrottlingException`. Ketika kesalahan pengiriman terjadi, transaksi tidak akan antri.

Kesalahan pemrosesan

Kesalahan pemrosesan terjadi setelah transaksi telah antri dan dikembalikan melalui URL polling. Ini termasuk konflik transaksi, di mana operasi lain memodifikasi sumber daya yang merupakan bagian dari transaksi, dan kesalahan server selama pemrosesan. Ketika kesalahan pemrosesan terjadi, tidak ada mutasi sumber daya yang dilakukan untuk sumber daya dalam transaksi. URL polling akan mengembalikan `0peration0utcome` dengan rincian kesalahan.

Menghapus sumber daya FHIR

`delete`Interaksi FHIR menghapus sumber daya FHIR yang ada dari penyimpanan HealthLake data. Untuk informasi tambahan, lihat `delete`di dokumentasi FHIR R4 RESTful API.

Untuk menghapus sumber daya FHIR

1. Kumpulkan HealthLake `region` dan `datastoreId` nilai. Untuk informasi selengkapnya, lihat Mendapatkan properti penyimpanan data.
2. Tentukan jenis FHIR `Resource` untuk menghapus dan mengumpulkan `id` nilai terkait. Untuk informasi selengkapnya, lihat Jenis sumber daya.

3. Buat URL untuk permintaan menggunakan nilai yang dikumpulkan untuk HealthLake `region` dan`datastoreId`. Juga termasuk `Resource` jenis FHIR dan yang terkait`id`. Untuk melihat seluruh jalur URL dalam contoh berikut, gulir ke atas tombol Salin.

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

4. Kirim permintaan . `delete`Interaksi FHIR menggunakan `DELETE` permintaan dengan AWS Signature Version 4 atau SMART pada otorisasi FHIR. `curl`Contoh berikut menghapus `Patient` sumber daya FHIR yang ada dari penyimpanan HealthLake data. Untuk melihat seluruh contoh, gulir ke atas tombol Salin.

   SigV4

   Otorisasi SiGv4

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

   Server mengembalikan kode status `204` HTTP yang mengonfirmasi sumber daya telah dihapus dari penyimpanan HealthLake data. Jika permintaan penghapusan gagal, Anda akan menerima kode status HTTP `400` seri yang menunjukkan mengapa permintaan gagal.

   SMART on FHIR

   SMART pada contoh otorisasi FHIR untuk tipe `IdentityProviderConfiguration`data.

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

Penelepon dapat menetapkan izin di lambda otorisasi. Untuk informasi selengkapnya, lihat OAuth 2.0 cakupan.

AWS Console

1. Masuk ke halaman Jalankan kueri di HealthLake Konsol.
2. Di bawah bagian Pengaturan kueri, buat pilihan berikut.
   - ID Penyimpanan Data — pilih ID penyimpanan data untuk menghasilkan string kueri.
   - Jenis kueri - pilih`Delete`.
   - Jenis sumber daya - pilih jenis sumber daya FHIR untuk dihapus.
   - ID Sumber Daya — masukkan ID sumber daya FHIR.
3. Pilih Run query (Jalankan kueri).

Menghapus sumber daya FHIR berdasarkan kondisi

Penghapusan bersyarat sangat berguna ketika Anda tidak mengetahui ID sumber daya FHIR tertentu tetapi memiliki informasi pengenal lain tentang sumber daya yang ingin Anda hapus.

Penghapusan bersyarat memungkinkan Anda untuk menghapus sumber daya yang ada berdasarkan kriteria pencarian, bukan dengan ID FHIR logis. Saat server memproses permintaan penghapusan, server melakukan pencarian menggunakan kemampuan pencarian standar untuk jenis sumber daya untuk menyelesaikan satu ID logis untuk permintaan tersebut.

Cara kerja penghapusan bersyarat

Tindakan server tergantung pada berapa banyak kecocokan yang ditemukannya:

1. Tidak ada kecocokan: Server mencoba menghapus biasa dan merespons dengan tepat (404 Tidak Ditemukan untuk sumber daya yang tidak ada, 204 Tidak Ada Konten untuk sumber daya yang sudah dihapus)
2. Satu kecocokan: Server melakukan penghapusan biasa pada sumber daya yang cocok
3. Beberapa kecocokan: Mengembalikan kesalahan 412 Prasyarat Gagal yang menunjukkan kriteria klien tidak cukup selektif

Skenario respons

AWS HealthLake menangani operasi penghapusan bersyarat dengan pola respons berikut:

Operasi yang Sukses

- Ketika kriteria pencarian Anda berhasil mengidentifikasi satu sumber daya aktif, sistem mengembalikan 204 Tanpa Konten setelah menyelesaikan penghapusan, seperti operasi penghapusan standar.

Penghapusan Bersyarat Berbasis ID

Saat melakukan penghapusan bersyarat berdasarkan `id` parameter tambahan (`createdAt`,`tag`, atau`_lastUpdated`):

- 204 Tidak Ada Konten: Sumber daya sudah dihapus
- 404 Tidak Ditemukan: Sumber daya tidak ada
- 409 Konflik: ID cocok tetapi parameter lain tidak cocok

Non-ID-Based Hapus Bersyarat

Kapan tidak `id` disediakan atau saat menggunakan parameter selain`createdAt`,`tag`, atau`_lastUpdated`:

- 404 Not Found: Tidak ditemukan kecocokan

Situasi Konflik

Beberapa skenario menghasilkan 412 tanggapan Prasyarat Gagal:

- Beberapa sumber daya cocok dengan kriteria pencarian Anda (kriteria tidak cukup spesifik)
- Konflik versi saat menggunakan ETag header dengan `If-Match`
- Pembaruan sumber daya yang terjadi antara operasi pencarian dan penghapusan

Contoh Penghapusan Bersyarat yang Berhasil

Contoh berikut menghapus sumber daya Pasien berdasarkan kriteria tertentu:

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

Permintaan ini menghapus sumber daya Pasien di mana:

- Nama adalah “Peter”
- Tanggal lahir adalah 1 Januari 2000
- Nomor telepon adalah 1234567890

Praktik Terbaik

1. Gunakan kriteria pencarian khusus untuk menghindari beberapa kecocokan dan mencegah 412 kesalahan.
2. Pertimbangkan ETag header untuk kontrol versi bila diperlukan untuk menangani modifikasi bersamaan.
3. Tangani respons kesalahan dengan tepat:
   - Untuk 404: Perbaiki kriteria pencarian Anda
   - Untuk 412: Buat kriteria lebih spesifik atau selesaikan konflik versi
4. Bersiaplah untuk konflik waktu di lingkungan konkurensi tinggi di mana sumber daya dapat dimodifikasi antara operasi pencarian dan penghapusan.

Idempotensi dan Konkurensi

Kunci Idempotensi

AWS HealthLake mendukung kunci idempotensi untuk `POST` operasi FHIR, menyediakan mekanisme yang kuat untuk memastikan integritas data selama pembuatan sumber daya. Dengan memasukkan

UUID unik sebagai kunci idempotensi di header permintaan, aplikasi perawatan kesehatan dapat menjamin bahwa setiap sumber daya FHIR dibuat tepat sekali, bahkan dalam skenario yang melibatkan ketidakstabilan jaringan atau percobaan ulang otomatis.

Fitur ini sangat penting untuk sistem perawatan kesehatan di mana rekam medis duplikat dapat memiliki konsekuensi serius. Ketika permintaan diterima dengan kunci idempotensi yang sama dengan permintaan sebelumnya, HealthLake akan mengembalikan sumber daya asli alih-alih membuat duplikat. Misalnya, ini dapat terjadi selama loop coba lagi atau karena pipeline permintaan yang berlebihan. Menggunakan kunci idempotensi memungkinkan HealthLake untuk menjaga konsistensi data sambil memberikan pengalaman yang mulus untuk aplikasi klien yang menangani masalah konektivitas intermiten.

Implementasi

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

Skenario Respons

Permintaan Pertama (201 Dibuat)

- Sumber daya baru berhasil dibuat
- Respons termasuk ID sumber daya

Permintaan Duplikat (409 Konflik)

- Kunci idempotensi yang sama terdeteksi
- Sumber daya asli dikembalikan
- Tidak ada sumber daya baru yang dibuat

Permintaan Tidak Valid (400 Permintaan Buruk)

- UUID cacat
- Kolom wajib hilang

Praktik Terbaik

- Hasilkan UUID unik untuk setiap pembuatan sumber daya baru

- Simpan kunci idempotensi untuk logika coba lagi
- Gunakan format kunci yang konsisten: UUID v4 direkomendasikan
- Implementasikan dalam aplikasi klien yang menangani pembuatan sumber daya

> **Note**
>
> Fitur ini sangat berharga untuk sistem perawatan kesehatan yang membutuhkan akurasi data yang ketat dan mencegah duplikat rekam medis.

ETag di AWS HealthLake

AWS HealthLake digunakan ETags untuk kontrol konkurensi optimis dalam sumber daya FHIR, menyediakan mekanisme yang andal untuk mengelola modifikasi bersamaan dan menjaga konsistensi data. An ETag adalah pengidentifikasi unik yang mewakili versi tertentu dari sumber daya, berfungsi sebagai sistem kontrol versi melalui header HTTP. Saat membaca atau memodifikasi sumber daya, aplikasi dapat digunakan ETags untuk mencegah penimpaan yang tidak diinginkan dan memastikan integritas data, terutama dalam skenario dengan potensi pembaruan bersamaan.

Contoh Implementasi

```
// 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
```

Skenario Respons

Operasi yang Berhasil (200 OK atau 204 Tanpa Konten)

- ETag cocok dengan versi saat ini
- Operasi berlangsung sebagaimana dimaksud

Konflik Versi (412 Prasyarat Gagal)

- ETag tidak cocok dengan versi saat ini
- Pembaruan ditolak untuk mencegah kehilangan data

Praktik Terbaik

- Sertakan ETags dalam semua operasi pembaruan dan hapus
- Menerapkan logika coba lagi untuk menangani konflik versi
- Gunakan If-None-Match: * untuk create-if-not-exists skenario
- Selalu verifikasi ETag kesegaran sebelum modifikasi

Sistem kontrol konkurensi ini sangat penting untuk menjaga integritas data perawatan kesehatan, terutama di lingkungan dengan banyak pengguna atau sistem yang mengakses dan memodifikasi sumber daya yang sama.

Mencari sumber daya FHIR di AWS HealthLake

Gunakan `search`interaksi FHIR untuk mencari satu set sumber daya FHIR di penyimpanan HealthLake data berdasarkan beberapa kriteria filter. `search`Interaksi dapat dilakukan dengan menggunakan permintaan `GET` atau `POST` permintaan. Untuk pencarian yang melibatkan informasi identitas pribadi (PII) atau informasi kesehatan yang dilindungi (PHI), disarankan untuk menggunakan `POST` permintaan, karena PII dan PHI ditambahkan sebagai bagian dari badan permintaan dan dienkripsi dalam perjalanan.

> **Note**
>
> `search`Interaksi FHIR yang dijelaskan dalam Bab ini dibangun sesuai dengan standar HL7 FHIR R4 untuk pertukaran data perawatan kesehatan. Karena merupakan representasi dari layanan HL7 FHIR, itu tidak ditawarkan melalui AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat `search`di dokumentasi FHIR R4 RESTful API.

HealthLake mendukung subset parameter pencarian FHIR R4. Lihat informasi yang lebih lengkap di Parameter pencarian FHIR R4 untuk HealthLake.

Topik

- Mencari sumber daya FHIR dengan GET
- Mencari sumber daya FHIR dengan POST
- Tingkat Konsistensi Pencarian FHIR

Mencari sumber daya FHIR dengan GET

Anda dapat menggunakan `GET` permintaan untuk mencari penyimpanan HealthLake data. Saat menggunakan`GET`, HealthLake mendukung penyediaan parameter pencarian sebagai bagian dari URL, tetapi bukan sebagai bagian dari badan permintaan. Untuk informasi selengkapnya, lihat Parameter pencarian FHIR R4 untuk HealthLake.

> **Penting:**
>
> Untuk pencarian yang melibatkan informasi identitas pribadi (PII) atau informasi kesehatan yang dilindungi (PHI), praktik terbaik keamanan memerlukan penggunaan `POST` permintaan,

karena PII dan PHI ditambahkan sebagai bagian dari badan permintaan dan dienkripsi dalam perjalanan. Untuk informasi selengkapnya, lihat Mencari sumber daya FHIR dengan POST.

Prosedur berikut diikuti oleh contoh yang digunakan `GET` untuk mencari penyimpanan HealthLake data.

Untuk mencari penyimpanan HealthLake data dengan **`GET`**

1. Kumpulkan HealthLake `region` dan `datastoreId` nilai. Untuk informasi selengkapnya, lihat Mendapatkan properti penyimpanan data.
2. Tentukan jenis sumber daya FHIR untuk mencari dan mengumpulkan `id` nilai terkait. Untuk informasi selengkapnya, lihat Jenis sumber daya.
3. Buat URL untuk permintaan menggunakan nilai yang dikumpulkan untuk HealthLake `region` dan`datastoreId`. Juga termasuk `Resource` jenis FHIR dan parameter pencarian yang didukung. Untuk melihat seluruh jalur URL dalam contoh berikut, gulir ke atas tombol Salin.

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

4. Kirim `GET` permintaan dengan AWS Signature Version 4 atau SMART pada otorisasi FHIR. `curl`Contoh berikut mengembalikan jumlah total sumber `Patient` daya dalam penyimpanan HealthLake data. Untuk melihat seluruh contoh, gulir ke atas tombol Salin.

   SigV4

   Otorisasi SiGv4

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

   SMART on FHIR

   SMART pada contoh otorisasi FHIR untuk tipe `IdentityProviderConfiguration`data.

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

Penelepon dapat menetapkan izin di lambda otorisasi. Untuk informasi selengkapnya, lihat OAuth 2.0 cakupan.

AWS Console

> **Note**
>
> HealthLake Konsol hanya mendukung otorisasi SiGv4. SMART pada otorisasi FHIR didukung melalui AWS CLI dan. AWS SDKs

1. Masuk ke halaman Jalankan kueri di HealthLake Konsol.

2. Di bawah bagian Pengaturan kueri, buat pilihan berikut.

- ID Penyimpanan Data — pilih ID penyimpanan data untuk menghasilkan string kueri.
- Jenis kueri - pilih`Search with GET`.
- Jenis sumber daya - pilih jenis sumber daya FHIR untuk dicari.
- Parameter pencarian — Pilih parameter pencarian atau kombinasi parameter pencarian untuk memfokuskan kueri Anda pada catatan tertentu.

3. Pilih Run query (Jalankan kueri).

Contoh: cari dengan GET

Tab berikut memberikan contoh untuk mencari pada jenis sumber daya FHIR tertentu dengan. GET Contoh menunjukkan cara menentukan parameter pencarian dalam permintaan URLs.

> **Note**
>
> HealthLake Konsol hanya mendukung otorisasi SiGv4. SMART pada otorisasi FHIR didukung melalui AWS CLI dan. AWS SDKs
> HealthLake mendukung subset parameter pencarian FHIR R4. Untuk informasi selengkapnya, lihat Parameter pencarian.

Patient (age)

Meskipun usia bukan tipe sumber daya yang ditentukan di FHIR, ia ditangkap sebagai elemen dalam tipe `Patient`sumber daya. Gunakan contoh berikut untuk membuat permintaan pencarian GET berbasis pada jenis `Patient`sumber daya menggunakan elemen BirthDate dan pembanding `eq` pencarian untuk mencari individu yang lahir pada tahun 1997.

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

Condition

Gunakan contoh berikut untuk membuat GET permintaan pada jenis `Condition`sumber daya. Pencarian menemukan kondisi di penyimpanan HealthLake data Anda yang berisi kode medis SNOMED`72892002`, yang diterjemahkan menjadi. `Normal pregnancy`

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

DocumentationReference

Contoh berikut menunjukkan cara membuat GET permintaan pada jenis DocumentReferencesumber daya untuk Patient (s) dengan diagnosis streptokokus dan yang juga telah diresepkan amoksisilin.

```
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

Gunakan contoh berikut untuk membuat GET permintaan pada jenis Locationsumber daya. Pencarian berikut menemukan lokasi di toko HealthLake data Anda yang berisi nama kota Boston sebagai bagian dari alamat.

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

Observation

Gunakan contoh berikut untuk membuat permintaan pencarian GET berbasis pada jenis Observationsumber daya. Pencarian ini menggunakan parameter value-concept pencarian untuk mencari kode medis266919005, yang diterjemahkan menjadiNever smoker.

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

Mencari sumber daya FHIR dengan POST

Anda dapat menggunakan searchinteraksi FHIR dengan POST permintaan untuk mencari penyimpanan HealthLake data. Saat menggunakanPOST, HealthLake mendukung parameter pencarian baik di URL atau di badan permintaan, tetapi Anda tidak dapat menggunakan keduanya dalam satu permintaan.

> **Penting:**
>
> Untuk pencarian yang melibatkan informasi identitas pribadi (PII) atau informasi kesehatan yang dilindungi (PHI), praktik terbaik keamanan memerlukan penggunaan POST permintaan,

karena PII dan PHI ditambahkan sebagai bagian dari badan permintaan dan dienkripsi dalam perjalanan.

Prosedur berikut diikuti dengan contoh menggunakan `search` interaksi FHIR R4 dengan `POST` untuk mencari penyimpanan HealthLake data. Contoh menunjukkan cara menentukan parameter pencarian di badan permintaan JSON.

Untuk mencari penyimpanan HealthLake data dengan **`POST`**

1. Kumpulkan HealthLake `region` dan `datastoreId` nilai. Untuk informasi selengkapnya, lihat Mendapatkan properti penyimpanan data.
2. Tentukan jenis sumber daya FHIR untuk mencari dan mengumpulkan `id` nilai terkait. Untuk informasi selengkapnya, lihat Jenis sumber daya.
3. Buat URL untuk permintaan menggunakan nilai yang dikumpulkan untuk HealthLake `region` dan`datastoreId`. Juga termasuk `Resource` jenis dan `_search` interaksi FHIR. Untuk melihat seluruh jalur URL dalam contoh berikut, gulir ke atas tombol Salin.

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

4. Membangun badan JSON untuk permintaan, menentukan data FHIR untuk mencari. Untuk tujuan prosedur ini, Anda akan mencari `Observation` sumber daya untuk menemukan pasien yang tidak pernah merokok. Untuk menentukan status kode medis`Never smoker`, atur `value-concept=266919005` di badan permintaan JSON. Simpan file sebagai `search-observation.json`.

```
value-concept=266919005
```

5. Kirim permintaan . `search`Interaksi FHIR menggunakan `GET` permintaan dengan AWS Signature Version 4 atau SMART pada otorisasi FHIR.

> **Note**
>
> Saat membuat `POST` permintaan dengan parameter pencarian di badan permintaan, gunakan `Content-Type: application/x-www-form-urlencoded` sebagai bagian dari header.

`curl`Contoh berikut membuat permintaan pencarian berbasis Pos pada jenis `Observation` sumber daya. Permintaan menggunakan parameter `value-concept`pencarian untuk mencari kode medis `266919005` yang menunjukkan nilai`Never smoker`. Untuk melihat seluruh contoh, gulir ke atas tombol Salin.

SigV4

Otorisasi SiGv4

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

SMART on FHIR

SMART pada contoh otorisasi FHIR untuk tipe `IdentityProviderConfiguration`data.

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

```
}
```

Penelepon dapat menetapkan izin di lambda otorisasi. Untuk informasi selengkapnya, lihat OAuth 2.0 cakupan.

Contoh: cari dengan POST

Tab berikut memberikan contoh untuk mencari pada jenis sumber daya FHIR tertentu dengan. `POST` Contoh menunjukkan cara menentukan permintaan di URLs.

> **Note**
>
> HealthLake Konsol hanya mendukung otorisasi SiGv4. SMART pada otorisasi FHIR didukung melalui AWS CLI dan. AWS SDKs
> HealthLake mendukung subset parameter pencarian FHIR R4. Untuk informasi selengkapnya, lihat Parameter pencarian.

Patient (age)

Meskipun usia bukan tipe sumber daya yang ditentukan di FHIR, ia ditangkap sebagai elemen dalam tipe `Patient`sumber daya. Gunakan contoh berikut untuk membuat permintaan pencarian `POST` berbasis pada jenis `Patient` sumber daya. Contoh pencarian berikut menggunakan pembanding `eq` pencarian untuk mencari individu yang lahir pada tahun 1997.

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

Untuk menentukan tahun 1997 dalam pencarian, tambahkan elemen berikut ke badan permintaan.

```
birthdate=eq1997
```

Condition

Menggunakan berikut ini untuk membuat `POST` permintaan pada jenis `Condition` sumber daya. Pencarian ini menemukan lokasi di penyimpanan HealthLake data Anda yang berisi kode medis`72892002`.

Anda harus menentukan URL permintaan dan badan permintaan. Berikut adalah contoh URL permintaan.

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

Untuk menentukan kode medis yang ingin Anda cari, Anda menambahkan elemen JSON berikut ke badan permintaan.

```
code=72892002
```

DocumentReference

Untuk melihat hasil pemrosesan HealthLake bahasa alami terintegrasi (NLP) saat membuat POST permintaan pada jenis `DocumentReference` sumber daya, format permintaan sebagai berikut.

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

Untuk menentukan parameter `DocumentReference` pencarian untuk referensi, lihatCari jenis parameter. String kueri berikut menggunakan beberapa parameter penelusuran untuk mencari di operasi Amazon Comprehend Medical API yang digunakan untuk menghasilkan hasil NLP terintegrasi.

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

Location

Gunakan contoh berikut untuk membuat POST permintaan pada jenis `Location` sumber daya. Pencarian menemukan lokasi di toko HealthLake data Anda yang berisi nama kota Boston sebagai bagian dari alamat.

Anda harus menentukan URL permintaan dan badan permintaan. Berikut adalah contoh URL permintaan.

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

Untuk menentukan `Boston` dalam pencarian, tambahkan elemen berikut ke badan permintaan:

```
address=Boston
```

Observation

Gunakan contoh berikut untuk membuat permintaan pencarian POST berbasis pada jenis Observation sumber daya. Pencarian menggunakan parameter value-concept pencarian untuk mencari kode medis, 266919005 yang menunjukkanNever smoker.

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

Untuk menentukan status,Never smoker, atur value-concept=266919005 di badan JSON.

```
value-concept=266919005
```

Tingkat Konsistensi Pencarian FHIR

HealthLakeIndeks pencarian AWS beroperasi pada model Konsistensi Akhir untuk GET dan POST dengan operasi SEARCH. Dengan konsistensi akhirnya, jika ada pembaruan indeks penelusuran yang tertunda untuk sumber daya, hasil pencarian mengecualikan versi N-1 sumber daya hingga pembaruan indeks selesai.

AWS HealthLake sekarang menyertakan kemampuan untuk memilih bagaimana model konsistensi akan berperilaku untuk sumber daya yang diperbarui. Pengembang dapat menyertakan 'Konsistensi Akhir', perilaku default yang dijelaskan di atas atau 'Konsistensi Kuat'. Konsistensi Kuat akan memungkinkan versi N-1 sumber daya untuk sumber daya dengan pembaruan indeks pencarian yang tertunda untuk disertakan dalam hasil pencarian. Ini dapat digunakan untuk skenario kasus penggunaan di mana semua sumber daya diperlukan dalam hasil bahkan ketika pembaruan indeks pencarian belum selesai. Pelanggan dapat mengontrol perilaku ini menggunakan header x-amz-fhir-history-consistency-level permintaan.

Tingkat konsistensi

Konsistensi kuat

Setel x-amz-fhir-history-consistency-level: strong untuk mengembalikan semua catatan yang cocok, termasuk yang memiliki pembaruan indeks penelusuran yang tertunda. Gunakan opsi ini ketika Anda perlu mencari sumber daya segera setelah pembaruan.

Konsistensi akhirnya

Setel `x-amz-fhir-history-consistency-level: eventual` untuk mengembalikan hanya catatan yang telah menyelesaikan pembaruan indeks pencarian. Ini adalah perilaku default jika tidak ada tingkat konsistensi yang ditentukan.

Contoh penggunaan

1. Saat memperbarui sumber daya:

```
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. Pencarian selanjutnya:

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

Praktik terbaik

- Gunakan konsistensi yang kuat ketika Anda perlu segera mencari sumber daya yang baru diperbarui
- Gunakan konsistensi akhirnya untuk kueri umum di mana visibilitas langsung tidak penting
- Pertimbangkan trade-off antara visibilitas langsung dan potensi dampak kinerja

> **Note**
>
> Pengaturan tingkat konsistensi memengaruhi seberapa cepat sumber daya yang diperbarui muncul di hasil penelusuran tetapi tidak memengaruhi penyimpanan sumber daya yang sebenarnya.
> Menyetel `x-amz-fhir-history-consistency-level` header opsional ke 'kuat' menggandakan konsumsi kapasitas tulis per sumber daya.
> Fitur ini hanya berlaku untuk penyimpanan data yang mengaktifkan riwayat versi (semua datastores yang dibuat setelah 25 Oktober 2024 mengaktifkannya secara default).

Mengekspor data FHIR dengan AWS HealthLake

Gunakan AWS HealthLake tindakan asli untuk memulai, mendeskripsikan, dan membuat daftar pekerjaan ekspor FHIR. Anda dapat mengantri pekerjaan ekspor. Pekerjaan ekspor asinkron diproses dengan cara FIFO (First In First Out). Anda dapat mengantri pekerjaan dengan cara yang sama Anda memulai pekerjaan ekspor. Jika sedang berlangsung, itu hanya akan mengantri. Anda dapat membuat, membaca, memperbarui, atau menghapus sumber daya FHIR saat pekerjaan ekspor sedang berlangsung.

> **Note**
>
> Anda juga dapat mengekspor data FHIR dari penyimpanan HealthLake data menggunakan operasi FHIR `$export` R4. Lihat informasi yang lebih lengkap di Mengekspor HealthLake data dengan FHIR `$export`.

Topik

- Memulai pekerjaan ekspor FHIR
- Mendapatkan properti pekerjaan ekspor FHIR
- Listing lowongan kerja ekspor FHIR

Memulai pekerjaan ekspor FHIR

Gunakan `StartFHIRExportJob` untuk memulai pekerjaan ekspor FHIR dari penyimpanan HealthLake data. Menu berikut memberikan prosedur untuk contoh Konsol Manajemen AWS dan kode untuk AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat `StartFHIRExportJob` di dalam Referensi API AWS HealthLake .

> **Catatan**
>
> HealthLake mendukung spesifikasi FHIR R4 untuk pertukaran data perawatan kesehatan. Oleh karena itu, semua data kesehatan diekspor dalam format FHIR R4.

Untuk memulai pekerjaan ekspor FHIR

Pilih menu berdasarkan preferensi akses Anda AWS HealthLake.

AWS CLI dan SDKs

CLI

AWS CLI

Untuk memulai pekerjaan ekspor FHIR

`start-fhir-export-job`Contoh berikut menunjukkan bagaimana memulai pekerjaan ekspor FHIR menggunakan AWS HealthLake.

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

Output:

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

- Untuk detail API, lihat Memulai FHIRExport Job di Referensi AWS CLI Perintah.

Python

SDK untuk 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
```

- Untuk detail API, lihat Memulai FHIRExport Job di AWS SDK for Python (Boto3) Referensi API.

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

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

- Untuk detail API, lihat Memulai FHIRExport Job di AWS SDK untuk referensi API SAP ABAP.

> **Ketersediaan contoh**
>
> Tidak dapat menemukan apa yang Anda butuhkan? Minta contoh kode menggunakan tautan Berikan umpan balik di bilah sisi kanan halaman ini.

AWS Konsol

1. Masuk ke halaman Penyimpanan data di HealthLake Konsol.
2. Pilih penyimpanan data.
3. Pilih Ekspor.

   Halaman Ekspor terbuka.
4. Di bawah bagian Data keluaran, masukkan informasi berikut:

   - Lokasi data keluaran di Amazon S3
   - Ensikripsi keluaran
5. Di bawah bagian Izin akses, pilih Gunakan peran layanan IAM yang ada dan pilih peran dari menu Nama peran atau pilih Buat peran IAM.

6. Pilih Ekspor data.

> **Note**
>
> Selama ekspor, pilih Salin ID pekerjaan pada spanduk di bagian atas halaman. Anda dapat menggunakan `JobID`untuk meminta properti pekerjaan ekspor menggunakan file AWS CLI. Lihat informasi yang lebih lengkap di Mendapatkan properti pekerjaan ekspor FHIR.

Mendapatkan properti pekerjaan ekspor FHIR

Gunakan `DescribeFHIRExportJob` untuk mendapatkan properti pekerjaan ekspor dari penyimpanan HealthLake data. Menu berikut memberikan prosedur untuk contoh Konsol Manajemen AWS dan kode untuk AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat `DescribeFHIRExportJob` di dalam Referensi API AWS HealthLake .

> **Catatan**
>
> HealthLake mendukung spesifikasi FHIR R4 untuk pertukaran data perawatan kesehatan. Oleh karena itu, semua data kesehatan diekspor dalam format FHIR R4.

Untuk menggambarkan pekerjaan ekspor FHIR

Pilih menu berdasarkan preferensi akses Anda AWS HealthLake.

AWS CLI dan SDKs

CLI

AWS CLI

Untuk menggambarkan pekerjaan ekspor FHIR

`describe-fhir-export-job`Contoh berikut menunjukkan bagaimana menemukan properti pekerjaan ekspor FHIR di AWS HealthLake.

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

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

Output:

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

- Untuk detail API, lihat Menjelaskan FHIRExport Pekerjaan dalam Referensi AWS CLI Perintah.

Python

SDK untuk 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
```

- Untuk detail API, lihat Menjelaskan FHIRExport Pekerjaan dalam AWS SDK untuk Referensi API Python (Boto3).

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

- Untuk detail API, lihat Menjelaskan FHIRExport Job in AWS SDK untuk referensi SAP ABAP API.

> **Ketersediaan contoh**
>
> Tidak dapat menemukan apa yang Anda butuhkan? Minta contoh kode menggunakan tautan Berikan umpan balik di bilah sisi kanan halaman ini.

AWS Konsol

> **Note**
>
> Informasi pekerjaan ekspor FHIR tidak tersedia di HealthLake Konsol. Sebagai gantinya, gunakan AWS CLI with `DescribeFHIRExportJob` untuk meminta properti pekerjaan ekspor seperti `JobStatus`. Untuk informasi lebih lanjut, lihat AWS CLI contoh di halaman ini.

Listing lowongan kerja ekspor FHIR

Gunakan `ListFHIRExportJobs` untuk daftar pekerjaan ekspor FHIR untuk penyimpanan HealthLake data. Menu berikut memberikan prosedur untuk contoh Konsol Manajemen AWS dan kode untuk AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat `ListFHIRExportJobs` di dalam Referensi API AWS HealthLake .

> **Catatan**
>
> HealthLake mendukung spesifikasi FHIR R4 untuk pertukaran data perawatan kesehatan. Oleh karena itu, semua data kesehatan diekspor dalam format FHIR R4.

Untuk mencantumkan lowongan kerja ekspor FHIR

Pilih menu berdasarkan preferensi akses Anda AWS HealthLake.

AWS CLI dan SDKs

CLI

AWS CLI

Untuk mencantumkan semua pekerjaan ekspor FHIR

`list-fhir-export-jobs`Contoh berikut menunjukkan cara menggunakan perintah untuk melihat daftar pekerjaan ekspor yang terkait dengan akun.

```
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)
```

Output:

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

- Untuk detail API, lihat Daftar FHIRExport Pekerjaan di Referensi AWS CLI Perintah.

Python

SDK untuk 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
```

- Untuk detail API, lihat Daftar FHIRExport Lowongan di AWS SDK for Python (Boto3) Referensi API.

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        IF iv_submitted_after IS NOT INITIAL.
          oo_result = lo_hll->listfhirexportjobs(
            iv_datastoreid = iv_datastore_id
            iv_submittedafter = iv_submitted_after
          ).
        ELSE.
          oo_result = lo_hll->listfhirexportjobs(
            iv_datastoreid = iv_datastore_id
          ).
        ENDIF.
        DATA(lt_export_jobs) = oo_result->get_exportjobpropertieslist( ).
        DATA(lv_job_count) = lines( lt_export_jobs ).
        MESSAGE |Found { lv_job_count } export job(s).| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- Untuk detail API, lihat Daftar FHIRExport Lowongan di AWS SDK untuk referensi SAP ABAP API.

Ketersediaan contoh

Tidak dapat menemukan apa yang Anda butuhkan? Minta contoh kode menggunakan tautan Berikan umpan balik di bilah sisi kanan halaman ini.

AWS Konsol

Note

Informasi pekerjaan ekspor FHIR tidak tersedia di HealthLake Konsol. Sebagai gantinya, gunakan AWS CLI with `ListFHIRExportJobs` untuk mencantumkan semua pekerjaan ekspor FHIR. Untuk informasi lebih lanjut, lihat AWS CLI contoh di halaman ini.

Contoh kode untuk HealthLake menggunakan AWS SDKs

Contoh kode berikut menunjukkan cara menggunakan HealthLake kit pengembangan AWS perangkat lunak (SDK).

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan HealthLake dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Contoh kode

- Contoh dasar untuk HealthLake menggunakan AWS SDKs
  - Tindakan untuk HealthLake menggunakan AWS SDKs
    - Gunakan CreateFHIRDatastore dengan AWS SDK atau CLI
    - Gunakan DeleteFHIRDatastore dengan AWS SDK atau CLI
    - Gunakan DescribeFHIRDatastore dengan AWS SDK atau CLI
    - Gunakan DescribeFHIRExportJob dengan AWS SDK atau CLI
    - Gunakan DescribeFHIRImportJob dengan AWS SDK atau CLI
    - Gunakan ListFHIRDatastores dengan AWS SDK atau CLI
    - Gunakan ListFHIRExportJobs dengan AWS SDK atau CLI
    - Gunakan ListFHIRImportJobs dengan AWS SDK atau CLI
    - Gunakan ListTagsForResource dengan AWS SDK atau CLI
    - Gunakan StartFHIRExportJob dengan AWS SDK atau CLI
    - Gunakan StartFHIRImportJob dengan AWS SDK atau CLI
    - Gunakan TagResource dengan AWS SDK atau CLI
    - Gunakan UntagResource dengan AWS SDK atau CLI

Contoh dasar untuk HealthLake menggunakan AWS SDKs

Contoh kode berikut menunjukkan cara menggunakan dasar-dasar AWS HealthLake dengan AWS SDKs.

Contoh

- Tindakan untuk HealthLake menggunakan AWS SDKs
  - Gunakan CreateFHIRDatastore dengan AWS SDK atau CLI
  - Gunakan DeleteFHIRDatastore dengan AWS SDK atau CLI
  - Gunakan DescribeFHIRDatastore dengan AWS SDK atau CLI
  - Gunakan DescribeFHIRExportJob dengan AWS SDK atau CLI
  - Gunakan DescribeFHIRImportJob dengan AWS SDK atau CLI
  - Gunakan ListFHIRDatastores dengan AWS SDK atau CLI
  - Gunakan ListFHIRExportJobs dengan AWS SDK atau CLI
  - Gunakan ListFHIRImportJobs dengan AWS SDK atau CLI
  - Gunakan ListTagsForResource dengan AWS SDK atau CLI
  - Gunakan StartFHIRExportJob dengan AWS SDK atau CLI
  - Gunakan StartFHIRImportJob dengan AWS SDK atau CLI
  - Gunakan TagResource dengan AWS SDK atau CLI
  - Gunakan UntagResource dengan AWS SDK atau CLI

Tindakan untuk HealthLake menggunakan AWS SDKs

Contoh kode berikut menunjukkan bagaimana melakukan HealthLake tindakan individu dengan AWS SDKs. Setiap contoh menyertakan tautan ke GitHub, di mana Anda dapat menemukan instruksi untuk mengatur dan menjalankan kode.

Contoh berikut hanya mencakup tindakan yang paling umum digunakan. Untuk daftar lengkapnya, lihat Referensi AWS HealthLake API.

Contoh

- Gunakan CreateFHIRDatastore dengan AWS SDK atau CLI
- Gunakan DeleteFHIRDatastore dengan AWS SDK atau CLI
- Gunakan DescribeFHIRDatastore dengan AWS SDK atau CLI
- Gunakan DescribeFHIRExportJob dengan AWS SDK atau CLI
- Gunakan DescribeFHIRImportJob dengan AWS SDK atau CLI
- Gunakan ListFHIRDatastores dengan AWS SDK atau CLI
- Gunakan ListFHIRExportJobs dengan AWS SDK atau CLI

- Gunakan ListFHIRImportJobs dengan AWS SDK atau CLI
- Gunakan ListTagsForResource dengan AWS SDK atau CLI
- Gunakan StartFHIRExportJob dengan AWS SDK atau CLI
- Gunakan StartFHIRImportJob dengan AWS SDK atau CLI
- Gunakan TagResource dengan AWS SDK atau CLI
- Gunakan UntagResource dengan AWS SDK atau CLI

Gunakan **`CreateFHIRDatastore`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`CreateFHIRDatastore`.

CLI

AWS CLI

Contoh 1: Buat penyimpanan data berkemampuan SIGV4 HealthLake

`create-fhir-datastore`Contoh berikut menunjukkan cara membuat penyimpanan data baru di AWS HealthLake.

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

Output:

```
{
    "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)"
}
```

Contoh 2: Buat SMART di penyimpanan data berkemampuan FHIR HealthLake

`create-fhir-datastore`Contoh berikut menunjukkan cara membuat SMART baru di penyimpanan data berkemampuan FHIR di. AWS HealthLake

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

Isi dari `identity_provider_configuration.json`:

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

Output:

```
{
    "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)"
}
```

- Untuk detail API, lihat Membuat FHIRDatastore di Referensi AWS CLI Perintah.

Python

SDK untuk 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
```

Kode berikut menunjukkan contoh parameter untuk SMART pada penyimpanan data berkemampuan FHIR HealthLake .

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

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

- Untuk detail API, lihat Membuat FHIRDatastore di AWS SDK for Python (Boto3) Referensi API.

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

- Untuk detail API, lihat Create FHIRDatastore in AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan HealthLake dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **`DeleteFHIRDatastore`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`DeleteFHIRDatastore`.

CLI

AWS CLI

Untuk menghapus penyimpanan data FHIR

`delete-fhir-datastore`Contoh berikut menunjukkan cara menghapus penyimpanan data dan semua isinya di AWS HealthLake.

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

Output:

```
{
    "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)"
}
```

- Untuk detail API, lihat Menghapus FHIRDatastore di Referensi AWS CLI Perintah.

Python

SDK untuk 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
```

- Untuk detail API, lihat Menghapus FHIRDatastore di AWS SDK for Python (Boto3) Referensi API.

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

- Untuk detail API, lihat Menghapus FHIRDatastore di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan HealthLake dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **`DescribeFHIRDatastore`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`DescribeFHIRDatastore`.

CLI

AWS CLI

Untuk menggambarkan penyimpanan data FHIR

`describe-fhir-datastore`Contoh berikut menunjukkan bagaimana menemukan properti penyimpanan data di AWS HealthLake.

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

Output:

```
{
    "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
        }
    }
}
```

- Untuk detail API, lihat Menjelaskan FHIRDatastore di Referensi AWS CLI Perintah.

Python

SDK untuk 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
```

- Untuk detail API, lihat Menjelaskan FHIRDatastore dalam AWS SDK for Python (Boto3) Referensi API.

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

- Untuk detail API, lihat Menjelaskan FHIRDatastore di AWS SDK untuk referensi API SAP ABAP.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan HealthLake dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **`DescribeFHIRExportJob`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`DescribeFHIRExportJob`.

CLI

AWS CLI

Untuk menggambarkan pekerjaan ekspor FHIR

`describe-fhir-export-job`Contoh berikut menunjukkan bagaimana menemukan properti pekerjaan ekspor FHIR di AWS HealthLake.

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

Output:

```
{
    "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)"
```

```
    }
}
```

- Untuk detail API, lihat Menjelaskan FHIRExport Pekerjaan dalam Referensi AWS CLI Perintah.

Python

SDK untuk 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
```

- Untuk detail API, lihat Menjelaskan FHIRExport Pekerjaan dalam AWS SDK untuk Referensi API Python (Boto3).

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

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

- Untuk detail API, lihat Menjelaskan FHIRExport Job in AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan HealthLake dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **`DescribeFHIRImportJob`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`DescribeFHIRImportJob`.

CLI

AWS CLI

Untuk menggambarkan pekerjaan impor FHIR

`describe-fhir-import-job`Contoh berikut menunjukkan bagaimana mempelajari properti pekerjaan impor FHIR menggunakan AWS HealthLake.

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

Output:

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

- Untuk detail API, lihat Menjelaskan FHIRImport Pekerjaan dalam Referensi AWS CLI Perintah.

Python

SDK untuk 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
```

- Untuk detail API, lihat Menjelaskan FHIRImport Pekerjaan dalam AWS SDK untuk Referensi API Python (Boto3).

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
TRY.
    " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
    " iv_job_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
    oo_result = lo_hll->describefhirimportjob(
      iv_datastoreid = iv_datastore_id
      iv_jobid = iv_job_id
    ).
    DATA(lo_import_job_properties) = oo_result->get_importjobproperties( ).
    IF lo_import_job_properties IS BOUND.
      DATA(lv_job_status) = lo_import_job_properties->get_jobstatus( ).
      MESSAGE |Import job status: { lv_job_status }.| TYPE 'I'.
    ENDIF.
```

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

- Untuk detail API, lihat Menjelaskan FHIRImport Job in AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan HealthLake dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **`ListFHIRDatastores`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`ListFHIRDatastores`.

CLI

AWS CLI

Untuk daftar toko data FHIR

`list-fhir-datastores`Contoh berikut menunjukkan cara menggunakan perintah dan bagaimana pengguna dapat memfilter hasil berdasarkan status penyimpanan data di AWS HealthLake.

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

Output:

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

- Untuk detail API, lihat Daftar FHIRDatastores di Referensi AWS CLI Perintah.

Python

SDK untuk 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
```

- Untuk detail API, lihat Daftar FHIRDatastores di AWS SDK for Python (Boto3) Referensi API.

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

- Untuk detail API, lihat Daftar FHIRDatastores di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan HealthLake dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **`ListFHIRExportJobs`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`ListFHIRExportJobs`.

CLI

AWS CLI

Untuk mencantumkan semua pekerjaan ekspor FHIR

`list-fhir-export-jobs`Contoh berikut menunjukkan cara menggunakan perintah untuk melihat daftar pekerjaan ekspor yang terkait dengan akun.

```
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)
```

Output:

```
{
    "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)"
            }
```

```
        }
    ]
}
```

- Untuk detail API, lihat Daftar FHIRExport Pekerjaan di Referensi AWS CLI Perintah.

Python

SDK untuk 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
```

- Untuk detail API, lihat Daftar FHIRExport Lowongan di AWS SDK for Python (Boto3) Referensi API.

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        IF iv_submitted_after IS NOT INITIAL.
          oo_result = lo_hll->listfhirexportjobs(
            iv_datastoreid = iv_datastore_id
            iv_submittedafter = iv_submitted_after
          ).
        ELSE.
          oo_result = lo_hll->listfhirexportjobs(
            iv_datastoreid = iv_datastore_id
          ).
        ENDIF.
        DATA(lt_export_jobs) = oo_result->get_exportjobpropertieslist( ).
        DATA(lv_job_count) = lines( lt_export_jobs ).
        MESSAGE |Found { lv_job_count } export job(s).| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- Untuk detail API, lihat Daftar FHIRExport Lowongan di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan HealthLake dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **`ListFHIRImportJobs`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`ListFHIRImportJobs`.

CLI

AWS CLI

Untuk mencantumkan semua pekerjaan impor FHIR

`list-fhir-import-jobs`Contoh berikut menunjukkan cara menggunakan perintah untuk melihat daftar semua pekerjaan impor yang terkait dengan akun.

```
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)
```

Output:

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

- Untuk detail API, lihat Daftar FHIRImport Pekerjaan di Referensi AWS CLI Perintah.

Python

SDK untuk 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
```

- Untuk detail API, lihat Daftar FHIRImport Lowongan di AWS SDK for Python (Boto3) Referensi API.

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        IF iv_submitted_after IS NOT INITIAL.
          oo_result = lo_hll->listfhirimportjobs(
            iv_datastoreid = iv_datastore_id
            iv_submittedafter = iv_submitted_after
          ).
        ELSE.
          oo_result = lo_hll->listfhirimportjobs(
            iv_datastoreid = iv_datastore_id
          ).
        ENDIF.
        DATA(lt_import_jobs) = oo_result->get_importjobpropertieslist( ).
        DATA(lv_job_count) = lines( lt_import_jobs ).
        MESSAGE |Found { lv_job_count } import job(s).| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
```

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

- Untuk detail API, lihat Daftar FHIRImport Lowongan di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan HealthLake dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **`ListTagsForResource`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`ListTagsForResource`.

CLI

AWS CLI

Untuk mencantumkan tag untuk penyimpanan data

`list-tags-for-resource`Contoh berikut mencantumkan tag yang terkait dengan penyimpanan data yang ditentukan. :

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

Output:

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

```
    }
}
```

- Untuk detail API, lihat ListTagsForResourcedi Referensi AWS CLI Perintah.

Python

SDK untuk 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
```

- Untuk detail API, lihat ListTagsForResourcedi AWS SDK for Python (Boto3) Referensi API.

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

- Untuk detail API, lihat ListTagsForResourcedi AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan HealthLake dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **`StartFHIRExportJob`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`StartFHIRExportJob`.

CLI

AWS CLI

Untuk memulai pekerjaan ekspor FHIR

`start-fhir-export-job`Contoh berikut menunjukkan bagaimana memulai pekerjaan ekspor FHIR menggunakan AWS HealthLake.

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

Output:

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

- Untuk detail API, lihat Memulai FHIRExport Job di Referensi AWS CLI Perintah.

Python

SDK untuk 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
```

- Untuk detail API, lihat Memulai FHIRExport Job di AWS SDK for Python (Boto3) Referensi API.

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    TRY.
        " iv_job_name = 'MyExportJob'
        " iv_output_s3_uri = 's3://my-bucket/export/output/'
        " iv_kms_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012'
        " iv_data_access_role_arn = 'arn:aws:iam::123456789012:role/
HealthLakeExportRole'
        oo_result = lo_hll->startfhirexportjob(
          iv_jobname = iv_job_name
          io_outputdataconfig = NEW /aws1/cl_hlloutputdataconfig(
            io_s3configuration = NEW /aws1/cl_hlls3configuration(
              iv_s3uri = iv_output_s3_uri
              iv_kmskeyid = iv_kms_key_id
            )
          )
          iv_dataaccessrolearn = iv_data_access_role_arn
          iv_datastoreid = iv_datastore_id
        ).
```

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

- Untuk detail API, lihat Memulai FHIRExport Job di AWS SDK untuk referensi API SAP ABAP.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan HealthLake dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **`StartFHIRImportJob`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`StartFHIRImportJob`.

CLI

AWS CLI

Untuk memulai pekerjaan impor FHIR

`start-fhir-import-job`Contoh berikut menunjukkan bagaimana memulai pekerjaan impor FHIR menggunakan AWS HealthLake.

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

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

Output:

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

- Untuk detail API, lihat Memulai FHIRImport Job di Referensi AWS CLI Perintah.

Python

SDK untuk 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
```

- Untuk detail API, lihat Memulai FHIRImport Job di AWS SDK for Python (Boto3) Referensi API.

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

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

- Untuk detail API, lihat Memulai FHIRImport Job di AWS SDK untuk referensi API SAP ABAP.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan HealthLake dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **`TagResource`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`TagResource`.

CLI

AWS CLI

Untuk menambahkan tag ke penyimpanan data

`tag-resource`Contoh berikut menunjukkan cara menambahkan tag ke penyimpanan data.

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

Perintah ini tidak menghasilkan output.

.

- Untuk detail API, lihat TagResourcedi Referensi AWS CLI Perintah.

Python

SDK untuk 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
```

- Untuk detail API, lihat TagResourcedi AWS SDK for Python (Boto3) Referensi API.

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

- Untuk detail API, lihat TagResourcedi AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan HealthLake dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **`UntagResource`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`UntagResource`.

CLI

AWS CLI

Untuk menghapus tag dari penyimpanan data.

`untag-resource`Contoh berikut menunjukkan cara menghapus tag dari penyimpanan data.

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

Perintah ini tidak menghasilkan output.

- Untuk detail API, lihat UntagResourcedi Referensi AWS CLI Perintah.

Python

SDK untuk 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
```

- Untuk detail API, lihat UntagResourcedi AWS SDK for Python (Boto3) Referensi API.

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    TRY.
        " iv_resource_arn = 'arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
```

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

- Untuk detail API, lihat UntagResourcedi AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan HealthLake dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Mengintegrasikan AWS HealthLake

AWS Layanan berikut terintegrasi langsung dengan AWS HealthLake untuk memungkinkan pemrosesan bahasa alami terintegrasi, kueri SQL, dan pergudangan data.

- Amazon Comprehend Medical adalah layanan pemrosesan bahasa alami (NLP) yang memenuhi syarat HIPAA yang menggunakan perpustakaan pembelajaran mesin untuk mengekstrak data kesehatan yang bermakna dari teks medis yang tidak terstruktur di. HealthLake Untuk informasi selengkapnya, lihat Panduan Pengembang Medis Amazon Comprehend Medical.
- Amazon Athena adalah layanan kueri interaktif yang memungkinkan Anda menganalisis HealthLake data secara langsung di bucket Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3) menggunakan SQL standar. Untuk informasi selengkapnya, lihat Panduan Pengembang Amazon Athena.

Topik

- Pemrosesan bahasa alami terintegrasi (NLP) untuk HealthLake
- Meminta HealthLake data dengan Amazon Athena

Pemrosesan bahasa alami terintegrasi (NLP) untuk HealthLake

AWS HealthLake menyediakan pustaka pemrosesan bahasa alami (NLP) terintegrasi untuk mengurai, mengidentifikasi, dan memetakan data tidak terstruktur yang disimpan dalam tipe sumber daya FHIR. `DocumentReference`

> **Penting:**
>
> NLP terintegrasi untuk HealthLake dimatikan secara default. Untuk mengaktifkannya, kirimkan kasus dukungan menggunakan AWS Support Center Console. Untuk membuat kasus Anda, masuk ke kasing Anda Akun AWS dan pilih Buat kasus.

HealthLake NLP terintegrasi bekerja dengan memanggil `DetectEntities-V2` Amazon `InferICD10-CM` Comprehend Medical,, dan tindakan API. `InferRxNorm` Tindakan menambahkan hasil mereka sebagai ekstensi ke `DocumentReference` sumber daya. Ketika tindakan Amazon Comprehend Medical API mendeteksi sifat yang,, `DIAGNOSIS` dan/atau`SYMPTOM`, `SIGN` mereka

menghasilkan sumber daya FHIR. `Linkage` Baru `Condition` dan `Observation` sumber daya dibuat dari entitas yang diidentifikasi dengan ciri-ciri `SIGNSYMPTOM`,, atau `DIAGNOSIS` dan mereka terkait dengan dokumen sumber menggunakan `Linkage` sumber daya.

> **Note**
>
> Meskipun `GET` permintaan didukung untuk sumber daya FHIR yang dihasilkan oleh NLP HealthLake terintegrasi, fungsionalitas API `search` FHIR tidak. Untuk mempelajari lebih lanjut tentang mencari ekstensi NLP menggunakan HealthLake integrasi dengan Athena, lihat. Indeks SQL dan kueri

Daftar Isi

- HealthLake Pustaka NLP terintegrasi
- Menggunakan interaksi FHIR REST API
- Parameter pencarian untuk HealthLake NLP terintegrasi
- HealthLake permintaan contoh NLP terintegrasi

HealthLake Pustaka NLP terintegrasi

HealthLake menyimpulkan data yang ditemukan dalam jenis `DocumentReference` sumber daya menggunakan library Amazon Comprehend Medical. Amazon Comprehend Medical API *`DetectEntities-V2`* operasi*`InferICD10-CM`*,, *`InferRxNorm`* dan mendeteksi kondisi medis sebagai ciri. Setiap operasi memberikan wawasan yang berbeda.

> **Dukungan bahasa**
>
> Operasi Amazon Comprehend Medical API hanya mendeteksi entitas medis dalam teks bahasa Inggris.

- DetectEntities-V2: Memeriksa teks klinis untuk berbagai entitas medis dan mengembalikan informasi spesifik tentang mereka, seperti kategori entitas, lokasi, dan skor kepercayaan.
- Menyimpulkan ICD1 0-CM: Mendeteksi kondisi medis dalam catatan pasien sebagai entitas, dan menghubungkan entitas tersebut dengan pengidentifikasi konsep yang dinormalisasi dalam basis

pengetahuan ICD-10-CM dari Pusat Statistik Kesehatan Nasional CDC di bawah otorisasi oleh Organisasi Kesehatan Dunia (WHO).

- InferRxNorm: Mendeteksi obat sebagai entitas yang terdaftar dalam catatan pasien, dan menghubungkannya dengan pengidentifikasi konsep yang dinormalisasi dalam RxNorm database dari Perpustakaan Kedokteran Nasional.

Ciri-ciri yang didukung untuk setiap operasi API adalah`SIGN`,`SYMPTOM`, dan`DIAGNOSIS`. Jika sifat terdeteksi, mereka ditambahkan sebagai ekstensi yang sesuai dengan FHIR ke lokasi berbeda di penyimpanan data Anda. HealthLake

Lokasi di mana ekstensi ditambahkan.

- `DocumentReference`: Hasil dari operasi Amazon Comprehend Medical API ditambahkan sebagai dokumen ke setiap dokumen yang ditemukan `extension` dalam jenis sumber daya. `DocumentReference` Hasil dalam ekstensi dibagi menjadi dua kelompok. Anda dapat menemukannya di hasil berdasarkan merekaURL.
  - `http://healthlake.amazonaws.com/system-generated-resources/`
    - Ini adalah jenis sumber daya yang telah dibuat atau ditambahkan oleh HealthLake.
  - `http://healthlake.amazonaws.com/aws-cm/`
    - Di mana output mentah operasi Amazon Comprehend Medical API ditambahkan ke penyimpanan data Anda. HealthLake
- `Linkage`: Jenis sumber daya ini ditambahkan atau dibuat sebagai hasil dari NLP terintegrasi. `GET`Permintaan pada spesifik `Linkage` mengembalikan daftar sumber daya yang ditautkan. Untuk mengidentifikasi apakah a `Linkage` ditambahkan oleh HealthLake, cari pasangan `"tag": [{"display": "SYSTEM_GENERATED"}]` nilai kunci tambahan. Untuk mempelajari lebih lanjut tentang spesifikasi FHIR untuk Linkage, lihat `Linkage`di dokumentasi FHIR R4.
- Jenis sumber daya FHIR dihasilkan sebagai hasil dari operasi Amazon Comprehend Medical.
  - `Observation`: termasuk hasil dari `DetectEntities-V2` tindakan Amazon Comprehend Medical API `InferICD10-CM` dan kapan ciri-cirinya atau. `SIGN` `SYMPTOM`
  - `Condition`: termasuk hasil dari `DetectEntities-V2` tindakan `InferICD10-CM` Amazon Comprehend Medical API dan kapan sifatnya. `DIAGNOSIS`
  - `MedicationStatement`: termasuk hasil dari tindakan Amazon Comprehend Medical API. `InferRxNorm`

Menggunakan interaksi FHIR REST API

Secara default, sifat yang terdeteksi oleh operasi Amazon Comprehend Medical API tidak dikembalikan saat membuat permintaan. `GET` Untuk melihat hasil operasi NLP terintegrasi, Anda harus menentukan yang dikenal `ID` untuk jenis sumber daya FHIR berikut.

- `Linkage`
- `Observation`
- `Condition`
- `MedicationStatement`

Hasil tindakan NLP HealthLake terintegrasi di luar jenis `DocumentReference` sumber daya tersedia menggunakan `GET` permintaan yang ditentukan `ID` diketahui berisi hasil dari operasi Amazon Comprehend Medical API.

Parameter pencarian untuk HealthLake NLP terintegrasi

Tabel berikut mencantumkan atribut yang dapat dicari untuk NLP HealthLake terintegrasi.

Parameter pencarian untuk HealthLake NLP

Parameter pencarian	Menemukan kecocokan untuk
Detecte ntities-entity-category	Kategori Entitas dalam DetectEntities subextension dalam Ekstensi CM AWS
Detecte ntities-entity-text	Teks Entitas dalam DetectEntities subextension dalam Ekstensi CM AWS
Detecte ntities-entity-type	Jenis Entitas dalam DetectEntities subextension dalam Ekstensi CM AWS
Detecte ntities-entity-score	Skor Entitas dalam DetectEntities subextension dalam Ekstensi CM AWS
infer-icd10 cm-entity-text	Teks Entitas dalam subextension ICD1 Inf 0CM dalam Ekstensi CM AWS

Parameter pencarian	Menemukan kecocokan untuk
infer-icd10 cm-entity-score	Skor Entitas dalam subextension ICD1 Inf 0CM dalam Ekstensi CM AWS
infer-icd10 cm-entity-concept-code	Kode Konsep Entitas dalam subextension ICD1 Inf 0CM dalam Ekstensi CM AWS
infer-icd10 cm-entity-concept-description	Deskripsi Konsep Entitas dalam subextension ICD1 Inf 0CM dalam Ekstensi CM AWS
infer-icd10 cm-entity-concept-score	Skor Konsep Entitas dalam subextension ICD1 Inf 0CM dalam Ekstensi CM AWS
infer-rxnorm-entity-score	Skor Entitas dalam InferRxNorm subextension dalam Ekstensi CM AWS
infer-rxnorm-entity-text	Teks Entitas dalam InferRxNorm subextension dalam Ekstensi CM AWS
infer-rxnorm-entity-concept-kode	Kode Konsep Entitas dalam InferRxNorm subextens ion dalam Ekstensi CM AWS
infer-rxnorm-entity-concept-deskripsi	Deskripsi Konsep Entitas dalam InferRxNorm subextension dalam Ekstensi CM AWS
infer-rxnorm-entity-concept-skor	Skor Konsep Entitas dalam InferRxNorm subextens ion dalam Ekstensi CM AWS

HealthLake menyediakan pencarian khusus untuk mencocokkan kriteria di mana `EntityText` dan `EntityCategory` merupakan bagian dari entitas yang sama. Tabel berikut menjelaskan parameter pencarian khusus yang didukung oleh HealthLake.

