aws

Panduan Developerr

AWS HealthImaging

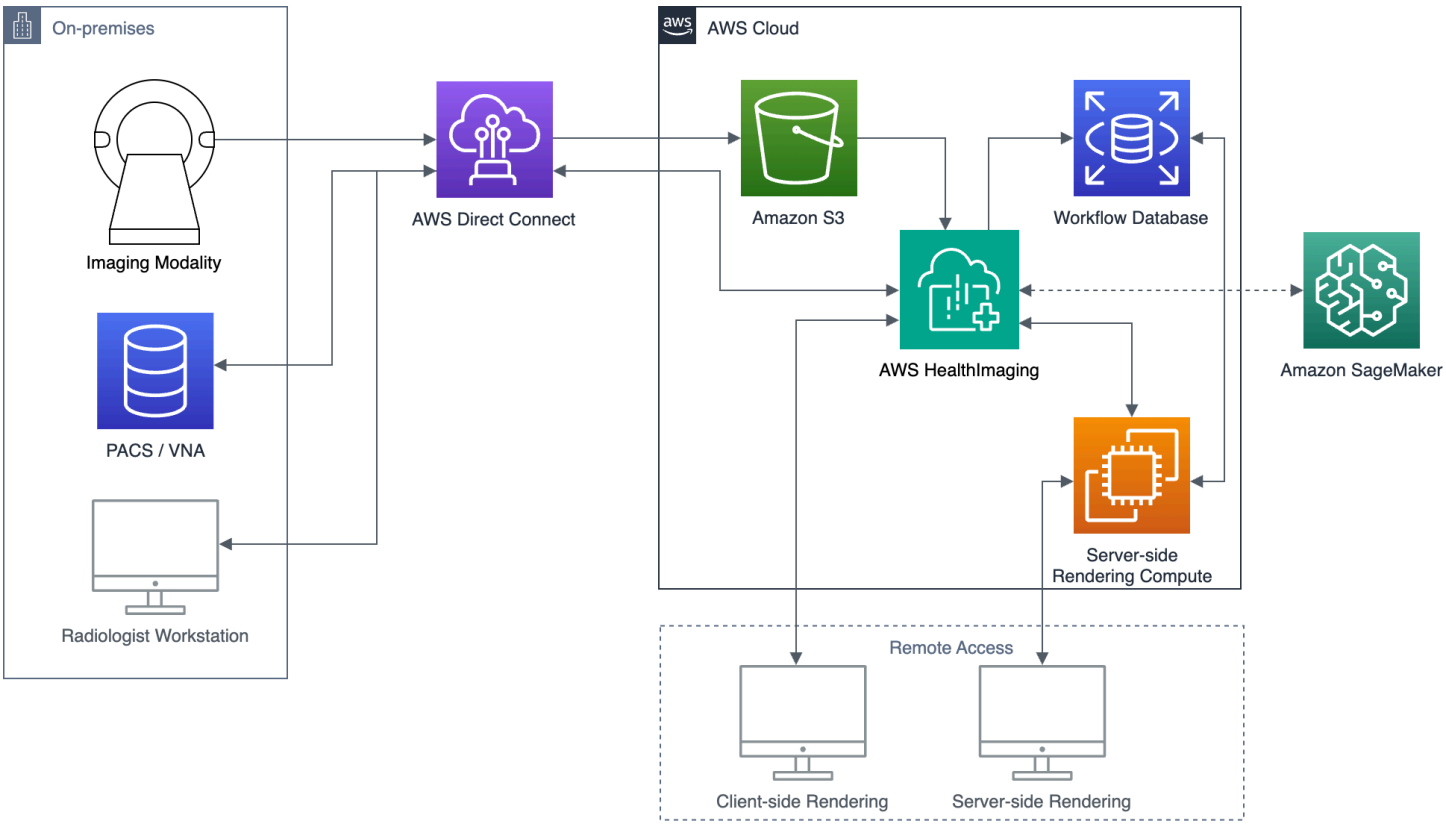
AWS HealthImaging: Panduan Developerr

Table of Contents

Apa itu AWS HealthImaging?

AWS HealthImaging adalah layanan yang memenuhi syarat HIPAA yang memberdayakan penyedia layanan kesehatan, organisasi ilmu hayati, dan mitra perangkat lunak mereka untuk menyimpan, menganalisis, dan berbagi gambar medis di cloud pada skala petabyte. HealthImaging kasus penggunaan meliputi:

- Pencitraan perusahaan — Simpan dan streaming data pencitraan medis Anda langsung dari AWS Cloud sambil mempertahankan kinerja latensi rendah dan ketersediaan tinggi.
- Arsip gambar jangka panjang - Hemat biaya arsip gambar jangka panjang sambil mempertahankan akses pengambilan gambar subdetik.
- Pengembangan AI/ML — Jalankan inferensi kecerdasan buatan dan pembelajaran mesin (AI/ML) melalui arsip pencitraan Anda dengan dukungan dari alat dan layanan lain.
- Analisis multimodal — Gabungkan data pencitraan klinis Anda dengan AWS HealthLake (data kesehatan) dan AWS HealthOmics (data omics) untuk memberikan wawasan tentang pengobatan presisi.



AWS HealthImaging menyediakan akses ke data gambar (misalnya X-Ray, CT, MRI, Ultrasound) sehingga aplikasi pencitraan medis yang dibangun di cloud dapat mencapai kinerja yang sebelumnya hanya mungkin dilakukan di tempat. Dengan HealthImaging, Anda mengurangi biaya infrastruktur dengan menjalankan aplikasi pencitraan medis Anda dalam skala besar dari satu salinan resmi dari setiap gambar medis. AWS Cloud

Topik

Pemberitahuan penting

AWS HealthImaging bukan pengganti saran, diagnosis, atau perawatan medis profesional, 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 AWS apa pun HealthImaging, termasuk terkait dengan produk pihak ketiga yang dimaksudkan untuk menginformasikan pengambilan keputusan klinis. AWS hanya HealthImaging boleh digunakan dalam perawatan pasien atau skenario klinis setelah ditinjau oleh profesional medis terlatih yang menerapkan penilaian medis yang baik.

Fitur AWS HealthImaging

AWS HealthImaging menyediakan fitur-fitur berikut.

Metadata DICOM yang ramah pengembang

AWS HealthImaging menyederhanakan pengembangan aplikasi dengan mengembalikan metadata DICOM dalam format yang ramah pengembang. Setelah mengimpor data pencitraan Anda, atribut metadata individual dapat diakses menggunakan kata kunci yang ramah manusia daripada angka heksadesimal yang tidak dikenal. group/element Elemen DICOM tingkat Pasien, Studi, dan Seri dinormalisasi, menghilangkan kebutuhan pengembang aplikasi untuk menangani

inkonsistensi antara Instans SOP. Selain itu, nilai atribut metadata dapat langsung diakses dalam tipe runtime asli.

Penguraian kode gambar yang dipercepat SIMD

AWS HealthImaging mengembalikan bingkai gambar (data piksel) yang dikodekan sebagai gambar High Throughput JPEG 2000 (HTJ2K), codec kompresi gambar tingkat lanjut. HTJ2K mengambil keuntungan dari single instruction multiple data (SIMD) pada prosesor modern untuk memberikan tingkat kinerja baru. HTJ2K adalah urutan besarnya lebih cepat dari JPEG2 000 dan setidaknya dua kali lebih cepat dari semua sintaks transfer DICOM lainnya. WASM-SIMD dapat digunakan untuk membawa kecepatan ekstrim ini ke nol pemirsa web footprint. Untuk informasi selengkapnya, lihat Sintaks transfer yang didukung.

Verifikasi data Pixel

AWS HealthImaging menyediakan verifikasi data piksel bawaan dengan memeriksa status encoding dan decoding lossless dari setiap gambar selama impor. Untuk informasi selengkapnya, lihat Verifikasi data Pixel.

Kinerja terdepan di industri

AWS HealthImaging menetapkan standar baru untuk kinerja pemuatan gambar berkat pengkodean metadata yang efisien, kompresi lossless, dan akses data resolusi progresif. Pengkodean metadata yang efisien memungkinkan pemirsa gambar dan algoritme AI untuk memahami konten studi DICOM tanpa harus memuat data gambar. Gambar dimuat lebih cepat tanpa kompromi dalam kualitas gambar berkat kompresi gambar tingkat lanjut. Resolusi progresif memungkinkan pemuatan gambar yang lebih cepat untuk thumbnail, wilayah yang diminati, dan perangkat seluler beresolusi rendah.

Impor DICOM yang dapat diskalakan

HealthImaging Impor AWS memanfaatkan teknologi cloud native modern untuk mengimpor beberapa studi DICOM secara paralel. Arsip historis dapat diimpor dengan cepat tanpa memengaruhi beban kerja klinis untuk data baru. Untuk informasi tentang instans SOP yang didukung dan sintaks transfer, lihat. DICOM

Hirarki data DICOM yang dikelola layanan

AWS HealthImaging secara otomatis mengatur data DICOM P10 pada impor oleh elemen data DICOM tingkat Pasien, Studi, dan Seri. Layanan ini mengatur data DICOM ini ke dalam kumpulan gambar yang sesuai dengan seri DICOM, menyederhanakan alur kerja pasca impor. Organisasi tingkat Studi dan Seri dipertahankan saat data baru diimpor.

DICOMweb Kompatibilitas API

AWS HealthImaging menawarkan DICOMweb kesesuaian APIs untuk menyederhanakan integrasi dan memungkinkan interoperabilitas dengan aplikasi yang ada. Layanan ini juga menawarkan cloud-native APIs yang mengaktifkan tindakan yang tidak didukung oleh DICOMweb standar, seperti operasi pembaruan metadata.

AWS Layanan terkait

AWS HealthImaging memiliki integrasi yang ketat dengan AWS layanan lain. Pengetahuan tentang layanan berikut berguna untuk memanfaatkan sepenuhnya HealthImaging.

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

Mengakses AWS HealthImaging

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

Konsol Manajemen AWS

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

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.

Permintaan HTTP

Anda dapat memanggil HealthImaging tindakan menggunakan permintaan HTTP, tetapi Anda harus menentukan titik akhir yang berbeda tergantung pada jenis tindakan yang digunakan. Untuk informasi selengkapnya, lihat Tindakan API yang didukung untuk permintaan HTTP.

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

Untuk informasi tentang kepatuhan, lihatValidasi kepatuhan untuk AWS HealthImaging.

Harga

HealthImaging membantu Anda mengotomatiskan manajemen siklus hidup data klinis dengan tingkatan cerdas. Untuk informasi selengkapnya, lihat Pengoptimalan Biaya.

Untuk informasi harga umum, lihat HealthImaging harga AWS. Untuk memperkirakan biaya, gunakan kalkulator HealthImaging harga AWS.

Memulai AWS HealthImaging

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

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

Topik

- HealthImaging Konsep AWS
- Menyiapkan AWS HealthImaging
- AWS HealthImaging tutorial

HealthImaging Konsep AWS

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

Konsep

- Penyimpanan data
- Set gambar
- Metadata
- Bingkai gambar

Penyimpanan data

Penyimpanan data adalah gudang data pencitraan medis yang berada dalam satu. Wilayah AWS Sebuah AWS akun dapat memiliki nol atau banyak penyimpanan data. Penyimpanan data memiliki kunci AWS KMS enkripsi sendiri, sehingga data dalam satu penyimpanan data dapat secara fisik dan logis diisolasi dari data di penyimpanan data lain. Penyimpanan data mendukung kontrol akses menggunakan peran IAM, izin, dan kontrol akses berbasis atribut.

Untuk informasi selengkapnya, lihat Mengelola penyimpanan data dan Pengoptimalan Biaya.

Set gambar

Kumpulan gambar adalah AWS konsep yang mendefinisikan mekanisme pengelompokan abstrak untuk mengoptimalkan data pencitraan medis terkait. Saat Anda mengimpor data pencitraan DICOM P10 ke penyimpanan HealthImaging data AWS, data tersebut diubah menjadi kumpulan gambar yang terdiri dari metadata dan bingkai gambar (data piksel). HealthImaging upaya untuk mengatur data yang diimpor sesuai dengan hierarki DICOM Studi, Seri, dan Instance. Instance DICOM yang berhasil ditambahkan ke hierarki HealthImaging terkelola dilambangkan sebagai kumpulan gambar utama. Mengimpor data DICOM P10 akan: membuat kumpulan gambar primer baru; menggabungkan instance ke dalam kumpulan gambar utama yang ada jika instance sudah ada dalam koleksi primer; atau, dalam kasus konflik elemen metadata, buat kumpulan gambar non-primer baru.

Untuk informasi selengkapnya, lihat Mengimpor data pencitraan dan Memahami set gambar.

Metadata

Metadata adalah atribut non-piksel yang ada dalam kumpulan gambar. Untuk DICOM, ini termasuk demografi pasien, detail prosedur, dan parameter khusus akuisisi lainnya. AWS HealthImaging memisahkan gambar yang disetel menjadi metadata dan bingkai gambar (data piksel) sehingga aplikasi dapat mengaksesnya dengan cepat. Ini berguna untuk pemirsa gambar, analitik, dan kasus AI/ML penggunaan yang tidak memerlukan data piksel. Data DICOM dinormalisasi pada tingkat Pasien, Studi, dan Seri, menghilangkan inkonsistensi. Ini menyederhanakan penggunaan data, meningkatkan keamanan, dan meningkatkan kinerja akses.

Untuk informasi selengkapnya, lihat Mendapatkan metadata set gambar dan Normalisasi metadata.

Bingkai gambar

Bingkai gambar adalah data piksel yang ada dalam gambar yang diatur untuk membuat gambar medis 2D. Beberapa file mempertahankan pengkodean sintaks transfer aslinya selama impor, sementara yang lain ditranskode. Penyimpanan data Amazon Web Services dapat dikonfigurasi untuk mentranskode bingkai gambar lossless ke lossless JPEG 2000 (K) High-Throughput JPEG 2000 (HTJ2K) atau lossless JPEG 2000. Jika bingkai gambar dikodekan dalam HTJ2 K atau JPEG 2000, itu harus diterjemahkan sebelum dilihat di penampil gambar. Lihat informasi selengkapnya di Sintaks transfer yang didukung, Mendapatkan data piksel set gambar, dan Pustaka decoding bingkai gambar.

Menyiapkan AWS HealthImaging

Anda harus mengatur AWS lingkungan Anda sebelum menggunakan AWS HealthImaging. Topik berikut adalah prasyarat untuk tutorial yang terletak di bagian berikutnya.

Topik

- Mendaftar untuk Akun AWS
- Buat pengguna dengan akses administratif
- Buat ember S3
- Buat penyimpanan data
- Buat pengguna IAM dengan izin akses HealthImaging penuh
- Buat peran IAM untuk impor
- Instal AWS CLI (opsional)

Mendaftar untuk Akun AWS

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar untuk Akun AWS

1. Buka https://portal.aws.amazon.com/billing/pendaftaran.
2. Ikuti petunjuk online.

   Bagian dari prosedur pendaftaran melibatkan menerima panggilan telepon atau pesan teks dan memasukkan kode verifikasi pada keypad telepon.

   Saat Anda mendaftar untuk sebuah Akun AWS, sebuah Pengguna root akun AWSdibuat. Pengguna root memiliki akses ke semua Layanan AWS dan sumber daya di akun. Sebagai praktik keamanan terbaik, tetapkan akses administratif ke pengguna, dan gunakan hanya pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

AWS mengirimi Anda email konfirmasi setelah proses pendaftaran selesai. Kapan saja, Anda dapat melihat aktivitas akun Anda saat ini dan mengelola akun Anda dengan masuk https://aws.amazon.comke/ dan memilih Akun Saya.

Buat pengguna dengan akses administratif

Setelah Anda mendaftar Akun AWS, amankan Pengguna root akun AWS, aktifkan AWS IAM Identity Center, dan buat pengguna administratif sehingga Anda tidak menggunakan pengguna root untuk tugas sehari-hari.

Amankan Anda Pengguna root akun AWS

1. Masuk ke Konsol Manajemen AWSsebagai pemilik akun dengan memilih pengguna Root dan memasukkan alamat Akun AWS email Anda. Di laman berikutnya, masukkan kata sandi.

   Untuk bantuan masuk dengan menggunakan pengguna root, lihat Masuk sebagai pengguna root di AWS Sign-In Panduan Pengguna.
2. Mengaktifkan autentikasi multi-faktor (MFA) untuk pengguna root Anda.

   Untuk petunjuk, lihat Mengaktifkan perangkat MFA virtual untuk pengguna Akun AWS root (konsol) Anda di Panduan Pengguna IAM.

Buat pengguna dengan akses administratif

1. Aktifkan Pusat Identitas IAM.

   Untuk mendapatkan petunjuk, silakan lihat Mengaktifkan AWS IAM Identity Center di Panduan Pengguna AWS IAM Identity Center .
2. Di Pusat Identitas IAM, berikan akses administratif ke pengguna.

   Untuk tutorial tentang menggunakan Direktori Pusat Identitas IAM sebagai sumber identitas Anda, lihat Mengkonfigurasi akses pengguna dengan default Direktori Pusat Identitas IAM di Panduan AWS IAM Identity Center Pengguna.

Masuk sebagai pengguna dengan akses administratif

- Untuk masuk dengan pengguna Pusat Identitas IAM, gunakan URL masuk yang dikirim ke alamat email saat Anda membuat pengguna Pusat Identitas IAM.

  Untuk bantuan masuk menggunakan pengguna Pusat Identitas IAM, lihat Masuk ke portal AWS akses di Panduan AWS Sign-In Pengguna.

Tetapkan akses ke pengguna tambahan

1. Di Pusat Identitas IAM, buat set izin yang mengikuti praktik terbaik menerapkan izin hak istimewa paling sedikit.

   Untuk petunjuknya, lihat Membuat set izin di Panduan AWS IAM Identity Center Pengguna.
2. Tetapkan pengguna ke grup, lalu tetapkan akses masuk tunggal ke grup.

   Untuk petunjuk, lihat Menambahkan grup di Panduan AWS IAM Identity Center Pengguna.

Buat ember S3

Untuk mengimpor data DICOM P10 ke AWS HealthImaging, dua bucket Amazon S3 direkomendasikan. Bucket input Amazon S3 menyimpan data DICOM P10 untuk diimpor dan HealthImaging dibaca dari bucket ini. Bucket keluaran Amazon S3 menyimpan hasil pemrosesan pekerjaan impor dan HealthImaging menulis ke bucket ini. Untuk representasi visual dari ini, lihat diagram diMemahami pekerjaan impor.

> **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 dalam peran IAM untuk impor.

- 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

Saat Anda mengimpor data pencitraan medis, penyimpanan HealthImaging data AWS menyimpan hasil file DICOM P10 Anda yang diubah, yang disebut kumpulan gambar. Untuk representasi visual dari ini, lihat diagram diMemahami pekerjaan impor.

> **Tip**
>
> A `datastoreID` dihasilkan saat Anda membuat penyimpanan data. Anda harus menggunakan `datastoreID` saat menyelesaikan trust relationship for import nanti di bagian ini.

Untuk membuat penyimpanan data, lihatMembuat penyimpanan data.

Buat pengguna IAM dengan izin akses HealthImaging penuh

> **Praktik terbaik**
>
> Kami menyarankan Anda membuat pengguna IAM terpisah untuk kebutuhan yang berbeda seperti mengimpor, akses data, dan manajemen data. Ini sejalan dengan akses hak istimewa paling sedikit Grant di Well-Architected AWS Framework.
> Untuk keperluan Tutorial di bagian selanjutnya, Anda akan menggunakan satu pengguna IAM.

Untuk membuat pengguna IAM

1. Ikuti petunjuk untuk Membuat pengguna IAM di AWS akun Anda di Panduan Pengguna IAM. Pertimbangkan penamaan pengguna `ahiadmin` (atau yang serupa) untuk tujuan klarifikasi.
2. Tetapkan kebijakan `AWSHealthImagingFullAccess` terkelola ke pengguna IAM. Untuk informasi selengkapnya, lihat AWS kebijakan terkelola: AWSHealth ImagingFullAccess.

   > **Note**
   >
   > Izin IAM dapat dipersempit. Untuk informasi selengkapnya, lihat AWS kebijakan terkelola untuk AWS HealthImaging.

Buat peran IAM untuk impor

> **Note**
>
> Petunjuk berikut mengacu pada peran AWS Identity and Access Management (IAM) yang memberikan akses baca dan tulis ke bucket Amazon S3 untuk mengimpor data DICOM Anda. Meskipun peran diperlukan untuk tutorial di bagian berikutnya, kami sarankan Anda menambahkan izin IAM ke pengguna, grup, dan peran yang menggunakanAWS kebijakan terkelola untuk AWS HealthImaging, karena mereka lebih mudah digunakan daripada menulis kebijakan sendiri.

Sebuah peran IAM adalah identitas IAM yang dapat Anda buat di akun yang memiliki izin tertentu. Untuk memulai pekerjaan impor, peran IAM yang memanggil `StartDICOMImportJob` tindakan harus dilampirkan ke kebijakan pengguna yang memberikan akses ke bucket Amazon S3 yang digunakan untuk membaca data DICOM P10 Anda dan menyimpan hasil pemrosesan pekerjaan impor. Itu juga harus diberi hubungan kepercayaan (kebijakan) yang memungkinkan AWS HealthImaging untuk mengambil peran tersebut.

Untuk membuat peran IAM untuk tujuan impor

1. Menggunakan Konsol IAM, buat peran bernama`ImportJobDataAccessRole`. Anda menggunakan peran ini untuk tutorial di bagian berikutnya. Untuk informasi selengkapnya, lihat Membuat peran IAM di Panduan Pengguna IAM.

   > **Tip**
   >
   > Untuk keperluan panduan ini, contoh kode Memulai pekerjaan impor mengacu pada peran `ImportJobDataAccessRole` IAM.

2. Lampirkan kebijakan izin IAM ke peran IAM. Kebijakan izin ini memberikan akses ke bucket input dan output Amazon S3. Lampirkan kebijakan izin berikut ke peran `ImportJobDataAccessRole` IAM.

   JSON

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

```
    "Statement": [
        {
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-source-bucket",
                "arn:aws:s3:::amzn-s3-demo-logging-bucket"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject"
            ],
            "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"
        }
    ]
}
```

3. Lampirkan hubungan kepercayaan (kebijakan) berikut ke peran `ImportJobDataAccessRole` IAM. Kebijakan kepercayaan mensyaratkan `datastoreId` yang dihasilkan saat Anda menyelesaikan bagian tersebutBuat penyimpanan data. Tutorial yang mengikuti topik ini mengasumsikan Anda menggunakan satu penyimpanan HealthImaging data AWS, tetapi dengan bucket Amazon S3 khusus penyimpanan data, peran IAM, dan kebijakan kepercayaan.

> **Note**
>
> `Condition`Pemblokiran dalam kebijakan kepercayaan ini membantu mencegah masalah deputi yang membingungkan dengan memastikan bahwa hanya penyimpanan

HealthImaging data AWS spesifik Anda yang dapat diakses. Untuk informasi lebih lanjut tentang tindakan pengamanan ini, lihat Pencegahan Deputi Bingung Lintas Layanan di HealthImaging.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "medical-imaging.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Untuk mempelajari selengkapnya tentang membuat dan menggunakan kebijakan IAM dengan AWS HealthImaging, lihatIdentity and Access Management untuk AWS HealthImaging.

Untuk mempelajari selengkapnya tentang peran IAM secara umum, lihat peran IAM di Panduan Pengguna IAM. Untuk mempelajari selengkapnya tentang kebijakan dan izin IAM secara umum, lihat Kebijakan dan Izin IAM di Panduan Pengguna IAM.

Instal AWS CLI (opsional)

Prosedur berikut diperlukan jika Anda menggunakan AWS Command Line Interface. Jika Anda menggunakan Konsol Manajemen AWS atau AWS SDKs, Anda dapat melewati prosedur berikut.

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 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 medical-imaging help
```

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

AWS HealthImaging tutorial

Tujuan

Tujuan dari tutorial ini adalah untuk mengimpor binari (`.dcm`file) DICOM P10 ke dalam penyimpanan HealthImaging data AWS dan mengubahnya menjadi kumpulan gambar yang terdiri dari metadata dan bingkai gambar (data piksel). Setelah mengimpor data DICOM, Anda menggunakan tindakan bawaan HealthImaging cloud untuk mengakses kumpulan gambar, metadata, dan bingkai gambar berdasarkan preferensi akses Anda.

Prasyarat

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

Langkah-langkah tutorial

1. Mulai pekerjaan impor
2. Dapatkan properti pekerjaan impor
3. Cari set gambar

4. Dapatkan properti set gambar
5. Dapatkan metadata set gambar
6. Dapatkan data piksel set gambar
7. Hapus penyimpanan data

Mengelola penyimpanan data dengan AWS HealthImaging

Dengan AWS HealthImaging, Anda membuat dan mengelola penyimpanan data untuk sumber daya gambar medis. Topik berikut menjelaskan cara menggunakan tindakan HealthImaging cloud native untuk membuat, mendeskripsikan, membuat daftar, dan menghapus penyimpanan data menggunakan Konsol Manajemen AWS, AWS CLI, dan AWS SDKs.

> **Note**
>
> Topik terakhir dalam artikel ini adalah tentang optimasi biaya. Setelah Anda mengimpor data pencitraan medis Anda ke penyimpanan HealthImaging data, secara otomatis bergerak di antara dua tingkatan penyimpanan berdasarkan waktu dan penggunaan. Tingkat penyimpanan memiliki tingkat harga yang berbeda, jadi penting untuk memahami proses pergerakan tingkat dan HealthImaging sumber daya yang diakui untuk tujuan penagihan.

Topik

Membuat penyimpanan data

Gunakan `CreateDatastore` tindakan untuk membuat penyimpanan HealthImaging data AWS untuk mengimpor file DICOM P10. Menu berikut memberikan prosedur untuk contoh Konsol Manajemen AWS dan kode untuk AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat `CreateDatastore`di AWS HealthImaging API Referensi. Saat membuat penyimpanan data, Anda dapat memilih sintaks transfer default yang HealthImaging digunakan AWS untuk mentranskode dan menyimpan bingkai gambar lossless. Konfigurasi ini tidak dapat diubah setelah penyimpanan data dibuat.

Throughput Tinggi JPEG 2000 (HTJ2K)

HTJ2K (High Throughput JPEG 2000) adalah format penyimpanan default untuk HealthImaging datastores. Ini adalah perpanjangan dari standar JPEG 2000 yang menawarkan kinerja pengkodean

dan decoding yang ditingkatkan secara signifikan. Saat Anda membuat datastore tanpa menentukan a–`lossless-storage-format`, HealthImaging secara otomatis menggunakan K. HTJ2 Lihat AWS CLI dan SDKs bagian di bawah ini untuk membuat penyimpanan data menggunakan K. HTJ2

JPEG 2000 Tanpa Kehilangan

Pengkodean Lossless JPEG 2000 memungkinkan pembuatan datastores yang bertahan dan mengambil bingkai gambar lossless dalam format JPEG 2000 tanpa transcoding, memungkinkan pengambilan latensi yang lebih rendah untuk aplikasi yang memerlukan JPEG 2000 Lossless (DICOM Transfer Syntax UID 1.2.840.10008.1.2.4.90) lihat untuk lebih jelasnya. Sintaks transfer yang didukung Lihat AWS CLI dan SDKs bagian di bawah ini untuk membuat penyimpanan data menggunakan format lossless JPEG 2000.

> **Penting:**
>
> - Jangan beri nama penyimpanan data dengan informasi kesehatan yang dilindungi (PHI), informasi identitas pribadi (PII), atau informasi rahasia atau sensitif lainnya.
> - AWS Console mendukung pembuatan penyimpanan data dengan pengaturan default. Gunakan AWS CLI atau AWS SDK untuk membuat penyimpanan data dengan opsional yang ditentukan. `–lossless-storage-format`

Untuk membuat penyimpanan data

Pilih menu berdasarkan preferensi akses Anda ke AWS HealthImaging.

AWS Konsol

1. Buka HealthImaging konsol Buat halaman penyimpanan data.
2. Di bawah Detail, untuk nama penyimpanan data, masukkan nama untuk penyimpanan data Anda.
3. Di bawah Enkripsi data, pilih AWS KMS kunci untuk mengenkripsi sumber daya Anda. Untuk informasi selengkapnya, lihat Perlindungan data di AWS HealthImaging.
4. Di bawah Tag - opsional, Anda dapat menambahkan tag ke penyimpanan data Anda saat Anda membuatnya. Untuk informasi selengkapnya, lihat Menandai sumber daya.
5. Pilih Buat penyimpanan data.

AWS CLI dan SDKs

Bash

AWS CLI dengan skrip Bash

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function imaging_create_datastore
#
# This function creates an AWS HealthImaging data store for importing DICOM P10
 files.
#
# Parameters:
#       -n data_store_name - The name of the data store.
#
# Returns:
#       The datastore ID.
#    And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function imaging_create_datastore() {
  local datastore_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function imaging_create_datastore"
    echo "Creates an AWS HealthImaging data store for importing DICOM P10 files."
    echo "  -n data_store_name - The name of the data store."
    echo ""
  }

  # Retrieve the calling parameters.
```

```
  while getopts "n:h" option; do
    case "${option}" in
      n) datastore_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$datastore_name" ]]; then
    errecho "ERROR: You must provide a data store name with the -n parameter."
    usage
    return 1
  fi

  response=$(aws medical-imaging create-datastore \
    --datastore-name "$datastore_name" \
    --output text \
    --query 'datastoreId')

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports medical-imaging create-datastore operation
 failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

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

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

CLI

AWS CLI

Contoh 1: Untuk membuat penyimpanan data

Contoh `create-datastore` kode berikut membuat penyimpanan data dengan nama`my-datastore`. Saat Anda membuat datastore tanpa menentukan a`--lossless-storage-format`, AWS HealthImaging default ke HTJ2 K (High Throughput JPEG 2000).

```
aws medical-imaging create-datastore \
    --datastore-name "my-datastore"
```

Output:

```
{
    "datastoreId": "12345678901234567890123456789012",
    "datastoreStatus": "CREATING"
}
```

Contoh 2: Untuk membuat penyimpanan data dengan format penyimpanan Lossless JPEG 2000

Penyimpanan data yang dikonfigurasi dengan format penyimpanan Lossless JPEG 2000 akan mentranskode dan mempertahankan bingkai gambar lossless dalam format JPEG 2000. Bingkai gambar kemudian dapat diambil dalam JPEG 2000 Lossless tanpa transcoding. Contoh `create-datastore` kode berikut membuat penyimpanan data dikonfigurasi untuk format penyimpanan Lossless JPEG 2000 dengan nama. `my-datastore`

```
aws medical-imaging create-datastore \
    --datastore-name "my-datastore" \
    --lossless-storage-format JPEG_2000_LOSSLESS
```

Output:

```
{
    "datastoreId": "12345678901234567890123456789012",
    "datastoreStatus": "CREATING"
}
```

Untuk informasi selengkapnya, lihat Membuat penyimpanan data di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

```
   public static String createMedicalImageDatastore(MedicalImagingClient
medicalImagingClient,
           String datastoreName) {
       try {
           CreateDatastoreRequest datastoreRequest =
CreateDatastoreRequest.builder()
                   .datastoreName(datastoreName)
                   .build();
           CreateDatastoreResponse response =
medicalImagingClient.createDatastore(datastoreRequest);
           return response.datastoreId();
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return "";
   }
```

- Untuk detail API, lihat CreateDatastoredi Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

```
import { CreateDatastoreCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreName - The name of the data store to create.
 */
export const createDatastore = async (datastoreName = "DATASTORE_NAME") => {
  const response = await medicalImagingClient.send(
    new CreateDatastoreCommand({ datastoreName: datastoreName }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //       httpStatusCode: 200,
  //       requestId: 'a71cd65f-2382-49bf-b682-f9209d8d399b',
  //       extendedRequestId: undefined,
  //       cfId: undefined,
  //       attempts: 1,
  //       totalRetryDelay: 0
  //    },
  //    datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //    datastoreStatus: 'CREATING'
  // }
  return response;
};
```

- Untuk detail API, lihat CreateDatastoredi Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def create_datastore(self, name):
        """
        Create a data store.

        :param name: The name of the data store to create.
        :return: The data store ID.
        """
        try:
            data_store =
 self.health_imaging_client.create_datastore(datastoreName=name)
        except ClientError as err:
            logger.error(
                "Couldn't create data store %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return data_store["datastoreId"]
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Untuk detail API, lihat CreateDatastoredi 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

```
    TRY.
        " iv_datastore_name = 'my-datastore-name'
        oo_result = lo_mig->createdatastore( iv_datastorename =
iv_datastore_name ).
        DATA(lv_datastore_id) = oo_result->get_datastoreid( ).
        MESSAGE 'Data store created.' TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict. Data store may already exist.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migservicequotaexcdex.
        MESSAGE 'Service quota exceeded.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

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

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

Ketersediaan contoh

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

Mendapatkan properti penyimpanan data

Gunakan `GetDatastore` tindakan untuk mengambil properti penyimpanan HealthImaging data AWS. Menu berikut memberikan prosedur untuk contoh Konsol Manajemen AWS dan kode untuk AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat `GetDatastore`di AWS HealthImaging API Referensi.

Untuk mendapatkan properti penyimpanan data

Pilih menu berdasarkan preferensi akses Anda ke AWS HealthImaging.

AWS Konsol

1. Buka halaman penyimpanan data HealthImaging konsol.
2. Pilih penyimpanan data.

   Halaman detail penyimpanan data terbuka. Di bawah bagian Detail, semua properti penyimpanan data tersedia. Untuk melihat kumpulan gambar terkait, impor, dan tag, pilih tab yang berlaku.

AWS CLI dan SDKs

Bash

AWS CLI dengan skrip Bash

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}
```

```
##############################################################################
# function imaging_get_datastore
#
# Get a data store's properties.
#
# Parameters:
#       -i data_store_id - The ID of the data store.
#
# Returns:
#       [datastore_name, datastore_id, datastore_status, datastore_arn,
 created_at, updated_at]
#    And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function imaging_get_datastore() {
  local datastore_id option OPTARG # Required to use getopts command in a
 function.
  local error_code
  # bashsupport disable=BP5008
  function usage() {
    echo "function imaging_get_datastore"
    echo "Gets a data store's properties."
    echo "  -i datastore_id - The ID of the data store."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) datastore_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1
```

```
  if [[ -z "$datastore_id" ]]; then
    errecho "ERROR: You must provide a data store ID with the -i parameter."
    usage
    return 1
  fi

  local response

  response=$(
    aws medical-imaging get-datastore \
      --datastore-id "$datastore_id" \
      --output text \
      --query "[ datastoreProperties.datastoreName,
 datastoreProperties.datastoreId, datastoreProperties.datastoreStatus,
 datastoreProperties.datastoreArn,  datastoreProperties.createdAt,
 datastoreProperties.updatedAt]"
  )
  error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports list-datastores operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

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

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

CLI

AWS CLI

Contoh 1: Untuk mendapatkan properti penyimpanan data

Contoh `get-datastore` kode berikut mendapatkan properti penyimpanan data.

```
aws medical-imaging get-datastore \
    --datastore-id 12345678901234567890123456789012
```

Output:

```
{
    "datastoreProperties": {
        "datastoreId": "12345678901234567890123456789012",
        "datastoreName": "TestDatastore123",
        "datastoreStatus": "ACTIVE",
        "losslessStorageFormat": "HTJ2K"
        "datastoreArn": "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012",
        "createdAt": "2022-11-15T23:33:09.643000+00:00",
        "updatedAt": "2022-11-15T23:33:09.643000+00:00"
    }
}
```

Contoh 2: Untuk mendapatkan properti penyimpanan data yang dikonfigurasi untuk JPEG2 000

Contoh `get-datastore` kode berikut mendapatkan properti penyimpanan data untuk penyimpanan data yang dikonfigurasi untuk format penyimpanan Lossless JPEG 2000.

```
aws medical-imaging get-datastore \
    --datastore-id 12345678901234567890123456789012
```

Output:

```
{
    "datastoreProperties": {
        "datastoreId": "12345678901234567890123456789012",
```

```
        "datastoreName": "TestDatastore123",
        "datastoreStatus": "ACTIVE",
        "losslessStorageFormat": "JPEG_2000_LOSSLESS",
        "datastoreArn": "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012",
        "createdAt": "2022-11-15T23:33:09.643000+00:00",
        "updatedAt": "2022-11-15T23:33:09.643000+00:00"
    }
}
```

Untuk informasi selengkapnya, lihat Mendapatkan properti penyimpanan data di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

```
   public static DatastoreProperties
getMedicalImageDatastore(MedicalImagingClient medicalImagingClient,
           String datastoreID) {
       try {
           GetDatastoreRequest datastoreRequest = GetDatastoreRequest.builder()
                   .datastoreId(datastoreID)
                   .build();
           GetDatastoreResponse response =
medicalImagingClient.getDatastore(datastoreRequest);
           return response.datastoreProperties();
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return null;
   }
```

- Untuk detail API, lihat GetDatastoredi Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

```
import { GetDatastoreCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreID - The ID of the data store.
 */
export const getDatastore = async (datastoreID = "DATASTORE_ID") => {
  const response = await medicalImagingClient.send(
    new GetDatastoreCommand({ datastoreId: datastoreID }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //       httpStatusCode: 200,
  //       requestId: '55ea7d2e-222c-4a6a-871e-4f591f40cadb',
  //       extendedRequestId: undefined,
  //       cfId: undefined,
  //       attempts: 1,
  //       totalRetryDelay: 0
  //    },
  //   datastoreProperties: {
  //        createdAt: 2023-08-04T18:50:36.239Z,
  //         datastoreArn: 'arn:aws:medical-imaging:us-
east-1:xxxxxxxxx:datastore/xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //         datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //         datastoreName: 'my_datastore',
  //         datastoreStatus: 'ACTIVE',
  //         updatedAt: 2023-08-04T18:50:36.239Z
  //   }
  // }
  return response.datastoreProperties;
};
```

- Untuk detail API, lihat GetDatastoredi Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def get_datastore_properties(self, datastore_id):
        """
        Get the properties of a data store.

        :param datastore_id: The ID of the data store.
        :return: The data store properties.
        """
        try:
            data_store = self.health_imaging_client.get_datastore(
                datastoreId=datastore_id
            )
        except ClientError as err:
            logger.error(
                "Couldn't get data store %s. Here's why: %s: %s",
                id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return data_store["datastoreProperties"]
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
client = boto3.client("medical-imaging")
medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Untuk detail API, lihat GetDatastoredi 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

```
TRY.
    " iv_datastore_id = '12345678901234567890123456789012345678901234567890'
    oo_result = lo_mig->getdatastore( iv_datastoreid = iv_datastore_id ).
    DATA(lo_properties) = oo_result->get_datastoreproperties( ).
    DATA(lv_name) = lo_properties->get_datastorename( ).
    DATA(lv_status) = lo_properties->get_datastorestatus( ).
    MESSAGE 'Data store properties retrieved.' TYPE 'I'.
  CATCH /aws1/cx_migaccessdeniedex.
    MESSAGE 'Access denied.' TYPE 'I'.
  CATCH /aws1/cx_miginternalserverex.
    MESSAGE 'Internal server error.' TYPE 'I'.
  CATCH /aws1/cx_migresourcenotfoundex.
    MESSAGE 'Data store not found.' TYPE 'I'.
  CATCH /aws1/cx_migthrottlingex.
    MESSAGE 'Request throttled.' TYPE 'I'.
  CATCH /aws1/cx_migvalidationex.
    MESSAGE 'Validation error.' TYPE 'I'.
ENDTRY.
```

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

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

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

Menyimpan data daftar

Gunakan `ListDatastores` tindakan untuk mencantumkan penyimpanan data yang tersedia di AWS HealthImaging. Menu berikut memberikan prosedur untuk contoh Konsol Manajemen AWS dan kode untuk AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat `ListDatastores`di AWS HealthImaging API Referensi.

Untuk daftar penyimpanan data

Pilih menu berdasarkan preferensi akses Anda ke AWS HealthImaging.

AWS Konsol

- Buka halaman HealthImaging Console Data Stores.

  Semua penyimpanan data terdaftar di bawah bagian Penyimpanan data.

AWS CLI dan SDKs

Bash

AWS CLI dengan skrip Bash

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
```

```
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#############################################################################
# function imaging_list_datastores
#
# List the HealthImaging data stores in the account.
#
# Returns:
#       [[datastore_name, datastore_id, datastore_status]]
#    And:
#       0 - If successful.
#       1 - If it fails.
#############################################################################
function imaging_list_datastores() {
  local option OPTARG # Required to use getopts command in a function.
  local error_code
  # bashsupport disable=BP5008
  function usage() {
    echo "function imaging_list_datastores"
    echo "Lists the AWS HealthImaging data stores in the account."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "h" option; do
    case "${option}" in
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local response
  response=$(aws medical-imaging list-datastores \
    --output text \
```

```
    --query "datastoreSummaries[*][datastoreName, datastoreId, datastoreStatus]")
  error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports list-datastores operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

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

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

CLI

AWS CLI

Untuk daftar penyimpanan data

Contoh `list-datastores` kode berikut mencantumkan penyimpanan data yang tersedia.

```
aws medical-imaging list-datastores
```

Output:

```
{
    "datastoreSummaries": [
        {
            "datastoreId": "12345678901234567890123456789012",
            "datastoreName": "TestDatastore123",
            "datastoreStatus": "ACTIVE",
```

```
            "datastoreArn": "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012",
            "createdAt": "2022-11-15T23:33:09.643000+00:00",
            "updatedAt": "2022-11-15T23:33:09.643000+00:00"
        }
    ]
}
```

Untuk informasi selengkapnya, lihat Menyimpan penyimpanan data di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

```
   public static List<DatastoreSummary>
listMedicalImagingDatastores(MedicalImagingClient medicalImagingClient) {
       try {
           ListDatastoresRequest datastoreRequest =
ListDatastoresRequest.builder()
                   .build();
           ListDatastoresIterable responses =
medicalImagingClient.listDatastoresPaginator(datastoreRequest);
           List<DatastoreSummary> datastoreSummaries = new ArrayList<>();

           responses.stream().forEach(response ->
datastoreSummaries.addAll(response.datastoreSummaries()));

           return datastoreSummaries;
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return null;
   }
```

- Untuk detail API, lihat ListDatastoresdi Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

```
import { paginateListDatastores } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

export const listDatastores = async () => {
  const paginatorConfig = {
    client: medicalImagingClient,
    pageSize: 50,
  };

  const commandParams = {};
  const paginator = paginateListDatastores(paginatorConfig, commandParams);

  /**
   * @type {import("@aws-sdk/client-medical-imaging").DatastoreSummary[]}
   */
  const datastoreSummaries = [];
  for await (const page of paginator) {
    // Each page contains a list of `jobSummaries`. The list is truncated if is
 larger than `pageSize`.
    datastoreSummaries.push(...page.datastoreSummaries);
    console.log(page);
  }
  // {
  //   '$metadata': {
  //       httpStatusCode: 200,
  //       requestId: '6aa99231-d9c2-4716-a46e-edb830116fa3',
  //       extendedRequestId: undefined,
  //       cfId: undefined,
  //       attempts: 1,
  //       totalRetryDelay: 0
  //   },
  //   datastoreSummaries: [
```

```
  //     {
  //       createdAt: 2023-08-04T18:49:54.429Z,
  //       datastoreArn: 'arn:aws:medical-imaging:us-east-1:xxxxxxxxx:datastore/
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //       datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //       datastoreName: 'my_datastore',
  //       datastoreStatus: 'ACTIVE',
  //       updatedAt: 2023-08-04T18:49:54.429Z
  //     }
  //     ...
  //   ]
  // }

  return datastoreSummaries;
};
```

- Untuk detail API, lihat ListDatastoresdi Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def list_datastores(self):
        """
        List the data stores.

        :return: The list of data stores.
        """
        try:
```

```
            paginator =
self.health_imaging_client.get_paginator("list_datastores")
            page_iterator = paginator.paginate()
            datastore_summaries = []
            for page in page_iterator:
                datastore_summaries.extend(page["datastoreSummaries"])
        except ClientError as err:
            logger.error(
                "Couldn't list data stores. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return datastore_summaries
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Untuk detail API, lihat ListDatastoresdi 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

```
    TRY.
        oo_result = lo_mig->listdatastores( ).
        DATA(lt_datastores) = oo_result->get_datastoresummaries( ).
        DATA(lv_count) = lines( lt_datastores ).
        MESSAGE |Found { lv_count } data stores.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
```

```
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

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

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

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

Menghapus penyimpanan data

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

> **Note**
>
> Sebelum penyimpanan data dapat dihapus, Anda harus terlebih dahulu menghapus semua set gambar di dalamnya. Untuk informasi selengkapnya, lihat Menghapus set gambar.

Untuk menghapus penyimpanan data

Pilih menu berdasarkan preferensi akses Anda ke AWS HealthImaging.

AWS Konsol

1. Buka halaman penyimpanan data HealthImaging konsol.
2. Pilih penyimpanan data.
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 penyimpanan data.

AWS CLI dan SDKs

Bash

AWS CLI dengan skrip Bash

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function imaging_delete_datastore
#
# This function deletes an AWS HealthImaging data store.
#
# Parameters:
#       -i datastore_id - The ID of the data store.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function imaging_delete_datastore() {
  local datastore_id response
```

```
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function imaging_delete_datastore"
    echo "Deletes an AWS HealthImaging data store."
    echo "  -i datastore_id - The ID of the data store."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) datastore_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$datastore_id" ]]; then
    errecho "ERROR: You must provide a data store ID with the -i parameter."
    usage
    return 1
  fi

  response=$(aws medical-imaging delete-datastore \
    --datastore-id "$datastore_id")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports medical-imaging delete-datastore operation
failed.$response"
    return 1
  fi
```

```
  return 0
}
```

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

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

CLI

AWS CLI

Untuk menghapus penyimpanan data

Contoh `delete-datastore` kode berikut menghapus penyimpanan data.

```
aws medical-imaging delete-datastore \
    --datastore-id "12345678901234567890123456789012"
```

Output:

```
{
    "datastoreId": "12345678901234567890123456789012",
    "datastoreStatus": "DELETING"
}
```

Untuk informasi selengkapnya, lihat Menghapus penyimpanan data di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

```
   public static void deleteMedicalImagingDatastore(MedicalImagingClient
medicalImagingClient,
           String datastoreID) {
       try {
           DeleteDatastoreRequest datastoreRequest =
DeleteDatastoreRequest.builder()
                   .datastoreId(datastoreID)
                   .build();
           medicalImagingClient.deleteDatastore(datastoreRequest);
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Untuk detail API, lihat DeleteDatastoredi Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

```
import { DeleteDatastoreCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store to delete.
 */
export const deleteDatastore = async (datastoreId = "DATASTORE_ID") => {
  const response = await medicalImagingClient.send(
    new DeleteDatastoreCommand({ datastoreId }),
  );
```

```
  console.log(response);
  // {
  //   '$metadata': {
  //           httpStatusCode: 200,
  //           requestId: 'f5beb409-678d-48c9-9173-9a001ee1ebb1',
  //           extendedRequestId: undefined,
  //           cfId: undefined,
  //           attempts: 1,
  //           totalRetryDelay: 0
  //         },
  //     datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //     datastoreStatus: 'DELETING'
  // }

  return response;
};
```

- Untuk detail API, lihat DeleteDatastoredi Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def delete_datastore(self, datastore_id):
        """
        Delete a data store.

        :param datastore_id: The ID of the data store.
        """
        try:
```

```
            self.health_imaging_client.delete_datastore(datastoreId=datastore_id)
        except ClientError as err:
            logger.error(
                "Couldn't delete data store %s. Here's why: %s: %s",
                datastore_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
client = boto3.client("medical-imaging")
medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Untuk detail API, lihat DeleteDatastoredi 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

```
TRY.
    " iv_datastore_id = '12345678901234567890123456789012345678901234567890'
    oo_result = lo_mig->deletedatastore( iv_datastoreid = iv_datastore_id ).
    MESSAGE 'Data store deleted.' TYPE 'I'.
  CATCH /aws1/cx_migaccessdeniedex.
    MESSAGE 'Access denied.' TYPE 'I'.
  CATCH /aws1/cx_migconflictexception.
    MESSAGE 'Conflict. Data store may contain resources.' TYPE 'I'.
  CATCH /aws1/cx_miginternalserverex.
    MESSAGE 'Internal server error.' TYPE 'I'.
  CATCH /aws1/cx_migresourcenotfoundex.
    MESSAGE 'Data store not found.' TYPE 'I'.
```

```
    CATCH /aws1/cx_migthrottlingex.
      MESSAGE 'Request throttled.' TYPE 'I'.
    CATCH /aws1/cx_migvalidationex.
      MESSAGE 'Validation error.' TYPE 'I'.
  ENDTRY.
```

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

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

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

Menggunakan DICOMweb dengan AWS HealthImaging

Anda dapat mengambil objek DICOM dari AWS HealthImaging menggunakan representasi DICOMweb APIs, yang berbasis web APIs yang mengikuti standar DICOM untuk pencitraan medis. Fungsionalitas ini memungkinkan Anda untuk berinteraksi dengan sistem yang menggunakan binari DICOM Part 10 sekaligus memanfaatkan tindakan cloud native HealthImaging. Fokus dari Bab ini adalah bagaimana menggunakan HealthImaging implementasi DICOMweb APIs untuk mengembalikan DICOMweb tanggapan.

> **Penting:**
>
> HealthImaging menyimpan data DICOM sebagai kumpulan gambar. Gunakan tindakan HealthImaging cloud-native untuk mengelola dan mengambil set gambar. HealthImaging's DICOMweb APIs dapat digunakan untuk mengembalikan informasi kumpulan gambar dengan respons DICOMweb -conformant.
>
> Yang APIs tercantum dalam Bab ini dibangun sesuai dengan DICOMwebstandar untuk pencitraan medis berbasis web. Karena mereka adalah representasi dari DICOMweb APIs, mereka tidak ditawarkan melalui AWS CLI dan AWS SDKs.

Topik

Menyimpan instans dengan STOW-RS

AWS HealthImaging menawarkan representasi `DICOMweb STOW-RS` APIs untuk mengimpor data. Gunakan ini APIs untuk menyimpan data DICOM secara sinkron ke penyimpanan HealthImaging data Anda.

Tabel berikut menjelaskan HealthImaging representasi DICOMweb STOW-RS yang APIs tersedia untuk mengimpor data.

HealthImaging representasi dari STOW-RS DICOMweb APIs

Nama	Deskripsi
`StoreDICOM`	Menyimpan satu atau lebih instance ke penyimpanan HealthImaging data.
`StoreDICOMStudy`	Simpan satu atau beberapa instance yang sesuai dengan UID Instans Studi tertentu ke penyimpanan HealthImaging data.

Data yang diimpor dengan `StoreDICOMStudy` tindakan `StoreDICOM` dan akan diatur sebagai kumpulan gambar utama baru, atau ditambahkan ke kumpulan gambar utama yang ada, menggunakan logika yang sama dengan pekerjaan impor asinkron. Jika elemen metadata data DICOM P10 yang baru diimpor bertentangan dengan kumpulan gambar primer yang ada, data baru akan ditambahkan ke kumpulan gambar non-primer.

> **Note**
>
> - Tindakan ini mendukung pengunggahan hingga 1GB data DICOM per permintaan.
> - Respons API akan dalam format JSON, sesuai dengan standar STOW-RS. DICOMweb

Untuk memulai permintaan StoreDicom

1. Kumpulkan wilayah AWS Anda, HealthImaging `datastoreId`, dan nama file DICOM P10.
2. Membangun URL untuk permintaan formulir: `https://dicom-medical-imaging.`*`region`*`.amazonaws.com/datastore/`*`datastore-id`*`/studies`
3. Tentukan panjang konten file DICOM P10 menggunakan perintah pilihan Anda, misalnya. `$(stat -f %z $FILENAME)`
4. Siapkan dan kirim permintaan Anda. `StoreDICOM`menggunakan permintaan HTTP POST dengan protokol penandatanganan AWS Signature Version 4.

Example Contoh 1: Untuk menyimpan file DICOM P10 menggunakan tindakan **`StoreDICOM`**

Shell

```
curl -X POST -v \
  'https://dicom-medical-imaging.us-east-1.amazonaws.com/datastore/
d9a2a515ab294163a2d2f4069eed584c/studies' \
  --aws-sigv4 "aws:amz:$AWS_REGION:medical-imaging" \
  --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
  --header "x-amz-security-token:$AWS_SESSION_TOKEN" \
  --header "x-amz-content-sha256: STREAMING-AWS4-HMAC-SHA256-PAYLOAD" \
  --header "x-amz-decoded-content-length: $CONTENT_LENGTH" \
  --header 'Accept: application/dicom+json' \
  --header "Content-Type: application/dicom" \
  --upload-file $FILENAME
```

Example Contoh 2: Untuk menyimpan file DICOM P10 menggunakan tindakan **`StoreDICOMStudy`**

Satu-satunya perbedaan antara StoreDicom dan Store DICOMStudy adalah bahwa Study Instance UID diteruskan sebagai parameter ke StoreDICOMStudy, dan instance yang diunggah harus menjadi anggota studi yang ditentukan.

Shell

```
curl -X POST -v \
  'https://dicom-medical-imaging.us-east-1.amazonaws.com/datastore/
d9a2a515ab294163a2d2f4069eed584c/studies/1.3.6.1.4.1.5962.1.2.4.20040826285059.5457'
 \
  --aws-sigv4 "aws:amz:$AWS_REGION:medical-imaging" \
  --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
  --header "x-amz-security-token:$AWS_SESSION_TOKEN" \
  --header "x-amz-content-sha256: STREAMING-AWS4-HMAC-SHA256-PAYLOAD" \
  --header "x-amz-decoded-content-length: $CONTENT_LENGTH" \
  --header 'Accept: application/dicom+json' \
  --header "Content-Type: application/dicom" \
  --upload-file $FILENAME
```

Example Contoh 3: Untuk menyimpan file DICOM P10 dengan muatan HTTP multi-bagian

Beberapa file P10 dapat diunggah dengan satu tindakan unggahan multi-bagian. Perintah shell berikut menunjukkan cara merakit muatan multi-bagian yang berisi dua file P10, dan mengunggahnya dengan tindakan. `StoreDICOM`

Shell

```
#!/bin/sh
FILENAME=multipart.payload
BOUNDARY=2a8a02b9-0ed3-c8a7-7ebd-232427531940
boundary_str="--$BOUNDARY\r\n"
mp_header="${boundary_str}Content-Type: application/dicom\r\n\r\n"

##Encapsulate the binary DICOM file 1.
printf '%b' "$mp_header" > $FILENAME
cat file1.dcm >> $FILENAME

##Encapsulate the binary DICOM file 2 (note the additional CRLF before the part
 header).
printf '%b' "\r\n$mp_header" >> $FILENAME
cat file2.dcm >> $FILENAME

## Add the closing boundary.
printf '%b' "\r\n--$BOUNDARY--" >> $FILENAME

## Obain the payload size in bytes.
multipart_payload_size=$(stat -f%z "$FILENAME")

# Execute CURL POST request with AWS SIGv4
curl -X POST -v \
  'https://iad-dicom.external-healthlake-imaging.ai.aws.dev/datastore/
b5f34e91ca734b39a54ac11ea42416cf/studies' \
  --aws-sigv4 "aws:amz:us-east-1:medical-imaging" \
  --user "AKIAIOSFODNN7EXAMPLE:wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY" \
  --header "x-amz-content-sha256: STREAMING-AWS4-HMAC-SHA256-PAYLOAD" \
  --header "x-amz-decoded-content-length: ${multipart_payload_size}" \
  --header 'Accept: application/dicom+json' \
  --header "Content-Type: multipart/related; type=\"application/dicom\"; boundary=
\"${BOUNDARY}\"" \
  --data-binary "@$FILENAME"

# Delete the payload file
```

```
rm $FILENAME
```

Mengambil data DICOM dari HealthImaging

AWS HealthImaging menawarkan representasi DICOMweb WADO-RS APIs untuk mengambil data pada tingkat seri dan instans. Dengan ini APIs, dimungkinkan untuk mengambil semua metadata untuk seri DICOM dari penyimpanan data. HealthImaging Dimungkinkan juga untuk mengambil instance DICOM, metadata instance DICOM, dan frame instance DICOM (data piksel). HealthImagingDICOMweb WADO-RS APIs menawarkan fleksibilitas dalam cara Anda mengambil data yang disimpan HealthImaging dan menyediakan interoperabilitas dengan aplikasi lama.

> **Penting:**
>
> HealthImaging menyimpan data DICOM sebagai kumpulan gambar. Gunakan tindakan bawaan HealthImaging cloud untuk mengelola dan mengambil kumpulan gambar. HealthImaging's DICOMweb APIs dapat digunakan untuk mengembalikan informasi kumpulan gambar dengan respons DICOMweb -conformant.
> Yang APIs tercantum di bagian ini dibangun sesuai dengan standar DICOMweb (WADO-RS) untuk pencitraan medis berbasis web. Karena mereka adalah representasi dari DICOMweb APIs, mereka tidak ditawarkan melalui AWS CLI dan AWS SDKs.

Tabel berikut menjelaskan semua HealthImaging representasi DICOMweb WADO-RS yang APIs tersedia untuk mengambil data dari. HealthImaging

HealthImaging representasi WADO-RS DICOMweb APIs

Nama	Deskripsi
`GetDICOMSeriesMetadata`	Ambil metadata instance DICOM (`.json`file) untuk seri DICOM di penyimpanan HealthIma ging data dengan menentukan Studi dan Seri yang terkait dengan sumber daya. UIDs Lihat Ambil metadata seri.
`GetDICOMInstance`	Ambil instance DICOM (`.dcm`file) dari penyimpanan HealthImaging data dengan menentukan Seri, Studi, dan Instance yang

Nama	Deskripsi
	UIDs terkait dengan sumber daya. Lihat Mengambil sebuah instance.
`GetDICOMInstanceMetadata`	Ambil metadata instance DICOM (`.json`file) dari instance DICOM di penyimpanan HealthImaging data dengan menentukan Seri, Studi, dan Instance yang terkait dengan sumber daya. UIDs Lihat Mengambil metadata instans.
`GetDICOMInstanceFrames`	Ambil bingkai gambar tunggal atau batch (`multipart` permintaan) dari instance DICOM di penyimpanan HealthImaging data dengan menentukan UID Seri, UID Studi, Instance UIDs, dan nomor bingkai yang terkait dengan sumber daya. Lihat Ambil bingkai.

Topik

- Mendapatkan instance DICOM dari HealthImaging
- Mendapatkan metadata instans DICOM dari HealthImaging
- Mendapatkan metadata seri DICOM dari HealthImaging
- Mendapatkan bingkai instans DICOM dari HealthImaging
- Mendapatkan DICOM bulkdata dari HealthImaging

Mendapatkan instance DICOM dari HealthImaging

Gunakan `GetDICOMInstance` tindakan untuk mengambil instance DICOM (`.dcm`file) dari penyimpanan HealthImaging data dengan menentukan Seri, Studi, dan Instance yang UIDs terkait dengan sumber daya. API hanya akan mengembalikan instance dari kumpulan gambar utama kecuali parameter kumpulan gambar opsional disediakan. Anda dapat mengambil instance apa pun (dari kumpulan gambar primer atau non-primer) di penyimpanan data dengan menentukan `imageSetId` sebagai parameter kueri. Data DICOM dapat diambil baik dalam sintaks transfer yang disimpan atau sebagai format tidak terkompresi (ELE).

Untuk mendapatkan instance DICOM () **`.dcm`**

1. Kumpulkan HealthImaging `datastoreId` dan nilai `imageSetId` parameter.
2. Gunakan `GetImageSetMetadata`tindakan dengan nilai `imageSetId` parameter `datastoreId` dan untuk mengambil nilai metadata terkait untuk`studyInstanceUID`,, `seriesInstanceUID` dan. `sopInstanceUID` Untuk informasi selengkapnya, lihat Mendapatkan metadata set gambar.
3. Membangun URL untuk permintaan menggunakan nilai-nilai untuk`datastoreId`,,`studyInstanceUID`, `seriesInstanceUID``sopInstanceUID`, dan`imageSetId`. Untuk melihat seluruh jalur URL dalam contoh berikut, gulir ke atas tombol Salin. URL adalah dari bentuk:

   ```
   GET https://dicom-medical-imaging.region.amazonaws.com/datastore/datastore-id/
   studies/study-instance-uid/series/series-instance-uid/instances/sop-instance-uid?
   imageSetId=image-set-id
   ```

4. Siapkan dan kirim permintaan Anda. `GetDICOMInstance`menggunakan permintaan HTTP GET dengan protokol penandatanganan AWS Signature Version 4. Contoh kode berikut menggunakan alat baris `curl` perintah untuk mendapatkan instance DICOM (`.dcm`file) dari HealthImaging.

   Shell

   ```
   curl --request GET \
     'https://dicom-medical-imaging.us-east-1.amazonaws.com/datastore/
   d9a2a515ab294163a2d2f4069eed584c/
   studies/1.3.6.1.4.1.5962.1.2.4.20040826285059.5457/
   series/1.3.6.1.4.1.5962.1.3.4.1.20040825185059.5457/
   instances/1.3.6.1.4.1.5962.1.1.4.1.1.20040826186059.5457?
   imageSetId=459e50687f121185f747b67bb60d1bc8' \
     --aws-sigv4 'aws:amz:us-east-1:medical-imaging' \
     --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
     --header "x-amz-security-token:$AWS_SESSION_TOKEN" \
     --header 'Accept: application/dicom; transfer-syntax=1.2.840.10008.1.2.1' \
     --output 'dicom-instance.dcm'
   ```

> **Note**
>
> `transfer-syntax`UID bersifat opsional dan default ke Explicit VR Little Endian jika tidak disertakan. Sintaks transfer yang didukung meliputi:
>
> - Eksplisit VR Little Endian (ELE) - `1.2.840.10008.1.2.1` (default untuk bingkai gambar lossless)
> - Jika `transfer-syntax=*` kemudian bingkai gambar akan dikembalikan dalam sintaks transfer yang disimpan.
> - High-Throughput JPEG 2000 dengan Kompresi Gambar Opsi RPCL (Hanya Tanpa Lossless) - `1.2.840.10008.1.2.4.202` - jika instance disimpan sebagai HealthImaging `1.2.840.10008.1.2.4.202`
> - JPEG 2000 Lossless - `1.2.840.10008.1.2.4.90` - jika instance disimpan sebagai lossless. HealthImaging
> - JPEG Baseline (Proses 1): Sintaks Transfer Default untuk Kompresi Gambar 8-bit Lossy JPEG - - `1.2.840.10008.1.2.4.50` jika instance disimpan sebagai HealthImaging `1.2.840.10008.1.2.4.50`
> - Kompresi Gambar JPEG 2000 `1.2.840.10008.1.2.4.91` - - jika instance disimpan HealthImaging sebagai `1.2.840.10008.1.2.4.91`
> - Kompresi Gambar JPEG 2000 High-Throughput `1.2.840.10008.1.2.4.203` - - jika instance disimpan sebagai HealthImaging `1.2.840.10008.1.2.4.203`
> - Kompresi Gambar JPEG XL `1.2.840.10008.1.2.4.112` - - jika instance disimpan sebagai HealthImaging `1.2.840.10008.1.2.4.112`
> - Contoh yang disimpan HealthImaging dengan satu atau lebih bingkai gambar yang dikodekan dalam keluarga MPEG Transfer Syntax (yang mencakup MPEG2, MPEG-4 AVC/H.264 and HEVC/H .265) dapat diambil dengan UID sintaks transfer yang sesuai. Misalnya, `1.2.840.10008.1.2.4.100` jika instance disimpan sebagai MPEG2 Main Profile Main Level.

Lihat informasi yang lebih lengkap di Sintaks transfer yang didukung dan Pustaka decoding bingkai gambar untuk AWS HealthImaging.

Mendapatkan metadata instans DICOM dari HealthImaging

Gunakan `GetDICOMInstanceMetadata` tindakan untuk mengambil metadata dari instance DICOM di penyimpanan HealthImaging data dengan menentukan Seri, Studi, dan Instance UIDs yang terkait dengan sumber daya. API hanya akan mengembalikan metadata instance dari kumpulan gambar utama kecuali parameter kumpulan gambar opsional disediakan. Anda dapat mengambil metadata instance apa pun (dari kumpulan gambar primer atau non-primer) di penyimpanan data dengan menentukan parameter sebagai `imageSetId` kueri.

Untuk mendapatkan metadata instance DICOM () **`.json`**

1. Kumpulkan HealthImaging `datastoreId` dan nilai `imageSetId` parameter.
2. Gunakan `GetImageSetMetadata`tindakan dengan nilai `imageSetId` parameter `datastoreId` dan untuk mengambil nilai metadata terkait untuk`studyInstanceUID`,, `seriesInstanceUID` dan. `sopInstanceUID` Untuk informasi selengkapnya, lihat Mendapatkan metadata set gambar.
3. Membangun URL untuk permintaan menggunakan nilai-nilai untuk`datastoreId`,,`studyInstanceUID`, `seriesInstanceUIDsopInstanceUID`, dan`imageSetId`. Untuk melihat seluruh jalur URL dalam contoh berikut, gulir ke atas tombol Salin. URL adalah dari bentuk:

   ```
   GET https://dicom-medical-imaging.region.amazonaws.com/datastore/datastore-id/
   studies/study-instance-uid/series/series-instance-uid/instances/sop-instance-uid/
   metadata?imageSetId=image-set-id
   ```

4. Siapkan dan kirim permintaan Anda. `GetDICOMInstanceMetadata`menggunakan permintaan HTTP GET dengan protokol penandatanganan AWS Signature Version 4. Contoh kode berikut menggunakan alat baris `curl` perintah untuk mendapatkan metadata instance DICOM (`.json`file) dari. HealthImaging

   Shell

   ```
   curl --request GET \
     'https://dicom-medical-imaging.us-east-1.amazonaws.com/datastore/
   d9a2a515ab294163a2d2f4069eed584c/
   studies/1.3.6.1.4.1.5962.1.2.4.20040826285059.5457/
   series/1.3.6.1.4.1.5962.1.3.4.1.20040825185059.5457/
   instances/1.3.6.1.4.1.5962.1.1.4.1.1.20040826186059.5457/metadata?
   imageSetId=459e50687f121185f747b67bb60d1bc8' \
   ```

```
  --aws-sigv4 'aws:amz:us-east-1:medical-imaging' \
  --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
  --header "x-amz-security-token:$AWS_SESSION_TOKEN" \
  --header 'Accept: application/dicom+json'
```

> **Note**
>
> UID Sintaks Transfer yang ditunjukkan dalam metadata cocok dengan Sintaks Transfer Tersimpan UID () di. `StoredTransferSyntaxUID` HealthImaging

Mendapatkan metadata seri DICOM dari HealthImaging

Gunakan `GetDICOMSeriesMetadata` tindakan untuk mengambil metadata untuk seri DICOM (`.json`file) dari penyimpanan data. HealthImaging Anda dapat mengambil metadata seri untuk kumpulan gambar utama apa pun di penyimpanan HealthImaging data dengan menentukan Studi dan Seri yang UIDs terkait dengan sumber daya. Anda dapat mengambil metadata seri untuk kumpulan gambar non-primer dengan memberikan ID kumpulan gambar sebagai parameter kueri. Metadata seri dikembalikan dalam `DICOM JSON` format.

Untuk mendapatkan metadata seri DICOM () **`.json`**

1. Kumpulkan HealthImaging `datastoreId` dan nilai `imageSetId` parameter.
2. Buat URL untuk permintaan menggunakan nilai untuk`datastoreId`,, `studyInstanceUIDseriesInstanceUID`, dan `imageSetId` opsional. Untuk melihat seluruh jalur URL dalam contoh berikut, gulir ke atas tombol Salin. URL adalah dari bentuk:

   ```
   GET https://dicom-medical-imaging.region.amazonaws.com/datastore/datastore-id/
   studies/study-instance-uid/series/series-instance-uid/metadata
   ```
3. Siapkan dan kirim permintaan Anda. `GetDICOMSeriesMetadata`menggunakan permintaan HTTP GET dengan protokol penandatanganan AWS Signature Version 4. Contoh kode berikut menggunakan alat baris `curl` perintah untuk mendapatkan metadata (`.json`file) dari. HealthImaging

   Shell

   ```
   curl --request GET \
   ```

```
 'https://dicom-medical-imaging.us-east-1.amazonaws.com/datastore/
d9a2a515ab294163a2d2f4069eed584c/
studies/1.3.6.1.4.1.5962.1.2.4.20040826285059.5457/
series/1.3.6.1.4.1.5962.1.3.4.1.20040825185059.5457/metadata \
  --aws-sigv4 'aws:amz:us-east-1:medical-imaging' \
  --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
  --header "x-amz-security-token:$AWS_SESSION_TOKEN" \
  --header 'Accept: application/dicom+json' \
  --output 'series-metadata.json'
```

Dengan `imageSetId` parameter opsional.

Shell

```
curl --request GET \
  'https://dicom-medical-imaging.us-east-1.amazonaws.com/datastore/
d9a2a515ab294163a2d2f4069eed584c/
studies/1.3.6.1.4.1.5962.1.2.4.20040826285059.5457/
series/1.3.6.1.4.1.5962.1.3.4.1.20040825185059.5457/metadata?
imageSetId=459e50687f121185f747b67bb60d1bc8' \
  --aws-sigv4 'aws:amz:us-east-1:medical-imaging' \
  --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
  --header "x-amz-security-token:$AWS_SESSION_TOKEN" \
  --header 'Accept: application/dicom+json' \
  --output 'series-metadata.json'
```

> **Note**
>
> - `imageSetId`Parameter diperlukan untuk mengambil metadata seri untuk kumpulan gambar non-primer. `GetDICOMInstanceMetadata`Tindakan hanya akan mengembalikan metadata seri untuk kumpulan gambar utama jika`datastoreId`,`studyInstanceUID`, `seriesInstanceUID` ditentukan (tanpa). `imagesetID`

Mendapatkan bingkai instans DICOM dari HealthImaging

Gunakan `GetDICOMInstanceFrames` tindakan untuk mengambil bingkai gambar tunggal atau batch (`multipart`permintaan) dari instance DICOM di penyimpanan HealthImaging data dengan menentukan UID Seri, UID Studi, Instance UIDs, dan nomor bingkai yang terkait dengan sumber daya. Anda dapat menentukan kumpulan gambar dari mana bingkai instance harus diambil dengan memberikan ID kumpulan gambar sebagai parameter kueri. API hanya akan mengembalikan frame instance dari kumpulan gambar utama kecuali parameter set gambar opsional disediakan. Anda dapat mengambil frame instance apa pun (dari kumpulan gambar primer atau non-primer) di penyimpanan data dengan menentukan parameter `imageSetId` sebagai kueri.

Data DICOM dapat diambil baik dalam sintaks transfer yang disimpan atau sebagai format tidak terkompresi (ELE).

Untuk mendapatkan frame instance DICOM () **`multipart`**

1. Kumpulkan HealthImaging `datastoreId` dan nilai `imageSetId` parameter.
2. Gunakan `GetImageSetMetadata`tindakan dengan nilai `imageSetId` parameter `datastoreId` dan untuk mengambil nilai metadata terkait untuk`studyInstanceUID`,, `seriesInstanceUID` dan. `sopInstanceUID` Untuk informasi selengkapnya, lihat Mendapatkan metadata set gambar.
3. Tentukan bingkai gambar untuk diambil dari metadata terkait untuk membentuk parameter. `frameList` `frameList`Parameternya adalah daftar terpisah koma dari satu atau lebih nomor bingkai non-duplikat, dalam urutan apa pun. Misalnya, bingkai gambar pertama dalam metadata adalah bingkai 1.
   - Permintaan bingkai tunggal: `/frames/1`
   - Permintaan multi-bingkai: `/frames/1,2,3,4`
4. Membangun URL untuk permintaan menggunakan nilai-nilai untuk`datastoreId`,,`studyInstanceUID`,`seriesInstanceUID`, `sopInstanceUIDimageSetId`, dan`frameList`. Untuk melihat seluruh jalur URL dalam contoh berikut, gulir ke atas tombol Salin. URL adalah dari bentuk:

```
GET https://dicom-medical-imaging.region.amazonaws.com/datastore/datastore-id/
studies/study-instance-uid/series/series-instance-uid/instances/sop-instance-uid/
frames/1?imageSetId=image-set-id
```

5. Siapkan dan kirim permintaan Anda. `GetDICOMInstanceFrames`menggunakan permintaan HTTP GET dengan protokol penandatanganan AWS Signature Version 4. Contoh kode berikut menggunakan alat baris `curl` perintah untuk mendapatkan bingkai gambar dalam `multipart` respons dari HealthImaging.

   Shell