Parameter pencarian

Parameter pencarian	Pertandingan kembali
Detecte ntities-entity-text-category	Jika setidaknya ada satu entitas dalam DetectEnt ities subextension yang cocok dengan EntityText dan EntityCategory.
Detecte ntities-entity-type-score	Jika setidaknya ada satu entitas dalam DetectEnt ities subextension yang cocok dengan EntityType dan EntityScore.
Detecte ntities-entity-text-score	Jika setidaknya ada satu entitas dalam DetectEnt ities subextension yang cocok dengan EntityText dan EntityScore.
Detecte ntities-entity-text-type	Jika setidaknya ada satu entitas dalam DetectEnt ities subextension yang cocok dengan EntityText dan EntityType.
Detecte ntities-entity-category-score	Jika setidaknya ada satu entitas yang cocok dengan EntityCategory dan EntityScore.
infer-icd10 -kode cm-entity-text-concept	Jika setidaknya ada satu entitas dalam sub-ekstensi Infer ICD1 0CM yang cocok dengan EntityText dan setidaknya ada satu ConceptCode untuk entitas yang cocok dengan kode.
infer-icd10 -skor cm-entity-text-concept	Jika setidaknya ada satu entitas dalam sub-ekste nsi Infer ICD1 0CM yang cocok dengan EntityText dan setidaknya ada satu ConceptScore untuk entitas yang cocok dengan skor.
infer-icd10 -konsep-skor cm-entity-concept-description	Jika setidaknya ada satu konsep dalam entitas dalam sub-ekstensi Infer ICD1 0CM yang cocok dengan deskripsi konsep dan ConceptScore.
infer-rxnorm-entity-text-konsep-kode	Jika setidaknya ada satu entitas dalam InferRxNo rm sub-ekstensi yang cocok dengan EntityText dan

Parameter pencarian	Pertandingan kembali
	setidaknya ada satu ConceptCode untuk entitas yang cocok dengan kode.
infer-rxnorm-entity-text-konsep-skor	Jika setidaknya ada satu entitas dalam InferRxNorm sub-ekstensi yang cocok dengan EntityText dan setidaknya ada satu ConceptScore untuk entitas yang cocok dengan skor.
infer-rxnorm-entity-concept-description-concept-score	Jika setidaknya ada satu konsep dalam entitas dalam InferRxNorm sub-ekstensi yang cocok dengan deskripsi konsep dan ConceptScore.

HealthLake permintaan contoh NLP terintegrasi

Contoh 1: **`Patient`** catatan yang dicerna ke dalam penyimpanan HealthLake data

Berikut ini adalah contoh catatan klinis berdasarkan `Patient` pertemuan dengan profesional perawatan kesehatan.

> **Data sintetis**
>
> Teks dalam contoh berikut adalah konten sintetis dan tidak mengandung informasi kesehatan yang dilindungi (PHI).

```
1991-08-31

# Chief Complaint
- Headache
- Sinus Pain
- Nasal Congestion
- Sore Throat
- Pain with Bright Lights
- Nasal Discharge
- Cough

# History of Present Illness
```

```
Jerónimo599 is a 4 month-old non-hispanic white male.

# Social History
Patient has never smoked.

Patient comes from a middle socioeconomic background.

Patient currently has Aetna.

# Allergies
No Known Allergies.

# Medications
No Active Medications.

# Assessment and Plan
Patient is presenting with bee venom (substance), mold (organism), house dust
 mite (organism), animal dander (substance), grass pollen (substance), tree pollen
 (substance), lisinopril, sulfamethoxazole / trimethoprim, fish (substance).

## Plan

The patient was prescribed the following medications:
- astemizole 10 mg oral tablet
- nda020800 0.3 ml epinephrine 1 mg/ml auto-injector
The patient was placed on a careplan:
- self-care interventions (procedure)
```

Sebagai pengingat, informasi ini dikodekan dalam format base64 di sumber daya. `DocumentReference` Ketika dokumen ini dimasukkan ke dalam HealthLake dan operasi Amazon Comprehend Medical API selesai, untuk melihat hasilnya, Anda dapat memulai `GET` dengan permintaan pada jenis sumber daya. `DocumentReference`

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

Jika operasi Amazon Comprehend Medical API berhasil, cari pasangan nilai kunci ini di dalam yang ditautkan ke berikut `extension` `"url": "http://healthlake.amazonaws.com/aws-cm/"`

```
{
  "url": "http://healthlake.amazonaws.com/aws-cm/status/",
  "valueString": "SUCCESS"
```

```
},
{
 "url": "http://healthlake.amazonaws.com/aws-cm/message/",
 "valueString": "The AWS HealthLake integrated medical NLP operation was successful."
}
```

Tab berikut menunjukkan kepada Anda bagaimana rekam medis yang dicerna dilaporkan di penyimpanan HealthLake data Anda berdasarkan jenis sumber daya.

DocumentReference

Untuk melihat hasil untuk jenis `DocumentReference` sumber daya tunggal, buat `GET` permintaan di `id` mana sumber daya tertentu disediakan.

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

Ketika berhasil, Anda mendapatkan kode respons `200` HTTP, dan respons JSON berikut (yang telah dipotong untuk kejelasan).

Ini `http://healthlake.amazonaws.com/system-generated-resources/` porsinya. Anda dapat melihat bahwa yang baru `Linkage/e366d29f-2c22-4c19-866e-09603937935a` telah ditambahkan. Anda juga dapat melihat di mana HealthLake telah menambahkan temuan berbasis inferensi ke jenis spesifik `Observation` dan `Condition` sumber daya.

Untuk melihat bagaimana jenis sumber daya ini telah diubah, pilih tab terkait.

```
{
  "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

Untuk melihat hasil untuk jenis `Linkage` sumber daya tunggal, buat `GET` permintaan di `ID` mana sumber daya tertentu disediakan.

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

Ketika berhasil, Anda mendapatkan kode respons `200` HTTP, dan respons JSON terpotong berikut.

Respons berisi `item` elemen. Di dalamnya, pasangan kunci-nilai `"type": "source"` menunjukkan `DocumentReference` entri spesifik yang digunakan untuk memodifikasi `Condition` dan `Observations` terdaftar di bawah pasangan `"type": "alternate"` kunci-nilai.

Anda juga melihat *`meta`* elemen, dan pasangan kunci-nilai yang sesuai,*`"tag": [{"display": "SYSTEM_GENERATED"}]`*, yang menunjukkan sumber daya ini dibuat oleh. HealthLake

```
{
  "resourceType": "Linkage",
  "id": "e366d29f-2c22-4c19-866e-09603937935a",
  "active": true,
  "item":
  [
   {
    "type": "alternate",
    "resource": {
     "reference": "Observation/c6e0a3ff-7a17-4d8b-bfd0-d02d7da090c5",
     "type": "Observation"
    }
```

```
   },
   {
    "type": "alternate",
    "resource": {
     "reference": "Condition/9d5c1ef6-f822-4faf-b55f-7c70f2a4aa8d",
     "type": "Condition"
    }
   },
   {
    "type": "source",
    "resource": {
     "reference": "DocumentReference/0e938f03-da7f-4178-acd8-eea9586c46ed",
     "type": "DocumentReference"
    }
   }
  ],
  "meta": {
   "lastUpdated": "2022-10-21T19:38:31.327Z",
   "tag": [{
    "display": "SYSTEM_GENERATED"
   }]
  }
 }
```

Resource type: Observation

Untuk melihat hasil untuk jenis `Observation` sumber daya tunggal, buat `GET` permintaan di `ID` mana sumber daya tertentu disediakan.

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

Hasil operasi Amazon Comprehend Medical API diubah menjadi elemen-elemen berikut:,, dan. `code` `meta` `modifierExtension`

`code`

Elemen tipe`CodeableConcept`. Untuk mempelajari lebih lanjut, lihat `CodeableConcept`di dokumentasi FHIR R4.

HealthLake menambahkan tiga pasangan kunci-nilai berikut.

- `"system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/"`: Di mana URL mengacu pada operasi Amazon Comprehend Medical API tertentu. Dalam hal ini, Inferensi ICD1 0CM.
- `"code": "A52.06"`: Di `A52.06` mana kode ICD-10-CM yang mengidentifikasi konsep yang ditemukan dalam basis pengetahuan dari Pusat Pengendalian Penyakit.
- `"display": "Other syphilitic heart involvement"`: Di `"Other syphilitic heart involvement"` mana deskripsi panjang kode ICD-10-CM dalam ontologi.

Respon JSON terpotong berikut hanya berisi elemen. `code`

```
"code": {
  "coding":
  [
    {
   "system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/",
   "code": "A52.06",
   "display": "Other syphilitic heart involvement"
    }
  ],
  "text": "Other syphilitic heart involvement"
 }
```

Untuk memahami keyakinan model bahwa kode ICD-10-CM yang ditetapkan sudah benar, gunakan elemen. `modifierExtension`

`meta`

`meta`Elemen berisi metadata yang menunjukkan apakah `code` elemen berisi detail yang telah ditambahkan oleh operasi Amazon Comprehend Medical API.

Respon JSON terpotong berikut hanya berisi elemen. `meta`

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

`modifierExtension`

`modifierExtension`Elemen berisi rincian lebih lanjut tentang tingkat kepercayaan dari kode yang ditetapkan ditemukan dalam `code` elemen. Ini juga memiliki pasangan kunci-nilai yang menyediakan tautan kembali ke aslinya yang DocumentReference digunakan untuk menghasilkan hasil dan jenis sumber daya Linkage terkait.

Untuk setiap `coding` elemen yang ditambahkan, Anda akan melihat `entity-score` dan `entity-Concept-Score` ditambahkan ke ModifierExtension. Untuk setiap nilai dalam pasangan kunci-nilai, Anda melihat skor. Sebab`entity-score`, skor ini adalah tingkat kepercayaan yang dimiliki Amazon Comprehend Medical dalam keakuratan deteksi. Sebab`entity-Concept-Score`, skor ini adalah tingkat kepercayaan yang dimiliki Amazon Comprehend Medical bahwa entitas tersebut secara akurat terkait dengan konsep ICD-10-CM.

Respon JSON terpotong berikut hanya berisi elemen. `modifierExtension`

```
"modifierExtension": [{
   "url": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/aws-cm-icd10-entity-
score",
   "valueDecimal": 0.45005733
  },
  {
   "url": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/aws-cm-icd10-entity-
Concept-Score",
   "valueDecimal": 0.1111792
  },
  {
   "url": "http://healthlake.amazonaws.com/system-generated-linkage",
   "valueReference": {
    "reference": "Linkage/e366d29f-2c22-4c19-866e-09603937935a"
   }
  },
  {
   "url": "http://healthlake.amazonaws.com/source-document-reference",
   "valueReference": {
    "reference": "DocumentReference/0e938f03-da7f-4178-acd8-eea9586c46ed"
   }
  }
 ]
```

Respon JSON Penuh

```
{
  "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

Untuk melihat hasil untuk jenis `Condition` sumber daya tunggal, buat `GET` permintaan di `ID` mana sumber daya tertentu disediakan.

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

Hasil operasi Amazon Comprehend Medical API diubah menjadi elemen-elemen berikut:, , dan. `code` `meta` `modifierExtension`

`code`

Elemen tipe`CodeableConcept`. Untuk mempelajari lebih lanjut, lihat `CodeableConcept`di dokumentasi FHIR R4.

HealthLake menambahkan tiga pasangan kunci-nilai berikut.

- `"system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/"`: Di mana URL mengacu pada operasi Amazon Comprehend Medical API tertentu. Dalam hal ini, Inferensi ICD1 0CM.
- `"code": "I70.0"`: Di `A52.06` mana kode ICD-10-CM yang mengidentifikasi konsep yang ditemukan dalam basis pengetahuan dari Pusat Pengendalian Penyakit.
- `"display": "Atherosclerosis of aorta"`: Di `"Other syphilitic heart involvement"` mana deskripsi panjang kode ICD-10-CM dalam ontologi.

Respon JSON terpotong berikut hanya berisi elemen. `code`

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

Untuk memahami keyakinan model bahwa kode ICD-10-CM yang ditetapkan sudah benar, gunakan elemen. `modifierExtension`

`meta`

`meta`Elemen berisi metadata yang menunjukkan apakah `code` elemen berisi detail yang telah ditambahkan oleh operasi Amazon Comprehend Medical API.

Respon JSON terpotong berikut hanya berisi elemen. `meta`

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

`modifierExtension`

`modifierExtension`Elemen berisi rincian lebih lanjut tentang tingkat kepercayaan dari kode yang ditetapkan ditemukan dalam `code` elemen. Ini juga memiliki pasangan kunci-nilai yang menyediakan tautan kembali ke aslinya yang DocumentReference digunakan untuk menghasilkan hasil dan jenis sumber daya Linkage terkait.

Untuk setiap `coding` elemen yang ditambahkan, Anda akan melihat `entity-score` dan `entity-Concept-Score` ditambahkan ke ModifierExtension. Untuk setiap nilai dalam pasangan kunci-nilai, Anda melihat skor. Sebab`entity-score`, skor ini adalah tingkat kepercayaan yang dimiliki Amazon Comprehend Medical dalam keakuratan deteksi. Sebab`entity-Concept-Score`, skor ini adalah tingkat kepercayaan yang dimiliki Amazon Comprehend Medical bahwa entitas tersebut secara akurat terkait dengan konsep ICD-10-CM.

Respon JSON terpotong berikut hanya berisi elemen. `modifierExtension`

```
"modifierExtension": [{
   "url": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/aws-cm-icd10-entity-
score",
   "valueDecimal": 0.94417894
  },
  {
   "url": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/aws-cm-icd10-entity-
Concept-Score",
```

```
   "valueDecimal": 0.8458298
  },
  {
   "url": "http://healthlake.amazonaws.com/system-generated-linkage",
   "valueReference": {
    "reference": "Linkage/e366d29f-2c22-4c19-866e-09603937935a"
   }
  },
  {
   "url": "http://healthlake.amazonaws.com/source-document-reference",
   "valueReference": {
    "reference": "DocumentReference/0e938f03-da7f-4178-acd8-eea9586c46ed"
   }
  }
 ]
```

Respon JSON Penuh

```
{
  "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"
 }
```

Contoh 2: A **`DocumentReference`** yang berisi tipe **`MedicationStatement`** sumber daya

Berikut adalah contoh catatan klinis berdasarkan pertemuan pasien dengan seorang profesional medis.

> ⚠ Data sintetis
>
> Teks dalam contoh ini adalah konten sintetis dan tidak mengandung informasi kesehatan yang dilindungi (PHI).

```
Tom is not prescribed Advil
```

Tab berikut menunjukkan bagaimana rekam medis yang dicerna dilaporkan di penyimpanan HealthLake data Anda berdasarkan jenis sumber daya.

DocumentReference

Untuk melihat hasil untuk jenis `DocumentReference` sumber daya tunggal, buat `GET` permintaan di `ID` mana sumber daya tertentu disediakan.

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

Ketika berhasil, Anda mendapatkan kode respons `200` HTTP dan respons JSON terpotong berikut.

Pasangan kunci-nilai,`"url": "http://healthlake.amazonaws.com/system-generated-resources/"`, menunjukkan bahwa jenis sumber daya di dalamnya `extension` telah ditambahkan oleh operasi Amazon Comprehend Medical API. Anda dapat melihat jenis `Linkage` sumber daya baru, dan beberapa `MedicationStatement` sumber daya.

```
"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

Untuk melihat hasil untuk jenis `Linkage` sumber daya tunggal, buat `GET` permintaan di `ID` mana sumber daya tertentu disediakan.

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

Ketika berhasil, Anda mendapatkan kode respons `200` HTTP dan respons JSON berikut.

Respons berisi `item` elemen. Di dalamnya, pasangan kunci-nilai `"type": "source"` menunjukkan `DocumentReference` entri spesifik yang digunakan untuk memodifikasi jenis `MedicationStatement` sumber daya.

Anda juga dapat melihat `meta` elemen dan pasangan kunci-nilai yang sesuai`"tag": [{"display": "SYSTEM_GENERATED"}]`, yang menunjukkan bahwa sumber daya ini dibuat oleh. HealthLake

```
{
  "resourceType": "Linkage",
  "id": "394bb244-177b-4409-8657-26b20ed56dd7",
  "active": true,
  "item": [{
    "type": "alternate",
    "resource": {
     "reference": "MedicationStatement/cbf6af10-b0b9-451c-bdde-99611e3498a8",
     "type": "MedicationStatement"
    }
   },
   {
    "type": "alternate",
```

```
    "resource": {
     "reference": "MedicationStatement/9a89b0d3-6681-45ca-9926-27951edce5c7",
     "type": "MedicationStatement"
    }
   },
   {
    "type": "alternate",
    "resource": {
     "reference": "MedicationStatement/4a01f6c8-5f3a-4122-80ab-405312f96aa2",
     "type": "MedicationStatement"
    }
   },
   {
    "type": "alternate",
    "resource": {
     "reference": "MedicationStatement/fbfb77d8-70cf-4579-b4c0-d6fe3c01656b",
     "type": "MedicationStatement"
    }
   },
   {
    "type": "alternate",
    "resource": {
     "reference": "MedicationStatement/1340c9ce-9c48-4bf9-9b2f-d0ab027f5e0b",
     "type": "MedicationStatement"
    }
   },
   {
    "type": "source",
    "resource": {
     "reference": "DocumentReference/c549125d-a218-421f-b8bf-23614c5e796c",
     "type": "DocumentReference"
    }
   }
  ],
  "meta": {
   "lastUpdated": "2022-10-24T20:05:03.501Z",
   "tag": [{
    "display": "SYSTEM_GENERATED"
   }]
  }
 }
```

MedicationStatement

Untuk melihat hasil untuk jenis `MedicationStatement` sumber daya tunggal, buat `GET` permintaan di `ID` mana sumber daya tertentu disediakan.

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

Jenis `MedicationStatement` sumber daya adalah tempat hasil operasi Amazon Comprehend InferRxNorm Medical API ditemukan. Hasilnya diubah menjadi elemen-elemen berikut:`medicationCodeableConcept`,`meta`, dan`modifierExtension`.

medicationCodeableConcept

Elemen tipe`CodeableConcept`. Untuk mempelajari lebih lanjut, lihat `CodeableConcept`di dokumentasi FHIR R4.

HealthLake menambahkan tiga pasangan kunci-nilai berikut.

- `"system": ""http://healthlake.amazonaws.com/aws-cm/infer-rxnorm/`: Di mana URL mengacu pada operasi Amazon Comprehend Medical API tertentu. Dalam hal ini, InferRxNorm.
- `"code": "731533"`: Di mana `731533` adalah ID RxNorm konsep, juga dikenal sebagai RxCui.
- `"display": "ibuprofen 200 MG Oral Capsule [Advil]"`: Di `ibuprofen 200 MG Oral Capsule [Advil]` mana deskripsi RxNorm konsepnya.

Respon JSON terpotong berikut hanya berisi elemen. `MedicationStatement`

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

`meta`

`meta`Elemen berisi metadata yang menunjukkan apakah `code` elemen berisi detail yang telah ditambahkan oleh operasi Amazon Comprehend Medical API.

Respon JSON terpotong berikut hanya berisi elemen. `meta`

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

`modifierExtension`

`modifierExtension`Elemen berisi pasangan kunci-nilai yang menyediakan tautan kembali ke aslinya yang `DocumentReference` digunakan untuk menghasilkan hasil dan jenis sumber daya Linkage terkait.

```
"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"
    }
   }
 ]
```

Meminta HealthLake data dengan Amazon Athena

Selama pekerjaan HealthLake impor, data FHIR JSON bersarang mengalami proses ETL dan disimpan dalam format tabel terbuka Apache Iceberg, di mana setiap jenis sumber daya FHIR

direpresentasikan sebagai tabel individual di Athena. Hal ini memungkinkan pengguna untuk query data FHIR menggunakan SQL, tetapi tanpa harus mengekspornya terlebih dahulu. Ini berharga, karena memberdayakan dokter dan ilmuwan untuk menanyakan data FHIR untuk memvalidasi keputusan mereka atau memajukan penelitian mereka. Untuk informasi selengkapnya tentang bagaimana tabel Apache Iceberg berfungsi di Athena, lihat Menanyakan tabel Apache Iceberg di Panduan Pengguna Athena.

> **Note**
>
> HealthLake mendukung `read` interaksi FHIR R4 pada HealthLake data Anda di Athena. Untuk informasi selengkapnya, lihat Membaca sumber daya FHIR.

Topik di bagian ini menjelaskan cara menghubungkan penyimpanan HealthLake data Anda ke Athena, cara menanyakannya menggunakan SQL, dan cara menghubungkan hasil dengan AWS layanan lain untuk analisis lebih lanjut.

Topik

- Memulai dengan Amazon Athena
- Memeriksa HealthLake data dengan SQL
- Contoh kueri SQL dengan penyaringan kompleks

Memulai dengan Amazon Athena

Untuk berintegrasi HealthLake dengan Amazon Athena, Anda harus mengatur izin. Untuk melakukan ini, Anda akan membuat pengguna, grup, atau peran Athena, dan memberi mereka akses ke sumber daya FHIR yang terletak di dalam penyimpanan data. HealthLake

- Memberikan pengguna, grup, atau akses peran ke penyimpanan HealthLake data (AWS Lake Formation Console)
- Menyiapkan akun Athena

Memberikan pengguna, grup, atau akses peran ke penyimpanan HealthLake data (AWS Lake Formation Console)

> Persona: administrator HealthLake
>
> HealthLake Administrator persona adalah administrator danau data di AWS Lake Formation. Mereka memberikan akses ke penyimpanan HealthLake data di Lake Formation.

Untuk setiap penyimpanan data yang dibuat, ada dua entri yang terlihat di konsol AWS Lake Formation. Satu entri adalah tautan sumber daya. Nama tautan sumber daya selalu ditampilkan dalam huruf miring. Setiap tautan sumber daya ditampilkan dengan nama dan pemilik sumber daya bersama yang ditautkan. Untuk semua penyimpanan HealthLake data, pemilik sumber daya bersama adalah akun HealthLake layanan. Entri lainnya adalah penyimpanan HealthLake data di akun HealthLake layanan. Langkah-langkah dalam prosedur ini menggunakan penyimpanan data yang merupakan tautan sumber daya.

Untuk mempelajari lebih lanjut tentang tautan sumber daya, lihat Cara kerja tautan sumber daya di Lake Formation di Panduan Pengembang AWS Lake Formation.

Agar pengguna, grup, atau peran dapat menanyakan data di Athena, Anda harus memberikan izin Jelaskan pada database sumber daya. Kemudian, Anda harus memberikan Pilih dan Jelaskan pada tabel.

LANGKAH 1: Untuk memberikan izin DESCRIBE pada database tautan sumber daya penyimpanan HealthLake data

1. Buka konsol AWS Lake Formation: https://console.aws.amazon.com/lakeformation/
2. Di bilah navigasi utama, pilih Database.
3. Pada halaman Database, pilih tombol radio di sebelah nama penyimpanan data yang dicetak miring.
4. Pilih Tindakan (▾).
5. PilihIzin.
6. Pada halaman Berikan izin data, di bawah Prinsipal, pilih pengguna atau peran IAM.
7. Di bawah pengguna atau peran IAM, gunakan panah bawah (▾), atau cari pengguna, peran, atau grup IAM yang ingin Anda ajukan kueri di Athena.

8. Di bawah LF-tag atau kartu sumber daya katalog, pilih opsi Sumber daya katalog data bernama.
9. Di bawah Database, gunakan panah bawah (▼) untuk memilih database penyimpanan HealthLake data yang ingin Anda bagikan aksesnya.
10. Di kartu izin tautan Sumber daya, di bawah Izin tautan sumber daya, pilih Jelaskan.

Ketika hibah berhasil, spanduk sukses izin Hibah muncul. Untuk melihat izin yang baru saja Anda berikan, pilih Izin data lake. Temukan pengguna, grup, dan peran dalam tabel. Di bawah kolom Izin, Anda akan melihat Jelaskan terdaftar.

Sekarang Anda harus menggunakan Hibah pada target untuk memberikan Pilih dan Jelaskan pada semua tabel dalam database.

LANGKAH 2: Berikan akses ke semua tabel di tautan sumber daya penyimpanan HealthLake data

1. Buka konsol AWS Lake Formation: https://console.aws.amazon.com/lakeformation/
2. Di bilah navigasi utama, pilih Database.
3. Pada halaman Database, pilih tombol radio di sebelah nama penyimpanan data yang dicetak miring.
4. Pilih Tindakan (▼).
5. Pilih Grant sesuai target.
6. Pada halaman Berikan izin data, di bawah Prinsipal, pilih pengguna atau peran IAM.
7. Di bawah pengguna atau peran IAM, gunakan panah bawah (▼) atau cari pengguna, grup, atau peran IAM yang ingin Anda ajukan kueri di Athena.
8. Di bawah LF-tag atau kartu sumber daya katalog, pilih opsi Sumber daya katalog data bernama.
9. Di bawah Database, gunakan panah bawah (▼) untuk memilih database penyimpanan HealthLake data yang ingin Anda berikan akses.
10. Di bawah Tabel, pilih Semua tabel untuk berbagi semua tabel dengan HealthLake pengguna.
11. Di kartu izin Tabel, di bawah Izin tabel, pilih Jelaskan dan Pilih.
12. PilihIzin.

Setelah memilih hibah, spanduk sukses izin Hibah muncul. Pengguna yang ditentukan sekarang dapat membuat kueri pada penyimpanan HealthLake data di Athena.

Memulai dengan Athena

> **HealthLake pengguna**
>
> HealthLake Pengguna akan menggunakan konsol Athena, AWS CLI, atau AWS SDKs untuk menanyakan penyimpanan HealthLake data yang dibagikan dengan mereka oleh administrator. HealthLake

Untuk menanyakan penyimpanan data menggunakan Athena, Anda harus melakukan tiga hal berikut.

- Berikan pengguna IAM atau akses peran ke penyimpanan HealthLake data melalui Lake Formation. Untuk mempelajari selengkapnya, lihat Memberikan pengguna, grup, atau akses peran ke penyimpanan HealthLake data (AWS Lake Formation Console).
- Buat workgroup untuk penyimpanan HealthLake data Anda.
- Tentukan bucket Amazon S3 untuk menyimpan hasil kueri Anda.

Untuk memulai Athena, tambahkan kebijakan FullAccess AWS terkelola AmazonAthenaFullAccessdan AmazonS3 ke pengguna, grup, atau peran Anda. Menggunakan kebijakan AWS terkelola adalah cara yang bagus untuk mulai menggunakan layanan baru. Perlu diingat bahwa kebijakan yang dikelola AWS mungkin tidak memberikan izin hak istimewa paling sedikit untuk kasus penggunaan spesifik Anda karena tersedia untuk digunakan oleh semua pelanggan AWS. Saat Anda menetapkan izin dengan kebijakan IAM, berikan hanya izin yang diperlukan untuk melakukan tugas. Untuk mempelajari IAM selengkapnya dan menerapkan hak istimewa paling sedikit, lihat Menerapkan izin hak istimewa paling sedikit di Panduan Pengguna IAM.

> **Important**
>
> Untuk menanyakan penyimpanan HealthLake data di Athena, Anda harus menggunakan mesin Athena versi 3.

Kelompok kerja adalah sumber daya, dan oleh karena itu Anda dapat menggunakan kebijakan berbasis IAM untuk mengontrol akses ke grup kerja tertentu. Untuk mempelajari selengkapnya, lihat Menggunakan grup kerja untuk mengontrol akses kueri dan biaya di Panduan Pengguna Athena.

Untuk mempelajari selengkapnya tentang menyiapkan grup kerja, lihat https://docs.aws.amazon.com/athena/latest/ug/workgroups-procedure.html di Panduan Pengguna Athena.

> **Note**
>
> Wilayah bucket Amazon S3 Anda berada dan konsol Athena harus cocok.

Sebelum Anda dapat menjalankan kueri, kueri hasil bucket lokasi di Amazon S3 harus ditentukan, atau Anda harus menggunakan grup kerja yang telah ditentukan bucket dan konfigurasi yang menimpa pengaturan klien. File output disimpan secara otomatis untuk setiap kueri yang berjalan.

Untuk detail selengkapnya tentang menentukan lokasi hasil kueri di konsol Athena, lihat Menentukan lokasi hasil kueri menggunakan konsol Athena di Panduan Pengguna Amazon Athena.

Untuk melihat contoh cara menanyakan penyimpanan HealthLake data Anda di Athena, lihat. Memeriksa HealthLake data dengan SQL

Memeriksa HealthLake data dengan SQL

Saat Anda mengimpor data FHIR Anda ke penyimpanan HealthLake data, data JSON FHIR bersarang secara bersamaan mengalami proses ETL dan disimpan dalam format tabel terbuka Apache Iceberg di Amazon S3. Setiap jenis sumber daya FHIR dari penyimpanan HealthLake data Anda diubah menjadi tabel, yang dapat ditanyakan menggunakan Amazon Athena. Tabel dapat ditanyakan secara individual atau sebagai grup menggunakan kueri berbasis SQL. Karena struktur penyimpanan data, data Anda diimpor ke Athena sebagai beberapa tipe data yang berbeda. Untuk mempelajari selengkapnya tentang membuat kueri SQL yang dapat mengakses tipe data ini, lihat Array kueri dengan tipe kompleks dan struktur bersarang di Panduan Pengguna Amazon Athena.

> **Note**
>
> Semua contoh dalam topik ini menggunakan data fiksi yang dibuat menggunakan Synthea. Untuk mempelajari selengkapnya tentang membuat penyimpanan data yang dimuat sebelumnya dengan data Synthea, lihat. Membuat penyimpanan HealthLake data

Untuk setiap elemen dalam tipe sumber daya, spesifikasi FHIR mendefinisikan kardinalitas. Kardinalitas suatu elemen mendefinisikan batas bawah dan atas berapa kali elemen ini dapat

muncul. Saat membuat kueri SQL, Anda harus mempertimbangkan hal ini. Sebagai contoh, mari kita lihat beberapa elemen dalam tipe Sumber Daya: Pasien.

- Elemen: Nama Spesifikasi FHIR menetapkan kardinalitas sebagai. `0..*`

  Elemen ditangkap sebagai array.

```
[{
 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
}]
```

  Di Athena, untuk melihat bagaimana jenis sumber daya telah dicerna, cari di bawah Tabel dan tampilan. Untuk mengakses elemen dalam array ini, Anda dapat menggunakan notasi titik. Berikut adalah contoh sederhana yang akan mengakses nilai untuk `given` dan`family`.

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

- Elemen: MaritalStatus Spesifikasi FHIR menetapkan kardinalitas sebagai. `0..1`

  Elemen ini ditangkap sebagai JSON.

```
{
 id = null,
 extension = null,
 coding = [
```

```
  {
   id = null,
   extension = null,
   system = http: //terminology.hl7.org/CodeSystem/v3-MaritalStatus,
    _system = null,
   version = null,
   _version = null,
   code = S,
   _code = null,
   display = Never Married,
   _display = null,
   userSelected = null,
   _userSelected = null
  }

 ],
 text = Never Married,
 _text = null
}
```

Di Athena, untuk melihat bagaimana jenis sumber daya telah dicerna, cari di bawah Tabel dan tampilan. Untuk mengakses pasangan kunci-nilai di JSON, Anda dapat menggunakan notasi titik. Karena ini bukan array, tidak ada indeks array yang diperlukan. Berikut adalah contoh sederhana yang akan mengakses nilai untuk`text`.

```
SELECT
    maritalstatus.text as MaritalStatus
FROM Patient
```

Untuk mempelajari lebih lanjut tentang mengakses dan mencari JSON, lihat Menanyakan JSON di Panduan Pengguna Athena.

Pernyataan kueri Athena Data Manipulation Language (DHTML) didasarkan pada Trino. Athena tidak mendukung semua fitur Trino, dan ada perbedaan yang signifikan. Untuk mempelajari selengkapnya, lihat kueri, fungsi, dan operator DHTML di Panduan Pengguna Amazon Athena.

Selain itu, Athena mendukung beberapa tipe data yang mungkin Anda temui saat membuat kueri penyimpanan data Anda HealthLake . Untuk mempelajari lebih lanjut tentang tipe data di Athena, lihat Jenis data di Amazon Athena di Panduan Pengguna Amazon Athena.

Untuk mempelajari lebih lanjut tentang cara kerja kueri SQL di Athena, lihat referensi SQL untuk Amazon Athena di Panduan Pengguna Amazon Athena.

Setiap tab menunjukkan contoh cara mencari pada jenis sumber daya yang ditentukan dan elemen terkait menggunakan Athena.

Element: Extension

Elemen `extension` ini digunakan untuk membuat bidang kustom di penyimpanan data.

Contoh ini menunjukkan kepada Anda cara mengakses fitur `extension` elemen yang ditemukan dalam tipe `Patient` sumber daya.

Saat penyimpanan HealthLake data Anda diimpor ke Athena, elemen tipe sumber daya diuraikan secara berbeda. Karena struktur variabel `element` is, itu tidak dapat sepenuhnya ditentukan dalam skema. Untuk menangani variabilitas itu, elemen di dalam array dilewatkan sebagai string.

Dalam deskripsi tabel`Patient`, Anda dapat melihat elemen yang `extension` dijelaskan sebagai`array<string>`, yang berarti Anda dapat mengakses elemen array dengan menggunakan nilai indeks. Namun, untuk mengakses elemen string, Anda harus menggunakan`json_extract`.

Berikut adalah entri tunggal dari `extension` elemen yang ditemukan di tabel pasien.

```
[{
  "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"
```

```
 }
]
```

Meskipun ini adalah JSON yang valid, Athena memperlakukannya sebagai string.

Contoh query SQL ini menunjukkan bagaimana Anda dapat membuat tabel yang berisi `patient-mothersMaidenName` dan `patient-birthPlace` elemen. Untuk mengakses elemen-elemen ini, Anda perlu menggunakan indeks array yang berbeda dan `json_extract.`

```
SELECT
    extension[1],
    json_extract(extension[1], '$.valueString') AS MothersMaidenName,
    extension[2],
    json_extract(extension[2], '$.valueAddress.city') AS birthPlace
FROM patient
```

Untuk mempelajari lebih lanjut tentang kueri yang melibatkan JSON, lihat Mengekstrak data dari JSON di Panduan Pengguna Amazon Athena.

Element: birthDate (Age)

Usia bukanlah elemen dari tipe sumber daya Pasien di FHIR. Berikut adalah dua contoh pencarian yang memfilter berdasarkan usia.

Karena usia bukan elemen, kita menggunakan `birthDate` untuk query SQL. Untuk melihat bagaimana elemen telah dicerna ke dalam FHIR, cari nama tabel di bawah Tabel dan tampilan. Anda dapat melihat bahwa itu adalah tipe string.

Contoh 1: Menghitung nilai untuk usia

Dalam contoh query SQL ini, kita menggunakan built-in SQL tool, `current_date` dan `year` untuk mengekstrak komponen-komponen tersebut. Kemudian, kami menguranginya untuk mengembalikan usia sebenarnya pasien sebagai kolom yang disebut`age`.

```
SELECT
 (year(current_date) - year(date(birthdate))) as age
FROM patient
```

Contoh 2: Penyaringan untuk pasien yang lahir sebelum `2019-01-01` dan sedang`male`.

Kueri SQL menunjukkan cara menggunakan `CAST` fungsi untuk mentransmisikan `birthDate` elemen sebagai tipe`DATE`, dan cara memfilter berdasarkan dua kriteria dalam `WHERE` klausa.

Karena elemen dicerna sebagai tipe string secara default, kita harus `CAST` itu sebagai tipe`DATE`. Kemudian Anda dapat menggunakan < operator untuk membandingkannya dengan tanggal yang berbeda,`2019-01-01`. Dengan menggunakan`AND`, Anda dapat menambahkan kriteria kedua ke `WHERE` klausa.

```
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

Contoh ini menunjukkan pencarian lokasi dalam jenis sumber daya Lokasi di mana nama kota adalah Attleboro.

```
SELECT *
FROM Location
WHERE address.city='ATTLEBORO'
LIMIT 10;
```

Element: Age

```
SELECT birthdate
FROM patient
-- we convert birthdate (varchar) to date  > cast that as date too
WHERE CAST(birthdate AS DATE) < CAST('2019-01-01' AS DATE) AND gender = 'male'
```

Resource type: Condition

Kondisi tipe sumber daya menyimpan data diagnosis yang terkait dengan masalah yang telah meningkat ke tingkat kekhawatiran. HealthLakeIntegrated Medical Natural Language Processing (NLP) menghasilkan `Condition` sumber daya baru berdasarkan rincian yang ditemukan dalam jenis DocumentReference sumber daya. Saat sumber daya baru dihasilkan, HealthLake tambahkan tag `SYSTEM_GENERATED` ke `meta` elemen. Contoh kueri SQL ini menunjukkan bagaimana Anda dapat mencari tabel kondisi dan mengembalikan hasil di mana `SYSTEM_GENERATED` hasil telah dihapus.

Untuk mempelajari lebih lanjut tentang Integrated HealthLake Natural Language Processing (NLP), lihat. Pemrosesan bahasa alami terintegrasi (NLP) untuk HealthLake

```
SELECT *
FROM condition
WHERE meta.tag[1] is NULL
```

Anda juga dapat mencari dalam elemen string tertentu untuk memfilter kueri Anda lebih lanjut. `modifierextension`Elemen berisi rincian tentang `DocumentReference` sumber daya yang digunakan untuk menghasilkan satu set kondisi. Sekali lagi, Anda harus menggunakan `json_extract` untuk mengakses elemen JSON bersarang yang dibawa ke Athena sebagai string.

Contoh kueri SQL ini menunjukkan bagaimana Anda dapat mencari semua `Condition` yang telah dihasilkan berdasarkan spesifik. `DocumentReference` Gunakan `CAST` untuk mengatur elemen JSON sebagai string sehingga Anda dapat menggunakan `LIKE` untuk membandingkan.

```
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

Jenis sumber daya, Observasi menyimpan pengukuran dan pernyataan sederhana yang dibuat tentang pasien, perangkat, atau subjek lainnya. HealthLakeIntegrated Natural Language Processing (NLP) menghasilkan `Observation` sumber daya baru berdasarkan detail yang ditemukan dalam sumber daya. `DocumentReference` Contoh kueri SQL ini termasuk `WHERE meta.tag[1] is NULL` dikomentari, yang berarti bahwa `SYSTEM_GENERATED` hasilnya disertakan.

```
SELECT valueCodeableConcept.coding[1].code
FROM Observation
WHERE  valueCodeableConcept.coding[1].code = '266919005'
-- WHERE meta.tag[1] is NULL
```

Kolom ini diimpor sebagai file `struct`. Oleh karena itu, Anda dapat mengakses elemen di dalamnya menggunakan notasi titik.

Resource type: MedicationStatement

MedicationStatement adalah jenis sumber daya FHIR yang dapat Anda gunakan untuk menyimpan detail tentang obat yang telah dikonsumsi, dikonsumsi, atau akan dikonsumsi pasien di masa depan. HealthLakeIntegrated Medical Natural Language Processing (NLP) menghasilkan MedicationStatement sumber daya baru berdasarkan dokumen yang ditemukan dalam jenis DocumentReference sumber daya. Saat sumber daya baru dihasilkan, HealthLake tambahkan tag `SYSTEM_GENERATED` ke `meta` elemen. Contoh kueri SQL ini menunjukkan cara membuat kueri yang memfilter berdasarkan satu pasien dengan menggunakan pengenal mereka dan menemukan sumber daya yang telah ditambahkan oleh HealthLake NLP terintegrasi.

```
SELECT *
FROM medicationstatement
WHERE meta.tag[1].display = 'SYSTEM_GENERATED' AND subject.reference =
 'Patient/0679b7b7-937d-488a-b48d-6315b8e7003b';
```

Untuk mempelajari lebih lanjut tentang Integrated HealthLake Natural Language Processing (NLP), lihat. Pemrosesan bahasa alami terintegrasi (NLP) untuk HealthLake

Contoh kueri SQL dengan penyaringan kompleks

Contoh berikut menunjukkan cara menggunakan kueri SQL Amazon Athena dengan pemfilteran kompleks untuk menemukan data FHIR dari penyimpanan data. HealthLake

Example Buat kriteria penyaringan berdasarkan data demografis

Mengidentifikasi demografi pasien yang benar adalah penting saat membuat kohort pasien. Contoh query ini menunjukkan bagaimana Anda dapat menggunakan Trino dot notasi dan `json_extract` untuk memfilter data di penyimpanan data Anda HealthLake .

```
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;
```

Dengan menggunakan Konsol Athena, Anda dapat mengurutkan dan mengunduh hasilnya lebih lanjut.

Example Buat filter untuk pasien dan kondisi terkait

Contoh query berikut menunjukkan bagaimana Anda dapat menemukan dan mengurutkan semua kondisi terkait untuk pasien yang ditemukan di penyimpanan HealthLake data.

```
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;
```

Anda dapat menggunakan konsol Athena untuk mengurutkan hasil lebih lanjut atau mengunduhnya untuk analisis lebih lanjut.

Example Buat filter untuk pasien dan pengamatan terkait

Contoh kueri berikut menunjukkan bagaimana menemukan dan mengurutkan semua pengamatan terkait untuk pasien yang ditemukan di penyimpanan HealthLake data.

```
SELECT
 patient.id as patientId
    , observation.id as observationId
    , CONCAT(name[1].family, ' ', name[1].given[1]) as name
    , meta.tag[1].display
    , json_extract(modifierextension[1], '$.valueDecimal') AS confidenceScore
    , status
    , category[1].coding[1].code as categoryCode
    , category[1].coding[1].display as categoryDescription
    , code.coding[1].code as observationCode
    , code.coding[1].display as observationDescription
    , effectivedatetime
    , CASE
  WHEN valuequantity.value IS NOT NULL THEN CONCAT(CAST(valuequantity.value AS
 VARCHAR),' ',valuequantity.unit)
       WHEN valueCodeableConcept.coding [ 1 ].code IS NOT NULL THEN
 CAST(valueCodeableConcept.coding [ 1 ].code AS VARCHAR)
       WHEN valuestring IS NOT NULL THEN CAST(valuestring AS VARCHAR)
       WHEN valueboolean IS NOT NULL THEN CAST(valueboolean AS VARCHAR)
       WHEN valueinteger IS NOT NULL THEN CAST(valueinteger AS VARCHAR)
       WHEN valueratio IS NOT NULL THEN CONCAT(CAST(valueratio.numerator.value AS
 VARCHAR),'/',CAST(valueratio.denominator.value AS VARCHAR))
       WHEN valuerange IS NOT NULL THEN CONCAT(CAST(valuerange.low.value AS
 VARCHAR),'-',CAST(valuerange.high.value AS VARCHAR))
       WHEN valueSampledData IS NOT NULL THEN CAST(valueSampledData.data AS VARCHAR)
       WHEN valueTime IS NOT NULL THEN CAST(valueTime AS VARCHAR)
       WHEN valueDateTime IS NOT NULL THEN CAST(valueDateTime AS VARCHAR)
       WHEN valuePeriod IS NOT NULL THEN valuePeriod.start
       WHEN component[1] IS NOT NULL THEN CONCAT(CAST(component[2].valuequantity.value
 AS VARCHAR),' ',CAST(component[2].valuequantity.unit AS VARCHAR),
 '/', CAST(component[1].valuequantity.value AS VARCHAR),'
 ',CAST(component[1].valuequantity.unit AS VARCHAR))
    END AS observationvalue
 , encounter.reference as encounterId
    , encounter.type as encountertype
FROM database.patient, observation
WHERE CONCAT('Patient/', patient.id) = observation.subject.reference
ORDER BY name;
```

Example Buat kondisi penyaringan untuk pasien dan prosedur terkait

Menghubungkan prosedur dengan pasien merupakan aspek penting dari perawatan kesehatan. Contoh query SQL berikut menunjukkan bagaimana menggunakan FHIR `Patient` dan jenis `Procedure` sumber daya untuk mencapai hal ini. Kueri SQL berikut akan mengembalikan semua pasien dan prosedur terkait mereka yang ditemukan di penyimpanan HealthLake data Anda.

```
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;
```

Anda dapat menggunakan konsol Athena untuk mengunduh hasil untuk analisis lebih lanjut atau mengurutkannya untuk lebih memahami hasilnya.

Example Buat kondisi penyaringan untuk pasien dan resep terkait

Melihat daftar obat saat ini yang dikonsumsi pasien adalah penting. Menggunakan Athena, Anda dapat menulis kueri SQL yang menggunakan jenis `MedicationRequest` sumber daya `Patient` dan sumber daya yang ditemukan di penyimpanan data Anda HealthLake .

Kueri SQL berikut bergabung dengan `Patient` dan `MedicationRequest` tabel diimpor ke Athena. Ini juga mengatur resep ke dalam entri masing-masing dengan menggunakan notasi titik.

```
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
```

Anda dapat menggunakan konsol Athena untuk mengurutkan hasil atau mengunduhnya untuk analisis lebih lanjut.

Example Lihat obat yang ditemukan dalam jenis **`MedicationStatement`** sumber daya

Contoh kueri berikut menunjukkan cara mengatur JSON bersarang yang diimpor ke Athena menggunakan SQL. Kueri menggunakan `meta` elemen FHIR untuk menunjukkan kapan obat telah ditambahkan oleh HealthLake pemrosesan bahasa alami terintegrasi (NLP). Ini juga digunakan `json_extract` untuk mencari data di dalam array string JSON. Untuk informasi selengkapnya, lihat Pemrosesan bahasa alami.

```
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;
```

Anda dapat menggunakan konsol Athena untuk mengunduh hasil ini atau mengurutkannya.

Example Filter untuk jenis penyakit tertentu

Contoh ini menunjukkan bagaimana Anda dapat menemukan sekelompok pasien, berusia 18 hingga 75 tahun, yang telah didiagnosis menderita diabetes.

```
SELECT patient.id as patientId,
 condition.id as conditionId,
 CONCAT(name [ 1 ].family, ' ', name [ 1 ].given [ 1 ]) as name,
```

```
(year(current_date) - year(date(birthdate))) AS age,
CASE
 WHEN condition.encounter.reference IS NOT NULL THEN condition.encounter.reference
 WHEN observation.encounter.reference IS NOT NULL THEN observation.encounter.reference
END as encounterId,
CASE
 WHEN condition.encounter.type IS NOT NULL THEN observation.encounter.type
 WHEN observation.encounter.type IS NOT NULL THEN observation.encounter.type
END AS encountertype,
condition.code.coding [ 1 ].code as diagnosisCode,
condition.code.coding [ 1 ].display as diagnosisDescription,
observation.category [ 1 ].coding [ 1 ].code as categoryCode,
observation.category [ 1 ].coding [ 1 ].display as categoryDescription,
observation.code.coding [ 1 ].code as observationCode,
observation.code.coding [ 1 ].display as observationDescription,
effectivedatetime AS observationDateTime,
CASE
     WHEN valuequantity.value IS NOT NULL THEN CONCAT(CAST(valuequantity.value AS
VARCHAR),' ',valuequantity.unit)
     WHEN valueCodeableConcept.coding [ 1 ].code IS NOT NULL THEN
CAST(valueCodeableConcept.coding [ 1 ].code AS VARCHAR)
     WHEN valuestring IS NOT NULL THEN CAST(valuestring AS VARCHAR)
     WHEN valueboolean IS NOT NULL THEN CAST(valueboolean AS VARCHAR)
     WHEN valueinteger IS NOT NULL THEN CAST(valueinteger AS VARCHAR)
     WHEN valueratio IS NOT NULL THEN CONCAT(CAST(valueratio.numerator.value AS
VARCHAR),'/',CAST(valueratio.denominator.value AS VARCHAR))
     WHEN valuerange IS NOT NULL THEN CONCAT(CAST(valuerange.low.value AS
VARCHAR),'-',CAST(valuerange.high.value AS VARCHAR))
     WHEN valueSampledData IS NOT NULL THEN CAST(valueSampledData.data AS VARCHAR)
     WHEN valueTime IS NOT NULL THEN CAST(valueTime AS VARCHAR)
     WHEN valueDateTime IS NOT NULL THEN CAST(valueDateTime AS VARCHAR)
     WHEN valuePeriod IS NOT NULL THEN valuePeriod.start
     WHEN component[1] IS NOT NULL THEN CONCAT(CAST(component[2].valuequantity.value
AS VARCHAR),' ',CAST(component[2].valuequantity.unit AS VARCHAR),
'/', CAST(component[1].valuequantity.value AS VARCHAR),'
',CAST(component[1].valuequantity.unit AS VARCHAR))
   END AS observationvalue,
CASE
 WHEN condition.meta.tag [ 1 ].display = 'SYSTEM GENERATED' THEN 'YES'
 WHEN condition.meta.tag [ 1 ].display IS NULL THEN 'NO'
 WHEN observation.meta.tag [ 1 ].display = 'SYSTEM GENERATED' THEN 'YES'
 WHEN observation.meta.tag [ 1 ].display IS NULL THEN 'NO'
  END AS IsSystemGenerated,
 CAST(
```

```
    json_extract(
      condition.modifierextension [ 1 ],
      '$.valueDecimal'
    ) AS int
  ) AS confidenceScore
FROM database.patient,
 database.condition,
 database.observation
WHERE CONCAT('Patient/', patient.id) = condition.subject.reference
 AND CONCAT('Patient/', patient.id) = observation.subject.reference
   AND (year(current_date) - year(date(birthdate))) >= 18
   AND (year(current_date) - year(date(birthdate))) <= 75
   AND condition.code.coding [ 1 ].display like ('%diabetes%');
```

Sekarang Anda dapat menggunakan konsol Athena untuk mengurutkan hasil atau mengunduhnya untuk analisis lebih lanjut.

Pemantauan AWS HealthLake

Pemantauan dan pencatatan adalah bagian penting dalam menjaga keamanan, keandalan, ketersediaan, dan kinerja AWS HealthLake. AWS menyediakan layanan berikut untuk menonton HealthLake, melaporkan ketika ada sesuatu yang salah, dan mengambil tindakan otomatis bila perlu.

- AWS CloudTrailmenangkap panggilan API dan peristiwa terkait yang dibuat oleh atau atas nama AWS akun Anda dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Anda dapat mengidentifikasi pengguna dan akun mana yang dipanggil AWS, alamat IP sumber dari mana panggilan dilakukan, dan kapan panggilan terjadi. Untuk informasi selengkapnya, silakan lihat Panduan Pengguna AWS CloudTrail.
- Amazon CloudWatch memantau AWS sumber daya Anda dan aplikasi yang Anda jalankan AWS secara real time. Anda dapat mengumpulkan dan melacak metrik, membuat dasbor yang disesuaikan, dan mengatur alarm yang memberi tahu Anda atau mengambil tindakan saat metrik tertentu mencapai ambang batas yang ditentukan. Misalnya, Anda dapat CloudWatch melacak penggunaan CPU atau metrik lain dari EC2 instans Amazon Anda dan secara otomatis meluncurkan instans baru bila diperlukan. Untuk informasi selengkapnya, lihat Panduan CloudWatch Pengguna Amazon.
- Amazon EventBridge adalah layanan bus acara tanpa server yang memudahkan untuk menghubungkan aplikasi Anda dengan data dari berbagai sumber. EventBridge mengirimkan aliran data real-time dari aplikasi Anda sendiri, aplikasi Software-as-a-Service (SaaS), AWS dan layanan dan rute data tersebut ke target seperti Lambda. Hal ini memungkinkan Anda memantau kejadian yang terjadi dalam layanan, dan membangun arsitektur yang didorong kejadian. Untuk informasi selengkapnya, lihat Panduan EventBridge Pengguna Amazon.

Topik

- Pencatatan panggilan HealthLake API menggunakan AWS CloudTrail
- Memantau HealthLake metrik menggunakan Amazon CloudWatch
- Memantau HealthLake peristiwa menggunakan Amazon EventBridge

Pencatatan panggilan HealthLake API menggunakan AWS CloudTrail

AWS HealthLake terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di HealthLake. CloudTrail menangkap semua panggilan API untuk HealthLake sebagai peristiwa. Panggilan yang diambil termasuk panggilan dari HealthLake konsol dan panggilan kode ke operasi HealthLake API. Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman CloudTrail acara secara berkelanjutan ke bucket Amazon S3, termasuk acara untuk. HealthLake Jika Anda tidak mengonfigurasi jejak, Anda masih dapat melihat peristiwa terbaru di CloudTrail konsol dalam Riwayat acara. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat HealthLake, alamat IP dari mana permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan.

Untuk mempelajari selengkapnya CloudTrail, lihat Panduan AWS CloudTrail Pengguna.

AWS HealthLake Informasi di CloudTrail

CloudTrail diaktifkan di AWS akun Anda saat Anda membuat akun. Ketika aktivitas terjadi di HealthLake, aktivitas tersebut dicatat dalam suatu CloudTrail peristiwa bersama dengan peristiwa AWS layanan lainnya dalam riwayat Acara. Anda dapat melihat, mencari, dan mengunduh peristiwa terbaru di akun AWS . Untuk informasi selengkapnya, lihat Melihat Acara dengan Riwayat CloudTrail Acara.

Untuk catatan peristiwa yang sedang berlangsung di AWS akun Anda, termasuk acara untuk HealthLake, buat jejak. Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon S3. Secara default, ketika Anda membuat jejak di konsol tersebut, jejak tersebut diterapkan ke semua Wilayah AWS. Jejak mencatat peristiwa dari semua Wilayah di AWS partisi dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasi AWS layanan lain untuk menganalisis lebih lanjut dan menindaklanjuti data peristiwa yang dikumpulkan dalam CloudTrail log. Untuk informasi selengkapnya, lihat berikut:

- Gambaran Umum untuk Membuat Jejak
- CloudTrail Layanan dan Integrasi yang Didukung
- Mengonfigurasi Notifikasi Amazon SNS untuk CloudTrail
- Menerima File CloudTrail Log dari Beberapa Wilayah dan Menerima File CloudTrail Log dari Beberapa Akun

Semua HealthLake tindakan dicatat oleh CloudTrail dan didokumentasikan dalam Referensi HealthLake API dan dalam Panduan Pengembang ini untuk tindakan yang dilakukan menggunakan FHIR REST API. Misalnya, panggilan ke tindakan berikut menghasilkan entri dalam file CloudTrail log:

- `DescribeFHIRImportJob`
- `DescribeFHIRExportJob`
- `StartFHIRImportJob`
- `ListFHIRImportJobs`
- `StartFHIRExportJob`
- `ListFHIRExportJobs`
- `CreateFHIRDatastore`
- `ListFHIRDatastores`
- `DeleteFHIRDatastore`
- `DescribeFHIRDatastore`
- `UpdateResource`
- `CreateResource`
- `DeleteResource`
- `ReadResource`
- `GetCapabilities`
- `SearchWithGet`
- `SearchWithPost`

Setiap entri peristiwa atau log berisi informasi tentang entitas yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan hal berikut ini:

- Apakah permintaan itu dibuat dengan kredenal pengguna root atau AWS Identity and Access Management (IAM).
- Apakah permintaan tersebut dibuat dengan kredensial keamanan sementara untuk satu peran atau pengguna gabungan.
- Apakah permintaan itu dibuat oleh AWS layanan lain.

Untuk informasi lain, lihat Elemen userIdentity CloudTrail .

Memahami Entri File AWS HealthLake Log

Trail adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai file log ke bucket Amazon S3 yang Anda tentukan. CloudTrail file log berisi satu atau lebih entri log. Peristiwa mewakili permintaan tunggal dari sumber mana pun dan mencakup informasi tentang tindakan yang diminta, tanggal dan waktu tindakan, parameter permintaan, dan sebagainya. CloudTrail file log bukanlah jejak tumpukan yang diurutkan dari panggilan API publik, jadi file tersebut tidak muncul dalam urutan tertentu.

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan `CreateFHIRDatastore` tindakan.