```
curl --request GET \
  'https://dicom-medical-imaging.us-east-1.amazonaws.com/datastore/
d9a2a515ab294163a2d2f4069eed584c/
studies/1.3.6.1.4.1.5962.1.2.4.20040826285059.5457/
series/1.3.6.1.4.1.5962.1.3.4.1.20040825185059.5457/
instances/1.3.6.1.4.1.5962.1.1.4.1.1.20040826186059.5457/frames/1?
imageSetId=459e50687f121185f747b67bb60d1bc8' \
  --aws-sigv4 'aws:amz:us-east-1:medical-imaging' \
  --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
  --header "x-amz-security-token:$AWS_SESSION_TOKEN" \
  --header 'Accept: multipart/related; type=application/octet-stream; transfer-
syntax=1.2.840.10008.1.2.1'
```

> **Note**
>
> `transfer-syntax`UID bersifat opsional dan default ke Explicit VR Little Endian jika tidak disertakan. Jika transcoding ke ELE tidak layak (karena impor dengan peringatan) maka piksel akan dikembalikan tanpa transcoding. Sintaks transfer yang didukung meliputi:
>
> - Eksplisit VR Little Endian (ELE) - `1.2.840.10008.1.2.1` (default untuk bingkai gambar lossless)
> - Jika `transfer-syntax=*` kemudian bingkai gambar akan dikembalikan dalam sintaks transfer yang disimpan.
> - High-Throughput JPEG 2000 dengan Kompresi Gambar Opsi RPCL (Hanya Tanpa Lossless) - `1.2.840.10008.1.2.4.202` - jika instance disimpan sebagai HealthImaging `1.2.840.10008.1.2.4.202`
> - JPEG 2000 Lossless - `1.2.840.10008.1.2.4.90` - jika instance disimpan sebagai lossless. HealthImaging

- JPEG Baseline (Proses 1): Sintaks Transfer Default untuk Kompresi Gambar 8-bit Lossy JPEG - - `1.2.840.10008.1.2.4.50` jika instance disimpan sebagai HealthImaging `1.2.840.10008.1.2.4.50`
- Kompresi Gambar JPEG 2000 `1.2.840.10008.1.2.4.91` - - jika instance disimpan HealthImaging sebagai `1.2.840.10008.1.2.4.91`
- Kompresi Gambar JPEG 2000 High-Throughput `1.2.840.10008.1.2.4.203` - - jika instance disimpan sebagai HealthImaging `1.2.840.10008.1.2.4.203`
- Kompresi Gambar JPEG XL `1.2.840.10008.1.2.4.112` - - jika instance disimpan sebagai HealthImaging `1.2.840.10008.1.2.4.112`
- Contoh yang disimpan HealthImaging dengan satu atau lebih bingkai gambar yang dikodekan dalam keluarga MPEG Transfer Syntax (yang mencakup MPEG2, MPEG-4 AVC/H.264 and HEVC/H .265) dapat diambil dengan UID sintaks transfer yang sesuai. Misalnya, `1.2.840.10008.1.2.4.100` jika instance disimpan sebagai MPEG2 Main Profile Main Level.
- Anda mungkin menerima 406 `NotAcceptableException` jika sintaks transfer yang diminta tidak dapat dikembalikan berdasarkan sintaks transfer yang disimpan, atau jika ada peringatan pemrosesan khusus untuk instance tersebut. Jika ini terjadi, coba lagi panggilan dengan`transfer-syntax=*`.

Lihat informasi yang lebih lengkap di Sintaks transfer yang didukung dan Pustaka decoding bingkai gambar untuk AWS HealthImaging.

Mendapatkan DICOM bulkdata dari HealthImaging

Gunakan `GetDICOMBulkdata` tindakan untuk mengambil data biner yang telah dipisahkan dari metadata DICOM di penyimpanan data. HealthImaging Saat mengambil metadata instance atau seri, atribut biner yang lebih besar dari 1MB akan diwakili oleh `BulkDataURI` bukan nilai inline. Anda dapat mengambil data biner untuk kumpulan gambar utama apa pun di penyimpanan HealthImaging data dengan menggunakan yang `BulkDataURI` disediakan dalam respons metadata. Anda dapat mengambil data massal untuk kumpulan gambar non-primer dengan memberikan ID kumpulan gambar sebagai parameter kueri.

Untuk mendapatkan DICOM bulkdata

Ketika Anda mengambil metadata DICOM dari tindakan HealthImaging DICOMweb WADO-RS, seperti `GetDICOMInstanceMetadata` atau, atribut biner besar akan diganti sejalan dengan`GetDICOMSeriesMetadata`, seperti yang ditunjukkan di bawah ini: BulkData URIs

```
"00451026": {
    "vr": "UN",
    "BulkDataURI": "https://dicom-medical-imaging.us-west-2.amazonaws.com/datastore/
<datastoreId>/studies/<StudyInstanceUID>/series/<SeriesInstanceUID>/instances/
<SOPInstanceUID>/bulkdata/<bulkdataUriHash>"
}
```

Untuk mengambil elemen DICOM dengan `GetDICOMBulkdata` tindakan, gunakan langkah-langkah berikut.

1. Membangun URL untuk permintaan menggunakan nilai-nilai dari`BulkDataURI`, dari bentuk:

   ```
   https://dicom-medical-imaging.region.amazonaws.com/datastore/datastore-id/
   studies/study-instance-uid/series/series-instance-uid/instances/sop-instance-uid/
   bulkdata/bulkdata-uri-hash
   ```

2. Keluarkan `GetDICOMBulkdata` perintah Anda sebagai permintaan HTTP GET dengan protokol penandatanganan AWS Signature Version 4. Contoh kode berikut menggunakan alat baris `curl` perintah untuk mengambil elemen DICOM dari kumpulan gambar utama:

   ```
   curl --request GET \
     'https://dicom-medical-imaging.us-east-1.amazonaws.com/datastore/
   d9a2a515ab294163a2d2f4069eed584c/
   studies/1.3.6.1.4.1.5962.1.2.4.20040826285059.5457/
   series/1.3.6.1.4.1.5962.1.3.4.1.20040825185059.5457/
   instances/1.2.840.10008.5.1.4.1.1.7/bulkdata/b026324c6904b2a9cb4b88d6d61c81d1' \
     --aws-sigv4 'aws:amz:us-east-1:medical-imaging' \
     --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
     --header "x-amz-security-token:$AWS_SESSION_TOKEN" \
     --header 'Accept: application/octet-stream' \
     --output 'bulkdata.bin'
   ```

   Untuk mengambil elemen data DICOM dari kumpulan gambar non-primer, berikan parameter: `ImageSetId`

   ```
   curl --request GET \
   ```

```
  'https://dicom-medical-imaging.us-east-1.amazonaws.com/datastore/
d9a2a515ab294163a2d2f4069eed584c/
studies/1.3.6.1.4.1.5962.1.2.4.20040826285059.5457/
series/1.3.6.1.4.1.5962.1.3.4.1.20040825185059.5457/
instances/1.2.840.10008.5.1.4.1.1.7/bulkdata/b026324c6904b2a9cb4b88d6d61c81d1?
imageSetId=459e50687f121185f747b67bb60d1bc8' \
  --aws-sigv4 'aws:amz:us-east-1:medical-imaging' \
  --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
  --header "x-amz-security-token:$AWS_SESSION_TOKEN" \
  --header 'Accept: application/octet-stream' \
  --output 'bulkdata.bin'
```

> **Note**
>
> `imageSetId`Parameter diperlukan untuk mengambil bulkdata untuk kumpulan gambar non-primer. DICOMBulkdata Tindakan Dapatkan hanya akan mengembalikan bulkdata untuk kumpulan gambar utama jika`datastoreId`,, `studyInstanceUID``seriesInstanceUID`, dan `SOPInstanceUID` ditentukan (tanpa). `imagesetID`

Mencari data DICOM di HealthImaging

AWS HealthImaging menawarkan representasi DICOMweb QIDO-RS APIs untuk mencari studi, seri, dan instance berdasarkan ID Pasien, dan menerima pengenal uniknya untuk penggunaan lebih lanjut. HealthImaging DICOMweb QIDO-RS APIs menawarkan fleksibilitas dalam cara Anda mencari data yang disimpan HealthImaging dan menyediakan interoperabilitas dengan aplikasi lama.

> **Penting:**
>
> HealthImaging DICOMweb APIs dapat digunakan untuk mengembalikan informasi kumpulan gambar dengan QIDO-RS. HealthImaging DICOMweb APIs referensi hanya kumpulan gambar kecuali dinyatakan lain. Gunakan tindakan bawaan HealthImaging cloud, atau parameter DICOMweb tindakan kumpulan gambar opsional untuk mengambil kumpulan gambar non-primer. HealthImaging's DICOMweb APIs dapat digunakan untuk mengembalikan informasi kumpulan gambar dengan respons DICOMweb -conformant. HealthImaging DICOMweb Tindakan QIDO-RS dapat mengembalikan maksimum 10.000 catatan. Jika ada lebih dari 10.000 sumber daya, mereka tidak akan dapat diambil melalui

tindakan QIDO-RS, tetapi dapat diambil melalui tindakan DICOMweb WADO-RS atau tindakan cloud native.

Yang APIs tercantum di bagian ini dibangun sesuai dengan standar DICOMweb (QIDO-RS) untuk pencitraan medis berbasis web. Mereka tidak ditawarkan melalui AWS CLI dan AWS SDKs.

DICOMweb APIs mencari HealthImaging

Tabel berikut menjelaskan semua HealthImaging representasi DICOMweb QIDO-RS yang APIs tersedia untuk mencari data di. HealthImaging

HealthImaging representasi QIDO-RS DICOMweb APIs

Nama	Deskripsi
`SearchDICOMStudies`	Cari studi DICOM HealthImaging dengan menentukan elemen permintaan pencarian menggunakan permintaan GET. Hasil pencarian studi dikembalikan dalam format JSON, diurutkan berdasarkan pembaruan terakhir, tanggal turun (terbaru ke terlama). Lihat Cari studi.
`SearchDICOMSeries`	Cari seri DICOM HealthImaging dengan menentukan elemen permintaan pencarian menggunakan permintaan GET. Hasil pencarian seri dikembalikan dalam format JSON, diurutkan berdasarkan `Series Number (0020, 0011)` urutan menaik (tertua hingga terbaru). Lihat Cari seri.
`SearchDICOMInstances`	Cari instance DICOM HealthImaging dengan menentukan elemen kueri penelusuran menggunakan permintaan GET. Hasil pencarian instance dikembalikan dalam format JSON, diurutkan berdasarkan `Instance`

Nama	Deskripsi
	`Number (0020, 0013)` urutan menaik (tertua hingga terbaru). Lihat Cari contoh.

Jenis DICOMweb kueri yang didukung untuk HealthImaging

HealthImaging mendukung kueri sumber daya hierarkis QIDO-RS di tingkat Studi, Seri, dan SOP Instans. Saat menggunakan pencarian hierarkis QIDO-RS untuk: HealthImaging

- Mencari studi mengembalikan daftar Studi
- Mencari Seri Studi membutuhkan yang diketahui `StudyInstanceUID` dan mengembalikan daftar Seri
- Mencari daftar Instans membutuhkan yang diketahui `StudyInstanceUID` dan `SeriesInstanceUID`

Tabel berikut menjelaskan jenis kueri hierarkis QIDO-RS yang didukung untuk mencari data di. HealthImaging

HealthImaging jenis kueri QIDO-RS yang didukung

Jenis kueri	Contoh

Kueri nilai atribut	Cari semua seri dalam Studi di manamodality=CT .
	
	`.../studies/1.3.6.1.4.1.14519.5.2.1.6279.6001.101370605276577556143013894866/series?00080060=CT`
	
	Cari semua studi di mana ID pasien dan tanggal studi masing-masing adalah nilai-nilai ini.
	
	`…/studies?PatientID=11235813&StudyDate=20130509`

Jenis kueri	Contoh
Pertanyaan kata kunci	Cari semua seri menggunakan `SeriesInstanceUID` kata kunci. `.../studies/1.3.6.1.4.1.14519.5.2.1.6279.6001.101370605276577556143013894866/series?SeriesInstanceUID=1.3.6.1.4.1.14519.5.2.1.6279.6001.101370605276577556143013894868`
Kueri tag	Cari tag menggunakan parameter kueri yang diteruskan dalam group/element formulir. {group} {element} seperti 0020000D
Kueri rentang	`...?Modality=CT&StudyDate=AABBYYYY-BBCCYYYY`
Hasil paging dengan `limit` dan `offset`	`.../studies?limit=1&offset=0&00080020=20000101` Anda dapat menggunakan parameter limit dan offset untuk melakukan paginasi respons penelusuran. Nilai default limit adalah 1000, dan lihat HealthImaging Titik akhir dan kuota AWS untuk nilai maksimum. Batas maks = 1000, Max offset = 9000

Jenis kueri	Contoh

Kueri wildcard	Kueri wildcard memberikan lebih banyak fleksibilitas pada pencarian menggunakan “*” dan “?”. “*” cocok dengan urutan karakter apa pun (termasuk nilai panjang nol) dan “?” cocok dengan karakter tunggal apa pun.
	
	Cari semua studi di datastore yang StudyDescription berisi “Nuklir”:
	
	`.../studies?StudyDescription=*Nuclear*`
	
	Cari semua studi yang StudyDescription diakhiri dengan “Nuklir”:
	
	`.../studies?StudyDescription=*Nuclear`
	
	Cari semua studi yang StudyDescription dimulai dengan “Nuklir”:
	
	`.../studies?StudyDescription=Nuclear*`
	
	Cari semua studi di mana PatientID memiliki persis 3 karakter setelah 200965981:
	
	`.../studies?PatientID=200965981???`

Jenis kueri	Contoh
FuzzyMatching pertanyaan	Aktifkan pencocokan fuzzy pada atribut DICOM nama ((0010.0010), PatientName ReferringPhysicianName (0008.0090)) dengan menambahkan parameter kueri opsional fuzzymatching: `.../studies?fuzzymatching=true&PatientName="Thomas^Albert"` Kueri ini melakukan pencocokan kata awalan case-insensitive pada bagian mana pun dari nilai. PatientName Ia mengembalikan hasil dengan PatientName nilai-nilai seperti “thomas”, “Albert”, “Thomas Albert”, “Thomas^Albert”, tetapi tidak “hom” atau “ber”.

Topik

- Mencari studi DICOM di HealthImaging
- Mencari seri DICOM di HealthImaging
- Mencari instance DICOM di HealthImaging

Mencari studi DICOM di HealthImaging

Gunakan `SearchDICOMStudies` API untuk mencari studi DICOM di penyimpanan HealthImaging data. Anda dapat mencari studi DICOM HealthImaging dengan membuat URL yang menyertakan elemen data DICOM yang didukung (atribut). Hasil pencarian studi dikembalikan dalam format JSON, diurutkan berdasarkan pembaruan terakhir, tanggal turun (terbaru ke terlama).

Untuk mencari studi DICOM

1. Kumpulkan HealthImaging `region` dan `datastoreId` nilai. Untuk informasi selengkapnya, lihat Mendapatkan properti penyimpanan data.
2. Buat URL untuk permintaan, termasuk semua elemen Studi yang berlaku. Untuk melihat seluruh jalur URL dalam contoh berikut, gulir ke atas tombol Salin. URL adalah dari bentuk:

```
GET https://dicom-medical-imaging.region.amazonaws.com/datastore/datastoreId/
studies[?query]
```

Elemen studi untuk **`SearchDICOMStudies`**

Tag elemen DICOM	Nama elemen DICOM
(0008,0020)	Study Date
(0008,0030)	StudyTime
(0008,0050)	Accession Number
(0008,0061)	Modalities in Study
(0008,0090)	Referring Physician Name
(0008,1030)	Study Description
(0010,0010)	Patient Name
(0010,0020)	Patient ID
(0010,0030)	Patient BirthDate
(0010,0032)	Patient BirthTime
(0020,000D)	Study Instance UID
(0020,0010)	Study ID

3. Siapkan dan kirim permintaan Anda. `SearchDICOMStudies`menggunakan permintaan HTTP GET dengan protokol penandatanganan AWS Signature Version 4. Contoh berikut menggunakan alat baris `curl` perintah untuk mencari informasi tentang studi DICOM.

   curl

   ```
   curl --request GET \
     "https://dicom-medical-imaging.us-east-1.amazonaws.com/datastore/datastoreId/
   studies[?query]"
     --aws-sigv4 'aws:amz:us-east-1:medical-imaging' \
   ```

```
  --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
  --header "x-amz-security-token:$AWS_SESSION_TOKEN" \
  --header 'Accept: application/dicom+json' \
  --output results.json
```

Hasil pencarian studi dikembalikan dalam format JSON, diurutkan berdasarkan pembaruan terakhir, tanggal turun (terbaru ke terlama).

Mencari seri DICOM di HealthImaging

Gunakan `SearchDICOMSeries` API untuk mencari seri DICOM di penyimpanan HealthImaging data. Anda dapat mencari seri DICOM HealthImaging dengan membuat URL yang menyertakan elemen data DICOM yang didukung (atribut). Hasil pencarian seri dikembalikan dalam format JSON, diurutkan berdasarkan ascending (tertua hingga terbaru).

Untuk mencari seri DICOM

1. Kumpulkan HealthImaging `region` dan `datastoreId` nilai. Untuk informasi selengkapnya, lihat Mendapatkan properti penyimpanan data.
2. Kumpulkan `StudyInstanceUID` nilainya. Untuk informasi selengkapnya, lihat Mendapatkan metadata set gambar.
3. Buat URL untuk permintaan, termasuk semua elemen Seri yang berlaku. Untuk melihat seluruh jalur URL dalam contoh berikut, gulir ke atas tombol Salin. URL adalah dari bentuk:

   ```
   GET https://dicom-medical-imaging.region.amazonaws.com/datastore/datastoreId/
   studies/StudyInstanceUID/series[?query]
   ```

   Elemen seri untuk **`SearchDICOMSeries`**

Tag elemen DICOM	Nama elemen DICOM
`(0008,0060)`	`Modality`
`(0020,000E)`	`Series Instance UID`

4. Siapkan dan kirim permintaan Anda. `SearchDICOMSeries`menggunakan permintaan HTTP GET dengan protokol penandatanganan AWS Signature Version 4. Contoh berikut menggunakan alat baris `curl` perintah untuk mencari informasi seri DICOM.

curl

```
curl --request GET \
  "https://dicom-medical-imaging.us-east-1.amazonaws.com/datastore/datastoreId/
studies/StudyInstanceUID/series[?query]"
  --aws-sigv4 'aws:amz:us-east-1:medical-imaging' \
  --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
  --header "x-amz-security-token:$AWS_SESSION_TOKEN" \
  --header 'Accept: application/dicom+json' \
  --output results.json
```

Hasil pencarian seri dikembalikan dalam format JSON, diurutkan berdasarkan `Series Number (0020,0011)` urutan menaik (tertua hingga terbaru).

Mencari instance DICOM di HealthImaging

Gunakan `SearchDICOMInstances` API untuk mencari instance DICOM di penyimpanan HealthImaging data. Anda dapat mencari instance DICOM HealthImaging dengan membuat URL yang menyertakan elemen data DICOM (atribut) yang didukung. Hasil Instance dikembalikan dalam format JSON, diurutkan berdasarkan ascending (tertua ke terbaru).

Untuk mencari instance DICOM

1. Kumpulkan HealthImaging `region` dan `datastoreId` nilai. Untuk informasi selengkapnya, lihat Mendapatkan properti penyimpanan data.
2. Kumpulkan nilai untuk `StudyInstanceUID` dan`SeriesInstanceUID`. Untuk informasi selengkapnya, lihat Mendapatkan metadata set gambar.
3. Buat URL untuk permintaan, termasuk semua elemen pencarian yang berlaku. Untuk melihat seluruh jalur URL dalam contoh berikut, gulir ke atas tombol Salin. URL adalah dari bentuk:

```
GET https://dicom-medical-imaging.region.amazonaws.com/datastore/datastoreId/
studies/StudyInstanceUID/series/SeriesInstanceUID/instances[?query]
```

Elemen contoh untuk **`SearchDICOMInstances`**

Tag elemen DICOM	Nama elemen DICOM
`(0008,0016)`	`SOP Class UID`
`(0008,0018)`	`SOP Instance UID`
`(0008,1196)`	`WarningReason`

HealthImaging menggunakan elemen DICOM (0008,1196) untuk mempertahankan kode peringatan impor. Kode peringatan impor dapat dicari di tingkat instans. Kode peringatan impor dapat dicari dengan wildcard atau kode peringatan tertentu. Lihat HealthImaging Kode Peringatan.

4. Siapkan dan kirim permintaan Anda. `SearchDICOMInstances`menggunakan permintaan HTTP GET dengan protokol penandatanganan AWS Signature Version 4. Contoh berikut menggunakan alat baris `curl` perintah untuk mencari informasi tentang instance DICOM.

   curl

   ```
   curl --request GET \
     "https://dicom-medical-imaging.us-east-1.amazonaws.com/datastore/datastoreId/
   studies/StudyInstanceUID/series/SeriesInstanceUID/instances[?query]"
     --aws-sigv4 'aws:amz:us-east-1:medical-imaging' \
     --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
     --header "x-amz-security-token:$AWS_SESSION_TOKEN" \
     --header 'Accept: application/dicom+json' \
     --output results.json
   ```

   Hasil pencarian instance dikembalikan dalam format JSON, diurutkan berdasarkan `Instance Number (0020,0013)` urutan menaik (tertua hingga terbaru)

Otentikasi OIDC untuk DICOMweb APIs

AWS HealthImaging mendukung otentikasi berbasis OAuth 2.0 untuk permintaan DICOMweb API menggunakan OpenID Connect (OIDC), selain otentikasi Signature Version 4 (SigV4) yang ada.AWS OIDC memungkinkan Anda untuk berintegrasi HealthImaging langsung dengan penyedia identitas

eksternal (IdPs) dan memungkinkan Anda untuk menyediakan aplikasi berbasis standar akses ke data pencitraan medis Anda melalui HealthImaging DICOMweb titik akhir tanpa mengharuskan setiap aplikasi memiliki kredensional. AWS

Topik

Verifikasi Token Kustom dengan Otorisasi Lambda

HealthImaging mengimplementasikan dukungan OIDC melalui arsitektur yang menggunakan otorisasi Lambda, yang memungkinkan pelanggan untuk menerapkan logika verifikasi token mereka sendiri. Desain ini memberi Anda kontrol yang fleksibel atas bagaimana token divalidasi dan bagaimana keputusan akses diberlakukan, mengakomodasi beragam lanskap Penyedia Identitas yang kompatibel dengan OIDC (IdPs) dan berbagai metode verifikasi token.

Alur Otentikasi

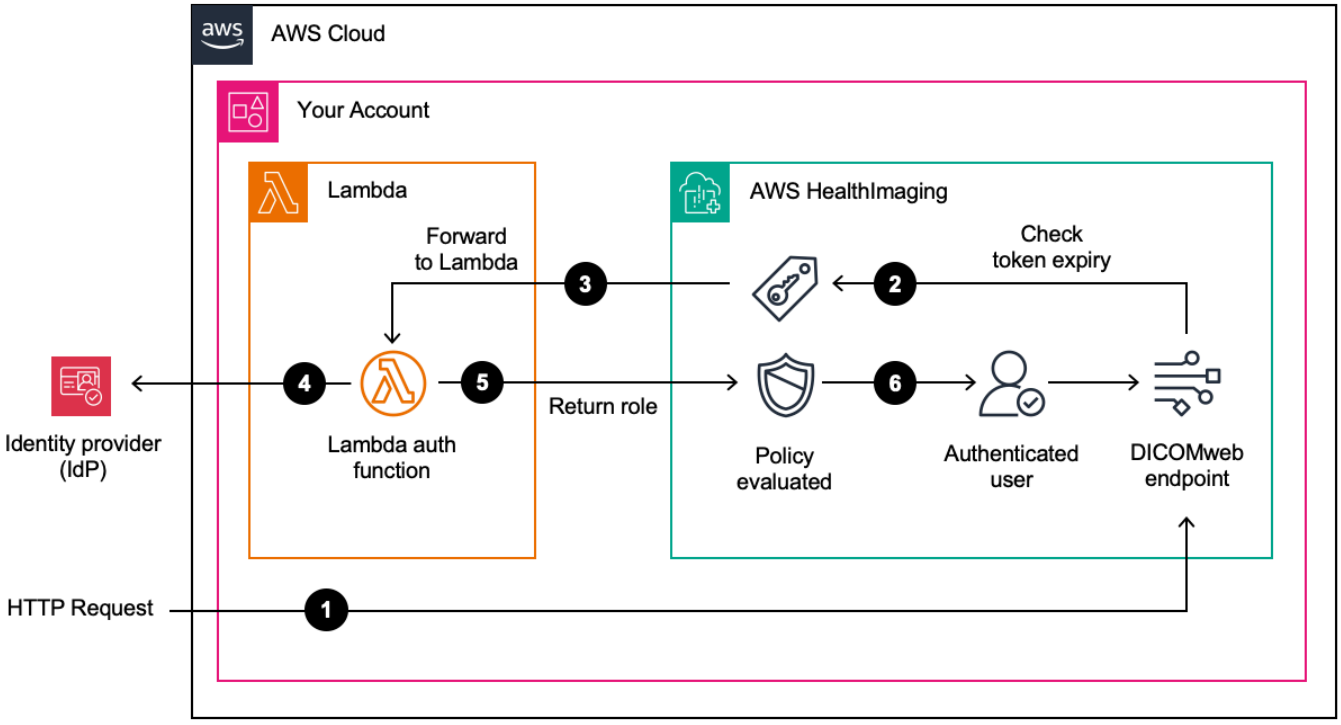
Inilah cara kerja otentikasi pada tingkat tinggi:

1. Klien memanggil DICOMweb API: Aplikasi Anda mengautentikasi dengan penyedia identitas OIDC pilihan Anda dan menerima token ID yang ditandatangani (JWT). Untuk setiap permintaan DICOMweb HTTP, klien harus menyertakan token akses OIDC di header Otorisasi (biasanya token Pembawa). Sebelum permintaan mencapai data Anda, HealthImaging ekstrak token ini dari permintaan yang masuk dan memanggil otorisasi Lambda yang Anda konfigurasikan.
   a. Header biasanya mengikuti format:`Authorization: Bearer <token>`.
2. Verifikasi awal: HealthImaging memverifikasi klaim token akses untuk dengan cepat menolak token yang jelas tidak valid atau kedaluwarsa tanpa menjalankan fungsi Lambda secara tidak perlu. HealthImaging melakukan verifikasi awal klaim standar tertentu dalam token akses sebelum memanggil otorisasi Lambda:
   a. `iat`(Diterbitkan Pada): HealthImaging memeriksa apakah waktu penerbitan token berada dalam batas yang dapat diterima.
   b. `exp`(Waktu Kedaluwarsa): HealthImaging memverifikasi bahwa token belum kedaluwarsa.
   c. `nbf`(Tidak Sebelum Waktu): Jika ada HealthImaging , pastikan token tidak digunakan sebelum waktu mulai yang valid.

3. HealthImaging memanggil otorisasi Lambda: Jika verifikasi klaim awal lolos HealthImaging , maka delegasikan verifikasi token lebih lanjut ke fungsi otorisasi Lambda yang dikonfigurasi pelanggan. HealthImaging meneruskan token yang diekstraksi dan informasi permintaan relevan lainnya ke fungsi Lambda. Fungsi Lambda memverifikasi tanda tangan dan klaim token.
4. Verifikasi dengan penyedia identitas: Lambda berisi kode kustom yang memeriksa tanda tangan token ID, melakukan verifikasi token yang lebih luas (misalnya, penerbit, audiens, klaim khusus), dan memvalidasi klaim tersebut terhadap IDP bila diperlukan.
5. Authorizer mengembalikan kebijakan akses: Setelah verifikasi berhasil, fungsi Lambda menentukan izin yang sesuai untuk penggunaan yang diautentikasi. Authorizer Lambda kemudian mengembalikan nama sumber daya amazon (ARN) dari peran IAM yang mewakili kumpulan izin yang akan diberikan.
6. Permintaan eksekusi: Jika peran IAM yang diasumsikan memiliki izin yang diperlukan, HealthImaging lanjutkan dengan mengembalikan sumber daya yang diminta. DICOMWeb Jika izin tidak mencukupi, HealthImaging menolak permintaan dan mengembalikan kesalahan respons kesalahan yang sesuai (yaitu, 403 Terlarang).

> **Note**
>
> Lambda otorisasi berfungsi tidak dikelola oleh layanan HealthImaging AWS. Ini mengeksekusi di AWS akun Anda. Pelanggan dikenakan biaya untuk pemanggilan fungsi dan waktu eksekusi secara terpisah dari biaya mereka HealthImaging .

Ikhtisar Arsitektur



Alur kerja otentikasi OIDC dengan otorisasi Lambda

Prasyarat

Persyaratan Token Akses

HealthImaging mengharuskan token akses dalam format JSON Web Token (JWT). Banyak Penyedia Identitas (IDPs) menawarkan format token ini secara asli, sementara yang lain memungkinkan Anda untuk memilih atau mengonfigurasi formulir token akses. Pastikan IDP pilihan Anda dapat mengeluarkan token JWT sebelum melanjutkan integrasi.

Format Token

Token akses harus dalam format JWT (JSON Web Token)

Klaim yang Diperlukan

`exp`(Waktu Kedaluwarsa)

Klaim wajib yang menentukan kapan token menjadi tidak valid.

- Harus setelah waktu saat ini di UTC
- Merupakan saat token menjadi tidak valid

`iat`(Dikeluarkan Pada)

Klaim wajib yang menentukan kapan token dikeluarkan.

- Harus sebelum waktu saat ini di UTC
- TIDAK boleh lebih awal dari 12 jam sebelum waktu saat ini di UTC
- Ini secara efektif memberlakukan masa pakai token maksimum 12 jam

`nbf`(Tidak sebelum waktu)

Klaim opsional yang menentukan waktu paling awal token dapat digunakan.

- Jika ada, akan dievaluasi oleh HealthImaging
- Menentukan waktu sebelum token tidak boleh diterima

Persyaratan waktu respons otorisasi Lambda

HealthImaging memberlakukan persyaratan waktu yang ketat untuk respons otorisasi Lambda untuk memastikan kinerja API yang optimal. Fungsi Lambda Anda harus kembali dalam 1 detik.

Praktik terbaik

Optimalkan Verifikasi Token

- Cache JWKS (Set Kunci Web JSON) bila memungkinkan
- Cache token akses yang valid bila memungkinkan
- Minimalkan panggilan jaringan ke Penyedia Identitas Anda
- Menerapkan logika validasi token yang efisien

Konfigurasi Lambda

- Fungsi berbasis Python dan Node.js biasanya diinisialisasi lebih cepat
- Kurangi jumlah pustaka eksternal yang akan dimuat
- Konfigurasikan alokasi memori yang sesuai untuk memastikan kinerja yang konsisten
- Pantau waktu eksekusi menggunakan CloudWatch metrik

Pengaktifan Otentikasi OIDC

- Otentikasi OIDC hanya dapat diaktifkan saat membuat datastore baru
- Mengaktifkan OIDC untuk datastores yang ada tidak didukung melalui API
- Untuk mengaktifkan OIDC pada datastore yang ada, pelanggan harus menghubungi Support AWS

Siapkan otorisasi AWS Lambda untuk otentikasi OIDC

Panduan ini mengasumsikan Anda telah mengonfigurasi Identity Provider (IDP) pilihan Anda untuk menyediakan token akses yang kompatibel dengan persyaratan fitur otentikasi HealthImaging OIDC.

1. Konfigurasikan Peran IAM untuk Akses DICOMWeb API

Sebelum mengonfigurasi otorisasi Lambda, buat peran IAM HealthImaging untuk diasumsikan saat memproses permintaan API. DICOMWeb Fungsi Lambda otorisasi mengembalikan salah satu peran ini ARN setelah verifikasi token berhasil, HealthImaging memungkinkan untuk mengeksekusi permintaan dengan izin yang sesuai.

1. Buat kebijakan IAM yang menentukan hak istimewa DICOMWeb API yang diinginkan. Lihat bagian "Menggunakan DICOMweb" dari HealthImaging dokumentasi untuk izin yang tersedia.
2. Buat peran IAM yang:
   - Lampirkan kebijakan ini
   - Sertakan hubungan kepercayaan yang memungkinkan AWS HealthImaging service principal (`medical-imaging.amazonaws.com`) untuk mengambil peran ini.

Berikut adalah contoh kebijakan yang mengizinkan peran terkait mengakses API HealthImaging DICOMWeb hanya-baca:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "MedicalImagingDicomWebOperations",
            "Effect": "Allow",
            "Action": [
```

```
                "medical-imaging:SearchDICOMInstances",
                "medical-imaging:GetImageSetMetadata",
                "medical-imaging:GetDICOMSeriesMetadata",
                "medical-imaging:SearchDICOMStudies",
                "medical-imaging:GetDICOMBulkdata",
                "medical-imaging:SearchDICOMSeries",
                "medical-imaging:GetDICOMInstanceMetadata",
                "medical-imaging:GetDICOMInstance",
                "medical-imaging:GetDICOMInstanceFrames"
            ],
            "Resource": "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/datastore-123"
        }
    ]
}
```

Berikut adalah contoh kebijakan hubungan kepercayaan yang harus dikaitkan dengan peran:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "OIDCRoleFederation",
            "Effect": "Allow",
            "Principal": {
                "Service": "medical-imaging.amazonaws.com"
        },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Authorizer Lambda yang akan Anda buat pada langkah berikutnya dapat mengevaluasi klaim token dan mengembalikan ARN dari peran yang sesuai. AWS kemudian HealthImaging akan meniru peran ini untuk menjalankan permintaan DICOMWeb API dengan izin yang sesuai.

Contoh:

- Token dengan klaim “admin” mungkin mengembalikan ARN untuk peran dengan akses penuh
- Token dengan klaim “pembaca” mungkin mengembalikan ARN untuk peran dengan akses hanya-baca
- Token dengan klaim “Department_a” mungkin mengembalikan ARN untuk peran khusus untuk tingkat akses departemen tersebut

Mekanisme ini memungkinkan Anda memetakan model otorisasi IDP Anda ke HealthImaging izin AWS tertentu melalui peran IAM.

2. Buat dan Konfigurasikan Fungsi Lambda Authorizer

Buat fungsi Lambda yang akan memverifikasi token JWT dan mengembalikan ARN peran IAM yang sesuai berdasarkan evaluasi klaim token. Fungsi ini dipanggil oleh layanan pencitraan kesehatan dan meneruskan peristiwa yang berisi Id HealthImaging datastore, DICOMWeb operasi, dan token akses yang ditemukan dalam permintaan HTTP:

```
{
  "datastoreId": "{datastore id}",
  "operation": "{Healthimaging API name e.g. GetDICOMInstance}",
  "bearerToken": "{access token}"
}
```

Fungsi otorisasi Lambda harus mengembalikan respons JSON dengan struktur berikut:

```
{
  "isTokenValid": {true or false},
  "roleArn": "{role arn or empty string meaning to deny the request explicitly}"
}
```

Anda dapat merujuk ke contoh implementasi untuk informasi lebih lanjut.

> **Note**
>
> Karena DICOMWeb permintaan hanya dijawab setelah token akses diverifikasi oleh otorisasi lambda, penting bahwa eksekusi fungsi ini secepat mungkin untuk menyediakan waktu respons DICOMWeb API terbaik.

Agar HealthImaging layanan diotorisasi untuk menjalankan fungsi otorisasi lambda, layanan harus memiliki kebijakan sumber daya yang memungkinkan HealthImaging layanan untuk memanggilnya. Kebijakan sumber daya ini dapat dibuat di menu izin tab konfigurasi lambda atau Menggunakan AWS CLI:

```
aws lambda add-permission \
    --function-name YourAuthorizerFunctionName \
    --statement-id HealthImagingInvoke \
    --action lambda:InvokeFunction \
    --principal medical-imaging.amazonaws.com
```

Kebijakan sumber daya ini memungkinkan HealthImaging layanan untuk memanggil otorisasi Lambda Anda saat DICOMWeb mengautentikasi permintaan API.

> **Note**
>
> Kebijakan sumber daya lambda dapat diperbarui nanti dengan kondisi "ArnLike" yang cocok dengan ARN dari datastore tertentu HealthImaging .

Berikut adalah contoh kebijakan sumber daya lambda:

JSON

```
{
  "Version":"2012-10-17",
  "Id": "default",
  "Statement": [
    {
      "Sid": "LambaAuthorizer-HealthImagingInvokePermission",
      "Effect": "Allow",
      "Principal": {
        "Service": "medical-imaging.amazonaws.com"
      },
      "Action": "lambda:InvokeFunction",
      "Resource": "arn:aws:lambda:us-east-1:123456789012::function:
{LambdaAuthorizerFunctionName}",
      "Condition": {
        "ArnLike": {
          "AWS:SourceArn": "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/datastore-123"
```

```
        }
      }
    }
  ]
}
```

3. Buat Datastore Baru dengan Otentikasi OIDC

Untuk mengaktifkan otentikasi OIDC, Anda harus membuat datastore baru menggunakan AWS CLI dengan parameter "". lambda-authorizer-arn Otentikasi OIDC tidak dapat diaktifkan pada datastores yang ada tanpa menghubungi Support. AWS

Berikut adalah contoh cara membuat datastore baru dengan otentikasi OIDC diaktifkan:

```
aws medical-imaging create-datastore \
    --datastore-name YourDatastoreName \
    --lambda-authorizer-arn YourAuthorizerFunctionArn
```

Anda dapat memeriksa apakah datastore tertentu memiliki fitur otentikasi OIDC diaktifkan dengan menggunakan perintah AWS CLI get-datastore, dan memverifikasi apakah atribut "" ada: lambdaAuthorizerArn

```
aws medical-imaging get-datastore --datastore-id YourDatastoreId
```

```
{
    "datastoreProperties": {
        "datastoreId": YourdatastoreId,
        "datastoreName": YourDatastoreName,
        "datastoreStatus": "ACTIVE",
        "lambdaAuthorizerArn": YourAuthorizerFunctionArn,
        "datastoreArn": YourDatastoreArn,
        "createdAt": "2025-09-30T14:16:04.015000-05:00",
        "updatedAt": "2025-09-30T14:16:04.015000-05:00"
    }
}
```

> **Note**
>
> Peran eksekusi untuk perintah pembuatan AWS CLI datastore harus memiliki izin yang sesuai untuk menjalankan fungsi otorisasi Lambda. Ini mengurangi serangan eskalasi hak

istimewa di mana pengguna jahat dapat menjalankan fungsi Lambda yang tidak sah melalui konfigurasi otorisasi datastore.

Kode Pengecualian

Dalam kasus kegagalan otentikasi HealthImaging mengembalikan kode respon kesalahan HTTP berikut dan pesan tubuh:

Kondisi	Tanggapan AHI
Lambda Authorizer tidak ada atau tidak valid	424 Kesalahan Konfigurasi Otorisasi
Authorizer dihentikan karena kegagalan eksekusi	424 Authorizer Gagal
Kesalahan otorisasi lain yang tidak dipetakan	424 Authorizer Gagal
Authorizer mengembalikan respons yang tidak valid/tidak terbentuk	424 Kesalahan Konfigurasi Otorisasi
Authorizer berjalan lebih dari 1 d	408 Batas Waktu Otorisasi
Token kedaluwarsa atau tidak valid	403 Token Tidak Valid atau Kedaluwarsa
AHI tidak dapat menggabungkan Peran IAM yang dikembalikan karena kesalahan konfigurasi otorisasi	424 Kesalahan Konfigurasi Otorisasi
Authorizer mengembalikan Peran kosong	403 Akses Ditolak

Kondisi	Tanggapan AHI
Peran yang Dikembalikan tidak dapat dipanggil (assume-role/trust misconfig)	424 Kesalahan Konfigurasi Otorisasi
Tingkat permintaan melebihi batas DICOMweb Gateway	429 Terlalu banyak permintaan
Datastore, Peran Pengembalian, atau Authorizer Lintas Wilayah Account/Cross	424 Akses Account/Cross Lintas Wilayah Authorizer

Contoh Implementasi

Contoh Python ini menunjukkan fungsi otorisasi lambda yang memverifikasi token akses AWS Cognito dari peristiwa HealthImaging dan mengembalikan peran IAM ARN dengan hak istimewa yang sesuai. DICOMWeb

Authorizer Lambda mengimplementasikan dua mekanisme caching untuk mengurangi panggilan eksternal dan latensi respons. JWKS (JSON Web Key Set) diambil sekali setiap jam dan disimpan dalam folder sementara fungsi, memungkinkan pemanggilan fungsi berikutnya untuk membacanya secara lokal alih-alih mengambil dari jaringan publik. Anda juga akan melihat bahwa objek kamus token_cache dipakai dalam konteks global fungsi Lambda ini. Variabel global dibagi oleh semua pemanggilan yang menggunakan kembali konteks Lambda hangat yang sama. Berkat ini, token yang berhasil diverifikasi dapat disimpan dalam kamus ini dan dicari dengan cepat selama eksekusi berikutnya dari fungsi Lambda yang sama ini. Metode caching merupakan pendekatan generalis yang dapat menyesuaikan token akses yang dikeluarkan dari sebagian besar penyedia identitas. Untuk opsi caching khusus AWS Cognito, lihat bagian Mengelola kumpulan Pengguna dan bagian caching dari dokumentasi Cognito.AWS

```
import json
import os
import time
import logging
from jose import jwk, jwt
from jose.exceptions import ExpiredSignatureError, JWTClaimsError, JWTError
import requests
import tempfile
```

```
# Configure logging
logger = logging.getLogger()
log_level = os.environ.get('LOG_LEVEL', 'WARNING').upper()
logger.setLevel(getattr(logging, log_level, logging.WARNING))

# Global token cache with TTL
token_cache = {}

# JWKS cache file path
JWKS_CACHE_FILE = os.path.join(tempfile.gettempdir(), 'jwks.json')
JWKS_CACHE_TTL = 3600  # 1 hour

# Load environment variables once
USER_POOL_ID = os.environ['USER_POOL_ID']
CLIENT_ID = os.environ['CLIENT_ID']
ROLE_ARN = os.environ.get('AHIDICOMWEB_READONLY_ROLE_ARN', '')

def cleanup_expired_tokens():
    """Remove expired tokens from cache"""
    now = int(time.time())
    expired_keys = [token for token, data in token_cache.items() if now >
 data['cache_expiry']]
    for token in expired_keys:
        del token_cache[token]

def get_cached_jwks():
    """Get JWKS from cache file if valid, otherwise return None """
    try:
        if os.path.exists(JWKS_CACHE_FILE):
            # Check if cache file is still valid
            cache_age = time.time() - os.path.getmtime(JWKS_CACHE_FILE)
            if cache_age < JWKS_CACHE_TTL:
                with open(JWKS_CACHE_FILE, 'r') as f:
                    jwks = json.load(f)
                    logger.debug(f'Using cached JWKS (age: {int(cache_age)}s)')
                    return jwks
            else:
                logger.debug(f'JWKS cache expired (age: {int(cache_age)}s)')
    except Exception as e:
        logger.debug(f'Error reading JWKS cache: {e}')

    return None
```

```
def cache_jwks(jwks):
    """Cache JWKS to file"""
    try:
        with open(JWKS_CACHE_FILE, 'w') as f:
            json.dump(jwks, f)
        logger.debug('JWKS cached successfully')
    except Exception as e:
        logger.debug(f'Error caching JWKS: {e}')

def fetch_jwks(jwks_url):
    """Fetch JWKS from URL and cache it"""
    logger.debug('Fetching JWKS from URL')
    jwks = requests.get(jwks_url, timeout=10).json()
    # Convert to dict for faster lookups
    jwks['keys_by_kid'] = {key['kid']: key for key in jwks['keys']}
    cache_jwks(jwks)
    return jwks

def is_token_cached(token):
    if token not in token_cache:
        return None

    cached = token_cache[token]
    now = int(time.time())

    if now > cached['cache_expiry']:
        del token_cache[token]
        return None

    return cached

def cache_token(token, payload):
    now = int(time.time())
    token_exp = payload.get('exp')
    cache_expiry = min(now + 60, token_exp)  # 1 minute or token expiry, whichever is
 sooner

    token_cache[token] = {
        'payload': payload,
        'cache_expiry': cache_expiry,
        'role_arn': ROLE_ARN
    }

def handler(event, context):
```

```
    cleanup_expired_tokens() # start be removing expired tokens from the cache
    try:
        # Extract token from bearerToken or authorizationToken field
        token = event.get('bearerToken')
        if not token:
            raise Exception('No token provided')

        # Check cache first
        cached = is_token_cached(token)
        if cached:
            logger.debug('Token found in cache, skipping verification')
            return {
                'isTokenValid': True,
                'roleArn': cached['role_arn']
            }

        # Get Cognito configuration
        region = context.invoked_function_arn.split(':')[3]

        # Get JWKS (cached or fresh)
        jwks_url = f'https://cognito-idp.{region}.amazonaws.com/{USER_POOL_ID}/.well-
known/jwks.json'
        jwks = get_cached_jwks()
        if not jwks:
            jwks = fetch_jwks(jwks_url)

        # Decode token header to get kid
        headers = jwt.get_unverified_headers(token)
        kid = headers['kid']

        # Find the correct key
        key = None
        for jwk_key in jwks['keys']:
            if jwk_key['kid'] == kid:
                key = jwk_key
                break

        if not key:
            # Key not found - try refreshing JWKS in case of key rotation
            logger.debug('Key not found in cached JWKS, fetching fresh JWKS')
            jwks = fetch_jwks(jwks_url)
            for jwk_key in jwks['keys']:
                if jwk_key['kid'] == kid:
                    key = jwk_key
```

```
                    break

        if not key:
            raise Exception('Public key not found')

        # Construct the public key
        public_key = jwk.construct(key)

        # Verify and decode the token (includes expiry validation)
        payload = jwt.decode(
            token,
            public_key,
            algorithms=['RS256'],
            audience=CLIENT_ID,
            issuer=f'https://cognito-idp.{region}.amazonaws.com/{USER_POOL_ID}'
        )

        logger.debug('Token validated successfully')
        logger.debug('User: %s', payload.get('username', 'unknown'))

        # Cache the validated token
        cache_token(token, payload)

        # Return authorization response
        return {
            'isTokenValid': True,
            'roleArn': ROLE_ARN
        }

    except ExpiredSignatureError:
        logger.debug('Token expired')
        return {
            'isTokenValid': False,
            'roleArn': ''
        }
    except JWTClaimsError:
        logger.debug('Invalid token claims')
        return {
            'isTokenValid': False,
            'roleArn': ''
        }
    except JWTError as e:
        logger.debug('JWT validation error: %s', e)
        return {
```

```
            'isTokenValid': False,
            'roleArn': ''
        }
    except Exception as e:
        logger.debug('Authorization failed: %s', e)
        return {
            'isTokenValid': False,
            'roleArn': ''
        }
```

Mengimpor data pencitraan dengan AWS HealthImaging

Mengimpor adalah proses memindahkan data pencitraan medis Anda dari bucket input Amazon S3 ke penyimpanan data HealthImaging AWS. Selama impor, AWS HealthImaging melakukan pemeriksaan verifikasi data piksel sebelum mengubah file DICOM P10 Anda menjadi kumpulan gambar yang terdiri dari metadata dan bingkai gambar (data piksel).

> **Penting:**
>
> HealthImaging mengimpor pekerjaan memproses binari instance DICOM (`.dcm`file) dan mengubahnya menjadi kumpulan gambar. Gunakan tindakan bawaan HealthImaging cloud (APIs) untuk mengelola penyimpanan data dan kumpulan gambar. Gunakan HealthImaging representasi DICOMweb layanan untuk mengembalikan DICOMweb respons.

Topik berikut menjelaskan cara mengimpor data pencitraan medis Anda ke penyimpanan HealthImaging data menggunakan Konsol Manajemen AWS, AWS CLI, dan AWS SDKs.

Topik

Memahami pekerjaan impor

Setelah membuat penyimpanan data di AWS HealthImaging, Anda harus mengimpor data pencitraan medis dari bucket input Amazon S3 ke penyimpanan data untuk membuat kumpulan gambar. Anda dapat menggunakan Konsol Manajemen AWS, AWS CLI, dan AWS SDKs untuk memulai, mendeskripsikan, dan daftar pekerjaan impor.

Saat Anda mengimpor data DICOM P10 ke penyimpanan HealthImaging data AWS, layanan akan mencoba mengatur instans secara otomatis sesuai dengan hierarki DICOM Study UID, Series UID, Instance UID, berdasarkan elemen metadata. Data yang diimpor akan dibuat primer jika elemen metadata dari data yang diimpor tidak bertentangan dengan kumpulan gambar primer yang ada di

penyimpanan data. Jika elemen metadata data DICOM P10 yang baru diimpor bertentangan dengan kumpulan gambar primer yang ada, data baru akan ditambahkan ke kumpulan gambar non-primer. Saat data mengimpor membuat kumpulan gambar non-primer, AWS HealthImaging memancarkan EventBridge Peristiwa dengan`isPrimary: False`, dan catatan yang ditulis ke dalam objek juga `success.ndjson` akan ada di `isPrimary: False` dalam objek. `importResponse`

Saat Anda mengimpor data, HealthImaging lakukan hal berikut:

- Jika instance yang terdiri dari seri DICOM diimpor dalam satu pekerjaan impor dan instance tidak bertentangan dengan instance yang sudah ada di penyimpanan data, maka semua instance diatur ke dalam satu set gambar utama.
- Jika instance yang terdiri dari seri DICOM diimpor dalam dua atau lebih pekerjaan impor dan instance tidak bertentangan dengan instance yang sudah ada di penyimpanan data, maka semua instance diatur sebagai satu set gambar Primer.
- Jika instance diimpor lebih dari sekali, versi terbaru akan menimpa versi lama yang disimpan dalam kumpulan gambar utama, dan nomor versi kumpulan gambar utama akan bertambah.

Anda dapat memperbarui instance di primer dengan langkah-langkah yang dijelaskan dalam Memperbarui metadata set Gambar.

Selama impor, nilai biner dalam tag pribadi (dengan tipe VR OB, OD, OF, OL, OV, OW, UN) yang melebihi 1MB dalam ukuran disimpan secara terpisah dari metadata. Saat mengambil metadata untuk instance ini menggunakan `GetDICOMInstanceMetadata` or`GetDICOMSeriesMetadata`, nilai biner besar ini diganti dengan BulkDataURIs, dan data biner aktual dapat diambil menggunakan API. `GetDICOMBulkdata`

Ingatlah hal-hal berikut saat mengimpor file pencitraan medis Anda dari Amazon S3 ke HealthImaging penyimpanan data:

- Instance yang sesuai dengan Seri DICOM akan secara otomatis digabungkan dalam satu set gambar, dilambangkan primer.
- Anda dapat mengimpor data DICOM P10 dalam satu pekerjaan impor, atau beberapa pekerjaan impor, dan layanan akan mengatur instance ke dalam kumpulan gambar utama yang sesuai dengan Seri DICOM
- Kendala panjang berlaku untuk elemen DICOM tertentu selama impor. Untuk memastikan pekerjaan impor berhasil, verifikasi bahwa data pencitraan medis Anda tidak melebihi batasan panjang. Untuk informasi selengkapnya, lihat Kendala elemen DICOM.

- Pemeriksaan verifikasi data piksel dilakukan di awal pekerjaan impor. Untuk informasi selengkapnya, lihat Verifikasi data Pixel.
- Ada titik akhir, kuota, dan batas pelambatan yang terkait dengan tindakan impor. HealthImaging Untuk informasi selengkapnya, lihat Titik akhir dan kuota dan Batas pelambatan.
- Untuk setiap pekerjaan impor, hasil pemrosesan disimpan di `outputS3Uri` lokasi. Hasil pemrosesan diatur sebagai `job-output-manifest.json` file SUCCESS dan FAILURE folder.

> **Note**
>
> Anda dapat menyertakan hingga 10.000 folder bersarang untuk satu pekerjaan impor.

- `job-output-manifest.json`File berisi `jobSummary` output dan rincian tambahan tentang data yang diproses. Contoh berikut menunjukkan output dari `job-output-manifest.json` file.

```
{
"jobSummary": {
"jobId": "09876543210987654321098765432109",
        "datastoreId": "12345678901234567890123456789012",
        "inputS3Uri": "s3://medical-imaging-dicom-input/dicom_input/",
        "outputS3Uri": "s3://medical-imaging-output/
job_output/12345678901234567890123456789012-
DicomImport-09876543210987654321098765432109/",
        "successOutputS3Uri": "s3://medical-imaging-
output/job_output/12345678901234567890123456789012-
DicomImport-09876543210987654321098765432109/SUCCESS/",
        "failureOutputS3Uri": "s3://medical-imaging-
output/job_output/12345678901234567890123456789012-
DicomImport-09876543210987654321098765432109/FAILURE/",
        "warningsOutputS3Uri": "s3://medical-imaging-
output/job_output/12345678901234567890123456789012-
DicomImport-09876543210987654321098765432109/WARNING/",
        "numberOfScannedFiles": 5,
        "numberOfImportedFiles": 3,
        "numberOfFilesWithCustomerError": 2,
        "numberOfFilesWithServerError": 0,
        "numberOfGeneratedImageSets": 2,
        "imageSetsSummary": [{
"imageSetId": "12345612345612345678907890789012",
                "numberOfMatchedSOPInstances": 2
```

```
            },
            {
"imageSetId": "12345612345612345678917891789012",
                "numberOfMatchedSOPInstances": 1
            }
        ]
    }
}
```

- SUCCESSFolder menyimpan `success.ndjson` file yang berisi hasil dari semua file pencitraan yang berhasil diimpor. Contoh berikut menunjukkan output dari `success.ndjson` file.

```
{"inputFile":"dicomInputFolder/1.3.51.5145.5142.20010109.1105620.1.0.1.dcm","importResponse
{"imageSetId":"12345612345612345678907890789012", "isPrimary": True}}
{"inputFile":"dicomInputFolder/1.3.51.5145.5142.20010109.1105630.1.0.1.dcm","importResponse
{"imageSetId":"12345612345612345678917891789012", "isPrimary": True}}
```

- FAILUREFolder menyimpan `failure.ndjson` file yang berisi hasil dari semua file pencitraan yang tidak berhasil diimpor. Contoh berikut menunjukkan output dari `failure.ndjson` file.

```
{"inputFile":"dicom_input/invalidDicomFile1.dcm","exception":
{"exceptionType":"ValidationException","message":"DICOM attribute TransferSyntaxUID
 does not exist"}}
{"inputFile":"dicom_input/invalidDicomFile2.dcm","exception":
{"exceptionType":"ValidationException","message":"DICOM attributes does not
 exist"}}
```

- WARNINGFolder menyimpan `warning.ndjson` file yang berisi hasil dari semua file pencitraan yang berhasil diimpor tetapi dengan peringatan. Contoh berikut menunjukkan output dari `warning.ndjson` file.

```
{"inputFile":"dicom_input/warningDicomFile1.dcm","importResponse":
{"imageSetId":"12345612345612345678907890789012","imageSetVersion":1,"isPrimary":true,"warn
[{"warning_reason_code":45330,"type":"InvalidOffsetTable","message":"The file was
 imported but contains an invalid offset table, may see issues when retrieving
 certain frames."}]}}
```

- Pekerjaan impor disimpan dalam daftar pekerjaan selama 90 hari dan kemudian diarsipkan.

Memulai pekerjaan impor

Gunakan `StartDICOMImportJob` tindakan untuk memulai pemeriksaan verifikasi data piksel dan impor data massal ke penyimpanan HealthImaging data AWS. Pekerjaan impor mengimpor file DICOM P10 yang terletak di bucket masukan Amazon S3 yang ditentukan oleh parameter. `inputS3Uri` Hasil pemrosesan pekerjaan impor disimpan di bucket keluaran Amazon S3 yang ditentukan oleh parameter. `outputS3Uri`

> **Note**
>
> Ingatlah hal-hal berikut sebelum memulai pekerjaan impor:
>
> - HealthImaging mendukung mengimpor file DICOM P10 dengan sintaks transfer yang berbeda. Beberapa file mempertahankan pengkodean sintaks transfer aslinya selama impor, sementara yang lain ditranskode ke HTJ2 K lossless secara default atau JPEG 2000 Lossless tergantung pada konfigurasi datastore Anda. Untuk informasi selengkapnya, lihat Sintaks transfer yang didukung.
> - HealthImaging mendukung impor data dari bucket Amazon S3 yang terletak di Wilayah lain yang didukung. Untuk mencapai fungsi ini, berikan `inputOwnerAccountId` parameter saat memulai pekerjaan impor. Untuk informasi selengkapnya, lihat Impor lintas akun untuk AWS HealthImaging.
> - HealthImaging menerapkan batasan panjang untuk elemen DICOM tertentu selama impor. Untuk informasi selengkapnya, lihat Kendala elemen DICOM.

Menu berikut memberikan prosedur untuk contoh Konsol Manajemen AWS dan kode untuk AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat `StartDICOMImportJob`di AWS HealthImaging API Referensi.

Untuk memulai pekerjaan impor

Pilih menu berdasarkan preferensi akses Anda ke AWS HealthImaging.

AWS Konsol

1. Buka halaman penyimpanan data HealthImaging konsol.
2. Pilih penyimpanan data.
3. Pilih Impor data DICOM.

Halaman data Impor DICOM terbuka.

4. Di bawah bagian Detail, masukkan informasi berikut:
   - Nama (opsional)
   - Impor lokasi sumber di S3
   - ID akun pemilik bucket sumber (opsional)
   - Kunci enkripsi (opsional)
   - Tujuan keluaran di S3
5. Di bawah bagian Akses layanan, pilih Gunakan peran layanan yang ada dan pilih peran dari menu Nama peran layanan atau pilih Buat dan gunakan peran layanan baru.
6. Pilih Impor.

AWS CLI dan SDKs

C++

SDK untuk C++

```
//! Routine which starts a HealthImaging import job.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param inputBucketName: The name of the Amazon S3 bucket containing the DICOM
 files.
  \param inputDirectory: The directory in the S3 bucket containing the DICOM
 files.
  \param outputBucketName: The name of the S3 bucket for the output.
  \param outputDirectory: The directory in the S3 bucket to store the output.
  \param roleArn: The ARN of the IAM role with permissions for the import.
  \param importJobId: A string to receive the import job ID.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
  */
bool AwsDoc::Medical_Imaging::startDICOMImportJob(
```

```
        const Aws::String &dataStoreID, const Aws::String &inputBucketName,
        const Aws::String &inputDirectory, const Aws::String &outputBucketName,
        const Aws::String &outputDirectory, const Aws::String &roleArn,
        Aws::String &importJobId,
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient medicalImagingClient(clientConfig);
    Aws::String inputURI = "s3://" + inputBucketName + "/" + inputDirectory +
"/";
    Aws::String outputURI = "s3://" + outputBucketName + "/" + outputDirectory +
"/";
    Aws::MedicalImaging::Model::StartDICOMImportJobRequest
startDICOMImportJobRequest;
    startDICOMImportJobRequest.SetDatastoreId(dataStoreID);
    startDICOMImportJobRequest.SetDataAccessRoleArn(roleArn);
    startDICOMImportJobRequest.SetInputS3Uri(inputURI);
    startDICOMImportJobRequest.SetOutputS3Uri(outputURI);

    Aws::MedicalImaging::Model::StartDICOMImportJobOutcome
startDICOMImportJobOutcome = medicalImagingClient.StartDICOMImportJob(
            startDICOMImportJobRequest);

    if (startDICOMImportJobOutcome.IsSuccess()) {
        importJobId = startDICOMImportJobOutcome.GetResult().GetJobId();
    }
    else {
        std::cerr << "Failed to start DICOM import job because "
                  << startDICOMImportJobOutcome.GetError().GetMessage() <<
std::endl;
    }

    return startDICOMImportJobOutcome.IsSuccess();
}
```

- Untuk detail API, lihat Memulai DICOMImport Job di Referensi AWS SDK untuk C++ API.

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

CLI

AWS CLI

Untuk memulai pekerjaan impor dicom

Contoh `start-dicom-import-job` kode berikut memulai pekerjaan impor dicom.

```
aws medical-imaging start-dicom-import-job \
    --job-name "my-job" \
    --datastore-id "12345678901234567890123456789012" \
    --input-s3-uri "s3://medical-imaging-dicom-input/dicom_input/" \
    --output-s3-uri "s3://medical-imaging-output/job_output/" \
    --data-access-role-arn "arn:aws:iam::123456789012:role/
ImportJobDataAccessRole"
```

Output:

```
{
    "datastoreId": "12345678901234567890123456789012",
    "jobId": "09876543210987654321098765432109",
    "jobStatus": "SUBMITTED",
    "submittedAt": "2022-08-12T11:28:11.152000+00:00"
}
```

Untuk informasi selengkapnya, lihat Memulai pekerjaan impor di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

```
    public static String startDicomImportJob(MedicalImagingClient
 medicalImagingClient,
            String jobName,
            String datastoreId,
            String dataAccessRoleArn,
            String inputS3Uri,
            String outputS3Uri) {
```

```
        try {
            StartDicomImportJobRequest startDicomImportJobRequest =
 StartDicomImportJobRequest.builder()
                    .jobName(jobName)
                    .datastoreId(datastoreId)
                    .dataAccessRoleArn(dataAccessRoleArn)
                    .inputS3Uri(inputS3Uri)
                    .outputS3Uri(outputS3Uri)
                    .build();
            StartDicomImportJobResponse response =
 medicalImagingClient.startDICOMImportJob(startDicomImportJobRequest);
            return response.jobId();
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
```

- Untuk detail API, lihat Memulai DICOMImport Job di Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

```
import { StartDICOMImportJobCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} jobName - The name of the import job.
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} dataAccessRoleArn - The Amazon Resource Name (ARN) of the role
 that grants permission.
```

```
 * @param {string} inputS3Uri - The URI of the S3 bucket containing the input
 files.
 * @param {string} outputS3Uri - The URI of the S3 bucket where the output files
 are stored.
 */
export const startDicomImportJob = async (
  jobName = "test-1",
  datastoreId = "12345678901234567890123456789012",
  dataAccessRoleArn = "arn:aws:iam::xxxxxxxxxxxx:role/ImportJobDataAccessRole",
  inputS3Uri = "s3://medical-imaging-dicom-input/dicom_input/",
  outputS3Uri = "s3://medical-imaging-output/job_output/",
) => {
  const response = await medicalImagingClient.send(
    new StartDICOMImportJobCommand({
      jobName: jobName,
      datastoreId: datastoreId,
      dataAccessRoleArn: dataAccessRoleArn,
      inputS3Uri: inputS3Uri,
      outputS3Uri: outputS3Uri,
    }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //     httpStatusCode: 200,
  //         requestId: '6e81d191-d46b-4e48-a08a-cdcc7e11eb79',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  // },
  //     datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //     jobId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //     jobStatus: 'SUBMITTED',
  //     submittedAt: 2023-09-22T14:48:45.767Z
  // }
  return response;
};
```

- Untuk detail API, lihat Memulai DICOMImport Job di Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def start_dicom_import_job(
        self, job_name, datastore_id, role_arn, input_s3_uri, output_s3_uri
    ):
        """
        Start a DICOM import job.

        :param job_name: The name of the job.
        :param datastore_id: The ID of the data store.
        :param role_arn: The Amazon Resource Name (ARN) of the role to use for
 the job.
        :param input_s3_uri: The S3 bucket input prefix path containing the DICOM
 files.
        :param output_s3_uri: The S3 bucket output prefix path for the result.
        :return: The job ID.
        """
        try:
            job = self.health_imaging_client.start_dicom_import_job(
                jobName=job_name,
                datastoreId=datastore_id,
                dataAccessRoleArn=role_arn,
                inputS3Uri=input_s3_uri,
                outputS3Uri=output_s3_uri,
            )
        except ClientError as err:
            logger.error(
                "Couldn't start DICOM import job. Here's why: %s: %s",
                err.response["Error"]["Code"],
```

```
                err.response["Error"]["Message"],
            )
            raise
        else:
            return job["jobId"]
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
client = boto3.client("medical-imaging")
medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Untuk detail API, lihat Memulai DICOMImport 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

```
TRY.
    " iv_job_name = 'import-job-1'
    " iv_datastore_id = '12345678901234567890123456789012345678901234567890'
    " iv_role_arn = 'arn:aws:iam::123456789012:role/ImportJobRole'
    " iv_input_s3_uri = 's3://my-bucket/input/'
    " iv_output_s3_uri = 's3://my-bucket/output/'
    oo_result = lo_mig->startdicomimportjob(
      iv_jobname = iv_job_name
      iv_datastoreid = iv_datastore_id
      iv_dataaccessrolearn = iv_role_arn
      iv_inputs3uri = iv_input_s3_uri
      iv_outputs3uri = iv_output_s3_uri ).
    DATA(lv_job_id) = oo_result->get_jobid( ).
    MESSAGE |DICOM import job started with ID: { lv_job_id }.| TYPE 'I'.
```

```
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
      CATCH /aws1/cx_migservicequotaexcdex.
        MESSAGE 'Service quota exceeded.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

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

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

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

Mendapatkan properti pekerjaan impor

Gunakan `GetDICOMImportJob` tindakan untuk mempelajari lebih lanjut tentang properti pekerjaan HealthImaging impor AWS. Misalnya, setelah memulai pekerjaan impor, Anda dapat menjalankan `GetDICOMImportJob` untuk menemukan status pekerjaan. Setelah `jobStatus` kembali sebagai`COMPLETED`, Anda siap untuk mengakses set gambar Anda.

> **Note**
>
> `jobStatus`Mengacu pada pelaksanaan pekerjaan impor. Oleh karena itu, pekerjaan impor dapat mengembalikan seolah-olah masalah validasi ditemukan selama proses impor. `jobStatus` `COMPLETED` Jika `jobStatus` pengembalian sebagai`COMPLETED`, kami tetap menyarankan Anda meninjau manifes keluaran yang ditulis ke Amazon S3, karena mereka memberikan detail tentang keberhasilan atau kegagalan impor objek P10 individual.

Menu berikut memberikan prosedur untuk contoh Konsol Manajemen AWS dan kode untuk AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat `GetDICOMImportJob`di AWS HealthImaging API Referensi.

Untuk mendapatkan properti pekerjaan impor

Pilih menu berdasarkan preferensi akses Anda ke AWS HealthImaging.

AWS Konsol

1. Buka halaman HealthImaging Console Data Stores.
2. Pilih penyimpanan data.

   Halaman detail penyimpanan data terbuka. Tab Image sets dipilih secara default.
3. Pilih tab Impor.
4. Pilih pekerjaan impor.

   Halaman Impor detail pekerjaan membuka dan menampilkan properti tentang pekerjaan impor.

AWS CLI dan SDKs

C++

SDK untuk C++

```
//! Routine which gets a HealthImaging DICOM import job's properties.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param importJobID: The DICOM import job ID
  \param clientConfig: Aws client configuration.
```

```
  \return GetDICOMImportJobOutcome: The import job outcome.
*/
Aws::MedicalImaging::Model::GetDICOMImportJobOutcome
AwsDoc::Medical_Imaging::getDICOMImportJob(const Aws::String &dataStoreID,
                                           const Aws::String &importJobID,
                                           const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::GetDICOMImportJobRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetJobId(importJobID);
    Aws::MedicalImaging::Model::GetDICOMImportJobOutcome outcome =
 client.GetDICOMImportJob(
            request);
    if (!outcome.IsSuccess()) {
        std::cerr << "GetDICOMImportJob error: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome;
}
```

- Untuk detail API, lihat Mendapatkan DICOMImport Job di Referensi AWS SDK untuk C++ API.

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

CLI

AWS CLI

Untuk mendapatkan properti pekerjaan impor dicom

Contoh `get-dicom-import-job` kode berikut mendapatkan properti pekerjaan dicom import.

```
aws medical-imaging get-dicom-import-job \
```

```
    --datastore-id "12345678901234567890123456789012" \
    --job-id "09876543210987654321098765432109"
```

Output:

```
{
    "jobProperties": {
        "jobId": "09876543210987654321098765432109",
        "jobName": "my-job",
        "jobStatus": "COMPLETED",
        "datastoreId": "12345678901234567890123456789012",
        "dataAccessRoleArn": "arn:aws:iam::123456789012:role/
ImportJobDataAccessRole",
        "endedAt": "2022-08-12T11:29:42.285000+00:00",
        "submittedAt": "2022-08-12T11:28:11.152000+00:00",
        "inputS3Uri": "s3://medical-imaging-dicom-input/dicom_input/",
        "outputS3Uri": "s3://medical-imaging-output/
job_output/12345678901234567890123456789012-
DicomImport-09876543210987654321098765432109/"
    }
}
```

Untuk informasi selengkapnya, lihat Mendapatkan properti pekerjaan impor di Panduan AWS HealthImaging Pengembang.

- Untuk detail API, lihat Mendapatkan DICOMImport Job di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

```
   public static DICOMImportJobProperties getDicomImportJob(MedicalImagingClient
 medicalImagingClient,
            String datastoreId,
            String jobId) {

        try {
            GetDicomImportJobRequest getDicomImportJobRequest =
 GetDicomImportJobRequest.builder()
                    .datastoreId(datastoreId)
                    .jobId(jobId)
                    .build();
```

```
            GetDicomImportJobResponse response =
 medicalImagingClient.getDICOMImportJob(getDicomImportJobRequest);
            return response.jobProperties();
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

- Untuk detail API, lihat Mendapatkan DICOMImport Job di Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

```
import { GetDICOMImportJobCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} jobId - The ID of the import job.
 */
export const getDICOMImportJob = async (
  datastoreId = "xxxxxxxxxxxxxxxxxxxx",
  jobId = "xxxxxxxxxxxxxxxxxxxx",
) => {
  const response = await medicalImagingClient.send(
    new GetDICOMImportJobCommand({ datastoreId: datastoreId, jobId: jobId }),
  );
  console.log(response);
  // {
  //     '$metadata': {
```

```
  //     httpStatusCode: 200,
  //         requestId: 'a2637936-78ea-44e7-98b8-7a87d95dfaee',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  // },
  //     jobProperties: {
  //         dataAccessRoleArn: 'arn:aws:iam::xxxxxxxxxxxx:role/dicom_import',
  //             datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxx',
  //             endedAt: 2023-09-19T17:29:21.753Z,
  //             inputS3Uri: 's3://healthimaging-source/CTStudy/',
  //             jobId: ''xxxxxxxxxxxxxxxxxxxxxxxx'',
  //             jobName: 'job_1',
  //             jobStatus: 'COMPLETED',
  //             outputS3Uri: 's3://health-imaging-dest/
ouput_ct/'xxxxxxxxxxxxxxxxxxxxxxxx'-DicomImport-'xxxxxxxxxxxxxxxxxxxxxxxx'/',
  //             submittedAt: 2023-09-19T17:27:25.143Z
  //     }
  // }

  return response;
};
```

- Untuk detail API, lihat Mendapatkan DICOMImport Job di Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client
```

```
    def get_dicom_import_job(self, datastore_id, job_id):
        """
        Get the properties of a DICOM import job.

        :param datastore_id: The ID of the data store.
        :param job_id: The ID of the job.
        :return: The job properties.
        """
        try:
            job = self.health_imaging_client.get_dicom_import_job(
                jobId=job_id, datastoreId=datastore_id
            )
        except ClientError as err:
            logger.error(
                "Couldn't get DICOM import job. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return job["jobProperties"]
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Untuk detail API, lihat Mendapatkan DICOMImport 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

```
TRY.
    " iv_datastore_id = '12345678901234567890123456789012345678901234567890'
    " iv_job_id = '12345678901234567890123456789012'
    oo_result = lo_mig->getdicomimportjob(
      iv_datastoreid = iv_datastore_id
      iv_jobid = iv_job_id ).
    DATA(lo_job_props) = oo_result->get_jobproperties( ).
    DATA(lv_job_status) = lo_job_props->get_jobstatus( ).
    MESSAGE |Job status: { lv_job_status }.| TYPE 'I'.
  CATCH /aws1/cx_migaccessdeniedex.
    MESSAGE 'Access denied.' TYPE 'I'.
  CATCH /aws1/cx_migconflictexception.
    MESSAGE 'Conflict error.' TYPE 'I'.
  CATCH /aws1/cx_miginternalserverex.
    MESSAGE 'Internal server error.' TYPE 'I'.
  CATCH /aws1/cx_migresourcenotfoundex.
    MESSAGE 'Job not found.' TYPE 'I'.
  CATCH /aws1/cx_migthrottlingex.
    MESSAGE 'Request throttled.' TYPE 'I'.
  CATCH /aws1/cx_migvalidationex.
    MESSAGE 'Validation error.' TYPE 'I'.
ENDTRY.
```

- Untuk detail API, lihat Mendapatkan DICOMImport Job di AWS SDK untuk referensi SAP ABAP API.