```
{
    "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"
}
```


Memantau HealthLake metrik menggunakan Amazon CloudWatch

Anda dapat memantau HealthLake menggunakan Amazon CloudWatch, yang mengumpulkan data mentah dan memprosesnya menjadi metrik yang dapat dibaca, mendekati waktu nyata. Statistik ini disimpan selama 15 bulan, sehingga Anda dapat menggunakan informasi historis itu dan mendapatkan perspektif yang lebih baik tentang kinerja aplikasi atau layanan web Anda. Anda juga dapat mengatur alarm yang memperhatikan ambang batas tertentu dan mengirim notifikasi atau mengambil tindakan saat ambang batas tersebut terpenuhi. Untuk informasi selengkapnya, lihat Panduan CloudWatch Pengguna Amazon.

> **Note**
>
> Metrik dilaporkan untuk semua HealthLaketindakan asli.

Tabel berikut mencantumkan HealthLake metrik dan dimensi yang dilaporkan. CloudWatch Masing-masing disajikan sebagai jumlah frekuensi untuk rentang data yang ditentukan pengguna.

HealthLake Metrik berikut dilaporkan ke CloudWatch.

HealthLake metrik dilaporkan CloudWatch

Metrik	Deskripsi
Hitungan Panggilan	Jumlah panggilan ke APIs. Ini dapat dilaporkan untuk akun atau penyimpanan data tertentu. Unit: Hitungan Statistik yang Valid: Jumlah, Hitung Dimensi: Operasi, ID datastore, tipe penyimpan an data
Permintaan Berhasil	Jumlah permintaan API yang berhasil. Unit: Count (Jumlah) Statistik yang Valid: Sum, Rata-rata Dimensi: Operasi, ID penyimpanan data, tipe penyimpanan data
Kesalahan Pengguna	Jumlah permintaan yang gagal karena kesalahan pengguna. Unit: Count (Jumlah) Statistik yang Valid: Sum, Rata-rata Dimensi: Operasi, ID penyimpanan data, tipe penyimpanan data
Kesalahan Server	Jumlah permintaan yang gagal karena kesalahan server. Unit: Count (Jumlah)

Metrik	Deskripsi
	Statistik yang Valid: Sum, Rata-rata Dimensi: Operasi, ID penyimpanan data, tipe penyimpanan data
Permintaan Terjorok	Jumlah permintaan yang telah dibatasi. Metrik ini tidak termasuk dalam hitungan kesalahan pengguna atau server. Unit: Count (Jumlah) Statistik yang Valid: Sum, Rata-rata Dimensi: Operasi, ID penyimpanan data, tipe penyimpanan data
Latensi	Waktu yang dibutuhkan dalam milidetik untuk memproses permintaan pengguna. Satuan: Milidetik Statistik yang valid: Minimum, Maksimum, Jumlah, Rata-rata Dimensi: Operasi, ID penyimpanan data, tipe penyimpanan data

HealthLake Dimensi berikut dilaporkan ke CloudWatch.

HealthLake Dimensi dilaporkan CloudWatch

Dimensi	Deskripsi
Operasi	Operasi API yang digunakan dalam permintaan
DataStoreID	ID penyimpanan data yang digunakan dalam permintaan

Dimensi	Deskripsi
DataStoreType	Jenis penyimpanan data yang digunakan dalam permintaan

Anda bisa mendapatkan metrik HealthLake dengan AWS Management Console, the AWS CLI, atau CloudWatch API. Anda dapat menggunakan CloudWatch API melalui salah satu Kit Pengembangan Perangkat Lunak Amazon AWS (SDKs) atau alat CloudWatch API. HealthLake Konsol menampilkan grafik berdasarkan data mentah dari CloudWatch API.

Anda harus memiliki CloudWatch izin yang sesuai untuk dipantau HealthLake . CloudWatch Untuk informasi selengkapnya, lihat Identitas dan manajemen akses untuk Amazon CloudWatch di Panduan CloudWatch Pengguna Amazon.

Melihat HealthLake metrik

Untuk melihat metrik (CloudWatch konsol)

1. Masuk ke Konsol Manajemen AWS dan buka CloudWatchkonsol.
2. Pilih Metrik, pilih Semua Metrik, lalu pilih AWS/. HealthLake
3. Pilih dimensi, pilih nama metrik, lalu pilih Tambahkan ke grafik.
4. Pilih nilai untuk rentang tanggal. Hitungan metrik untuk rentang tanggal yang dipilih akan ditampilkan dalam grafik.

Membuat alarm menggunakan CloudWatch

CloudWatch Alarm mengawasi satu metrik selama periode waktu tertentu, dan melakukan satu atau beberapa tindakan: mengirimkan pemberitahuan ke topik Amazon Simple Notification Service (SNS) atau kebijakan Auto Scaling. Tindakan atau tindakan didasarkan pada nilai metrik relatif terhadap ambang batas tertentu selama sejumlah periode waktu yang Anda tentukan. CloudWatch juga dapat mengirimi Anda pesan SNS ketika alarm berubah status.

> **Note**
>
> CloudWatch alarm memanggil tindakan hanya ketika status berubah dan telah bertahan selama periode yang Anda tentukan.

Untuk melihat metrik (CloudWatch konsol)

1. Masuk ke konsol CloudWatch tersebut.
2. Pilih Alarm, lalu pilih Buat Alarm.
3. Pilih AWS/ HealthLake, lalu pilih metrik.
4. Untuk Rentang Waktu, pilih rentang waktu untuk dipantau, lalu pilih Selanjutnya.
5. Masukkan Nama dan Deskripsi.
6. Untuk Kapan pun, pilih >=, dan ketik nilai maksimum.
7. Jika Anda CloudWatch ingin mengirim email saat status alarm tercapai, di bagian Tindakan, untuk Setiap kali alarm ini, pilih Status adalah ALARM. Untuk Kirim pemberitahuan ke, pilih milis atau pilih Daftar baru dan buat milis baru.
8. Pratinjau alarm di bagian Pratinjau Alarm. Jika Anda puas dengan alarm, pilih Buat Alarm.

Memantau HealthLake peristiwa menggunakan Amazon EventBridge

Amazon EventBridge adalah layanan tanpa server yang menggunakan peristiwa untuk menghubungkan komponen aplikasi bersama-sama, sehingga memudahkan Anda untuk membangun aplikasi berbasis peristiwa yang dapat diskalakan. Dasarnya EventBridge adalah membuat aturan yang mengarahkan peristiwa ke target. AWS HealthLake menyediakan pengiriman perubahan negara yang tahan lama ke EventBridge. Untuk informasi selengkapnya, lihat Apa itu Amazon EventBridge? di Panduan EventBridge Pengguna Amazon.

> **Note**
>
> Untuk mempelajari cara mengirim HealthLake acara ke Amazon EventBridge, lihat EventBridge Integrasi Amazon AWS HealthLake di blog AWS for Industries.

Topik

- HealthLake peristiwa dikirim ke EventBridge
- HealthLake struktur acara

HealthLake peristiwa dikirim ke EventBridge

Tabel berikut mencantumkan semua HealthLake peristiwa yang dikirim EventBridge untuk diproses.

HealthLake jenis acara	Status
Acara penyimpanan data	
Membuat Toko Data	`CREATING`
Toko Data Aktif	`ACTIVE`
Penghapusan Toko Data	`DELETING`
Toko Data Dihapus	`DELETED`
Pembuatan Penyimpanan Data Gagal	`CREATE_FAILED`
Untuk informasi selengkapnya, lihat `datastoreStatus` di dalam Referensi API AWS HealthLake .	
Impor acara pekerjaan	
Impor Job yang Dikirim	`SUBMITTED`
Impor Job Sedang Berlangsung	`IN_PROGRESS`
Impor Job Dilengkapi Dengan Kesalahan	`COMPLETED_WITH_ERRORS`
Impor Job Selesai	`COMPLETED`
Impor Job Gagal	`FAILED`
Untuk informasi selengkapnya, lihat `jobStatus` di dalam Referensi API AWS HealthLake .	
Acara kerja ekspor	
Ekspor Job Dikirim	`SUBMITTED`
Ekspor Job Sedang Berlangsung	`IN_PROGRESS`

HealthLake jenis acara	Status
Ekspor Job Dilengkapi Dengan Kesalahan	`COMPLETED_WITH_ERRORS`
Export Job Selesai	`COMPLETED`
Ekspor Job Gagal	`FAILED`

Untuk informasi selengkapnya, lihat `jobStatus` di dalam Referensi API AWS HealthLake .

HealthLake struktur acara

HealthLake peristiwa adalah objek dengan struktur JSON yang juga berisi detail metadata. Anda dapat menggunakan metadata sebagai masukan untuk membuat ulang acara atau mempelajari informasi selengkapnya. Semua bidang metadata terkait tercantum dalam tabel di bawah contoh kode di menu berikut. Untuk informasi selengkapnya, lihat metadata peristiwa AWS layanan di EventBridge Panduan Pengguna Amazon.

> **Note**
>
> Untuk mempelajari cara mengirim HealthLake acara ke Amazon EventBridge, lihat EventBridge Integrasi Amazon AWS HealthLake di blog AWS for Industries.

Acara penyimpanan data

Data Store Creating

Negara - **`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

Negara - **`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

Negara - **`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

Negara - **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/"
    }
}
```

Acara penyimpanan data - deskripsi metadata

Nama	Tipe	Deskripsi
`version`	string	Versi skema EventBridge acara.
`id`	string	Versi 4 UUID dihasilkan untuk setiap acara.
`detail-type`	string	Jenis acara yang sedang dikirim.
`source`	string	Mengidentifikasi layanan yang menghasilkan peristiwa.
`account`	string	ID akun AWS 12 digit dari pemilik penyimpanan data.
`time`	string	Waktu peristiwa itu terjadi.
`region`	string	Mengidentifikasi AWS Wilayah penyimpanan data.
`resources`	array (string)	Sebuah array JSON yang berisi ARN dari penyimpanan data.
`detail`	object	Objek JSON yang berisi informasi tentang peristiwa.

Nama	Tipe	Deskripsi
`detail.datastoreId`	string	ID penyimpanan data yang terkait dengan peristiwa perubahan status.
`detail.datastoreName`	string	Nama penyimpanan data.
`detail.datastoreTypeVersion`	string	Data menyimpan versi FHIR.
`detail.datastoreEndpoint`	string	Titik akhir penyimpanan data.

Impor acara pekerjaan

Import Job Submitted

Negara - **`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

Negara - **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

Negara - **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

Negara - **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

Negara - **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/"
        }
    }
}
```

Impor acara pekerjaan - deskripsi metadata

Nama	Tipe	Deskripsi
`version`	string	Versi skema EventBridge acara.

Nama	Tipe	Deskripsi
`id`	string	Versi 4 UUID dihasilkan untuk setiap acara.
`detail-type`	string	Jenis acara yang sedang dikirim.
`source`	string	Mengidentifikasi layanan yang menghasilkan peristiwa.
`account`	string	ID akun AWS 12 digit dari pemilik penyimpanan data.
`time`	string	Waktu peristiwa itu terjadi.
`region`	string	Mengidentifikasi AWS Wilayah penyimpanan data.
`resources`	array (string)	Sebuah array JSON yang berisi ARN dari penyimpanan data.
`detail`	object	Objek JSON yang berisi informasi tentang peristiwa.
`detail.jobId`	string	ID pekerjaan impor yang terkait dengan peristiwa perubahan status.
`detail.submitTime`	string	Waktu pekerjaan impor diajukan.
`detail.datastoreId`	string	Penyimpanan data yang menghasilkan peristiwa perubahan status.

Nama	Tipe	Deskripsi
`detail.inputDataConfig`	string	Jalur awalan input untuk bucket Amazon S3 yang berisi file FHIR yang akan diimpor.

Acara kerja ekspor

Export Job Submitted

Negara - **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

Negara - **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

Negara - **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

Negara - **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

Negara - **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/"
        }
    }
}
```

Ekspor acara pekerjaan - deskripsi metadata

Nama	Tipe	Deskripsi
`version`	string	Versi skema EventBridge acara.
`id`	string	Versi 4 UUID dihasilkan untuk setiap acara.
`detail-type`	string	Jenis acara yang sedang dikirim.
`source`	string	Mengidentifikasi layanan yang menghasilkan peristiwa.

Nama	Tipe	Deskripsi
`account`	string	ID akun AWS 12 digit dari pemilik penyimpanan data.
`time`	string	Waktu peristiwa itu terjadi.
`region`	string	Mengidentifikasi AWS Wilayah penyimpanan data.
`resources`	array (string)	Sebuah array JSON yang berisi ARN dari penyimpanan data.
`detail`	object	Objek JSON yang berisi informasi tentang peristiwa.
`detail.jobId`	string	ID pekerjaan ekspor yang terkait dengan peristiwa perubahan status.
`detail.submitTime`	string	Waktu pekerjaan ekspor diajukan.
`detail.datastoreId`	string	Penyimpanan data yang menghasilkan peristiwa perubahan status.
`detail.outputDataConfig`	string	Jalur awalan keluaran untuk bucket Amazon S3 yang berisi file FHIR yang akan diekspor.

Keamanan di AWS HealthLake

Keamanan cloud di AWS adalah prioritas tertinggi. Sebagai AWS pelanggan, Anda mendapat manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. Model tanggung jawab bersama menjelaskan hal ini sebagai keamanan dari cloud dan keamanan dalam cloud:

- Keamanan cloud AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan AWS layanan di AWS Cloud. AWS juga memberi Anda layanan yang dapat Anda gunakan dengan aman. Third-party auditor secara teratur menguji dan memverifikasi efektivitas keamanan kami sebagai bagian dari Program Kepatuhan Program AWS Kepatuhan Program AWS . Untuk mempelajari tentang program kepatuhan yang berlaku HealthLake, lihat AWS Services in Scope by Compliance Program .
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh AWS layanan yang Anda gunakan. Anda juga bertanggung jawab atas faktor lain, yang mencakup sensitivitas data Anda, persyaratan perusahaan Anda, serta undang-undang dan peraturan yang berlaku.

Dokumentasi ini membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan HealthLake. Topik berikut menunjukkan cara mengonfigurasi HealthLake untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga mempelajari cara menggunakan layanan AWS lain yang membantu Anda memantau dan mengamankan HealthLake sumber daya Anda.

Topik

- Perlindungan Data di AWS HealthLake
- Enkripsi di REST untuk AWS HealthLake
- Enkripsi dalam perjalanan untuk AWS HealthLake
- Identitas dan manajemen akses untuk AWS HealthLake
- Validasi kepatuhan untuk AWS HealthLake
- Keamanan infrastruktur di AWS HealthLake
- Menciptakan AWS HealthLake sumber daya dengan AWS CloudFormation
- AWS HealthLake dan antarmuka titik akhir VPC ()AWS PrivateLink
- Praktik terbaik keamanan di AWS HealthLake

- Ketahanan di AWS HealthLake

Perlindungan Data di AWS HealthLake

Model tanggung jawab AWS bersama model berlaku untuk perlindungan data di AWS HealthLake. Seperti yang dijelaskan dalam model AWS ini, bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk mempertahankan kendali atas konten yang di-host pada infrastruktur ini. Anda juga bertanggung jawab atas tugas-tugas konfigurasi dan manajemen keamanan untuk Layanan AWS yang Anda gunakan. Untuk informasi selengkapnya tentang privasi data, lihat FAQ Privasi Data AWS . Untuk informasi tentang perlindungan data di Eropa, lihat Pusat Peraturan Perlindungan Data Umum (GDPR).

Untuk tujuan perlindungan data, kami menyarankan Anda melindungi Akun AWS kredensyal dan mengatur pengguna individu dengan AWS IAM Identity Center atau AWS Identity and Access Management (IAM). Dengan cara itu, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugasnya. Kami juga menyarankan supaya Anda mengamankan data dengan cara-cara berikut:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk berkomunikasi dengan AWS sumber daya. Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Siapkan API dan pencatatan aktivitas pengguna dengan AWS CloudTrail. Untuk informasi tentang penggunaan CloudTrail jejak untuk menangkap AWS aktivitas, lihat Bekerja dengan CloudTrail jejak di AWS CloudTrail Panduan Pengguna.
- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default di dalamnya Layanan AWS.
- Gunakan layanan keamanan terkelola tingkat lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data sensitif yang disimpan di Amazon S3.
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-3 saat mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Lihat informasi selengkapnya tentang titik akhir FIPS yang tersedia di Standar Pemrosesan Informasi Federal (FIPS) 140-3.

Kami sangat merekomendasikan agar Anda tidak pernah memasukkan informasi identifikasi yang sensitif, seperti nomor rekening pelanggan Anda, ke dalam tanda atau bidang isian bebas seperti bidang Nama. Ini termasuk saat Anda bekerja dengan HealthLake atau lainnya Layanan AWS

menggunakan konsol, API AWS CLI, atau AWS SDK. Data apa pun yang Anda masukkan ke dalam tanda atau bidang isian bebas yang digunakan untuk nama dapat digunakan untuk log penagihan atau log diagnostik. Saat Anda memberikan URL ke server eksternal, kami sangat menganjurkan supaya Anda tidak menyertakan informasi kredensial di dalam URL untuk memvalidasi permintaan Anda ke server itu.

Enkripsi di REST untuk AWS HealthLake

HealthLake menyediakan enkripsi secara default untuk melindungi data pelanggan sensitif saat istirahat dengan menggunakan kunci AWS Key Management Service (AWS KMS) yang dimiliki layanan. Customer-managed Kunci KMS juga didukung dan diperlukan untuk mengimpor dan mengekspor file dari penyimpanan data. Untuk mempelajari lebih lanjut tentang Customer-managed KMS Key, lihat Amazon Key Management Service. Pelanggan dapat memilih kunci KMS yang dimiliki AWS atau kunci Customer-managed KMS saat membuat penyimpanan data. Konfigurasi enkripsi tidak dapat diubah setelah penyimpanan data dibuat. Jika penyimpanan data menggunakan Kunci KMS milik AWS, itu akan dilambangkan sebagai AWS_OWNED_KMS_KEY dan Anda tidak akan melihat kunci spesifik yang digunakan untuk enkripsi saat istirahat.

Kunci KMS milik AWS

HealthLake menggunakan kunci ini secara default untuk secara otomatis mengenkripsi informasi yang berpotensi sensitif seperti data yang dapat diidentifikasi secara pribadi atau Informasi Kesehatan Pribadi (PHI) saat istirahat. Kunci KMS yang dimiliki AWS tidak disimpan di akun Anda. Mereka adalah bagian dari kumpulan kunci KMS yang dimiliki dan dikelola AWS untuk digunakan di beberapa akun AWS. Layanan AWS dapat menggunakan kunci KMS milik AWS untuk melindungi data Anda. Anda tidak dapat melihat, mengelola, menggunakan kunci KMS milik AWS, atau mengaudit penggunaannya. Namun, Anda tidak perlu melakukan pekerjaan apa pun atau mengubah program apa pun untuk melindungi kunci yang mengenkripsi data Anda.

Anda tidak dikenakan biaya bulanan atau biaya penggunaan jika Anda menggunakan kunci KMS milik AWS, dan kunci tersebut tidak dihitung terhadap kuota AWS KMS untuk akun Anda. Untuk informasi selengkapnya, lihat kunci yang dimiliki AWS.

Kunci KMS yang dikelola pelanggan

HealthLake mendukung penggunaan kunci KMS terkelola pelanggan simetris yang Anda buat, miliki, dan kelola untuk menambahkan enkripsi lapisan kedua di atas enkripsi milik AWS yang ada. Karena Anda memiliki kontrol penuh atas lapisan enkripsi ini, Anda dapat melakukan tugas-tugas seperti:

- Menetapkan dan memelihara kebijakan utama, kebijakan IAM, dan hibah
- Memutar bahan kriptografi kunci
- Mengaktifkan dan menonaktifkan kebijakan utama
- Menambahkan tanda
- Membuat alias kunci
- Kunci penjadwalan untuk penghapusan

Anda juga dapat menggunakan CloudTrail untuk melacak permintaan yang HealthLake dikirim AWS KMS atas nama Anda. AWS KMS Biaya tambahan.Untuk informasi lebih lanjut, lihat kunci milik pelanggan.

Buat kunci terkelola pelanggan

Anda dapat membuat kunci terkelola pelanggan simetris dengan menggunakan AWS Management Console, atau AWS KMS API.

Ikuti langkah-langkah untuk Membuat kunci terkelola pelanggan simetris di Panduan Pengembang AWS Key Management Service.

Kebijakan utama mengontrol akses ke kunci yang dikelola pelanggan Anda. Setiap kunci yang dikelola pelanggan harus memiliki persis satu kebijakan utama, yang berisi pernyataan yang menentukan siapa yang dapat menggunakan kunci dan bagaimana mereka dapat menggunakannya. Saat membuat kunci terkelola pelanggan, Anda dapat menentukan kebijakan kunci. Untuk informasi selengkapnya, lihat Mengelola akses ke kunci terkelola pelanggan di Panduan Pengembang AWS Key Management Service.

Untuk menggunakan kunci yang dikelola pelanggan dengan HealthLake sumber daya Anda, kms: CreateGrant operasi harus diizinkan dalam kebijakan utama. Ini menambahkan hibah ke kunci terkelola pelanggan yang mengontrol akses ke kunci KMS tertentu, yang memberikan akses pengguna ke kms:grant operations require. HealthLake Lihat Menggunakan hibah untuk informasi lebih lanjut.

Untuk menggunakan kunci KMS yang dikelola pelanggan dengan HealthLake sumber daya Anda, operasi API berikut harus diizinkan dalam kebijakan kunci:

- kms: CreateGrant menambahkan hibah ke kunci KMS yang dikelola pelanggan tertentu yang memungkinkan akses ke operasi hibah.

- kms: DescribeKey memberikan detail kunci yang dikelola pelanggan yang diperlukan untuk memvalidasi kunci. Ini diperlukan untuk semua operasi.
- kms: GenerateDataKey menyediakan akses untuk mengenkripsi sumber daya saat istirahat untuk semua operasi penulisan.
- KMS: Decrypt menyediakan akses untuk membaca atau mencari operasi untuk sumber daya terenkripsi.

Berikut ini adalah contoh pernyataan kebijakan yang memungkinkan pengguna untuk membuat dan berinteraksi dengan penyimpanan data AWS HealthLake yang dienkripsi oleh kunci tersebut:

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

Izin IAM yang diperlukan untuk menggunakan kunci KMS yang dikelola pelanggan

Saat membuat penyimpanan data dengan AWS KMS enkripsi diaktifkan menggunakan kunci KMS yang dikelola pelanggan, ada izin yang diperlukan untuk kebijakan kunci dan kebijakan IAM untuk pengguna atau peran yang membuat penyimpanan data. HealthLake

Anda dapat menggunakan kms: ViaService condition key untuk membatasi penggunaan kunci KMS hanya untuk permintaan yang berasal dari. HealthLake

Untuk informasi selengkapnya tentang kebijakan utama, lihat Mengaktifkan kebijakan IAM di Panduan Pengembang AWS Key Management Service.

Pengguna IAM, peran IAM, atau akun AWS yang membuat repositori Anda harus memiliki izin kms:, kms: CreateGrantGenerateDataKey, dan kms: ditambah izin yang diperlukan. DescribeKey HealthLake

Cara HealthLake menggunakan hibah di AWS KMS

HealthLake membutuhkan hibah untuk menggunakan kunci KMS yang dikelola pelanggan Anda. Saat Anda membuat Penyimpanan Data yang dienkripsi dengan kunci KMS yang dikelola pelanggan, HealthLake buat hibah atas nama Anda dengan mengirimkan CreateGrantpermintaan ke AWS KMS. Hibah di AWS KMS digunakan untuk HealthLake memberikan akses ke kunci KMS di akun pelanggan.

Hibah yang HealthLake dibuat atas nama Anda tidak boleh dicabut atau pensiun. Jika Anda mencabut atau menghentikan hibah yang memberikan HealthLake izin untuk menggunakan kunci AWS KMS di akun Anda, HealthLake tidak dapat mengakses data ini, mengenkripsi sumber daya FHIR baru yang didorong ke penyimpanan data, atau mendekripsi ketika ditarik. Ketika Anda mencabut atau pensiun hibah untuk HealthLake, perubahan terjadi segera. Untuk mencabut hak akses, Anda harus menghapus penyimpanan data daripada mencabut hibah. Ketika penyimpanan data dihapus, HealthLake pensiun hibah atas nama Anda.

Memantau kunci enkripsi Anda untuk HealthLake

Anda dapat menggunakan CloudTrail untuk melacak permintaan yang HealthLake dikirim atas nama Anda saat menggunakan kunci KMS yang dikelola pelanggan. AWS KMS Entri log di CloudTrail log menampilkan healthlake.amazonaws.com di bidang UserAgent untuk membedakan dengan jelas permintaan yang dibuat oleh. HealthLake

Contoh berikut adalah CloudTrail peristiwa untuk CreateGrant,, Dekripsi GenerateDataKey, dan DescribeKey untuk memantau AWS KMS operasi yang dipanggil oleh HealthLake untuk mengakses data yang dienkripsi oleh kunci yang dikelola pelanggan Anda.

Berikut ini menunjukkan cara menggunakan untuk memungkinkan CreateGrant HealthLake untuk mengakses kunci KMS yang disediakan pelanggan, memungkinkan HealthLake untuk menggunakan kunci KMS itu untuk mengenkripsi semua data pelanggan saat istirahat.

Pengguna tidak diharuskan untuk membuat hibah mereka sendiri. HealthLake membuat hibah atas nama Anda dengan mengirimkan CreateGrant permintaan ke AWS KMS. Hibah AWS KMS digunakan untuk memberikan HealthLake akses ke AWS KMS kunci di akun pelanggan.


```
         {
"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"
}
```

Contoh berikut menunjukkan cara menggunakan GenerateDataKey untuk memastikan pengguna memiliki izin yang diperlukan untuk mengenkripsi data sebelum menyimpannya.

```
            {
    "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"
}
```

Contoh berikut menunjukkan cara HealthLake memanggil operasi Dekripsi untuk menggunakan kunci data terenkripsi yang disimpan untuk mengakses data terenkripsi.

```
         {
    "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"
}
```

Contoh berikut menunjukkan cara HealthLake menggunakan DescribeKey operasi untuk memverifikasi apakah AWS KMS kunci milik AWS KMS pelanggan berada dalam keadaan yang dapat digunakan dan untuk membantu pengguna memecahkan masalah jika tidak berfungsi.

```
            {
    "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"
}
```


Pelajari selengkapnya

Sumber daya berikut memberikan informasi lebih lanjut tentang enkripsi data saat istirahat.

Untuk informasi selengkapnya tentang konsep dasar AWS Key Management Service, lihat AWS KMS dokumentasinya.

Untuk informasi selengkapnya tentang praktik terbaik Keamanan dalam AWS KMS dokumentasi.

Enkripsi dalam perjalanan untuk AWS HealthLake

AWS HealthLake menggunakan TLS 1.2 untuk mengenkripsi data dalam perjalanan melalui titik akhir publik dan melalui layanan backend.

Identitas dan manajemen akses untuk AWS HealthLake

AWS Identity and Access Management (IAM) adalah Layanan AWS yang membantu administrator mengontrol akses ke AWS sumber daya dengan aman. Administrator IAM mengontrol siapa yang dapat diautentikasi (masuk) dan diberi wewenang (memiliki izin) untuk menggunakan sumber daya. HealthLake IAM adalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Topik

- Audiens
- Mengautentikasi dengan identitas
- Mengelola akses menggunakan kebijakan
- Bagaimana AWS HealthLake bekerja dengan IAM
- Contoh kebijakan berbasis identitas untuk AWS HealthLake
- AWS kebijakan terkelola untuk AWS HealthLake
- Memecahkan masalah AWS HealthLake identitas dan akses

Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda berdasarkan peran Anda:

- Pengguna layanan - minta izin dari administrator Anda jika Anda tidak dapat mengakses fitur (lihat Memecahkan masalah AWS HealthLake identitas dan akses)
- Administrator layanan - tentukan akses pengguna dan mengirimkan permintaan izin (lihat Bagaimana AWS HealthLake bekerja dengan IAM)
- Administrator IAM - tulis kebijakan untuk mengelola akses (lihat Contoh kebijakan berbasis identitas untuk AWS HealthLake)

Mengautentikasi dengan identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensi identitas Anda. Anda harus diautentikasi sebagai Pengguna root akun AWS, pengguna IAM, atau dengan mengasumsikan peran IAM.

Anda dapat masuk sebagai identitas federasi menggunakan kredensional dari sumber identitas seperti AWS IAM Identity Center (Pusat Identitas IAM), autentikasi masuk tunggal, atau kredensional. Google/Facebook Untuk informasi selengkapnya tentang cara masuk, lihat Cara masuk ke Akun AWS Anda dalam Panduan Pengguna AWS Sign-In .

Untuk akses terprogram, AWS sediakan SDK dan CLI untuk menandatangani permintaan secara kriptografis. Untuk informasi selengkapnya, lihat AWS Signature Version 4 untuk permintaan API dalam Panduan Pengguna IAM.

Akun AWS pengguna root

Saat Anda membuat Akun AWS, Anda mulai dengan satu identitas masuk yang disebut pengguna Akun AWS root yang memiliki akses lengkap ke semua Layanan AWS dan sumber daya. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari. Untuk tugas yang memerlukan kredensial pengguna root, lihat Tugas yang memerlukan kredensial pengguna root dalam Panduan Pengguna IAM.

Identitas terfederasi

Sebagai praktik terbaik, mewajibkan pengguna manusia untuk menggunakan federasi dengan penyedia identitas untuk mengakses Layanan AWS menggunakan kredensi sementara.

Identitas federasi adalah pengguna dari direktori perusahaan Anda, penyedia identitas web, atau Directory Service yang mengakses Layanan AWS menggunakan kredensi dari sumber identitas. Identitas terfederasi mengambil peran yang memberikan kredensial sementara.

Untuk manajemen akses terpusat, kami menyarankan AWS IAM Identity Center. Untuk informasi selengkapnya, lihat Apa itu Pusat Identitas IAM? dalam Panduan Pengguna AWS IAM Identity Center .

Pengguna dan grup IAM

Pengguna IAM adalah identitas dengan izin khusus untuk satu orang atau aplikasi. Sebaiknya gunakan kredensial sementara alih-alih pengguna IAM dengan kredensial jangka panjang. Untuk

informasi selengkapnya, lihat Mewajibkan pengguna manusia untuk menggunakan federasi dengan penyedia identitas untuk mengakses AWS menggunakan kredensi sementara di Panduan Pengguna IAM.

Grup IAM menentukan kumpulan pengguna IAM dan mempermudah pengelolaan izin untuk pengguna dalam jumlah besar. Untuk mempelajari selengkapnya, lihat Kasus penggunaan untuk pengguna IAM dalam Panduan Pengguna IAM.

Peran IAM

Peran IAM adalah identitas dengan izin khusus yang menyediakan kredensial sementara. Anda dapat mengambil peran dengan beralih dari pengguna ke peran IAM (konsol) atau dengan memanggil operasi AWS CLI atau AWS API. Untuk informasi selengkapnya, lihat Metode untuk mengambil peran dalam Panduan Pengguna IAM.

Peran IAM berguna untuk akses pengguna terfederasi, izin pengguna IAM sementara, akses lintas akun, akses lintas layanan, dan aplikasi yang berjalan di Amazon EC2. Untuk informasi selengkapnya, lihat Akses sumber daya lintas akun di IAM dalam Panduan Pengguna IAM.

Mengelola akses menggunakan kebijakan

Anda mengontrol akses AWS dengan membuat kebijakan dan melampirkannya ke AWS identitas atau sumber daya. Kebijakan menentukan izin saat dikaitkan dengan identitas atau sumber daya. AWS mengevaluasi kebijakan ini ketika kepala sekolah membuat permintaan. Sebagian besar kebijakan disimpan AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang dokumen kebijakan JSON, lihat Gambaran umum kebijakan JSON dalam Panduan Pengguna IAM.

Menggunakan kebijakan, administrator menentukan siapa yang memiliki akses ke apa dengan mendefinisikan principal mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Secara default, pengguna dan peran tidak memiliki izin. Administrator IAM membuat kebijakan IAM dan menambahkannya ke peran, yang kemudian dapat diambil oleh pengguna. Kebijakan IAM mendefinisikan izin terlepas dari metode yang Anda gunakan untuk melakukan operasinya.

Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang Anda lampirkan ke identitas (pengguna, grup, atau peran). Kebijakan ini mengontrol tindakan apa yang bisa dilakukan oleh

identitas tersebut, terhadap sumber daya yang mana, dan dalam kondisi apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Tentukan izin IAM kustom dengan kebijakan yang dikelola pelanggan dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat berupa kebijakan inline (disematkan langsung ke dalam satu identitas) atau kebijakan terkelola (kebijakan mandiri yang dilampirkan pada banyak identitas). Untuk mempelajari cara memilih antara kebijakan terkelola dan kebijakan inline, lihat Pilih antara kebijakan terkelola dan kebijakan inline dalam Panduan Pengguna IAM.

Kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contohnya termasuk kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Anda harus menentukan principal dalam kebijakan berbasis sumber daya.

Kebijakan berbasis sumber daya merupakan kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan AWS terkelola dari IAM dalam kebijakan berbasis sumber daya.

Jenis-jenis kebijakan lain

AWS mendukung jenis kebijakan tambahan yang dapat menetapkan izin maksimum yang diberikan oleh jenis kebijakan yang lebih umum:

- Batasan izin – Menetapkan izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas IAM. Untuk informasi selengkapnya, lihat Batasan izin untuk entitas IAM dalam Panduan Pengguna IAM.
- Kebijakan kontrol layanan (SCPs) — Tentukan izin maksimum untuk organisasi atau unit organisasi di AWS Organizations. Untuk informasi selengkapnya, lihat Kebijakan kontrol layanan dalam Panduan Pengguna AWS Organizations .
- Kebijakan kontrol sumber daya (RCPs) — Tetapkan izin maksimum yang tersedia untuk sumber daya di akun Anda. Untuk informasi selengkapnya, lihat Kebijakan kontrol sumber daya (RCPs) di Panduan AWS Organizations Pengguna.
- Kebijakan sesi – Kebijakan lanjutan yang diteruskan sebagai parameter saat membuat sesi sementara untuk peran atau pengguna terfederasi. Untuk informasi selengkapnya, lihat Kebijakan sesi dalam Panduan Pengguna IAM.

Berbagai jenis kebijakan

Ketika beberapa jenis kebijakan berlaku pada suatu permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan saat beberapa jenis kebijakan terlibat, lihat Logika evaluasi kebijakan di Panduan Pengguna IAM.

Bagaimana AWS HealthLake bekerja dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses HealthLake, pelajari fitur IAM yang tersedia untuk digunakan. HealthLake

Fitur IAM yang dapat Anda gunakan AWS HealthLake

Fitur IAM	HealthLake dukungan
Kebijakan berbasis identitas	Ya
Kebijakan berbasis sumber daya	Tidak
Tindakan kebijakan	Ya
Sumber daya kebijakan	Ya
Kunci kondisi kebijakan	Ya
ACLs	Tidak
ABAC (tanda dalam kebijakan)	Ya
Kredensial sementara	Ya
Izin principal	Ya
Peran layanan	Ya
Peran terkait layanan	Tidak

Untuk mendapatkan tampilan tingkat tinggi tentang cara HealthLake dan AWS layanan lain bekerja dengan sebagian besar fitur IAM, lihat AWS layanan yang bekerja dengan IAM di Panduan Pengguna IAM.

Kebijakan berbasis identitas untuk AWS HealthLake

Mendukung kebijakan berbasis identitas: Ya

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, seperti pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan oleh pengguna dan peran, di sumber daya mana, dan berdasarkan kondisi seperti apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Tentukan izin IAM kustom dengan kebijakan terkelola pelanggan dalam Panduan Pengguna IAM.

Dengan kebijakan berbasis identitas IAM, Anda dapat menentukan secara spesifik apakah tindakan dan sumber daya diizinkan atau ditolak, serta kondisi yang menjadi dasar dikabulkan atau ditolaknya tindakan tersebut. Untuk mempelajari semua elemen yang dapat Anda gunakan dalam kebijakan JSON, lihat Referensi elemen kebijakan JSON IAM dalam Panduan Pengguna IAM.

Contoh kebijakan berbasis identitas untuk AWS HealthLake

Untuk melihat contoh kebijakan HealthLake berbasis identitas, lihat. Contoh kebijakan berbasis identitas untuk AWS HealthLake

Kebijakan berbasis sumber daya dalam AWS HealthLake

Mendukung kebijakan berbasis sumber daya: Tidak

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya tempat kebijakan dilampirkan, kebijakan menentukan tindakan apa yang dapat dilakukan oleh principal tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus menentukan principal dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau. Layanan AWS

Untuk mengaktifkan akses lintas akun, Anda dapat menentukan secara spesifik seluruh akun atau entitas IAM di akun lain sebagai principal dalam kebijakan berbasis sumber daya. Untuk informasi selengkapnya, lihat Akses sumber daya lintas akun di IAM dalam Panduan Pengguna IAM.

Tindakan kebijakan untuk AWS HealthLake

Mendukung tindakan kebijakan: Ya

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, di mana utama dapat melakukan tindakan pada sumber daya, dan dalam kondisi apa.

Elemen `Action` dari kebijakan JSON menjelaskan tindakan yang dapat Anda gunakan untuk mengizinkan atau menolak akses dalam sebuah kebijakan. Sertakan tindakan dalam kebijakan untuk memberikan izin untuk melakukan operasi terkait.

Untuk melihat daftar HealthLake tindakan, lihat Tindakan yang ditentukan oleh AWS HealthLake dalam Referensi Otorisasi Layanan.

Tindakan kebijakan HealthLake menggunakan awalan berikut sebelum tindakan:

```
healthlake
```

Untuk menentukan beberapa tindakan dalam satu pernyataan, pisahkan setiap tindakan dengan koma.

```
"Action": [
      "healthlake:action1",
      "healthlake:action2"
         ]
```

Untuk melihat contoh kebijakan HealthLake berbasis identitas, lihat. Contoh kebijakan berbasis identitas untuk AWS HealthLake

Sumber daya kebijakan untuk AWS HealthLake

Mendukung sumber daya kebijakan: Ya

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, di mana utama dapat melakukan tindakan pada sumber daya, dan dalam kondisi apa.

Elemen kebijakan JSON `Resource` menentukan objek yang menjadi target penerapan tindakan. Praktik terbaiknya, tentukan sumber daya menggunakan Amazon Resource Name (ARN). Untuk tindakan yang tidak mendukung izin di tingkat sumber daya, gunakan wildcard (*) untuk menunjukkan bahwa pernyataan tersebut berlaku untuk semua sumber daya.

```
"Resource": "*"
```

Untuk melihat daftar jenis sumber daya dan jenis HealthLake sumber daya ARNs, lihat Sumber daya yang ditentukan oleh AWS HealthLake dalam Referensi Otorisasi Layanan. Untuk mempelajari tindakan yang dengannya Anda dapat menentukan ARN dari setiap sumber daya, lihat Tindakan yang ditentukan oleh. AWS HealthLake

Untuk melihat contoh kebijakan HealthLake berbasis identitas, lihat. Contoh kebijakan berbasis identitas untuk AWS HealthLake

Kunci kondisi kebijakan untuk AWS HealthLake

Mendukung kunci kondisi kebijakan khusus layanan: Yes

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, principal dapat melakukan tindakan pada suatu sumber daya, dan dalam suatu syarat.

Elemen `Condition` menentukan ketika pernyataan dieksekusi berdasarkan kriteria yang ditetapkan. Anda dapat membuat ekspresi bersyarat yang menggunakan operator kondisi, misalnya sama dengan atau kurang dari, untuk mencocokkan kondisi dalam kebijakan dengan nilai-nilai yang diminta. Untuk melihat semua kunci kondisi AWS global, lihat kunci konteks kondisi AWS global di Panduan Pengguna IAM.

Untuk melihat daftar kunci HealthLake kondisi, lihat Kunci kondisi untuk AWS HealthLake dalam Referensi Otorisasi Layanan. Untuk mempelajari tindakan dan sumber daya yang dapat digunakan untuk menggunakan kunci kondisi, lihat Tindakan yang ditentukan oleh AWS HealthLake.

Untuk melihat contoh kebijakan HealthLake berbasis identitas, lihat. Contoh kebijakan berbasis identitas untuk AWS HealthLake

Daftar kontrol akses (ACLs) di AWS HealthLake

Mendukung ACLs: Tidak

Access control lists (ACLs) mengontrol prinsipal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACLs mirip dengan kebijakan berbasis sumber daya, meskipun mereka tidak menggunakan format dokumen kebijakan JSON.

Kontrol akses berbasis atribut (ABAC) dengan AWS HealthLake

Mendukung ABAC (tanda dalam kebijakan): Ya

Kontrol akses berbasis atribut (ABAC) adalah strategi otorisasi yang menentukan izin berdasarkan atribut tanda. Anda dapat melampirkan tag ke entitas dan AWS sumber daya IAM, lalu merancang kebijakan ABAC untuk mengizinkan operasi saat tag prinsipal cocok dengan tag pada sumber daya.

Untuk mengendalikan akses berdasarkan tanda, berikan informasi tentang tanda di elemen kondisi dari kebijakan menggunakan kunci kondisi `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, atau `aws:TagKeys`.

Jika sebuah layanan mendukung ketiga kunci kondisi untuk setiap jenis sumber daya, nilainya adalah Ya untuk layanan tersebut. Jika suatu layanan mendukung ketiga kunci kondisi untuk hanya beberapa jenis sumber daya, nilainya adalah Parsial.

Untuk informasi selengkapnya tentang ABAC, lihat Tentukan izin dengan otorisasi ABAC dalam Panduan Pengguna IAM. Untuk melihat tutorial yang menguraikan langkah-langkah pengaturan ABAC, lihat Menggunakan kontrol akses berbasis atribut (ABAC) dalam Panduan Pengguna IAM.

Menggunakan kredensi sementara dengan AWS HealthLake

Mendukung kredensial sementara: Ya

Kredensi sementara menyediakan akses jangka pendek ke AWS sumber daya dan secara otomatis dibuat saat Anda menggunakan federasi atau beralih peran. AWS merekomendasikan agar Anda secara dinamis menghasilkan kredensi sementara alih-alih menggunakan kunci akses jangka panjang. Untuk informasi selengkapnya, lihat Kredensial keamanan sementara di IAM dan Layanan AWS yang berfungsi dengan IAM dalam Panduan Pengguna IAM.

Izin utama lintas layanan untuk AWS HealthLake

Mendukung sesi akses terusan (FAS): Ya

Sesi akses teruskan (FAS) menggunakan izin dari pemanggilan utama Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat Sesi akses terusan.

Peran layanan untuk AWS HealthLake

Mendukung peran layanan: Ya

Peran layanan adalah peran IAM yang diambil oleh sebuah layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Buat sebuah peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.

Untuk informasi tentang peran layanan dan kebijakan sebaris yang diperlukan untuk akses penuh AWS HealthLake, lihatMenyiapkan AWS HealthLake.

> **Warning**
>
> Mengubah izin untuk peran layanan dapat merusak HealthLake fungsionalitas. Edit peran layanan hanya jika HealthLake memberikan panduan untuk melakukannya.

Peran terkait layanan untuk AWS HealthLake

Mendukung peran terkait layanan: Tidak

Peran terkait layanan adalah jenis peran layanan yang ditautkan ke. Layanan AWS Layanan tersebut dapat menjalankan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.

Untuk detail tentang pembuatan atau manajemen peran terkait layanan, lihat Layanan AWS yang berfungsi dengan IAM. Cari layanan dalam tabel yang memiliki `Yes` di kolom Peran terkait layanan. Pilih tautan Ya untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

Contoh kebijakan berbasis identitas untuk AWS HealthLake

Secara default, pengguna dan peran tidak memiliki izin untuk membuat atau mengubah sumber daya HealthLake. Untuk memberikan izin kepada pengguna untuk melakukan tindakan di sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM.

Untuk mempelajari cara membuat kebijakan berbasis identitas IAM dengan menggunakan contoh dokumen kebijakan JSON ini, lihat Membuat kebijakan IAM (konsol) di Panduan Pengguna IAM.

Untuk detail tentang tindakan dan jenis sumber daya yang ditentukan oleh HealthLake, termasuk format ARNs untuk setiap jenis sumber daya, lihat Kunci tindakan, sumber daya, dan kondisi untuk AWS HealthLake dalam Referensi Otorisasi Layanan.

Topik

- Praktik terbaik kebijakan
- Menggunakan AWS HealthLake konsol
- Mengakses penyimpanan AWS HealthLake data di Amazon Athena
- Memungkinkan pengguna untuk melihat izin mereka sendiri

Praktik terbaik kebijakan

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus HealthLake sumber daya di akun Anda. Tindakan ini membuat Akun AWS Anda dikenai biaya. Ketika Anda membuat atau mengedit kebijakan berbasis identitas, ikuti panduan dan rekomendasi ini:

- Mulailah dengan kebijakan AWS terkelola dan beralih ke izin hak istimewa paling sedikit — Untuk mulai memberikan izin kepada pengguna dan beban kerja Anda, gunakan kebijakan AWS terkelola yang memberikan izin untuk banyak kasus penggunaan umum. Mereka tersedia di Anda Akun AWS. Kami menyarankan Anda mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola AWS pelanggan yang khusus untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat Kebijakan yang dikelola AWS atau Kebijakan yang dikelola AWS untuk fungsi tugas dalam Panduan Pengguna IAM.
- Menerapkan izin dengan hak akses paling rendah – Ketika Anda menetapkan izin dengan kebijakan IAM, hanya berikan izin yang diperlukan untuk melakukan tugas. Anda melakukannya dengan mendefinisikan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi tertentu, yang juga dikenal sebagai izin dengan hak akses paling rendah. Untuk informasi selengkapnya tentang cara menggunakan IAM untuk mengajukan izin, lihat Kebijakan dan izin dalam IAM dalam Panduan Pengguna IAM.
- Gunakan kondisi dalam kebijakan IAM untuk membatasi akses lebih lanjut – Anda dapat menambahkan suatu kondisi ke kebijakan Anda untuk membatasi akses ke tindakan dan sumber daya. Sebagai contoh, Anda dapat menulis kondisi kebijakan untuk menentukan bahwa semua permintaan harus dikirim menggunakan SSL. Anda juga dapat menggunakan ketentuan untuk memberikan akses ke tindakan layanan jika digunakan melalui yang spesifik Layanan AWS, seperti CloudFormation. Untuk informasi selengkapnya, lihat Elemen kebijakan JSON IAM: Kondisi dalam Panduan Pengguna IAM.
- Gunakan IAM Access Analyzer untuk memvalidasi kebijakan IAM Anda untuk memastikan izin yang aman dan fungsional – IAM Access Analyzer memvalidasi kebijakan baru dan yang sudah

ada sehingga kebijakan tersebut mematuhi bahasa kebijakan IAM (JSON) dan praktik terbaik IAM. IAM Access Analyzer menyediakan lebih dari 100 pemeriksaan kebijakan dan rekomendasi yang dapat ditindaklanjuti untuk membantu Anda membuat kebijakan yang aman dan fungsional. Untuk informasi selengkapnya, lihat Validasi kebijakan dengan IAM Access Analyzer dalam Panduan Pengguna IAM.

- Memerlukan otentikasi multi-faktor (MFA) - Jika Anda memiliki skenario yang mengharuskan pengguna IAM atau pengguna root di Anda, Akun AWS aktifkan MFA untuk keamanan tambahan. Untuk meminta MFA ketika operasi API dipanggil, tambahkan kondisi MFA pada kebijakan Anda. Untuk informasi selengkapnya, lihat Amankan akses API dengan MFA dalam Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang praktik terbaik dalam IAM, lihat Praktik terbaik keamanan di IAM dalam Panduan Pengguna IAM.

Menggunakan AWS HealthLake konsol

Untuk mengakses AWS HealthLake konsol, Anda harus memiliki set izin minimum. Izin ini harus memungkinkan Anda untuk membuat daftar dan melihat detail tentang HealthLake sumber daya di Anda Akun AWS. Jika Anda membuat kebijakan berbasis identitas yang lebih ketat daripada izin minimum yang diperlukan, konsol tidak akan berfungsi sebagaimana mestinya untuk entitas (pengguna atau peran) dengan kebijakan tersebut.

Anda tidak perlu mengizinkan izin konsol minimum untuk pengguna yang melakukan panggilan hanya ke AWS CLI atau AWS API. Sebagai gantinya, izinkan akses hanya ke tindakan yang sesuai dengan operasi API yang coba mereka lakukan.

Untuk akses penuh HealthLake, lampirkan kebijakan berikut ke pengguna atau peran IAM: `AmazonHealthLakeFullAccess` dan`AWSLakeFormationDataAdmin`. Anda juga perlu melampirkan kebijakan HealthLake inline yang merupakan peran layanan. Peran layanan adalah peran IAM yang diambil oleh sebuah layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Buat sebuah peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM. Untuk informasi tentang kebijakan inline yang membuat peran layanan yang diperlukan, lihatMenyiapkan AWS HealthLake. Anda juga harus menggunakan AWS Lake Formation konsol atau CLI untuk menetapkan HealthLake administrator Anda menjadi administrator AWS Lake Formation Data Lake. Untuk informasi selengkapnya, lihat Menyiapkan AWS HealthLake.

Mengakses penyimpanan AWS HealthLake data di Amazon Athena

Jika Anda ingin memberi pengguna dan peran akses ke penyimpanan HealthLake data Amazon Athena, lampirkan kebijakan IAM berikut ke peran atau pengguna: `AmazonAthenaFullAccess` dan`AmazonS3FullAccess`. `Select`dan `Describe` izin juga diperlukan pada tabel yang dikelola oleh AWS Lake Formation. AWS Lake Formation izin tabel diberikan oleh AWS Lake Formation administrator di AWS Lake Formation konsol atau melalui CLI. Untuk informasi selengkapnya, lihat Menyiapkan AWS HealthLake

Memungkinkan pengguna untuk melihat izin mereka sendiri

Contoh ini menunjukkan cara membuat kebijakan yang mengizinkan pengguna IAM melihat kebijakan inline dan terkelola yang dilampirkan ke identitas pengguna mereka. Kebijakan ini mencakup izin untuk menyelesaikan tindakan ini di konsol atau menggunakan API atau secara terprogram. AWS CLI AWS

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
```

```
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS kebijakan terkelola untuk AWS HealthLake

Kebijakan AWS terkelola adalah kebijakan mandiri yang dibuat dan dikelola oleh AWS. AWS Kebijakan terkelola dirancang untuk memberikan izin bagi banyak kasus penggunaan umum sehingga Anda dapat mulai menetapkan izin kepada pengguna, grup, dan peran.

Perlu diingat bahwa kebijakan AWS terkelola mungkin tidak memberikan izin hak istimewa paling sedikit untuk kasus penggunaan spesifik Anda karena tersedia untuk digunakan semua pelanggan. AWS Kami menyarankan Anda untuk mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola pelanggan yang khusus untuk kasus penggunaan Anda.

Anda tidak dapat mengubah izin yang ditentukan dalam kebijakan AWS terkelola. Jika AWS memperbarui izin yang ditentukan dalam kebijakan AWS terkelola, pemutakhiran akan memengaruhi semua identitas utama (pengguna, grup, dan peran) yang dilampirkan kebijakan tersebut. AWS kemungkinan besar akan memperbarui kebijakan AWS terkelola saat baru Layanan AWS diluncurkan atau operasi API baru tersedia untuk layanan yang ada.

Untuk informasi selengkapnya, lihat Kebijakan terkelola AWS dalam Panduan Pengguna IAM.

AWS kebijakan terkelola: AmazonHealthLakeFullAccess

`AmazonHealthLakeFullAccess`Kebijakan ini menyediakan akses penuh ke HealthLake. Dengan kebijakan ini yang dilampirkan pada pengguna atau peran mereka, pengguna dapat menggunakannya HealthLake untuk mengakses, menanyakan, mengimpor, dan mengekspor data HealthLake. Untuk melakukan banyak tindakan umum di HealthLake, Anda harus menambahkan kebijakan tambahan ke pengguna atau peran. Untuk informasi selengkapnya, lihat HealthLake operasi Menyiapkan AWS HealthLake dan izin.

Anda dapat melampirkan kebijakan `AmazonHealthLakeFullAccess` ke identitas IAM Anda.

Kebijakan ini memberikan izin administratif dan kontributor yang memungkinkan pengguna dan peran untuk melakukan kueri, mencari, mengimpor HealthLake, dan mengekspor dengan, dan juga memungkinkan HealthLake untuk melakukan tindakan atas nama pengguna dan peran yang memiliki izin ini.

Detail izin

Kebijakan ini mencakup pernyataan berikut.

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 kebijakan terkelola: AmazonHealthLakeReadOnlyAccess

`AmazonHealthLakeReadOnlyAccess`kebijakan memberikan akses dan izin hanya-baca ke dan sumber daya terkait di HealthLake layanan lain. AWS Terapkan kebijakan ini kepada pengguna yang ingin Anda berikan kemampuan untuk menanyakan dan melihat penyimpanan HealthLake data, tetapi bukan kemampuan untuk membuat atau membuat perubahan pada mereka.

Anda dapat melampirkan kebijakan `AmazonHealthLakeReadOnlyAccess` ke identitas IAM Anda.

Kebijakan ini memberikan *`read-only`* izin yang memungkinkan pengguna dan peran untuk melakukan kueri. HealthLake

Detail izin

Kebijakan ini mencakup pernyataan berikut.

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 operasi dan izin

Tabel berikut mencantumkan operasi umum HealthLake dan izin yang diperlukan untuk menjalankannya.

HealthLake operasi	Izin yang diperlukan
Buat penyimpanan data di HealthLake	`AmazonHealthLakeFullAccess` ,`AmazonLakeFormationDataAdmin` , kebijakan sebaris, dan izin AWS Lake Formation Administrator yang dikelola oleh AWS Lake Formation
Hapus penyimpanan data di HealthLake	`AmazonHealthLakeFullAccess` ,`AmazonLakeFormationDataAdmin` , kebijakan sebaris, dan izin AWS Lake Formation Administrator yang dikelola oleh AWS Lake Formation
Daftar, cari, atau kueri penyimpanan data di HealthLake	`AmazonHealthLakeReadOnlyAccess`
Kueri penyimpanan data menggunakan Amazon Athena	`AmazonAthenaFullAccess` ,`AmazonS3FullAccess` , AWS Lake Formation `Select` dan `Describe` izin pada tabel yang dikelola oleh AWS Lake Formation
Impor data dari HealthLake	Lihat Menyiapkan izin untuk pekerjaan impor.
Ekspor data dari HealthLake	Lihat Menyiapkan izin untuk pekerjaan ekspor.

HealthLake pembaruan kebijakan AWS terkelola

Lihat detail tentang pembaruan kebijakan AWS terkelola HealthLake sejak layanan ini mulai melacak perubahan ini. Untuk peringatan otomatis tentang perubahan pada halaman ini, berlangganan umpan RSS di halaman Riwayat HealthLake dokumen.

Ubah	Deskripsi	Date
AmazonHealthLakeFullAccess	`AmazonHealthLakeFullAccess` kebijakan yang diperlukan untuk memungkinkan akses penuh ke HealthLake.	November, 14, 2022
AmazonHealthLakeReadOnlyAccess	`AmazonHealthLakeReadOnlyAccess` kebijakan yang diperlukan untuk akses hanya-baca ke. HealthLake	November, 14, 2022
HealthLake mulai melacak perubahan	HealthLake mulai melacak perubahan untuk kebijakan AWS terkelolanya.	November, 14, 2022

Memecahkan masalah AWS HealthLake identitas dan akses

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan HealthLake dan IAM.

Topik

- Saya tidak berwenang untuk melakukan tindakan di AWS HealthLake
- Saya tidak berwenang untuk melakukan iam: PassRole
- Saya ingin mengizinkan orang di luar AWS akun saya untuk mengakses AWS HealthLake sumber daya saya

Saya tidak berwenang untuk melakukan tindakan di AWS HealthLake

Jika Konsol Manajemen AWS memberitahu Anda bahwa Anda tidak berwenang untuk melakukan suatu tindakan, maka Anda harus menghubungi administrator Anda untuk bantuan. Administrator Anda adalah orang yang memberikan nama pengguna dan kata sandi Anda.

Contoh kesalahan berikut terjadi ketika pengguna IAM `mateojackson` mencoba menggunakan konsol untuk melihat detail tentang suatu sumber daya fiktif *`my-example-widget`*, tetapi tidak memiliki izin fiktif `healthlake:`*`GetWidget`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 healthlake:GetWidget on resource: my-example-widget
```

Dalam hal ini, Mateo meminta administratornya untuk memperbarui kebijakannya agar dia dapat mengakses *`my-example-widget`* menggunakan `healthlake:`*`GetWidget`* tindakan.

Saya tidak berwenang untuk melakukan iam: PassRole

Jika Anda menerima kesalahan yang tidak diizinkan untuk melakukan `iam:PassRole` tindakan, kebijakan Anda harus diperbarui agar Anda dapat meneruskan peran HealthLake.

Beberapa Layanan AWS memungkinkan Anda untuk meneruskan peran yang ada ke layanan tersebut alih-alih membuat peran layanan baru atau peran terkait layanan. Untuk melakukannya, Anda harus memiliki izin untuk meneruskan peran ke layanan.

Contoh kesalahan berikut terjadi ketika pengguna IAM bernama `marymajor` mencoba menggunakan konsol tersebut untuk melakukan tindakan di HealthLake. Namun, tindakan tersebut memerlukan layanan untuk mendapatkan izin yang diberikan oleh peran layanan. Mary tidak memiliki izin untuk meneruskan peran tersebut pada layanan.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dalam kasus ini, kebijakan Mary harus diperbarui agar dia mendapatkan izin untuk melakukan tindakan `iam:PassRole` tersebut.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya ingin mengizinkan orang di luar AWS akun saya untuk mengakses AWS HealthLake sumber daya saya

Anda dapat membuat peran yang dapat digunakan pengguna di akun lain atau orang-orang di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa saja yang dipercaya untuk mengambil peran tersebut. Untuk layanan yang mendukung kebijakan berbasis

sumber daya atau daftar kontrol akses (ACLs), Anda dapat menggunakan kebijakan tersebut untuk memberi orang akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa referensi berikut:

- Untuk mempelajari apakah HealthLake mendukung fitur-fitur ini, lihatBagaimana AWS HealthLake bekerja dengan IAM.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda di seluruh sumber daya Akun AWS yang Anda miliki, lihat Menyediakan akses ke pengguna IAM di pengguna lain Akun AWS yang Anda miliki di Panduan Pengguna IAM.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda kepada pihak ketiga Akun AWS, lihat Menyediakan akses yang Akun AWS dimiliki oleh pihak ketiga dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses melalui federasi identitas, lihat Menyediakan akses ke pengguna terautentikasi eksternal (federasi identitas) dalam Panduan Pengguna IAM.
- Untuk mempelajari perbedaan antara menggunakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Akses sumber daya lintas akun di IAM di Panduan Pengguna IAM.

Validasi kepatuhan untuk AWS HealthLake

Third-party auditor menilai keamanan dan kepatuhan AWS HealthLake sebagai bagian dari beberapa program AWS kepatuhan. Untuk HealthLake ini termasuk HIPAA.

Untuk mengetahui apakah an Layanan AWS berada dalam lingkup program kepatuhan tertentu, lihat Layanan AWS di Lingkup oleh Program Kepatuhan Layanan AWS dan pilih program kepatuhan yang Anda minati. Untuk informasi umum, lihat Program AWS Kepatuhan Program AWS .

Anda dapat mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat Mengunduh Laporan di AWS Artifact .

Tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, dan hukum dan peraturan yang berlaku. Untuk informasi selengkapnya tentang tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS, lihat Dokumentasi AWS Keamanan.

Keamanan infrastruktur di AWS HealthLake

Sebagai layanan terkelola, AWS HealthLake dilindungi oleh prosedur keamanan jaringan AWS global yang dijelaskan dalam whitepaper Amazon Web Services: Tinjauan Proses Keamanan.

Anda menggunakan panggilan API yang AWS dipublikasikan untuk mengakses HealthLake melalui jaringan. Klien harus mendukung Keamanan Lapisan Pengangkutan (TLS) 1.0 atau versi yang lebih baru. Kami merekomendasikan TLS 1.2 atau versi yang lebih baru. Klien juga harus mendukung cipher suite dengan perfect forward secrecy (PFS) seperti Ephemeral (DHE) atau Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Diffie-Hellman Sebagian besar sistem modern seperti Java 7 dan versi lebih baru mendukung mode-mode ini.

Selain itu, permintaan harus ditandatangani menggunakan ID kunci akses dan kunci akses rahasia yang terkait dengan principal IAM. Atau Anda dapat menggunakan AWS Security Token Service (AWS STS) untuk menghasilkan kredensial keamanan sementara untuk menandatangani permintaan.

Menciptakan AWS HealthLake sumber daya dengan AWS CloudFormation

AWS HealthLake terintegrasi dengan AWS CloudFormation, layanan yang membantu Anda memodelkan dan mengatur AWS sumber daya Anda sehingga Anda dapat menghabiskan lebih sedikit waktu untuk membuat dan mengelola sumber daya dan infrastruktur Anda. Anda membuat template yang menjelaskan semua AWS sumber daya yang Anda inginkan dan CloudFormation ketentuan dan mengonfigurasi sumber daya tersebut untuk Anda.

Ketika Anda menggunakan CloudFormation, Anda dapat menggunakan kembali template Anda untuk mengatur HealthLake sumber daya Anda secara konsisten dan berulang kali. Jelaskan sumber daya Anda sekali, lalu sediakan sumber daya yang sama berulang-ulang di beberapa Akun AWS dan Wilayah.

HealthLake dan CloudFormation template

Untuk menyediakan dan mengonfigurasi sumber daya untuk HealthLake dan layanan terkait, Anda harus memahami CloudFormation templat. Templat adalah file teks dengan format JSON atau YAML. Template ini menjelaskan sumber daya yang ingin Anda sediakan di CloudFormation tumpukan Anda. Jika Anda tidak terbiasa dengan JSON atau YAMAL, Anda dapat menggunakan

CloudFormation Designer untuk membantu Anda memulai dengan template. CloudFormation Untuk informasi selengkapnya, lihat Apa itu CloudFormation Designer? di Panduan Pengguna AWS CloudFormation .

> **Note**
>
> AWS HealthLake mendukung pembuatan penyimpanan data dengan CloudFormation. Untuk informasi selengkapnya, termasuk contoh template JSON dan YAMAL untuk menyediakan penyimpanan HealthLake data, lihat referensi tipe AWS HealthLake sumber daya di Panduan Pengguna.AWS CloudFormation

Pelajari lebih lanjut tentang CloudFormation

Untuk mempelajari selengkapnya CloudFormation, lihat sumber daya berikut:

- AWS CloudFormation
- AWS CloudFormation Panduan Pengguna
- Referensi CloudFormation API
- AWS CloudFormation Panduan Pengguna Antarmuka Baris Perintah

AWS HealthLake dan antarmuka titik akhir VPC ()AWS PrivateLink

Anda dapat membuat koneksi pribadi antara VPC Anda dan AWS HealthLake dengan membuat antarmuka VPC endpoint. Endpoint VPC antarmuka didukung oleh AWS PrivateLink, teknologi yang dapat Anda gunakan untuk mengakses secara pribadi HealthLake; APIs tanpa gateway internet, perangkat NAT, koneksi VPN, atau koneksi. Direct Connect Instans di VPC Anda tidak memerlukan alamat IP publik untuk berkomunikasi HealthLake; APIs Lalu lintas antara VPC Anda dan HealthLake; tidak meninggalkan jaringan Amazon.

Setiap titik akhir antarmuka diwakili oleh satu atau beberapa Antarmuka Jaringan Elastis di subnet Anda.

Untuk informasi selengkapnya, lihat Titik akhir VPC Antarmuka (AWS PrivateLink) di Panduan Pengguna Amazon VPC.

Pertimbangan untuk titik akhir HealthLake VPC

Sebelum menyiapkan titik akhir VPC antarmuka HealthLake, pastikan Anda meninjau properti dan batasan titik akhir Antarmuka di Panduan Pengguna Amazon VPC.

HealthLake mendukung panggilan ke semua tindakan API-nya dari VPC Anda.

Membuat titik akhir VPC antarmuka untuk; HealthLake

Anda dapat membuat titik akhir VPC untuk layanan HealthLake; menggunakan konsol VPC Amazon atau (). AWS Command Line Interface AWS CLI Untuk informasi selengkapnya, lihat Membuat titik akhir antarmuka dalam Panduan Pengguna Amazon VPC.

Buat titik akhir VPC untuk HealthLake; menggunakan nama layanan berikut:

- com.amazonaws. *region*.healthlake

Jika Anda mengaktifkan DNS pribadi untuk titik akhir, Anda dapat membuat permintaan API untuk HealthLake menggunakan nama DNS default untuk Wilayah. Misalnya, *healthlake.us-east-1.amazonaws.com*.

Untuk informasi selengkapnya, lihat Mengakses layanan melalui titik akhir antarmuka dalam Panduan Pengguna Amazon VPC.

Membuat kebijakan titik akhir VPC untuk HealthLake

Anda dapat melampirkan kebijakan titik akhir ke VPC endpoint yang mengendalikan akses ke HealthLake. Kebijakan menentukan informasi berikut ini:

- Prinsipal yang dapat melakukan tindakan.
- Tindakan yang dapat dilakukan.
- Sumber daya yang menjadi target tindakan.

Untuk informasi selengkapnya, lihat Mengontrol Akses ke Layanan dengan titik akhir VPC dalam Panduan Pengguna Amazon VPC.

Contoh: Kebijakan titik akhir VPC untuk tindakan HealthLake

Berikut ini adalah contoh kebijakan endpoint untuk HealthLake. Ketika dilampirkan ke titik akhir, kebijakan ini memberikan akses ke HealthLake `CreateFHIRDatastore` tindakan untuk semua prinsipal di semua sumber daya.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "healthlake:create-fhir-datastore"
         ],
         "Resource":"*"
      }
   ]
}
```

Praktik terbaik keamanan di AWS HealthLake

AWS HealthLake menyediakan sejumlah fitur keamanan untuk dipertimbangkan saat Anda mengembangkan dan menerapkan kebijakan keamanan Anda sendiri. Praktik terbaik berikut adalah pedoman umum dan tidak mewakili solusi keamanan yang lengkap. Karena praktik terbaik ini mungkin tidak sesuai atau tidak memadai untuk lingkungan Anda, perlakukan itu sebagai pertimbangan yang bermanfaat, bukan sebagai resep.