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

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

Daftar pekerjaan impor

Gunakan `ListDICOMImportJobs` tindakan untuk mencantumkan pekerjaan impor yang dibuat untuk penyimpanan HealthImaging data tertentu. Menu berikut memberikan prosedur untuk contoh Konsol Manajemen AWS dan kode untuk AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat `ListDICOMImportJobs`di AWS HealthImaging API Referensi.

> **Note**
>
> Pekerjaan impor disimpan dalam daftar pekerjaan selama 90 hari dan kemudian diarsipkan.

Untuk daftar pekerjaan impor

Pilih menu berdasarkan preferensi akses Anda ke AWS HealthImaging.

AWS Konsol

1. Buka halaman HealthImaging Console Data Stores.
2. Pilih penyimpanan data.

   Halaman detail penyimpanan data terbuka. Tab Image sets dipilih secara default.
3. Pilih tab Impor untuk mencantumkan semua pekerjaan impor terkait.

AWS CLI dan SDKs

CLI

AWS CLI

Untuk daftar pekerjaan dicom import

Contoh `list-dicom-import-jobs` kode berikut mencantumkan pekerjaan impor dicom.

```
aws medical-imaging list-dicom-import-jobs \
    --datastore-id "12345678901234567890123456789012"
```

Output:

```
{
    "jobSummaries": [
        {
            "jobId": "09876543210987654321098765432109",
            "jobName": "my-job",
            "jobStatus": "COMPLETED",
            "datastoreId": "12345678901234567890123456789012",
            "dataAccessRoleArn": "arn:aws:iam::123456789012:role/
ImportJobDataAccessRole",
            "endedAt": "2022-08-12T11:21:56.504000+00:00",
            "submittedAt": "2022-08-12T11:20:21.734000+00:00"
        }
    ]
}
```

Untuk informasi selengkapnya, lihat Daftar lowongan impor di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

```
   public static List<DICOMImportJobSummary>
listDicomImportJobs(MedicalImagingClient medicalImagingClient,
            String datastoreId) {

        try {
            ListDicomImportJobsRequest listDicomImportJobsRequest =
ListDicomImportJobsRequest.builder()
                    .datastoreId(datastoreId)
                    .build();
            ListDicomImportJobsResponse response =
medicalImagingClient.listDICOMImportJobs(listDicomImportJobsRequest);
            return response.jobSummaries();
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return new ArrayList<>();
```

```
    }
```

- Untuk detail API, lihat Daftar DICOMImport Pekerjaan di Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

```
import { paginateListDICOMImportJobs } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store.
 */
export const listDICOMImportJobs = async (
  datastoreId = "xxxxxxxxxxxxxxxxx",
) => {
  const paginatorConfig = {
    client: medicalImagingClient,
    pageSize: 50,
  };

  const commandParams = { datastoreId: datastoreId };
  const paginator = paginateListDICOMImportJobs(paginatorConfig, commandParams);

  const jobSummaries = [];
  for await (const page of paginator) {
    // Each page contains a list of `jobSummaries`. The list is truncated if is
 larger than `pageSize`.
    jobSummaries.push(...page.jobSummaries);
    console.log(page);
  }
  // {
  //     '$metadata': {
```

```
  //     httpStatusCode: 200,
  //         requestId: '3c20c66e-0797-446a-a1d8-91b742fd15a0',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  // },
  //     jobSummaries: [
  //         {
  //             dataAccessRoleArn: 'arn:aws:iam::xxxxxxxxxxxx:role/
dicom_import',
  //             datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxx',
  //             endedAt: 2023-09-22T14:49:51.351Z,
  //             jobId: 'xxxxxxxxxxxxxxxxxxxxxxxx',
  //             jobName: 'test-1',
  //             jobStatus: 'COMPLETED',
  //             submittedAt: 2023-09-22T14:48:45.767Z
  // }
  // ]}

  return jobSummaries;
};
```

- Untuk detail API, lihat Daftar DICOMImport Pekerjaan di Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client
```

```
    def list_dicom_import_jobs(self, datastore_id):
        """
        List the DICOM import jobs.

        :param datastore_id: The ID of the data store.
        :return: The list of jobs.
        """
        try:
            paginator = self.health_imaging_client.get_paginator(
                "list_dicom_import_jobs"
            )
            page_iterator = paginator.paginate(datastoreId=datastore_id)
            job_summaries = []
            for page in page_iterator:
                job_summaries.extend(page["jobSummaries"])
        except ClientError as err:
            logger.error(
                "Couldn't list DICOM import jobs. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return job_summaries
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
client = boto3.client("medical-imaging")
medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Untuk detail API, lihat Daftar DICOMImport 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

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678901234567890'
        oo_result = lo_mig->listdicomimportjobs( iv_datastoreid =
iv_datastore_id ).
        DATA(lt_jobs) = oo_result->get_jobsummaries( ).
        DATA(lv_count) = lines( lt_jobs ).
        MESSAGE |Found { lv_count } DICOM import jobs.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

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

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

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

Mengakses set gambar dengan AWS HealthImaging

Mengakses data pencitraan medis di AWS HealthImaging biasanya melibatkan pencarian kumpulan gambar dengan kunci unik dan mendapatkan metadata dan bingkai gambar terkait (data piksel).

> **Penting:**
>
> Selama impor, HealthImaging memproses binari instance DICOM (`.dcm`file) dan mengubahnya menjadi kumpulan gambar. Gunakan tindakan bawaan HealthImaging cloud (APIs) untuk mengelola penyimpanan data dan kumpulan gambar. Gunakan HealthImaging representasi DICOMweb layanan untuk mengembalikan DICOMweb respons.

Topik berikut menjelaskan cara menggunakan tindakan asli HealthImaging cloud di Konsol Manajemen AWS, AWS CLI, dan AWS SDKs untuk mencari kumpulan gambar dan mendapatkan properti, metadata, dan bingkai gambar terkait.

Topik

Memahami set gambar

Kumpulan gambar adalah AWS yang menyerupai Seri DICOM, dan berfungsi sebagai dasar untuk AWS HealthImaging. Kumpulan gambar dibuat saat Anda mengimpor data DICOM Anda ke dalam HealthImaging. Layanan ini mencoba untuk mengatur data P10 yang diimpor sesuai dengan hierarki DICOM Studi, Seri, dan Instance.

Kumpulan gambar diperkenalkan karena alasan berikut:

- Mendukung berbagai macam alur kerja pencitraan medis (klinis dan nonklinis) melalui fleksibel. APIs

- Menyediakan mekanisme untuk menyimpan dan merekonsiliasi data duplikat dan tidak konsisten secara tahan lama. Data P10 yang diimpor yang bertentangan dengan kumpulan gambar utama yang sudah ada di toko akan dipertahankan sebagai non-primer. Setelah menyelesaikan konflik metadata, data dapat dibuat primer.
- Maksimalkan keselamatan pasien dengan mengelompokkan hanya data terkait.
- Dorong data untuk dibersihkan untuk membantu meningkatkan visibilitas inkonsistensi. Untuk informasi selengkapnya, lihat Memodifikasi set gambar.

> **Penting:**
>
> Penggunaan klinis data DICOM sebelum dibersihkan dapat mengakibatkan kerusakan pasien.

Menu berikut menjelaskan kumpulan gambar secara lebih rinci dan memberikan contoh dan diagram untuk membantu Anda memahami fungsionalitas dan tujuannya. HealthImaging

Apa itu set gambar?

Kumpulan gambar adalah AWS konsep yang mendefinisikan mekanisme pengelompokan abstrak untuk mengoptimalkan data pencitraan medis terkait yang sangat mirip dengan Seri DICOM. Saat Anda mengimpor data pencitraan DICOM P10 ke penyimpanan HealthImaging data AWS, data tersebut diubah menjadi kumpulan gambar yang terdiri dari metadata dan bingkai gambar (data piksel).

> **Note**
>
> Metadata set gambar dinormalisasi. Dengan kata lain, satu set atribut dan nilai umum dipetakan ke elemen tingkat Pasien, Studi, dan Seri yang tercantum dalam Registry of DICOM Data Elements. HealthImaging menggunakan elemen DICOM berikut saat mengelompokkan objek DICOM P10 yang masuk ke dalam kumpulan gambar.
>
> Elemen DICOM yang digunakan untuk pembuatan set gambar

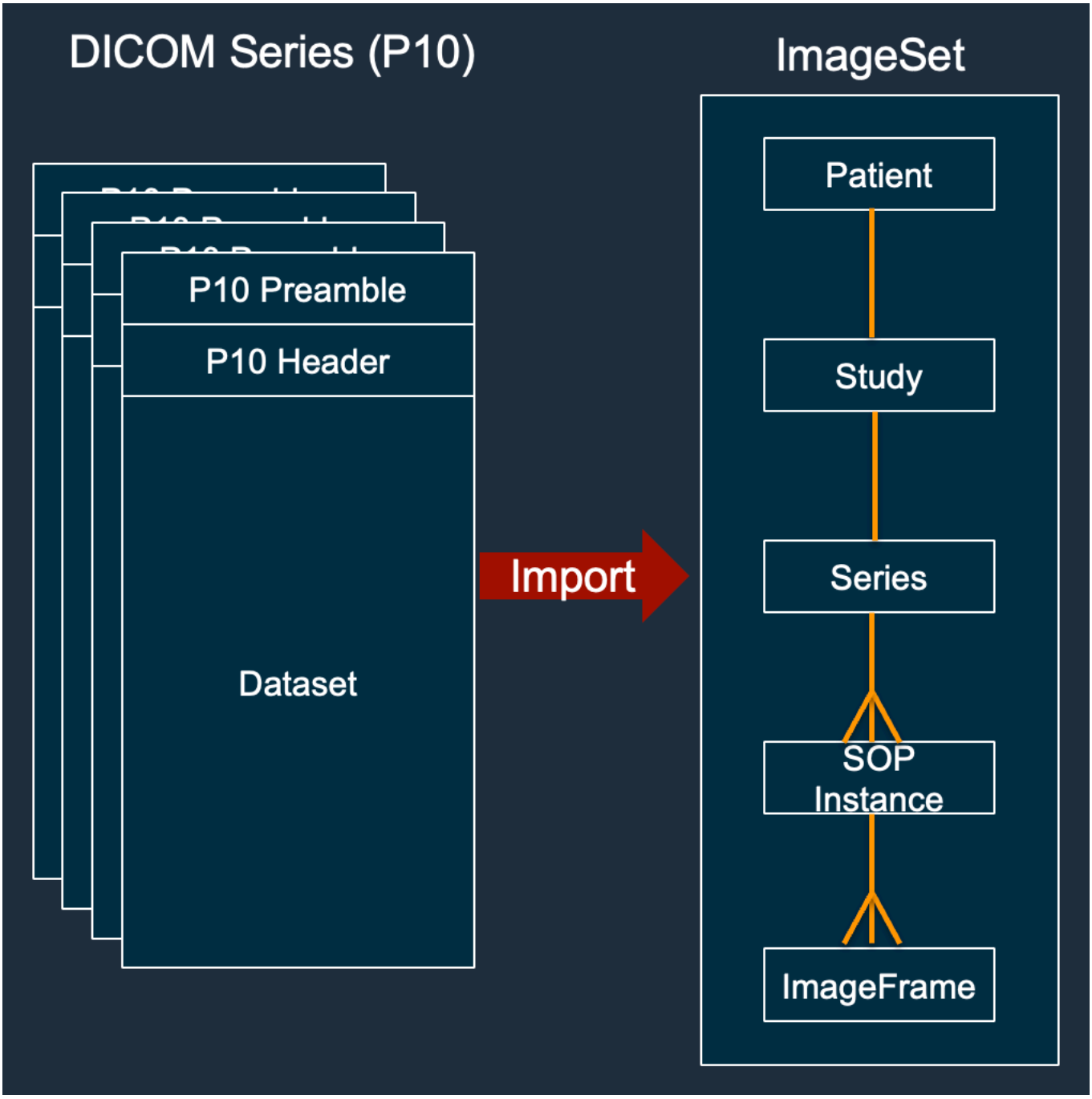
Nama elemen	Tag elemen
Elemen tingkat studi	
`Study Date`	`(0008,0020)`

Nama elemen	Tag elemen
`Accession Number`	`(0008,0050)`
`Patient ID`	`(0010,0020)`
`Study Instance UID`	`(0020,000D)`
`Study ID`	`(0020,0010)`
Elemen tingkat seri	
`Series Instance UID`	`(0020,000E)`
`Series Number`	`(0020,0011)`

Selama impor, beberapa set gambar mempertahankan pengkodean sintaks transfer aslinya, sementara yang lain ditranskode ke High-Throughput JPEG 2000 (K) lossless secara default. HTJ2 Jika kumpulan gambar dikodekan dalam HTJ2 K, itu harus diterjemahkan sebelum dilihat. Untuk informasi selengkapnya, lihat Sintaks transfer yang didukung dan Pustaka decoding bingkai gambar.
Bingkai gambar (data piksel) dikodekan dalam High-Throughput JPEG 2000 (HTJ2K) dan harus diterjemahkan sebelum dilihat.

Kumpulan gambar adalah AWS sumber daya, sehingga mereka diberi Nama Sumber Daya Amazon (ARNs). Mereka dapat ditandai dengan hingga 50 pasangan nilai kunci dan diberikan kontrol akses berbasis peran (RBAC) dan kontrol akses berbasis atribut (ABAC) melalui IAM. Selain itu, kumpulan gambar diberi versi, sehingga semua perubahan dipertahankan dan versi sebelumnya dapat diakses.

Mengimpor data DICOM P10 menghasilkan kumpulan gambar yang berisi metadata DICOM dan bingkai gambar untuk satu atau beberapa instance Service-Object Pair (SOP) dalam Seri DICOM yang sama.



Note

Pekerjaan impor DICOM:

- Selalu buat set gambar baru atau naikkan versi set gambar yang ada.

- Jangan deduplikat penyimpanan Instance SOP. Setiap impor Instance SOP yang sama menggunakan penyimpanan tambahan sebagai kumpulan gambar non-primer baru, atau versi tambahan dari kumpulan gambar utama yang ada.
- Secara otomatis mengatur instans SOP dengan metadata yang konsisten dan tidak bertentangan sebagai kumpulan gambar utama, yang berisi instance dengan elemen metadata Pasien, Studi, dan Seri yang konsisten.
  - Jika instance yang terdiri dari seri DICOM diimpor dalam dua atau lebih pekerjaan impor, dan instance tidak bertentangan dengan instance yang sudah ada di penyimpanan data, maka semua instance akan diatur dalam satu set gambar Primer.
- Buat kumpulan gambar non-primer yang berisi data DICOM P10 yang bertentangan dengan kumpulan gambar utama yang sudah ada di penyimpanan data.
- Pertahankan data yang paling baru diterima sebagai versi terbaru dari kumpulan gambar utama.
  - Jika instance yang terdiri dari seri DICOM adalah kumpulan gambar utama, dan satu instance diimpor lagi, salinan baru akan dimasukkan ke dalam kumpulan gambar utama, dan versi akan bertambah.

Seperti apa tampilan metadata set gambar?

Gunakan `GetImageSetMetadata` tindakan untuk mengambil metadata set gambar. Metadata yang dikembalikan dikompresi dengan`zip`, jadi Anda harus membuka ritsletingnya sebelum melihat. Untuk informasi selengkapnya, lihat Mendapatkan metadata set gambar.

Contoh berikut menunjukkan struktur metadata set gambar dalam format JSON.

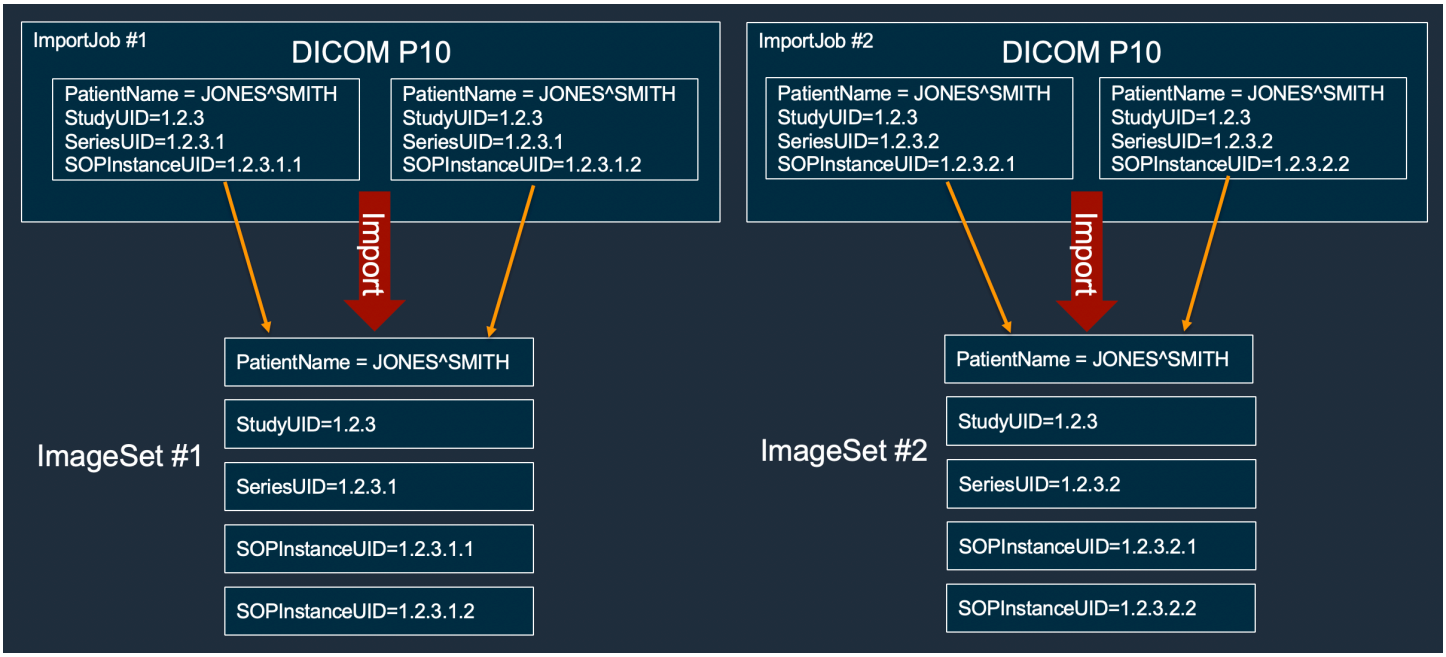
```
{
 "SchemaVersion": "1.1",
 "DatastoreID": "2aa75d103f7f45ab977b0e93f00e6fe9",
 "ImageSetID": "46923b66d5522e4241615ecd64637584",
 "Patient": {
  "DICOM": {
   "PatientBirthDate": null,
   "PatientSex": null,
   "PatientID": "2178309",
   "PatientName": "MISTER^CT"
  }
 },
```

```
 "Study": {
  "DICOM": {
   "StudyTime": "083501",
   "PatientWeight": null
  },
  "Series": {
   "1.2.840.113619.2.30.1.1762295590.1623.978668949.887": {
    "DICOM": {
        "Modality": "CT",
     "PatientPosition": "FFS"
    },
    "Instances": {
     "1.2.840.113619.2.30.1.1762295590.1623.978668949.888": {
      "DICOM": {
       "SourceApplicationEntityTitle": null,
       "SOPClassUID": "1.2.840.10008.5.1.4.1.1.2",
       "HighBit": 15,
       "PixelData": null,
       "Exposure": "40",
       "RescaleSlope": "1",
      "ImageFrames": [
       {
        "ID": "0d1c97c51b773198a3df44383a5fd306",
        "PixelDataChecksumFromBaseToFullResolution": [
         {
          "Width": 256,
          "Height": 188,
          "Checksum": 2598394845
         },
         {
          "Width": 512,
          "Height": 375,
          "Checksum": 1227709180
         }
        ],
        "MinPixelValue": 451,
        "MaxPixelValue": 1466,
        "FrameSizeInBytes": 384000
       }
      ]
     }
    }
   }
  }
 }
```

```
  }
}
```

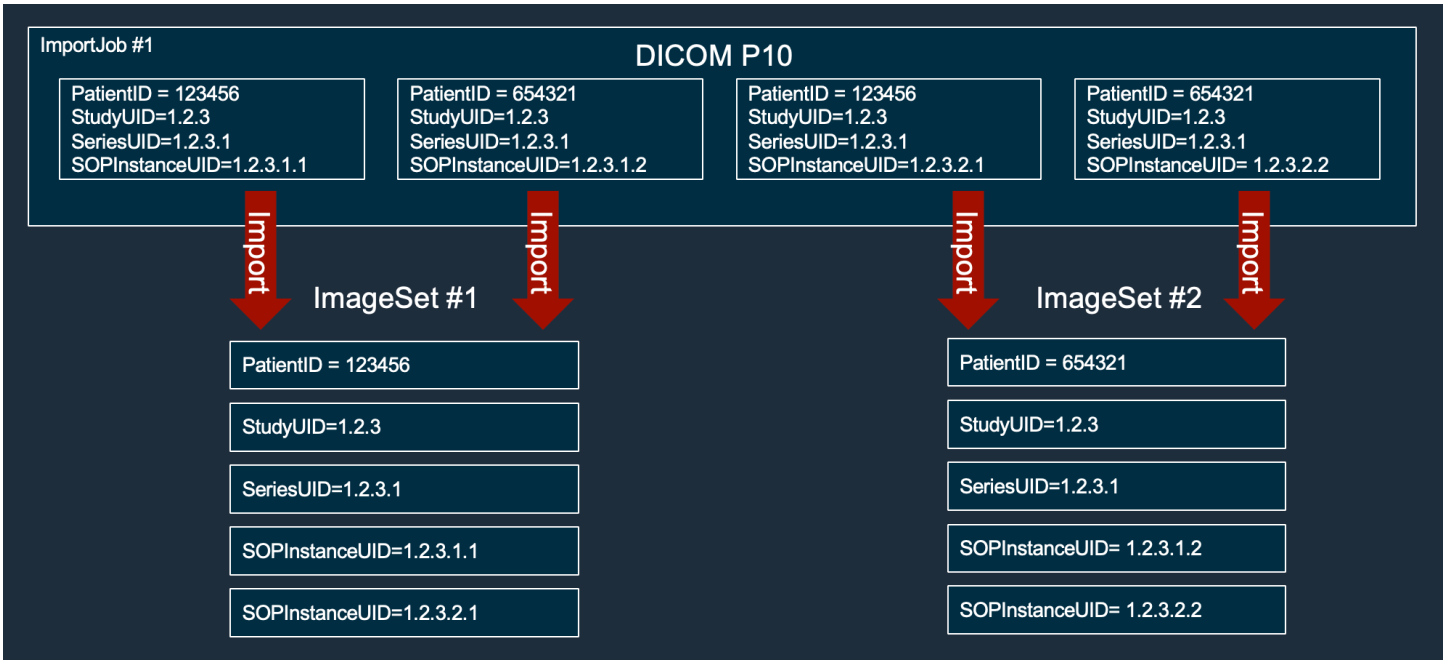
Contoh pembuatan set gambar: beberapa pekerjaan impor

Contoh berikut menunjukkan bagaimana beberapa pekerjaan impor selalu membuat set gambar baru dan tidak pernah menambahkan ke yang sudah ada.



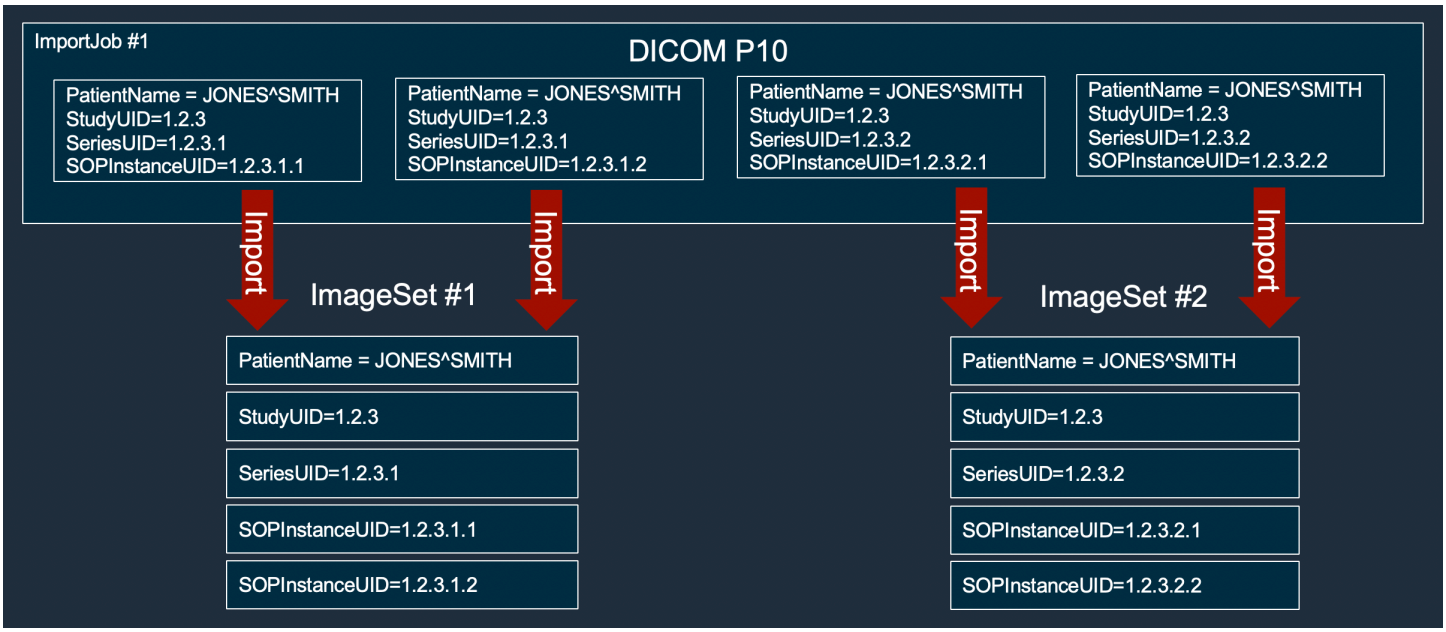
Contoh pembuatan set gambar: pekerjaan impor tunggal dengan dua varian

Contoh berikut menunjukkan pekerjaan impor tunggal yang akan gagal digabungkan menjadi satu set gambar karena instance 1 dan 3 memiliki Pasien yang berbeda IDs dari instance 2 dan 4. Untuk mengatasi hal ini, Anda dapat menggunakan `UpdateImageSetMetadata` tindakan untuk menyelesaikan konflik ID Pasien dengan set gambar Primer yang ada. Setelah konflik diselesaikan, Anda dapat menggunakan `CopyImageSet` tindakan dengan argumen `--promoteToPrimary` untuk menambahkan gambar yang disetel ke kumpulan gambar Primer.



Contoh pembuatan set gambar: pekerjaan impor tunggal dengan pengoptimalan

Contoh berikut menunjukkan pekerjaan impor tunggal yang membuat dua set gambar untuk meningkatkan throughput, meskipun nama pasien cocok.



Mencari set gambar

Gunakan `SearchImageSets` tindakan untuk menjalankan kueri penelusuran terhadap semua kumpulan gambar di penyimpanan `ACTIVE` HealthImaging data. Menu berikut memberikan prosedur

untuk contoh Konsol Manajemen AWS dan kode untuk AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat `SearchImageSets`di AWS HealthImaging API Referensi.

> **Note**
>
> Ingatlah poin-poin berikut saat mencari set gambar.
>
> - `SearchImageSets`menerima parameter kueri penelusuran tunggal dan mengembalikan respons paginasi dari semua kumpulan gambar yang memiliki kriteria pencocokan. Semua kueri rentang tanggal harus dimasukkan sebagai`(lowerBound, upperBound)`.
> - Secara default, `SearchImageSets` gunakan `updatedAt` bidang untuk menyortir dalam urutan menurun dari yang terbaru ke yang terlama.
> - Jika Anda membuat penyimpanan data dengan AWS KMS kunci milik pelanggan, Anda harus memperbarui kebijakan AWS KMS kunci sebelum berinteraksi dengan kumpulan gambar. Untuk informasi selengkapnya, lihat Membuat kunci terkelola pelanggan.

Untuk mencari set gambar

Pilih menu berdasarkan preferensi akses Anda ke AWS HealthImaging.

AWS Konsol

> **Note**
>
> Prosedur berikut menunjukkan cara mencari set gambar menggunakan filter `Series Instance UID` dan `Updated at` properti.

Series Instance UID

Cari set gambar menggunakan filter **`Series Instance UID`** properti

1. Buka halaman HealthImaging Console Data Stores.
2. Pilih penyimpanan data.

   Halaman detail penyimpanan data terbuka dan tab Image sets dipilih secara default.
3. Pilih menu filter properti dan pilih`Series Instance UID`.
4. Di kolom Masukkan nilai untuk dicari, masukkan (tempel) UID Instans Seri yang menarik.

> **Note**
>
> Nilai UID Instance Seri harus identik dengan yang tercantum dalam Registry of DICOM Unique Identifiers (). UIDs Perhatikan persyaratan termasuk serangkaian angka yang berisi setidaknya satu periode di antara mereka. Periode tidak diperbolehkan pada awal atau akhir Instans Seri UIDs. Huruf dan spasi putih tidak diperbolehkan, jadi berhati-hatilah saat menyalin dan menempelkan UIDs.

5. Pilih menu Rentang tanggal, pilih rentang tanggal untuk UID Instans Seri, dan pilih Terapkan.
6. Pilih Cari.

   Instance Seri UIDs yang termasuk dalam rentang tanggal yang dipilih dikembalikan dalam urutan Terbaru secara default.

Updated at

Cari set gambar menggunakan filter **`Updated at`** properti

1. Buka halaman HealthImaging Console Data Stores.
2. Pilih penyimpanan data.

   Halaman detail penyimpanan data terbuka dan tab Image sets dipilih secara default.
3. Pilih menu filter properti dan pilih`Updated at`.
4. Pilih menu Rentang tanggal, pilih rentang tanggal yang ditetapkan gambar, dan pilih Terapkan.
5. Pilih Cari.

   Kumpulan gambar yang termasuk dalam rentang tanggal yang dipilih dikembalikan dalam urutan Terbaru secara default.

AWS CLI dan SDKs

C++

SDK untuk C++

Fungsi utilitas untuk mencari set gambar.

```
//! Routine which searches for image sets based on defined input attributes.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param searchCriteria: A search criteria instance.
  \param imageSetResults: Vector to receive the image set IDs.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
  */
bool AwsDoc::Medical_Imaging::searchImageSets(const Aws::String &dataStoreID,
                                              const
 Aws::MedicalImaging::Model::SearchCriteria &searchCriteria,
                                              Aws::Vector<Aws::String>
 &imageSetResults,
                                              const
 Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::SearchImageSetsRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetSearchCriteria(searchCriteria);

    Aws::String nextToken; // Used for paginated results.
    bool result = true;
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::MedicalImaging::Model::SearchImageSetsOutcome outcome =
 client.SearchImageSets(
                request);
        if (outcome.IsSuccess()) {
            for (auto &imageSetMetadataSummary:
 outcome.GetResult().GetImageSetsMetadataSummaries()) {

 imageSetResults.push_back(imageSetMetadataSummary.GetImageSetId());
            }

            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cout << "Error: " << outcome.GetError().GetMessage() <<
 std::endl;
            result = false;
```

```
        }
    } while (!nextToken.empty());

    return result;
}
```

Kasus penggunaan #1: operator EQUAL.

```
       Aws::Vector<Aws::String> imageIDsForPatientID;
       Aws::MedicalImaging::Model::SearchCriteria searchCriteriaEqualsPatientID;
       Aws::Vector<Aws::MedicalImaging::Model::SearchFilter>
patientIDSearchFilters = {

Aws::MedicalImaging::Model::SearchFilter().WithOperator(Aws::MedicalImaging::Model::Oper

.WithValues({Aws::MedicalImaging::Model::SearchByAttributeValue().WithDICOMPatientId(pat
       };

       searchCriteriaEqualsPatientID.SetFilters(patientIDSearchFilters);
       bool result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,

searchCriteriaEqualsPatientID,

imageIDsForPatientID,
                                                              clientConfig);
       if (result) {
           std::cout << imageIDsForPatientID.size() << " image sets found for
the patient with ID '"
           <<  patientID << "'." << std::endl;
           for (auto &imageSetResult : imageIDsForPatientID) {
               std::cout << "  Image set with ID '" << imageSetResult <<
std::endl;
           }
       }
```

Kasus penggunaan #2: ANTARA operator menggunakan DICOMStudy Tanggal dan DICOMStudy Waktu.

```
        Aws::MedicalImaging::Model::SearchByAttributeValue useCase2StartDate;
```

```
 useCase2StartDate.SetDICOMStudyDateAndTime(Aws::MedicalImaging::Model::DICOMStudyDateAnd
        .WithDICOMStudyDate("19990101")
        .WithDICOMStudyTime("000000.000"));

        Aws::MedicalImaging::Model::SearchByAttributeValue useCase2EndDate;

 useCase2EndDate.SetDICOMStudyDateAndTime(Aws::MedicalImaging::Model::DICOMStudyDateAndTi

 .WithDICOMStudyDate(Aws::Utils::DateTime(std::chrono::system_clock::now()).ToLocalTimeSt
%m%d"))
        .WithDICOMStudyTime("000000.000"));

        Aws::MedicalImaging::Model::SearchFilter useCase2SearchFilter;
        useCase2SearchFilter.SetValues({useCase2StartDate, useCase2EndDate});

 useCase2SearchFilter.SetOperator(Aws::MedicalImaging::Model::Operator::BETWEEN);

        Aws::MedicalImaging::Model::SearchCriteria useCase2SearchCriteria;
        useCase2SearchCriteria.SetFilters({useCase2SearchFilter});

        Aws::Vector<Aws::String> usesCase2Results;
        result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,
                                                          useCase2SearchCriteria,
                                                          usesCase2Results,
                                                          clientConfig);
        if (result) {
            std::cout << usesCase2Results.size() << " image sets found for
 between 1999/01/01 and present."
                      <<  std::endl;
            for (auto &imageSetResult : usesCase2Results) {
                std::cout << "  Image set with ID '" << imageSetResult <<
 std::endl;
            }
        }
```

Kasus penggunaan #3: ANTARA operator menggunakan createDat. Studi waktu sebelumnya bertahan.

```
        Aws::MedicalImaging::Model::SearchByAttributeValue useCase3StartDate;

 useCase3StartDate.SetCreatedAt(Aws::Utils::DateTime("20231130T000000000Z",Aws::Utils::Da
```

```
        Aws::MedicalImaging::Model::SearchByAttributeValue useCase3EndDate;

useCase3EndDate.SetCreatedAt(Aws::Utils::DateTime(std::chrono::system_clock::now()));

        Aws::MedicalImaging::Model::SearchFilter useCase3SearchFilter;
        useCase3SearchFilter.SetValues({useCase3StartDate, useCase3EndDate});

useCase3SearchFilter.SetOperator(Aws::MedicalImaging::Model::Operator::BETWEEN);

        Aws::MedicalImaging::Model::SearchCriteria useCase3SearchCriteria;
        useCase3SearchCriteria.SetFilters({useCase3SearchFilter});

        Aws::Vector<Aws::String> usesCase3Results;
        result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,
                                                          useCase3SearchCriteria,
                                                          usesCase3Results,
                                                          clientConfig);
        if (result) {
            std::cout << usesCase3Results.size() << " image sets found for
created between 2023/11/30 and present."
                      <<  std::endl;
            for (auto &imageSetResult : usesCase3Results) {
                std::cout << "  Image set with ID '" << imageSetResult <<
std::endl;
            }
        }
```

Kasus penggunaan #4: Operator EQUAL di DICOMSeries InstanceUID dan BETWEET di UpdateDAT dan mengurutkan respons dalam urutan ASC di bidang UpdateDAT.

```
        Aws::MedicalImaging::Model::SearchByAttributeValue useCase4StartDate;

useCase4StartDate.SetUpdatedAt(Aws::Utils::DateTime("20231130T000000000Z",Aws::Utils::Da

        Aws::MedicalImaging::Model::SearchByAttributeValue useCase4EndDate;

useCase4EndDate.SetUpdatedAt(Aws::Utils::DateTime(std::chrono::system_clock::now()));

        Aws::MedicalImaging::Model::SearchFilter useCase4SearchFilterBetween;
        useCase4SearchFilterBetween.SetValues({useCase4StartDate,
useCase4EndDate});
```

```
useCase4SearchFilterBetween.SetOperator(Aws::MedicalImaging::Model::Operator::BETWEEN);

       Aws::MedicalImaging::Model::SearchByAttributeValue seriesInstanceUID;
       seriesInstanceUID.SetDICOMSeriesInstanceUID(dicomSeriesInstanceUID);

       Aws::MedicalImaging::Model::SearchFilter useCase4SearchFilterEqual;
       useCase4SearchFilterEqual.SetValues({seriesInstanceUID});

useCase4SearchFilterEqual.SetOperator(Aws::MedicalImaging::Model::Operator::EQUAL);

       Aws::MedicalImaging::Model::SearchCriteria useCase4SearchCriteria;
       useCase4SearchCriteria.SetFilters({useCase4SearchFilterBetween,
useCase4SearchFilterEqual});

       Aws::MedicalImaging::Model::Sort useCase4Sort;

useCase4Sort.SetSortField(Aws::MedicalImaging::Model::SortField::updatedAt);
       useCase4Sort.SetSortOrder(Aws::MedicalImaging::Model::SortOrder::ASC);

       useCase4SearchCriteria.SetSort(useCase4Sort);

       Aws::Vector<Aws::String> usesCase4Results;
       result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,
                                                         useCase4SearchCriteria,
                                                         usesCase4Results,
                                                         clientConfig);
       if (result) {
           std::cout << usesCase4Results.size() << " image sets found for EQUAL
operator "
           << "on DICOMSeriesInstanceUID and BETWEEN on updatedAt and sort
response\n"
           <<  "in ASC order on updatedAt field." <<  std::endl;
           for (auto &imageSetResult : usesCase4Results) {
               std::cout << "  Image set with ID '" << imageSetResult <<
std::endl;
           }
       }
```

- Untuk detail API, lihat SearchImageSetsdi Referensi AWS SDK untuk C++ API.

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

CLI

AWS CLI

Contoh 1: Untuk mencari set gambar dengan operator EQUAL

Contoh `search-image-sets` kode berikut menggunakan operator EQUAL untuk mencari set gambar berdasarkan nilai tertentu.

```
aws medical-imaging search-image-sets \
    --datastore-id 12345678901234567890123456789012 \
    --search-criteria file://search-criteria.json
```

Isi dari `search-criteria.json`

```
{
    "filters": [{
        "values": [{"DICOMPatientId" : "SUBJECT08701"}],
        "operator": "EQUAL"
    }]
}
```

Output:

```
{
    "imageSetsMetadataSummaries": [{
        "imageSetId": "09876543210987654321098765432109",
        "createdAt": "2022-12-06T21:40:59.429000+00:00",
        "version": 1,
        "DICOMTags": {
            "DICOMStudyId": "2011201407",
            "DICOMStudyDate": "19991122",
             "DICOMPatientSex": "F",
             "DICOMStudyInstanceUID": "1.2.840.99999999.84710745.943275268089",
             "DICOMPatientBirthDate": "19201120",
```

```
            "DICOMStudyDescription": "UNKNOWN",
            "DICOMPatientId": "SUBJECT08701",
            "DICOMPatientName": "Melissa844 Huel628",
            "DICOMNumberOfStudyRelatedInstances": 1,
            "DICOMStudyTime": "140728",
            "DICOMNumberOfStudyRelatedSeries": 1
        },
        "updatedAt": "2022-12-06T21:40:59.429000+00:00"
    }]
}
```

Contoh 2: Untuk mencari set gambar dengan operator BETWEET menggunakan DICOMStudy Tanggal dan DICOMStudy Waktu

Contoh `search-image-sets` kode berikut mencari kumpulan gambar dengan Studi DICOM yang dihasilkan antara 1 Januari 1990 (12:00 AM) dan 1 Januari 2023 (12:00 AM).

Catatan: DICOMStudy Waktu adalah opsional. Jika tidak ada, 12:00 AM (awal hari) adalah nilai waktu untuk tanggal yang disediakan untuk penyaringan.

```
aws medical-imaging search-image-sets \
    --datastore-id 12345678901234567890123456789012 \
    --search-criteria file://search-criteria.json
```

Isi dari `search-criteria.json`

```
{
    "filters": [{
        "values": [{
            "DICOMStudyDateAndTime": {
                "DICOMStudyDate": "19900101",
                "DICOMStudyTime": "000000"
            }
        },
        {
            "DICOMStudyDateAndTime": {
                "DICOMStudyDate": "20230101",
                "DICOMStudyTime": "000000"
            }
        }],
        "operator": "BETWEEN"
    }]
```

```
}
```

Output:

```
{
    "imageSetsMetadataSummaries": [{
        "imageSetId": "09876543210987654321098765432109",
        "createdAt": "2022-12-06T21:40:59.429000+00:00",
        "version": 1,
        "DICOMTags": {
            "DICOMStudyId": "2011201407",
            "DICOMStudyDate": "19991122",
            "DICOMPatientSex": "F",
            "DICOMStudyInstanceUID": "1.2.840.99999999.84710745.943275268089",
            "DICOMPatientBirthDate": "19201120",
            "DICOMStudyDescription": "UNKNOWN",
            "DICOMPatientId": "SUBJECT08701",
            "DICOMPatientName": "Melissa844 Huel628",
            "DICOMNumberOfStudyRelatedInstances": 1,
            "DICOMStudyTime": "140728",
            "DICOMNumberOfStudyRelatedSeries": 1
        },
        "updatedAt": "2022-12-06T21:40:59.429000+00:00"
    }]
}
```

Contoh 3: Untuk mencari set gambar dengan operator BETWEET menggunakan createDat (studi waktu sebelumnya dipertahankan)

Contoh `search-image-sets` kode berikut mencari set gambar dengan Studi DICOM bertahan di HealthImaging antara rentang waktu di zona waktu UTC.

Catatan: Berikan CreateDat dalam format contoh (“1985-04-12T 23:20:50.52 Z”).

```
aws medical-imaging search-image-sets \
    --datastore-id 12345678901234567890123456789012 \
    --search-criteria  file://search-criteria.json
```

Isi dari `search-criteria.json`

```
{
```

```
    "filters": [{
        "values": [{
            "createdAt": "1985-04-12T23:20:50.52Z"
        },
        {
            "createdAt": "2022-04-12T23:20:50.52Z"
        }],
        "operator": "BETWEEN"
    }]
}
```

Output:

```
{
    "imageSetsMetadataSummaries": [{
        "imageSetId": "09876543210987654321098765432109",
        "createdAt": "2022-12-06T21:40:59.429000+00:00",
        "version": 1,
        "DICOMTags": {
            "DICOMStudyId": "2011201407",
            "DICOMStudyDate": "19991122",
            "DICOMPatientSex": "F",
            "DICOMStudyInstanceUID": "1.2.840.99999999.84710745.943275268089",
            "DICOMPatientBirthDate": "19201120",
            "DICOMStudyDescription": "UNKNOWN",
            "DICOMPatientId": "SUBJECT08701",
            "DICOMPatientName": "Melissa844 Huel628",
            "DICOMNumberOfStudyRelatedInstances": 1,
            "DICOMStudyTime": "140728",
            "DICOMNumberOfStudyRelatedSeries": 1
        },
        "lastUpdatedAt": "2022-12-06T21:40:59.429000+00:00"
    }]
}
```

Contoh 4: Untuk mencari set gambar dengan operator EQUAL di DICOMSeries InstanceUID dan BETWEET pada UpdateDat dan mengurutkan respons dalam urutan ASC di bidang UpdateDAT

Contoh `search-image-sets` kode berikut mencari kumpulan gambar dengan operator EQUAL di DICOMSeries InstanceUID dan BETWEET pada UpdateDat dan mengurutkan respons dalam urutan ASC di bidang UpdateDAT.

Catatan: Berikan UpdateDat dalam format contoh (“1985-04-12T 23:20:50.52 Z”).

```
aws medical-imaging search-image-sets \
    --datastore-id 12345678901234567890123456789012 \
    --search-criteria  file://search-criteria.json
```

Isi dari `search-criteria.json`

```
{
    "filters": [{
        "values": [{
            "updatedAt": "2024-03-11T15:00:05.074000-07:00"
        }, {
            "updatedAt": "2024-03-11T16:00:05.074000-07:00"
        }],
        "operator": "BETWEEN"
    }, {
        "values": [{
            "DICOMSeriesInstanceUID": "1.2.840.99999999.84710745.943275268089"
        }],
        "operator": "EQUAL"
    }],
    "sort": {
        "sortField": "updatedAt",
        "sortOrder": "ASC"
    }
}
```

Output:

```
{
    "imageSetsMetadataSummaries": [{
        "imageSetId": "09876543210987654321098765432109",
        "createdAt": "2022-12-06T21:40:59.429000+00:00",
        "version": 1,
        "DICOMTags": {
            "DICOMStudyId": "2011201407",
            "DICOMStudyDate": "19991122",
            "DICOMPatientSex": "F",
            "DICOMStudyInstanceUID": "1.2.840.99999999.84710745.943275268089",
            "DICOMPatientBirthDate": "19201120",
            "DICOMStudyDescription": "UNKNOWN",
```

```
            "DICOMPatientId": "SUBJECT08701",
            "DICOMPatientName": "Melissa844 Huel628",
            "DICOMNumberOfStudyRelatedInstances": 1,
            "DICOMStudyTime": "140728",
            "DICOMNumberOfStudyRelatedSeries": 1
        },
        "lastUpdatedAt": "2022-12-06T21:40:59.429000+00:00"
    }]
}
```

Untuk informasi selengkapnya, lihat Mencari kumpulan gambar di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

Fungsi utilitas untuk mencari set gambar.

```
    public static List<ImageSetsMetadataSummary> searchMedicalImagingImageSets(
            MedicalImagingClient medicalImagingClient,
            String datastoreId, SearchCriteria searchCriteria) {
        try {
            SearchImageSetsRequest datastoreRequest =
SearchImageSetsRequest.builder()
                    .datastoreId(datastoreId)
                    .searchCriteria(searchCriteria)
                    .build();
            SearchImageSetsIterable responses = medicalImagingClient
                    .searchImageSetsPaginator(datastoreRequest);
            List<ImageSetsMetadataSummary> imageSetsMetadataSummaries = new
ArrayList<>();

            responses.stream().forEach(response -> imageSetsMetadataSummaries
                    .addAll(response.imageSetsMetadataSummaries()));

            return imageSetsMetadataSummaries;
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
        return null;
    }
```

Kasus penggunaan #1: operator EQUAL.

```
        List<SearchFilter> searchFilters =
 Collections.singletonList(SearchFilter.builder()
                .operator(Operator.EQUAL)
                .values(SearchByAttributeValue.builder()
                        .dicomPatientId(patientId)
                        .build())
                .build());

        SearchCriteria searchCriteria = SearchCriteria.builder()
                .filters(searchFilters)
                .build();

        List<ImageSetsMetadataSummary> imageSetsMetadataSummaries =
 searchMedicalImagingImageSets(
                medicalImagingClient,
                datastoreId, searchCriteria);
        if (imageSetsMetadataSummaries != null) {
            System.out.println("The image sets for patient " + patientId + " are:
\n"
                    + imageSetsMetadataSummaries);
            System.out.println();
        }
```

Kasus penggunaan #2: ANTARA operator menggunakan DICOMStudy Tanggal dan DICOMStudy Waktu.

```
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyyMMdd");
        searchFilters = Collections.singletonList(SearchFilter.builder()
                .operator(Operator.BETWEEN)
                .values(SearchByAttributeValue.builder()

 .dicomStudyDateAndTime(DICOMStudyDateAndTime.builder()
                                        .dicomStudyDate("19990101")
                                        .dicomStudyTime("000000.000")
                                        .build())
```

```
                                .build(),
                        SearchByAttributeValue.builder()

.dicomStudyDateAndTime(DICOMStudyDateAndTime.builder()
                                        .dicomStudyDate((LocalDate.now()
                                                .format(formatter)))
                                        .dicomStudyTime("000000.000")
                                        .build())
                                .build())
                .build());

        searchCriteria = SearchCriteria.builder()
                .filters(searchFilters)
                .build();

        imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
                datastoreId, searchCriteria);
        if (imageSetsMetadataSummaries != null) {
            System.out.println(
                    "The image sets searched with BETWEEN operator using
DICOMStudyDate and DICOMStudyTime are:\n"
                            +
                            imageSetsMetadataSummaries);
            System.out.println();
        }
```

Kasus penggunaan #3: ANTARA operator menggunakan createDat. Studi waktu sebelumnya bertahan.

```
        searchFilters = Collections.singletonList(SearchFilter.builder()
                .operator(Operator.BETWEEN)
                .values(SearchByAttributeValue.builder()

.createdAt(Instant.parse("1985-04-12T23:20:50.52Z"))
                                .build(),
                        SearchByAttributeValue.builder()
                                .createdAt(Instant.now())
                                .build())
                .build());

        searchCriteria = SearchCriteria.builder()
```

```
                .filters(searchFilters)
                .build();
        imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
                datastoreId, searchCriteria);
        if (imageSetsMetadataSummaries != null) {
            System.out.println("The image sets searched with BETWEEN operator
using createdAt are:\n "
                    + imageSetsMetadataSummaries);
            System.out.println();
        }
```

Kasus penggunaan #4: Operator EQUAL di DICOMSeries InstanceUID dan BETWEET di UpdateDAT dan mengurutkan respons dalam urutan ASC di bidang UpdateDAT.

```
        Instant startDate = Instant.parse("1985-04-12T23:20:50.52Z");
        Instant endDate = Instant.now();

        searchFilters = Arrays.asList(
                SearchFilter.builder()
                        .operator(Operator.EQUAL)
                        .values(SearchByAttributeValue.builder()
                                .dicomSeriesInstanceUID(seriesInstanceUID)
                                .build())
                        .build(),
                SearchFilter.builder()
                        .operator(Operator.BETWEEN)
                        .values(

SearchByAttributeValue.builder().updatedAt(startDate).build(),

SearchByAttributeValue.builder().updatedAt(endDate).build()
                        ).build());

        Sort sort =
Sort.builder().sortOrder(SortOrder.ASC).sortField(SortField.UPDATED_AT).build();

        searchCriteria = SearchCriteria.builder()
                .filters(searchFilters)
                .sort(sort)
                .build();
```

```
        imageSetsMetadataSummaries =
 searchMedicalImagingImageSets(medicalImagingClient,
                datastoreId, searchCriteria);
        if (imageSetsMetadataSummaries != null) {
            System.out.println("The image sets searched with EQUAL operator on
 DICOMSeriesInstanceUID and BETWEEN on updatedAt and sort response\n" +
                    "in ASC order on updatedAt field are:\n "
                    + imageSetsMetadataSummaries);
            System.out.println();
        }
```

- Untuk detail API, lihat SearchImageSetsdi Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

Fungsi utilitas untuk mencari set gambar.

```
import { paginateSearchImageSets } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The data store's ID.
 * @param { import('@aws-sdk/client-medical-imaging').SearchFilter[] } filters -
 The search criteria filters.
 * @param { import('@aws-sdk/client-medical-imaging').Sort } sort - The search
 criteria sort.
 */
export const searchImageSets = async (
  datastoreId = "xxxxxxxx",
  searchCriteria = {},
) => {
  const paginatorConfig = {
    client: medicalImagingClient,
    pageSize: 50,
```

```
  };

  const commandParams = {
    datastoreId: datastoreId,
    searchCriteria: searchCriteria,
  };

  const paginator = paginateSearchImageSets(paginatorConfig, commandParams);

  const imageSetsMetadataSummaries = [];
  for await (const page of paginator) {
    // Each page contains a list of `jobSummaries`. The list is truncated if is
 larger than `pageSize`.
    imageSetsMetadataSummaries.push(...page.imageSetsMetadataSummaries);
    console.log(page);
  }
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: 'f009ea9c-84ca-4749-b5b6-7164f00a5ada',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     imageSetsMetadataSummaries: [
  //         {
  //             DICOMTags: [Object],
  //             createdAt: "2023-09-19T16:59:40.551Z",
  //             imageSetId: '7f75e1b5c0f40eac2b24cf712f485f50',
  //             updatedAt: "2023-09-19T16:59:40.551Z",
  //             version: 1
  //         }]
  // }

  return imageSetsMetadataSummaries;
};
```

Kasus penggunaan #1: operator EQUAL.

```
  const datastoreId = "12345678901234567890123456789012";
```

```
try {
  const searchCriteria = {
    filters: [
      {
        values: [{ DICOMPatientId: "1234567" }],
        operator: "EQUAL",
      },
    ],
  };

  await searchImageSets(datastoreId, searchCriteria);
} catch (err) {
  console.error(err);
}
```

Kasus penggunaan #2: ANTARA operator menggunakan DICOMStudy Tanggal dan DICOMStudy Waktu.

```
const datastoreId = "12345678901234567890123456789012";

try {
  const searchCriteria = {
    filters: [
      {
        values: [
          {
            DICOMStudyDateAndTime: {
              DICOMStudyDate: "19900101",
              DICOMStudyTime: "000000",
            },
          },
          {
            DICOMStudyDateAndTime: {
              DICOMStudyDate: "20230901",
              DICOMStudyTime: "000000",
            },
          },
        ],
        operator: "BETWEEN",
      },
    ],
  };
```

```
  await searchImageSets(datastoreId, searchCriteria);
} catch (err) {
  console.error(err);
}
```

Kasus penggunaan #3: ANTARA operator menggunakan createDat. Studi waktu sebelumnya bertahan.

```
const datastoreId = "1234567890123456789012345678901 2";

try {
  const searchCriteria = {
    filters: [
      {
        values: [
          { createdAt: new Date("1985-04-12T23:20:50.52Z") },
          { createdAt: new Date() },
        ],
        operator: "BETWEEN",
      },
    ],
  };

  await searchImageSets(datastoreId, searchCriteria);
} catch (err) {
  console.error(err);
}
```

Kasus penggunaan #4: Operator EQUAL di DICOMSeries InstanceUID dan BETWEET di UpdateDAT dan mengurutkan respons dalam urutan ASC di bidang UpdateDAT.

```
const datastoreId = "1234567890123456789012345678901 2";

try {
  const searchCriteria = {
    filters: [
      {
        values: [
          { updatedAt: new Date("1985-04-12T23:20:50.52Z") },
          { updatedAt: new Date() },
```

```
          ],
          operator: "BETWEEN",
        },
        {
          values: [
            {
              DICOMSeriesInstanceUID:
                "1.1.123.123456.1.12.1.1234567890.1234.12345678.123",
            },
          ],
          operator: "EQUAL",
        },
      ],
      sort: {
        sortOrder: "ASC",
        sortField: "updatedAt",
      },
    };

    await searchImageSets(datastoreId, searchCriteria);
  } catch (err) {
    console.error(err);
  }
```

- Untuk detail API, lihat SearchImageSetsdi Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

Fungsi utilitas untuk mencari set gambar.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client
```

```
   def search_image_sets(self, datastore_id, search_filter):
       """
       Search for image sets.

       :param datastore_id: The ID of the data store.
       :param search_filter: The search filter.
           For example: {"filters" : [{ "operator": "EQUAL", "values":
[{"DICOMPatientId": "3524578"}]}]}.
       :return: The list of image sets.
       """
       try:
           paginator =
self.health_imaging_client.get_paginator("search_image_sets")
           page_iterator = paginator.paginate(
               datastoreId=datastore_id, searchCriteria=search_filter
           )
           metadata_summaries = []
           for page in page_iterator:
               metadata_summaries.extend(page["imageSetsMetadataSummaries"])
       except ClientError as err:
           logger.error(
               "Couldn't search image sets. Here's why: %s: %s",
               err.response["Error"]["Code"],
               err.response["Error"]["Message"],
           )
           raise
       else:
           return metadata_summaries
```

Kasus penggunaan #1: operator EQUAL.

```
       search_filter = {
           "filters": [
               {"operator": "EQUAL", "values": [{"DICOMPatientId": patient_id}]}
           ]
       }

       image_sets = self.search_image_sets(data_store_id, search_filter)
       print(f"Image sets found with EQUAL operator\n{image_sets}")
```

Kasus penggunaan #2: ANTARA operator menggunakan DICOMStudy Tanggal dan DICOMStudy Waktu.

```
        search_filter = {
            "filters": [
                {
                    "operator": "BETWEEN",
                    "values": [
                        {
                            "DICOMStudyDateAndTime": {
                                "DICOMStudyDate": "19900101",
                                "DICOMStudyTime": "000000",
                            }
                        },
                        {
                            "DICOMStudyDateAndTime": {
                                "DICOMStudyDate": "20230101",
                                "DICOMStudyTime": "000000",
                            }
                        },
                    ],
                }
            ]
        }

        image_sets = self.search_image_sets(data_store_id, search_filter)
        print(
            f"Image sets found with BETWEEN operator using DICOMStudyDate and
DICOMStudyTime\n{image_sets}"
        )
```

Kasus penggunaan #3: ANTARA operator menggunakan createDat. Studi waktu sebelumnya bertahan.

```
        search_filter = {
            "filters": [
                {
                    "values": [
                        {
                            "createdAt": datetime.datetime(
                                2021, 8, 4, 14, 49, 54, 429000
                            )
```

```
                        },
                        {
                            "createdAt": datetime.datetime.now()
                            + datetime.timedelta(days=1)
                        },
                    ],
                    "operator": "BETWEEN",
                }
            ]
        }

        recent_image_sets = self.search_image_sets(data_store_id, search_filter)
        print(
            f"Image sets found with with BETWEEN operator using createdAt
\n{recent_image_sets}"
        )
```

Kasus penggunaan #4: Operator EQUAL di DICOMSeries InstanceUID dan BETWEET di UpdateDAT dan mengurutkan respons dalam urutan ASC di bidang UpdateDAT.

```
        search_filter = {
            "filters": [
                {
                    "values": [
                        {
                            "updatedAt": datetime.datetime(
                                2021, 8, 4, 14, 49, 54, 429000
                            )
                        },
                        {
                            "updatedAt": datetime.datetime.now()
                            + datetime.timedelta(days=1)
                        },
                    ],
                    "operator": "BETWEEN",
                },
                {
                    "values": [{"DICOMSeriesInstanceUID": series_instance_uid}],
                    "operator": "EQUAL",
                },
            ],
            "sort": {
```

```
                "sortOrder": "ASC",
                "sortField": "updatedAt",
            },
        }

        image_sets = self.search_image_sets(data_store_id, search_filter)
        print(
            "Image sets found with EQUAL operator on DICOMSeriesInstanceUID and
BETWEEN on updatedAt and"
        )
        print(f"sort response in ASC order on updatedAt field\n{image_sets}")
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Untuk detail API, lihat SearchImageSetsdi 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

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678901234567890'
        oo_result = lo_mig->searchimagesets(
          iv_datastoreid = iv_datastore_id
          io_searchcriteria = io_search_criteria ).
        DATA(lt_imagesets) = oo_result->get_imagesetsmetadatasums( ).
        DATA(lv_count) = lines( lt_imagesets ).
        MESSAGE |Found { lv_count } image sets.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
```

```
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

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

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

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

Mendapatkan properti set gambar

Gunakan `GetImageSet` tindakan untuk mengembalikan properti untuk gambar tertentu yang disetel HealthImaging. Menu berikut memberikan prosedur untuk contoh Konsol Manajemen AWS dan kode untuk AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat `GetImageSet`di AWS HealthImaging API Referensi.

> **Note**
>
> Secara default, AWS HealthImaging mengembalikan properti untuk versi terbaru dari kumpulan gambar. Untuk melihat properti untuk versi yang lebih lama dari kumpulan gambar, berikan permintaan Anda. `versionId`

GunakanGetDICOMInstance, HealthImaging representasi DICOMweb layanan, untuk mengembalikan biner instance DICOM (.dcmfile). Untuk informasi selengkapnya, lihat Mendapatkan instance DICOM dari HealthImaging.

Untuk mendapatkan properti set gambar

Pilih menu berdasarkan preferensi akses Anda ke AWS HealthImaging.

AWS Konsol

1. Buka halaman penyimpanan data HealthImaging konsol.
2. Pilih penyimpanan data.

   Halaman detail penyimpanan data terbuka dan tab Image sets dipilih secara default.
3. Pilih satu set gambar.

   Halaman detail set Gambar membuka dan menampilkan properti set gambar.

AWS CLI dan SDKs

CLI

AWS CLI

Untuk mendapatkan properti set gambar

Contoh `get-image-set` kode berikut mendapatkan properti untuk set gambar.

```
aws medical-imaging get-image-set \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id 18f88ac7870584f58d56256646b4d92b \
    --version-id 1
```

Output:

```
{
    "versionId": "1",
    "imageSetWorkflowStatus": "COPIED",
    "updatedAt": 1680027253.471,
    "imageSetId": "18f88ac7870584f58d56256646b4d92b",
```

```
    "imageSetState": "ACTIVE",
    "createdAt": 1679592510.753,
    "datastoreId": "12345678901234567890123456789012"
}
```

Untuk informasi selengkapnya, lihat Mendapatkan properti set gambar di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

```
   public static GetImageSetResponse getMedicalImageSet(MedicalImagingClient
medicalImagingClient,
           String datastoreId,
           String imagesetId,
           String versionId) {
       try {
           GetImageSetRequest.Builder getImageSetRequestBuilder =
GetImageSetRequest.builder()
                   .datastoreId(datastoreId)
                   .imageSetId(imagesetId);

           if (versionId != null) {
               getImageSetRequestBuilder =
getImageSetRequestBuilder.versionId(versionId);
           }

           return
medicalImagingClient.getImageSet(getImageSetRequestBuilder.build());
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return null;
   }
```

- Untuk detail API, lihat GetImageSetdi Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

```
import { GetImageSetCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} imageSetId - The ID of the image set.
 * @param {string} imageSetVersion - The optional version of the image set.
 *
 */
export const getImageSet = async (
  datastoreId = "xxxxxxxxxxxxxxx",
  imageSetId = "xxxxxxxxxxxxxxx",
  imageSetVersion = "",
) => {
  const params = { datastoreId: datastoreId, imageSetId: imageSetId };
  if (imageSetVersion !== "") {
    params.imageSetVersion = imageSetVersion;
  }
  const response = await medicalImagingClient.send(
    new GetImageSetCommand(params),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //     httpStatusCode: 200,
  //         requestId: '0615c161-410d-4d06-9d8c-6e1241bb0a5a',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  // },
  //     createdAt: 2023-09-22T14:49:26.427Z,
```

```
  //     datastoreId: 'xxxxxxxxxxxxxx',
  //     imageSetArn: 'arn:aws:medical-imaging:us-east-1:xxxxxxxxxx:datastore/
xxxxxxxxxxxxxxxxxxx/imageset/xxxxxxxxxxxxxxxxxxx',
  //     imageSetId: 'xxxxxxxxxxxxxx',
  //     imageSetState: 'ACTIVE',
  //     imageSetWorkflowStatus: 'CREATED',
  //     updatedAt: 2023-09-22T14:49:26.427Z,
  //     versionId: '1'
  // }

  return response;
};
```

- Untuk detail API, lihat GetImageSetdi Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def get_image_set(self, datastore_id, image_set_id, version_id=None):
        """
        Get the properties of an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The optional version of the image set.
        :return: The image set properties.
        """
        try:
```

```
            if version_id:
                image_set = self.health_imaging_client.get_image_set(
                    imageSetId=image_set_id,
                    datastoreId=datastore_id,
                    versionId=version_id,
                )
            else:
                image_set = self.health_imaging_client.get_image_set(
                    imageSetId=image_set_id, datastoreId=datastore_id
                )
        except ClientError as err:
            logger.error(
                "Couldn't get image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return image_set
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
client = boto3.client("medical-imaging")
medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Untuk detail API, lihat GetImageSetdi 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

```
TRY.
    " iv_datastore_id = '12345678901234567890123456789012345678901234567890'
```

```
        " iv_image_set_id = '1234567890123456789012345678901234567890'
        " iv_version_id = '1' (optional)
        IF iv_version_id IS NOT INITIAL.
          oo_result = lo_mig->getimageset(
            iv_datastoreid = iv_datastore_id
            iv_imagesetid = iv_image_set_id
            iv_versionid = iv_version_id ).
        ELSE.
          oo_result = lo_mig->getimageset(
            iv_datastoreid = iv_datastore_id
            iv_imagesetid = iv_image_set_id ).
        ENDIF.
        DATA(lv_state) = oo_result->get_imagesetstate( ).
        MESSAGE |Image set retrieved with state: { lv_state }.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Image set not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

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

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

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

Mendapatkan metadata set gambar

Gunakan `GetImageSetMetadata` tindakan untuk mengambil metadata untuk gambar tertentu yang disetel. HealthImaging Menu berikut memberikan prosedur untuk contoh Konsol Manajemen AWS dan kode untuk AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat `GetImageSetMetadata`di AWS HealthImaging API Referensi.

> **Note**
>
> Secara default, HealthImaging mengembalikan atribut metadata untuk versi terbaru dari kumpulan gambar. Untuk melihat metadata untuk versi yang lebih lama dari kumpulan gambar, berikan permintaan `versionId` Anda.
> Metadata set gambar dikompresi dengan `gzip` dan dikembalikan sebagai objek JSON. Oleh karena itu, Anda harus mendekompresi objek JSON sebelum melihat metadata yang dinormalisasi. Untuk informasi selengkapnya, lihat Normalisasi metadata.
> Jika metadata set gambar besar masih diproses setelah impor, 409 `ConflictException` dapat dikembalikan. Coba lagi permintaan setelah beberapa detik setelah pemrosesan selesai.
> Gunakan`GetDICOMInstanceMetadata`, HealthImaging representasi DICOMweb layanan, untuk mengembalikan metadata instance DICOM (`.json`file). Untuk informasi selengkapnya, lihat Mendapatkan metadata instans DICOM dari HealthImaging.

Untuk mendapatkan metadata set gambar

Pilih menu berdasarkan preferensi akses Anda ke AWS HealthImaging.

AWS Konsol

1. Buka halaman penyimpanan data HealthImaging konsol.
2. Pilih penyimpanan data.

   Halaman detail penyimpanan data terbuka dan tab Image sets dipilih secara default.
3. Pilih satu set gambar.

   Halaman detail set Gambar terbuka dan metadata set gambar ditampilkan di bawah bagian Penampil metadata set gambar.

AWS CLI dan SDKs

C++

SDK untuk C++

Fungsi utilitas untuk mendapatkan metadata set gambar.

```
//! Routine which gets a HealthImaging image set's metadata.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param imageSetID: The HealthImaging image set ID.
  \param versionID: The HealthImaging image set version ID, ignored if empty.
  \param outputFilePath: The path where the metadata will be stored as gzipped
 json.
  \param clientConfig: Aws client configuration.
  \\return bool: Function succeeded.
*/
bool AwsDoc::Medical_Imaging::getImageSetMetadata(const Aws::String &dataStoreID,
                                                  const Aws::String &imageSetID,
                                                  const Aws::String &versionID,
                                                  const Aws::String
 &outputFilePath,
                                                  const
 Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::Model::GetImageSetMetadataRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);
    if (!versionID.empty()) {
        request.SetVersionId(versionID);
    }
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::GetImageSetMetadataOutcome outcome =
 client.GetImageSetMetadata(
            request);
    if (outcome.IsSuccess()) {
        std::ofstream file(outputFilePath, std::ios::binary);
        auto &metadata = outcome.GetResult().GetImageSetMetadataBlob();
        file << metadata.rdbuf();
    }
    else {
        std::cerr << "Failed to get image set metadata: "
                  << outcome.GetError().GetMessage() << std::endl;
    }
```

```
    return outcome.IsSuccess();
}
```

Dapatkan metadata set gambar tanpa versi.

```
       if (AwsDoc::Medical_Imaging::getImageSetMetadata(dataStoreID, imageSetID,
"", outputFilePath, clientConfig))
       {
           std::cout << "Successfully retrieved image set metadata." <<
std::endl;
           std::cout << "Metadata stored in: " << outputFilePath << std::endl;
       }
```

Dapatkan metadata set gambar dengan versi.

```
       if (AwsDoc::Medical_Imaging::getImageSetMetadata(dataStoreID, imageSetID,
versionID, outputFilePath, clientConfig))
       {
           std::cout << "Successfully retrieved image set metadata." <<
std::endl;
           std::cout << "Metadata stored in: " << outputFilePath << std::endl;
       }
```

- Untuk detail API, lihat GetImageSetMetadatadi Referensi AWS SDK untuk C++ API.

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

CLI

AWS CLI

Contoh 1: Untuk mendapatkan metadata set gambar tanpa versi

Contoh `get-image-set-metadata` kode berikut mendapatkan metadata untuk kumpulan gambar tanpa menentukan versi.

Catatan: `outfile` adalah parameter yang diperlukan

```
aws medical-imaging get-image-set-metadata \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id ea92b0d8838c72a3f25d00d13616f87e \
    studymetadata.json.gz
```

Metadata yang dikembalikan dikompresi dengan gzip dan disimpan dalam file studymetadata.json.gz. Untuk melihat isi objek JSON yang dikembalikan, Anda harus terlebih dahulu mendekompresnya.

Output:

```
{
    "contentType": "application/json",
    "contentEncoding": "gzip"
}
```

Contoh 2: Untuk mendapatkan metadata set gambar dengan versi

Contoh `get-image-set-metadata` kode berikut mendapatkan metadata untuk set gambar dengan versi tertentu.

Catatan: `outfile` adalah parameter yang diperlukan

```
aws medical-imaging get-image-set-metadata \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id ea92b0d8838c72a3f25d00d13616f87e \
    --version-id 1 \
    studymetadata.json.gz
```

Metadata yang dikembalikan dikompresi dengan gzip dan disimpan dalam file studymetadata.json.gz. Untuk melihat isi objek JSON yang dikembalikan, Anda harus terlebih dahulu mendekompresnya.

Output:

```
{
    "contentType": "application/json",
```

```
    "contentEncoding": "gzip"
}
```

Untuk informasi selengkapnya, lihat Mendapatkan metadata set gambar di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

```
   public static void getMedicalImageSetMetadata(MedicalImagingClient
medicalImagingClient,
            String destinationPath,
            String datastoreId,
            String imagesetId,
            String versionId) {

        try {
            GetImageSetMetadataRequest.Builder getImageSetMetadataRequestBuilder
= GetImageSetMetadataRequest.builder()
                    .datastoreId(datastoreId)
                    .imageSetId(imagesetId);

            if (versionId != null) {
                getImageSetMetadataRequestBuilder =
getImageSetMetadataRequestBuilder.versionId(versionId);
            }


medicalImagingClient.getImageSetMetadata(getImageSetMetadataRequestBuilder.build(),
                    FileSystems.getDefault().getPath(destinationPath));

            System.out.println("Metadata downloaded to " + destinationPath);
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
   }
```

- Untuk detail API, lihat GetImageSetMetadatadi Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

Fungsi utilitas untuk mendapatkan metadata set gambar.

```
import { GetImageSetMetadataCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";
import { writeFileSync } from "node:fs";

/**
 * @param {string} metadataFileName - The name of the file for the gzipped
 metadata.
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} imagesetId - The ID of the image set.
 * @param {string} versionID - The optional version ID of the image set.
 */
export const getImageSetMetadata = async (
  metadataFileName = "metadata.json.gzip",
  datastoreId = "xxxxxxxxxxxxx",
  imagesetId = "xxxxxxxxxxxxx",
  versionID = "",
) => {
  const params = { datastoreId: datastoreId, imageSetId: imagesetId };

  if (versionID) {
    params.versionID = versionID;
  }

  const response = await medicalImagingClient.send(
    new GetImageSetMetadataCommand(params),
  );
  const buffer = await response.imageSetMetadataBlob.transformToByteArray();
  writeFileSync(metadataFileName, buffer);

  console.log(response);
  // {
```

```
  //     '$metadata': {
  //     httpStatusCode: 200,
  //         requestId: '5219b274-30ff-4986-8cab-48753de3a599',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  // },
  //     contentType: 'application/json',
  //     contentEncoding: 'gzip',
  //     imageSetMetadataBlob: <ref *1> IncomingMessage {}
  // }

  return response;
};
```

Dapatkan metadata set gambar tanpa versi.