- Menerapkan akses hak istimewa paling sedikit.
- Bila memungkinkan, gunakan Customer-Managed-Keys (CMK) untuk mengenkripsi data Anda. Untuk mempelajari CMK selengkapnya, lihat Amazon Key Management Service.
- Gunakan Pencarian dengan POST, bukan Cari dengan GET saat menanyakan PHI atau PII di penyimpanan data Anda.
- Batasi akses ke fungsi audit yang sensitif dan penting.
- Saat membuat sumber daya melalui pembaruan atau API impor massal, jangan gunakan PHI atau PII, termasuk nama penyimpanan data dan pekerjaan, di bidang apa pun yang terlihat atau di ID FHIR logis (LID).
- Saat mengirim permintaan buat, baca, perbarui, hapus, atau cari, jangan gunakan PHI di header HTTP.

- Memungkinkan AWS CloudTrail untuk mengaudit AWS HealthLake penggunaan dan untuk memastikan bahwa tidak ada aktivitas yang tidak terduga.
- Tinjau praktik terbaik untuk menggunakan bucket Amazon S3 dengan aman. Untuk mempelajari selengkapnya, lihat Praktik terbaik keamanan di panduan pengguna Amazon S3.

Ketahanan di AWS HealthLake

Infrastruktur AWS global dibangun di sekitar AWS Wilayah dan Zona Ketersediaan. AWS Wilayah menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan latensi rendah, throughput tinggi, dan jaringan yang sangat redundan. Dengan Availability Zone, Anda dapat mendesain dan mengoperasikan aplikasi dan basis data yang secara otomatis mengalami kegagalan di antara zona tanpa gangguan. Zona Ketersediaan memiliki ketersediaan lebih tinggi, toleransi kesalahan lebih baik, dan lebih dapat diskalakan dibandingkan infrastruktur pusat data tunggal atau beberapa pusat data tradisional.

Jika Anda harus melakukan replikasi data atau aplikasi Anda pada jarak geografis yang lebih luas, gunakan Wilayah Lokal AWS . Wilayah AWS Lokal adalah pusat data tunggal yang dirancang untuk melengkapi AWS Wilayah yang ada. Seperti semua AWS Wilayah, Wilayah AWS Lokal benar-benar terisolasi dari AWS Wilayah lain.

Untuk informasi selengkapnya tentang AWS Wilayah dan Availability Zone, lihat Infrastruktur AWS Global.

AWS HealthLake referensi

Materi referensi pendukung berikut tersedia untuk SMART di FHIR, FHIR, dan. AWS HealthLake

> **Note**
>
> Semua HealthLake tindakan asli dan tipe data dijelaskan dalam referensi terpisah. Untuk informasi lebih lanjut, lihat Referensi API AWS HealthLake .

Topik

- SMART pada dukungan FHIR untuk AWS HealthLake
- Dukungan FHIR R4 untuk AWS HealthLake
- Referensi kepatuhan untuk AWS HealthLake
- Referensi Support untuk AWS HealthLake

SMART pada dukungan FHIR untuk AWS HealthLake

Aplikasi Medis yang Dapat Diganti dan Teknologi yang Dapat Digunakan Kembali (SMART) pada penyimpanan HealthLake data berkemampuan FHIR memungkinkan akses ke SMART pada aplikasi yang sesuai dengan FHIR. HealthLake Data diakses dengan mengautentikasi dan mengotorisasi permintaan menggunakan server otorisasi pihak ketiga. Jadi, alih-alih mengelola kredensil pengguna melalui AWS Identity and Access Management, Anda melakukannya menggunakan SMART pada server otorisasi yang sesuai dengan FHIR.

> **Note**
>
> HealthLake mendukung SMART pada FHIR versi 1.0 dan 2.0. Untuk mempelajari lebih lanjut tentang kerangka kerja ini, lihat Peluncuran Aplikasi SMART di dokumentasi FHIR R4. HealthLake penyimpanan data mendukung kerangka kerja otentikasi dan otorisasi berikut untuk SMART pada permintaan FHIR:
>
> - OpenID (AuthN): untuk mengautentikasi aplikasi orang atau klien adalah siapa (atau apa) yang mereka klaim.

- OAuth 2.0 (AuthZ): untuk mengotorisasi sumber daya FHIR mana di penyimpanan HealthLake data Anda, permintaan yang diautentikasi dapat dibaca atau ditulis. Ini ditentukan oleh cakupan yang diatur di server otorisasi Anda.

Anda dapat membuat SMART di penyimpanan data berkemampuan FHIR menggunakan AWS CLI atau AWS SDKs. Lihat informasi yang lebih lengkap di Membuat penyimpanan HealthLake data.

Topik

- Memulai SMART di FHIR
- HealthLake persyaratan otentikasi untuk SMART di FHIR
- SMART pada cakupan FHIR OAuth 2.0 didukung oleh HealthLake
- Validasi Token menggunakan AWS Lambda
- Menggunakan otorisasi berbutir halus dengan SMART pada penyimpanan data yang diaktifkan FHIR HealthLake
- Mengambil SMART pada Dokumen Penemuan FHIR
- Membuat permintaan FHIR REST API pada penyimpanan data berkemampuan SMART HealthLake

Memulai SMART di FHIR

Topik berikut menjelaskan cara memulai dengan SMART pada otorisasi FHIR untuk. AWS HealthLake Mereka termasuk sumber daya yang harus Anda sediakan di AWS akun Anda, pembuatan SMART di penyimpanan HealthLake data yang diaktifkan FHIR, dan contoh bagaimana SMART pada aplikasi klien FHIR berinteraksi dengan server otorisasi dan penyimpanan data. HealthLake

Topik

- Menyiapkan sumber daya untuk SMART di FHIR
- Alur kerja aplikasi klien untuk SMART di FHIR

Menyiapkan sumber daya untuk SMART di FHIR

Langkah-langkah berikut menentukan bagaimana SMART pada permintaan FHIR ditangani oleh HealthLake dan sumber daya yang dibutuhkan agar mereka berhasil. Elemen-elemen berikut bekerja sama dalam alur kerja untuk membuat SMART pada permintaan FHIR:

- Pengguna akhir: Umumnya, pasien atau dokter menggunakan SMART pihak ketiga pada aplikasi FHIR untuk mengakses data di penyimpanan data. HealthLake
- SMART pada aplikasi FHIR (disebut sebagai aplikasi klien): Aplikasi yang ingin mengakses data yang ditemukan di penyimpanan HealthLake data.
- Server otorisasi: Server yang sesuai dengan OpenID Connect yang dapat mengautentikasi pengguna dan mengeluarkan token akses.
- Penyimpanan HealthLake data: SMART pada penyimpanan HealthLake data berkemampuan FHIR yang menggunakan fungsi Lambda untuk menanggapi permintaan FHIR REST yang menyediakan token pembawa.

Agar elemen-elemen ini bekerja sama, Anda harus membuat sumber daya berikut.

> **Note**
>
> Sebaiknya buat SMART Anda di penyimpanan HealthLake data berkemampuan FHIR setelah Anda menyiapkan server otorisasi, menentukan cakupan yang diperlukan di dalamnya, dan membuat AWS Lambda fungsi untuk menangani introspeksi token.

1. Siapkan titik akhir server otorisasi

Untuk menggunakan kerangka kerja SMART pada FHIR, Anda perlu menyiapkan server otorisasi pihak ketiga yang dapat memvalidasi permintaan FHIR REST yang dibuat di penyimpanan data. Untuk informasi selengkapnya, lihat HealthLake persyaratan otentikasi untuk SMART di FHIR.

2. Tentukan cakupan pada server otorisasi Anda untuk mengontrol tingkat akses penyimpanan HealthLake data

Kerangka kerja SMART on FHIR menggunakan OAuth cakupan untuk menentukan sumber daya FHIR apa yang dapat diakses oleh permintaan yang diautentikasi dan sejauh mana. Mendefinisikan cakupan adalah cara untuk merancang hak istimewa paling sedikit. Untuk informasi selengkapnya, lihat SMART pada cakupan FHIR OAuth 2.0 didukung oleh HealthLake.

3. Siapkan AWS Lambda fungsi yang mampu melakukan introspeksi token

Permintaan FHIR REST yang dikirim oleh aplikasi klien pada SMART pada penyimpanan data berkemampuan FHIR berisi JSON Web Token (JWT). Untuk informasi lebih lanjut, lihat Menguraikan kode JWT.

4. Buat SMART di penyimpanan HealthLake data berkemampuan FHIR

Untuk membuat SMART di penyimpanan HealthLake data FHIR, Anda perlu menyediakan file`IdentityProviderConfiguration`. Untuk informasi selengkapnya, lihat Membuat penyimpanan HealthLake data.

Alur kerja aplikasi klien untuk SMART di FHIR

Bagian berikut menjelaskan cara meluncurkan aplikasi klien dan membuat permintaan FHIR REST yang berhasil pada penyimpanan HealthLake data dalam konteks SMART di FHIR.

1. Buat **`GET`** permintaan ke Pengenal Sumber Daya Seragam Terkenal menggunakan aplikasi klien

Aplikasi klien berkemampuan SMART harus membuat `GET` permintaan untuk menemukan titik akhir otorisasi penyimpanan HealthLake data Anda. Ini dilakukan melalui permintaan Pengenal Sumber Daya Seragam Terkenal (URI). Untuk informasi selengkapnya, lihat Mengambil SMART pada Dokumen Penemuan FHIR.

2. Minta akses dan cakupan

Aplikasi klien menggunakan titik akhir otorisasi dari server otorisasi, sehingga pengguna dapat login. Proses ini mengotentikasi pengguna. Cakupan digunakan untuk menentukan sumber daya FHIR apa yang ada di penyimpanan HealthLake data Anda yang dapat diakses oleh aplikasi klien. Untuk informasi selengkapnya, lihat SMART pada cakupan FHIR OAuth 2.0 didukung oleh HealthLake.

3. Token akses

Sekarang pengguna telah diautentikasi, aplikasi klien menerima token akses JWT dari server otorisasi. Token ini disediakan ketika aplikasi klien mengirimkan permintaan FHIR REST ke HealthLake. Untuk informasi selengkapnya, lihat Validasi token.

4. Buat permintaan FHIR REST API di SMART di penyimpanan data berkemampuan HealthLake FHIR

Aplikasi klien sekarang dapat mengirim permintaan FHIR REST API ke titik akhir penyimpanan HealthLake data menggunakan token akses yang disediakan oleh server otorisasi. Untuk informasi

selengkapnya, lihat Membuat permintaan FHIR REST API pada penyimpanan data berkemampuan SMART HealthLake.

5. Validasi token akses JWT

Untuk memvalidasi token akses yang dikirim dalam permintaan FHIR REST, gunakan fungsi Lambda. Lihat informasi yang lebih lengkap di Validasi Token menggunakan AWS Lambda.

HealthLake persyaratan otentikasi untuk SMART di FHIR

Untuk mengakses sumber daya FHIR di SMART pada penyimpanan HealthLake data berkemampuan FHIR, aplikasi klien harus diotorisasi oleh server otorisasi yang OAuth sesuai dengan 2.0 dan menyajikan token OAuth Pembawa sebagai bagian dari permintaan FHIR REST API. Untuk menemukan titik akhir server otorisasi, gunakan HealthLake SMART pada FHIR Discovery Document melalui `Well-Known` Uniform Resource Identifier. Untuk mempelajari lebih lanjut tentang proses ini, lihatMengambil SMART pada Dokumen Penemuan FHIR.

Saat Anda membuat SMART di penyimpanan HealthLake data FHIR, Anda harus menentukan titik akhir server otorisasi dan titik akhir token dalam `metadata` elemen permintaan. `CreateFHIRDatastore` Untuk mempelajari lebih lanjut tentang mendefinisikan `metadata` elemen, lihatMembuat penyimpanan HealthLake data.

Menggunakan titik akhir server otorisasi, aplikasi klien akan mengautentikasi pengguna dengan layanan otorisasi. Setelah diotorisasi dan diautentikasi, JSON Web Token (JWT) dihasilkan oleh layanan otorisasi dan diteruskan ke aplikasi klien. Token ini berisi cakupan sumber daya FHIR yang diizinkan untuk digunakan oleh aplikasi klien, yang pada gilirannya membatasi data apa yang dapat diakses pengguna. Secara opsional, jika ruang lingkup peluncuran disediakan maka respons akan berisi detail tersebut. Untuk mempelajari lebih lanjut tentang SMART pada cakupan FHIR yang didukung oleh HealthLake, lihat. SMART pada cakupan FHIR OAuth 2.0 didukung oleh HealthLake

Menggunakan JWT yang diberikan oleh server otorisasi, aplikasi klien membuat panggilan FHIR REST API ke SMART di penyimpanan data berkemampuan FHIR. HealthLake Untuk memvalidasi dan memecahkan kode JWT, Anda perlu membuat fungsi Lambda. HealthLake memanggil fungsi Lambda ini atas nama Anda saat permintaan FHIR REST API diterima. Untuk melihat contoh fungsi Lambda starter, lihat. Validasi Token menggunakan AWS Lambda

Elemen server otorisasi diperlukan untuk membuat SMART di penyimpanan data berkemampuan HealthLake FHIR

Dalam `CreateFHIRDatastore` permintaan, Anda perlu memberikan titik akhir otorisasi dan titik akhir token sebagai bagian dari `metadata` elemen dalam objek. `IdentityProviderConfiguration` Baik titik akhir otorisasi dan titik akhir token diperlukan. Untuk melihat contoh bagaimana ini ditentukan dalam `CreateFHIRDatastore` permintaan, lihatMembuat penyimpanan HealthLake data.

Klaim yang diperlukan untuk menyelesaikan permintaan FHIR REST API pada SMART di penyimpanan data berkemampuan HealthLake FHIR

AWS Lambda Fungsi Anda harus berisi klaim berikut agar menjadi permintaan FHIR REST API yang valid pada SMART di penyimpanan HealthLake data berkemampuan FHIR.

- `nbf`: (Bukan Sebelum) Klaim - Klaim "nbf" (bukan sebelumnya) mengidentifikasi waktu sebelum JWT TIDAK BOLEH diterima untuk diproses. Pemrosesan klaim "nbf" mensyaratkan bahwa arus date/time HARUS setelah atau sama dengan yang tidak date/time tercantum sebelumnya dalam klaim "nbf". Contoh fungsi Lambda yang kami sediakan mengkonversi `iat` dari respons server menjadi. `nbf`
- `exp`: (Waktu Kedaluwarsa) Klaim - Klaim "exp" (waktu kedaluwarsa) mengidentifikasi waktu kedaluwarsa pada atau setelah itu JWT tidak boleh diterima untuk diproses.
- `isAuthorized`: Sebuah boolean diatur ke`True`. Menunjukkan bahwa permintaan telah diotorisasi pada server otorisasi.
- `aud`: (Audiens) Klaim - Klaim "aud" (audiens) mengidentifikasi penerima yang dimaksudkan untuk JWT. Ini harus menjadi SMART pada titik akhir penyimpanan HealthLake data yang diaktifkan FHIR.
- `scope`: Ini harus setidaknya satu ruang lingkup terkait sumber daya FHIR. Cakupan ini didefinisikan pada server otorisasi Anda. Untuk mempelajari lebih lanjut tentang cakupan terkait sumber daya FHIR yang diterima oleh HealthLake, lihat. SMART pada cakupan sumber daya FHIR untuk HealthLake

SMART pada cakupan FHIR OAuth 2.0 didukung oleh HealthLake

HealthLake menggunakan OAuth 2.0 sebagai protokol otorisasi. Menggunakan protokol ini di server otorisasi Anda memungkinkan Anda untuk menentukan izin penyimpanan HealthLake data

(membuat, membaca, memperbarui, menghapus, dan mencari) untuk sumber daya FHIR yang dapat diakses oleh aplikasi klien.

SMART on FHIR framework mendefinisikan satu set cakupan yang dapat diminta dari server otorisasi. Misalnya, aplikasi klien yang hanya dirancang untuk memungkinkan pasien melihat hasil lab mereka atau melihat detail kontak mereka hanya boleh diizinkan untuk meminta `read` cakupan.

> **Note**
>
> HealthLake menyediakan dukungan untuk SMART pada FHIR V1 dan V2 seperti yang dijelaskan di bawah ini. SMART di FHIR `AuthorizationStrategy`diatur ke salah satu dari tiga nilai berikut saat penyimpanan data Anda dibuat:
>
> - `SMART_ON_FHIR_V1`— Dukungan hanya untuk SMART di FHIR V1, yang mencakup izin `read` (baca/cari) dan `write` (). create/update/delete
> - `SMART_ON_FHIR`— Support untuk SMART pada FHIR V1 dan V2, yang mencakup`create`,,`read`, `update`, dan `search` izin.
> - `AWS_AUTH`— Strategi AWS HealthLake otorisasi default; tidak berafiliasi dengan SMART di FHIR.

Lingkup peluncuran mandiri

HealthLake mendukung lingkup `launch/patient` mode peluncuran mandiri.

Dalam mode peluncuran mandiri aplikasi klien meminta akses ke data klinis pasien karena pengguna dan pasien tidak diketahui oleh aplikasi klien. Dengan demikian, permintaan otorisasi aplikasi klien secara eksplisit meminta ruang lingkup pasien dikembalikan. Setelah otentikasi berhasil, server otorisasi mengeluarkan token akses yang berisi lingkup pasien peluncuran yang diminta. Konteks pasien yang diperlukan disediakan bersama token akses dalam respons server otorisasi.

Cakupan mode peluncuran yang didukung

Lingkup	Deskripsi
`launch/patient`	Parameter dalam permintaan otorisasi OAuth 2.0 yang meminta agar data pasien dikembalikan dalam respons otorisasi.

SMART pada cakupan sumber daya FHIR untuk HealthLake

HealthLake mendefinisikan tiga tingkat SMART pada cakupan sumber daya FHIR.

- `patient`cakupan memberikan akses ke data spesifik tentang satu Pasien.
- `user`cakupan memberikan akses ke data tertentu yang dapat diakses pengguna.
- `system`cakupan memberikan akses ke semua sumber daya FHIR yang ditemukan di penyimpanan HealthLake data.

Bagian berikut mencantumkan sintaks untuk membangun cakupan sumber daya FHIR menggunakan SMART pada FHIR V1 atau SMART di FHIR V2.

> **Note**
>
> Strategi otorisasi SMART on FHIR diatur saat penyimpanan data Anda dibuat. Untuk informasi selengkapnya, lihat `AuthorizationStrategy` di dalam Referensi API AWS HealthLake .

SMART pada cakupan FHIR V1 didukung oleh HealthLake

Saat menggunakan SMART pada FHIR V1, sintaks umum untuk membangun cakupan sumber daya FHIR untuk berikut. HealthLake Untuk melihat seluruh jalur URL dalam contoh berikut, gulir ke atas tombol Salin.

```
('patient' | 'user' | 'system') '/' (fhir-resource | '*') '.' ('read' | 'write' | '*')
```

SMART pada cakupan otorisasi yang didukung FHIR v1

Sintaks lingkup	Contoh ruang lingkup	Hasil
`patient/(fhir-resource | '*').('read' | 'write' | '*')`	`patient/AllergyIntolerance.*`	Aplikasi klien pasien memiliki akses baca/tulis tingkat instance ke semua alergi yang tercatat.
`user/(fhir-resource | '*').('read' | 'write' | '*')`	`user/Observation.read`	Aplikasi klien pengguna memiliki

Sintaks lingkup	Contoh ruang lingkup	Hasil
		read/write akses tingkat instance ke semua pengamatan yang direkam.
`system/('read' | 'write' | *)`	`system/*.*`	Aplikasi klien sistem memiliki read/write akses ke semua data sumber daya FHIR.

SMART pada cakupan FHIR V2 didukung oleh HealthLake

Saat menggunakan SMART di FHIR V2, sintaks umum untuk membangun cakupan sumber daya FHIR untuk berikut. HealthLake Untuk melihat seluruh jalur URL dalam contoh berikut, gulir ke atas tombol Salin.

```
('patient' | 'user' | 'system') '/' (fhir-resource | '*') '.' ('c' | 'r' | 'u' | 'd' |
 's')
```

> **Note**
>
> Untuk menggunakan SMART pada FHIR V2, Anda harus meneruskan nilai `permission-v2`ke dalam `capabilities` string metadata, yang merupakan anggota dari tipe data. `IdentityProviderConfiguration`
> HealthLake mendukung cakupan granular. Untuk informasi lebih lanjut, lihat cakupan granular yang didukung di Panduan Implementasi Inti AS FHIR.

SMART pada cakupan otorisasi yang didukung FHIR V2

Sintaks lingkup	Contoh lingkup V1	Hasil
`patient/Observation.rs`	`user/Observation.read`	Izin untuk membaca dan mencari `Observation`

Sintaks lingkup	Contoh lingkup V1	Hasil
		sumber daya untuk pasien saat ini.
`system/*.cruds`	`system/*.*`	Aplikasi klien sistem memiliki create/read/update/delete/search akses penuh ke semua data sumber daya FHIR.

Validasi Token menggunakan AWS Lambda

Saat Anda membuat HealthLake SMART di penyimpanan data berkemampuan FHIR, Anda harus memberikan ARN fungsi AWS Lambda dalam `CreateFHIRDatastore` permintaan. ARN dari fungsi Lambda ditentukan dalam `IdentityProviderConfiguration` objek menggunakan parameter. `IdpLambdaArn`

Anda harus membuat fungsi Lambda sebelum membuat SMART Anda di penyimpanan data berkemampuan FHIR. Setelah Anda membuat penyimpanan data, Lambda ARN tidak dapat diubah. Untuk melihat Lambda ARN yang Anda tentukan saat penyimpanan data dibuat, gunakan tindakan API. `DescribeFHIRDatastore`

Agar permintaan FHIR REST berhasil di SMART di penyimpanan data berkemampuan FHIR, fungsi Lambda Anda harus melakukan hal berikut:

- Kembalikan respons dalam waktu kurang dari 1 detik ke titik akhir penyimpanan HealthLake data.
- Dekode token akses yang disediakan di header otorisasi permintaan REST API yang dikirim oleh aplikasi klien.
- Tetapkan peran layanan IAM yang memiliki izin yang cukup untuk melaksanakan permintaan FHIR REST API.
- Klaim berikut diperlukan untuk menyelesaikan permintaan FHIR REST API. Untuk mempelajari selengkapnya, lihat Klaim yang diperlukan.
  - `nbf`
  - `exp`
  - `isAuthorized`

- aud
- scope

Saat bekerja dengan Lambda, Anda perlu membuat peran eksekusi dan kebijakan berbasis sumber daya selain fungsi Lambda Anda. Peran eksekusi fungsi Lambda adalah peran IAM yang memberikan izin fungsi untuk mengakses layanan AWS dan sumber daya yang diperlukan pada waktu berjalan. Kebijakan berbasis sumber daya yang Anda berikan harus memungkinkan HealthLake untuk menjalankan fungsi Anda atas nama Anda.

Bagian dalam topik ini menjelaskan contoh permintaan dari aplikasi klien dan respons yang diterjemahkan, langkah-langkah yang diperlukan untuk membuat fungsi AWS Lambda, dan cara membuat kebijakan berbasis sumber daya yang dapat diasumsikan. HealthLake

- Bagian 1: Membuat fungsi Lambda
- Bagian 2: Membuat peran HealthLake layanan yang digunakan oleh fungsi AWS Lambda
- Bagian 3: Memperbarui peran eksekusi fungsi Lambda
- Bagian 4: Menambahkan kebijakan sumber daya ke fungsi Lambda Anda
- Bagian 5: Menyediakan konkurensi untuk fungsi Lambda Anda

Membuat fungsi AWS Lambda

Fungsi Lambda yang dibuat dalam topik ini dipicu saat HealthLake menerima permintaan ke SMART di penyimpanan data berkemampuan FHIR. Permintaan dari aplikasi klien berisi panggilan REST API, dan header otorisasi yang berisi token akses.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
Authorization: Bearer i8hweunweunweofiwweoijewiwe
```

Contoh fungsi Lambda dalam topik ini digunakan AWS Secrets Manager untuk mengaburkan kredensyal yang terkait dengan server otorisasi. Kami sangat menyarankan untuk tidak memberikan rincian login server otorisasi secara langsung dalam fungsi Lambda.

Example memvalidasi permintaan FHIR REST yang berisi token pembawa otorisasi

Contoh fungsi Lambda menunjukkan kepada Anda cara memvalidasi permintaan FHIR REST yang dikirim ke SMART di penyimpanan data berkemampuan FHIR. Untuk melihat step-by-steps petunjuk

tentang cara menerapkan fungsi Lambda ini, lihat. Membuat fungsi Lambda menggunakan Konsol Manajemen AWS

Jika permintaan FHIR REST API tidak berisi titik akhir penyimpanan data yang valid, token akses, dan operasi REST, fungsi Lambda akan gagal. Untuk mempelajari lebih lanjut tentang elemen server otorisasi yang diperlukan, lihatKlaim yang diperlukan.

```
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()
```

Membuat fungsi Lambda menggunakan Konsol Manajemen AWS

Prosedur berikut mengasumsikan Anda telah membuat peran layanan yang HealthLake ingin Anda asumsikan saat menangani permintaan FHIR REST API pada SMART di penyimpanan data berkemampuan FHIR. Jika Anda belum membuat peran layanan, Anda masih dapat membuat fungsi Lambda. Anda harus menambahkan ARN peran layanan sebelum fungsi Lambda berfungsi. Untuk mempelajari selengkapnya tentang membuat peran layanan dan menentukannya dalam fungsi Lambda, lihat Membuat peran HealthLake layanan untuk digunakan dalam fungsi AWS Lambda yang digunakan untuk memecahkan kode JWT

Untuk membuat fungsi Lambda ()Konsol Manajemen AWS

1. Buka halaman Fungsi di konsol Lambda.
2. Pilih Buat fungsi.
3. Pilih Penulis dari awal.
4. Di bawah Informasi dasar masukkan nama Fungsi. Di bawah Runtime pilih runtime berbasis python.

5. Untuk peran Eksekusi, pilih Buat peran baru dengan izin Lambda dasar.

   Lambda membuat peran eksekusi yang memberikan izin fungsi untuk mengunggah log ke Amazon. CloudWatch Fungsi Lambda mengasumsikan peran eksekusi saat Anda memanggil fungsi, dan menggunakan peran eksekusi untuk membuat kredensional SDK. AWS

6. Pilih tab Kode, dan tambahkan contoh fungsi Lambda.

   Jika Anda belum membuat peran layanan untuk fungsi Lambda untuk digunakan, Anda harus membuatnya sebelum fungsi Lambda sampel berfungsi. Untuk mempelajari selengkapnya tentang membuat peran layanan untuk fungsi Lambda, lihat. Membuat peran HealthLake layanan untuk digunakan dalam fungsi AWS Lambda yang digunakan untuk memecahkan kode JWT

```
import base64
import boto3
import logging
import json
import os
from urllib import request, parse

logger = logging.getLogger()
logger.setLevel(logging.INFO)

## Uses Secrets manager to gain access to the access key ID and secret access key
 for the authorization server
client = boto3.client('secretsmanager', region_name="region-of-datastore")
response = client.get_secret_value(SecretId='name-specified-by-customer-in-
secretsmanager')
secret = json.loads(response['SecretString'])
client_id = secret['client_id']
client_secret = secret['client_secret']


unencoded_auth = f'{client_id}:{client_secret}'
headers = {
  'Authorization': f'Basic {base64.b64encode(unencoded_auth.encode()).decode()}',
  'Content-Type': 'application/x-www-form-urlencoded'
}

auth_endpoint = os.environ['auth-server-base-url'] # Base URL of the Authorization
 server
```

```
user_role_arn = os.environ['iam-role-arn'] # The IAM role client application will
 use to complete the HTTP request on the datastore

def lambda_handler(event, context):
    if 'datastoreEndpoint' not in event or 'operationName' not in event or
 'bearerToken' not in event:
    return {}

    datastore_endpoint = event['datastoreEndpoint']
    operation_name = event['operationName']
    bearer_token = event['bearerToken']
    logger.info('Datastore Endpoint [{}], Operation Name:
 [{}]'.format(datastore_endpoint, operation_name))

    ## To validate the token
    auth_response = auth_with_provider(bearer_token)
    logger.info('Auth response: [{}]'.format(auth_response))
    auth_payload = json.loads(auth_response)
    ## Required parameters needed to be sent to the datastore endpoint for the HTTP
 request to go through
    auth_payload["isAuthorized"] = bool(auth_payload["active"])
    auth_payload["nbf"] = auth_payload["iat"]
    return {"authPayload": auth_payload, "iamRoleARN": user_role_arn}

## Access the server
def auth_with_provider(token):
    data = {'token': token, 'token_type_hint': 'access_token'}
    req = request.Request(url=auth_endpoint + '/v1/introspect',
 data=parse.urlencode(data).encode(), headers=headers)
    with request.urlopen(req) as resp:
    return resp.read().decode()
```

Memodifikasi peran eksekusi fungsi Lambda

Setelah membuat fungsi Lambda, Anda perlu memperbarui peran eksekusi untuk menyertakan izin yang diperlukan untuk memanggil Secrets Manager. Di Secrets Manager, setiap rahasia yang Anda buat memiliki ARN. Untuk menerapkan hak istimewa paling sedikit, peran eksekusi seharusnya hanya memiliki akses ke sumber daya yang diperlukan agar fungsi Lambda dapat dijalankan.

Anda dapat memodifikasi peran eksekusi fungsi Lambda dengan mencarinya di konsol IAM atau dengan memilih Konfigurasi di konsol Lambda. Untuk mempelajari selengkapnya tentang mengelola peran eksekusi fungsi Lambda, lihat. Peran pelaksanaan Lambda

Example Peran eksekusi fungsi Lambda yang memberikan akses ke **`GetSecretValue`**

Menambahkan tindakan IAM `GetSecretValue` ke peran eksekusi memberikan izin yang diperlukan agar fungsi Lambda sampel berfungsi.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:secret-name-DKodTA"
        }
    ]
}
```

Pada titik ini Anda telah membuat fungsi Lambda yang dapat digunakan untuk memvalidasi token akses yang disediakan sebagai bagian dari permintaan FHIR REST yang dikirim ke SMART Anda di penyimpanan data berkemampuan FHIR.

Membuat peran HealthLake layanan untuk digunakan dalam fungsi AWS Lambda yang digunakan untuk memecahkan kode JWT

Persona: IAM Administrator

Pengguna yang dapat menambah atau menghapus kebijakan IAM, dan membuat identitas IAM baru.

Peran layanan

Peran layanan adalah peran IAM yang diambil oleh sebuah layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Buat sebuah peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.

Setelah Token Web JSON (JWT) diterjemahkan, otorisasi Lambda juga perlu mengembalikan peran IAM ARN. Peran ini harus memiliki izin yang diperlukan untuk melaksanakan permintaan REST API atau akan gagal karena izin yang tidak mencukupi.

Saat menyiapkan kebijakan khusus menggunakan IAM, yang terbaik adalah memberikan izin minimum yang diperlukan. Untuk mempelajari selengkapnya, lihat Menerapkan izin hak istimewa terkecil di Panduan Pengguna IAM.

Membuat peran HealthLake layanan untuk menunjuk dalam fungsi Lambda otorisasi membutuhkan dua langkah.

- Pertama, Anda perlu membuat kebijakan IAM. Kebijakan harus menentukan akses ke sumber daya FHIR yang telah Anda berikan cakupan di server otorisasi.
- Kedua, Anda perlu membuat peran layanan. Saat Anda membuat peran, Anda menetapkan hubungan kepercayaan dan melampirkan kebijakan yang Anda buat di langkah pertama. Hubungan kepercayaan menunjuk HealthLake sebagai kepala layanan. Anda perlu menentukan ARN penyimpanan HealthLake data dan ID AWS akun di langkah ini.

Membuat kebijakan IAM baru

Cakupan yang Anda tentukan di server otorisasi Anda menentukan sumber daya FHIR apa yang dapat diakses oleh pengguna yang diautentikasi di penyimpanan data. HealthLake

Kebijakan IAM yang Anda buat dapat disesuaikan agar sesuai dengan cakupan yang telah Anda tetapkan.

Tindakan berikut dalam `Action` elemen pernyataan kebijakan IAM dapat didefinisikan. Untuk masing-masing `Action` dalam tabel Anda dapat mendefinisikan a`Resource types`. Dalam penyimpanan data HealthLake adalah satu-satunya jenis sumber daya yang didukung yang dapat didefinisikan dalam `Resource` elemen pernyataan kebijakan izin IAM.

Sumber daya FHIR individu bukanlah sumber daya yang dapat Anda definisikan sebagai elemen dalam kebijakan izin IAM.

Tindakan didefinisikan oleh HealthLake

Tindakan	Deskripsi	Tingk akses	Jenis sumber daya (Wajib)
CreateResource	Memberikan izin untuk membuat sumber daya	Tulis	Datastore ARN: arn:aws:healthlake: ::datastore/fhir/ **your-region 111122223333 your-datastore-id**
DeleteResource	Memberikan izin untuk menghapus sumber daya	Tulis	Datastore ARN: arn:aws:healthlake: ::datastore/fhir/ **your-region 111122223333 your-datastore-id**
ReadResource	Memberikan izin untuk membaca sumber daya	Baca	Datastore ARN: arn:aws:healthlake: ::datastore/fhir/ **your-region 111122223333 your-datastore-id**
SearchWithGet	Memberikan izin untuk mencari sumber daya dengan metode GET	Baca	Datastore ARN: arn:aws:healthlake: ::datastore/fhir/ **your-region 111122223333 your-datastore-id**
SearchWithPost	Memberikan izin untuk mencari sumber daya dengan metode POST	Baca	Datastore ARN: arn:aws:healthlake: ::datastore/fhir/ **your-region 111122223333 your-datastore-id**
MulaiFHIRExportJobWithPost	Memberikan izin untuk memulai pekerjaan Ekspor FHIR dengan GET	Tulis	Datastore ARN: arn:aws:healthlake: ::datastore/fhir/ **your-region 111122223333 your-datastore-id**
UpdateResource	Memberikan izin untuk memperbarui sumber daya	Tulis	Datastore ARN: arn:aws:healthlake: ::datastore/fhir/ **your-region 111122223333 your-datastore-id**

Untuk memulai, Anda dapat menggunakan`AmazonHealthLakeFullAccess`. Kebijakan ini akan memberikan pembacaan, tulis, pencarian, dan ekspor pada semua sumber daya FHIR

yang ditemukan di penyimpanan data. Untuk memberikan izin hanya-baca pada penggunaan penyimpanan data. `AmazonHealthLakeReadOnlyAccess`

Untuk mempelajari selengkapnya tentang membuat kebijakan kustom menggunakan Konsol Manajemen AWS, AWS CLI, atau IAM SDKs, lihat Membuat kebijakan IAM di Panduan Pengguna IAM.

Membuat peran layanan untuk HealthLake (konsol IAM)

Gunakan prosedur ini untuk membuat peran layanan. Saat Anda membuat layanan, Anda juga perlu menetapkan kebijakan IAM.

Untuk membuat peran layanan untuk HealthLake (konsol IAM)

1. Masuk ke Konsol Manajemen AWS dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi konsol IAM, pilih Peran.
3. Kemudian, pilih Buat peran.
4. Pada halaman Pilih entitas kepercayaan, pilih Kebijakan kepercayaan khusus.
5. Selanjutnya, di bawah Kebijakan kepercayaan khusus, perbarui kebijakan sampel sebagai berikut. Ganti **`your-account-id`** dengan nomor akun Anda, dan tambahkan ARN dari penyimpanan data yang ingin Anda gunakan dalam pekerjaan impor atau ekspor Anda.

   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. Lalu, pilih Selanjutnya.
7. Pada halaman Tambah izin, pilih kebijakan yang ingin diasumsikan oleh HealthLake layanan. Untuk menemukan kebijakan Anda, cari kebijakan tersebut di bawah Kebijakan Izin.
8. Kemudian, pilih Lampirkan kebijakan.
9. Kemudian pada halaman Nama, tinjau, dan buat di bawah Nama peran masukkan nama.
10. (Opsional) Kemudian di bawah Deskripsi, tambahkan deskripsi singkat untuk peran Anda.
11. Jika memungkinkan, masukkan nama peran atau akhiran nama peran untuk membantu Anda mengidentifikasi tujuan peran ini. Nama peran harus unik dalam diri Anda Akun AWS. Grup tidak dibedakan berdasarkan huruf besar-kecil. Misalnya, Anda tidak dapat membuat peran dengan nama **PRODROLE** dan **prodrole**. Anda tidak dapat mengubah nama peran setelah dibuat karena berbagai entitas mungkin mereferensikan peran tersebut.
12. Tinjau detail peran, lalu pilih Buat peran.

Untuk mempelajari cara menentukan peran ARN dalam contoh fungsi Lambda, lihat. Membuat fungsi AWS Lambda

Peran pelaksanaan Lambda

Peran eksekusi fungsi Lambda adalah peran IAM yang memberikan izin fungsi untuk mengakses AWS layanan dan sumber daya. Halaman ini memberikan informasi tentang cara membuat, melihat, dan mengelola peran eksekusi fungsi Lambda.

Secara default, Lambda membuat peran eksekusi dengan izin minimal saat Anda membuat fungsi Lambda baru menggunakan. Konsol Manajemen AWS Untuk mengelola izin yang diberikan dalam peran eksekusi, lihat Membuat peran eksekusi di konsol IAM di Panduan Pengembang Lambda.

Contoh fungsi Lambda yang disediakan dalam topik ini menggunakan Secrets Manager untuk mengaburkan kredensyal server otorisasi.

Seperti halnya peran IAM yang Anda buat, penting untuk mengikuti praktik terbaik yang paling tidak istimewa. Selama frase pengembangan, terkadang Anda mungkin memberikan izin di luar apa yang

diperlukan. Sebelum memublikasikan fungsi Anda di lingkungan produksi, sebagai praktik terbaik, sesuaikan kebijakan agar hanya menyertakan izin yang diperlukan. Untuk informasi selengkapnya, lihat Menerapkan hak istimewa terkecil di Panduan Pengguna IAM.

Izinkan HealthLake untuk memicu fungsi Lambda Anda

Jadi HealthLake dapat memanggil fungsi Lambda atas nama Anda, Anda harus melakukan berikut:

- Anda perlu mengatur `IdpLambdaArn` sama dengan ARN dari fungsi Lambda yang HealthLake ingin Anda panggil dalam permintaan. `CreateFHIRDatastore`
- Anda memerlukan kebijakan berbasis sumber daya yang memungkinkan untuk HealthLake menjalankan fungsi Lambda atas nama Anda.

Saat HealthLake menerima permintaan FHIR REST API pada SMART di penyimpanan data berkemampuan FHIR, diperlukan izin untuk menjalankan fungsi Lambda yang ditentukan pada pembuatan penyimpanan data atas nama Anda. Untuk memberikan HealthLake akses, Anda akan menggunakan kebijakan berbasis sumber daya. Untuk mempelajari selengkapnya tentang membuat kebijakan berbasis sumber daya untuk fungsi Lambda, lihat Mengizinkan layanan AWS memanggil fungsi Lambda di Panduan Pengembang.AWS Lambda

Menyediakan konkurensi untuk fungsi Lambda Anda

> **Important**
>
> HealthLake mengharuskan waktu berjalan maksimum untuk fungsi Lambda Anda kurang dari satu detik (1000 milidetik).
> Jika fungsi Lambda Anda melebihi batas waktu berjalan, Anda mendapatkan pengecualian. TimeOut

Untuk menghindari pengecualian ini, kami sarankan untuk mengonfigurasi konkurensi yang disediakan. Dengan mengalokasikan konkurensi yang disediakan sebelum peningkatan pemanggilan, Anda dapat memastikan bahwa semua permintaan dilayani oleh instance yang diinisialisasi dengan latensi rendah. Untuk mempelajari lebih lanjut tentang mengonfigurasi konkurensi yang disediakan, lihat Mengonfigurasi konkurensi yang disediakan di Panduan Pengembang Lambda

Untuk melihat rata-rata waktu berjalan untuk fungsi Lambda Anda saat ini gunakan halaman Pemantauan untuk fungsi Lambda Anda di konsol Lambda. Secara default, konsol Lambda menyediakan grafik Durasi yang menunjukkan jumlah waktu rata-rata, minimum, dan maksimum waktu yang dihabiskan kode fungsi untuk memproses suatu peristiwa. Untuk mempelajari selengkapnya tentang memantau fungsi Lambda, lihat Memantau fungsi di konsol Lambda di Panduan Pengembang Lambda.

Jika Anda telah menyediakan konkurensi untuk fungsi Lambda Anda dan ingin memantaunya, lihat Memantau konkurensi di Panduan Pengembang Lambda.

Menggunakan otorisasi berbutir halus dengan SMART pada penyimpanan data yang diaktifkan FHIR HealthLake

Cakupan saja tidak memberi Anda kekhususan yang diperlukan tentang data apa yang diizinkan untuk diakses oleh pemohon di penyimpanan data. Menggunakan otorisasi berbutir halus memungkinkan tingkat spesifisitas yang lebih tinggi saat memberikan akses ke SMART di penyimpanan data berkemampuan FHIR. HealthLake Untuk menggunakan otorisasi berbutir halus, tetapkan `FineGrainedAuthorizationEnabled` sama dengan `IdentityProviderConfiguration` parameter `True` permintaan Anda. `CreateFHIRDatastore`

Jika Anda mengaktifkan otorisasi berbutir halus, server otorisasi Anda mengembalikan `fhirUser` cakupan `id_token` bersama dengan token akses. Hal ini memungkinkan informasi tentang Pengguna untuk diambil oleh aplikasi klien. Aplikasi klien harus memperlakukan `fhirUser` klaim sebagai URI dari sumber daya FHIR yang mewakili pengguna saat ini. Ini dapat berupa `Patient`, `Practitioner`, atau `RelatedPerson`. Respons server otorisasi juga mencakup `user/` ruang lingkup yang mendefinisikan data apa yang dapat diakses pengguna. Ini menggunakan sintaks yang ditentukan untuk cakupan yang terkait dengan cakupan spesifik sumber daya FHIR:

```
user/(fhir-resource | '*').('read' | 'write' | '*')
```

Berikut ini adalah contoh bagaimana otorisasi halus dapat digunakan untuk lebih menentukan akses data terkait jenis sumber daya FHIR.

- `fhirUser`Kapan otorisasi `Practitioner` berbutir halus menentukan kumpulan pasien yang dapat diakses pengguna. Akses ke hanya `fhirUser` diperbolehkan untuk pasien di mana Pasien memiliki referensi ke `fhirUser` sebagai Dokter Umum.

  ```
  Patient.generalPractitioner : [{Reference(Practitioner)}]
  ```

- `fhirUser`Kapan `Patient` atau `RelatedPerson` dan pasien yang dirujuk dalam permintaan berbeda dari`fhirUser`, otorisasi berbutir halus menentukan akses ke `fhirUser` pasien yang diminta. Akses diizinkan ketika ada hubungan yang ditentukan dalam `Patient` sumber daya yang diminta.

```
Patient.link.other : {Reference(Patient|RelatedPerson)}
```

Mengambil SMART pada Dokumen Penemuan FHIR

SMART mendefinisikan Dokumen Penemuan yang memungkinkan klien mempelajari titik akhir otorisasi URLs dan fitur yang didukung penyimpanan HealthLake data. Informasi ini membantu klien mengarahkan permintaan otorisasi ke titik akhir yang tepat dan membuat permintaan otorisasi yang didukung penyimpanan data. HealthLake

Agar aplikasi klien dapat membuat permintaan FHIR REST yang berhasil HealthLake, aplikasi tersebut harus mengumpulkan persyaratan otorisasi yang ditentukan oleh penyimpanan HealthLake data. Token pembawa (otorisasi) tidak diperlukan agar permintaan ini berhasil..

Untuk meminta Discovery Document untuk penyimpanan HealthLake data

1. Kumpulkan HealthLake `region` dan `datastoreId` nilai. Untuk informasi selengkapnya, lihat Mendapatkan properti penyimpanan data.
2. Buat URL untuk permintaan menggunakan nilai yang dikumpulkan untuk HealthLake `region` dan`datastoreId`. Tambahkan `/.well-known/smart-configuration` ke titik akhir URL. Untuk melihat seluruh jalur URL dalam contoh berikut, gulir ke atas tombol Salin.

   ```
   https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/.well-known/smart-configuration
   ```

3. Kirim permintaan menggunakan `GET` protokol penandatanganan AWS Signature Version 4. Untuk melihat seluruh contoh, gulir ke atas tombol Salin.

   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'
```

Dokumen Discovery untuk penyimpanan HealthLake data kembali sebagai gumpalan JSON, di mana Anda dapat menemukan `authorization_endpoint` dan`token_endpoint`, bersama dengan spesifikasi dan kemampuan yang ditentukan untuk penyimpanan data.

```
{
    "authorization_endpoint": "https://oidc.example.com/authorize",
    "token_endpoint": "https://oidc.example.com/oauth/token",
    "capabilities": [
        "launch-ehr",
        "client-public"
    ]
}
```

Baik `authorization_endpoint` dan yang `token_endpoint` diperlukan untuk meluncurkan aplikasi klien.

- Titik akhir otorisasi — URL yang diperlukan untuk mengotorisasi aplikasi klien atau pengguna.
- Titik akhir Token — Titik akhir dari server otorisasi yang digunakan aplikasi klien untuk berkomunikasi dengan.

Membuat permintaan FHIR REST API pada penyimpanan data berkemampuan SMART HealthLake

Anda dapat membuat permintaan FHIR REST API pada SMART di penyimpanan data berkemampuan FHIR HealthLake . Contoh berikut menunjukkan permintaan dari aplikasi klien yang berisi JWT di header otorisasi dan bagaimana Lambda harus memecahkan kode respons. Setelah permintaan aplikasi klien diotorisasi dan diautentikasi, ia harus menerima token pembawa dari server otorisasi. Gunakan token pembawa di header otorisasi saat mengirim permintaan FHIR REST API pada SMART di penyimpanan data berkemampuan HealthLake FHIR.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/[ID]
Authorization: Bearer auth-server-provided-bearer-token
```

Karena token pembawa ditemukan di header otorisasi dan tidak ada identitas AWS IAM yang terdeteksi HealthLake memanggil fungsi Lambda yang ditentukan saat penyimpanan data yang diaktifkan SMART pada HealthLake FHIR dibuat. Ketika token berhasil diterjemahkan oleh fungsi Lambda Anda, contoh respons berikut dikirim ke. HealthLake

```
{
  "authPayload": {
    "iss": "https://authorization-server-endpoint/oauth2/token", # The issuer
 identifier of the authorization server
    "aud": "https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/", #
 Required, data store endpoint
    "iat": 1677115637,  # Identifies the time at which the token was issued
    "nbf": 1677115637,  # Required, the earliest time the JWT would be valid
    "exp": 1997877061,  # Required, the time at which the JWT is no longer valid
    "isAuthorized": "true",  # Required, boolean indicating the request has been
 authorized
    "uid": "100101",  # Unique identifier returned by the auth server
    "scope": "system/*.*" # Required, the scope of the request
  },
  "iamRoleARN": "iam-role-arn" #Required, IAM role to complete the request
}
```

Dukungan FHIR R4 untuk AWS HealthLake

AWS HealthLake mendukung spesifikasi FHIR R4 untuk pertukaran data kesehatan. Bagian berikut memberikan informasi pendukung tentang bagaimana HealthLake memanfaatkan spesifikasi FHIR R4 untuk membantu Anda mengelola dan mencari sumber daya FHIR di penyimpanan HealthLake data Anda menggunakan FHIR R4. RESTful APIs

Topik

- Pernyataan Kemampuan FHIR R4 untuk AWS HealthLake
- Validasi profil FHIR untuk HealthLake
- Jenis sumber daya yang didukung FHIR R4 untuk HealthLake
- Parameter pencarian FHIR R4 untuk HealthLake
- FHIR $operations R4 untuk HealthLake

Pernyataan Kemampuan FHIR R4 untuk AWS HealthLake

Untuk menemukan kemampuan (perilaku) terkait fHIR dari penyimpanan HealthLake data aktif, Anda harus mengambil Pernyataan Kemampuannya. Pernyataan Kemampuan digunakan sebagai pernyataan fungsionalitas server aktual atau pernyataan implementasi server yang diperlukan atau diinginkan. `capabilities`Interaksi FHIR mengambil informasi tentang kemampuan penyimpanan HealthLake data dan bagian mana dari spesifikasi FHIR yang didukungnya. HealthLake memvalidasi jenis sumber daya FHIR sesuai dengan sumber daya FHIR R4. `StructureDefinition`

Untuk mendapatkan Pernyataan Kemampuan untuk penyimpanan HealthLake data

1. Kumpulkan HealthLake `region` dan `datastoreId` nilai. Untuk informasi selengkapnya, lihat Mendapatkan properti penyimpanan data.
2. Buat URL untuk permintaan menggunakan nilai yang dikumpulkan untuk HealthLake `region` dan`datastoreId`. Sertakan juga `metadata` elemen FHIR di URL. Untuk melihat seluruh jalur URL dalam contoh berikut, gulir ke atas tombol Salin.

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

3. Kirim permintaan . `capabilities`Interaksi FHIR menggunakan `GET` permintaan dengan protokol penandatanganan AWS Signature Version 4. `curl`Contoh berikut mendapatkan Pernyataan Kemampuan untuk penyimpanan HealthLake data yang ditentukan oleh`datastoreId`. Untuk melihat seluruh contoh, gulir ke atas tombol Salin.

   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'
   ```

   Anda akan menerima kode respons `200` HTTP dan Pernyataan Kemampuan untuk penyimpanan HealthLake data Anda. Untuk informasi lebih lanjut, lihat `CapabilityStatement`di dokumentasi FHIR R4.

Validasi profil FHIR untuk HealthLake

AWS HealthLake mendukung spesifikasi dasar FHIR R4. Termasuk dalam spesifikasi dasar FHIR R4 adalah Profil FHIR. Profil digunakan pada tipe sumber daya FHIR untuk menentukan definisi jenis sumber daya yang lebih spesifik menggunakan and/or ekstensi kendala pada jenis sumber daya dasar. Misalnya, Profil FHIR dapat mengidentifikasi bidang wajib seperti ekstensi dan set nilai. Sumber daya dapat mendukung banyak profil. Semua HealthLake data menyimpan dukungan menggunakan Profil FHIR.

> **Note**
>
> Menambahkan profil FHIR tidak diperlukan saat menambahkan data ke penyimpanan HealthLake data. Jika profil FHIR tidak ditentukan ketika sumber daya ditambahkan atau diperbarui, sumber daya divalidasi hanya terhadap skema FHIR R4 dasar.
> Profil FHIR, yang sesuai dengan sumber daya FHIR, termasuk dalam sumber daya sebelum diimpor. HealthLake Oleh karena itu, profil FHIR divalidasi oleh HealthLake selama impor.

Profil FHIR ditentukan dalam panduan implementasi. Panduan Implementasi FHIR (IG) adalah seperangkat instruksi yang menjelaskan cara menggunakan standar FHIR untuk tujuan tertentu. HealthLake memvalidasi Profil FHIR didefinisikan dalam panduan implementasi berikut.

Profil FHIR didukung oleh AWS HealthLake

Nama	Ve	Panduan implementasi	Kemampuan	AS Tin (Ol	AS Tin (Vi Uta	US We (Or	Asi Pa (Mu	Asi Pa (Sy	Ka (Pu	Erc (Irl)	Eropa (London)
Inti AS	3.1	http://hl7.org/fhir/us/core/STU3.1.1/	Default	X	X	X	X	X	X	X	X
Inti AS	4.0	https://hl7.org/fhir/us/core/STU4/index.html	Didukung	X	X	X	X	X	X	X	X
Inti AS	5.0	https://hl7.org/fhir/us/core/STU5.0.1/index.html	Didukung	X	X	X	X	X	X	X	X

Nama	Ve	Panduan implementasi	Kemampuan	AS Tin (Ol	AS Tin (Vi Uta	US We (Or	Asi Pa (Mu	Asi Pa (Sy	Ka (Pu	Erc (Irl)	Eropa (London)
Inti AS	6.1	https://hl7.org/fhir/us/core/STU6.1/index.html	Didukung	X	X	X	X	X	X	X	X
Inti AS	7.0	https://hl7.org/fhir/us/core/STU7/	Didukung	X	X	X	X	X	X	X	X
Inti Inggris	2.0	https://simplifier.net/guide/uk-core-implementation-guide-stu2/Home/ProfilesandExtensions/ProfilesIndex?version=2.0.1	Didukung	X	X	X	X			X	X
Tombol Biru CARIN	1.1	http://hl7.org/fhir/us/carin-bb/STU1.1/	Default	X	X	X	X	X	X	X	X
Tombol Biru CARIN	1.0 2.0 2.1	https://hl7.org/fhir/us/carin-bb/history.html	Didukung	X	X	X	X	X	X	X	X
Da Vinci Payer Data Exchange	1.0	https://hl7.org/fhir/us/davinci-pdex/	Default	X	X	X	X	X	X	X	X
Da Vinci Payer Data Exchange	2.0 2.1	https://hl7.org/fhir/us/davinci-pdex/history.html	Didukung	X	X	X	X	X	X	X	X

Nama	Ve	Panduan implementasi	Kemampuan	AS Tin (Ol	AS Tin (Vi Uta	US We (Or	Asi Pa (Mu	Asi Pa (Sy	Ka (Pu	Erc (Irl)	Eropa (London)
Pertukaran Catatan Kesehatan Da Vinci (HRex)	0.2	https://hl7.org/fhir/us/davinci-hrex/2020Sep/	Default	X	X	X	X	X	X	X	X
DaVinci Paket PDEX Bersih	1.1	https://hl7.org/fhir/us/davinci-pdex-plan-net/STU1.1/	Default	X	X	X	X	X	X	X	X
DaVinci Paket PDEX Bersih	1.0	https://hl7.org/fhir/us/davinci-pdex-plan-net/STU1/	Didukung	X	X	X	X	X	X	X	X
DaVinci Pembayar Data Exchange (PDeX) Formularium Obat AS	1.1	https://hl7.org/fhir/us/davinci-drug-formulary/STU1.1/	Default	X	X	X	X	X	X	X	X
DaVinci Pembayar Data Exchange (PDeX) Formularium Obat AS	1.0 2.0 2.1	https://hl7.org/fhir/us/davinci-drug-formulary/history.html	Didukung	X	X	X	X	X	X	X	X

Nama	Ve	Panduan implementasi	Kemampuan	AS Tin (Ol	AS Tin (Vi Uta	US We (Or	Asi Pa (Mu	Asi Pa (Sy	Ka (Pu	Erc (Irl)	Eropa (London)
Da Vinci Clinical Data Exchange (CDeX)	2.1	https://build.fhir.org/ig/HL7/davinci-ecdx/index.html	Default	X	X	X	X	X	X	X	X
Da Vinci Prior Authorization Support (PAS) FHIR IG	2.1	https://hl7.org/fhir/us/davinci-pas/	Default	X	X	X	X	X	X	X	X
Panduan Implementasi NCQA HEDIS®	0.3	https://www.ncqa.org/resources/hedis-ig-resource-page/	Default	X	X	X	X			X	X
Ringkasan Pasien Internasional (IPS)	2.0 sur at sua	https://hl7.org/fhir/uv/ips/2024Sep/	Default	X	X	X	X	X	X	X	X
Ukuran Kualitas	5.0	https://registry.fhir.org/package/hl7.fhir.us.cqfmeasures%7C5.0.0	Default	X	X	X	X			X	X
Pelaporan Genomik	3.0	https://build.fhir.org/ig/HL7/genomics-reporting/index.html	Default	X	X	X	X			X	X

Nama	Ve	Panduan implementasi	Kemampuan	AS Tin (Ol	AS Tin (Vi Uta	US We (Or	Asi Pa (Mu	Asi Pa (Sy	Ka (Pu	Erc (Irl)	Eropa (London)
Misi Digital Ayushman Bharat Otoritas Kesehatan Nasional (ABDM)	2.0	https://www.nrces.in/ndhm/fhir/r4/index.html	Default	X	X	X	X			X	X
Inti CA +	1.1	https://simplifier.net/ca-core	Didukung						X		
CA:eReC Pan-Canadian e Referral-eConsult	1.1	https://simplifier.net/CA-eReC/~introduction	Didukung						X		
Ringkasan Pasien Edisi Kanada - (PS-CA)	2.1	https://simplifier.net/PS-CA-R1/~introduction	Didukung						X		
Repositori Obat Kesehatan Digital Ontario	4.0	https://simplifier.net/ca-on-dhdr-r4/~introduction	Didukung						X		
Inti AU	1.0	https://hl7.org.au/fhir/core/	Didukung					X			

Nama	Ve	Panduan implementasi	Kemampuan	AS Tin (Ol	AS Tin (Vi Uta	US We (Or	Asi Pa (Mu	Asi Pa (Sy	Ka (Pu	Erc (Irl)	Eropa (London)
Manajemen Praktik Magentus	1.2	https://fhir-versions.dev.geniesolutions.io/1.2.16/downloads.html	Didukung					X			

Memvalidasi profil FHIR yang ditentukan dalam sumber daya

Untuk Profil FHIR yang akan divalidasi tambahkan ke `profile` elemen sumber daya individu menggunakan URL profil yang ditunjuk dalam panduan implementasi.

Profil FHIR divalidasi saat Anda menambahkan sumber daya baru ke penyimpanan data Anda. Untuk menambahkan sumber daya baru, Anda dapat menggunakan operasi `StartFHIRImportJob` API, membuat `POST` permintaan untuk menambahkan sumber daya baru, atau `PUT` membuat pembaruan sumber daya yang ada.

Example— Untuk melihat profil FHIR mana yang direferensikan dalam sumber daya

URL profil ditambahkan ke `profile` elemen dalam pasangan `"meta" : "profile"` kunci-nilai. Sumber daya ini dipotong untuk kejelasan.