```
  try {
    await getImageSetMetadata(
      "metadata.json.gzip",
      "12345678901234567890123456789012",
      "12345678901234567890123456789012",
    );
  } catch (err) {
    console.log("Error", err);
  }
```

Dapatkan metadata set gambar dengan versi.

```
  try {
    await getImageSetMetadata(
      "metadata2.json.gzip",
      "12345678901234567890123456789012",
      "12345678901234567890123456789012",
      "1",
    );
  } catch (err) {
    console.log("Error", err);
  }
```

- Untuk detail API, lihat GetImageSetMetadatadi Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

Fungsi utilitas untuk mendapatkan metadata set gambar.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def get_image_set_metadata(
        self, metadata_file, datastore_id, image_set_id, version_id=None
    ):
        """
        Get the metadata of an image set.

        :param metadata_file: The file to store the JSON gzipped metadata.
        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The version of the image set.
        """
        try:
            if version_id:
                image_set_metadata =
 self.health_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id,
                    datastoreId=datastore_id,
                    versionId=version_id,
                )
            else:
```

```
                image_set_metadata =
self.health_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id, datastoreId=datastore_id
                )
            print(image_set_metadata)
            with open(metadata_file, "wb") as f:
                for chunk in
image_set_metadata["imageSetMetadataBlob"].iter_chunks():
                    if chunk:
                        f.write(chunk)

        except ClientError as err:
            logger.error(
                "Couldn't get image metadata. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Dapatkan metadata set gambar tanpa versi.

```
                image_set_metadata =
self.health_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id, datastoreId=datastore_id
                )
```

Dapatkan metadata set gambar dengan versi.

```
                image_set_metadata =
self.health_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id,
                    datastoreId=datastore_id,
                    versionId=version_id,
                )
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Untuk detail API, lihat GetImageSetMetadatadi 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

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678901234567890'
        " iv_image_set_id = '12345678901234567890123456789012345678901234567890'
        " iv_version_id = '1' (optional)
        IF iv_version_id IS NOT INITIAL.
          oo_result = lo_mig->getimagesetmetadata(
            iv_datastoreid = iv_datastore_id
            iv_imagesetid = iv_image_set_id
            iv_versionid = iv_version_id ).
        ELSE.
          oo_result = lo_mig->getimagesetmetadata(
            iv_datastoreid = iv_datastore_id
            iv_imagesetid = iv_image_set_id ).
        ENDIF.
        DATA(lv_metadata_blob) = oo_result->get_imagesetmetadatablob( ).
        MESSAGE 'Image set metadata retrieved.' TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Image set not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
```

```
    CATCH /aws1/cx_migvalidationex.
      MESSAGE 'Validation error.' TYPE 'I'.
ENDTRY.
```

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

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

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

Transfer Sintaks Metadata

Saat mengimpor data DICOM, HealthImaging simpan nilai asli untuk atribut sintaks transfer dalam metadata kumpulan gambar. Sintaks transfer data DICOM asli yang diimpor disimpan sebagai. `TransferSyntaxUID` HealthImaging digunakan `StoredTransferSyntaxUID` untuk menunjukkan format yang digunakan untuk menyandikan data bingkai gambar di penyimpanan data: `1.2.840.10008.1.2.4.202` untuk penyimpanan data berkemampuan HTJ2 K (default) dan `1.2.840.10008.1.2.4.90` untuk penyimpanan data yang diaktifkan JPEG 2000 Lossless.

Mendapatkan data piksel set gambar

Bingkai gambar adalah data piksel yang ada dalam gambar yang diatur untuk membuat gambar medis 2D. Gunakan `GetImageFrame` tindakan untuk mengambil bingkai gambar lossless JPEG 2000 HTJ2 yang disandikan K atau asli untuk gambar tertentu yang disetel. HealthImaging Menu berikut memberikan contoh kode untuk AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat `GetImageFrame`di AWS HealthImaging API Referensi.

> **Note**
>
> Ingatlah poin-poin berikut saat menggunakan `GetImageFrame` tindakan:

- Selama impor, HealthImaging mempertahankan pengkodean untuk beberapa sintaks transfer, dan mentranskode yang lain ke HTJ2 K lossless (default) atau JPEG 2000 Lossless. `GetImageFrame`Tindakan mengembalikan bingkai gambar dalam sintaks transfer tersimpan dari instance. Tidak ada transcoding yang dilakukan selama pengambilan untuk memastikan latensi pengambilan minimal. Bingkai gambar mungkin perlu diterjemahkan sebelum dilihat di penampil gambar, tergantung pada sintaks transfer. Untuk informasi selengkapnya, lihat Sintaks transfer yang didukung dan Pustaka decoding bingkai gambar.
- Untuk contoh yang disimpan HealthImaging dengan satu atau lebih bingkai gambar yang dikodekan dalam keluarga MPEG Transfer Syntax (yang mencakup MPEG2, MPEG-4 AVC/H.264 and HEVC/H .265) `GetImageFrame` tindakan akan mengembalikan objek video dalam sintaks transfer yang disimpan.
- Sintaks transfer frame gambar ditentukan dalam elemen respon `Content-Type` HTTP header. Misalnya, bingkai gambar yang dikodekan dalam HTJ2 K akan memiliki. `Content-Type: image/jph header` Untuk informasi selengkapnya, lihat `GetImageFrame`di AWS HealthImaging API Referensi.
- Anda juga dapat menggunakan representasi DICOMweb layanan`GetDICOMInstanceFrames`, HealthImaging untuk mengambil frame (`multipart`permintaan) instans DICOM untuk pemirsa dan aplikasi DICOMweb yang kompatibel. Untuk informasi selengkapnya, lihat Mendapatkan bingkai instans DICOM dari HealthImaging.

Untuk mendapatkan data piksel set gambar

Pilih menu berdasarkan preferensi akses Anda ke AWS HealthImaging.

AWS Konsol

> **Note**
>
> Bingkai gambar harus diakses dan diterjemahkan secara terprogram, karena penampil gambar tidak tersedia di file. Konsol Manajemen AWS
> Untuk informasi selengkapnya tentang decoding dan melihat bingkai gambar, lihat. Pustaka decoding bingkai gambar

AWS CLI dan SDKs

C++

SDK untuk C++

```
//! Routine which downloads an AWS HealthImaging image frame.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param imageSetID: The image set ID.
  \param frameID: The image frame ID.
  \param jphFile: File to store the downloaded frame.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::Medical_Imaging::getImageFrame(const Aws::String &dataStoreID,
                                            const Aws::String &imageSetID,
                                            const Aws::String &frameID,
                                            const Aws::String &jphFile,
                                            const
 Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);

    Aws::MedicalImaging::Model::GetImageFrameRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);

    Aws::MedicalImaging::Model::ImageFrameInformation imageFrameInformation;
    imageFrameInformation.SetImageFrameId(frameID);
    request.SetImageFrameInformation(imageFrameInformation);

    Aws::MedicalImaging::Model::GetImageFrameOutcome outcome =
 client.GetImageFrame(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully retrieved image frame." << std::endl;
        auto &buffer = outcome.GetResult().GetImageFrameBlob();

        std::ofstream outfile(jphFile, std::ios::binary);
        outfile << buffer.rdbuf();
    }
    else {
```

```
        std::cout << "Error retrieving image frame." <<
 outcome.GetError().GetMessage()
                  << std::endl;

    }

    return outcome.IsSuccess();
}
```

- Untuk detail API, lihat GetImageFramedi Referensi AWS SDK untuk C++ API.

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

CLI

AWS CLI

Untuk mendapatkan data piksel set gambar

Contoh `get-image-frame` kode berikut mendapat bingkai gambar.

```
aws medical-imaging get-image-frame \
    --datastore-id "12345678901234567890123456789012" \
    --image-set-id "98765412345612345678907890789012" \
    --image-frame-information imageFrameId=3abf5d5d7ae72f80a0ec81b2c0de3ef4 \
    imageframe.jph
```

Catatan: Contoh kode ini tidak menyertakan output karena GetImageFrame tindakan mengembalikan aliran data piksel ke file imageframe.jph. Untuk informasi tentang decoding dan melihat bingkai gambar, lihat HTJ2 K decoding library.

Untuk informasi selengkapnya, lihat Mendapatkan data piksel yang disetel gambar di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

```
       public static void getMedicalImageSetFrame(MedicalImagingClient
medicalImagingClient,
                        String destinationPath,
                        String datastoreId,
                        String imagesetId,
                        String imageFrameId) {

                try {
                        GetImageFrameRequest getImageSetMetadataRequest =
GetImageFrameRequest.builder()
                                                .datastoreId(datastoreId)
                                                .imageSetId(imagesetId)

.imageFrameInformation(ImageFrameInformation.builder()

.imageFrameId(imageFrameId)
                                                                .build())
                                                .build();

medicalImagingClient.getImageFrame(getImageSetMetadataRequest,

FileSystems.getDefault().getPath(destinationPath));

                        System.out.println("Image frame downloaded to " +
destinationPath);
                } catch (MedicalImagingException e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
       }
```

- Untuk detail API, lihat GetImageFramedi Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

```
import { GetImageFrameCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} imageFrameFileName - The name of the file for the HTJ2K-
encoded image frame.
 * @param {string} datastoreID - The data store's ID.
 * @param {string} imageSetID - The image set's ID.
 * @param {string} imageFrameID - The image frame's ID.
 */
export const getImageFrame = async (
  imageFrameFileName = "image.jph",
  datastoreID = "DATASTORE_ID",
  imageSetID = "IMAGE_SET_ID",
  imageFrameID = "IMAGE_FRAME_ID",
) => {
  const response = await medicalImagingClient.send(
    new GetImageFrameCommand({
      datastoreId: datastoreID,
      imageSetId: imageSetID,
      imageFrameInformation: { imageFrameId: imageFrameID },
    }),
  );
  const buffer = await response.imageFrameBlob.transformToByteArray();
  writeFileSync(imageFrameFileName, buffer);

  console.log(response);
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: 'e4ab42a5-25a3-4377-873f-374ecf4380e1',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     contentType: 'application/octet-stream',
  //     imageFrameBlob: <ref *1> IncomingMessage {}
  // }
  return response;
```

```
};
```

- Untuk detail API, lihat GetImageFramedi Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def get_pixel_data(
        self, file_path_to_write, datastore_id, image_set_id, image_frame_id
    ):
        """
        Get an image frame's pixel data.

        :param file_path_to_write: The path to write the image frame's HTJ2K
 encoded pixel data.
        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param image_frame_id: The ID of the image frame.
        """
        try:
            image_frame = self.health_imaging_client.get_image_frame(
                datastoreId=datastore_id,
                imageSetId=image_set_id,
                imageFrameInformation={"imageFrameId": image_frame_id},
            )
            with open(file_path_to_write, "wb") as f:
                for chunk in image_frame["imageFrameBlob"].iter_chunks():
                    if chunk:
                        f.write(chunk)
```

```
        except ClientError as err:
            logger.error(
                "Couldn't get image frame. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
client = boto3.client("medical-imaging")
medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Untuk detail API, lihat GetImageFramedi 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

```
TRY.
    " iv_datastore_id = '12345678901234567890123456789012345678901234567890'
    " iv_image_set_id = '12345678901234567890123456789012345678901234567890'
    " iv_image_frame_id = '12345678901234567890123456789012345678901234567890'
    oo_result = lo_mig->getimageframe(
      iv_datastoreid = iv_datastore_id
      iv_imagesetid = iv_image_set_id
      io_imageframeinformation = NEW /aws1/cl_migimageframeinfmtion(
        iv_imageframeid = iv_image_frame_id ) ).
    DATA(lv_frame_blob) = oo_result->get_imageframeblob( ).
    MESSAGE 'Image frame retrieved.' TYPE 'I'.
  CATCH /aws1/cx_migaccessdeniedex.
    MESSAGE 'Access denied.' TYPE 'I'.
  CATCH /aws1/cx_migconflictexception.
```

```
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Image frame not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

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

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

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

Memodifikasi set gambar dengan AWS HealthImaging

Pekerjaan impor DICOM biasanya mengharuskan Anda untuk memodifikasi set gambar Anda karena alasan berikut:

- Keamanan pasien
- Konsistensi data
- Mengurangi biaya penyimpanan

> **Penting:**
>
> Selama impor, HealthImaging memproses binari instance DICOM (`.dcm`file) dan mengubahnya menjadi kumpulan gambar. Gunakan tindakan bawaan HealthImaging cloud (APIs) untuk mengelola penyimpanan data dan kumpulan gambar. Gunakan HealthImaging representasi DICOMweb layanan untuk mengembalikan DICOMweb respons.

HealthImaging menyediakan beberapa cloud native APIs untuk menyederhanakan proses modifikasi set gambar. Topik berikut menjelaskan cara memodifikasi set gambar menggunakan AWS CLI dan AWS SDKs.

Topik

Daftar versi set gambar

Gunakan `ListImageSetVersions` tindakan untuk mencantumkan riwayat versi untuk gambar yang disetel HealthImaging. Menu berikut memberikan prosedur untuk contoh Konsol Manajemen AWS dan kode untuk AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat `ListImageSetVersions`di AWS HealthImaging API Referensi.

> **Note**
>
> AWS HealthImaging mencatat setiap perubahan yang dilakukan pada kumpulan gambar. Memperbarui metadata set gambar membuat versi baru dalam riwayat kumpulan gambar. Untuk informasi selengkapnya, lihat Memperbarui metadata set gambar.

Untuk daftar versi untuk set gambar

Pilih menu berdasarkan preferensi akses Anda ke AWS HealthImaging.

AWS Konsol

1. Buka halaman HealthImaging Console Data Stores.
2. Pilih penyimpanan data.

   Halaman detail penyimpanan data terbuka dan tab Image sets dipilih secara default.
3. Pilih satu set gambar.

   Halaman detail set Gambar terbuka.

   Versi set gambar ditampilkan di bawah bagian Detail set Gambar.

AWS CLI dan SDKs

CLI

AWS CLI

Untuk daftar versi set gambar

Contoh `list-image-set-versions` kode berikut mencantumkan riwayat versi untuk kumpulan gambar.

```
aws medical-imaging list-image-set-versions \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id ea92b0d8838c72a3f25d00d13616f87e
```

Output:

```
{
    "imageSetPropertiesList": [
        {
            "ImageSetWorkflowStatus": "UPDATED",
            "versionId": "4",
            "updatedAt": 1680029436.304,
            "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
            "imageSetState": "ACTIVE",
            "createdAt": 1680027126.436
        },
        {
            "ImageSetWorkflowStatus": "UPDATED",
            "versionId": "3",
            "updatedAt": 1680029163.325,
            "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
            "imageSetState": "ACTIVE",
            "createdAt": 1680027126.436
        },
        {
            "ImageSetWorkflowStatus": "COPY_FAILED",
            "versionId": "2",
            "updatedAt": 1680027455.944,
            "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
            "imageSetState": "ACTIVE",
            "message": "INVALID_REQUEST:  Series of SourceImageSet and
 DestinationImageSet don't match.",
            "createdAt": 1680027126.436
        },
        {
            "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
            "imageSetState": "ACTIVE",
            "versionId": "1",
            "ImageSetWorkflowStatus": "COPIED",
            "createdAt": 1680027126.436
        }
    ]
}
```


Untuk informasi selengkapnya, lihat Daftar versi kumpulan gambar di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

```
   public static List<ImageSetProperties>
listMedicalImageSetVersions(MedicalImagingClient medicalImagingClient,
           String datastoreId,
           String imagesetId) {
       try {
           ListImageSetVersionsRequest getImageSetRequest =
ListImageSetVersionsRequest.builder()
                   .datastoreId(datastoreId)
                   .imageSetId(imagesetId)
                   .build();

           ListImageSetVersionsIterable responses = medicalImagingClient
                   .listImageSetVersionsPaginator(getImageSetRequest);
           List<ImageSetProperties> imageSetProperties = new ArrayList<>();
           responses.stream().forEach(response ->
imageSetProperties.addAll(response.imageSetPropertiesList()));

           return imageSetProperties;
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return null;
   }
```

- Untuk detail API, lihat ListImageSetVersionsdi Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

```
import { paginateListImageSetVersions } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} imageSetId - The ID of the image set.
 */
export const listImageSetVersions = async (
  datastoreId = "xxxxxxxxxxxx",
  imageSetId = "xxxxxxxxxxxx",
) => {
  const paginatorConfig = {
    client: medicalImagingClient,
    pageSize: 50,
  };

  const commandParams = { datastoreId, imageSetId };
  const paginator = paginateListImageSetVersions(
    paginatorConfig,
    commandParams,
  );

  const imageSetPropertiesList = [];
  for await (const page of paginator) {
    // Each page contains a list of `jobSummaries`. The list is truncated if is
 larger than `pageSize`.
    imageSetPropertiesList.push(...page.imageSetPropertiesList);
    console.log(page);
  }
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: '74590b37-a002-4827-83f2-3c590279c742',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     imageSetPropertiesList: [
  //         {
```

```
  //             ImageSetWorkflowStatus: 'CREATED',
  //             createdAt: 2023-09-22T14:49:26.427Z,
  //             imageSetId: 'xxxxxxxxxxxxxxxxxxxxxx',
  //             imageSetState: 'ACTIVE',
  //             versionId: '1'
  //         }]
  // }
  return imageSetPropertiesList;
};
```

- Untuk detail API, lihat ListImageSetVersionsdi Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def list_image_set_versions(self, datastore_id, image_set_id):
        """
        List the image set versions.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :return: The list of image set versions.
        """
        try:
            paginator = self.health_imaging_client.get_paginator(
                "list_image_set_versions"
            )
            page_iterator = paginator.paginate(
                imageSetId=image_set_id, datastoreId=datastore_id
```

```
            )
            image_set_properties_list = []
            for page in page_iterator:
                image_set_properties_list.extend(page["imageSetPropertiesList"])
        except ClientError as err:
            logger.error(
                "Couldn't list image set versions. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return image_set_properties_list
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
client = boto3.client("medical-imaging")
medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Untuk detail API, lihat ListImageSetVersionsdi 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

```
TRY.
    " iv_datastore_id = '1234567890123456789012345678901234567890'
    " iv_image_set_id = '1234567890123456789012345678901234567890'
    oo_result = lo_mig->listimagesetversions(
      iv_datastoreid = iv_datastore_id
      iv_imagesetid = iv_image_set_id ).
    DATA(lt_versions) = oo_result->get_imagesetpropertieslist( ).
    DATA(lv_count) = lines( lt_versions ).
```

```
        MESSAGE |Found { lv_count } image set versions.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Image set not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

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

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

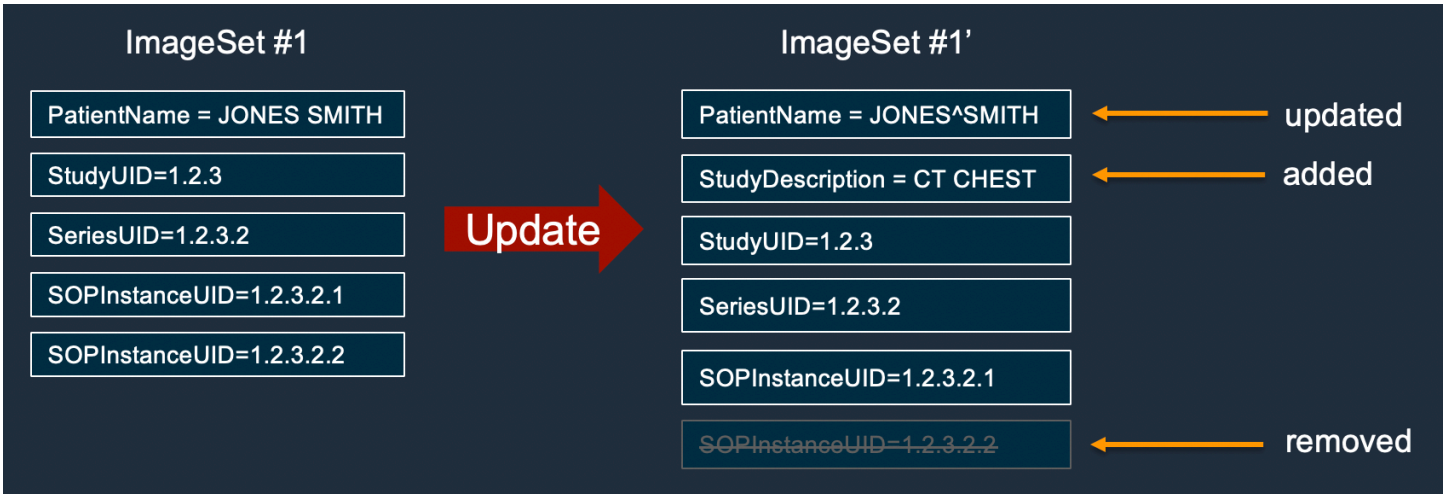
> **Ketersediaan contoh**
>
> Tidak dapat menemukan apa yang Anda butuhkan? Minta contoh kode menggunakan tautan Berikan umpan balik di bilah sisi kanan halaman ini.

Memperbarui metadata set gambar

Gunakan `UpdateImageSetMetadata` tindakan untuk memperbarui metadata set gambar di AWS. HealthImaging Anda dapat menggunakan proses asinkron ini untuk menambah, memperbarui, dan menghapus atribut metadata kumpulan gambar, yang merupakan manifestasi dari elemen normalisasi DICOM yang dibuat selama impor. Dengan menggunakan `UpdateImageSetMetadata` tindakan, Anda juga dapat menghapus Instans Seri dan SOP agar set gambar tetap sinkron dengan sistem eksternal dan untuk menghilangkan identifikasi metadata kumpulan gambar. Untuk informasi selengkapnya, lihat `UpdateImageSetMetadata`di AWS HealthImaging API Referensi.

> **Note**
>
> Impor DICOM dunia nyata memerlukan pembaruan, penambahan, dan penghapusan atribut dari metadata kumpulan gambar. Ingatlah poin-poin berikut saat memperbarui metadata set gambar:
>
> - Memperbarui metadata set gambar membuat versi baru dalam riwayat kumpulan gambar. Untuk informasi selengkapnya, lihat Daftar versi set gambar. Untuk kembali ke ID versi set gambar sebelumnya, gunakan `revertToVersionId`parameter opsional.
> - Memperbarui metadata kumpulan gambar adalah proses asinkron. Oleh karena itu, `imageSetState`dan elemen `imageSetWorkflowStatus`respons tersedia untuk memberikan status dan status masing-masing dari kumpulan gambar yang sedang diperbarui. Anda tidak dapat melakukan operasi penulisan lainnya pada kumpulan `LOCKED` gambar.
> - Jika `UpdateImageSetMetadata` tindakan tidak berhasil, panggil dan tinjau elemen `message`respons untuk melihatnya `common errors.`
> - Kendala elemen DICOM diterapkan pada pembaruan metadata. Parameter `force`permintaan memungkinkan Anda memperbarui elemen kumpulan gambar non-primer jika Anda ingin menggantiKendala metadata DICOM.
> - Elemen metadata tingkat Pasien dan Seri tidak dapat diperbarui untuk set gambar utama. Tidak UpdateImageSet akan mendukung -- `force` untuk memperbarui StudyInstance UID, UID, dan SeriesInstance SOPInstance UID untuk kumpulan gambar utama.
> - Tetapkan parameter `force`permintaan untuk memaksa penyelesaian `UpdateImageSetMetadata` tindakan pada kumpulan gambar non-primer. Menyetel parameter ini memungkinkan pembaruan berikut ke kumpulan gambar:
>   - Memperbarui`Tag.StudyInstanceUID`,`Tag.SeriesInstanceUID`,`Tag.SOPInstanceUID`, dan `Tag.StudyID` atribut
>   - Menambahkan, menghapus, atau memperbarui elemen data DICOM pribadi tingkat instans
> - Tindakan mempromosikan gambar yang disetel ke primer akan mengubah ID kumpulan gambar.

Diagram berikut merupakan metadata kumpulan gambar yang diperbarui di. HealthImaging



Untuk memperbarui metadata set gambar

Pilih tab berdasarkan preferensi akses Anda ke AWS HealthImaging.

AWS CLI dan SDKs

CLI

AWS CLI

Contoh 1: Untuk menyisipkan atau memperbarui atribut dalam metadata set gambar

`update-image-set-metadata`Contoh berikut menyisipkan atau memperbarui atribut dalam metadata set gambar.

```
aws medical-imaging update-image-set-metadata \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id ea92b0d8838c72a3f25d00d13616f87e \
    --latest-version-id 1 \
    --cli-binary-format raw-in-base64-out \
    --update-image-set-metadata-updates file://metadata-updates.json
```

Isi dari `metadata-updates.json`

```
{
    "DICOMUpdates": {
        "updatableAttributes": "{\"SchemaVersion\":1.1,\"Patient\":{\"DICOM\":
{\"PatientName\":\"MX^MX\"}}}"
    }
```

```
}
```

Output:

```
{
    "latestVersionId": "2",
    "imageSetWorkflowStatus": "UPDATING",
    "updatedAt": 1680042257.908,
    "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
    "imageSetState": "LOCKED",
    "createdAt": 1680027126.436,
    "datastoreId": "12345678901234567890123456789012"
}
```

Contoh 2: Untuk menghapus atribut dari metadata set gambar

`update-image-set-metadata`Contoh berikut menghapus atribut dari metadata set gambar.

```
aws medical-imaging update-image-set-metadata \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id ea92b0d8838c72a3f25d00d13616f87e \
    --latest-version-id 1 \
    --cli-binary-format raw-in-base64-out \
    --update-image-set-metadata-updates file://metadata-updates.json
```

Isi dari `metadata-updates.json`

```
{
    "DICOMUpdates": {
        "removableAttributes": "{\"SchemaVersion\":1.1,\"Study\":{\"DICOM\":
{\"StudyDescription\":\"CHEST\"}}}"
    }
}
```

Output:

```
{
    "latestVersionId": "2",
    "imageSetWorkflowStatus": "UPDATING",
    "updatedAt": 1680042257.908,
    "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
```

```
    "imageSetState": "LOCKED",
    "createdAt": 1680027126.436,
    "datastoreId": "12345678901234567890123456789012"
}
```

Contoh 3: Untuk menghapus instance dari metadata set gambar

`update-image-set-metadata`Contoh berikut menghapus instance dari metadata set gambar.

```
aws medical-imaging update-image-set-metadata \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id ea92b0d8838c72a3f25d00d13616f87e \
    --latest-version-id 1 \
    --cli-binary-format raw-in-base64-out \
    --update-image-set-metadata-updates file://metadata-updates.json \
    --force
```

Isi dari `metadata-updates.json`

```
{
    "DICOMUpdates": {
        "removableAttributes": "{\"SchemaVersion\": 1.1,\"Study\": {\"Series
\": {\"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1\": {\"Instances\":
 {\"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1\": {}}}}}}"
    }
}
```

Output:

```
{
    "latestVersionId": "2",
    "imageSetWorkflowStatus": "UPDATING",
    "updatedAt": 1680042257.908,
    "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
    "imageSetState": "LOCKED",
    "createdAt": 1680027126.436,
    "datastoreId": "12345678901234567890123456789012"
}
```

Contoh 4: Untuk mengembalikan gambar yang disetel ke versi sebelumnya

`update-image-set-metadata`Contoh berikut menunjukkan cara mengembalikan gambar yang disetel ke versi sebelumnya. CopyImageSet dan UpdateImageSetMetadata tindakan membuat versi baru dari set gambar.

```
aws medical-imaging update-image-set-metadata \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id 53d5fdb05ca4d46ac7ca64b06545c66e \
    --latest-version-id 3 \
    --cli-binary-format raw-in-base64-out \
    --update-image-set-metadata-updates '{"revertToVersionId": "1"}'
```

Output:

```
{
    "datastoreId": "12345678901234567890123456789012",
    "imageSetId": "53d5fdb05ca4d46ac7ca64b06545c66e",
    "latestVersionId": "4",
    "imageSetState": "LOCKED",
    "imageSetWorkflowStatus": "UPDATING",
    "createdAt": 1680027126.436,
    "updatedAt": 1680042257.908
}
```

Contoh 5: Untuk menambahkan elemen data DICOM pribadi ke sebuah instance

`update-image-set-metadata`Contoh berikut menunjukkan bagaimana menambahkan elemen pribadi untuk contoh tertentu dalam set gambar. Standar DICOM memungkinkan elemen data pribadi untuk komunikasi informasi yang tidak dapat terkandung dalam elemen data standar. Anda dapat membuat, memperbarui, dan menghapus elemen data pribadi dengan UpdateImageSetMetadata tindakan.

```
aws medical-imaging update-image-set-metadata \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id 53d5fdb05ca4d46ac7ca64b06545c66e \
    --latest-version-id 1 \
    --cli-binary-format raw-in-base64-out \
    --force \
    --update-image-set-metadata-updates file://metadata-updates.json
```

Isi dari `metadata-updates.json`

```
{
    "DICOMUpdates": {
        "updatableAttributes": "{\"SchemaVersion\": 1.1,\"Study\": {\"Series
\": {\"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1\": {\"Instances
\": {\"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1\": {\"DICOM\":
 {\"001910F9\": \"97\"},\"DICOMVRs\": {\"001910F9\": \"DS\"}}}}}}"
    }
}
```

Output:

```
{
    "latestVersionId": "2",
    "imageSetWorkflowStatus": "UPDATING",
    "updatedAt": 1680042257.908,
    "imageSetId": "53d5fdb05ca4d46ac7ca64b06545c66e",
    "imageSetState": "LOCKED",
    "createdAt": 1680027126.436,
    "datastoreId": "12345678901234567890123456789012"
}
```

Contoh 6: Untuk memperbarui elemen data DICOM pribadi ke sebuah instance

`update-image-set-metadata`Contoh berikut menunjukkan cara memperbarui nilai elemen data pribadi milik sebuah instance dalam kumpulan gambar.

```
aws medical-imaging update-image-set-metadata \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id 53d5fdb05ca4d46ac7ca64b06545c66e \
    --latest-version-id 1 \
    --cli-binary-format raw-in-base64-out \
    --force \
    --update-image-set-metadata-updates file://metadata-updates.json
```

Isi dari `metadata-updates.json`

```
{
    "DICOMUpdates": {
        "updatableAttributes": "{\"SchemaVersion\": 1.1,\"Study\": {\"Series
\": {\"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1\": {\"Instances
\": {\"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1\": {\"DICOM\":
 {\"00091001\": \"GE_GENESIS_DD\"}}}}}}"
```

```
    }
}
```

Output:

```
{
    "latestVersionId": "2",
    "imageSetWorkflowStatus": "UPDATING",
    "updatedAt": 1680042257.908,
    "imageSetId": "53d5fdb05ca4d46ac7ca64b06545c66e",
    "imageSetState": "LOCKED",
    "createdAt": 1680027126.436,
    "datastoreId": "12345678901234567890123456789012"
}
```

Contoh 7: Untuk memperbarui SOPInstance UID dengan parameter gaya

`update-image-set-metadata`Contoh berikut menunjukkan cara memperbarui SOPInstance UID, menggunakan parameter gaya untuk mengganti kendala metadata DICOM.

```
aws medical-imaging update-image-set-metadata \
        --datastore-id 12345678901234567890123456789012 \
        --image-set-id 53d5fdb05ca4d46ac7ca64b06545c66e \
        --latest-version-id 1 \
        --cli-binary-format raw-in-base64-out \
        --force \
        --update-image-set-metadata-updates file://metadata-updates.json
```

Isi dari `metadata-updates.json`

```
{
    "DICOMUpdates": {
        "updatableAttributes": "{\"SchemaVersion\":1.1,\"Study\":{\"Series
\":{\"1.3.6.1.4.1.5962.99.1.3633258862.2104868982.1369432891697.3656.0\":
{\"Instances\":
{\"1.3.6.1.4.1.5962.99.1.3633258862.2104868982.1369432891697.3659.0\":{\"DICOM\":
{\"SOPInstanceUID\":
\"1.3.6.1.4.1.5962.99.1.3633258862.2104868982.1369432891697.3659.9\"}}}}}}}"
    }
}
```

Output:

```
{
    "latestVersionId": "2",
    "imageSetWorkflowStatus": "UPDATING",
    "updatedAt": 1680042257.908,
    "imageSetId": "53d5fdb05ca4d46ac7ca64b06545c66e",
    "imageSetState": "LOCKED",
    "createdAt": 1680027126.436,
    "datastoreId": "12345678901234567890123456789012"
}
```

Untuk informasi selengkapnya, lihat Memperbarui metadata set gambar di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

```
   /**
    * Update the metadata of an AWS HealthImaging image set.
    *
    * @param medicalImagingClient - The AWS HealthImaging client object.
    * @param datastoreId          - The datastore ID.
    * @param imageSetId           - The image set ID.
    * @param versionId            - The version ID.
    * @param metadataUpdates      - A MetadataUpdates object containing the
updates.
    * @param force                - The force flag.
    * @throws MedicalImagingException - Base exception for all service
exceptions thrown by AWS HealthImaging.
    */
   public static void updateMedicalImageSetMetadata(MedicalImagingClient
medicalImagingClient,
                                                    String datastoreId,
                                                    String imageSetId,
                                                    String versionId,
                                                    MetadataUpdates
metadataUpdates,
                                                    boolean force) {
       try {
```

```
            UpdateImageSetMetadataRequest updateImageSetMetadataRequest =
UpdateImageSetMetadataRequest
                    .builder()
                    .datastoreId(datastoreId)
                    .imageSetId(imageSetId)
                    .latestVersionId(versionId)
                    .updateImageSetMetadataUpdates(metadataUpdates)
                    .force(force)
                    .build();

            UpdateImageSetMetadataResponse response =
medicalImagingClient.updateImageSetMetadata(updateImageSetMetadataRequest);

            System.out.println("The image set metadata was updated" + response);
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            throw e;
        }
    }
```

Kasus penggunaan #1: Menyisipkan atau memperbarui atribut.

```
                final String insertAttributes = """
                        {
                          "SchemaVersion": 1.1,
                          "Study": {
                            "DICOM": {
                              "StudyDescription": "CT CHEST"
                            }
                          }
                        }
                        """;
                MetadataUpdates metadataInsertUpdates = MetadataUpdates.builder()
                        .dicomUpdates(DICOMUpdates.builder()
                                .updatableAttributes(SdkBytes.fromByteBuffer(
                                        ByteBuffer.wrap(insertAttributes

.getBytes(StandardCharsets.UTF_8))))
                                .build())
                        .build();
```

```
                updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
imagesetId,
                        versionid, metadataInsertUpdates, force);
```

Use case #2: Hapus atribut.

```
                final String removeAttributes = """
                        {
                          "SchemaVersion": 1.1,
                          "Study": {
                            "DICOM": {
                              "StudyDescription": "CT CHEST"
                            }
                          }
                        }
                        """;
                MetadataUpdates metadataRemoveUpdates = MetadataUpdates.builder()
                        .dicomUpdates(DICOMUpdates.builder()
                                .removableAttributes(SdkBytes.fromByteBuffer(
                                        ByteBuffer.wrap(removeAttributes

.getBytes(StandardCharsets.UTF_8))))
                                .build())
                        .build();

                updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
imagesetId,
                        versionid, metadataRemoveUpdates, force);
```

Use case #3: Hapus sebuah instance.

```
                final String removeInstance = """
                        {
                          "SchemaVersion": 1.1,
                          "Study": {
                            "Series": {

"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {
                                "Instances": {

"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {}
```

```
                                }
                            }
                        }
                    }
                }
                """;
        MetadataUpdates metadataRemoveUpdates = MetadataUpdates.builder()
                .dicomUpdates(DICOMUpdates.builder()
                        .removableAttributes(SdkBytes.fromByteBuffer(
                                ByteBuffer.wrap(removeInstance

.getBytes(StandardCharsets.UTF_8))))
                        .build())
                .build();

        updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
imagesetId,
                versionid, metadataRemoveUpdates, force);
```

Kasus penggunaan #4: Kembalikan ke versi sebelumnya.