```
{
    "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— Cara mereferensikan profil FHIR yang didukung non-default

Untuk memvalidasi profil non-default yang didukung, tambahkan URL profil berversi ke elemen. `meta.profile` URL berversi menyertakan URL profil dasar diikuti oleh karakter pipa (`|`) dan nomor versi. Sumber daya contoh ini dipotong untuk kejelasan.

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

Anda juga dapat menyertakan URL berversi dan URL profil dasar. Kedua format tersebut valid.

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

Jenis sumber daya yang didukung FHIR R4 untuk HealthLake

Tabel berikut mencantumkan jenis sumber daya FHIR R4 yang didukung oleh. AWS HealthLake Untuk informasi selengkapnya, lihat Indeks Sumber Daya dalam dokumentasi FHIR R4.

Jenis sumber daya FHIR R4 didukung oleh HealthLake

Akun	DetectedIssue	Faktur	Praktisi

ActivityDefinition	Perangkat	Perpustakaan	PractitionerRole
AdverseEvent	DeviceDefinition	Keterkaitan	Prosedur
AllergyIntolerance	DeviceMetric	Daftar	Asal
Janji	DeviceUseStatement	Lokasi	Kuesioner
AppointmentResponse	DeviceRequest	Ukur	QuestionnaireResponse
AuditEvent - Lihat Catatan	DiagnosticReport	MeasureReport	RelatedPerson
Biner	DocumentManifest	Media	RequestGroup
BodyStructure	DocumentReference	Obat-obatan	ResearchStudy
Bundel - Lihat Catatan	EffectEvidenceSynthesis	MedicationAdministration	ResearchSubject
CapabilityStatement	Perjumpaan	MedicationDispense	RiskAssessment
CarePlan	Titik akhir	MedicationKnowledge	RiskEvidenceSynthesis
CareTeam	EpisodeOfCare	MedicationRequest	Jadwal
ChargeItem	EnrollmentRequest	MedicationStatement	ServiceRequest
ChargeItemDefinition	EnrollmentResponse	MessageHeader	Slot
Klaim	ExplanationOfBenefit	MolecularSequence	Spesimen
ClaimResponse	FamilyMemberHistory	NutritionOrder	StructureDefinition
Komunikasi	Bendera	Observasi	StructureMap
CommunicationRequest	Tujuan	OperationOutcome - Lihat Catatan	Substansi

Komposisi	Kelompok	Organisasi	SupplyDelivery
ConceptMap	GuidanceResponse	OrganizationAffiliation	SupplyRequest
Kondisi	HealthcareService	Parameter - Lihat Catatan	Tugas
Persetujuan	ImagingStudy	Pasien	ValueSet
Kontrak	Imunisasi	PaymentNotice	VisionPrescription
Cakupan	ImmunizationEvaluation	PaymentReconciliation	VerificationResult - Lihat Catatan
CoverageEligibilityRequest	ImmunizationRecommendation	Orang	
CoverageEligibilityResponse	InsurancePlan	PlanDefinition	

⚠ Spesifikasi dan spesifikasi FHIR HealthLake

- Anda tidak dapat membuat `GET` atau `POST` meminta dengan FHIR `OperationOutcome` dan jenis `Parameters` sumber daya.
- AuditEvent AuditEvent Sumber daya dapat dibuat atau dibaca, tetapi tidak dapat diperbarui atau dihapus.
- Bundel — Ada beberapa cara HealthLake mengelola permintaan Bundle. Untuk detail selengkapnya, lihat Bundling sumber daya FHIR.
- VerificationResult- Jenis sumber daya ini hanya didukung untuk penyimpanan data yang dibuat setelah 09 Desember 2023.

Parameter pencarian FHIR R4 untuk HealthLake

Gunakan `search`interaksi FHIR untuk mencari satu set sumber daya FHIR di penyimpanan HealthLake data berdasarkan beberapa kriteria filter. `search`Interaksi dapat dilakukan dengan menggunakan permintaan `GET` atau `POST` permintaan. Untuk pencarian yang melibatkan informasi

identitas pribadi (PII) atau informasi kesehatan yang dilindungi (PHI), disarankan untuk menggunakan `POST` permintaan, karena PII dan PHI ditambahkan sebagai bagian dari badan permintaan dan dienkripsi dalam perjalanan.

> **Note**
>
> `search`Interaksi FHIR yang dijelaskan dalam Bab ini dibangun sesuai dengan standar HL7 FHIR R4 untuk pertukaran data perawatan kesehatan. Karena merupakan representasi dari layanan HL7 FHIR, itu tidak ditawarkan melalui AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat `search`di dokumentasi FHIR R4 RESTful API.
> Anda juga dapat menanyakan penyimpanan HealthLake data dengan SQL menggunakan Amazon Athena. Untuk informasi selengkapnya, lihat Mengintegrasikan.

HealthLake mendukung subset parameter pencarian FHIR R4 berikut. Untuk informasi selengkapnya, lihat Parameter pencarian FHIR R4 untuk HealthLake.

Jenis parameter pencarian yang didukung

Tabel berikut menunjukkan jenis parameter pencarian yang didukung di HealthLake.

Jenis parameter pencarian yang didukung

Parameter pencarian	Deskripsi
_id	Id sumber daya (bukan URL lengkap)
_LastUpdated	Tanggal terakhir diperbarui. Server memiliki kebijaksanaan pada presisi batas.
_tag	Cari berdasarkan tag sumber daya.
_profil	Cari semua sumber daya yang ditandai dengan profil.
_keamanan	Cari pada label keamanan yang diterapkan ke sumber daya ini.
_sumber	Cari dari mana sumber daya berasal.

Parameter pencarian	Deskripsi
_teks	Cari di narasi sumber daya.
createdAt	Cari di ekstensi kustom CreateDat.

> **Note**
>
> Parameter pencarian berikut hanya didukung untuk datastores yang dibuat setelah 09 Desember 2023: _security, _source, _text, createDat.

Tabel berikut menunjukkan contoh cara memodifikasi string kueri berdasarkan tipe data tertentu untuk jenis sumber daya tertentu. Untuk kejelasan, karakter khusus di kolom contoh belum dikodekan. Untuk membuat kueri yang berhasil, pastikan bahwa string kueri telah dikodekan dengan benar.

Cari contoh parameter

Cari Jenis Parameter	Detail	Contoh

Bilangan	Mencari nilai numerik dalam sumber daya tertentu. Angka-angka signifikan diamati. Jumlah digit signifikan spesifik berdasarkan nilai parameter pencarian, tidak termasuk nol di depan. Awalan perbandin gan diperbolehkan.	`[parameter]=100`
		`[parameter]=1e2`
		`[parameter]=lt100`

Tanggal/ DateTime	Mencari tanggal atau waktu tertentu. Format yang diharapkan adalah `yyyy-mm-ddThh:mm:ss[Z|(+|-)hh:mm]` tetapi dapat bervariasi.	`[parameter]=eq2013-01-14`
		`[parameter]=gt2013-01-14T10:00`
		`[parameter]=ne2013-01-14`

Cari Jenis Parameter	Detail	Contoh
	Menerima tipe data berikut:`date`,,`dateTime`, `instantPeriod`, dan`Timing`. Untuk detail selengkapnya menggunakan tipe data ini dalam penelusuran, lihat tanggal dalam dokumentasi FHIR R4 API RESTful . Awalan perbandingan diperbolehkan.	

String	Mencari urutan karakter dengan cara yang peka huruf besar/kecil.
	
	Mendukung keduanya `HumanName` dan `Address` jenis. Untuk lebih jelasnya, lihat entri tipe HumanName data dan entri tipe `Address` data dalam dokumentasi FHIR R4.
	
	Pencarian lanjutan didukung menggunakan `:text` pengubah.
		`[base]/Patient?given=eve`
		
		`[base]/Patient?given:contains=eve`

Cari Jenis Parameter	Detail	Contoh
Token	Mencari close-to-exact kecocokan terhadap serangkaian karakter, sering dibandingkan dengan sepasang nilai kode medis. Sensitivitas kasus ditautkan ke sistem kode yang digunakan saat membuat kueri. Kueri berbasis subsumption dapat membantu mengurangi masalah yang terkait dengan sensitivitas huruf besar. Untuk kejelasan `|` belum dikodekan.	`[parameter]=[system]|[code]` Di sini `[system]` mengacu pada sistem pengkodean, dan `[code]` mengacu pada nilai kode yang ditemukan dalam sistem tertentu. `[parameter]=[code]` : Di sini masukan Anda akan cocok dengan kode atau sistem. `[parameter]=|[code]` : Di sini masukan Anda akan cocok dengan kode, dan properti sistem tidak memiliki pengenal.

Komposit	Mencari beberapa parameter dalam satu jenis sumber daya, menggunakan pengubah `$` dan `,` operasi.
	
	Awalan perbandingan diperbolehkan.
		`/Patient?language=FR,NL&language=EN`
		
		`Observation?component-code-value-quantity=http://loinc.org|8480-6$lt60`
		
		`[base]/Group?characteristic-value=gender$mixed`

Cari Jenis Parameter	Detail	Contoh

Kuantitas	Mencari nomor, sistem, dan kode sebagai nilai. Diperlukan nomor, tetapi sistem dan kode bersifat opsional. Berdasarkan tipe data Kuantitas. Untuk lebih jelasnya, lihat Kuantitas dalam dokumentasi FHIR R4.
	
	Menggunakan sintaks yang diasumsikan berikut `[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`

Referensi	Mencari referensi ke sumber daya lain.	`[base]/Observation?subject=Patient/23`
		
		`test`

URI	Mencari serangkaian karakter yang secara jelas mengidentifikasi sumber daya tertentu.	`[base]/ValueSet?url=http://acme.org/fhir/ValueSet/123`
Khusus	Pencarian berdasarkan ekstensi NLP medis terintegrasi.	

Parameter pencarian lanjutan yang didukung oleh HealthLake

HealthLake mendukung parameter pencarian lanjutan berikut.

Nama	Deskripsi	Contoh	Kemampuan
`_includ`	Digunakan untuk meminta agar sumber daya tambahan dikembalikan dalam permintaan pencarian. Ia mengembalikan sumber daya yang direferensikan oleh instance sumber daya target.	`Encounter?_include=Encounter:subject`	
`_revinc de`	Digunakan untuk meminta agar sumber daya tambahan dikembalikan dalam permintaan pencarian. Ia mengembalikan sumber daya yang mereferensikan contoh sumber daya utama.	`Patient?_id=` **`patient-identifier`** `&_revinclude=Encounter:patient`	
`_summary`	Ringkasan dapat digunakan untuk meminta subset dari sumber daya.	`Patient?_summary=text`	Parameter ringkasan berikut didukung:`_summary=true` ,`_summary=false` ,`_summary=text` ,`_summary=data` .
`_elemen`	Minta sekumpulan elemen tertentu yang akan dikembalikan sebagai bagian dari sumber daya dalam hasil pencarian.	`Patient?_elements=identifier,active,link`	
`_total`	Mengembalikan jumlah sumber daya yang cocok dengan parameter pencarian.	`Patient?_total=accurate`	Support`_total=accurate` ,`_total=none` .
`_sort`	Tunjukkan urutan pengurutan hasil pencarian yang dikembalikan menggunakan daftar yang dipisahkan koma. -Awalan	`Observation?_sort=status,-date`	Support mengurutkan berdasarkan bidang dengan tipeNumber, String,

Nama	Deskripsi	Contoh	Kemampuan
	dapat digunakan untuk aturan pengurutan apa pun dalam daftar yang dipisahkan koma untuk menunjukkan urutan menurun.		`Quantity, Token, URI, Reference` . Urutkan berdasarkan hanya `Date` didukung untuk penyimpan an data yang dibuat setelah 09 Desember 2023. Support hingga 5 aturan pengurutan.
`_count`	Kontrol berapa banyak sumber daya yang dikembalikan per halaman bundel pencarian.	`Patient?_ count=100`	Ukuran halaman maksimum adalah 100.
`chainin`	Cari elemen sumber daya yang direferensikan. `.`Mengarahkan pencarian dirantai ke elemen dalam sumber daya yang direferen sikan.	`Diagnosti cReport?s ubject:Pa tient.nam e=peter`	
`reverse chainin (_has)`	Cari sumber daya berdasark an elemen sumber daya yang merujuknya.	`Patient?_ has:Obser vation:pa tient:cod e=1234-5`	

`_include`

Menggunakan `_include` dalam kueri penelusuran memungkinkan sumber daya FHIR tambahan yang ditentukan juga dikembalikan. Gunakan `_include` untuk menyertakan sumber daya yang ditautkan ke depan.

Example— _include Untuk digunakan untuk menemukan pasien atau kelompok pasien yang telah didiagnosis dengan batuk

Anda akan mencari pada jenis `Condition` sumber daya yang menentukan kode diagnostik untuk batuk, dan kemudian menggunakan `_include` tentukan bahwa Anda ingin diagnosis itu

dikembalikan juga. `subject` Dalam tipe `Condition` sumber daya `subject` mengacu pada jenis sumber daya pasien atau tipe sumber daya grup.

Untuk kejelasan, karakter khusus dalam contoh belum dikodekan. Untuk membuat kueri yang berhasil, pastikan bahwa string kueri telah dikodekan dengan benar.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
Condition?code=49727002&_include=Condition:subject
```

`_include:iterate`Pengubah

`_include:iterate`Pengubah memungkinkan penyertaan rekursif sumber daya yang direferensikan di dua tingkatan. Misalnya,

```
GET /ServiceRequest?
identifier=025C0931195&_include=ServiceRequest:requester&_include:iterate=PractitionerRole:prac
```

akan mengembalikan ServiceRequest, yang terkait PractitionerRole (melalui referensi pemohon), dan kemudian secara rekursif menyertakan Praktisi yang direferensikan oleh itu. PractitionerRole Pengubah ini tersedia untuk semua jenis sumber daya di HealthLake.

`_include=*`Pengubah

`_include=*`Pengubah adalah wildcard yang secara otomatis menyertakan semua sumber daya yang direferensikan langsung oleh hasil pencarian. Misalnya,

```
GET /ServiceRequest?specimen.accession=12345&_include=*
```

akan mengembalikan pencocokan ServiceRequest bersama dengan semua sumber daya yang dirujukannya (seperti Pasien, Praktisi, Spesimen, dll.) Tanpa perlu menentukan setiap jalur referensi secara individual. Pengubah ini tersedia untuk semua jenis sumber daya di HealthLake.

`_revinclude`

Menggunakan `_revinclude` dalam kueri penelusuran memungkinkan sumber daya FHIR tambahan yang ditentukan juga dikembalikan. Gunakan `_revinclude` untuk menyertakan sumber daya yang ditautkan ke belakang.

Example— Untuk digunakan _revinclude untuk memasukkan jenis sumber daya Pertemuan dan Pengamatan terkait yang terkait dengan Pasien tertentu

Untuk melakukan pencarian ini, pertama-tama Anda akan menentukan individu `Patient` dengan menentukan pengenal mereka di parameter `_id` pencarian. Kemudian Anda akan menentukan sumber daya FHIR tambahan menggunakan struktur `Encounter:patient` dan`Observation:patient`.

Untuk kejelasan, karakter khusus dalam contoh belum dikodekan. Untuk membuat kueri yang berhasil, pastikan bahwa string kueri telah dikodekan dengan benar.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
Patient?_id=patient-
identifier&_revinclude=Encounter:patient&_revinclude=Observation:patient
```

`_summary`

Menggunakan `_summary` dalam permintaan pencarian memungkinkan pengguna untuk meminta subset dari sumber daya FHIR. Ini dapat berisi salah satu nilai berikut:`true, text, data, false`. Nilai lainnya akan diperlakukan sebagai tidak valid. Sumber daya yang dikembalikan akan ditandai dengan `'SUBSETTED'` meta.tag, untuk menunjukkan bahwa sumber daya tidak lengkap.

- `true`: Kembalikan semua elemen yang didukung yang ditandai sebagai 'ringkasan' dalam definisi dasar sumber daya.
- `text`: Kembalikan hanya elemen 'teks', 'id', 'meta', dan hanya elemen wajib tingkat atas.
- `data`: Kembalikan semua bagian kecuali elemen 'teks'.
- `false`: Kembalikan semua bagian sumber daya

Dalam satu permintaan pencarian, `_summary=text` tidak dapat digabungkan dengan `_include` atau parameter `_revinclude` pencarian.

Example— Dapatkan elemen “teks” dari sumber daya Pasien di penyimpanan data.

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

`_elements`

Menggunakan `_elements` dalam permintaan pencarian memungkinkan untuk elemen sumber daya FHIR tertentu yang akan diminta. Sumber daya yang dikembalikan akan ditandai dengan `'SUBSETTED'` meta.tag, untuk menunjukkan bahwa sumber daya tidak lengkap.

`_elements`Parameter terdiri dari daftar nama elemen dasar yang dipisahkan koma seperti elemen yang didefinisikan pada tingkat root dalam sumber daya. Hanya elemen yang terdaftar yang akan dikembalikan. Jika nilai `_elements` parameter mengandung elemen yang tidak valid, server akan mengabaikannya dan mengembalikan elemen wajib dan elemen yang valid.

`_elements`tidak akan berlaku untuk sumber daya yang disertakan (sumber daya yang dikembalikan yang mode pencariannya`include`).

Dalam satu permintaan pencarian, `_elements` tidak dapat digabungkan dengan parameter `_summary` pencarian.

Example— Dapatkan elemen “pengenal”, “aktif”, “tautan” dari sumber daya Pasien di penyimpanan HealthLake data Anda.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?
_elements=identifier,active,link
```

`_total`

Menggunakan `_total` dalam permintaan pencarian akan mengembalikan jumlah sumber daya yang cocok dengan parameter pencarian yang diminta. HealthLake akan mengembalikan jumlah total sumber daya yang cocok (sumber daya yang dikembalikan yang mode pencariannya`match`) dalam `Bundle.total` respons pencarian.

`_total`mendukung`accurate`, nilai none parameter. `_total=estimate`tidak didukung. Nilai lainnya akan diperlakukan sebagai tidak valid. `_total`tidak berlaku untuk sumber daya yang disertakan (sumber daya yang dikembalikan yang mode pencariannya`include`).

Example— Dapatkan jumlah total sumber daya Pasien di penyimpanan data:

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

`_sort`

Menggunakan `_sort` dalam permintaan pencarian mengatur hasil dalam urutan tertentu. Hasilnya diurutkan berdasarkan daftar aturan pengurutan yang dipisahkan koma dalam urutan prioritas. Aturan

pengurutan harus berupa parameter pencarian yang valid. Nilai lainnya akan diperlakukan sebagai tidak valid.

Dalam satu permintaan pencarian, Anda dapat menggunakan hingga 5 parameter pencarian pengurutan. Anda secara opsional dapat menggunakan - awalan untuk menunjukkan urutan menurun. Server akan mengurutkan urutan menaik secara default.

Jenis parameter pencarian sortir yang didukung adalah:`Number, String, Date, Quantity, Token, URI, Reference`. Jika parameter pencarian mengacu pada elemen yang bersarang, parameter pencarian ini tidak didukung untuk pengurutan. Misalnya, pencarian pada 'nama' dari tipe sumber daya Pasien mengacu pada elemen Patient.name dengan tipe HumanName data dianggap sebagai bersarang. Dengan demikian, urutkan sumber daya Pasien dengan 'nama' tidak didukung.

Example— Dapatkan sumber daya Pasien di penyimpanan data dan urutkan berdasarkan tanggal lahir dalam urutan menaik:

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

`_count`

Parameter `_count` didefinisikan sebagai instruksi ke server mengenai berapa banyak sumber daya yang harus dikembalikan dalam satu halaman.

Ukuran halaman maksimum adalah 100. Nilai apa pun yang lebih besar dari 100 tidak valid. `_count=0`tidak didukung.

Example— Cari sumber daya Pasien dan atur ukuran halaman pencarian ke 25:

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?_count=25
```

Chaining and Reverse Chaining(_has)

Chaining dan reverse chaining di FHIR memberikan cara yang lebih efisien dan ringkas untuk mendapatkan data yang saling berhubungan, mengurangi kebutuhan akan beberapa kueri terpisah dan membuat pengambilan data lebih nyaman bagi pengembang dan pengguna.

Jika ada tingkat rekursi yang mengembalikan lebih dari 100 hasil, HealthLake akan mengembalikan 4xx untuk melindungi penyimpanan data agar tidak kelebihan beban dan menyebabkan beberapa paginasi.

Example— Chaining - Mendapat semua DiagnosticReport yang merujuk pada Pasien di mana nama Pasien adalah peter.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/DiagnosticReport?
subject:Patient.name=peter
```

Example— Reverse Chaining - Dapatkan sumber daya Pasien, di mana sumber daya pasien dirujuk oleh setidaknya satu Observasi di mana pengamatan memiliki kode 1234, dan di mana Observasi mengacu pada sumber daya pasien dalam parameter pencarian pasien.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?
_has:Observation:patient:code=1234
```

Pengubah pencarian yang didukung

Pengubah pencarian digunakan dengan bidang berbasis string. Semua pengubah pencarian HealthLake menggunakan logika berbasis Boolean. Misalnya, Anda dapat menentukan `:contains` untuk menentukan bahwa bidang string yang lebih besar harus menyertakan string kecil agar dapat disertakan dalam hasil pencarian Anda.

Pengubah pencarian yang didukung

Pengubah pencarian	Tipe
:hilang	Semua parameter kecuali `Composite`
:tepat	String
:berisi	String
:tidak	Token
:teks	Token
:pengidentifikasi	Referensi
:di bawah	URI

Komparator pencarian yang didukung

Anda dapat menggunakan pembanding pencarian untuk mengontrol sifat pencocokan dalam pencarian. Anda dapat menggunakan komparator saat mencari bidang angka, tanggal, dan kuantitas. Tabel berikut mencantumkan pembanding pencarian dan definisinya yang didukung oleh HealthLake.

Komparator pencarian yang didukung

Cari pembanding	Deskripsi
persamaan	Nilai untuk parameter dalam sumber daya sama dengan nilai yang diberikan.
ne	Nilai untuk parameter dalam sumber daya tidak sama dengan nilai yang diberikan.
gt	Nilai untuk parameter dalam sumber daya lebih besar dari nilai yang diberikan.
lt	Nilai untuk parameter dalam sumber daya kurang dari nilai yang diberikan.
ge	Nilai untuk parameter dalam sumber daya lebih besar atau sama dengan nilai yang diberikan.
le	Nilai untuk parameter dalam sumber daya kurang atau sama dengan nilai yang diberikan.
sa	Nilai untuk parameter dalam sumber daya dimulai setelah nilai yang diberikan.
eb	Nilai untuk parameter dalam sumber daya berakhir sebelum nilai yang diberikan.

Parameter pencarian FHIR tidak didukung oleh HealthLake

HealthLake mendukung semua parameter pencarian FHIR dengan pengecualian yang tercantum dalam tabel berikut. Untuk daftar lengkap parameter pencarian FHIR, lihat registri parameter pencarian FHIR.

Parameter pencarian yang tidak didukung

Komposisi bundel	Lokasi-dekat
Pengenal bundel	C onsent-source-reference
Pesan bundel	Kontrak-pasien
Jenis bundel	Konten sumber daya
Bundle-stempel waktu	Permintaan sumber daya

FHIR **`$operations`** R4 untuk HealthLake

Operasi FHIR $ (juga disebut “operasi dolar”) adalah fungsi sisi server khusus yang melampaui operasi CRUD (`Create`,, `ReadUpdate`,`Delete`) standar dalam spesifikasi FHIR. Operasi ini diidentifikasi dengan awalan “$” dan memungkinkan pemrosesan kompleks, transformasi data, dan operasi massal yang akan sulit atau tidak efisien untuk dilakukan menggunakan panggilan REST API standar. $ Operasi dapat dipanggil pada tingkat sistem, tingkat jenis sumber daya, atau pada contoh sumber daya tertentu, menyediakan cara yang fleksibel untuk berinteraksi dengan data FHIR. AWS HealthLake mendukung beberapa FHIR`$operations`. Silakan merujuk ke setiap halaman individu di bawah ini untuk detail tambahan.

Topik

- Operasi FHIR R4 $attribution-status untuk HealthLake
- Menghapus Jenis Sumber Daya dengan $bulk-delete
- Operasi $bulk-member-match untuk HealthLake
- Operasi FHIR R4 $confirm-attribution-list untuk HealthLake
- Operasi FHIR R4 $davinci-data-export untuk HealthLake
- Menghasilkan Dokumen Klinis dengan $document
- Menghapus Sumber Daya Secara Permanen dengan $erase
- Mendapatkan data pasien dengan Patient/$everything
- Mengambil ValueSet Kode dengan $expand
- Mengekspor HealthLake data dengan FHIR $export
- $inquireOperasi FHIR untuk HealthLake

- Mengambil Detail Konsep dengan $lookup
- $member-addoperasi untuk HealthLake
- $member-matchoperasi untuk HealthLake
- $member-removeoperasi untuk HealthLake
- Menghapus Sumber Daya Kompartemen Pasien dengan $purge
- $questionnaire-packageOperasi FHIR untuk HealthLake
- $submitOperasi FHIR untuk HealthLake
- Memvalidasi Sumber Daya FHIR dengan $validate

Operasi FHIR R4 **`$attribution-status`** untuk HealthLake

Mengambil status atribusi untuk anggota tertentu, mengembalikan Bundel yang berisi semua sumber atribusi yang terkait dengan Pasien. Operasi ini merupakan bagian dari implementasi Daftar Atribusi Anggota FHIR (ATR) IG 2.1.0.

Titik akhir

```
POST [base]/Group/[id]/$attribution-status
```

Parameter Permintaan

Operasi menerima parameter opsional berikut:

Parameter	Jenis	Deskripsi
MemberID	Pengidentifikasi	MemberId Anggota yang meminta status atribusi
PatientReferensi	Referensi	Referensi ke sumber daya pasien dalam sistem Produsen

> **Note**
>
> Baik `memberId` atau `patientReference` dapat disediakan, atau keduanya untuk tujuan validasi.

Permintaan Sampel

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "memberId",
      "valueIdentifier": {
        "system": "http://example.org",
        "value": "MBR123456789"
      }
    },
    {
      "name": "patientReference",
      "valueReference": {
        "reference": "Patient/patient-123",
        "display": "John Doe"
      }
    }
  ]
}
```

Respons

Mengembalikan Bundel yang berisi sumber daya atribusi yang terkait dengan Pasien:

Sumber daya	Kardinalitas	Lokasi
Pasien	1.. 1	Group.member.entity
Cakupan	0.. 1	Group.member.extension:CoverageReference
Organization/Practitioner/PractitionerRole	0.. 1	group.member.extension:AttributedProvider
Sumber Daya Apa Pun	0.. 1	Group.member.extension:AssociatedData

Contoh Respons

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

Penanganan Kesalahan

Operasi menangani kondisi kesalahan berikut:

Kesalahan	Status HTTP	Deskripsi
Permintaan operasi tidak valid	400	Parameter atau struktur permintaan yang tidak sesuai
Sumber daya grup tidak ditemukan	404	ID Grup yang ditentukan tidak ada
Sumber daya pasien tidak ditemukan	404	Referensi Pasien yang ditentukan tidak ada

Otorisasi dan Keamanan

Persyaratan Lingkup SMART

Klien harus memiliki hak istimewa yang sesuai untuk membaca sumber daya Grup dan sumber daya atribusi terkait

Mekanisme otorisasi FHIR standar berlaku untuk semua operasi

Menghapus Jenis Sumber Daya dengan **`$bulk-delete`**

AWS HealthLake mendukung `$bulk-delete` operasi, memungkinkan penghapusan semua sumber daya dari jenis tertentu dalam datastore. Operasi ini sangat berguna ketika Anda perlu:

- Lakukan audit dan pembersihan musiman
- Mengelola siklus hidup data dalam skala
- Hapus jenis sumber daya tertentu
- Mematuhi kebijakan penyimpanan data

Penggunaan

`$bulk-delete`Operasi dapat dipanggil menggunakan metode POST:

```
POST [base]/[ResourceType]/$bulk-delete?isHardDelete=false&deleteAuditEvent=true
```

Parameter

Parameter	Tipe	Diperlukan	Default	Deskripsi
`isHardDelete`	boolean	Tidak	false	Ketika benar, secara permanen menghapus sumber daya dari penyimpanan
`deleteAuditEvent`	boolean	Tidak	true	Jika benar, menghapus peristiwa audit terkait
`_since`	string	Tidak	Waktu pembuatan Datastore	Saat dimasukkan, pilih waktu cutoff awal untuk menemukan sumber daya berdasarkan waktu LastModified mereka. Tidak dapat digunakan dengan awal atau akhir
`start`	string	Tidak	Waktu pembuatan Datastore	Saat dimasukkan, pilih waktu cutoff untuk menemukan sumber daya berdasarkan waktu LastModified mereka. Dapat digunakan dengan akhir
`end`	string	Tidak	Waktu pengajuan Job	Saat dimasukkan, pilih waktu cutoff akhir untuk menemukan sumber daya berdasarkan waktu LastModified mereka

Contoh

Contoh Permintaan

```
POST [base]/Observation/$bulk-delete?isHardDelete=false
```

Contoh Respons

```
{
      "jobId": "jobId",
      "jobStatus": "SUBMITTED"
    }
```

Status Tugas

Untuk memeriksa status pekerjaan penghapusan massal:

```
GET [base]/$bulk-delete/[jobId]
```

Operasi mengembalikan informasi status pekerjaan:

```
{
      "datastoreId": "datastoreId",
      "jobId": "jobId",
      "status": "COMPLETED",
      "submittedTime": "2025-10-09T15:09:51.336Z"
    }
```

Perilaku

`$bulk-delete`Operasi:

1. Memproses secara asinkron untuk menangani volume sumber daya yang besar
2. Menjaga transaksi ACID untuk integritas data
3. Menyediakan pelacakan status pekerjaan dengan jumlah penghapusan sumber daya
4. Mendukung mode penghapusan lunak dan keras
5. Termasuk pencatatan audit yang komprehensif atas kegiatan penghapusan
6. Memungkinkan penghapusan selektif versi historis dan peristiwa audit

Pencatatan Audit

Log `$bulk-delete` operasi sebagai Mulai FHIRBulk DeleteJob dan Jelaskan FHIRBulk DeleteJob dengan informasi operasi terperinci.

Batasan

- Bila `isHardDelete` disetel ke true, sumber daya yang dihapus tidak akan muncul di hasil penelusuran atau `_history` kueri.
- Sumber daya yang dihapus melalui operasi ini mungkin sementara tidak dapat diakses selama pemrosesan
- Pengukuran penyimpanan disesuaikan hanya pada versi historis - deleteVersionHistory =false tidak akan menyesuaikan penyimpanan datastore

`Operasi $bulk-member-match` untuk HealthLake

AWS HealthLake mendukung `$bulk-member-match` operasi untuk memproses beberapa permintaan kecocokan anggota secara asinkron. Operasi ini memungkinkan organisasi perawatan kesehatan untuk secara efisien mencocokkan ratusan pengidentifikasi unik anggota di berbagai sistem perawatan kesehatan menggunakan informasi demografis dan cakupan dalam satu permintaan massal. Kemampuan ini sangat penting untuk pertukaran data pembayar-ke-pembayar skala besar, transisi anggota, dan persyaratan kepatuhan CMS dan mengikuti spesifikasi FHIR.

> **Note**
>
> `$bulk-member-match`Operasi ini didasarkan pada spesifikasi FHIR yang mendasari yang saat ini eksperimental dan dapat berubah. Seiring berkembangnya spesifikasi, perilaku dan antarmuka API ini akan diperbarui sesuai dengan itu. Pengembang disarankan untuk memantau catatan HealthLake rilis AWS dan pembaruan spesifikasi FHIR yang relevan agar tetap mendapat informasi tentang perubahan apa pun yang dapat memengaruhi integrasi mereka.

Operasi ini sangat berguna ketika Anda perlu:

- Memproses pencocokan anggota pada skala selama periode pendaftaran terbuka
- Memfasilitasi transisi anggota massal antar pembayar
- Mendukung persyaratan pertukaran data kepatuhan CMS skala besar
- Secara efisien mencocokkan kelompok anggota di seluruh jaringan perawatan kesehatan
- Minimalkan panggilan API dan tingkatkan efisiensi operasional untuk skenario pencocokan volume tinggi

Penggunaan

`$bulk-member-match`Operasi ini adalah operasi asinkron yang dipanggil pada sumber daya Grup menggunakan metode POST:

```
POST [base]/Group/$bulk-member-match
```

Setelah mengirimkan permintaan kecocokan massal, Anda dapat melakukan polling status pekerjaan menggunakan:

```
GET [base]/$bulk-member-match-status/{jobId}
```

Parameter yang didukung

HealthLake mendukung `$bulk-member-match` parameter FHIR berikut:

Parameter	Tipe	Diperlukan	Deskripsi
`MemberPatient`	Pasien	Ya	Sumber daya pasien yang berisi informasi demografis agar anggota dicocokkan.
`CoverageToMatch`	Cakupan	Ya	Sumber daya cakupan yang akan digunakan untuk pencocokan dengan catatan yang ada.
`CoverageToLink`	Cakupan	Tidak	Sumber daya cakupan yang akan ditautkan selama proses pencocokan.
`Consent`	Persetujuan	Ya	Sumber daya persetujuan untuk tujuan otorisasi juga disimpan. Ini berbeda dengan `$member-match` operasi individu di mana Persetujuan tidak diperlukan.

Permintaan POST untuk mengirimkan pekerjaan kecocokan anggota massal

Contoh berikut menunjukkan permintaan POST untuk mengirimkan pekerjaan kecocokan anggota massal. Setiap anggota dibungkus dalam `MemberBundle` parameter yang berisi yang diperlukan`MemberPatient`,`CoverageToMatch`, dan `Consent` sumber daya, bersama dengan opsional`CoverageToLink`.

```
POST [base]/Group/$bulk-member-match
Content-Type: application/fhir+json
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "MemberBundle",
      "part": [
        {
          "name": "MemberPatient",
          "resource": {
            "resourceType": "Patient",
            "identifier": [
              {
                "system": "http://example.org/patient-id",
                "value": "patient-0"
              }
            ],
            "name": [
              {
                "family": "Smith",
                "given": ["James"]
              }
            ],
            "gender": "male",
            "birthDate": "1950-01-01"
          }
        },
        {
          "name": "CoverageToMatch",
          "resource": {
            "resourceType": "Coverage",
            "status": "active",
            "identifier": [
              {
                "system": "http://example.org/coverage-id",
                "value": "cov-0"
              }
            ],
            "subscriberId": "sub-0",
            "beneficiary": {
              "reference": "Patient/patient123"
            },
```

```
            "relationship": {
              "coding": [
                {
                  "system": "http://terminology.hl7.org/CodeSystem/subscriber-
relationship",
                  "code": "self"
                }
              ]
            },
            "payor": [
              {
                "reference": "Organization/org123"
              }
            ]
          }
        },
        {
          "name": "Consent",
          "resource": {
            "resourceType": "Consent",
            "status": "active",
            "scope": {
              "coding": [
                {
                  "system": "http://terminology.hl7.org/CodeSystem/consentscope",
                  "code": "patient-privacy"
                }
              ]
            },
            "category": [
              {
                "coding": [
                  {
                    "system": "http://terminology.hl7.org/CodeSystem/v3-ActCode",
                    "code": "IDSCL"
                  }
                ]
              }
            ],
            "patient": {
              "reference": "Patient/patient123"
            },
            "performer": [
              {
```

```
                "reference": "Patient/patient123"
              }
            ],
            "sourceReference": {
              "reference": "http://example.org/DocumentReference/consent-source"
            },
            "policy": [
              {
                "uri": "http://hl7.org/fhir/us/davinci-hrex/StructureDefinition-hrex-
consent.html#regular"
              }
            ],
            "provision": {
              "type": "permit",
              "period": {
                "start": "2024-01-01",
                "end": "2025-12-31"
              },
              "actor": [
                {
                  "role": {
                    "coding": [
                      {
                        "system": "http://terminology.hl7.org/CodeSystem/provenance-
participant-type",
                        "code": "performer"
                      }
                    ]
                  },
                  "reference": {
                    "identifier": {
                      "system": "http://hl7.org/fhir/sid/us-npi",
                      "value": "9876543210"
                    },
                    "display": "Old Health Plan"
                  }
                },
                {
                  "role": {
                    "coding": [
                      {
                        "system": "http://terminology.hl7.org/CodeSystem/v3-
ParticipationType",
                        "code": "IRCP"
```

```
                    }
                  ]
                },
                "reference": {
                  "identifier": {
                    "system": "http://hl7.org/fhir/sid/us-npi",
                    "value": "0123456789"
                  },
                  "display": "New Health Plan"
                }
              }
            ],
            "action": [
              {
                "coding": [
                  {
                    "system": "http://terminology.hl7.org/CodeSystem/consentaction",
                    "code": "disclose"
                  }
                ]
              }
            ]
          }
        }
      },
      {
        "name": "CoverageToLink",
        "resource": {
          "resourceType": "Coverage",
          "status": "active",
          "identifier": [
            {
              "system": "http://example.org/coverage-link-id",
              "value": "cov-link-0"
            }
          ],
          "subscriberId": "new-sub-0",
          "beneficiary": {
            "reference": "Patient/patient123"
          },
          "relationship": {
            "coding": [
              {
```

```
                  "system": "http://terminology.hl7.org/CodeSystem/subscriber-
relationship",
                  "code": "self"
                }
              ]
            },
            "payor": [
              {
                "identifier": {
                  "system": "http://hl7.org/fhir/sid/us-npi",
                  "value": "0123456789"
                },
                "display": "New Health Plan"
              }
            ]
          }
        }
      ]
    }
  ]
}
```

Menyelesaikan respon pekerjaan dengan output

Ketika pekerjaan selesai, respons mencakup metadata pekerjaan dan sumber daya Parameter FHIR yang berisi tiga sumber daya Grup yang mengkategorikan hasil pencocokan.

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

Bagaimana HealthLake mengklasifikasikan anggota ke dalam Grup keluaran

Setiap anggota yang diajukan dalam `$bulk-member-match` permintaan dievaluasi melalui pipeline berurutan. Hasil dari setiap langkah menentukan keluaran Grup mana anggota ditempatkan.

1. Validasi struktural — Apakah MemberBundle sesuai dengan profil yang diperlukan? Pada kegagalan: Kesalahan (tidak di Grup mana pun).
2. Pencocokan pasien — Dapatkah HealthLake menemukan pasien yang cocok dengan demografi yang dikirimkan? Pada kegagalan: NonMatchedMembers.
3. Konfirmasi cakupan — Dapat HealthLake mempersempit tepat satu pasien dengan valid CoverageToMatch? Pada kegagalan: NonMatchedMembers.
4. Evaluasi persetujuan — Apakah Persetujuan yang diajukan terhormat saat ini? (status = aktif, periode mencakup tanggal saat ini, pemain dapat divalidasi). Pada kegagalan: ConsentConstrainedMembers.
5. Sukses — Semua cek lulus. Persetujuan disimpan di datastore. Anggota ditempatkan di MatchedMembers.

Prinsip utama: Anggota hanya dapat muncul di satu tujuan. Langkah gagal pertama menentukan penempatan. Anggota yang gagal Langkah 2 atau 3 tidak pernah ditempatkan di ConsentConstrainedMembers - Grup tersebut khusus untuk anggota yang berhasil dicocokkan tetapi yang persetujuannya tidak dapat dihormati.

Rincian evaluasi persetujuan (Langkah 4):

- Periksa 1 — Status persetujuan: Apakah `Consent.status` sama dengan “aktif”? Jika tidak → ConsentConstrainedMembers.
- Cek 2 — Periode ketentuan: Apakah `provision.period` mencakup tanggal saat ini? Jika tanggal saat ini sebelum `period.start` atau sesudah `period.end` → ConsentConstrainedMembers.
- Periksa 3 - Validasi pemain: Dapatkah **`Consent.performer`** referensi divalidasi? Jika sumber daya yang direferensikan tidak ditemukan di datastore atau tidak terkait dengan pasien yang cocok →. ConsentConstrainedMembers

Semua cek harus lolos agar anggota ditempatkan MatchedMembers dan agar Persetujuan disimpan.

Perilaku pencocokan cakupan

Selama pencocokan anggota, hanya `CoverageToMatch` divalidasi terhadap datastore pembayar yang merespons. `CoverageToLink`milik new/requesting pembayar dan tidak divalidasi terhadap datastore pembayar lama. Termasuk `CoverageToLink` dalam permintaan tidak akan mempengaruhi hasil pencocokan.

Setiap kombinasi Pasien+Cakupan dalam permintaan diproses secara independen. Pasien yang sama dapat diajukan beberapa kali dengan rencana pertanggungan yang berbeda, dan setiap entri menerima hasilnya sendiri berdasarkan cakupan spesifiknya.

Persetujuan penanganan referensi pemain

Pembayar baru dapat mengirim referensi pasien sementara atau lokal di `Consent.performer` (misalnya, referensi yang sama digunakan dalam`Consent.patient`). HealthLake menyelesaikan referensi ini secara otomatis:

- Jika `Consent.performer` berisi referensi lokal yang sama seperti`Consent.patient`, HealthLake gantilah dengan referensi pasien yang cocok aktual setelah pencocokan berhasil.
- HealthLake mendukung referensi pemain tipe Pasien, RelatedPerson, Praktisi PractitionerRole, dan Organisasi (referensi langsung dan referensi pengidentifikasi logis).
- Jika validasi pemain gagal (sumber daya tidak ditemukan atau tidak terkait dengan pasien yang cocok), anggota ditempatkan di ConsentConstrainedMembers alih-alih mengembalikan kesalahan.

Sumber daya Grup Keluaran

Pekerjaan yang diselesaikan mengembalikan sumber daya Parameter yang berisi tiga sumber daya Grup:

MatchedMembers Kelompok

Berisi referensi Pasien untuk semua anggota yang berhasil dicocokkan yang persetujuannya aktif dan valid pada saat permintaan. Sumber daya Persetujuan dibuat dan disimpan di datastore untuk setiap anggota yang cocok. Grup ini dipakai di datastore dan dapat digunakan langsung dengan. `$davinci-data-export`

NonMatchedMembers Kelompok

Berisi referensi ke anggota di mana tidak ada kecocokan unik yang ditemukan. Anggota ditempatkan di sini ketika tidak ada pasien di datastore yang cocok dengan demografi yang disediakan, tidak ada cakupan yang valid untuk kandidat pasien yang cocok, atau beberapa pasien yang cocok dengan demografi dan beberapa memiliki cakupan yang valid (ambigu).

ConsentConstrainedMembers Kelompok

Berisi referensi Pasien untuk anggota yang berhasil dicocokkan (demografi dan cakupan dikonfirmasi) tetapi persetujuannya tidak dapat dihormati pada saat permintaan. Sumber daya Persetujuan tidak disimpan untuk anggota yang dibatasi persetujuan. Identitas anggota yang cocok (MemberIdentifier dan MemberId) masih disertakan sehingga pembayar yang meminta tahu siapa yang dibatasi.

`Group.quantity`Bidang berisi jumlah total anggota di masing-masing kelompok masing-masing.

Referensi anggota grup:

- `Group.member.entity.reference`— Untuk MatchedMembers dan ConsentConstrainedMembers, berisi ID Pasien dari anggota yang cocok dalam sistem pembayar yang merespons. Untuk NonMatchedMembers, referensi yang terkandung masukan Pasien.
- `Group.member.entity.extension (base-ext-match-parameters)`— Berisi ID Pasien dari permintaan input asli (ID yang dikirimkan oleh pembayar yang meminta, berasal dari, `Patient.idCoverage.beneficiary.reference`, atau`Consent.patient.reference`).

Consent-Patient keterkaitan

> **Important**
>
> Sumber daya Persetujuan yang disimpan mempertahankan referensi pasien persis seperti yang dikirimkan oleh pembayar yang meminta. HealthLake tidak secara otomatis memperbarui bidang pasien Persetujuan untuk menunjuk ke Pasien yang cocok di datastore penerima.

Untuk menautkan Persetujuan yang disimpan ke Pasien yang cocok, gunakan output pekerjaan: setiap anggota dalam MatchedMembers Grup memiliki `member.entity.reference` penunjuk ke Pasien yang cocok dan `member.entity.extension` (`base-ext-match-parameters`) menunjuk ke masukan yang terkandung Pasien. Cross-reference ini dengan bidang pasien Persetujuan untuk membangun pemetaan di lapisan aplikasi Anda.

Apa yang disimpan vs. apa yang sementara

Tabel berikut mendokumentasikan apa yang HealthLake bertahan pada datastore selama `$bulk-member-match` pemrosesan dan apa yang hanya ada dalam respons pekerjaan:

Sumber daya	Disimpan?	Dapat ditanyai melalui REST?	Catatan
MemberPatient (masukan)	Tidak	Tidak	Digunakan hanya untuk pencocokan; tidak bertahan
CoverageToMatch (masukan)	Tidak	Tidak	Digunakan hanya untuk konfirmasi pertanggungan
CoverageToLink (masukan)	Tidak	Tidak	Tidak divalidasi terhadap datastore; milik pembayar baru
Persetujuan (anggota yang cocok)	Ya	Ya — DAPATKAN [dasar] /	Disimpan sebagai-diterima dari pembayar yang meminta

Sumber daya	Disimpan?	Dapat ditanyai melalui REST?	Catatan
		Persetujuan/ {id}	
Persetujuan (anggota terbatas)	Tidak	Tidak	Tidak disimpan. Identitas anggota masih termasuk dalam tanggapan.
MatchedMembers Grup (keluaran)	Ya	Ya — GET [base] / Group/ {id}	Instantiated; dapat digunakan dengan $davinci-data-export
NonMatchedMembers Kelompok	Tidak	Tidak	Respon Job saja
ConsentConstrained Members Kelompok	Tidak	Tidak	Respon Job saja

Integrasi dengan $davinci-data-export

Sumber daya MatchedMembers Grup yang dikembalikan oleh `$bulk-member-match` dapat langsung digunakan dengan `$davinci-data-export` operasi untuk mengambil data anggota massal:

```
POST [base]/Group/{matched-group-id}/$davinci-data-export
GET [base]/Group/{matched-group-id}
```

Integrasi ini memungkinkan alur kerja yang efisien di mana Anda pertama kali mengidentifikasi anggota yang cocok secara massal, kemudian mengekspor catatan kesehatan lengkap mereka menggunakan sumber daya Grup yang dihasilkan.

Menggunakan $member-remove sebelum ekspor

Jika Anda perlu mengecualikan anggota tertentu dari ekspor setelah pencocokan (misalnya, anggota mencabut persetujuan antara pencocokan dan ekspor), gunakan `$member-remove` di MatchedMembers Grup.

> **Important**
>
> Menghapus anggota melalui `$member-remove` menandai anggota sebagai tidak aktif dalam Grup, tetapi `$davinci-data-export` hanya mengecualikan anggota yang tidak aktif setelah Grup diperbarui ke status "final". Jika Anda memanggil `$davinci-data-export` Grup yang masih memiliki status default, anggota yang dihapus mungkin masih muncul di hasil ekspor.

Alur kerja:

1. `POST [base]/Group/{id}/$member-remove`— tandai anggota tidak aktif
2. `PUT [base]/Group/{id}`— perbarui status Grup ke "final"
3. `POST [base]/Group/{id}/$davinci-data-export`— ekspor sekarang tidak termasuk anggota yang dihapus

Karakteristik kinerja

`$bulk-member-match`Operasi ini dirancang untuk pemrosesan volume tinggi dan berjalan secara asinkron:

- Konkurensi: Maksimum 5 operasi bersamaan per penyimpanan data.
- Skalabilitas: Menangani hingga 500 anggota per permintaan (batas muatan 5 MB).
- Operasi paralel: Kompatibel dengan operasi impor, ekspor, atau penghapusan massal bersamaan.

Otorisasi

API menggunakan SMART pada protokol otorisasi FHIR dengan cakupan wajib berikut:

- `system/Patient.read`— Diperlukan untuk mencari dan mencocokkan sumber daya pasien.
- `system/Coverage.read`— Diperlukan untuk memvalidasi informasi cakupan.
- `system/Group.write`— Diperlukan untuk membuat sumber daya Grup hasil.
- `system/Organization.read`— Bersyarat, diperlukan jika cakupan referensi organisasi.
- `system/Practitioner.read`— Bersyarat, diperlukan jika cakupan referensi praktisi.
- `system/PractitionerRole.read`— Bersyarat, diperlukan jika cakupan referensi peran praktisi.

- `system/Consent.write`— Bersyarat, diperlukan jika sumber daya persetujuan disediakan.

Operasi ini juga mendukung otorisasi AWS IAM Signature Version 4 (SiGv4) untuk akses terprogram.

Aturan validasi

Aturan validasi berikut diterapkan untuk masing-masing MemberBundle pada Langkah 1. Anggota yang gagal validasi dilaporkan sebagai kesalahan dan tidak muncul di Grup keluaran apa pun.

MemberPatient

Bidang	Bagaimana HealthLake menggunakannya	Kegagalan validasi jika...
`name.family`	Pencarian demografis	Hilang
`name.given`	Pencarian demografis	Hilang (setidaknya satu diperlukan)
`birthDate`	Pencarian demografis	Hilang
`gender`	Pencarian demografis; jika tidak ada, ekstensi Jenis Kelamin Kelahiran digunakan	Baik jenis kelamin maupun Jenis Kelahiran Seks hadir (hrex-pat-1)
`identifier`	Termasuk dalam pencarian saat hadir; meningkatkan kepercayaan diri	Tidak pernah menyebabkan kegagalan (opsional)

CoverageToMatch / CoverageToLink

Bidang	Bagaimana HealthLake menggunakannya	Kegagalan validasi jika...
`status`	Mengonfirmasi cakupan dapat ditindaklanjuti	Hilang

Bidang	Bagaimana HealthLake menggunakannya	Kegagalan validasi jika...
`beneficiary`	Menautkan cakupan ke kandidat pasien	Hilang
`payor`	Disambiguasi ketika ada banyak kandidat	Hilang, atau lebih dari satu pembayar
`relationship`	Mengonfirmasi hubungan pelanggan-penerima	Hilang
`identifier (MB) or subscriberId`	Kunci disambiguasi primer	Tidak ada yang hadir

Persetujuan

Bidang	Bagaimana HealthLake menggunakannya	Kegagalan validasi jika...
`scope`	Mengonfirmasi ruang lingkup persetujuan adalah privasi pasien	Hilang atau tidak ada kode privasi pasien
`category`	Mengonfirmasi klasifikasi pengungkapan	Hilang
`patient`	Mengidentifikasi subjek persetuju an	Hilang
`performer`	Mengidentifikasi siapa yang setuju	Hilang
`sourceReference`	Dokumentasikan sumber persetuju an	Hilang
`policy.uri`	Menentukan ruang lingkup berbagi data	Hilang atau URI tidak berakhir di #regular atau #sensitive

Bidang	Bagaimana HealthLake menggunakannya	Kegagalan validasi jika...
`provision.type`	Harus “izin” per profil Persetujuan HRex	Hilang atau tidak “izin” (termasuk “tolak”)
`provision.period`	Dievaluasi pada Langkah 4 untuk pemeriksaan terbatas persetujuan	Hilang atau tidak start/end
`status`	Dievaluasi pada Langkah 4 (BUKAN Langkah 1)	Jangan pernah menyebabkan kegagalan Langkah 1 - HealthLake menerima status yang valid dan mengevaluasi pada Langkah 4

> **Note**
>
> Profil Persetujuan HRex mendefinisikan status dengan nilai tetap “aktif”. HealthLake sengaja melonggarkan kendala ini sehingga Persetujuan non-aktif menerima klasifikasi yang bermakna (ConsentConstrainedMembers) daripada penolakan validasi menyeluruh.

Perilaku pencocokan

- Pencarian pasien (Langkah 2) - HealthLake pencarian menggunakan `name.family` + `name.given` (tepat, tidak peka huruf besar/kecil), `birthDate` (tepat), `gender` (tepat; Jenis Kelamin Lahir digunakan jika jenis kelamin tidak ada), dan `identifier` (bila ada, opsional).
- Disambiguasi cakupan (Langkah 3) — Ketika beberapa kandidat pasien ditemukan, `CoverageToMatch` digunakan untuk mempersempit menjadi satu. Cakupan “valid” ketika sumber daya Cakupan aktif ada di datastore yang cocok pada `subscriberId` atau `identifier` (tipe MB) DAN. `payor`
- Evaluasi persetujuan (Langkah 4) — Hanya dilakukan setelah pertandingan unik yang sukses. Lihat bagian detail evaluasi persetujuan di atas.

Penanganan kesalahan

Operasi menangani kondisi kesalahan berikut:

- 400 Permintaan Buruk: Format permintaan tidak valid, parameter yang diperlukan tidak ada, atau muatan melebihi batas ukuran (500 anggota atau 5 MB).
- 422 Entitas yang Tidak Dapat Diproses: Memproses kesalahan selama eksekusi pekerjaan.
- Kesalahan anggota individu: Ketika anggota tertentu gagal untuk memproses, operasi berlanjut dengan anggota yang tersisa dan mencakup rincian kesalahan dalam NonMatchedMembers Grup dengan kode alasan yang sesuai. Misalnya, `MemberBundle` dengan Pasien yang hilang `birthDate` parameter akan mengembalikan kesalahan berikut:

```
"errors": [
  {
    "memberIndex": 1,
    "jsonBlob": {
      "resourceType": "OperationOutcome",
      "issue": [
        {
          "severity": "error",
          "code": "invalid",
          "diagnostics": "MemberPatient.birthDate is required"
        }
      ],
      "statusCode": 400
    }
  }
]
```

Detail kesalahan tersedia melalui titik akhir pemungutan suara status dan termasuk:

- `numberOfMembersWithCustomerError`: Hitungan anggota dengan validasi atau kesalahan input.
- `numberOfMembersWithServerError`: Hitungan anggota dengan kesalahan pemrosesan sisi server.

Operasi terkait

- the section called “$ pertandingan-anggota”— Operasi pencocokan anggota individu.
- the section called “$davinci-data-export”— Ekspor data massal menggunakan sumber daya Grup.
- the section called “$ Operasi”— Daftar lengkap operasi yang didukung.

Operasi FHIR R4 **$confirm-attribution-list** untuk HealthLake

Menunjukkan kepada Produser bahwa Konsumen tidak memiliki perubahan lagi untuk dilakukan pada Daftar Atribusi. Operasi ini menyelesaikan daftar atribusi dengan menghapus anggota yang tidak aktif dari daftar, mengubah status menjadi “final”, dan mengakui operasi. Operasi ini merupakan bagian dari implementasi Daftar Atribusi Anggota FHIR (ATR) IG 2.1.0.

Titik akhir

```
POST [base]/Group/[id]/$confirm-attribution-list
```

Permintaan

Tidak ada parameter yang diperlukan.

```
POST [base]/Group/[id]/$confirm-attribution-list
```

Respons

Mengembalikan HTTP 201 dengan pesan konfirmasi.

Contoh Respons

```
HTTP Status Code: 201

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "message",
      "valueString": "Accepted."
    }
  ]
}
```

Status Grup Setelah Konfirmasi

Setelah konfirmasi berhasil, sumber daya Grup akan memiliki status daftar atribusi disetel ke “final”:

```
{
  "resourceType": "Group",
```

```
  "id": "fullexample",
  "extension": [
    {
      "url": "http://hl7.org/fhir/us/davinci-atr/StructureDefinition/ext-
attributionListStatus",
      "valueCode": "final"
    }
  ]
  // ... other Group properties
}
```

Penanganan Kesalahan

Operasi menangani kondisi kesalahan berikut:

Kesalahan	Status HTTP	Deskripsi
Permintaan operasi tidak valid	400	Parameter atau struktur permintaan yang tidak sesuai
Sumber daya grup tidak ditemukan	404	ID Grup yang ditentukan tidak ada

Otorisasi dan Keamanan

Persyaratan Lingkup SMART

- Klien harus memiliki hak istimewa yang sesuai untuk membaca sumber daya Grup dan sumber daya atribusi terkait
- Untuk`$confirm-attribution-list`, klien juga harus memiliki hak menulis untuk memodifikasi sumber daya Grup
- Mekanisme otorisasi FHIR standar berlaku untuk semua operasi

Operasi FHIR R4 **`$davinci-data-export`** untuk HealthLake

`$davinci-data-export`Operasi ini adalah operasi FHIR asinkron yang dapat Anda gunakan untuk mengekspor data perawatan kesehatan. AWS HealthLake Operasi ini mendukung beberapa jenis ekspor, termasuk Atribusi Anggota (ATR), Akses PDex Penyedia Payer-to-Payer, dan Akses

Anggota. APIs Ini adalah versi khusus dari `$export` operasi FHIR standar, yang dirancang untuk memenuhi persyaratan panduan DaVinci implementasi.

Fitur Utama

- Pemrosesan Asinkron: Mengikuti pola permintaan asinkron FHIR standar
- Ekspor Tingkat Grup: Mengekspor data untuk anggota dalam sumber daya Grup tertentu
- Beberapa Jenis Ekspor: Mendukung ATR (Atribusi Anggota), Akses PDex Penyedia Payer-to-Payer, dan Akses Anggota APIs
- Dukungan Profil Komprehensif: Termasuk US Core, CARIN Blue Button, dan PDex profil
- Penyaringan Fleksibel: Mendukung penyaringan oleh pasien, jenis sumber daya, dan rentang waktu
- Output NDJSON: Menyediakan data dalam format JSON yang dibatasi baris baru

Titik Akhir Operasi

```
GET [base]/Group/[id]/$davinci-data-export
POST [base]/Group/[id]/$davinci-data-export
```

Parameter Permintaan

Parameter	Kardinalitas	Deskripsi
`patient`	0.. *	Anggota tertentu yang datanya akan diekspor. Ketika dihilangkan, semua anggota dalam Grup diekspor.
`_type`	0.. 1	Daftar tipe sumber daya FHIR yang dibatasi koma untuk diekspor. Saat dihilangkan, semua jenis sumber daya yang didukung untuk jenis ekspor yang ditentukan disertakan. Untuk ekspor ATR, ini default ke 8 jenis sumber daya atribusi. Untuk PDex ekspor, ini mencakup semua jenis sumber daya atribusi ditambah jenis sumber daya klinis dan klaim dari US Core, CARIN Blue Button, dan profil. PDex

Parameter	Kardinalitas	Deskripsi
`_since`	0.. 1	Hanya sertakan sumber daya yang diperbarui setelah tanggal dan waktu ini.
`_until`	0.. 1	Hanya sertakan sumber daya yang diperbarui sebelum tanggal dan waktu ini.
`exportType`	0.. 1	Jenis ekspor yang akan dilakukan. Nilai yang valid:`hl7.fhir.us.davinci-atr` ,`hl7.fhir.us.davinci-pdex`,`hl7.fhir.us.davinci-pdex.p2p` ,`hl7.fhir.us.davinci-pdex.member` . Default: `hl7.fhir.us.davinci-atr`.
`_includeEOB2xWoFinancial`	0.. 1	Menentukan apakah akan menyertakan ExplanationOfBenefit sumber daya CARIN BB 2.x dengan data keuangan dihapus. Default: `false`.
`_security`	0.. *	Filter sumber daya yang diekspor dengan nilai `meta.security` Coding. Gunakan `system|code` format (karakter pipa harus dikodekan URL sebagai). `%7C` Ketika beberapa nilai disediakan, sumber daya harus cocok dengan semuanya (DAN semantik). Gunakan `system|` (trailing pipe, no code) untuk mencocokkan kode apa pun dari sistem tertentu.
`_tag`	0.. *	Filter sumber daya yang diekspor dengan nilai `meta.tag` Coding. Menggunakan `system|code` format yang sama dan dan semantik sebagai. `_security` Ketika keduanya `_security` dan `_tag` ditentukan, sumber daya harus cocok dengan kedua filter.

Jenis Sumber Daya yang Didukung

Jenis sumber daya yang didukung bergantung pada jenis ekspor yang Anda tentukan. Untuk ekspor ATR, jenis sumber daya berikut didukung:

- `Group`
- `Patient`
- `Coverage`
- `RelatedPerson`
- `Practitioner`
- `PractitionerRole`
- `Organization`
- `Location`

Untuk PDex ekspor (Akses Penyedia Payer-to-Payer, dan Akses Anggota), semua jenis sumber daya klinis dan klaim didukung selain jenis sebelumnya. Untuk daftar lengkap jenis sumber daya yang didukung, lihat Panduan Implementasi Inti AS (STU 6.1), Panduan Implementasi Tombol Biru CARIN, dan Panduan Implementasi Dukungan Otorisasi Sebelumnya Da Vinci.

Jenis Ekspor

`$davinci-data-export`Operasi ini mendukung jenis ekspor berikut. Anda menentukan jenis ekspor dengan menggunakan `exportType` parameter.

Jenis Ekspor	Tujuan	Lingkup Data	Batas Temporal
`hl7.fhir.us.davinci-atr`	Daftar Atribusi Anggota	Sumber daya terkait atribusi	Tidak ada
`hl7.fhir.us.davinci-pdex`	API Akses Penyedia	Data klinis dan klaim untuk pasien yang dikaitkan	5 tahun
`hl7.fhir.us.davinci-pdex.p2p`	Payer-to-Payer Pertukaran	Data anggota historis untuk transisi asuransi	5 tahun

Jenis Ekspor	Tujuan	Lingkup Data	Batas Temporal
`hl7.fhir.us.davinci-pdex.member`	API Akses Anggota	Data kesehatan anggota sendiri	5 tahun

> **Note**
>
> Untuk PDex ekspor, batas temporal 5 tahun tidak berlaku untuk jenis sumber daya ATR (`Group`,,,`Patient`,`Coverage`, `RelatedPerson` `PractitionerPractitionerRole`,`Organization`). `Location` Sumber daya ini selalu disertakan tanpa memandang usia.

ATR (hl7.fhir.us.davinci-atr)

Dengan tipe ekspor ATR, Anda dapat mengekspor data Daftar Atribusi Anggota. Gunakan jenis ekspor ini untuk mengambil sumber daya terkait atribusi bagi anggota dalam Grup. Untuk informasi lebih lanjut, lihat Operasi Ekspor ATR Da Vinci.

Jenis Sumber Daya yang Didukung

`Group`, `Patient`, `Coverage`, `RelatedPerson`, `Practitioner`, `PractitionerRole`, `Organization`, `Location`

Penyaringan Temporal

Tidak ada penyaringan temporal yang diterapkan. Semua sumber daya yang cocok diekspor terlepas dari tanggal.

PDex Jenis Ekspor

Semua jenis PDex ekspor berbagi profil yang didukung dan logika pemfilteran yang sama. Untuk informasi selengkapnya, lihat Da Vinci PDex Provider Access API. Profil berikut didukung:

- US Core 3.1.1, 6.1.0, dan 7.0.0
- PDex Otorisasi Sebelumnya (tidak didukung untuk Akses Anggota)

- CARIN BB 2.x Profil dasar: Institusional Rawat Inap, Kelembagaan Rawat Jalan, Profesional, Lisan, Farmasi NonClinician

Untuk PDex ekspor, sumber daya klinis dan klaim secara otomatis ditemukan untuk setiap pasien dalam Grup. Anda tidak perlu secara eksplisit mereferensikan sumber daya ini di sumber daya Grup. Operasi mencari semua sumber daya kompartemen pasien (seperti`Observation`,,`Condition`, `Coverage` `RelatedPerson` `MedicationRequest`, dan`ExplanationOfBenefit`) milik pasien yang dikaitkan. Hanya`Patient`,`Group`, dan tipe non-patient-compartment ATR (`Practitioner`,, `PractitionerRole` `Organization`,`Location`) yang memerlukan referensi eksplisit dalam Grup.

Akses Penyedia (`hl7.fhir.us.davinci-pdex`)

Memungkinkan penyedia dalam jaringan untuk mengambil data pasien untuk pasien yang dikaitkan.

Payer-to-Payer (`hl7.fhir.us.davinci-pdex.p2p`)

Memungkinkan pertukaran data antara pembayar ketika pasien mengubah asuransi.

Akses Anggota (`hl7.fhir.us.davinci-pdex.member`)

Memungkinkan anggota untuk mengakses data kesehatan mereka sendiri. Jenis ekspor ini dapat mencakup data keuangan dalam sumber klaim.

Dukungan Profil dan Logika Inklusi

Untuk PDex ekspor, `$davinci-data-export` operasi menggunakan deklarasi profil dalam `meta.profile` elemen untuk menentukan sumber daya mana yang akan disertakan dalam ekspor.

ExplanationOfBenefit Penanganan Sumber Daya

`ExplanationOfBenefit`(EOB) sumber daya dimasukkan atau dikecualikan dari PDex ekspor berdasarkan deklarasinya`meta.profile`:

- ExplanationOfBenefit sumber daya dengan profil CARIN BB 1.x dikecualikan dari ekspor.
- ExplanationOfBenefit sumber daya tanpa `meta.profile` set dikecualikan dari ekspor.
- ExplanationOfBenefit sumber daya dengan profil CARIN BB 2.x Basis selalu disertakan.
- ExplanationOfBenefit sumber daya dengan profil CARIN BB 2.x yang berisi data keuangan dikecualikan secara default. Ketika `_includeEOB2xWoFinancial=true` diatur, mereka

disertakan dengan data keuangan yang dilucuti dan sumber daya diubah ke profil Basis yang sesuai.

- ExplanationOfBenefit sumber daya dengan profil Otorisasi PDex Sebelumnya selalu disertakan.

Transformasi Data Keuangan

Ketika Anda mengatur`_includeEOB2xWoFinancial=true`, operasi mengubah ExplanationOfBenefit sumber daya CARIN BB 2.x ke profil Basis yang sesuai dengan menghapus data keuangan. Misalnya, `C4BB ExplanationOfBenefit Oral` sumber daya diubah menjadi`C4BB ExplanationOfBenefit Oral Basis`, yang menghapus data keuangan dari catatan per spesifikasi FHIR.

Elemen data keuangan berikut dihapus selama transformasi:

- Semua mengiris elemen `total`
- Semua `adjudication` elemen dengan `amounttype` irisan
- Semua `item.adjudication` elemen dengan informasi jumlah

Operasi ini juga memperbarui metadata profil selama transformasi:

- `meta.profile`diperbarui ke URL kanonik profil Dasar
- Versi diperbarui ke versi CARIN BB 2.x Basis
- Sumber daya yang ada di penyimpanan data tidak dimodifikasi
- Sumber daya yang diekspor tidak disimpan kembali ke penyimpanan data

Aturan Deteksi Profil

Operasi menggunakan aturan berikut untuk mendeteksi dan memvalidasi profil:

- Deteksi versi didasarkan pada `meta.profile` kanonik URLs
- Sumber daya disertakan jika ADA profil yang dideklarasikan sesuai dengan kriteria ekspor
- Validasi profil terjadi selama pemrosesan ekspor

Penyaringan Temporal Lima Tahun untuk Ekspor PDex

Untuk semua jenis PDex ekspor, HealthLake terapkan filter temporal 5 tahun berdasarkan kapan sumber daya terakhir diperbarui. Filter temporal berlaku untuk semua sumber daya kecuali jenis sumber daya atribusi inti berikut, yang selalu diekspor tanpa memandang usia:

- `Patient`
- `Coverage`
- `Organization`
- `Practitioner`
- `PractitionerRole`
- `RelatedPerson`
- `Location`
- `Group`

Sumber daya administratif dan demografis ini dikecualikan karena menyediakan konteks penting untuk data yang diekspor. Ekspor ATR tidak tunduk pada penyaringan temporal apa pun.

Sampel Permintaan

Contoh berikut menunjukkan cara memulai pekerjaan ekspor untuk berbagai jenis ekspor.

Ekspor ATR

```
GET https://healthlake.{region}.amazonaws.com/datastore/
{datastoreId}/r4/Group/example-group/$davinci-data-export?
_type=Group,Patient,Coverage,Practitioner,Organization&exportType=hl7.fhir.us.davinci-
atr

POST https://healthlake.{region}.amazonaws.com/datastore/
{datastoreId}/r4/Group/example-group/$davinci-data-export?
_type=Group,Patient,Coverage,Practitioner,Organization&exportType=hl7.fhir.us.davinci-
atr
Content-Type: application/json

{
  "DataAccessRoleArn": "arn:aws:iam::444455556666:role/your-healthlake-service-role",
  "JobName": "attribution-export-job",
  "OutputDataConfig": {
    "S3Configuration": {
```

```
      "S3Uri": "s3://your-export-bucket/EXPORT-JOB",
      "KmsKeyId":
 "arn:aws:kms:region:444455556666:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    }
  }
}
```

Ekspor Akses Penyedia dengan penghapusan data ExplanationOfBenefit keuangan

```
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 ekspor

```
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
```

Ekspor Akses Anggota untuk pasien tertentu

```
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
```

Contoh Respons

```
{
  "datastoreId": "eaee622d8406b41eb86c0f4741201ff9",
  "jobStatus": "SUBMITTED",
  "jobId": "48d7b91dae4a64d00d54b70862f33f61"
}
```

Hubungan Sumber Daya

Operasi mengekspor sumber daya berdasarkan hubungan mereka dalam Daftar Atribusi Anggota:

```
Group (Attribution List)
```

```
### Patient (Members)
### Coverage # RelatedPerson (Subscribers)
### Practitioner (Attributed Providers)
### PractitionerRole # Location
### Organization (Attributed Providers)
```

> **Note**
>
> Diagram hubungan sumber daya sebelumnya berlaku untuk ekspor ATR. Untuk PDex ekspor, sumber daya klinis dan klaim secara otomatis ditemukan melalui pencarian pasien dan tidak memerlukan referensi eksplisit dalam sumber daya Grup.

Sumber Sumber Daya

Sumber daya	Lokasi Sumber	Deskripsi
`Patient`	`Group.member.entity`	Pasien yang menjadi anggota daftar atribusi
`Coverage`	`Group.member.extension:coverageReference`	Cakupan yang mengakibatkan keanggotaan pasien
`Organization`	`Group.member.extension:attributedProvider`	Organizations yang dikaitkan dengan pasien
`Practitioner`	`Group.member.extension:attributedProvider`	Praktisi individu yang dikaitkan dengan pasien
`PractitionerRole`	`Group.member.extension:attributedProvider`	Peran praktisi yang dikaitkan dengan pasien
`RelatedPerson`	`Coverage.subscriber`	Pelanggan cakupan
`Location`	`PractitionerRole.location`	Lokasi yang terkait dengan peran praktisi
`Group`	Titik akhir masukan	Daftar atribusi itu sendiri

Manajemen Job

Periksa Status Job

```
GET [base]/export/[job-id]
```

Batalkan Tugas

```
DELETE [base]/export/[job-id]
```

Siklus Hidup Tugas

- `SUBMITTED`- Job telah diterima dan diantrian
- `IN_PROGRESS`- Job sedang aktif memproses
- `COMPLETED`- Job selesai dengan sukses, file tersedia untuk diunduh
- `FAILED`- Job mengalami kesalahan

Format Output

- Format File: NDJSON (JSON Terbatas Baris Baru)
- Organisasi File: File terpisah untuk setiap jenis sumber daya
- Ekstensi File: .ndjson
- Lokasi: Bucket dan jalur S3 yang ditentukan

Penanganan Kesalahan

Operasi mengembalikan HTTP 400 Bad Request dengan OperationOutcome untuk kondisi berikut:

Kesalahan Otorisasi

Peran IAM yang ditentukan dalam `DataAccessRoleArn` tidak memiliki izin yang cukup untuk melakukan operasi ekspor. Untuk daftar lengkap izin S3 dan KMS yang diperlukan, lihat Menyiapkan izin untuk pekerjaan ekspor.

Kesalahan Validasi Parameter

- `patient`Parameter tidak diformat sebagai `Patient/id,Patient/id,...`
- Satu atau lebih referensi pasien tidak valid atau bukan milik Grup yang ditentukan

- Nilai `exportType` parameter bukan jenis ekspor yang didukung
- `_type`Parameter berisi jenis sumber daya yang tidak didukung untuk jenis ekspor yang ditentukan
- `_type`Parameter tidak memiliki tipe sumber daya yang diperlukan (`GroupPatient`,,`Coverage`) untuk jenis `hl7.fhir.us.davinci-atr` ekspor
- Nilai `_includeEOB2xWoFinancial` parameter bukan boolean yang valid

Kesalahan Validasi Sumber Daya

- Sumber daya Grup yang ditentukan tidak ada di penyimpanan data
- Sumber daya Grup yang ditentukan tidak memiliki anggota
- Satu atau lebih anggota Grup mereferensikan sumber daya Pasien yang tidak ada di penyimpanan data

Keamanan dan Otorisasi

- Mekanisme otorisasi FHIR standar berlaku
- Peran akses data harus memiliki izin IAM yang diperlukan untuk operasi S3 dan KMS. Untuk daftar lengkap izin yang diperlukan, lihat Menyiapkan izin untuk pekerjaan ekspor.

Praktik Terbaik

- Pemilihan Jenis Sumber Daya: Hanya minta jenis sumber daya yang Anda perlukan untuk meminimalkan ukuran ekspor dan waktu pemrosesan
- Penyaringan Berbasis Waktu: Gunakan `_since` parameter untuk ekspor tambahan
- Penyaringan Pasien: Gunakan `patient` parameter saat Anda hanya membutuhkan data untuk anggota tertentu
- Job Monitoring: Secara teratur memeriksa status pekerjaan untuk ekspor besar
- Penanganan Kesalahan: Menerapkan logika coba lagi yang tepat untuk pekerjaan yang gagal
- Kesadaran Filter Temporal: Untuk PDex ekspor, pertimbangkan filter temporal 5 tahun saat Anda memilih jenis sumber daya
- Penghapusan Data Keuangan: Gunakan `_includeEOB2xWoFinancial=true` saat Anda membutuhkan data klaim tanpa informasi keuangan
- Manajemen Profil: Pastikan sumber daya memiliki deklarasi profil yang sesuai, validasi terhadap profil target sebelum konsumsi, dan gunakan versi profil untuk mengontrol perilaku ekspor

Batasan

- Maksimal 500 pasien dapat ditentukan dalam `patient` parameter
- Ekspor terbatas hanya untuk operasi tingkat Grup
- Hanya mendukung kumpulan tipe sumber daya yang telah ditentukan untuk setiap jenis ekspor
- Output selalu dalam format NDJSON
- PDex Ekspor dibatasi hingga 5 tahun data klinis dan klaim
- Transformasi data keuangan hanya berlaku untuk profil CARIN BB 2.x ExplanationOfBenefit

Sumber Daya Tambahan

- Daftar Atribusi Anggota Da Vinci IG
- Da Vinci Payer Data Exchange IG
- CARIN IG Pertukaran Data Exchange Pembayar yang Diarahkan Konsumen
- Panduan Implementasi Inti AS
- Spesifikasi Akses Data Massal FHIR

Menghasilkan Dokumen Klinis dengan **`$document`**

AWS HealthLake sekarang mendukung `$document` operasi untuk sumber daya Komposisi, memungkinkan Anda untuk menghasilkan dokumen klinis lengkap dengan menggabungkan Komposisi dengan semua sumber daya yang direferensikan ke dalam satu paket kohesif. Operasi ini sangat penting untuk aplikasi perawatan kesehatan yang perlu:

- Buat dokumen klinis standar
- Tukar catatan pasien lengkap
- Simpan dokumentasi klinis yang komprehensif
- Hasilkan laporan yang mencakup semua konteks yang relevan

Penggunaan

`$document`Operasi dapat dipanggil pada sumber daya Komposisi menggunakan metode GET dan POST:

Operasi yang Didukung

```
GET/POST [base]/Composition/[id]/$document
```

Parameter yang Didukung

HealthLake mendukung `$document` parameter FHIR berikut:

Parameter	Tipe	Diperlukan	Default	Deskripsi
`persist`	boolean	Tidak	false	Boolean menunjukkan apakah server harus menyimpan bundel dokumen yang dihasilkan

Contoh

DAPATKAN Permintaan

```
GET [base]/Composition/180f219f-97a8-486d-99d9-ed631fe4fc57/$document?persist=true
```

Permintaan POST dengan Parameter

```
POST [base]/Composition/180f219f-97a8-486d-99d9-ed631fe4fc57/$document
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "persist",
      "valueBoolean": true
    }
  ]
}
```

Contoh Respons

Operasi mengembalikan sumber daya Bundle berjenis “dokumen” yang berisi Komposisi dan semua sumber daya yang direferensikan:

```
{
```

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

Perilaku

`$document` Operasi:

1. Mengambil sumber daya Komposisi yang ditentukan sebagai dasar untuk dokumen
2. Mengidentifikasi dan mengambil semua sumber daya yang direferensikan secara langsung oleh Komposisi
3. Mengemas Komposisi dan semua sumber daya yang direferensikan ke dalam Bundel jenis “dokumen”
4. Menyimpan bundel dokumen yang dihasilkan di datastore saat parameter persisten disetel ke true

5. Mengidentifikasi dan mengambil sumber daya yang secara tidak langsung direferensikan oleh Komposisi untuk pembuatan dokumen yang komprehensif

`$document`Operasi saat ini mendukung pengambilan referensi sumber daya dalam format berikut:

1. ```
   GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Resource/id
   ```
2. Sumber daya/ID

Referensi sumber daya yang tidak didukung dalam sumber daya Komposisi akan disaring dari dokumen yang dihasilkan.

Penanganan Kesalahan

Operasi menangani kondisi kesalahan berikut:

- 400 Permintaan Buruk: `$document` Operasi tidak valid (permintaan tidak sesuai) atau jika dokumen yang dihasilkan gagal validasi FHIR karena referensi yang disaring saat persisten disetel ke true
- 404 Tidak Ditemukan: Sumber daya komposisi tidak ditemukan

Untuk informasi lebih lanjut tentang spesifikasi `$document` operasi, lihat dokumentasi Komposisi `$document` FHIR R4.

Menghapus Sumber Daya Secara Permanen dengan **`$erase`**

AWS HealthLake mendukung `$erase` operasi, memungkinkan penghapusan permanen sumber daya tertentu dan versi historisnya. Operasi ini sangat berguna ketika Anda perlu:

- Hapus sumber daya individu secara permanen
- Hapus riwayat versi tertentu
- Mengelola siklus hidup sumber daya individu
- Mematuhi persyaratan penghapusan data tertentu

Penggunaan

`$erase`Operasi dapat dipanggil pada dua tingkat:
   ```

Tingkat Instans Sumber Daya

```
POST [base]/[ResourceType]/[ID]/$erase?deleteAuditEvent=true
```

Level Khusus Versi

```
POST [base]/[ResourceType]/[ID]/_history/[VersionID]/$erase
```

Parameter

Parameter	Tipe	Diperlukan	Default	Deskripsi
`deleteAuditEvent`	boolean	Tidak	false	Jika benar, menghapus peristiwa audit terkait

Contoh

Contoh Permintaan

```
POST [base]/Patient/example-patient/$erase
```

Contoh Respons

```
{
      "jobId": "5df47e2f51ff3c731847678cb8cad48e",
      "jobStatus": "SUBMITTED"
    }
```

Status Tugas

Untuk memeriksa status pekerjaan penghapusan:

```
GET [base]/$erase/[jobId]
```

Operasi mengembalikan informasi status pekerjaan:

```
{
      "datastoreId": "36622996b1fcecb7e12ee2ee085308d3",
      "jobId": "5df47e2f51ff3c731847678cb8cad48e",
      "status": "COMPLETED",
      "submittedTime": "2025-10-30T16:39:24.160Z"
    }
```

Perilaku

`$erase`Operasi:

1. Memproses secara asinkron untuk memastikan integritas data
2. Mempertahankan transaksi ACID
3. Menyediakan pelacakan status pekerjaan
4. Secara permanen menghapus sumber daya yang ditentukan dan versinya
5. Termasuk pencatatan audit yang komprehensif atas kegiatan penghapusan
6. Mendukung penghapusan selektif peristiwa audit

Pencatatan Audit

Log `$erase` operasi seperti DeleteResource ID pengguna, stempel waktu, dan detail sumber daya.

Batasan

- `$erased`sumber daya tidak akan muncul di hasil penelusuran atau `_history` kueri.
- Sumber daya yang dihapus mungkin sementara tidak dapat diakses selama pemrosesan
- Pengukuran penyimpanan segera disesuaikan karena sumber daya dihapus secara permanen

Mendapatkan data pasien dengan **`Patient/$everything`**

`Patient/$everything`Operasi ini digunakan untuk query sumber daya FHIR, bersama dengan `Patient` sumber daya lain yang terkait dengan itu`Patient`. Operasi ini dapat digunakan untuk memberi pasien akses ke seluruh catatan mereka atau untuk penyedia untuk melakukan pengunduhan data massal yang terkait dengan pasien. HealthLakemendukung `Patient/$everything` untuk pasien tertentu`id`.

`Patient/$everything`adalah operasi FHIR REST API yang dapat dipanggil seperti yang ditunjukkan pada contoh di bawah ini.

GET request

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything
```

> **Note**
>
> Sumber daya dalam tanggapan diurutkan berdasarkan jenis sumber daya dan sumber dayaid.
> Respon selalu diisi denganBundle.total.

Parameter **Patient/$everything**

HealthLake mendukung parameter query berikut

Parameter	Detail
start	Dapatkan semua `Patient` data setelah tanggal mulai yang ditentukan.
end	Dapatkan semua `Patient` data sebelum tanggal akhir yang ditentukan.
sejak	Dapatkan semua `Patient` data diperbarui setelah tanggal yang ditentukan.
_ketik	Dapatkan `Patient` data untuk jenis sumber daya tertentu.
_hitung	Dapatkan `Patient` data dan tentukan ukuran halaman.

Example- Dapatkan semua data pasien setelah tanggal mulai yang ditentukan

`Patient/$everything`dapat menggunakan `start` filter untuk menanyakan hanya data setelah tanggal tertentu.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?start=2024-03-15T00:00:00.000Z
```

Example- Dapatkan semua Patient data sebelum tanggal akhir yang ditentukan

Patient $everything dapat menggunakan `end` filter untuk hanya menanyakan data sebelum tanggal tertentu.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?end=2024-03-15T00:00:00.000Z
```

Example- Dapatkan semua Patient data diperbarui setelah tanggal yang ditentukan

`Patient/$everything`dapat menggunakan `since` filter untuk menanyakan hanya data yang diperbarui setelah tanggal tertentu.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?since=2024-03-15T00:00:00.000Z
```

Example- Dapatkan Patient data untuk jenis sumber daya tertentu

Patient $everything dapat menggunakan `_type` filter untuk menentukan jenis sumber daya tertentu yang akan disertakan dalam respons. Beberapa jenis sumber daya dapat ditentukan dalam daftar dipisahkan koma.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?_type=Observation,Condition
```

Example- Dapatkan Patient data dan tentukan ukuran halaman

Pasien $everything dapat menggunakan `_count` untuk mengatur ukuran halaman.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?_count=15
```

`Patient/$everything` **`start`** dan **`end`** atribut

HealthLake mendukung atribut sumber daya berikut untuk parameter `Patient/ $everything` `start` dan end kueri.

Sumber daya	Elemen Sumber Daya
Akun	Account.ServicePeriod.Start
AdverseEvent	AdverseEvent.tanggal
AllergyIntolerance	AllergyIntolerance.RecordedDate
Pengangkatan	Janji temu. Mulai
AppointmentResponse	AppointmentResponse.mulai
AuditEvent	AuditEvent.period.start
Basic	Dasar.dibuat
BodyStructure	TIDAK_TANGGAL
CarePlan	CarePlan.period.start
CareTeam	CareTeam.period.start
ChargeItem	ChargeItem. occurrenceDateTime, ChargeItem .OcuncencePeriod.start, .OcuncenceTiming.event ChargeItem
Klaim	Claim.billablePeriod.Start
ClaimResponse	ClaimResponse.dibuat
ClinicalImpression	ClinicalImpression.tanggal

Sumber daya	Elemen Sumber Daya
Komunikasi	Komunikasi.dikirim
CommunicationRequest	CommunicationRequest. occurrenceDateTime, CommunicationRequest .OcuncencePeriod.start
Komposisi	Komposisi.date
Kondisi	kondisi.RecordedDate
Persetujuan	Consent.DateTime
Cakupan	Cakupan. Period.Start
CoverageEligibilityRequest	CoverageEligibilityRequest.dibuat
CoverageEligibilityResponse	CoverageEligibilityResponse.dibuat
DetectedIssue	DetectedIssue.diidentifikasi
DeviceRequest	DeviceRequest.Authoredon
DeviceUseStatement	DeviceUseStatement.RecordEdon
DiagnosticReport	DiagnosticReport.efektif

Sumber daya	Elemen Sumber Daya
DocumentManifest	DocumentManifest.dibuat
DocumentReference	DocumentReference.context.period.start
Pertemuan	Encounter.period.start
EnrollmentRequest	EnrollmentRequest.dibuat
EpisodeOfCare	EpisodeOfCare.period.start
ExplanationOfBenefit	ExplanationOfBenefit.BillablePeriod.Start
FamilyMemberHistory	TIDAK_TANGGAL
Bendera	Bendera.period.mulai
Tujuan	Goal.statusDate
Kelompok	TIDAK_TANGGAL
ImagingStudy	ImagingStudy.dimulai
Imunisasi	imunisasi.Tercatat
ImmunizationEvaluation	ImmunizationEvaluation.tanggal

Sumber daya	Elemen Sumber Daya
ImmunizationRecommendation	ImmunizationRecommendation.tanggal
Faktur	Faktur.tanggal
Daftar	Daftar.tanggal
MeasureReport	MeasureReport.period.start
Media	media.Diterbitkan
MedicationAdministration	MedicationAdministration.efektif
MedicationDispense	MedicationDispense.Ketika Disiapkan
MedicationRequest	MedicationRequest.PenulisDon
MedicationStatement	MedicationStatement.dateAsserted
Molecular Sequence	TIDAK_TANGGAL
Nutrition Order	NutritionOrder.DateTime
Observasi	pengamatan. Efektif
Pasien	TIDAK_TANGGAL
Orang	TIDAK_TANGGAL

Sumber daya	Elemen Sumber Daya
Prosedur	prosedur.Dilakukan
Asal	Provenance.OcuncredPeriod.Start, Asal. occurredDateTime
QuestionnaireResponse	QuestionnaireResponse.menulis
RelatedPerson	TIDAK_TANGGAL
RequestGroup	RequestGroup.PenulisDon
ResearchSubject	ResearchSubject.periode
RiskAssessment	RiskAssessment. occurrenceDateTime, RiskAssessment .OcuncencePeriod.start
Jadwal	jadwal.planningHorizon
ServiceRequest	ServiceRequest.PenulisDon
Spesimen	Specimen.ReceivedTime
SupplyDelivery	SupplyDelivery. occurrenceDateTime, SupplyDelivery .OcuncencePeriod.start, .OcuncenceTiming.event SupplyDelivery
SupplyRequest	SupplyRequest.PenulisDon
VisionPrescription	VisionPrescription.dateDitulis

Mengambil ValueSet Kode dengan **$expand**

AWS HealthLake sekarang mendukung `$expand` operasi untuk ValueSets yang dicerna oleh Anda sebagai pelanggan, memungkinkan Anda untuk mengambil daftar lengkap kode yang terkandung dalam ValueSet sumber daya tersebut. Operasi ini sangat berguna ketika Anda perlu:

- Ambil semua kode yang mungkin untuk tujuan validasi
- Tampilkan opsi yang tersedia di antarmuka pengguna
- Lakukan pencarian kode komprehensif dalam konteks terminologi tertentu

Penggunaan

`$expand`Operasi dapat dipanggil pada ValueSet sumber daya menggunakan metode GET dan POST:

Operasi yang Didukung

```
GET/POST [base]/ValueSet/[id]/$expand
GET [base]/ValueSet/$expand?url=http://example.com
POST [base]/ValueSet/$expand
```

Parameter yang Didukung

HealthLake mendukung subset parameter FHIR `$expand` R4:

Parameter	Tipe	Diperlukan	Deskripsi
`url`	uri	Tidak	URL kanonik untuk memperluas ValueSet
`id`	id	Tidak	ValueSet id sumber daya untuk diperluas (untuk operasi GET atau POST)
`filter`	string	Tidak	Filter hasil ekspansi kode
`count`	integer	Tidak	Jumlah kode yang akan dikembalikan
`offset`	integer	Tidak	Jumlah kode yang cocok untuk dilewati sebelum kembali. Berlaku setelah pemfilter

Parameter	Tipe	Diperlukan	Deskripsi
			an dan hanya untuk kode yang cocok, bukan untuk konten asli yang lengkap dan tanpa filter ValueSet

Contoh

DAPATKAN Permintaan dengan ID

```
GET [base]/ValueSet/example-valueset/$expand
```

DAPATKAN Permintaan berdasarkan URL dengan Filter

```
GET [base]/ValueSet/$expand?url=http://example.com/ValueSet/my-
valueset&filter=male&count=5
```

Permintaan POST dengan Parameter (berdasarkan ID)

```
POST [base]/ValueSet/example-valueset/$expand
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "count",
      "valueInteger": 10
    },
    {
      "name": "filter",
      "valueString": "admin"
    }
  ]
}
```

Permintaan POST dengan Parameter (berdasarkan URL)

```
POST [base]/ValueSet/$expand
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "url",
      "valueUri": "http://hl7.org/fhir/ValueSet/administrative-gender"
    },
    {
      "name": "count",
      "valueInteger": 10
    }
  ]
}
```

Contoh Respons

Operasi mengembalikan ValueSet sumber daya dengan `expansion` elemen yang berisi kode diperluas:

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

Respons meliputi:

- expansion.total: Jumlah total kode yang diperluas ValueSet
- expansion.contains: Array kode yang diperluas dengan sistem, kode, dan nilai tampilannya
- expansion.parameter: Parameter yang digunakan dalam permintaan ekspansi

Untuk informasi lebih lanjut tentang spesifikasi `$expand` operasi, lihat dokumentasi FHIR R4 ValueSet `$expand`.

Mengekspor HealthLake data dengan FHIR **`$export`**

Anda dapat mengekspor data secara massal dari penyimpanan HealthLake data Anda menggunakan operasi FHIR $export. HealthLake mendukung `$export` penggunaan `POST` dan `GET` permintaan FHIR. Untuk membuat permintaan ekspor dengan`POST`, Anda harus memiliki pengguna, grup, atau peran IAM dengan izin yang diperlukan, tentukan `$export` sebagai bagian dari permintaan, dan sertakan parameter yang diinginkan dalam badan permintaan.

> **Note**
>
> Semua permintaan HealthLake ekspor yang dibuat menggunakan FHIR dikembalikan dalam `ndjson` format dan diekspor ke bucket Amazon S3, di `$export` mana setiap objek Amazon S3 hanya berisi satu jenis sumber daya FHIR.

Anda dapat mengantri permintaan ekspor per kuota layanan AWS akun. Untuk informasi selengkapnya, lihat Kuota layanan.

HealthLake mendukung tiga jenis permintaan titik akhir ekspor massal berikut.

HealthLake **`$export`**jenis massal

Tipe ekspor	Deskripsi	Sintaksis
Sistem	Ekspor semua data dari server HealthLake FHIR.	`POST https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/$export`

Semua pasien	Ekspor semua data yang berkaitan dengan semua pasien termasuk jenis sumber daya yang terkait dengan jenis sumber daya Pasien.	`POST https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/Patient/$export`
		
		`GET https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/Patient/$export`

Kelompo pasien	Ekspor semua data yang berkaitan dengan sekelompok pasien yang ditentukan dengan ID Grup.	`POST https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/Group/ id/$export`
		
		`GET https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/Group/ id/$export`

Sebelum Anda mulai

Memenuhi persyaratan berikut untuk membuat permintaan ekspor menggunakan FHIR REST API untuk HealthLake.

- Anda harus menyiapkan pengguna, grup, atau peran yang memiliki izin yang diperlukan untuk membuat permintaan ekspor. Untuk mempelajari selengkapnya, lihat Mengotorisasi permintaan `$export`.
- Anda harus telah membuat peran layanan yang memberikan HealthLake akses ke bucket Amazon S3 tempat Anda ingin data Anda diekspor. Peran layanan juga harus ditentukan HealthLake sebagai kepala layanan. Untuk informasi selengkapnya tentang menyiapkan izin, lihatMenyiapkan izin untuk pekerjaan ekspor.

Mengotorisasi permintaan **`$export`**

Untuk membuat permintaan ekspor yang berhasil menggunakan FHIR REST API, otorisasi pengguna, grup, atau peran Anda menggunakan IAM atau .0. OAuth2 Anda juga harus memiliki peran layanan.

Mengotorisasi permintaan menggunakan IAM

Saat Anda membuat `$export` permintaan, pengguna, grup, atau peran harus memiliki tindakan IAM yang disertakan dalam kebijakan. Untuk informasi selengkapnya, lihat Menyiapkan izin untuk pekerjaan ekspor.

Mengotorisasi permintaan menggunakan SMART di FHIR (2.0) OAuth

Saat Anda membuat `$export` permintaan pada SMART di penyimpanan HealthLake data berkemampuan FHIR, Anda harus memiliki cakupan yang sesuai yang ditetapkan. Untuk informasi selengkapnya, lihat SMART pada cakupan sumber daya FHIR untuk HealthLake.

> **Note**
>
> FHIR `$export` dengan `GET` permintaan memerlukan metode otentikasi atau token pembawa yang sama (dalam kasus SMART di FHIR) untuk meminta ekspor dan mengambil file. File yang diekspor menggunakan FHIR `$export` dengan `GET` tersedia untuk diunduh selama 48 jam.

Membuat **`$export`** permintaan

Bagian ini menjelaskan langkah-langkah yang diperlukan yang harus Anda ambil saat membuat permintaan ekspor menggunakan FHIR REST API.

Untuk menghindari tagihan yang tidak disengaja pada AWS akun Anda, kami sarankan untuk menguji permintaan Anda dengan membuat POST permintaan tanpa menyediakan sintaks. `$export`

Untuk membuat permintaan, Anda harus melakukan hal berikut:

1. Tentukan `$export` di URL POST permintaan untuk titik akhir yang didukung.
2. Tentukan parameter header yang diperlukan.
3. Tentukan badan permintaan yang mendefinisikan parameter yang diperlukan.

Langkah 1: Tentukan **`$export`** URL **`POST`** permintaan untuk titik akhir yang didukung.

HealthLake mendukung tiga jenis permintaan titik akhir ekspor massal. Untuk membuat permintaan ekspor massal, Anda harus membuat permintaan POST berbasis pada salah satu dari tiga titik akhir yang didukung. Contoh berikut menunjukkan di mana harus menentukan `$export` dalam URL permintaan.

- `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`

Anda dapat menggunakan parameter pencarian yang didukung berikut dalam string POST permintaan.

Parameter pencarian yang didukung

HealthLake mendukung pengubah pencarian berikut dalam permintaan ekspor massal.

Contoh berikut termasuk karakter khusus yang harus dikodekan sebelum mengirimkan permintaan Anda.

Nama	Wajib?	Deskripsi	Contoh
`_outputFormat`	Tidak	Format untuk file Data Massal yang diminta untuk dihasilkan.	

Nama	Wajib?	Deskripsi	Contoh
		Nilai yang diterima adalah`application/fhir+ndjson` ,`application/ndjson` ,`ndjson`.	
`_type`	Tidak	Serangkaian jenis sumber daya FHIR yang dibatasi koma yang ingin Anda sertakan dalam pekerjaan ekspor Anda. Kami merekomendasikan termasuk `_type` karena ini dapat memiliki implikasi biaya ketika semua sumber daya diekspor.	`&_type=MedicationStatement,Observation`
`_since`	Tidak	Jenis sumber daya dimodifikasi pada atau setelah stempel tanggal waktu. Jika jenis sumber daya tidak memiliki waktu pembaruan terakhir, mereka akan disertakan dalam respons Anda.	`&_since=2024-05-09T00%3A00%3A00Z`

Nama	Wajib?	Deskripsi	Contoh
`_until`	Tidak	Jenis sumber daya dimodifikasi pada atau sebelum stempel tanggal waktu. Digunakan dalam kombinasi dengan `_since` untuk menentukan rentang waktu tertentu untuk ekspor. Jika jenis sumber daya tidak memiliki waktu pembaruan terakhir, mereka akan dikecualikan dari respons Anda.	`&_until=2024-12-31T23%3A59%3A59Z`
`_security`	Tidak	Filter sumber daya yang diekspor dengan nilai `meta.security` Coding. Gunakan `system|code` formatnya. Ketika beberapa nilai disediakan, sumber daya harus cocok dengan semuanya (DAN semantik). Gunakan `system|` (trailing pipe, no code) untuk mencocokkan kode apa pun dari sistem tertentu.	`&_security=https://myorg.com/tenant%7Cclinic-A`

Nama	Wajib?	Deskripsi	Contoh
`_tag`	Tidak	Filter sumber daya yang diekspor dengan nilai `meta.tag` Coding. Menggunakan `system|code` format dan semantik yang sama dengan. `_security` Ketika keduanya `_security` dan `_tag` ditentukan, sumber daya harus cocok dengan kedua filter.	`&_tag=https://myorg.com/dept%7Ccardiology`

Langkah 2: Tentukan parameter header yang diperlukan

Untuk membuat permintaan ekspor menggunakan FHIR REST API, Anda harus menentukan parameter header berikut.

- Tipe Konten: `application/fhir+json`
- Lebih suka: `respond-async`

Selanjutnya, Anda harus menentukan elemen yang diperlukan di badan permintaan.

Langkah 3: Tentukan badan permintaan yang menentukan parameter yang diperlukan.

Permintaan ekspor juga membutuhkan badan dalam `JSON` format. Tubuh dapat mencakup parameter berikut.

Key	Wajib?	Deskripsi	Nilai
DataAccessRoleArn	Ya	ARN dari peran HealthLake layanan. Peran layanan yang	`arn:aws:iam:: 444455556`

Key	Wajib?	Deskripsi	Nilai
		digunakan harus ditentukan HealthLake sebagai prinsip layanan.	**666** :role/**your-healthlake-service-role**
JobName	Tidak	Nama permintaan ekspor.	**your-export-job-name**
S3Uri	Ya	Bagian dari OutputDataConfig kunci. URI S3 dari bucket tujuan tempat data Anda yang diekspor akan diunduh.	s3://amzn-s3-demo-bucket/ **EXPORT-JOB** /
KmsKeyId	Ya	Bagian dari OutputDataConfig kunci. ARN dari AWS KMS kunci yang digunakan untuk mengamankan bucket Amazon S3.	arn:aws:kms: **region-of-bucket:123456789012** :key/**1234abcd-12ab-34cd-56ef-1234567890ab**

Example Isi permintaan ekspor yang dibuat menggunakan FHIR REST API

Untuk membuat permintaan ekspor dengan menggunakan FHIR REST API, Anda harus menentukan isi, seperti yang ditunjukkan dalam berikut ini.

```
{
  "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"
    }
```

```
  }
}
```

Ketika permintaan Anda berhasil, Anda akan menerima tanggapan berikut.

Respon Header

```
content-location: https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
export/your-export-request-job-id
```

Respon Tubuh

```
{
  "datastoreId": "your-data-store-id",
  "jobStatus": "SUBMITTED",
  "jobId": "your-export-request-job-id"
}
```

Mengelola permintaan ekspor Anda

Setelah membuat permintaan ekspor berhasil, Anda dapat mengelola permintaan menggunakan `$export` untuk menjelaskan status permintaan ekspor saat ini, dan `$export` untuk membatalkan permintaan ekspor saat ini.

Saat membatalkan permintaan ekspor menggunakan REST API, Anda hanya ditagih untuk bagian data yang diekspor hingga saat Anda mengirimkan permintaan pembatalan.

Topik berikut menjelaskan bagaimana Anda bisa mendapatkan status atau membatalkan permintaan ekspor saat ini.

Membatalkan permintaan ekspor

Untuk membatalkan permintaan ekspor, buat `DELETE` permintaan dan berikan ID pekerjaan di URL permintaan.

```
DELETE https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/export/your-
export-request-job-id
```

Ketika permintaan Anda berhasil, Anda menerima yang berikut ini.

```
{
```

```
  "exportJobProperties": {
    "jobId": "your-original-export-request-job-id",
    "jobStatus": "CANCEL_SUBMITTED",
    "datastoreId": "your-data-store-id"
  }
}
```

Ketika permintaan Anda tidak berhasil, Anda menerima yang berikut ini.

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "not-supported",
      "diagnostics": "Interaction not supported."
    }
  ]
}
```

Menjelaskan permintaan ekspor

Untuk mendapatkan status permintaan ekspor, buat `GET` permintaan dengan menggunakan `export` dan Anda**`export-request-job-id`**.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/export/your-export-request-id
```

Respons JSON akan berisi `ExportJobProperties` objek. Ini mungkin berisi pasangan key:value berikut.

Nama	Wajib?	Deskripsi	Nilai
DataAccessRoleArn	Tidak	ARN dari peran HealthLake layanan. Peran layanan yang digunakan harus ditentukan HealthLake sebagai prinsip layanan.	`arn:aws:iam:: 444455556666 :role/your-healthlake-service-role`

Nama	Wajib?	Deskripsi	Nilai
SubmitTime	Tidak	Tanggal waktu pekerjaan ekspor diajukan.	`Apr 21, 2023 5:58:02`
EndTime	Tidak	Waktu pekerjaan ekspor selesai.	`Apr 21, 2023 6:00:08 PM`
JobName	Tidak	Nama permintaan ekspor.	**`your-export-job-name`**
JobStatus	Tidak		Nilai yang valid adalah: `SUBMITTED | IN_PROGRESS | COMPLETED_WITH_ERRORS | COMPLETED | FAILED`
S3Uri	Ya	Bagian dari suatu OutputDataConfigobjek. URI Amazon S3 dari bucket tujuan tempat data yang Anda ekspor akan diunduh.	`s3://amzn-s3-demo-bucket/` **`EXPORT-JOB /`**
KmsKeyId	Ya	Bagian dari suatu OutputDataConfigobjek. ARN dari AWS KMS kunci yang digunakan untuk mengamankan bucket Amazon S3.	`arn:aws:kms:` **`region-of-bucket:123456789012`** `:key/`**`1234abcd-12ab-34cd-56ef-1234567890ab`**

Example: Isi permintaan ekspor deskripsi yang dibuat menggunakan FHIR REST API

Ketika berhasil, Anda akan mendapatkan respon JSON berikut.

```
{
  "exportJobProperties": {
    "jobId": "your-export-request-id",
    "JobName": "your-export-job",
    "jobStatus": "SUBMITTED",
    "submitTime": "Apr 21, 2023 5:58:02 PM",
    "endTime": "Apr 21, 2023 6:00:08 PM",
    "datastoreId": "your-data-store-id",
    "outputDataConfig": {
      "s3Configuration": {
        "S3Uri": "s3://amzn-s3-demo-bucket/EXPORT-JOB",
        "KmsKeyId": "arn:aws:kms:region-of-
bucket:444455556666:key/1234abcd-12ab-34cd-56ef-1234567890ab""
      }
    },
    "DataAccessRoleArn": "arn:aws:iam::444455556666:role/your-healthlake-service-role",
  }
}
```

$inquire Operasi FHIR untuk HealthLake

`$inquire`Operasi ini memungkinkan Anda untuk memeriksa status permintaan otorisasi sebelumnya yang diajukan sebelumnya. Operasi ini mengimplementasikan Panduan Implementasi Da Vinci Prior Authorization Support (PAS), menyediakan alur kerja berbasis FHIR standar untuk mengambil keputusan otorisasi saat ini.

Cara kerjanya

- Kirim Pertanyaan: Anda mengirim Paket FHIR yang berisi Klaim yang ingin Anda periksa dan informasi pendukung
- Cari: HealthLake mencari yang sesuai ClaimResponse di penyimpanan data Anda
- Ambil: Status otorisasi terbaru diambil
- Tanggapan: Anda menerima tanggapan langsung dengan status otorisasi saat ini (mengantri, disetujui, ditolak, dll.)

> **Note**
>
> `$inquire`adalah operasi hanya-baca yang mengambil status otorisasi yang ada. Itu tidak mengubah atau memperbarui sumber daya apa pun di penyimpanan data Anda.

Titik akhir API

```
POST /datastore/{datastoreId}/r4/Claim/$inquire
Content-Type: application/fhir+json
```

Struktur permintaan

Persyaratan bundel

Permintaan Anda harus berupa sumber daya Bundel FHIR dengan:

- Bundle.type: Harus `"collection"`
- Bundle.entry: Harus berisi tepat satu sumber Klaim dengan:
  - `use = "preauthorization"`
  - `status = "active"`
- Sumber Daya yang Direferensikan: Semua sumber daya yang direferensikan oleh Klaim harus disertakan dalam Bundel

> **Kueri demi contoh**
>
> Sumber daya dalam Bundle masukan Anda berfungsi sebagai template pencarian. HealthLake menggunakan informasi yang diberikan untuk menemukan yang sesuai ClaimResponse.

Sumber daya yang dibutuhkan

Sumber daya	Kardinalitas	Profil	Deskripsi
Klaim	1	Permintaan Klaim PAS	Otorisasi sebelumnya yang Anda tanyakan

Sumber daya	Kardinalitas	Profil	Deskripsi
Pasien	1	Pasien Penerima PAS	Informasi demografis pasien
Organisasi (Penanggung)	1	Organisasi Penanggung PAS	Perusahaan asuransi
Organisasi (Penyedia)	1	Organisasi Permintaan PAS	Penyedia layanan kesehatan yang mengajukan permintaan

Kriteria pencarian penting

HealthLake pencarian untuk ClaimResponse menggunakan:

- Referensi pasien dari Klaim
- Referensi Penanggung dari Klaim
- Referensi penyedia dari Klaim
- Tanggal dibuat dari Klaim (sebagai filter waktu)

> **Hanya Pertanyaan Khusus Pasien**
>
> Semua pertanyaan harus dikaitkan dengan pasien tertentu. Pertanyaan di seluruh sistem tanpa identifikasi pasien tidak diizinkan.

Contoh permintaan

```
POST /datastore/example-datastore/r4/Claim/$inquire
Content-Type: application/fhir+json
Authorization: Bearer <your-token>

{
  "resourceType": "Bundle",
  "id": "PASClaimInquiryBundleExample",
  "meta": {
    "profile": ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-pas-
inquiry-request-bundle"]
  },
```

```
  "identifier": {
    "system": "http://example.org/SUBMITTER_TRANSACTION_IDENTIFIER",
    "value": "5269368"
  },
  "type": "collection",
  "timestamp": "2005-05-02T14:30:00+05:00",
  "entry": [
    {
      "fullUrl": "http://example.org/fhir/Claim/MedicalServicesAuthorizationExample",
      "resource": {
        "resourceType": "Claim",
        "id": "MedicalServicesAuthorizationExample",
        "meta": {
          "profile": ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
claim-inquiry"]
        },
        "status": "active",
        "type": {
          "coding": [{
            "system": "http://terminology.hl7.org/CodeSystem/claim-type",
            "code": "professional"
          }]
        },
        "use": "preauthorization",
        "patient": {
          "reference": "Patient/SubscriberExample"
        },
        "created": "2005-05-02T11:01:00+05:00",
        "insurer": {
          "reference": "Organization/InsurerExample"
        },
        "provider": {
          "reference": "Organization/UMOExample"
        }
      }
    },
    {
      "fullUrl": "http://example.org/fhir/Patient/SubscriberExample",
      "resource": {
        "resourceType": "Patient",
        "id": "SubscriberExample",
        "meta": {
          "profile": ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
beneficiary"]
```

```
        },
        "name": [{
          "family": "SMITH",
          "given": ["JOE"]
        }],
        "gender": "male"
      }
    },
    {
      "fullUrl": "http://example.org/fhir/Organization/UMOExample",
      "resource": {
        "resourceType": "Organization",
        "id": "UMOExample",
        "meta": {
          "profile": ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
requestor"]
        },
        "name": "Provider Organization"
      }
    },
    {
      "fullUrl": "http://example.org/fhir/Organization/InsurerExample",
      "resource": {
        "resourceType": "Organization",
        "id": "InsurerExample",
        "meta": {
          "profile": ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
insurer"]
        },
        "name": "Insurance Company"
      }
    }
  ]
}
```

Format respons

Respon sukses (200 OK)

Anda akan menerima Paket Respons Pertanyaan PAS yang berisi:

- ClaimResponsedengan status otorisasi saat ini; beberapa ClaimResponsejika cocok dengan kriteria pencarian

- Semua sumber daya asli dari permintaan Anda (bergema kembali)
- Stempel waktu saat respons dirakit

Kemungkinan ClaimResponse Hasil

Hasil	Deskripsi
`queued`	Permintaan otorisasi masih menunggu peninjauan
`complete`	Keputusan otorisasi telah dibuat (periksa `disposition` untuk disetujui /ditolak)
`error`	Terjadi kesalahan selama pemrosesan
`partial`	Otorisasi sebagian diberikan

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

```
}
```

Tanggapan kesalahan

400 Permintaan Buruk

Dikembalikan ketika format permintaan tidak valid atau validasi gagal.

```
{
    "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 Tidak Sah

Dikembalikan ketika kredensi otentikasi hilang atau tidak valid.

```
{
    "resourceType": "OperationOutcome",
    "issue": [
        {
            "severity": "error",
            "code": "forbidden",
            "diagnostics": "Invalid authorization header"
        }
    ]
}
```

403 Dilarang

Dikembalikan ketika pengguna yang diautentikasi tidak memiliki izin untuk mengakses sumber daya yang diminta.

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

```
        {
            "severity": "error",
            "code": "exception",
            "diagnostics": "Insufficient SMART scope permissions."
        }
    ]
}
```

400 Ketika tidak ada yang ditemukan

Dikembalikan ketika tidak ClaimResponse ada kecocokan yang ditemukan untuk penyelidikan.

```
{
  "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 Jenis Media yang Tidak Didukung

Dikembalikan ketika header Content-Type bukan application/fhir+json.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "value",
    "diagnostics": "Incorrect MIME-type. Update request Content-Type header."
  }]
}
```

429 Terlalu Banyak Permintaan

Dikembalikan ketika batas tarif terlampaui.

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

```
    "severity": "error",
    "code": "throttled",
    "diagnostics": "Rate limit exceeded. Please retry after some time."
  }]
}
```

Aturan validasi

HealthLake melakukan validasi komprehensif atas pertanyaan Anda:

Validasi bundel

- Harus sesuai dengan profil Paket Permintaan Pertanyaan PAS
- `Bundle.type`harus `"collection"`
- Harus berisi persis satu sumber Klaim
- Semua sumber daya yang direferensikan harus disertakan dalam Bundel

Validasi klaim

- Harus sesuai dengan profil Permintaan Klaim PAS
- `Claim.use`harus `"preauthorization"`
- `Claim.status`harus `"active"`
- Bidang yang diperlukan:`patient`,`insurer`,`provider`, `created`

Validasi sumber daya

- Semua sumber daya harus sesuai dengan profil Penyelidikan PAS masing-masing
- Sumber daya pendukung yang diperlukan harus ada (Pasien, Organisasi Penanggung, Organisasi Penyedia)
- Referensi silang harus valid dan dapat diselesaikan dalam Bundel

Spesifikasi kinerja

Metrik	Spesifikasi
Batas Hitungan Sumber Daya	500 sumber daya per Bundel

Metrik	Spesifikasi
Batas Ukuran Bundel	Maksimum 5 MB

Izin yang diperlukan

Untuk menggunakan `$inquire` operasi, pastikan peran IAM Anda memiliki:

- `healthlake:InquirePreAuthClaim`- Untuk memanggil operasi

SMART pada Lingkup FHIR

Cakupan minimum yang diperlukan:

- SMART v1: `user/ClaimResponse.read`
- SMART v2: `user/ClaimResponse.s`

Catatan implementasi penting

Perilaku pencarian

Saat Anda mengirimkan pertanyaan, HealthLake cari penggunaan: ClaimResponse

- Referensi pasien dari klaim masukan
- Referensi Penanggung dari Masukan Klaim
- Referensi penyedia dari Klaim masukan
- Tanggal dibuat dari Klaim masukan (sebagai filter waktu)

Beberapa Kecocokan: Jika beberapa ClaimResponses cocok dengan kriteria pencarian Anda, HealthLake mengembalikan semua hasil yang cocok. Anda harus menggunakan `ClaimResponse.created` stempel waktu terbaru untuk mengidentifikasi status terbaru.

Klaim yang diperbarui

Jika Anda telah mengirimkan beberapa pembaruan ke otorisasi sebelumnya yang sama (misalnya, Klaim v1.1, v1.2, v1.3), `$inquire` operasi akan mengambil yang ClaimResponse terkait dengan versi terbaru berdasarkan kriteria pencarian yang disediakan.

Operasi hanya-baca

`$inquire`Operasi:

- Apakah mengambil status otorisasi yang ada
- Apakah mengembalikan yang terbaru ClaimResponse
- Tidak mengubah atau memperbarui sumber daya apa pun
- Tidak membuat sumber daya baru
- Tidak memicu pemrosesan otorisasi baru

Contoh alur kerja

Alur Kerja Permintaan Otorisasi Sebelumnya yang Khas

```
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"
```

Operasi terkait

- `Claim/$submit`- Kirim permintaan otorisasi sebelumnya yang baru atau perbarui yang sudah ada
- `Patient/$everything`- Ambil data pasien yang komprehensif untuk konteks otorisasi sebelumnya

Mengambil Detail Konsep dengan **`$lookup`**

AWS HealthLake sekarang mendukung `$lookup` operasi untuk CodeSystem sumber daya, memungkinkan Anda untuk mengambil rincian tentang konsep tertentu dalam sistem kode dengan memberikan informasi identifikasi seperti kodenya. Operasi ini sangat berguna ketika Anda perlu:

- Ambil informasi rinci tentang kode medis tertentu
- Validasi arti dan properti kode
- Akses definisi dan hubungan konsep
- Support pengambilan keputusan klinis dengan data terminologi yang akurat

Penggunaan

`$lookup`Operasi dapat dipanggil pada CodeSystem sumber daya menggunakan metode GET dan POST:

Operasi yang Didukung

```
GET [base]/CodeSystem/$lookup?system=http://snomed.info/
sct&code=73211009&version=20230901
POST [base]/CodeSystem/$lookup
```

Parameter yang Didukung

HealthLake mendukung subset parameter FHIR `$lookup` R4:

Parameter	Tipe	Diperlukan	Deskripsi
`code`	code	Ya	Kode konsep yang Anda cari (misalnya, “71620000" di SNOMED CT)
`system`	uri	Ya	URL kanonik dari sistem kode (misalnya, "http://snomed.info/sct “)
`version`	string	Tidak	Versi spesifik dari sistem kode

Contoh

DAPATKAN Permintaan

```
GET [base]/CodeSystem/$lookup?system=http://snomed.info/
sct&code=71620000&version=2023-09
```

Permintaan POST

```
POST [base]/CodeSystem/$lookup
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "system",
      "valueUri": "http://snomed.info/sct"
    },
    {
      "name": "code",
      "valueCode": "71620000"
    },
    {
      "name": "version",
      "valueString": "2023-09"
    }
  ]
}
```

Contoh Respons

Operasi mengembalikan sumber daya Parameter yang berisi rincian konsep:

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

Parameter Respons

Respons mencakup parameter berikut bila tersedia:

Parameter	Jenis	Deskripsi
`name`	string	Nama sistem kode
`version`	string	Versi sistem kode
`display`	string	Nama tampilan konsep
`designation`	BackboneElement	Representasi tambahan untuk konsep ini.
`property`	BackboneElement	Properti tambahan dari konsep (definisi, hubungan, dll.)

Perilaku

`$lookup`Operasi:

1. Memvalidasi parameter yang diperlukan (`codedansystem`)
2. Mencari konsep dalam sistem kode tertentu yang disimpan dalam datastore
3. Mengembalikan informasi konsep rinci termasuk nama tampilan, sebutan, dan properti.
4. Mendukung pencarian khusus versi saat parameter disediakan `version`
5. Beroperasi hanya pada sistem kode yang disimpan secara eksplisit di datastore HealthLake

Penanganan Kesalahan

Operasi menangani kondisi kesalahan berikut:

- 400 Permintaan Buruk: `$lookup` Operasi tidak valid (permintaan tidak sesuai atau parameter yang diperlukan tidak ada)
- 404 Tidak Ditemukan: Sistem kode tidak ditemukan atau kode tidak ditemukan dalam sistem kode yang ditentukan

Peringatan

Untuk rilis ini, berikut ini tidak didukung:

- `$lookup`operasi dengan memanggil server terminologi eksternal
- `$lookup`operasi pada CodeSystems dikelola oleh HealthLake tetapi tidak secara eksplisit disimpan di datastore

Untuk informasi lebih lanjut tentang spesifikasi `$lookup` operasi, lihat dokumentasi FHIR R4 CodeSystem `$lookup`.

`$member-add`operasi untuk HealthLake

`$member-add`Operasi FHIR menambahkan anggota (pasien) ke sumber daya Grup, khususnya Daftar Atribusi Anggota. Operasi ini merupakan bagian dari Panduan Implementasi Atribusi DaVinci Anggota dan mendukung proses rekonsiliasi untuk mengelola atribusi anggota.

Titik Akhir Operasi

```
POST [base]/datastore/{datastoreId}/r4/Group/{groupId}/$member-add
Content-Type: application/json
```

Parameter

Operasi menerima sumber daya Parameter FHIR dengan kombinasi parameter berikut:

Opsi Parameter

Anda dapat menggunakan salah satu kombinasi parameter berikut:

Opsi 1: ID Member+NPI Penyedia

`memberId` + `providerNpi`

`memberId` + `providerNpi` + `attributionPeriod`

Opsi 2: Referensi Pasien+Referensi Penyedia

`patientReference` + `providerReference`

`patientReference` + `providerReference` + `attributionPeriod`

Rincian Parameter

MemberID (Opsional)

Pengenal anggota yang akan ditambahkan ke Grup.

Jenis: Identifier

Sistem: Sistem pengenal pasien

```
{
  "name": "memberId",
  "valueIdentifier": {
    "system": "http://example.org/patient-id",
    "value": "patient-new"
  }
}
```

ProviderNPI (Opsional)

Pengenal Penyedia Nasional (NPI) dari penyedia yang dikaitkan.

Jenis: Identifier

Sistem: http://terminology.hl7. org/CodeSystem/NPI

```
{
  "name": "providerNpi",
  "valueIdentifier": {
    "system": "http://terminology.hl7.org/CodeSystem/NPI",
    "value": "1234567890"
  }
}
```

PatientReference (Opsional)

Referensi langsung ke sumber daya pasien yang akan ditambahkan.

Jenis: Referensi

```
{
  "name": "patientReference",
  "valueReference": {
```

```
    "reference": "Patient/patient-123"
  }
}
```

ProviderReference (Opsional)

Referensi langsung ke sumber daya penyedia.

Jenis: Referensi

```
{
  "name": "providerReference",
  "valueReference": {
    "reference": "Practitioner/provider-456"
  }
}
```

Periode Atribusi (Opsional)

Periode waktu di mana pasien dikaitkan dengan penyedia.

Jenis: Periode

```
{
  "name": "attributionPeriod",
  "valuePeriod": {
    "start": "2024-07-15",
    "end": "2025-07-14"
  }
}
```

Minta Contoh

Menggunakan ID Anggota dan NPI Penyedia

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

Menggunakan Referensi Pasien dan Penyedia

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "patientReference",
      "valueReference": {
        "reference": "Patient/patient-123"
      }
    },
    {
      "name": "providerReference",
      "valueReference": {
        "reference": "Practitioner/provider-456"
      }
    },
    {
      "name": "attributionPeriod",
      "valuePeriod": {
        "start": "2024-07-15",
        "end": "2025-07-14"
      }
```

```
    }
  ]
}
```

Format Respons

Respon Penambahan yang Berhasil

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

Respons Kesalahan

Sintaks Permintaan Tidak Valid

Status HTTP: 400 Permintaan Buruk

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "invalid",
      "details": {
        "text": "Invalid parameter combination provided"
      }
    }
  ]
}
```

Sumber Daya Tidak Ditemukan

Status HTTP: 404 Tidak Ditemukan

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "not-found",
      "details": {
        "text": "Resource not found."
      }
    }
  ]
}
```

Konflik Versi

Status HTTP: 409 Konflik

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "conflict",
      "details": {
        "text": "Resource version conflict detected"
      }
    }
  ]
}
```

Status Atribusi Tidak Valid

Status HTTP: 422 Entitas yang Tidak Dapat Diproses

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

Aturan Bisnis

Validasi Status Atribusi

Operasi hanya dapat dilakukan jika Status Atribusi Grup adalah:

- `draft`
- `open`

Operasi tidak diperbolehkan ketika statusnya`final`.

Pencegahan Anggota Duplikat

Sistem mencegah penambahan anggota duplikat berdasarkan kombinasi unik dari:

- Pengenal Anggota
- Pengidentifikasi Pembayar
- Pengidentifikasi Cakupan

Validasi Periode Cakupan

Ketika `attributionPeriod` disediakan, itu harus berada dalam batas-batas periode pertanggungan anggota. Sistem akan:

- Cari sumber daya Cakupan anggota
- Gunakan Cakupan terbaru (versionId tertinggi) jika ada beberapa
- Memvalidasi bahwa periode atribusi tidak melebihi periode pertanggungan

Validasi Referensi

Ketika ID dan referensi disediakan untuk sumber daya yang sama (pasien atau penyedia), sistem memvalidasi bahwa mereka sesuai dengan sumber daya yang sama.

Ketika bidang ID dan reference.identifier disediakan untuk sumber daya yang sama (pasien atau penyedia), kesalahan akan muncul.

Otentikasi & Otorisasi

Operasi ini membutuhkan otorisasi SMART pada FHIR untuk:

- Izin baca - Untuk memvalidasi sumber daya pasien, penyedia, dan grup
- Izin pencarian - Untuk menemukan sumber daya berdasarkan pengenal
- Perbarui izin - Untuk memodifikasi sumber daya Grup

Perilaku Operasional

Pembaruan Sumber Daya

- Memperbarui ID versi sumber daya Grup
- Membuat entri riwayat dengan status sumber daya asli sebelum operasi
- Menambahkan informasi anggota ke array Group.member dengan:
  - Referensi pasien di entity.reference
  - Periode atribusi dalam periode
  - Informasi cakupan dan penyedia di bidang ekstensi

Langkah Validasi

- Validasi Parameter - Memastikan kombinasi parameter yang valid
- Keberadaan Sumber Daya - Memvalidasi sumber daya pasien, penyedia, dan kelompok yang ada
- Status Atribusi - Mengonfirmasi status grup memungkinkan modifikasi
- Pemeriksaan Duplikat - Mencegah penambahan anggota yang ada
- Validasi Cakupan - Memastikan periode atribusi berada dalam batas cakupan

Batasan

- Semua sumber daya yang direferensikan harus ada dalam datastore yang sama
- Operasi hanya berfungsi dengan sumber daya Grup Daftar Atribusi Anggota
- Periode atribusi harus berada dalam batas pertanggungan

- Tidak dapat mengubah grup dengan status “final”

`$member-match`operasi untuk HealthLake

AWS HealthLake sekarang mendukung `$member-match` operasi untuk sumber daya Pasien, memungkinkan organisasi perawatan kesehatan untuk menemukan pengenal unik anggota di berbagai sistem perawatan kesehatan menggunakan informasi demografis dan cakupan. Operasi ini sangat penting untuk mencapai kepatuhan CMS dan memfasilitasi pertukaran payer-to-payer data yang aman sambil menjaga privasi pasien.

Operasi ini sangat berguna ketika Anda perlu:

- Aktifkan pertukaran data perawatan kesehatan yang aman antar organisasi
- Pertahankan kontinuitas perawatan pasien di berbagai sistem
- Mendukung persyaratan kepatuhan CMS
- Memfasilitasi identifikasi anggota yang akurat di seluruh jaringan perawatan kesehatan

Penggunaan

`$member-match`Operasi dapat dipanggil pada sumber daya Pasien menggunakan metode POST:

```
POST [base]/Patient/$member-match
```

Parameter yang Didukung

HealthLake mendukung `$member-match` parameter FHIR berikut:

Parameter	Tipe	Diperlukan	Default	Deskripsi
MemberPatient	Pasien	Ya	—	Sumber daya pasien yang berisi informasi demografis agar anggota dicocokkan
CoverageToMatch	Cakupan	Ya	—	Sumber daya cakupan yang akan digunakan untuk pencocokan dengan catatan yang ada

Parameter	Tipe	Diperlukan	Default	Deskripsi
CoverageToLink	Cakupan	Tidak	—	Sumber daya cakupan yang akan ditautkan selama proses pencocokan
Persetujuan	Persetujuan	Tidak	—	Sumber daya persetujuan untuk tujuan otorisasi

Contoh

Permintaan POST dengan Parameter

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

Contoh Respons

Operasi mengembalikan sumber daya Parameter yang berisi hasil yang cocok:

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

Parameter Respons

Respons mencakup parameter berikut ketika kecocokan ditemukan:

Parameter	Jenis	Deskripsi
MemberIdentifier	Pengidentifikasi	Pengenal unik untuk anggota yang cocok
MemberId	Referensi	Referensi ke sumber daya Pasien
MatchAlgorithm	String	Jenis algoritma pencocokan yang digunakan (EXACT_MATCH, STRONG_MATCH, atau DEMOGRAPHIC_MATCH)
Detail pertandingan	String	Informasi terperinci tentang proses pencocokan
MatchedFields	String	Daftar bidang tertentu yang berhasil dicocokkan

Algoritma Pencocokan

`$member-match`API menggunakan pendekatan pencocokan multi-tier untuk memastikan identifikasi anggota yang akurat:

EXACT_MATCH

Menggunakan Patient Identifier dikombinasikan dengan Cakupan SubscriberId

Memberikan tingkat kepercayaan tertinggi untuk pencocokan anggota

STRONG_MATCH

Menggunakan Patient Identifier dengan informasi cakupan minimum

Menawarkan kepercayaan diri yang tinggi ketika kriteria kecocokan yang tepat tidak terpenuhi

DEMOGRAFIK_MATCH

Bergantung pada informasi demografis dasar

Digunakan saat pencocokan berbasis pengenal tidak dimungkinkan

Perilaku

`$member-match`Operasi:

- Menerima demografi pasien, detail cakupan, dan informasi persetujuan opsional sebagai masukan
- Mengembalikan pengenal anggota unik yang dapat digunakan untuk interaksi selanjutnya
- Menerapkan pencocokan multi-tier (tepat, kuat, demografis) untuk memastikan identifikasi anggota yang akurat di berbagai sistem perawatan kesehatan
- Menyimpan informasi persetujuan yang diberikan untuk tujuan otorisasi future
- Mendukung pertukaran payer-to-payer data yang aman sambil menjaga privasi pasien
- Memenuhi persyaratan CMS untuk pertukaran data perawatan kesehatan

Otorisasi

API menggunakan SMART pada protokol otorisasi FHIR dengan cakupan wajib berikut:

- `system/Patient.read`
- `system/Coverage.read`
- `system/Organization.read`(bersyarat)
- `system/Practitioner.read`(bersyarat)
- `system/PractitionerRole.read`(bersyarat)
- `system/Consent.write`(bersyarat)

Penanganan Kesalahan

Operasi menangani kondisi kesalahan berikut:

- `400 Bad Request`: `$member-match` Operasi tidak valid (permintaan yang tidak sesuai atau parameter yang diperlukan tidak ada)

- `422 Unprocessable Entity`: Tidak ada kecocokan atau beberapa kecocokan yang ditemukan

`$member-remove`operasi untuk HealthLake

`$member-remove`Operasi ini memungkinkan Anda untuk menghapus anggota dari Daftar Atribusi Anggota FHIR (sumber daya Grup) di. AWS HealthLake Operasi ini merupakan bagian dari Panduan Implementasi Atribusi DaVinci Anggota dan mendukung proses rekonsiliasi untuk mengelola atribusi anggota.

Prasyarat

- AWS HealthLake Datasore FHIR
- Izin IAM yang sesuai untuk operasi HealthLake
- Daftar Atribusi Anggota (Sumber daya grup) dalam draf atau status terbuka

Detail Operasi

Titik akhir

```
POST /Group/{id}/$member-remove
```

Jenis Konten

```
application/fhir+json
```

Parameter

Operasi menerima sumber daya Parameter FHIR dengan parameter opsional berikut:

Parameter	Kardinalitas	Tipe	Deskripsi
MemberID	0.. 1	Pengidentifikasi	Pengenal bisnis anggota yang akan dihapus
ProviderNPI	0.. 1	Pengidentifikasi	NPI dari penyedia yang dikaitkan
PatientReferensi	0.. 1	Referensi	Referensi langsung ke sumber daya Pasien

Parameter	Kardinalitas	Tipe	Deskripsi
Penyedia Referensi	0.. 1	Referensi	Referensi langsung ke sumber daya Penyedia (Praktisi, PractitionerRole, atau Organisasi)
CoverageReference	0.. 1	Referensi	Referensi ke sumber daya Cakupan

Kombinasi Parameter yang Didukung

Kombinasi parameter berikut didukung:

- `memberId`only - Menghapus semua atribusi untuk anggota yang ditentukan
- `memberId`+ `providerNpi` - Menghapus atribusi untuk kombinasi member-provider tertentu
- `patientReference`hanya - Menghapus semua atribusi untuk pasien yang ditentukan
- `patientReference`+ `providerReference` - Menghapus atribusi untuk kombinasi pasien-penyedia tertentu
- `patientReference`+ `providerReference` + `coverageReference` - Menghapus atribusi spesifik berdasarkan pasien, penyedia, dan cakupan

Contoh Permintaan

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "patientReference",
      "valueReference": {
        "reference": "Patient/12345"
      }
    },
    {
      "name": "providerReference",
      "valueReference": {
        "reference": "Practitioner/67890"
      }
    }
```

```
  ]
}
```

Respons

Respon yang Berhasil

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

Perilaku

Persyaratan Status

- Operasi hanya berfungsi pada daftar atribusi dengan status `draft` atau `open`
- Daftar dengan `final` status akan menolak operasi dengan kesalahan 422

Proses Penghapusan Anggota

- Daftar Status Draf: Anggota ditandai sebagai tidak aktif (`inactive: true`) dan `changeType` ekstensi mereka diperbarui ke `changed`
- Daftar Status Terbuka: Perilaku serupa dengan status draf

Daftar Status Akhir: Operasi ditolak

Validasi

Referensi divalidasi untuk memastikan mereka ada di datastore HealthLake

Jika pengenal dan referensi disediakan untuk jenis sumber daya yang sama, mereka harus sesuai dengan sumber daya yang sama

Kombinasi parameter divalidasi sesuai dengan pola yang didukung

Penanganan Kesalahan

Tanggapan Kesalahan Umum

Sumber Daya Tidak Ditemukan (404)

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "not-found",
      "details": {
        "text": "Patient with identifier 'http://example.org/fhir/identifiers|99999'
 not found in system"
      },
      "diagnostics": "Cannot remove member from attribution list. Verify patient
 identifier and try again.",
      "expression": ["memberId"]
    }
  ]
}
```

Status Akhir Daftar Atribusi (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"]
    }
  ]
}
```

Operasi Tidak Valid (400)

Dikembalikan ketika kombinasi parameter tidak valid atau cacat.

Ditemukan Beberapa Pertandingan (412)

Dikembalikan ketika parameter yang disediakan cocok dengan beberapa anggota dalam daftar atribusi.

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "processing",
      "diagnostics": "Multiple members found matching the criteria"
    }
  ]
}
```

Praktik Terbaik

- Gunakan Parameter Spesifik: Jika memungkinkan, gunakan kombinasi parameter yang paling spesifik untuk menghindari penghapusan yang tidak diinginkan
- Periksa Status Daftar: Verifikasi status daftar atribusi sebelum mencoba menghapus

- Menangani Kesalahan dengan Anggun: Menerapkan penanganan kesalahan yang tepat untuk semua kemungkinan kondisi kesalahan
- Validasi Referensi: Pastikan semua sumber daya yang direferensikan ada sebelum membuat permintaan

Menghapus Sumber Daya Kompartemen Pasien dengan **$purge**

AWS HealthLake mendukung `$purge` operasi, memungkinkan penghapusan permanen semua sumber daya dalam kompartemen pasien. Operasi ini sangat berguna ketika Anda perlu:

- Hapus semua data yang terkait dengan pasien
- Mematuhi permintaan penghapusan data pasien
- Kelola siklus hidup data pasien
- Jalankan pembersihan catatan pasien yang komprehensif

Penggunaan

`$purge`Operasi dapat dipanggil pada sumber daya Pasien:

```
POST [base]/Patient/[ID]/$purge?deleteAuditEvent=true
```

Parameter

Parameter	Tipe	Diperlukan	Default	Deskripsi
`deleteAuditEvent`	boolean	Tidak	false	Jika benar, menghapus peristiwa audit terkait
`_since`	string	Tidak	Waktu pembuatan Datastore	Saat dimasukkan, pilih waktu cutoff awal untuk menemukan sumber daya berdasarkan waktu LastModified mereka. Tidak dapat digunakan dengan awal atau akhir
`start`	string	Tidak	Waktu pembuatan Datastore	Saat dimasukkan, pilih waktu cutoff untuk menemukan sumber daya

Parameter	Tipe	Diperlukan	Default	Deskripsi
				berdasarkan waktu LastModified mereka. Dapat digunakan dengan akhir
end	string	Tidak	Waktu pengajuan Job	Saat dimasukkan, pilih waktu cutoff akhir untuk menemukan sumber daya berdasarkan waktu LastModified mereka

Contoh

Contoh Permintaan

```
POST [base]/Patient/example-patient/$purge?deleteAuditEvent=true
```

Contoh Respons

```
{
  "resourceType": "OperationOutcome",
  "id": "purge-job",
  "issue": [
    {
      "severity": "information",
      "code": "informational",
      "diagnostics": "Purge job started successfully. Job ID:
 12345678-1234-1234-1234-123456789012"
    }
  ]
}
```

Status Tugas

Untuk memeriksa status pekerjaan pembersihan:

```
GET [base]/$purge/[jobId]
```

Operasi mengembalikan informasi status pekerjaan:

```
{
      "datastoreId": "36622996b1fcecb7e12ee2ee085308d3",
      "jobId": "3dd1c7a5b6c0ef8c110f566eb87e2ef9",
      "status": "COMPLETED",
      "submittedTime": "2025-10-31T18:43:21.822Z"
    }
```

Perilaku

`$purge`Operasi:

1. Memproses secara asinkron untuk menangani banyak sumber daya
2. Menjaga transaksi ACID untuk integritas data
3. Menyediakan pelacakan status pekerjaan dengan jumlah penghapusan sumber daya
4. Secara permanen menghapus semua sumber daya di kompartemen pasien
5. Termasuk pencatatan audit komprehensif dari kegiatan penghapusan
6. Mendukung penghapusan selektif peristiwa audit

Pencatatan Audit

Log `$purge` operasi sebagai Mulai FHIRBulk DeleteJob dan Jelaskan FHIRBulk DeleteJob dengan informasi operasi terperinci.

Batasan

- Sumber daya yang dibersihkan tidak akan muncul di respons penelusuran
- Sumber daya yang sedang dibersihkan mungkin sementara tidak dapat diakses selama pemrosesan
- Semua sumber daya di kompartemen pasien dihapus secara permanen

`$questionnaire-package`Operasi FHIR untuk HealthLake

`$questionnaire-package`Operasi mengambil bundel komprehensif yang berisi Kuesioner FHIR dan semua dependensinya yang diperlukan untuk membuat dan memproses kuesioner. Operasi ini mengimplementasikan Panduan Implementasi Template dan Aturan Dokumentasi Da Vinci (DTR), memungkinkan rendering formulir dinamis untuk persyaratan dokumentasi dalam alur kerja perawatan kesehatan.

Cara kerjanya

- Permintaan: Anda mengirim parameter yang mengidentifikasi kuesioner yang diperlukan, bersama dengan cakupan dan konteks pesanan
- Ambil: HealthLake mengumpulkan Kuesioner dan semua dependensi (, Perpustakaan CQLValueSets, dll.)
- Package: Semua sumber daya dibundel bersama dalam format standar
- Menanggapi: Anda menerima paket lengkap yang siap untuk rendering dan pengumpulan data

Kasus penggunaan

- Dokumentasi Otorisasi Sebelumnya: Kumpulkan informasi klinis yang diperlukan untuk permintaan otorisasi sebelumnya
- Persyaratan Cakupan: Kumpulkan dokumentasi yang diperlukan untuk memenuhi persyaratan cakupan pembayar
- Clinical Data Exchange: Struktur data klinis untuk diserahkan kepada pembayar
- Formulir Dinamis: Render kuesioner dengan data pasien yang telah diisi sebelumnya dan logika bersyarat

Titik akhir API

```
POST /datastore/{datastoreId}/r4/Questionnaire/$questionnaire-package
Content-Type: application/fhir+json
```

Permintaan parameter

Parameter input

Badan permintaan harus berisi sumber daya Parameter FHIR dengan parameter berikut:

Parameter	Tipe	Kardinalitas	Deskripsi
`coverage`	Cakupan	1.. * (Diperlukan)	Sumber daya cakupan untuk menetapkan anggota dan cakupan untuk dokumentasi

Parameter	Tipe	Kardinalitas	Deskripsi
`questionnaire`	canonical	0.. *	URL kanonik untuk Kuesioner tertentu untuk dikembalikan (mungkin termasuk versi)
`order`	Sumber daya	0.. *	Pesan sumber daya (DeviceRequest, ServiceRequest, MedicationRequest, Encounter, Appointment) untuk menetapkan konteks
`changedSince`	dateTime	0.. 1	Jika ada, hanya sumber daya yang dikembalikan yang diubah setelah stempel waktu ini

Aturan validasi parameter

Setidaknya SATU dari berikut ini harus disediakan (selain yang diperlukan`coverage`):

- Satu atau lebih `questionnaire` kanonik URLs
- Satu atau lebih `order` sumber daya

Kombinasi Permintaan yang Valid:

- `coverage` + `questionnaire`
- `coverage` + `order`
- `coverage` + `questionnaire` + `order`

Contoh permintaan

```
POST /datastore/example-datastore/r4/Questionnaire/$questionnaire-package
Content-Type: application/fhir+json
Authorization: Bearer <your-token>

{
  "resourceType": "Parameters",
  "parameter": [
```

```
    {
      "name": "coverage",
      "resource": {
        "resourceType": "Coverage",
        "id": "example-coverage",
        "status": "active",
        "beneficiary": {
          "reference": "Patient/example-patient"
        },
        "payor": [{
          "reference": "Organization/example-payer"
        }],
        "class": [{
          "type": {
            "coding": [{
              "system": "http://terminology.hl7.org/CodeSystem/coverage-class",
              "code": "group"
            }]
          },
          "value": "12345"
        }]
      }
    },
    {
      "name": "questionnaire",
      "valueCanonical": "http://example.org/fhir/Questionnaire/home-oxygen-therapy|2.0"

    },
    {
      "name": "order",
      "resource": {
        "resourceType": "ServiceRequest",
        "id": "example-service-request",
        "status": "active",
        "intent": "order",
        "code": {
          "coding": [{
            "system": "http://www.ama-assn.org/go/cpt",
            "code": "94660",
            "display": "Continuous positive airway pressure ventilation (CPAP)"
          }]
        },
        "subject": {
          "reference": "Patient/example-patient"
```

```
        }
      }
    },
    {
      "name": "changedSince",
      "valueDateTime": "2024-01-01T00:00:00Z"
    }
  ]
}
```

Format respons

Respon sukses (200 OK)

Operasi mengembalikan sumber daya Parameter FHIR yang berisi satu atau lebih Package Bundle. Setiap Package Bundle meliputi:

Jenis Entri	Kardinalitas	Deskripsi
Kuesioner	1	Kuesioner yang akan diberikan
QuestionnaireResponse	0.. 1	Respons yang telah diisi sebelumnya atau sebagian selesai (jika ada)
Perpustakaan	0.. *	Perpustakaan CQL yang berisi pra-populasi dan logika bersyarat
ValueSet	0.. *	Diperluas ValueSets (untuk pilihan jawaban dengan <40 ekspansi)

Example Contoh tanggapan

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

Alur kerja operasi

Bagaimana HealthLake Memproses Permintaan Anda

Saat Anda menelepon `$questionnaire-package` HealthLake , lakukan langkah-langkah berikut:

1. Identifikasi Pasien & Pembayar: Ekstrak pasien dan organisasi asuransi dari `coverage` parameter Anda.
2. Temukan Kuesioner yang Tepat:
   - Dengan **`questionnaire`** parameter: Menggunakan URL kanonik yang Anda berikan
   - Dengan **`order`** parameter: Cocokkan kode pesanan (CPT/HCPCS/LOINC) dan pembayar untuk menemukan kuesioner yang sesuai

3. Kumpulkan Dependensi: Secara otomatis mengambil semua yang diperlukan untuk membuat kuesioner:
   - Perpustakaan CQL - Logika untuk pertanyaan pra-populasi dan bersyarat
   - ValueSets- Pilihan jawaban (secara otomatis diperluas jika <40 opsi)
   - QuestionnaireResponse- Setiap tanggapan yang sedang berlangsung atau selesai
4. Package Semuanya Bersama:
   - Bundel semua sumber daya (setiap sumber daya hanya disertakan sekali)
   - Filter berdasarkan `changedSince` stempel waktu jika disediakan
   - Menambahkan peringatan `Outcome` jika ada sumber daya yang hilang

Hasil: Paket lengkap dan mandiri yang siap untuk dirender.

Tanggapan kesalahan

400 Permintaan Buruk

Dikembalikan ketika validasi permintaan gagal.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "required",
    "details": {
      "text": "At least one of 'questionnaire' or 'order' must be provided along with
 'coverage'"
    }
  }]
}
```

424 Ketergantungan Gagal

Dikembalikan ketika sumber daya dependen tidak dapat diambil.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "warning",
    "code": "not-found",
    "details": {
```

```
      "text": "Referenced Library 'http://example.org/fhir/Library/missing-library'
 could not be retrieved"
    }
  }]
}
```

401 Tidak Sah

Dikembalikan ketika kredensi otentikasi hilang atau tidak valid.

403 Dilarang

Dikembalikan ketika pengguna yang diautentikasi tidak memiliki izin untuk mengakses sumber daya yang diminta.

406 Tidak Dapat Diterima

Dikembalikan ketika jenis konten yang diminta tidak dapat disediakan.

409 Konflik

Dikembalikan ketika ada versi atau konflik konkurensi.

410 Hilang

Dikembalikan ketika sumber daya yang diminta telah dihapus secara permanen.

429 Terlalu Banyak Permintaan

Dikembalikan ketika batas tarif terlampaui.

500 Kesalahan Server Internal

Dikembalikan ketika terjadi kesalahan server yang tidak terduga.

501 Tidak Diimplementasikan

Dikembalikan ketika operasi yang diminta belum dilaksanakan.

Aturan validasi

Validasi masukan

- `coverage`parameter diperlukan (1.. * kardinalitas)
- Setidaknya satu dari `questionnaire` atau `order` harus disediakan

- Semua sumber daya Cakupan harus sumber daya FHIR yang valid
- Semua sumber daya Pesanan harus sumber daya FHIR yang valid
- Canonical URLs harus diformat dengan benar
- `changedSince`harus berupa ISO 8601 DateTime yang valid

QuestionnaireResponse validasi

- `status`harus sesuai (`in-progress`,`completed`,`amended`)
- Struktur harus sesuai dengan Kuesioner yang direferensikan
- `basedOn`harus mereferensikan sumber daya Pesanan yang valid
- `subject`harus referensi sumber daya Pasien yang valid

Deduplikasi sumber daya

- Setiap sumber daya hanya muncul sekali dalam bundel
- Pengecualian: Versi berbeda dari sumber daya yang sama dapat disertakan
- Sumber daya diidentifikasi oleh URL dan versi kanonik mereka

Spesifikasi kinerja

Metrik	Spesifikasi
Batas Hitungan Sumber Daya	500 sumber daya per Bundel
Batas Ukuran Bundel	Maksimum 5 MB

Izin yang diperlukan

Untuk menggunakan `$questionnaire-package` operasi, pastikan peran IAM Anda memiliki:

- `healthlake:QuestionnairePackage`- Untuk memanggil operasi
- `healthlake:ReadResource`- Untuk mengambil Kuesioner dan sumber daya dependen
- `healthlake:SearchWithPost`- Untuk mencari QuestionnaireResponse dan sumber daya terkait

SMART pada Lingkup FHIR

Cakupan minimum yang diperlukan:

- 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`

Catatan implementasi penting

Strategi pengambilan sumber daya

Prioritas Identifikasi Kuesioner:

- URL kanonik (jika `questionnaire` parameter disediakan) - Prioritas tertinggi
- Analisis Pesanan (jika `order` parameter disediakan):
  - Cocokkan kode pesanan (CPT, HCPCS, LOINC) dengan kebijakan medis pembayar
  - Gunakan pembayar cakupan untuk memfilter kuesioner khusus pembayar
  - Pertimbangkan kode alasan untuk konteks tambahan

Resolusi ketergantungan

Perpustakaan CQL:

- Diperoleh melalui `cqf-library` ekstensi pada sumber Kuesioner
- Mengambil pustaka dependen secara rekursif melalui tipe `Library.relatedArtifact depends-on`
- Semua dependensi perpustakaan disertakan dalam paket

ValueSets:

- Secara otomatis diperluas jika mengandung kurang dari 40 konsep
- Lebih besar ValueSets disertakan tanpa ekspansi
- ValueSets direferensikan dalam Kuesioner dan sumber daya Perpustakaan disertakan

QuestionnaireResponse pra-populasi

Operasi dapat mengembalikan data yang telah QuestionnaireResponse diisi sebelumnya ketika:

- Respons yang sedang berlangsung atau selesai ditemukan
- Logika CQL di Perpustakaan terkait dapat mengekstrak data dari catatan pasien
- Tanggapan terkait dengan urutan dan cakupan yang relevan

Kriteria Pencarian untuk QuestionnaireResponse:

Parameter Pencarian	Jalan FHIR	Deskripsi
`based-on`	`QuestionnaireResponse.basedOn`	Tautan ke ServiceRequest atau CarePlan
`patient`	`QuestionnaireResponse.subject`	Pasien yang menjadi subjeknya
`questionnaire`	`QuestionnaireResponse.questionnaire`	Kuesioner yang dijawab

Pemfilteran sumber daya yang diubah

Ketika `changedSince` parameter disediakan:

- Hanya sumber daya yang dimodifikasi setelah stempel waktu yang ditentukan disertakan
- Jika tidak ada sumber daya yang berubah, kembali `200 OK` dengan paket kosong
- Berguna untuk pembaruan tambahan dan strategi caching
- Perbandingan stempel waktu menggunakan bidang sumber daya `meta.lastUpdated`

Beberapa paket bundel

Operasi dapat mengembalikan beberapa Package Bundle ketika:

- Beberapa kuesioner diminta melalui kanonik URLs

- Beberapa pesanan memerlukan kuesioner yang berbeda
- Versi berbeda dari kuesioner yang sama berlaku

Setiap Package Bundle mandiri dengan semua dependensi yang diperlukan.

Kasus penggunaan umum

Kasus Penggunaan 1: Dokumentasi Otorisasi Sebelumnya

Skenario: Penyedia perlu mengumpulkan dokumentasi untuk terapi oksigen rumah sebelum otorisasi.

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

Hasil: Mengembalikan paket dengan kuesioner terapi oksigen, diisi sebelumnya dengan tanda vital pasien dan kode diagnosis dari EHR.

Kasus Penggunaan 2: Ambil Versi Kuesioner Khusus

Skenario: Penyedia membutuhkan versi kuesioner tertentu untuk kepatuhan.

```
{
```

```
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "coverage",
      "resource": { /* Coverage resource */ }
    },
    {
      "name": "questionnaire",
      "valueCanonical": "http://example.org/fhir/Questionnaire/dme-request|3.1.0"
    }
  ]
}
```

Hasil: Mengembalikan persis versi 3.1.0 dari kuesioner permintaan DME dengan semua dependensi.

Kasus Penggunaan 3: Periksa Pembaruan

Skenario: Penyedia ingin memeriksa apakah ada sumber kuesioner yang telah diperbarui sejak pengambilan terakhir.

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

Hasil: Mengembalikan hanya sumber daya yang telah dimodifikasi setelah 1 Maret 2024, atau paket kosong jika tidak ada yang berubah.

Kasus Penggunaan 4: Beberapa Pesanan

Skenario: Penyedia mengirimkan beberapa permintaan layanan yang mungkin memerlukan kuesioner yang berbeda.

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "coverage",
      "resource": { /* Coverage resource */ }
    },
    {
      "name": "order",
      "resource": { /* ServiceRequest for imaging */ }
    },
    {
      "name": "order",
      "resource": { /* ServiceRequest for DME */ }
    }
  ]
}
```

Hasil: Mengembalikan beberapa Paket Paket, satu untuk setiap kuesioner yang berlaku.

Integrasi dengan Da Vinci Lainnya IGs

Penemuan Persyaratan Cakupan (CRD)

Integrasi Alur Kerja:

- Penyedia memesan layanan di EHR mereka
- CRD hook fire, memeriksa persyaratan cakupan
- Pembayar merespons yang menunjukkan dokumentasi diperlukan
- Panggilan penyedia `$questionnaire-package` untuk mengambil formulir dokumentasi
- Penyedia melengkapi kuesioner
- Dokumentasi disampaikan melalui PAS atau CDex

Prior Authorization Support (PAS)

Integrasi Alur Kerja:

- Gunakan `$questionnaire-package` untuk mengambil persyaratan dokumentasi
- Lengkapi QuestionnaireResponse dengan data klinis yang diperlukan
- Kirimkan otorisasi sebelumnya menggunakan `Claim/$submit` dengan yang sudah selesai QuestionnaireResponse
- Periksa status menggunakan `Claim/$inquire`

Data Exchange Klinis (CDex)

Integrasi Alur Kerja:

- Pembayar meminta dokumentasi tambahan untuk klaim
- Penyedia menggunakan `$questionnaire-package` untuk mengambil formulir pengumpulan data terstruktur
- Penyedia menyelesaikan QuestionnaireResponse
- Dokumentasi diserahkan kepada pembayar melalui alur kerja CDex lampiran

Panduan pemecahan masalah

Masalah: Tidak Ada Kuesioner yang Dikembalikan

Kemungkinan Penyebab:

- URL Canonical tidak cocok dengan Kuesioner apa pun di penyimpanan data
- Kode pesanan tidak dipetakan ke kuesioner apa pun dalam kebijakan medis pembayar
- Pembayar cakupan tidak memiliki kuesioner terkait

Solusi:

- Verifikasi URL kanonik benar dan Kuesioner ada
- Periksa apakah kode pesanan (CPT/HCPCS) ditentukan dengan benar
- Konfirmasikan bahwa organisasi pembayar memiliki kuesioner yang dikonfigurasi

Masalah: Dependensi Hilang dalam Package

Kemungkinan Penyebab:

- Perpustakaan yang direferensikan atau ValueSet tidak ada di penyimpanan data
- Referensi perpustakaan rusak atau salah
- ValueSet ekspansi gagal

Solusi:

- Periksa `Outcome` parameter untuk peringatan tentang sumber daya yang hilang
- Verifikasi semua sumber daya yang direferensikan ada di penyimpanan data Anda
- Pastikan ValueSet URLs benar dan dapat diselesaikan

Masalah: Paket Kosong dengan ChangedSince

Kemungkinan Penyebab:

- Ini adalah perilaku yang diharapkan - tidak ada sumber daya yang telah dimodifikasi sejak stempel waktu yang ditentukan

Solusi:

- Gunakan versi paket yang di-cache
- Hapus `changedSince` parameter untuk mengambil paket lengkap

Masalah: QuestionnaireResponse Tidak Terisi

Kemungkinan Penyebab:

- Tidak ada yang QuestionnaireResponse ditemukan
- Logika Perpustakaan CQL tidak dapat mengekstrak data yang diperlukan
- Data pasien hilang atau tidak lengkap

Solusi:

- Ini mungkin diharapkan - tidak semua kuesioner memiliki logika pra-populasi
- Periksa apakah data pasien ada di penyimpanan data
- Tinjau logika Perpustakaan CQL untuk persyaratan ekstraksi data

Praktik terbaik

1. Gunakan Canonical URLs dengan Versi

Selalu tentukan nomor versi saat meminta kuesioner tertentu:

```
{
  "name": "questionnaire",
  "valueCanonical": "http://example.org/fhir/Questionnaire/dme|2.1.0"
}
```

Mengapa: Memastikan konsistensi dan mencegah perubahan tak terduga saat kuesioner diperbarui.

2. Leverage BerubahSejak untuk Kinerja

Untuk kuesioner yang sering diakses, gunakan `changedSince` untuk meminimalkan transfer data:

```
{
  "name": "changedSince",
  "valueDateTime": "2024-03-10T15:30:00Z"
}
```

Mengapa: Mengurangi penggunaan latensi dan bandwidth dengan hanya mengambil sumber daya yang diperbarui.

3. Sertakan Informasi Cakupan Lengkap

Berikan detail cakupan yang komprehensif untuk memastikan pemilihan kuesioner yang benar:

```
{
  "name": "coverage",
  "resource": {
    "resourceType": "Coverage",
    "beneficiary": { "reference": "Patient/123" },
    "payor": [{ "reference": "Organization/payer-abc" }],
```

```
    "class": [{ /* Group/plan information */ }]
  }
}
```

Mengapa: Membantu HealthLake mengidentifikasi kuesioner dan persyaratan khusus pembayar.

Operasi terkait

- `Claim/$submit`- Kirim permintaan otorisasi sebelumnya dengan dokumentasi lengkap
- `Claim/$inquire`- Periksa status otorisasi sebelumnya yang dikirimkan
- `ValueSet/$expand`- Perluas ValueSets untuk pilihan jawaban

`$submit`Operasi FHIR untuk HealthLake

`$submit`Operasi ini memungkinkan Anda untuk secara elektronik mengirimkan permintaan otorisasi sebelumnya kepada pembayar untuk persetujuan. Operasi ini mengimplementasikan Panduan Implementasi Da Vinci Prior Authorization Support (PAS), menyediakan alur kerja berbasis FHIR standar untuk pengiriman otorisasi sebelumnya.

Cara kerjanya

- Kirim: Anda mengirim Bundel FHIR yang berisi permintaan otorisasi sebelumnya dan data klinis pendukung
- Validasi: HealthLake memvalidasi pengajuan terhadap persyaratan PAS
- Bertahan: Semua sumber daya disimpan di penyimpanan HealthLake data Anda
- Tanggapan: Anda menerima tanggapan langsung dengan status “antrian”
- Proses: Keputusan otorisasi diproses secara asinkron oleh pembayar

Titik akhir API

```
POST /datastore/{datastoreId}/r4/Claim/$submit
Content-Type: application/fhir+json
```

Struktur permintaan

Persyaratan bundel

Permintaan Anda harus berupa sumber daya FHIR Bundle dengan:

- Bundle.type: Harus `"collection"`
- Bundle.entry: Harus berisi tepat satu sumber daya Klaim dengan `use = "preauthorization"`
- Sumber Daya yang Direferensikan: Semua sumber daya yang direferensikan oleh Klaim harus disertakan dalam Bundel

Sumber daya yang dibutuhkan

Sumber daya	Kardinalitas	Profil	Deskripsi
Klaim	1	Klaim PAS	Permintaan otorisasi sebelumnya
Pasien	1	Pasien PAS	Informasi demografis pasien
Organisasi (Penanggung)	1	Penanggung PAS	Perusahaan asuransi
Organisasi (Penyedia)	1	Pemohon PAS	Penyedia layanan kesehatan mengirimkan permintaan
Cakupan	1 atau lebih	Cakupan PAS	Detail pertanggungan asuransi

Sumber daya opsional

Sumber daya	Kardinalitas	Profil	Deskripsi
Praktisi	0 atau lebih	Praktisi PAS	Praktisi kesehatan
PractitionerRole	0 atau lebih	PAS PractitionerRole	Peran praktisi
ServiceRequest	0 atau lebih	PAS ServiceRequest	Meminta layanan medis
DeviceRequest	0 atau lebih	PAS DeviceRequest	Alat kesehatan yang diminta
MedicationRequest	0 atau lebih	PAS MedicationRequest	Obat yang diminta

Sumber daya	Kardinalitas	Profil	Deskripsi
DocumentReference	0 atau lebih	PAS DocumentReference	Mendukung dokumentasi klinis

Contoh permintaan

```
POST /datastore/example-datastore/r4/Claim/$submit
Content-Type: application/fhir+json
Authorization: Bearer <your-token>

{
  "resourceType" : "Bundle",
  "id" : "MedicalServicesAuthorizationBundleExample",
  "meta" : {
    "profile" : ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-pas-
request-bundle"]
  },
  "identifier" : {
    "system" : "http://example.org/SUBMITTER_TRANSACTION_IDENTIFIER",
    "value" : "5269367"
  },
  "type" : "collection",
  "timestamp" : "2005-05-02T11:01:00+05:00",
  "entry" : [{
    "fullUrl" : "http://example.org/fhir/Claim/MedicalServicesAuthorizationExample",
    "resource" : {
      "resourceType" : "Claim",
      "id" : "MedicalServicesAuthorizationExample",
      "meta" : {
        "profile" : ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
claim"]
      },
      "identifier" : [{
        "system" : "http://example.org/PATIENT_EVENT_TRACE_NUMBER",
        "value" : "111099"
      }],
      "status" : "active",
      "type" : {
        "coding" : [{
          "system" : "http://terminology.hl7.org/CodeSystem/claim-type",
          "code" : "professional"
```

```
        }]
      },
      "use" : "preauthorization",
      "patient" : {
        "reference" : "Patient/SubscriberExample"
      },
      "created" : "2005-05-02T11:01:00+05:00",
      "insurer" : {
        "reference" : "Organization/InsurerExample"
      },
      "provider" : {
        "reference" : "Organization/UMOExample"
      },
      "priority" : {
        "coding" : [{
          "system" : "http://terminology.hl7.org/CodeSystem/processpriority",
          "code" : "normal"
        }]
      },
      "insurance" : [{
        "sequence" : 1,
        "focal" : true,
        "coverage" : {
          "reference" : "Coverage/InsuranceExample"
        }
      }],
      "item" : [{
        "extension" : [{
          "url" : "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/extension-
serviceItemRequestType",
          "valueCodeableConcept" : {
            "coding" : [{
              "system" : "https://codesystem.x12.org/005010/1525",
              "code" : "IN",
              "display" : "Initial Medical Services Reservation"
            }]
          }
        },
        {
          "url" : "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/extension-
certificationType",
          "valueCodeableConcept" : {
            "coding" : [{
              "system" : "https://codesystem.x12.org/005010/1322",
```

```
              "code" : "I",
              "display" : "Initial"
            }]
          }
        },
        {
          "url" : "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/extension-
authorizationNumber",
          "valueString" : "1122344"
        },
        {
          "url" : "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/extension-
administrationReferenceNumber",
          "valueString" : "33441122"
        }],
        "sequence" : 1,
        "category" : {
          "coding" : [{
            "system" : "https://codesystem.x12.org/005010/1365",
            "code" : "1",
            "display" : "Medical Care"
          }]
        },
        "productOrService" : {
          "coding" : [{
            "system" : "http://www.cms.gov/Medicare/Coding/HCPCSReleaseCodeSets",
            "code" : "99212",
            "display" : "Established Office Visit"
          }]
        },
        "servicedDate" : "2005-05-10",
        "locationCodeableConcept" : {
          "coding" : [{
            "system" : "https://www.cms.gov/Medicare/Coding/place-of-service-codes/
Place_of_Service_Code_Set",
            "code" : "11"
          }]
        }
      }]
    }
  },
  {
    "fullUrl" : "http://example.org/fhir/Organization/UMOExample",
    "resource" : {
```

```
      "resourceType" : "Organization",
      "id" : "UMOExample",
      "meta" : {
        "profile" : ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
requestor"]
      },
      "identifier" : [{
        "system" : "http://hl7.org/fhir/sid/us-npi",
        "value" : "8189991234"
      }],
      "active" : true,
      "type" : [{
        "coding" : [{
          "system" : "https://codesystem.x12.org/005010/98",
          "code" : "X3"
        }]
      }],
      "name" : "DR. JOE SMITH CORPORATION",
      "address" : [{
        "line" : ["111 1ST STREET"],
        "city" : "SAN DIEGO",
        "state" : "CA",
        "postalCode" : "92101",
        "country" : "US"
      }]
    }
  },
  {
    "fullUrl" : "http://example.org/fhir/Organization/InsurerExample",
    "resource" : {
      "resourceType" : "Organization",
      "id" : "InsurerExample",
      "meta" : {
        "profile" : ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
insurer"]
      },
      "identifier" : [{
        "system" : "http://hl7.org/fhir/sid/us-npi",
        "value" : "1234567893"
      }],
      "active" : true,
      "type" : [{
        "coding" : [{
          "system" : "https://codesystem.x12.org/005010/98",
```

```
          "code" : "PR"
        }]
      }],
      "name" : "MARYLAND CAPITAL INSURANCE COMPANY"
    }
  },
  {
    "fullUrl" : "http://example.org/fhir/Coverage/InsuranceExample",
    "resource" : {
      "resourceType" : "Coverage",
      "id" : "InsuranceExample",
      "meta" : {
        "profile" : ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
coverage"]
      },
      "status" : "active",
      "subscriberId" : "1122334455",
      "beneficiary" : {
        "reference" : "Patient/SubscriberExample"
      },
      "relationship" : {
        "coding" : [{
          "system" : "http://terminology.hl7.org/CodeSystem/subscriber-relationship",
          "code" : "self"
        },
        {
          "system" : "https://codesystem.x12.org/005010/1069",
          "code" : "18"
        }]
      },
      "payor" : [{
        "reference" : "Organization/InsurerExample"
      }]
    }
  },
  {
    "fullUrl" : "http://example.org/fhir/Patient/SubscriberExample",
    "resource" : {
      "resourceType" : "Patient",
      "id" : "SubscriberExample",
      "meta" : {
        "profile" : ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
subscriber"]
      },
```

```
      "extension" : [{
        "url" : "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/extension-
militaryStatus",
        "valueCodeableConcept" : {
          "coding" : [{
            "system" : "https://codesystem.x12.org/005010/584",
            "code" : "RU"
          }]
        }
      }],
      "identifier" : [{
        "type" : {
          "coding" : [{
            "system" : "http://terminology.hl7.org/CodeSystem/v2-0203",
            "code" : "MB"
          }]
        },
        "system" : "http://example.org/MIN",
        "value" : "12345678901"
      }],
      "name" : [{
        "family" : "SMITH",
        "given" : ["JOE"]
      }],
      "gender" : "male"
    }
  }]
}
```

Format respons

Respon sukses (200 OK)

Anda akan menerima PAS Response Bundle yang berisi:

- ClaimResponsedengan `outcome: "queued"` dan `status: "active"`
- Semua sumber daya asli dari permintaan Anda
- Tanda terima konfirmasi stempel waktu

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

Tanggapan kesalahan

400 Permintaan Buruk

Dikembalikan ketika format permintaan tidak valid atau cacat.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "invalid",
    "diagnostics": "The provided payload was invalid and could not be parsed
 correctly."
  }]
}
```

412 Prasyarat Gagal

Dikembalikan ketika permintaan otorisasi sebelumnya yang sama telah dikirimkan (duplikat pengiriman terdeteksi).

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "processing",
    "diagnostics": "PreAuth Claim already exists"
  }]
}
```

Idempotensi

`$submit`Operasi ini idempoten. Mengirimkan permintaan yang sama beberapa kali tidak akan membuat duplikat permintaan otorisasi sebelumnya. Sebagai gantinya, Anda akan menerima kesalahan 412 yang mengarahkan Anda untuk menggunakan `$inquire` untuk memeriksa status pengiriman asli Anda.

422 Entitas yang Tidak Dapat Diproses

Dikembalikan ketika validasi FHIR gagal.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "required",
    "diagnostics": "Bundle contains more than one preauthorization claim"
  }]
}
```

Aturan validasi

HealthLake melakukan validasi komprehensif pada kiriman Anda:

Validasi bundel

- Harus sesuai dengan profil PAS Request Bundle

- `Bundle.type`harus `"collection"`
- Dapat berisi beberapa sumber Klaim
- Namun, harus berisi persis satu sumber Klaim dengan penggunaan pra-otorisasi
  - Dan sumber daya Klaim ini harus menjadi entri pertama dalam bundel
- Semua sumber daya yang direferensikan harus disertakan dalam Bundel

Validasi klaim

- Harus sesuai dengan profil Klaim PAS
- `Claim.use`harus `"preauthorization"`
- Bidang yang diperlukan: `patient`,`insurer`,`provider`,`created`, `priority`
- Pengidentifikasi bisnis harus ada dan valid

Validasi sumber daya

- Semua sumber daya harus sesuai dengan profil PAS masing-masing
- Sumber daya pendukung yang diperlukan harus ada (Pasien, Cakupan, Organisasi)
- Referensi silang harus valid dan dapat diselesaikan dalam Bundel

Spesifikasi kinerja

Metrik	Spesifikasi
Batas Ukuran Bundel	Maksimum 5 MB
Batas Hitungan Sumber Daya	500 sumber daya per Bundel

Izin yang diperlukan

Untuk menggunakan `$submit` operasi, seseorang dapat menggunakan AWS Sigv4 atau SMART di FHIR:

- Pastikan peran IAM Anda memiliki: `healthlake:SubmitPreAuthClaim` - Untuk memanggil operasi

SMART pada Lingkup FHIR

Cakupan minimum yang diperlukan:

- SMART v1: `user/Claim.write & <all_resourceTypes_in_Bundle>.write`
- SMART v2: `user/Claim.c & <all_resourceTypes_in_Bundle>.c or system/*.*`

Catatan implementasi penting

Kegigihan sumber daya

- Semua entri Bundle dipertahankan sebagai sumber daya FHIR individual di penyimpanan data Anda
- Disediakan pelanggan dipertahankan saat IDs disediakan
- Riwayat versi dipertahankan untuk tujuan audit
- Deteksi duplikat mencegah konflik sumber daya

Perilaku pemrosesan

- Setiap pengiriman yang valid menghasilkan persis satu ClaimResponse dengan `"queued"` hasil
- Kiriman tidak valid mengembalikan kode status 400 atau 422 dengan informasi kesalahan terperinci
- Kesalahan sistem mengembalikan kode status 5xx yang sesuai
- Semua kiriman yang berhasil mengembalikan 200 status dengan pended ClaimResponse

Persyaratan bundel

- `Bundle.entry.fullUrl`nilai harus berupa REST URLs atau `"urn:uuid:[guid]"` format
- Semua GUIDs harus unik di seluruh kiriman (kecuali untuk contoh sumber daya yang sama)
- Sumber daya yang direferensikan harus ada di dalam Bundel atau dapat diselesaikan

Operasi terkait

- `Claim/$inquire`- Kueri status permintaan otorisasi sebelumnya yang diajukan
- `Patient/$everything`- Ambil data pasien yang komprehensif untuk konteks otorisasi sebelumnya

Memvalidasi Sumber Daya FHIR dengan **$validate**

AWS HealthLake sekarang mendukung `$validate` operasi untuk sumber daya FHIR, memungkinkan Anda untuk memvalidasi sumber daya terhadap spesifikasi FHIR dan memeriksa kesesuaiannya dengan profil tertentu atau definisi sumber daya dasar tanpa melakukan operasi penyimpanan apa pun. Operasi ini sangat berguna ketika Anda perlu:

- Validasi persyaratan kepatuhan FHIR CMS
- Uji sumber daya sebelum menggunakannya dalam produksi
- Berikan umpan balik validasi waktu nyata saat pengguna mengedit data klinis
- Mengurangi pengiriman data yang tidak valid untuk membuat dan memperbarui APIs

Penggunaan

`$validate`Operasi dapat dipanggil pada sumber daya FHIR menggunakan metode POST:

Operasi yang Didukung

```
POST [base]/[type]/[id]/$validate
POST [base]/[type]/$validate
```

Muatan yang Didukung

Parameter sumber daya

HealthLake mendukung `$validate` parameter FHIR berikut:

Parameter	Tipe	Diperlukan	Deskripsi
`resource`	Sumber daya	Ya	Sumber daya yang akan divalidasi
`profile`	canonical	Tidak	URL kanonik profil untuk memvalidasi
`mode`	code	Tidak	Mode validasi:`create`, atau `update`

Sumber daya langsung dengan Parameter Kueri

Parameter	Tipe	Diperlukan	Deskripsi
`profile`	canonical	Tidak	URL kanonik profil untuk memvalidasi
`mode`	code	Tidak	Mode validasi:`create`, atau `update`

Contoh

POST Permintaan Sumber Daya dengan ID dan Parameter payload

```
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"
    }
  ]
```

```
}
```

POST Permintaan untuk Jenis Sumber Daya dan Parameter payload

```
POST [base]/Patient/$validate
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "resource",
      "resource": {
        "resourceType": "Patient",
        "name": [
          {
            "family": "Doe",
            "given": ["Jane"]
          }
        ],
        "gender": "female",
        "birthDate": "1985-05-15"
      }
    },
    {
      "name": "profile",
      "valueCanonical": "http://hl7.org/fhir/us/core/StructureDefinition/us-core-
patient"
    },
    {
      "name": "mode",
      "valueString": "update"
    }
  ]
}
```

POST Permintaan Sumber Daya dengan ID dan muatan sumber daya langsung

```
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"
}
```

POST Permintaan untuk Jenis Sumber Daya dan muatan sumber daya langsung

```
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"
}
```

Contoh Respons

Operasi mengembalikan OperationOutcome sumber daya dengan hasil validasi:

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "information",
      "code": "informational",
```

```
      "diagnostics": "Validation successful"
    }
  ]
}
```

Sampel Respon dengan Kesalahan Validasi

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

Perilaku

`$validate`Operasi:

1. Memvalidasi sumber daya terhadap spesifikasi FHIR dan definisi sumber daya dasar
2. Memeriksa kesesuaian dengan profil yang ditentukan saat parameter disediakan `profile`
3. Memvalidasi berdasarkan mode yang ditentukan (`create`atau`update`)

4. Mengembalikan hasil validasi rinci termasuk kesalahan, peringatan, dan pesan informasi
5. Tidak melakukan operasi penyimpanan apa pun - hanya validasi
6. Mengembalikan HTTP 200 OK ketika validasi dapat dilakukan, terlepas dari apakah masalah validasi ditemukan

Mode Validasi

- create: Memvalidasi sumber daya seolah-olah sedang dibuat (sumber daya baru)
- update: Memvalidasi sumber daya seolah-olah sedang diperbarui (sumber daya yang ada)

Penanganan Kesalahan

Operasi kembali:

- 200 OK: Validasi berhasil dilakukan (terlepas dari hasil validasi)
- 400 Permintaan Buruk: Format atau parameter permintaan tidak valid
- 404 Tidak Ditemukan: Jenis sumber daya atau profil tidak ditemukan

Untuk informasi selengkapnya tentang spesifikasi `$validate` operasi, lihat dokumentasi FHIR R4 Resource `$validate`.

Referensi kepatuhan untuk AWS HealthLake

AWS HealthLake menyediakan fitur yang dirancang untuk membantu Anda melacak dan melaporkan penggunaan API sesuai dengan persyaratan interoperabilitas CMS (Pusat Layanan Medicare & Medicaid). Fitur-fitur ini memungkinkan Anda untuk mengkategorikan transaksi API berdasarkan kategori yang diamanatkan CMS dan secara otomatis menangkap metrik penggunaan untuk tujuan pelaporan kepatuhan.

⚠ Memahami Tanggung Jawab Kepatuhan Anda

Menggunakan AWS HealthLake dan titik akhir interoperabilitas CMS tidak cukup dengan sendirinya untuk mencapai kepatuhan CMS. Anda bertanggung jawab untuk:

- Memetakan alur kerja API Anda dengan benar ke titik akhir kategori CMS yang sesuai berdasarkan kasus penggunaan spesifik dan kewajiban peraturan Anda

- Menerapkan kontrol otentikasi dan otorisasi yang tepat yang memenuhi persyaratan CMS
- Memastikan sumber daya dan pertukaran data FHIR Anda mematuhi peraturan dan panduan implementasi CMS yang berlaku
- Mengkonfigurasi dan memantau CloudWatch metrik untuk mendukung kebutuhan pelaporan kepatuhan Anda
- Memahami aturan CMS mana yang berlaku untuk organisasi Anda dan menerapkan kontrol teknis dan operasional yang sesuai

AWS HealthLake menyediakan infrastruktur dan perkakas untuk mendukung upaya kepatuhan Anda, tetapi Anda harus menggunakan fitur ini dengan tepat berdasarkan persyaratan peraturan khusus Anda. Cukup merutekan panggilan API melalui titik akhir ini tidak secara otomatis membuat aplikasi Anda sesuai dengan peraturan CMS.

Topik

- Fitur kepatuhan CMS

Fitur kepatuhan CMS

AWS HealthLake menyediakan fitur untuk membantu Anda memenuhi persyaratan interoperabilitas dan kepatuhan CMS (Pusat Layanan Medicare & Medicaid). Fitur-fitur ini memungkinkan Anda melacak penggunaan API menurut kategori CMS dan selanjutnya melaporkan metrik penggunaan untuk tujuan kepatuhan.

Topik

- Titik akhir interoperabilitas CMS
- CloudWatch Metrik yang disempurnakan untuk kepatuhan CMS

Titik akhir interoperabilitas CMS

Ikhtisar

HealthLake menyediakan empat titik akhir interoperabilitas CMS yang sesuai dengan kategori API yang diamanatkan CMS. URL dasar yang mendasari penyimpanan HealthLake data Anda tidak berubah. Titik akhir ini hanya menyediakan cara untuk mengkategorikan dan melacak panggilan API Anda untuk tujuan pelaporan CMS.

Tujuan

Tujuan utama dari titik akhir interoperabilitas ini adalah untuk memungkinkan pelanggan untuk:

- Melacak transaksi API dengan mudah berdasarkan kategori CMS
- Secara otomatis melaporkan metrik penggunaan untuk kepatuhan CMS
- Pertahankan alur kerja FHIR yang ada dengan perubahan minimal

Semua panggilan API berfungsi secara identik apakah Anda menggunakan titik akhir interoperabilitas atau titik akhir FHIR standar—satu-satunya perbedaan adalah bagaimana transaksi dikategorikan dalam metrik. CloudWatch

Titik akhir interoperabilitas CMS yang didukung

Kategori CMS	Titik Akhir Interoperabilitas	Contoh Penggunaan
Akses Pasien	`/patientaccess/v2/r4`	`baseURL/patientaccess/v2/r4/Patient/123`
Akses Penyedia	`/provideraccess/v2/r4`	`baseURL/provideraccess/v2/r4/Observation?patient=123`
Pembayar ke Pembayar	`/payertopayerdx/v2/r4`	`baseURL/payertopayerdx/v2/r4/Practitioner/456`
Layanan Auth Sebelumnya	`/priorauthservice/v2/r4`	`baseURL/priorauthservice/v2/r4/ExplanationOfBenefit?patient=789`

Bagaimana titik akhir interoperabilitas bekerja

Panggilan HealthLake API Standar:

```
baseURL/resourceType/[id]
baseURL/resourceType?[parameters]
```

Dengan Titik Akhir Interoperabilitas CMS:

```
baseURL/interoperability-endpoint/resourceType/[id]
baseURL/interoperability-endpoint/resourceType?[parameters]
```

Jalur titik akhir interoperabilitas hanya disisipkan antara URL dasar Anda dan jenis sumber daya. Segala sesuatu setelah jalur titik akhir interoperabilitas tetap sama persis dengan panggilan API Anda saat ini.

Contoh penggunaan

Contoh 1: API Akses Pasien

Panggilan API saat ini (masih berfungsi):

```
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"
```

Dengan titik akhir interoperabilitas Akses Pasien (untuk pelacakan CMS):

```
curl -X GET \
  https://healthlake.us-east-1.amazonaws.com/datastore/abc123/patientaccess/v2/r4/
Patient/123 \
  -H "Authorization: Bearer <token>" \
  -H "Content-Type: application/fhir+json"
```

Poin Utama:

- URL dasar tetap: `https://healthlake.us-east-1.amazonaws.com/datastore/abc123`
- Titik akhir interoperabilitas dimasukkan: `/patientaccess/v2/r4`
- Jalur sumber daya tidak berubah: `/Patient/123`
- Kedua panggilan mengembalikan respons yang identik
- Panggilan titik akhir interoperabilitas secara otomatis dilacak di bawah `URIType=patient-access` CloudWatch
- Operasi POST, PUT, PATCH, DELETE bekerja secara identik.
- Badan permintaan tetap tidak berubah.

Referensi terjemahan titik akhir

Titik Akhir Interoperabilitas	Menerjemahkan Ke	Kategori CMS
`baseURL/patientaccess/v2/r4/Patient`	`baseURL/r4/Patient`	Akses Pasien
`baseURL/provideraccess/v2/r4/Observation`	`baseURL/r4/Observation`	Akses Penyedia
`baseURL/payertopayerdx/v2/r4/Practitioner/456`	`baseURL/r4/Practitioner/456`	Pembayar ke Pembayar Data Exchange
`baseURL/priorauthservice/v2/r4/ExplanationOfBenefit?patient=789`	`baseURL/r4/ExplanationOfBenefit?patient=789`	Otorisasi Sebelumnya

Catatan penting

- Tidak Ada Perbedaan Fungsional: Titik akhir interoperabilitas dan titik akhir FHIR standar mengembalikan respons yang identik dan mendukung operasi yang identik
- URL Dasar Tidak Berubah: Titik akhir penyimpanan HealthLake data Anda tetap sama
- Integrasi Sederhana: Masukkan jalur titik akhir interoperabilitas antara URL dasar dan tipe sumber daya Anda
- Pelacakan Otomatis: CloudWatch metrik secara otomatis mengkategorikan panggilan berdasarkan titik akhir interoperabilitas yang digunakan
- Kompatibel Mundur: Panggilan API yang ada tanpa titik akhir interoperabilitas terus bekerja secara normal

CloudWatch Metrik yang disempurnakan untuk kepatuhan CMS

Ikhtisar

Saat Anda menggunakan titik akhir interoperabilitas CMS, HealthLake secara otomatis memancarkan CloudWatch metrik yang disempurnakan dengan dimensi tambahan untuk mendukung persyaratan

pelaporan CMS. Metrik ini melacak penggunaan API berdasarkan identitas pemanggil, aplikasi, dan jenis URI khusus CMS tanpa memerlukan konfigurasi tambahan.

Cara kerjanya

Saat Anda melakukan panggilan API menggunakan titik akhir interoperabilitas:

```
# 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.)
```

Tidak diperlukan kode atau konfigurasi tambahan. Cukup gunakan titik akhir interoperabilitas, dan metrik yang disempurnakan akan ditangkap secara otomatis.

> **Note**
>
> Untuk penyimpanan data non-SMART di FHIR, `URIType` dimensi masih ditangkap, memungkinkan Anda melacak penggunaan API menurut kategori CMS. `ClientId`Dimensi Sub dan hanya tersedia saat menggunakan SMART pada otentikasi FHIR dengan token pembawa yang berisi klaim ini.

Dimensi metrik baru

Selain dimensi yang ada (`DatastoreId`,,`Operation`)`DatastoreType`, dimensi berikut secara otomatis ditambahkan saat menggunakan titik akhir interoperabilitas:

Dimensi	Deskripsi	Nilai contoh	Sumber
URIType	Kategori kepatuhan CMS	`patient-access` , `provider-access` , `payer-to-payer` , `prior-authorization`	Secara otomatis ditentukan dari jalur titik akhir interoperabilitas

Dimensi	Deskripsi	Nilai contoh	Sumber
Sub	Identitas penelepon	Pengidentifikasi pengguna/entitas	Diekstrak dari klaim token pembawa `sub`
ClientId	Pengidentifikasi aplikasi	`portal_app` , `ehr_system`	Diekstrak dari klaim token pembawa `client_id`

Metrik yang tersedia

Semua HealthLake metrik yang ada sekarang menyertakan dimensi tambahan saat menggunakan titik akhir interoperabilitas:

- CallCount- Jumlah total panggilan API
- Latensi - Waktu respons API dalam milidetik
- UserErrors- Hitungan kesalahan klien 4xx
- SystemErrors- Hitungan kesalahan server 5xx
- Throttles - Hitungan permintaan yang dibatasi
- SuccessfulRequests- Hitungan panggilan API yang berhasil

Kueri metrik di CloudWatch

CloudWatch Contoh kueri wawasan

Kueri semua panggilan API Akses Pasien berdasarkan aplikasi:

```
SELECT SUM(CallCount)
FROM "AWS/HealthLake"
WHERE DatastoreId = '75c1cf9b0d71cd38fec8f7fb317c4c1a'
    AND URIType = 'patient-access'
GROUP BY ClientId
```

Referensi Support untuk AWS HealthLake

Bahan referensi pendukung berikut tersedia untuk AWS HealthLake.

> **Note**
>
> Semua HealthLake tindakan asli dan tipe data dijelaskan dalam referensi terpisah. Untuk informasi lebih lanjut, lihat Referensi API AWS HealthLake .

Topik

- AWS HealthLake titik akhir dan kuota
- Synthea tipe data yang dimuat sebelumnya untuk HealthLake
- AWS HealthLake proyek sampel
- Pemecahan masalah AWS HealthLake
- Menggunakan HealthLake dengan AWS SDK

AWS HealthLake titik akhir dan kuota

Topik berikut berisi informasi tentang titik akhir AWS HealthLake layanan dan kuota.

Topik

- Titik akhir layanan
- Kuota layanan

Titik akhir layanan

Endpoint layanan adalah URL yang mengidentifikasi host dan port sebagai titik masuk untuk layanan web. Setiap permintaan layanan web berisi titik akhir. Sebagian besar AWS layanan menyediakan titik akhir untuk Wilayah tertentu untuk memungkinkan konektivitas yang lebih cepat. Tabel berikut mencantumkan titik akhir layanan untuk AWS HealthLake.

Nama Wilayah	Wilayah	Titik Akhir	Protokol	
AS Timur (Ohio)	us–east-2	healthlake.us-east-2.amazonaws.com	HTTPS	
		healthlake-fips.us-east-2.amazonaws.com	HTTPS	

Nama Wilayah	Wilayah	Titik Akhir	Protokol
AS Timur (Virginia Utara)	us-east-1	healthlake.us-east-1.amazonaws.com healthlake-fips.us-east-1.amazonaws.com	HTTPS HTTPS
US West (Oregon)	us-west-2	healthlake.us-west-2.amazonaws.com healthlake-fips.us-west-2.amazonaws.com	HTTPS HTTPS
Asia Pasifik (Mumbai)	ap-south-1	healthlake.ap-south-1.amazonaws.com	HTTPS
Asia Pacific (Sydney)	ap-southeast-2	healthlake.ap-southeast-2.amazonaws.com	HTTPS
Canada (Central)	ca-central-1	healthlake.ca-central-1.amazonaws.com	HTTPS
Eropa (Irlandia)	eu-west-1	healthlake.eu-west-1.amazonaws.com	HTTPS
Europe (London)	eu-west-2	healthlake.eu-west-2.amazonaws.com	HTTPS

Kuota layanan

Kuota layanan didefinisikan sebagai nilai maksimum untuk sumber daya, tindakan, dan item di AWS akun Anda.

> **Note**
>
> Untuk kuota yang dapat disesuaikan, Anda dapat meminta peningkatan kuota menggunakan konsol Service Quotas. Untuk informasi selengkapnya, lihat Meminta peningkatan kuota di Panduan Pengguna Service Quotas.

Tingkat Sync Write API meningkat secara proporsional dengan ukuran payload, dengan setiap kenaikan 1KB menghabiskan kapasitas tambahan (misalnya, payload 4KB menggunakan kapasitas tulis 4x). Mengatur `x-amz-fhir-history-consistency-level` header opsional untuk `strong` menggandakan konsumsi kapasitas tulis per sumber daya. Sumber daya dalam Bundel mengikuti read/write batas standar berdasarkan ukuran muatan 1 KB. Transaksi jenis bundel mengkonsumsi dua kali kapasitas tulis dibandingkan dengan jenis batch, yang berarti batch Bundle dapat memproses dua kali lebih banyak sumber daya per detik sebagai bundel transaksi.

Tabel berikut mencantumkan kuota default untuk AWS HealthLake.

Nama	Default	Dapat disesuaikan	Deskripsi
Jumlah Langganan aktif per akun	Setiap Wilayah yang didukung: 100	Ya	Jumlah maksimum sumber daya Berlangganan aktif per akun.
Jumlah sumber daya Berlangganan aktif per datastore	Setiap Wilayah yang didukung: 50	Ya	Jumlah maksimum sumber daya Berlangganan aktif per datastore.
Jumlah aktif SubscriptionTopic per akun	Setiap Wilayah yang didukung: 100	Ya	Jumlah maksimum SubscriptionTopic sumber daya aktif per akun.
Jumlah SubscriptionTopic sumber daya aktif per datastore	Setiap Wilayah yang didukung: 50	Ya	Jumlah maksimum SubscriptionTopic sumber daya aktif per datastore.
Jumlah karakter dalam catatan medis	Setiap Wilayah yang didukung: 10.000	Tidak	Jumlah maksimum karakter dalam catatan medis individu dalam jenis DocumentReference Sumber Daya (POST/PUT permintaan).

Nama	Default	Dapat disesuaikan	Deskripsi
Jumlah pekerjaan bersamaan StartFHIRExportJob	Setiap Wilayah yang didukung: 1	Tidak	StartFHIRExportJob Pekerjaan bersamaan maksimum.
Jumlah pekerjaan bersamaan StartFHIRImportJob	Setiap Wilayah yang didukung: 1	Tidak	StartFHIRImportJob Pekerjaan bersamaan maksimum.
Jumlah penyimpanan data per akun	Setiap Wilayah yang didukung: 10	Ya	Jumlah maksimum default penyimpanan data aktif per akun.
Jumlah file dalam StartFHIRImportJob	Setiap Wilayah yang didukung: 1.000.000	Ya	Jumlah maksimum file dalam file StartFHIRImportJob.
Jumlah sumber daya per Bundel	Setiap Wilayah yang didukung: 500	Tidak	Jumlah maksimum sumber daya yang diizinkan dalam permintaan Bundel.
Tingkat CancelFHIRExportJob permintaan menggunakan DELETE per akun	Setiap Wilayah yang didukung: 1	Tidak	Jumlah maksimum CancelFHIRExportJob permintaan menggunakan DELETE yang dapat Anda buat per detik per akun.
Tingkat permintaan createFhirDataStore per akun	Setiap Wilayah yang didukung: 1	Tidak	Jumlah maksimum permintaan createFhirDataStore yang dapat Anda buat per menit per akun.

Nama	Default	Dapat disesuaikan	Deskripsi
Tingkat permintaan DeleteFhirDataStore per akun	Setiap Wilayah yang didukung: 1	Tidak	Jumlah maksimum permintaan DeleteFhirDataStore yang dapat Anda buat per menit per akun.
Tarif DescribeFHIRBulkDeleteJob permintaan per akun	Setiap Wilayah yang didukung: 10	Ya	Jumlah maksimum DescribeFHIRBulkDeleteJob permintaan yang dapat Anda buat per detik per akun.
Tingkat DescribeFHIRBulkDeleteJob permintaan per penyimpanan data	Setiap Wilayah yang didukung: 10	Ya	Jumlah maksimum DescribeFHIRBulkDeleteJob permintaan yang dapat Anda buat per detik per penyimpanan data.
Tingkat permintaan DescribefHirDataStore per akun	Setiap Wilayah yang didukung: 10	Tidak	Jumlah maksimum permintaan DescribefHirDataStore yang dapat Anda buat per detik per akun.
Tarif DescribeFHIRExportJob permintaan per akun	Setiap Wilayah yang didukung: 10	Tidak	Jumlah maksimum DescribeFHIRExportJob permintaan yang dapat Anda buat per detik per akun.

Nama	Default	Dapat disesuaikan	Deskripsi
Tarif DescribeFHIRExportJob permintaan menggunakan GET per akun	Setiap Wilayah yang didukung: 10	Tidak	Jumlah maksimum DescribeFHIRExportJob permintaan menggunakan GET yang dapat Anda buat per detik per akun.
Tarif DescribeFHIRImportJob permintaan per akun	Setiap Wilayah yang didukung: 10	Tidak	Jumlah maksimum DescribeFHIRImportJob permintaan yang dapat Anda buat per detik per akun.
Tarif permintaan Discovery per akun	Setiap Wilayah yang didukung: 10	Tidak	Jumlah maksimum permintaan Discovery yang dapat Anda buat per menit per akun.
Tarif permintaan GET per akun	Setiap Wilayah yang didukung: 6.000	Ya	Jumlah maksimum permintaan GET yang dapat Anda buat per detik per akun.
Tarif permintaan GET per penyimpanan data	Setiap Wilayah yang didukung: 3.000	Ya	Jumlah maksimum permintaan GET yang dapat Anda buat per detik per penyimpanan data. Penyimpanan data yang dibuat sebelum 8/21 /2023 akan dibatasi hingga 100 permintaan per detik.

Nama	Default	Dapat disesu an	Deskripsi
Tarif GetCapabilities permintaan per akun	Setiap Wilayah yang didukung: 10	Tidak	Jumlah maksimum GetCapabilities permintaa n yang dapat Anda buat per detik per akun.
Tingkat GetExportedFile permintaan per datastore	Setiap Wilayah yang didukung: 10	Tidak	Jumlah maksimum GetExportedFile permintaan yang dapat Anda buat per detik per datastore.
Tingkat permintaan ListfHirDataStores per akun	Setiap Wilayah yang didukung: 10	Tidak	Jumlah maksimum permintaan ListfHirD ataStores yang dapat Anda buat per detik per akun.
Tarif ListFHIRExportJobs permintaan per akun	Setiap Wilayah yang didukung: 10	Tidak	Jumlah maksimum ListFHIRExportJobs permintaan yang dapat Anda buat per detik per akun.
Tarif ListFHIRImportJobs permintaan per akun	Setiap Wilayah yang didukung: 10	Tidak	Jumlah maksimum ListFHIRImportJobs permintaan yang dapat Anda buat per detik per akun.
Tarif ListTagsforResource permintaan per akun	Setiap Wilayah yang didukung: 10	Tidak	Jumlah maksimum ListTagsforResource permintaan yang dapat Anda buat per detik per akun.

Nama	Default	Dapat disesuaikan	Deskripsi
Tarif SearchEverything permintaan per akun	Setiap Wilayah yang didukung: 10	Ya	Jumlah maksimum SearchEverything permintaan yang dapat Anda buat per detik per akun.
Tingkat SearchEverything permintaan per penyimpanan data	Setiap Wilayah yang didukung: 10	Ya	Jumlah maksimum SearchEverything permintaan yang dapat Anda buat per detik per penyimpanan data.
Tarif StartFHIRBulkDeleteJob permintaan per akun	Setiap Wilayah yang didukung: 10	Ya	Jumlah maksimum StartFHIRBulkDeleteJob permintaan yang dapat Anda buat per detik per akun.
Tingkat StartFHIRBulkDeleteJob permintaan per penyimpanan data	Setiap Wilayah yang didukung: 10	Ya	Jumlah maksimum StartFHIRBulkDeleteJob permintaan yang dapat Anda buat per detik per penyimpanan data.
Tarif StartFHIRBulkMemberMatchJob permintaan per akun	Setiap Wilayah yang didukung: 10	Ya	Jumlah maksimum StartFHIRBulkMemberMatchJob permintaan yang dapat Anda buat per detik per akun.

Nama	Default	Dapat disesuaikan	Deskripsi
Tingkat StartFHIRBulkMemberMatchJob permintaan per penyimpanan data	Setiap Wilayah yang didukung: 10	Ya	Jumlah maksimum StartFHIRBulkMemberMatchJob permintaan yang dapat Anda buat per detik per penyimpanan data.
Tarif StartFHIRExportJob permintaan per akun	Setiap Wilayah yang didukung: 1	Tidak	Jumlah maksimum StartFHIRExportJob permintaan yang dapat Anda buat per detik per akun.
Tarif StartFHIRExportJob permintaan menggunakan GET per akun	Setiap Wilayah yang didukung: 1	Tidak	Jumlah maksimum StartFHIRExportJob permintaan menggunakan GET yang dapat Anda buat per detik per akun.
Tarif StartFHIRExportJob permintaan menggunakan POST per akun	Setiap Wilayah yang didukung: 1	Tidak	Jumlah maksimum StartFHIRExportJob permintaan menggunakan POST yang dapat Anda buat per detik per akun.
Tarif StartFHIRImportJob permintaan per akun	Setiap Wilayah yang didukung: 25	Tidak	Jumlah maksimum StartFHIRImportJob permintaan yang dapat Anda buat per detik per akun.

Nama	Default	Dapat disesuaikan	Deskripsi
Tarif TagResource permintaan per akun	Setiap Wilayah yang didukung: 10	Tidak	Jumlah maksimum TagResource permintaan yang dapat Anda buat per detik per akun.
Tarif UntagResource permintaan per akun	Setiap Wilayah yang didukung: 10	Tidak	Jumlah maksimum UntagResource permintaan yang dapat Anda buat per detik per akun.
Tarif ValidateResource permintaan per akun	Setiap Wilayah yang didukung: 2.000	Ya	Jumlah maksimum ValidateResource permintaan yang dapat Anda buat per detik per akun. Penyimpanan data yang dibuat sebelum 8/21 /2023 akan dibatasi hingga 300 permintaan per detik.
Tingkat ValidateResource permintaan per penyimpanan data	Setiap Wilayah yang didukung: 1.000	Ya	Jumlah maksimum ValidateResource permintaan yang dapat Anda buat per detik per penyimpanan data. Penyimpanan data yang dibuat sebelum 8/21 /2023 akan dibatasi hingga 300 permintaan per detik.

Nama	Default	Dapat disesuaikan	Deskripsi
Tarif permintaan WRITE per akun	Setiap Wilayah yang didukung: 6.000	Ya	Jumlah maksimum permintaan CREATE|UPDATE|DELETE yang dapat Anda buat per detik per akun.
Tingkat permintaan WRITE per penyimpanan data	Setiap Wilayah yang didukung: 3.000	Ya	Jumlah maksimum permintaan CREATE|UPDATE|DELETE yang dapat Anda buat per detik per penyimpanan data. Penyimpanan data yang dibuat sebelum 8/21 /2023 akan dibatasi hingga 300 permintaan per detik.
Tingkat permintaan pencarian menggunakan GET per akun	Setiap Wilayah yang didukung: 200	Ya	Jumlah maksimum permintaan pencarian menggunakan GET yang dapat Anda buat per detik per akun.
Tingkat permintaan pencarian menggunakan GET per penyimpanan data	Setiap Wilayah yang didukung: 100	Ya	Jumlah maksimum permintaan pencarian menggunakan GET yang dapat Anda buat per detik per penyimpanan data.

Nama	Default	Dapat disesu an	Deskripsi
Tingkat permintaan pencarian menggunakan POST per akun	Setiap Wilayah yang didukung: 200	Ya	Jumlah maksimum permintaan pencarian menggunakan POST yang dapat Anda buat per detik per akun.
Tingkat permintaan pencarian menggunakan POST per penyimpanan data	Setiap Wilayah yang didukung: 100	Ya	Jumlah maksimum permintaan pencarian menggunakan POST yang dapat Anda buat per detik per penyimpanan data.
Ukuran file yang diimpor individual	Setiap Wilayah yang didukung: 50 Gigabytes	Tidak	Ukuran maksimum (dalam GB) dari file individual yang disertaka n dalam file StartFHIR ImportJob.
Total Transaksi Bundel Async Antrian per datastore	Setiap Wilayah yang didukung: 500	Ya	Jumlah maksimum transaksi bundel async antrian per datastore pada waktu tertentu.
Total pekerjaan Ekspor Massal Antrian per datastore	Setiap Wilayah yang didukung: 25	Ya	Jumlah maksimum pekerjaan ekspor massal antrian per datastore pada waktu tertentu.
Total pekerjaan Impor Massal Antrian per datastore	Setiap Wilayah yang didukung: 25	Ya	Jumlah maksimum pekerjaan impor massal antrian per datastore pada waktu tertentu.

Nama	Default	Dapat disesuaikan	Deskripsi
Total ukuran pekerjaan impor	Setiap Wilayah yang didukung: 5.000 Gigabytes	Ya	Ukuran maksimum (dalam GB) dari semua file yang termasuk dalam pekerjaan impor.

Synthea tipe data yang dimuat sebelumnya untuk HealthLake

HealthLake hanya mendukung SYNTHEA sebagai tipe data yang dimuat sebelumnya. Synthea adalah generator pasien sintetis yang memodelkan riwayat `Patient` medis. Ini di-host di repositori Git open-source yang memungkinkan HealthLake untuk menghasilkan sumber daya yang sesuai dengan FHIR R4 `Bundle` sehingga pengguna dapat menguji model tanpa menggunakan data pasien yang sebenarnya.

Jenis sumber daya berikut tersedia di penyimpanan HealthLake data yang dimuat sebelumnya. Untuk informasi selengkapnya tentang preloading penyimpanan HealthLake data dengan data Synthea, lihat. Membuat penyimpanan HealthLake data

> **Note**
>
> Untuk melihat daftar lengkap sumber daya FHIR R4 yang HealthLake didukung, lihat. Jenis sumber daya yang didukung FHIR R4 untuk HealthLake

Jenis sumber daya Synthea FHIR didukung oleh HealthLake

AllergyIntolerance	Lokasi
CarePlan	MedicationAdministration
CareTeam	MedicationRequest
Klaim	Observasi
Kondisi	Organisasi

Perangkat	Pasien
DiagnosticReport	Praktisi
Perjumpaan	PractitionerRole
ExplanationofBenefit	Prosedur
ImagingStudy	Asal
Imunisasi	

AWS HealthLake proyek sampel

Untuk melanjutkan analisis Anda, Anda dapat menggunakan HealthLake dengan AWS layanan lain seperti yang ditunjukkan dalam contoh posting blog berikut.

HealthLake analitik terintegrasi

- Aplikasi kesehatan populasi dengan AWS HealthLake — Bagian 1: Analisis dan pemantauan menggunakan Amazon Quick.
- Membangun model penyakit prediktif menggunakan Amazon SageMaker AI dengan data yang AWS HealthLake dinormalisasi.
- Bangun pencarian kognitif dan grafik pengetahuan kesehatan menggunakan layanan AWS AI.

HealthLake pemantauan acara

- EventBridge Integrasi Amazon dengan AWS HealthLake

Pemecahan masalah AWS HealthLake

Topik berikut memberikan saran pemecahan masalah untuk kesalahan dan masalah yang mungkin Anda temui saat menggunakan AWS CLI, AWS SDKs, atau HealthLake konsol. Jika Anda menemukan masalah yang tidak tercantum di bagian ini, gunakan tombol Berikan umpan balik di bilah sisi kanan halaman ini untuk melaporkannya.

Topik

- Tindakan penyimpanan data
- Tindakan impor
- FHIR APIs
- Integrasi NLP
- Integrasi SQL

Tindakan penyimpanan data

Masalah: Ketika saya mencoba membuat penyimpanan HealthLake data, saya menerima kesalahan berikut:

```
AccessDeniedException: Insufficient Lake Formation permission(s): Required Database on
 Catalog
```

Pada 14 November 2022, HealthLake memperbarui izin IAM yang diperlukan untuk membuat penyimpanan data baru. Untuk informasi selengkapnya, lihat Konfigurasikan pengguna IAM atau peran yang akan digunakan HealthLake (Administrator IAM).

Masalah: Saat membuat penyimpanan HealthLake data menggunakan AWS SDKs, status pembuatan penyimpanan data mengembalikan pengecualian atau status tidak dikenal.

Perbarui AWS SDK Anda ke versi terbaru jika panggilan `DescribeFHIRDatastore` atau `ListFHIRDatastores` API menampilkan pengecualian atau status penyimpanan data yang tidak dikenal.

Tindakan impor

Masalah: Apakah saya masih dapat menggunakan HealthLake jika data saya tidak dalam format FHIR R4?

Hanya data berformat FHIR R4 yang dapat diimpor ke penyimpanan data. HealthLake Untuk daftar mitra yang dapat membantu mengubah data kesehatan yang ada ke format FHIR R4, lihat AWS HealthLake Mitra.

Masalah: Mengapa pekerjaan impor FHIR saya gagal?

Pekerjaan impor yang berhasil akan menghasilkan folder dengan hasil (log keluaran) dalam `.ndjson` format, namun, catatan individu dapat gagal diimpor. Ketika ini terjadi, `FAILURE` folder kedua

akan dihasilkan dengan manifes catatan yang gagal diimpor. Untuk informasi selengkapnya, lihat Mengimpor data FHIR dengan AWS HealthLake.

Untuk menganalisis mengapa pekerjaan impor gagal, gunakan `DescribeFHIRImportJob` API untuk menganalisis JobProperties. Berikut ini direkomendasikan:

- Jika status `FAILED` dan pesan ada, kegagalan terkait dengan parameter pekerjaan seperti ukuran data input atau jumlah file input yang berada di luar HealthLake kuota.
- Jika status pekerjaan impor adalah`COMPLETED_WITH_ERRORS`, periksa file manifes,`manifest.json`, untuk informasi tentang file mana yang tidak berhasil diimpor.
- Jika status pekerjaan impor `FAILED` dan pesan tidak ada, buka lokasi keluaran pekerjaan untuk mengakses file manifes,`manifest.json`.

Untuk setiap file input, ada file keluaran kegagalan dengan nama file input untuk sumber daya apa pun yang gagal diimpor. Tanggapan berisi nomor baris (lineID) yang sesuai dengan lokasi data input, objek respons FHIR (UpdateResourceResponse), dan kode status (statusCode) respons.

File keluaran sampel mungkin mirip dengan yang berikut ini:

```
{"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}
```

Contoh di atas menunjukkan bahwa ada kegagalan pada baris 3, 4, 7, 9, 15 dari baris input yang sesuai dari file input. Untuk masing-masing baris tersebut, penjelasannya adalah sebagai berikut:

- Pada Baris 3, respons menjelaskan bahwa `resourceType` disediakan di baris 3 file input tidak valid.
- Pada Baris 5, respons menjelaskan bahwa ada kesalahan validasi FHIR di baris 5 file input.
- Pada Baris 7, tanggapan menjelaskan bahwa ada masalah validasi dengan `resourceId` disediakan sebagai input.
- Pada Baris 9, respons menjelaskan bahwa file input harus berisi id sumber daya yang valid.
- Pada baris 15, respons file input adalah bahwa file tersebut tidak dalam format JSON yang valid.

FHIR APIs

Masalah: Bagaimana cara menerapkan otorisasi untuk FHIR? RESTful APIs

Tentukan Strategi otorisasi penyimpanan data yang akan digunakan.

Untuk membuat otorisasi SigV4 menggunakan AWS SDK untuk Python (Boto3), buat skrip yang mirip dengan contoh berikut.

```
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())
```

Masalah: Mengapa saya menerima *`AccessDenied`* kesalahan saat menggunakan FHIR RESTful APIs untuk penyimpanan data yang dienkripsi dengan kunci KMS yang dikelola pelanggan?

Izin untuk kunci yang dikelola pelanggan dan kebijakan IAM diperlukan bagi pengguna atau peran untuk mengakses penyimpanan data. Pengguna harus memiliki izin IAM yang diperlukan untuk menggunakan kunci yang dikelola pelanggan. Jika pengguna mencabut atau menghentikan hibah yang memberikan HealthLake izin untuk menggunakan kunci KMS yang dikelola pelanggan, HealthLake akan mengembalikan kesalahan. `AccessDenied`

HealthLake harus memiliki izin untuk mengakses data pelanggan, mengenkripsi sumber daya FHIR baru yang diimpor ke penyimpanan data, dan untuk mendekripsi sumber daya FHIR saat diminta. Untuk informasi selengkapnya, lihat Izin pemecahan masalah AWS KMS.

Masalah: Operasi *`POST`* API FHIR untuk HealthLake menggunakan dokumen 10MB mengembalikan kesalahan. *`413 Request Entity Too Large`*

AWS HealthLake memiliki batas Create and Update API sinkron sebesar 5MB untuk menghindari peningkatan latensi dan batas waktu. Anda dapat menyerap dokumen besar, hingga 164MB, menggunakan jenis `Binary` sumber daya menggunakan API Impor Massal.

Integrasi NLP

Masalah: Bagaimana cara mengaktifkan HealthLake fitur pemrosesan bahasa alami terintegrasi?

Per 14 November 2022, perilaku default penyimpanan HealthLake data berubah.

Penyimpanan data saat ini: Semua penyimpanan HealthLake data saat ini akan berhenti menggunakan pemrosesan bahasa alami (NLP) pada sumber daya yang dikodekan `DocumentReference` base64. Ini berarti bahwa `DocumentReference` sumber daya baru tidak akan dianalisis menggunakan NLP, dan tidak ada sumber daya baru yang akan dihasilkan berdasarkan teks dalam jenis `DocumentReference` sumber daya. Untuk `DocumentReference` sumber daya yang ada, data dan sumber daya yang dihasilkan melalui NLP tetap ada, tetapi tidak akan diperbarui setelah 20 Februari 2023.

Penyimpanan data baru: penyimpanan HealthLake data yang dibuat setelah 20 Februari 2023 tidak akan melakukan pemrosesan bahasa alami (NLP) pada sumber daya yang disandikan base64. `DocumentReference`

Untuk mengaktifkan integrasi HealthLake NLP, buat kasus dukungan menggunakan. AWS Support Center Console Untuk membuat kasus Anda, masuk ke kasing Anda Akun AWS, lalu pilih Buat kasus. Untuk mempelajari selengkapnya tentang membuat manajemen kasus dan kasus, lihat Membuat kasus dukungan dan manajemen kasus di Panduan Dukungan Pengguna.

Masalah: >Bagaimana cara menemukan *`DocumentReference`* sumber daya yang tidak dapat diproses oleh NLP terintegrasi?

Jika `DocumentReference` sumber daya tidak valid, HealthLake berikan ekstensi yang menunjukkan kesalahan validasi alih-alih menyediakannya dalam output NLP medis terintegrasi. Untuk menemukan **`DocumentReference`** sumber daya yang menyebabkan kesalahan validasi selama pemrosesan NLP, Anda dapat menggunakan HealthLake **`search`** fungsi FHIR dengan kunci pencarian cm-decoration-statusdan nilai pencarian VALIDATION_ERROR. Pencarian ini akan mencantumkan semua `DocumentReference` sumber daya yang menyebabkan kesalahan validasi, bersama dengan pesan kesalahan yang menjelaskan sifat kesalahan. Struktur bidang ekstensi dalam `DocumentReference` sumber daya tersebut dengan kesalahan validasi akan menyerupai contoh berikut.

```
"extension": [
          {
              "extension": [
                  {
                      "url": "http://healthlake.amazonaws.com/aws-cm/status/",
                      "valueString": "VALIDATION_ERROR"
                  },
                  {
                      "url": "http://healthlake.amazonaws.com/aws-cm/message/",
                      "valueString": "Resource led to too many nested objects after NLP
 operation processed the document. 10937 nested objects exceeds the limit of 10000."
                  }
              ],
              "url": "http://healthlake.amazonaws.com/aws-cm/"
          }
    ]
```

> **Note**
>
> A juga `VALIDATION_ERROR` dapat terjadi jika dekorasi NLP menciptakan lebih dari 10.000 objek bersarang. Ketika ini terjadi, dokumen harus dibagi menjadi dokumen yang lebih kecil sebelum diproses.

Integrasi SQL

Masalah: Mengapa saya mendapatkan Lake Formation *`permissions error: lakeformation:PutDataLakeSettings`* saat menambahkan administrator danau data baru?

Jika pengguna atau peran IAM Anda berisi kebijakan `AWSLakeFormationDataAdmin` AWS terkelola, Anda tidak dapat menambahkan administrator data lake baru. Anda akan mendapatkan kesalahan yang berisi berikut ini:

```
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
```

Kebijakan AWS terkelola `AdministratorAccess` diperlukan untuk menambahkan pengguna IAM atau peran sebagai administrator danau data AWS Lake Formation. Jika pengguna atau peran IAM Anda juga berisi `AWSLakeFormationDataAdmin` tindakan akan gagal. Kebijakan `AWSLakeFormationDataAdmin` AWS terkelola berisi penolakan eksplisit untuk operasi AWS Lake Formation API,`PutDataLakeSetting`. Bahkan administrator dengan akses penuh untuk AWS menggunakan kebijakan `AdministratorAccess` terkelola dapat dibatasi oleh `AWSLakeFormationDataAdmin` kebijakan.

Masalah: Bagaimana cara memigrasi penyimpanan HealthLake data yang ada untuk menggunakan integrasi SQL Amazon Athena?

HealthLake penyimpanan data yang dibuat sebelum 14 November 2022 berfungsi, tetapi tidak dapat ditanyakan di Athena menggunakan SQL. Untuk menanyakan penyimpanan data yang sudah ada sebelumnya dengan Athena, Anda harus terlebih dahulu memigrasikannya ke penyimpanan data baru.

Untuk memigrasikan HealthLake data Anda ke penyimpanan data baru

1. Buat toko data baru.

2. Ekspor data dari yang sudah ada sebelumnya ke bucket Amazon S3.
3. Impor data ke penyimpanan data baru dari bucket Amazon S3.

> **Note**
>
> Mengekspor data ke bucket Amazon S3 menimbulkan biaya tambahan. Biaya tambahan tergantung pada ukuran data yang Anda ekspor.

Masalah: Saat membuat penyimpanan HealthLake data baru untuk integrasi SQL, status penyimpanan data tidak berubah. `Creating`

Jika Anda mencoba membuat penyimpanan HealthLake data baru, dan status penyimpanan data Anda tidak berubah dari Membuat, Anda perlu memperbarui Athena untuk menggunakan. AWS Glue Data Catalog Untuk informasi selengkapnya, lihat Memutakhirkan ke Katalog Data AWS Glue step-by-step di Panduan Pengguna Amazon Athena.

Setelah berhasil memutakhirkan AWS Glue Data Catalog, Anda dapat membuat penyimpanan HealthLake data.

Untuk menghapus penyimpanan HealthLake data lama, buat kasus dukungan menggunakan AWS Support Center Console. Untuk membuat kasus Anda, masuk ke kasing Anda Akun AWS, lalu pilih Buat kasus. Untuk mempelajari selengkapnya, lihat Membuat kasus dukungan dan manajemen kasus di Panduan Dukungan Pengguna.

Masalah: Konsol Athena tidak berfungsi setelah mengimpor data ke penyimpanan data baru HealthLake

Setelah Anda mengimpor data ke penyimpanan HealthLake data baru, data mungkin tidak tersedia untuk segera digunakan. Ini untuk memberikan waktu bagi data untuk dicerna ke dalam tabel Apache Iceberg. Coba lagi di lain waktu.

Masalah: Bagaimana cara menghubungkan hasil pencarian di Athena ke layanan lain? AWS

Saat membagikan hasil penelusuran Anda dari Athena dengan AWS layanan lain, masalah dapat terjadi saat Anda menggunakan `json_extract[1]` sebagai bagian dari kueri penelusuran SQL. Untuk memperbaiki masalah ini, Anda harus memperbarui ke`CATVAR`.

Anda mungkin mengalami masalah ini saat mencoba Membuat hasil penyimpanan, Tabel (statis), atau Tampilan (dinamis).

Menggunakan HealthLake dengan AWS SDK

AWS kit pengembangan perangkat lunak (SDKs) tersedia untuk banyak bahasa pemrograman populer. Setiap SDK menyediakan API, contoh kode, dan dokumentasi yang memudahkan developer untuk membangun aplikasi dalam bahasa pilihan mereka.

Dokumentasi SDK	Contoh kode
AWS SDK untuk C++	AWS SDK untuk C++ contoh kode
AWS CLI	AWS CLI contoh kode
AWS SDK untuk Go	AWS SDK untuk Go contoh kode
AWS SDK untuk Java	AWS SDK untuk Java contoh kode
AWS SDK untuk JavaScript	AWS SDK untuk JavaScript contoh kode
AWS SDK untuk Kotlin	AWS SDK untuk Kotlin contoh kode
AWS SDK untuk .NET	AWS SDK untuk .NET contoh kode
AWS SDK untuk PHP	AWS SDK untuk PHP contoh kode
Alat AWS untuk PowerShell	Alat AWS untuk PowerShell contoh kode
AWS SDK untuk Python (Boto3)	AWS SDK untuk Python (Boto3) contoh kode
AWS SDK untuk Ruby	AWS SDK untuk Ruby contoh kode
AWS SDK for Rust	AWS SDK for Rust contoh kode
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP contoh kode
AWS SDK for Swift	AWS SDK for Swift contoh kode

Ketersediaan contoh

Tidak dapat menemukan apa yang Anda butuhkan? Minta contoh kode menggunakan tautan Berikan umpan balik di bagian bawah halaman ini.

AWS HealthLake rilis

Tabel berikut menunjukkan kapan fitur dan pembaruan dirilis untuk AWS HealthLake. Untuk informasi selengkapnya tentang rilis, lihat topik tertaut.

Perubahan	Deskripsi	Tanggal
Keamanan dan pemfilteran tag untuk $export dan $ davinci-data-export	`$davinci-data-export` Operasi `$export` dan sekarang mendukung `_security` dan parameter `_tag` kueri untuk memfilter sumber daya yang diekspor oleh `meta.security` dan nilai `meta.tag` Coding. Ini memungkinkan ekspor multi-penyewa dan kontrol akses granular menggunakan format FHIR standar. `system|code`	April 30, 2026
$ bulk-member-match operasi	AWS HealthLake sekarang mendukung `$bulk-member-match` operasi untuk memproses beberapa permintaan kecocokan anggota secara asinkron. Operasi ini memungkinkan organisasi perawatan kesehatan untuk secara efisien mencocokkan ratusan pengidentifikasi unik anggota di berbagai sistem perawatan kesehatan menggunakan informasi demografis dan cakupan dalam satu permintaan massal.	April 1, 2026

	• Menangani hingga 500 anggota per permintaan dengan hingga 5 operasi bersamaan per penyimpanan data • Hasil dikategorikan ke dalam MatchedMembers, NonMatchedMembers, dan kelompok ConsentConstrainedMembers • Terintegrasi dengan alur `$davinci-data-export` kerja data end-to-end massal Untuk informasi selengkapnya, lihat the section called "$ bulk-member-match".	
Transaksi bundel asinkron	AWS HealthLake sekarang mendukung `Bundle` tipe asinkron`transaction` , memungkinkan Anda untuk mengirimkan transaksi dengan hingga 500 sumber daya. HealthLake mengantri transaksi untuk diproses dan segera mengembalikan URL polling untuk memeriksa status dan mengambil hasil. Untuk informasi selengkapnya, lihat Transaksi bundel asinkron.	Maret 24, 2026

Dukungan Patch dalam Operasi Bundel	HealthLake sekarang mendukung Bundle Patch, mengaktifkan keduanya FHIRPath dan JSON Patch untuk paket Transaksi dan Batch, serta FHIRPath untuk Patch API. Kemampuan ini memungkinkan pelanggan untuk menambal sumber daya secara langsung dalam operasi Bundle tanpa memerlukan pengganti an sumber daya penuh, mengurangi ukuran muatan, menyederhanakan alur kerja integrasi, dan memungkin kan konsumsi data yang lebih cepat.	Maret 20, 2026
DaVinci PDex Jenis Ekspor untuk $ davinci-data-export	`$davinci-data-expo rt` Operasi sekarang mendukung jenis PDex ekspor untuk Akses Penyedia Payer-to-Payer, dan Akses Anggota APIs. • Logika inklusi berbasis profil untuk sumber daya ExplanationOfBenefit • Transformasi data keuangan dengan `_includeE 0B2xWoFinancial` parameter • Filter temporal 5 tahun pada data klinis dan klaim	Maret 20, 2026

_sampai Parameter di $export & $ davinci-data-export	Parameter penyaringan temporal untuk operasi ekspor	Februari 26, 2026
_sertakan Parameter Pencarian	HealthLake sekarang mendukung meliputi: * dan include:iterate	Februari 26, 2026
Titik akhir interoperabilitas CMS	Fitur ini memungkinkan Anda melacak penggunaan API menurut kategori CMS dan selanjutnya melaporkan metrik penggunaan untuk tujuan kepatuhan.	Februari 26, 2026
Dukungan Jenis Pesan Bundel	Dukungan terbatas untuk sumber daya FHIR Bundle dengan tipe Pesan	Februari 26, 2026

Ditambahkan dukungan untuk baru IGs	AWS HealthLake telah memperluas dukungan FHIR Implementation Guides (IGs) untuk CMS 0057F: • Support untuk CARIN Blue Button 2.0.0 & 2.1.0 • Support untuk Da Vinci Payer Data Exchange 2.0.0 & 2.1.0 • Support untuk DaVinci Payer Data Exchange (PDex) Formularium Obat AS 2.0.1 & 2.1.0 • Support untuk Da Vinci Clinical Data Exchange (CDex) 2.1.0 • Support untuk Da Vinci Prior Authorization Support (PAS) FHIR IG 2.1.0	Februari 26, 2026
$submit Operasi	`$submit`Operasi ini memungkinkan Anda untuk secara elektronik mengirimk an permintaan otorisasi sebelumnya kepada pembayar untuk persetujuan.	Februari 26, 2026
$ kuesioner paket Operasi	`$questionnaire-pac kage` Operasi mengambil bundel komprehensif yang berisi Kuesioner FHIR dan semua dependensinya yang diperlukan untuk membuat dan memproses kuesioner.	Februari 26, 2026

$inquire Operasi	`$inquire`Operasi ini memungkinkan Anda untuk memeriksa status permintaan otorisasi sebelumnya yang diajukan sebelumnya.	Februari 26, 2026
$member-remove Operasi	Operasi $member-remove memungkinkan Anda untuk menghapus anggota dari Daftar Atribusi Anggota FHIR (sumber daya Grup) di. AWS HealthLake	November 12, 2025
Operasi $member-match	AWS HealthLake sekarang mendukung operasi pencocokan anggota untuk sumber daya Pasien, memungkinkan organisasi perawatan kesehatan untuk menemukan pengidentifikasi unik anggota di berbagai sistem perawatan kesehatan menggunakan informasi demografis dan cakupan.	November 12, 2025
$member-add Operasi	Operasi FHIR $member-add menambahkan anggota (pasien) ke sumber daya Grup, khususnya Daftar Atribusi Anggota.	November 12, 2025
$ davinci-data-export Operasi	Operasi $ adalah davinci-data-export operasi FHIR asinkron yang memungkinkan ekspor data Daftar Atribusi Anggota dari. AWS HealthLake	November 12, 2025

$ confirm-attribution-list Operasi	Menunjukkan kepada Produser bahwa Konsumen tidak memiliki perubahan lagi untuk dilakukan pada Daftar Atribusi, menyelesaikan daftar atribusi dengan menghapus anggota yang tidak aktif dan mengubah status menjadi “final”.	November 12, 2025
$atribusi-status Operasi	Mengambil status atribusi untuk anggota tertentu, mengembalikan Bundel yang berisi semua sumber atribusi yang terkait dengan Pasien.	November 12, 2025
Langganan FHIR	HealthLake mendukung Langganan FHIR, memungkinkan Anda menerima pemberitahuan waktu nyata ketika perubahan data perawatan kesehatan tertentu terjadi dan membangun alur kerja berbasis peristiwa.	Oktober 30, 2025
Perluasan wilayah ke Montreal Kanada	HealthLake tersedia di wilayah Kanada (Montreal). Untuk informasi selengkapnya, lihat Titik akhir layanan.	Oktober 17, 2025

Ditambahkan dukungan untuk baru IGs	AWS HealthLake telah memperluas dukungan FHIR Implementation Guides (IGs) untuk pasar Kanada berikut: • CA Core+profil FHIR dasar Kanada mendefinisikan elemen data inti dan kendala untuk interoperabilitas di seluruh sistem perawatan kesehatan Kanada. • CA:EREC Pan-Canadian eReferral- Spesifikasi eConsultStandardized FHIR untuk rujukan elektronik dan konsultasi antara penyedia layanan kesehatan di seluruh Kanada. • Ringkasan Pasien Edisi Kanada (PS-CA) Adaptasi Kanada dari Ringkasan Pasien Internasional (IPS) untuk berbagi informasi kesehatan pasien penting di seluruh pengaturan perawatan. • Profil FHIR RepositoryOntario khusus Obat Kesehatan Digital Ontario untuk pengobatan standar dan pertukaran data resep dalam sistem kesehatan provinsi.	Oktober 17, 2025

Dukungan IG spesifik wilayah	HealthLake sekarang mendukung wilayah tertentu IGs. Untuk informasi selengkapnya, lihat validasi profil.	Oktober 8, 2025
Operasi Patch	HealthLake memungkinkan memodifikasi elemen tertentu dari sumber daya FHIR menggunakan operasi JSON Patch tanpa memperbarui seluruh sumber daya.	Agustus 18, 2025
$ validasi Operasi	HealthLake memungkinkan memvalidasi sumber daya FHIR terhadap spesifikasi dan profil tanpa melakukan operasi penyimpanan, mengembalikan hasil validasi terperinci.	Agustus 18, 2025
Operasi $ pembersihan	HealthLake termasuk fungsionalitas untuk menghapus semua sumber daya secara permanen dalam kompartemen pasien dari datastore.	Agustus 18, 2025
$lookup Operasi	HealthLake memberikan kemampuan untuk mengambil informasi rinci tentang konsep tertentu dalam a CodeSystem dengan menyediakan kode dan pengidentifikasi sistem.	Agustus 18, 2025

$ perluas Operasi	HealthLake sekarang memungkinkan perluasan ValueSet sumber daya untuk mengambil daftar lengkap kode yang terkandung dalam dicerna pelanggan ValueSets.	Agustus 18, 2025
$ Hapus Operasi	HealthLake sekarang menawarkan penghapus an permanen sumber daya tertentu dan semua versi historisnya dari datastore.	Agustus 18, 2025
$ dokumen Operasi	HealthLake mendukung pembuatan dokumen klinis lengkap dengan menggabun gkan sumber daya Komposisi dengan semua sumber daya yang direferensikan ke dalam satu Bundel dokumen.	Agustus 18, 2025
:di bawah Search Modifier	HealthLake memperkenalkan pencarian nilai URI yang secara hierarkis di bawah URI yang ditentukan dalam sistem terminologi.	Agustus 8, 2025

Hapus Bersyarat	HealthLake sekarang mendukung FHIR Conditional Delete, memungkinkan organisasi perawatan kesehatan untuk menghapus sumber daya yang ada berdasarkan kriteria pencarian daripada dengan ID FHIR logis. Lihat Menghapus sumber daya FHIR berdasarkan kondisi untuk informasi selengkapnya.	Juli 7, 2025
Ditambahkan dukungan untuk baru IGs	AWS HealthLake telah memperluas dukungan FHIR Implementation Guides (IGs) untuk hal-hal berikut: • US Core 7.0.0 yang menentukan cara menggunakan FHIR untuk mengimplementasikan standar USCDI 4.0 • Panduan Implementasi UK Core 2.0.1 untuk memberikan panduan implementasi FHIR di Inggris	Juli 7, 2025
Perluasan wilayah ke Dublin Irlandia	HealthLake tersedia di wilayah UE (Dublin). Untuk informasi selengkapnya, lihat Titik akhir layanan.	Juni 9, 2025

Transaksi Jenis Bundel	HealthLake sekarang mendukung jenis 'Transaksi' FHIR Bundle, memungkinkan organisasi perawatan kesehatan untuk mengirimkan beberapa sumber daya sebagai operasi atom tunggal. Hal ini memungkinkan pertukaran data yang lebih efisien dan alur kerja integrasi. Misalnya, penyedia layanan kesehatan sekarang dapat memperbarui catatan pasien, daftar obat, dan janji temu dalam satu transaksi, mengurangi kompleksitas dan potensi kesalahan. Lihat Bundling FHIR Resources untuk informasi lebih lanjut.	April 28, 2025

Ditambahkan dukungan untuk baru IGs	AWS HealthLake AWS HealthLake telah memperluas dukungan FHIR Implementation Guides (IGs) untuk hal-hal berikut: • Panduan Implementasi NCQA HEDIS® (0.3.1): Mendukung pengukuran dan pelaporan kualitas untuk Kumpulan Data dan Informasi Efektivitas Kesehatan (HEDIS). • Ringkasan Pasien Internasional (IPS) (2.0.0): Memungkinkan pertukaran informasi kesehatan penting untuk mendukung kesinambungan perawatan bagi pasien. • Pengukuran Kualitas (5.0.0): Mendukung representasi dan pertukaran definisi dan data ukuran kualitas. • Pelaporan Genomik (3.0.0): Memfasilitasi pertukaran data dan laporan genom.	April 28, 2025

Kunci Idempotensi	HealthLake sekarang mendukung kunci idempotensi untuk operasi FHIR POST, menyediakan mekanisme yang kuat untuk memastikan integritas data selama pembuatan sumber daya. Lihat Idempotensi dan Konkurensi untuk informasi lebih lanjut.	April 18, 2025
Konsistensi sejarah FHIR	HealthLake sekarang mendukung konsistensi yang kuat untuk penyimpanan data yang diaktifkan riwayat melalui `x-amz-fhir-history-consistency-level` header baru. Ketika disetel ke 'kuat', hasil pencarian FHIR mencakup semua catatan yang diindeks terlepas dari status pembaruan. Lihat Tingkat Konsistensi Pencarian FHIR untuk informasi selengkapnya.	April 18, 2025

Etag dan 'if-match'	HealthLake sekarang menyediakan dukungan ETag, memungkinkan klien untuk menggunakan header 'If-Match' untuk memastikan pembaruan idempoten. Ini membantu menjaga integrita s data dengan mencegah penimpaan yang tidak disengaja selama pembaruan bersamaan. Ini sangat berharga dalam lingkungan perawatan kesehatan volume tinggi di mana beberapa sistem mungkin mencoba memperbarui catatan yang sama secara bersamaan. Lihat ETag di AWS HealthLake untuk informasi selengkapnya.	April 18, 2025

Bersyarat PUTs dalam Bundel	HealthLake sekarang mendukung pembaruan bersyarat untuk FHIR Bundles, memberikan organisasi perawatan kesehatan lebih banyak fleksibilitas untuk mengelola dan memperbarui data mereka. Klien sekarang dapat menentukan kriteria untuk membuat, memperbar ui, atau menghapus sumber daya secara kondisional sebagai bagian dari transaksi Bundel. Ini menyederh anakan proses sinkronis asi data antar sistem dan mengurangi kebutuhan logika sisi klien yang kompleks. Lihat Bersyarat PUTs dalam Bundel untuk informasi lebih lanjut.	April 18, 2025

SMART pada cakupan FHIR V2	HealthLake mendukung SMART pada cakupan FHIR V2 untuk membuat, membaca, memperbarui, menghapus, dan mencari sumber daya FHIR. Untuk informasi selengkapnya, lihat SMART di cakupan sumber daya FHIR untuk. HealthLake • SMART pada cakupan FHIR V2 tersedia untuk semua penyimpanan HealthLake data yang dibuat setelah 01/22/2025. Jika penyimpanan data Anda dibuat sebelum tanggal ini, Anda dapat mengirimkan tiket dukungan agar SMART di cakupan FHIR V2 diaktifkan. Buat kasus menggunakan AWS Support Center Console. Untuk membuat kasus Anda, masuk ke kasing Anda Akun AWS dan pilih Buat kasus.	Januari 22, 2025
Profil Inti AS FHIR, versi 6.1.0	HealthLake mendukung versi 6.1.0 dari Profil Inti AS FHIR. Untuk informasi selengkapnya, lihat validasi profil FHIR untuk. HealthLake	Januari 22, 2025

FHIR $ ekspor dengan GET	HealthLake mendukung FHIR `$export` dengan`GET`. Untuk informasi selengkapnya, lihat Mengekspor HealthLake data dengan `$export` FHIR.	Januari 22, 2025
Panduan Pengembang Refactored dengan contoh kode yang diuji	HealthLake memperken alkan Panduan Pengemban g refactored dengan contoh kode yang diuji untuk tindakan native AWS CLI dan SDK. AWS Selain itu, prosedur sekarang tersedia untuk semua interaksi API FHIR yang didukung. Untuk informasi selengkapnya, lihat Contoh kode dan Mengelola sumber daya FHIR.	Desember 18, 2024

FHIR `history` dan interaksi `vread`	HealthLake mendukung `history` interaksi FHIR untuk mengambil riwayat sumber daya tertentu dan `vread` interaksi untuk melakukan pembacaan sumber daya khusus versi. Untuk informasi selengkapnya, lihat Membaca riwayat sumber daya FHIR. • Sumber daya FHIR `history` diaktifkan secara default ke semua penyimpanan HealthLake data yang dibuat setelah 10/25/2024. Jika penyimpanan data Anda dibuat sebelum tanggal ini, Anda dapat mengirimkan tiket dukungan untuk mengaktifkan `history` interaksi FHIR. Buat kasus menggunakan AWS Support Center Console. Untuk membuat kasus Anda, masuk ke kasing Anda Akun AWS dan pilih Buat kasus.	Oktober 25, 2024

Operasi FHIR `Patient/$everything`	HealthLake mendukung `Patient/$everything` operasi FHIR untuk mencari `Patient` sumber daya dan semua sumber daya terkait. Dengan menggunakan operasi ini, Anda dapat mengakses seluruh catatan pasien atau mengunduh `Patient` data secara massal. Untuk informasi lebih lanjut, lihat Mendapatkan data pasien dengan `Patient/$everything`. • FHIR `Patient/$everything` diaktifkan secara default ke semua penyimpanan HealthLake data yang dibuat setelah 02/27/2024. Jika penyimpanan data Anda dibuat sebelum tanggal ini, Anda dapat mengirimkan tiket dukungan agar `Patient/$everything` operasi diaktifkan. Buat kasus menggunakan AWS Support Center Console. Untuk membuat kasus Anda, masuk ke kasing Anda Akun AWS dan pilih Buat kasus.	Februari 27, 2024

Sumber daya FHIR `VerificationResult`	HealthLake mendukung jenis `VerificationResult` sumber daya FHIR untuk menjelaskan persyaratan validasi, sumber, status, dan tanggal untuk satu atau lebih elemen. Untuk informasi selengkapnya, lihat tipe sumber daya FHIR R4 untuk. HealthLake	Desember 9, 2023

Operasi FHIR `$export`	HealthLake mendukung `$export` operasi FHIR untuk mengekspor data kesehatan secara massal dari penyimpanan HealthLake data. Untuk informasi selengkapnya, lihat Mengekspor HealthLake data dengan `$export` FHIR. • FHIR `$export` diaktifkan secara default ke semua penyimpanan HealthLake data yang dibuat setelah 1 Juni 2023. Jika penyimpanan data Anda dibuat sebelum tanggal ini, Anda dapat mengirimkan tiket dukungan agar `$export` operasi diaktifkan. Buat kasus menggunakan AWS Support Center Console. Untuk membuat kasus Anda, masuk ke kasing Anda Akun AWS dan pilih Buat kasus. • HealthLake penyimpanan data yang dibuat sebelum 06/01/23 hanya mendukung permintaan `$export` pekerjaan untuk ekspor seluruh sistem. • HealthLake penyimpanan data yang dibuat sebelum 06/01/23 tidak mendukung mendapatkan status FHIR `$export` menggunakan	1 Juni 2023

	GET permintaan pada titik akhir penyimpanan data.	
SMART pada dukungan FHIR	HealthLake menambahkan dukungan untuk SMART pada otorisasi FHIR. Untuk informasi selengkapnya, lihat SMART pada dukungan FHIR untuk AWS HealthLake.	31 Mei 2023
Validasi profil FHIR	HealthLake mendukung validasi profil FHIR untuk mendefinisikan definisi jenis sumber daya tertentu menggunakan ekstensi kendala and/or pada jenis sumber daya dasar. Untuk informasi selengkapnya, lihat Validasi profil.	31 Mei 2023
Wilayah Asia Pasifik (Mumbai)	HealthLake tersedia di wilayah Asia Pasifik (Mumbai). Untuk informasi selengkapnya, lihat Titik akhir layanan.	4 April 2023

Pemrosesan bahasa alami dimatikan secara default	HealthLake mematikan pemrosesan bahasa alami terintegrasi (NLP) di semua penyimpanan data per 20 Februari 2023. Anda dapat mengirimkan tiket dukungan untuk mengaktifkan fungsiona litas NLP terintegrasi. Buat kasus menggunakan AWS Support Center Console. Untuk membuat kasus Anda, masuk ke kasing Anda Akun AWS dan pilih Buat kasus. Untuk mempelajari lebih lanjut tentang NLP terintegrasi, lihat Mengintegrasikan NLP dengan. HealthLake •	Februari 20, 2023

Indeks SQL dan kueri dengan Amazon Athena	HealthLake mendukung kueri data FHIR dengan SQL menggunakan Amazon Athena. Untuk informasi selengkapnya, lihat Menanyakan HealthLake data dengan Amazon Athena. • Fungsionalitas kueri SQL diaktifkan secara default ke semua penyimpanan HealthLake data yang dibuat setelah 11/14/2022. Jika penyimpanan data Anda dibuat sebelum tanggal ini, Anda dapat mengirimkan tiket dukungan agar fungsionalitas kueri SQL diaktifkan. Buat kasus menggunakan AWS Support Center Console. Untuk membuat kasus Anda, masuk ke kasing Anda Akun AWS dan pilih Buat kasus. • Dengan fungsionalitas kueri SQL, pengaturan IAM untuk mengakses HealthLake harus diperbarui. Untuk membuat penyimpanan HealthLake data dan memberikan akses ke mereka di Athena, Anda harus menambahkan kebijakan `AWSLakeFormationDataAdmin` terkelola ke pengguna,	November 14, 2022

	grup, atau peran IAM Anda. Anda dapat menggunakan `AWSLakeFormationDataAdmin` kebijakan untuk membuat administrator data lake dan memberikan akses ke penyimpanan data di Athena. Untuk informasi selengkapnya, lihat Mengonfigurasi pengguna atau peran IAM.	
Total ukuran pekerjaan impor meningkat	HealthLake memperbarui `StartFHIRImportJob` permintaan `Total import job size` untuk 500 GB. Untuk informasi selengkapnya, lihat Kuota layanan.	3 Oktober 2022
Sumber daya FHIR `Bundle`	HealthLake mendukung jenis `Bundle` sumber daya FHIR untuk memproses beberapa sumber daya FHIR secara bersamaan. Untuk informasi selengkapnya, lihat Menggabungkan sumber daya FHIR.	5 Agustus 2022
Pembaruan kuota untuk interaksi FHIR	HealthLake memperbarui kuota untuk interaksi manajemen sumber daya FHIR. Untuk informasi selengkapnya, lihat Kuota layanan.	Juli 16, 2022

Parameter `_include` pencarian FHIR	HealthLake menambahkan dukungan untuk parameter `_include` pencarian FHIR untuk mengembalikan sumber daya tambahan dalam `search` permintaan. Untuk informasi selengkapnya, lihat Parameter pencarian lanjutan.	Juli 16, 2022
AWS HealthLake umumnya tersedia	HealthLake umumnya tersedia di semua wilayah yang didukung. Untuk informasi selengkapnya, lihat Titik akhir layanan.	15 Juli 2021

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.