```
        // In this case, revert to previous version.
        String revertVersionId =
Integer.toString(Integer.parseInt(versionid) - 1);
        MetadataUpdates metadataRemoveUpdates = MetadataUpdates.builder()
                .revertToVersionId(revertVersionId)
                .build();
        updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
imagesetId,
                versionid, metadataRemoveUpdates, force);
```

- Untuk detail API, lihat UpdateImageSetMetadatadi Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

```
import { UpdateImageSetMetadataCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the HealthImaging data store.
 * @param {string} imageSetId - The ID of the HealthImaging image set.
 * @param {string} latestVersionId - The ID of the HealthImaging image set
 version.
 * @param {{}} updateMetadata - The metadata to update.
 * @param {boolean} force - Force the update.
 */
export const updateImageSetMetadata = async (
  datastoreId = "xxxxxxxxxx",
  imageSetId = "xxxxxxxxxx",
  latestVersionId = "1",
  updateMetadata = "{}",
  force = false,
) => {
  try {
    const response = await medicalImagingClient.send(
      new UpdateImageSetMetadataCommand({
        datastoreId: datastoreId,
        imageSetId: imageSetId,
        latestVersionId: latestVersionId,
        updateImageSetMetadataUpdates: updateMetadata,
        force: force,
      }),
    );
    console.log(response);
    // {
    //     '$metadata': {
    //     httpStatusCode: 200,
    //         requestId: '7966e869-e311-4bff-92ec-56a61d3003ea',
    //         extendedRequestId: undefined,
    //         cfId: undefined,
    //         attempts: 1,
    //         totalRetryDelay: 0
    // },
    //     createdAt: 2023-09-22T14:49:26.427Z,
    //     datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
```

```
    //     imageSetId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
    //     imageSetState: 'LOCKED',
    //     imageSetWorkflowStatus: 'UPDATING',
    //     latestVersionId: '4',
    //     updatedAt: 2023-09-27T19:41:43.494Z
    // }
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

Kasus penggunaan #1: Menyisipkan atau memperbarui atribut dan memaksa pembaruan.

```
    const insertAttributes = JSON.stringify({
      SchemaVersion: 1.1,
      Study: {
        DICOM: {
          StudyDescription: "CT CHEST",
        },
      },
    });

    const updateMetadata = {
      DICOMUpdates: {
        updatableAttributes: new TextEncoder().encode(insertAttributes),
      },
    };

    await updateImageSetMetadata(
      datastoreID,
      imageSetID,
      versionID,
      updateMetadata,
      true,
    );
```

Use case #2: Hapus atribut.

```
    // Attribute key and value must match the existing attribute.
    const remove_attribute = JSON.stringify({
```

```
  SchemaVersion: 1.1,
  Study: {
    DICOM: {
      StudyDescription: "CT CHEST",
    },
  },
});

const updateMetadata = {
  DICOMUpdates: {
    removableAttributes: new TextEncoder().encode(remove_attribute),
  },
};

await updateImageSetMetadata(
  datastoreID,
  imageSetID,
  versionID,
  updateMetadata,
);
```

Use case #3: Hapus sebuah instance.

```
const remove_instance = JSON.stringify({
  SchemaVersion: 1.1,
  Study: {
    Series: {
      "1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {
        Instances: {
          "1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {},
        },
      },
    },
  },
});

const updateMetadata = {
  DICOMUpdates: {
    removableAttributes: new TextEncoder().encode(remove_instance),
  },
};
```

```
    await updateImageSetMetadata(
      datastoreID,
      imageSetID,
      versionID,
      updateMetadata,
    );
```

Kasus penggunaan #4: Kembalikan ke versi sebelumnya.

```
    const updateMetadata = {
      revertToVersionId: "1",
    };

    await updateImageSetMetadata(
      datastoreID,
      imageSetID,
      versionID,
      updateMetadata,
    );
```

- Untuk detail API, lihat UpdateImageSetMetadatadi Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def update_image_set_metadata(
```

```
        self, datastore_id, image_set_id, version_id, metadata, force=False
    ):
        """
        Update the metadata of an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The ID of the image set version.
        :param metadata: The image set metadata as a dictionary.
            For example {"DICOMUpdates": {"updatableAttributes":
            "{\"SchemaVersion\":1.1,\"Patient\":{\"DICOM\":{\"PatientName\":
\"Garcia^Gloria\"}}}"}}
        :param: force: Force the update.
        :return: The updated image set metadata.
        """
        try:
            updated_metadata =
 self.health_imaging_client.update_image_set_metadata(
                imageSetId=image_set_id,
                datastoreId=datastore_id,
                latestVersionId=version_id,
                updateImageSetMetadataUpdates=metadata,
                force=force,
            )
        except ClientError as err:
            logger.error(
                "Couldn't update image set metadata. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return updated_metadata
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

Kasus penggunaan #1: Menyisipkan atau memperbarui atribut.

```
            attributes = """{
                    "SchemaVersion": 1.1,
                    "Study": {
                        "DICOM": {
                            "StudyDescription": "CT CHEST"
                        }
                    }
                }"""
            metadata = {"DICOMUpdates": {"updatableAttributes": attributes}}

            self.update_image_set_metadata(
                data_store_id, image_set_id, version_id, metadata, force
            )
```

Use case #2: Hapus atribut.

```
            # Attribute key and value must match the existing attribute.
            attributes = """{
                    "SchemaVersion": 1.1,
                    "Study": {
                        "DICOM": {
                            "StudyDescription": "CT CHEST"
                        }
                    }
                }"""
            metadata = {"DICOMUpdates": {"removableAttributes": attributes}}

            self.update_image_set_metadata(
                data_store_id, image_set_id, version_id, metadata, force
            )
```

Use case #3: Hapus sebuah instance.

```
            attributes = """{
                    "SchemaVersion": 1.1,
                    "Study": {
                        "Series": {

 "1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {
                                "Instances": {
```

```
"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {}
                                }
                            }
                        }
                    }
                }"""
            metadata = {"DICOMUpdates": {"removableAttributes": attributes}}

            self.update_image_set_metadata(
                data_store_id, image_set_id, version_id, metadata, force
            )
```

Kasus penggunaan #4: Kembalikan ke versi sebelumnya.

```
            metadata = {"revertToVersionId": "1"}

            self.update_image_set_metadata(
                data_store_id, image_set_id, version_id, metadata, force
            )
```

- Untuk detail API, lihat UpdateImageSetMetadatadi 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

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678901234567890'
        " iv_image_set_id = '12345678901234567890123456789012345678901234567890'
```

```
        " iv_latest_version_id = '1'
        " iv_force = abap_false
        oo_result = lo_mig->updateimagesetmetadata(
          iv_datastoreid = iv_datastore_id
          iv_imagesetid = iv_image_set_id
          iv_latestversionid = iv_latest_version_id
          io_updateimagesetmetupdates = io_metadata_updates
          iv_force = iv_force ).
        DATA(lv_new_version) = oo_result->get_latestversionid( ).
        MESSAGE |Image set metadata updated to version: { lv_new_version }.| TYPE
'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Image set not found.' TYPE 'I'.
      CATCH /aws1/cx_migservicequotaexcdex.
        MESSAGE 'Service quota exceeded.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

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

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

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

Anda dapat memindahkan Instans SOP antara kumpulan gambar, menyelesaikan konflik elemen metadata, dan menambah atau menghapus instance dari kumpulan gambar utama menggunakan,, dan. `CopyImageSet` `UpdateImageSetMetadata` `DeleteImageSet` APIs

Anda dapat menghapus kumpulan gambar dari koleksi utama dengan `DeleteImageSet` tindakan.

Untuk memperbarui metadata dari kumpulan gambar utama

1. Gunakan CopyImageSet tindakan untuk membuat kumpulan gambar non-primer yang merupakan salinan dari kumpulan gambar utama yang ingin Anda modifikasi. Katakanlah ini kembali `103785414bc2c89330f7ce51bbd13f7a` sebagai ID set gambar non-primer.

   ```
   aws medical-imaging copy-image-set --datastore-id
   a8d19e7875e1532d9b5652f6b25e12c9 --source-image-set-id
   0778b83b36eced0b76752bfe32192fb7 --copy-image-set-information
   '{"sourceImageSet": {"latestVersionId": "1" }}' --region us-west-2
   ```

2. Gunakan UpdateImageSetMetadata tindakan untuk membuat perubahan pada kumpulan (`103785414bc2c89330f7ce51bbd13f7a`) gambar non-primer. Misalnya, mengubah PatientID.

   ```
   aws medical-imaging update-image-set-metadata \
       --region us-west-2 \
       --datastore-id a8d19e7875e1532d9b5652f6b25e12c9 \
       --image-set-id 103785414bc2c89330f7ce51bbd13f7a \
       --latest-version-id 1 \
       --cli-binary-format raw-in-base64-out \
       --update-image-set-metadata-updates '{
       "DICOMUpdates": {
         "updatableAttributes": "{\"SchemaVersion\":1.1,\"Patient\":
         {\"DICOM\":{\"PatientID\":\"1234\"}}}"
       }
     }'
   ```

3. Hapus kumpulan gambar utama yang Anda modifikasi.

   ```
   aws medical-imaging delete-image-set --datastore-
             id a8d19e7875e1532d9b5652f6b25e12c9 --image-set-
             id 0778b83b36eced0b76752bfe32192fb7
   ```

4. Gunakan CopyImageSet tindakan dengan argumen `--promoteToPrimary` untuk menambahkan set gambar yang diperbarui ke koleksi utama.

```
aws medical-imaging copy-image-set --datastore-
          id a8d19e7875e1532d9b5652f6b25e12c9 --source-image-set-
          id 103785414bc2c89330f7ce51bbd13f7a --copy-image-set-information
          '{"sourceImageSet": {"latestVersionId": "2" }}' --region us-west-2 --
          promote-to-primary
```

5. Hapus kumpulan gambar non-primer.

```
aws medical-imaging delete-image-set --datastore-
          id a8d19e7875e1532d9b5652f6b25e12c9 --image-set-
          id 103785414bc2c89330f7ce51bbd13f7a
```

Untuk membuat set gambar non-primer primer

1. Gunakan UpdateImageSetMetadata tindakan untuk menyelesaikan konflik dengan set gambar Primer yang ada.

```
aws medical-imaging update-image-set-metadata \
    --region us-west-2 \
    --datastore-id a8d19e7875e1532d9b5652f6b25e12c9 \
    --image-set-id 103785414bc2c89330f7ce51bbd13f7a \
    --latest-version-id 1 \
    --cli-binary-format raw-in-base64-out \
    --update-image-set-metadata-updates '{
    "DICOMUpdates": {
      "updatableAttributes": "{\"SchemaVersion\":1.1,\"Patient\":{\"DICOM\":
      {\"PatientID\":\"1234\"}}}"
    }
  }'
```

2. Saat konflik diselesaikan, gunakan CopyImageSet tindakan dengan argumen `--promoteToPrimary` untuk menambahkan set gambar ke koleksi kumpulan gambar utama.

```
aws medical-imaging copy-image-set --datastore-
          id a8d19e7875e1532d9b5652f6b25e12c9 --source-image-set-
          id 103785414bc2c89330f7ce51bbd13f7a --copy-image-set-information
          '{"sourceImageSet": {"latestVersionId": "2" }}' --region us-west-2 --
```

```
            promote-to-primary
```

3. Setelah mengonfirmasi bahwa CopyImageSet tindakan berhasil, hapus kumpulan gambar non-primer sumber.

```
aws medical-imaging delete-image-set --datastore-
          id a8d19e7875e1532d9b5652f6b25e12c9 --image-set-
          id 103785414bc2c89330f7ce51bbd13f7a
```

Menyalin set gambar

Gunakan `CopyImageSet` tindakan untuk menyalin gambar yang disetel HealthImaging. Anda menggunakan proses asinkron ini untuk menyalin konten gambar yang disetel ke kumpulan gambar baru atau yang sudah ada. Anda dapat menyalin ke set gambar baru untuk membagi kumpulan gambar, serta untuk membuat salinan terpisah. Anda juga dapat menyalin ke set gambar yang ada untuk menggabungkan dua set gambar bersama-sama. Untuk informasi selengkapnya, lihat `CopyImageSet`di AWS HealthImaging API Referensi.

> **Note**
>
> Ingatlah poin-poin berikut saat menggunakan `CopyImageSet` tindakan:
>
> - `CopyImageSet`Tindakan akan membuat set gambar baru, atau versi baru dari file`destinationImageSet`. Untuk informasi selengkapnya, lihat Daftar versi set gambar.
> - Salin adalah proses asinkron. Oleh karena itu, elemen respons state (`imageSetState``imageSetWorkflowStatus`) dan status () tersedia untuk memberi tahu Anda operasi apa yang terjadi pada kumpulan gambar yang terkunci. Operasi penulisan lainnya tidak dapat dilakukan pada set gambar yang terkunci.
> - `CopyImageSet`membutuhkan SOP Instance UIDs menjadi unik dalam satu set gambar.
> - Anda dapat menyalin subset dari Instans SOP menggunakan. `copiableAttributes` Ini memungkinkan Anda untuk memilih satu atau lebih Instans SOP dari `sourceImageSet` untuk menyalin ke. `destinationImageSet`
> - Jika `CopyImageSet` tindakan tidak berhasil, hubungi `GetImageSet` dan tinjau `message`properti. Untuk informasi selengkapnya, lihat Mendapatkan properti set gambar.

- Impor DICOM dunia nyata dapat menghasilkan beberapa set gambar per Seri DICOM. `CopyImageSet`Tindakan membutuhkan `sourceImageSet` dan `destinationImageSet` memiliki metadata yang konsisten, kecuali jika `force`parameter opsional diberikan.
- Atur `force`parameter untuk memaksa operasi, bahkan jika ada elemen metadata yang tidak konsisten antara dan. `sourceImageSet` `destinationImageSet` Dalam kasus ini, metadata Pasien, Studi, dan Seri tetap tidak berubah di. `destinationImageSet`

Untuk menyalin set gambar

Pilih tab berdasarkan preferensi akses Anda ke AWS HealthImaging.

AWS CLI dan SDKs

CLI

AWS CLI

Contoh 1: Untuk menyalin set gambar tanpa tujuan.

`copy-image-set`Contoh berikut membuat salinan duplikat dari gambar yang ditetapkan tanpa tujuan.

```
aws medical-imaging copy-image-set \
    --datastore-id 12345678901234567890123456789012 \
    --source-image-set-id ea92b0d8838c72a3f25d00d13616f87e \
    --copy-image-set-information '{"sourceImageSet": {"latestVersionId": "1" } }'
```

Output:

```
{
    "destinationImageSetProperties": {
        "latestVersionId": "2",
        "imageSetWorkflowStatus": "COPYING",
        "updatedAt": 1680042357.432,
        "imageSetId": "b9a06fef182a5f992842f77f8e0868e5",
        "imageSetState": "LOCKED",
        "createdAt": 1680042357.432
    },
    "sourceImageSetProperties": {
```

```
        "latestVersionId": "1",
        "imageSetWorkflowStatus": "COPYING_WITH_READ_ONLY_ACCESS",
        "updatedAt": 1680042357.432,
        "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
        "imageSetState": "LOCKED",
        "createdAt": 1680027126.436
    },
    "datastoreId": "12345678901234567890123456789012"
}
```

Contoh 2: Untuk menyalin gambar yang ditetapkan dengan tujuan.

`copy-image-set`Contoh berikut membuat salinan duplikat dari gambar yang ditetapkan dengan tujuan.

```
aws medical-imaging copy-image-set \
    --datastore-id 12345678901234567890123456789012 \
    --source-image-set-id ea92b0d8838c72a3f25d00d13616f87e \
    --copy-image-set-information '{"sourceImageSet": {"latestVersionId": "1" },
 "destinationImageSet": { "imageSetId": "b9a06fef182a5f992842f77f8e0868e5",
 "latestVersionId": "1"} }'
```

Output:

```
{
    "destinationImageSetProperties": {
        "latestVersionId": "2",
        "imageSetWorkflowStatus": "COPYING",
        "updatedAt": 1680042505.135,
        "imageSetId": "b9a06fef182a5f992842f77f8e0868e5",
        "imageSetState": "LOCKED",
        "createdAt": 1680042357.432
    },
    "sourceImageSetProperties": {
        "latestVersionId": "1",
        "imageSetWorkflowStatus": "COPYING_WITH_READ_ONLY_ACCESS",
        "updatedAt": 1680042505.135,
        "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
        "imageSetState": "LOCKED",
        "createdAt": 1680027126.436
    },
    "datastoreId": "12345678901234567890123456789012"
```

```
}
```

Contoh 3: Untuk menyalin subset instance dari gambar sumber yang disetel ke kumpulan gambar tujuan.

`copy-image-set`Contoh berikut menyalin satu contoh DICOM dari gambar sumber yang disetel ke set gambar tujuan. Parameter gaya disediakan untuk mengesampingkan inkonsistensi dalam atribut tingkat Pasien, Studi, dan Seri.

```
aws medical-imaging copy-image-set \
    --datastore-id 12345678901234567890123456789012 \
    --source-image-set-id ea92b0d8838c72a3f25d00d13616f87e \
    --copy-image-set-information '{"sourceImageSet":
 {"latestVersionId": "1","DICOMCopies": {"copiableAttributes":
 "{\"SchemaVersion\":\"1.1\",\"Study\":{\"Series\":
{\"1.3.6.1.4.1.5962.99.1.3673257865.2104868982.1369432891697.3666.0\":
{\"Instances\":
{\"1.3.6.1.4.1.5962.99.1.3673257865.2104868982.1369432891697.3669.0\":
{}}}}}}"}},"destinationImageSet": {"imageSetId":
 "b9eb50d8ee682eb9fcf4acbf92f62bb7","latestVersionId": "1"}}' \
    --force
```

Output:

```
{
    "destinationImageSetProperties": {
        "latestVersionId": "2",
        "imageSetWorkflowStatus": "COPYING",
        "updatedAt": 1680042505.135,
        "imageSetId": "b9eb50d8ee682eb9fcf4acbf92f62bb7",
        "imageSetState": "LOCKED",
        "createdAt": 1680042357.432
    },
    "sourceImageSetProperties": {
        "latestVersionId": "1",
        "imageSetWorkflowStatus": "COPYING_WITH_READ_ONLY_ACCESS",
        "updatedAt": 1680042505.135,
        "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
        "imageSetState": "LOCKED",
        "createdAt": 1680027126.436
    },
    "datastoreId": "12345678901234567890123456789012"
```

```
}
```

Untuk informasi selengkapnya, lihat Menyalin set gambar di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

```
   /**
    * Copy an AWS HealthImaging image set.
    *
    * @param medicalImagingClient  - The AWS HealthImaging client object.
    * @param datastoreId           - The datastore ID.
    * @param imageSetId            - The image set ID.
    * @param latestVersionId       - The version ID.
    * @param destinationImageSetId - The optional destination image set ID,
ignored if null.
    * @param destinationVersionId  - The optional destination version ID,
ignored if null.
    * @param force                 - The force flag.
    * @param subsets               - The optional subsets to copy, ignored if
null.
    * @return                      - The image set ID of the copy.
    * @throws MedicalImagingException - Base exception for all service
exceptions thrown by AWS HealthImaging.
    */
   public static String copyMedicalImageSet(MedicalImagingClient
medicalImagingClient,
                                            String datastoreId,
                                            String imageSetId,
                                            String latestVersionId,
                                            String destinationImageSetId,
                                            String destinationVersionId,
                                            boolean force,
                                            Vector<String> subsets) {

       try {
           CopySourceImageSetInformation.Builder copySourceImageSetInformation =
CopySourceImageSetInformation.builder()
```

```
                    .latestVersionId(latestVersionId);

            // Optionally copy a subset of image instances.
            if (subsets != null) {
                String subsetInstanceToCopy =
getCopiableAttributesJSON(imageSetId, subsets);

copySourceImageSetInformation.dicomCopies(MetadataCopies.builder()
                        .copiableAttributes(subsetInstanceToCopy)
                        .build());
            }

            CopyImageSetInformation.Builder copyImageSetBuilder =
CopyImageSetInformation.builder()
                    .sourceImageSet(copySourceImageSetInformation.build());

            // Optionally designate a destination image set.
            if (destinationImageSetId != null) {
                copyImageSetBuilder =
copyImageSetBuilder.destinationImageSet(CopyDestinationImageSet.builder()
                        .imageSetId(destinationImageSetId)
                        .latestVersionId(destinationVersionId)
                        .build());
            }

            CopyImageSetRequest copyImageSetRequest =
CopyImageSetRequest.builder()
                    .datastoreId(datastoreId)
                    .sourceImageSetId(imageSetId)
                    .copyImageSetInformation(copyImageSetBuilder.build())
                    .force(force)
                    .build();

            CopyImageSetResponse response =
medicalImagingClient.copyImageSet(copyImageSetRequest);

            return response.destinationImageSetProperties().imageSetId();
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            throw e;
        }
    }
```

Fungsi utilitas untuk membuat atribut yang dapat disalin.

```
   /**
    * Create a JSON string of copiable image instances.
    *
    * @param imageSetId - The image set ID.
    * @param subsets    - The subsets to copy.
    * @return A JSON string of copiable image instances.
    */
   private static String getCopiableAttributesJSON(String imageSetId,
Vector<String> subsets) {
       StringBuilder subsetInstanceToCopy = new StringBuilder(
               """
                       {
                         "SchemaVersion": 1.1,
                         "Study": {
                           "Series": {
                               "
                                """
       );

       subsetInstanceToCopy.append(imageSetId);

       subsetInstanceToCopy.append(
               """
                               ": {
                               "Instances": {
                       """
       );

       for (String subset : subsets) {
           subsetInstanceToCopy.append('"' + subset + "\": {},");
       }
       subsetInstanceToCopy.deleteCharAt(subsetInstanceToCopy.length() - 1);
       subsetInstanceToCopy.append("""
                        }
                      }
                   }
                 }
               }
               """);
       return subsetInstanceToCopy.toString();
   }
```

- Untuk detail API, lihat CopyImageSetdi Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

Fungsi utilitas untuk menyalin set gambar.

```
import { CopyImageSetCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} imageSetId - The source image set ID.
 * @param {string} sourceVersionId - The source version ID.
 * @param {string} destinationImageSetId - The optional ID of the destination
 image set.
 * @param {string} destinationVersionId - The optional version ID of the
 destination image set.
 * @param {boolean} force - Force the copy action.
 * @param {[string]} copySubsets - A subset of instance IDs to copy.
 */
export const copyImageSet = async (
  datastoreId = "xxxxxxxxxx",
  imageSetId = "xxxxxxxxxxx",
  sourceVersionId = "1",
  destinationImageSetId = "",
  destinationVersionId = "",
  force = false,
  copySubsets = [],
) => {
  try {
    const params = {
      datastoreId: datastoreId,
      sourceImageSetId: imageSetId,
```

```
    copyImageSetInformation: {
      sourceImageSet: { latestVersionId: sourceVersionId },
    },
    force: force,
  };
  if (destinationImageSetId !== "" && destinationVersionId !== "") {
    params.copyImageSetInformation.destinationImageSet = {
      imageSetId: destinationImageSetId,
      latestVersionId: destinationVersionId,
    };
  }

  if (copySubsets.length > 0) {
    let copySubsetsJson;
    copySubsetsJson = {
      SchemaVersion: 1.1,
      Study: {
        Series: {
          imageSetId: {
            Instances: {},
          },
        },
      },
    };

    for (let i = 0; i < copySubsets.length; i++) {
      copySubsetsJson.Study.Series.imageSetId.Instances[copySubsets[i]] = {};
    }

    params.copyImageSetInformation.dicomCopies = copySubsetsJson;
  }

  const response = await medicalImagingClient.send(
    new CopyImageSetCommand(params),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: 'd9b219ce-cc48-4a44-a5b2-c5c3068f1ee8',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
```

```
    //      },
    //       datastoreId: 'xxxxxxxxxxxxx',
    //       destinationImageSetProperties: {
    //             createdAt: 2023-09-27T19:46:21.824Z,
    //             imageSetArn: 'arn:aws:medical-imaging:us-
east-1:xxxxxxxxxx:datastore/xxxxxxxxxxxx/imageset/xxxxxxxxxxxxxxxxx',
    //             imageSetId: 'xxxxxxxxxxxxxx',
    //             imageSetState: 'LOCKED',
    //             imageSetWorkflowStatus: 'COPYING',
    //             latestVersionId: '1',
    //             updatedAt: 2023-09-27T19:46:21.824Z
    //       },
    //       sourceImageSetProperties: {
    //             createdAt: 2023-09-22T14:49:26.427Z,
    //             imageSetArn: 'arn:aws:medical-imaging:us-
east-1:xxxxxxxxxx:datastore/xxxxxxxxxxxx/imageset/xxxxxxxxxxxxxxx',
    //             imageSetId: 'xxxxxxxxxxxxxxx',
    //             imageSetState: 'LOCKED',
    //             imageSetWorkflowStatus: 'COPYING_WITH_READ_ONLY_ACCESS',
    //             latestVersionId: '4',
    //             updatedAt: 2023-09-27T19:46:21.824Z
    //      }
    // }
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

Salin set gambar tanpa tujuan.

```
  await copyImageSet(
    "12345678901234567890123456789012",
    "12345678901234567890123456789012",
    "1",
  );
```

Salin set gambar dengan tujuan.

```
  await copyImageSet(
    "12345678901234567890123456789012",
    "12345678901234567890123456789012",
    "1",
    "12345678901234567890123456789012",
    "1",
    false,
  );
```

Salin subset dari kumpulan gambar dengan tujuan dan paksa salinannya.

```
  await copyImageSet(
    "12345678901234567890123456789012",
    "12345678901234567890123456789012",
    "1",
    "12345678901234567890123456789012",
    "1",
    true,
    ["12345678901234567890123456789012", "11223344556677889900112233445566"],
  );
```

- Untuk detail API, lihat CopyImageSetdi Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

Fungsi utilitas untuk menyalin set gambar.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
```

```
        self.health_imaging_client = health_imaging_client


    def copy_image_set(
        self,
        datastore_id,
        image_set_id,
        version_id,
        destination_image_set_id=None,
        destination_version_id=None,
        force=False,
        subsets=[],
    ):
        """
        Copy an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The ID of the image set version.
        :param destination_image_set_id: The ID of the optional destination image
set.
        :param destination_version_id: The ID of the optional destination image
set version.
        :param force: Force the copy.
        :param subsets: The optional subsets to copy. For example:
["12345678901234567890123456789012"].
        :return: The copied image set ID.
        """
        try:
            copy_image_set_information = {
                "sourceImageSet": {"latestVersionId": version_id}
            }
            if destination_image_set_id and destination_version_id:
                copy_image_set_information["destinationImageSet"] = {
                    "imageSetId": destination_image_set_id,
                    "latestVersionId": destination_version_id,
                }
            if len(subsets) > 0:
                copySubsetsJson = {
                    "SchemaVersion": "1.1",
                    "Study": {"Series": {"imageSetId": {"Instances": {}}}},
                }

                for subset in subsets:
```

```
                    copySubsetsJson["Study"]["Series"]["imageSetId"]["Instances"]
[
                        subset
                    ] = {}

                copy_image_set_information["sourceImageSet"]["DICOMCopies"] = {
                    "copiableAttributes": json.dumps(copySubsetsJson)
                }
            copy_results = self.health_imaging_client.copy_image_set(
                datastoreId=datastore_id,
                sourceImageSetId=image_set_id,
                copyImageSetInformation=copy_image_set_information,
                force=force,
            )
        except ClientError as err:
            logger.error(
                "Couldn't copy image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return copy_results["destinationImageSetProperties"]["imageSetId"]
```

Salin set gambar tanpa tujuan.

```
            copy_image_set_information = {
                "sourceImageSet": {"latestVersionId": version_id}
            }

            copy_results = self.health_imaging_client.copy_image_set(
                datastoreId=datastore_id,
                sourceImageSetId=image_set_id,
                copyImageSetInformation=copy_image_set_information,
                force=force,
            )
```

Salin set gambar dengan tujuan.

```
            copy_image_set_information = {
```

```
                "sourceImageSet": {"latestVersionId": version_id}
            }

            if destination_image_set_id and destination_version_id:
                copy_image_set_information["destinationImageSet"] = {
                    "imageSetId": destination_image_set_id,
                    "latestVersionId": destination_version_id,
                }

            copy_results = self.health_imaging_client.copy_image_set(
                datastoreId=datastore_id,
                sourceImageSetId=image_set_id,
                copyImageSetInformation=copy_image_set_information,
                force=force,
            )
```

Salin subset dari kumpulan gambar.

```
            copy_image_set_information = {
                "sourceImageSet": {"latestVersionId": version_id}
            }

            if len(subsets) > 0:
                copySubsetsJson = {
                    "SchemaVersion": "1.1",
                    "Study": {"Series": {"imageSetId": {"Instances": {}}}},
                }

                for subset in subsets:
                    copySubsetsJson["Study"]["Series"]["imageSetId"]["Instances"]
[
                        subset
                    ] = {}

                copy_image_set_information["sourceImageSet"]["DICOMCopies"] = {
                    "copiableAttributes": json.dumps(copySubsetsJson)
                }

            copy_results = self.health_imaging_client.copy_image_set(
                datastoreId=datastore_id,
                sourceImageSetId=image_set_id,
                copyImageSetInformation=copy_image_set_information,
```

```
                force=force,
            )
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Untuk detail API, lihat CopyImageSetdi 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

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678901234567890'
        " iv_source_image_set_id = '12345678901234567890123456789012345678901234567890'
        " iv_source_version_id = '1'
        " iv_destination_image_set_id =
'12345678901234567890123456789012345678901234567890' (optional)
        " iv_destination_version_id = '1' (optional)
        " iv_force = abap_false
        DATA(lo_source_info) = NEW /aws1/cl_migcpsrcimagesetinf00(
          iv_latestversionid = iv_source_version_id ).
        DATA(lo_copy_info) = NEW /aws1/cl_migcpimagesetinfmtion(
          io_sourceimageset = lo_source_info ).
        IF iv_destination_image_set_id IS NOT INITIAL AND
           iv_destination_version_id IS NOT INITIAL.
          DATA(lo_dest_info) = NEW /aws1/cl_migcopydstimageset(
            iv_imagesetid = iv_destination_image_set_id
            iv_latestversionid = iv_destination_version_id ).
          lo_copy_info = NEW /aws1/cl_migcpimagesetinfmtion(
            io_sourceimageset = lo_source_info
            io_destinationimageset = lo_dest_info ).
```

```
        ENDIF.
        oo_result = lo_mig->copyimageset(
          iv_datastoreid = iv_datastore_id
          iv_sourceimagesetid = iv_source_image_set_id
          io_copyimagesetinformation = lo_copy_info
          iv_force = iv_force ).
        DATA(lo_dest_props) = oo_result->get_dstimagesetproperties( ).
        DATA(lv_new_id) = lo_dest_props->get_imagesetid( ).
        MESSAGE |Image set copied with new ID: { lv_new_id }.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Image set not found.' TYPE 'I'.
      CATCH /aws1/cx_migservicequotaexcdex.
        MESSAGE 'Service quota exceeded.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

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

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

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

Menghapus set gambar

Gunakan `DeleteImageSet` tindakan untuk menghapus gambar yang disetel HealthImaging. Menu berikut memberikan prosedur untuk contoh Konsol Manajemen AWS dan kode untuk AWS CLI dan AWS SDKs. Untuk informasi selengkapnya, lihat `DeleteImageSet`di AWS HealthImaging API Referensi.

Untuk menghapus kumpulan gambar

Pilih menu berdasarkan preferensi akses Anda ke AWS HealthImaging.

AWS Konsol

1. Buka halaman HealthImaging Console Data Stores.
2. Pilih penyimpanan data.

   Halaman detail penyimpanan data terbuka dan tab Image sets dipilih secara default.
3. Pilih set gambar dan pilih Hapus.

   Modal set gambar Hapus terbuka.
4. Berikan ID set gambar dan pilih Hapus set gambar.

AWS CLI dan SDKs

C++

SDK untuk C++

```
//! Routine which deletes an AWS HealthImaging image set.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param imageSetID: The image set ID.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
  */
bool AwsDoc::Medical_Imaging::deleteImageSet(
        const Aws::String &dataStoreID, const Aws::String &imageSetID,
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::DeleteImageSetRequest request;
```

```
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);
    Aws::MedicalImaging::Model::DeleteImageSetOutcome outcome =
 client.DeleteImageSet(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted image set " << imageSetID
                  << " from data store " << dataStoreID << std::endl;
    }
    else {
        std::cerr << "Error deleting image set " << imageSetID << " from data
 store "
                  << dataStoreID << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Untuk detail API, lihat DeleteImageSetdi Referensi AWS SDK untuk C++ API.

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

CLI

AWS CLI

Untuk menghapus kumpulan gambar

Contoh `delete-image-set` kode berikut menghapus set gambar.

```
aws medical-imaging delete-image-set \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id ea92b0d8838c72a3f25d00d13616f87e
```

Output:

```
{
    "imageSetWorkflowStatus": "DELETING",
    "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
    "imageSetState": "LOCKED",
    "datastoreId": "12345678901234567890123456789012"
}
```

Untuk informasi selengkapnya, lihat Menghapus set gambar di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

```
   public static void deleteMedicalImageSet(MedicalImagingClient
medicalImagingClient,
           String datastoreId,
           String imagesetId) {
       try {
           DeleteImageSetRequest deleteImageSetRequest =
DeleteImageSetRequest.builder()
                   .datastoreId(datastoreId)
                   .imageSetId(imagesetId)
                   .build();

           medicalImagingClient.deleteImageSet(deleteImageSetRequest);

           System.out.println("The image set was deleted.");
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Untuk detail API, lihat DeleteImageSetdi Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

```
import { DeleteImageSetCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The data store ID.
 * @param {string} imageSetId - The image set ID.
 */
export const deleteImageSet = async (
  datastoreId = "xxxxxxxxxxxxxxx",
  imageSetId = "xxxxxxxxxxxxxxx",
) => {
  const response = await medicalImagingClient.send(
    new DeleteImageSetCommand({
      datastoreId: datastoreId,
      imageSetId: imageSetId,
    }),
  );
  console.log(response);
  // {
  //    '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: '6267bbd2-eaa5-4a50-8ee8-8fddf535cf73',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     datastoreId: 'xxxxxxxxxxxxxxx',
  //     imageSetId: 'xxxxxxxxxxxxxxx',
  //     imageSetState: 'LOCKED',
  //     imageSetWorkflowStatus: 'DELETING'
  // }
```

```
  return response;
};
```

- Untuk detail API, lihat DeleteImageSetdi Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def delete_image_set(self, datastore_id, image_set_id):
        """
        Delete an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :return: The delete results.
        """
        try:
            delete_results = self.health_imaging_client.delete_image_set(
                imageSetId=image_set_id, datastoreId=datastore_id
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return delete_results
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
client = boto3.client("medical-imaging")
medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Untuk detail API, lihat DeleteImageSetdi 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

```
TRY.
    " iv_datastore_id = '12345678901234567890123456789012345678901234567890'
    " iv_image_set_id = '12345678901234567890123456789012345678901234567890'
    oo_result = lo_mig->deleteimageset(
      iv_datastoreid = iv_datastore_id
      iv_imagesetid = iv_image_set_id ).
    MESSAGE 'Image set deleted.' TYPE 'I'.
  CATCH /aws1/cx_migaccessdeniedex.
    MESSAGE 'Access denied.' TYPE 'I'.
  CATCH /aws1/cx_migconflictexception.
    MESSAGE 'Conflict error.' TYPE 'I'.
  CATCH /aws1/cx_miginternalserverex.
    MESSAGE 'Internal server error.' TYPE 'I'.
  CATCH /aws1/cx_migresourcenotfoundex.
    MESSAGE 'Image set not found.' TYPE 'I'.
  CATCH /aws1/cx_migthrottlingex.
    MESSAGE 'Request throttled.' TYPE 'I'.
  CATCH /aws1/cx_migvalidationex.
    MESSAGE 'Validation error.' TYPE 'I'.
ENDTRY.
```

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

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

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

Menandai sumber daya dengan AWS HealthImaging

Anda dapat menetapkan metadata ke HealthImaging sumber daya (penyimpanan data dan kumpulan gambar) 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 sumber daya.

> **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 HealthImaging 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 sumber daya

Gunakan `TagResource`tindakan untuk menandai penyimpanan data dan set gambar di AWS HealthImaging. Contoh kode berikut menjelaskan cara menggunakan `TagResource` tindakan dengan Konsol Manajemen AWS, AWS CLI, dan AWS SDKs. Untuk informasi selengkapnya, lihat Menandai AWS sumber daya Anda di Referensi Umum AWS Panduan.

Untuk menandai sumber daya

Pilih menu berdasarkan preferensi akses Anda ke AWS HealthImaging.

AWS Konsol

1. Buka halaman HealthImaging Console Data Stores.

2. Pilih penyimpanan data.

   Halaman detail penyimpanan data terbuka.
3. Pilih tab Detail.
4. Di bawah bagian Tag, pilih Kelola tag.

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

AWS CLI dan SDKs

CLI

AWS CLI

Contoh 1: Untuk menandai penyimpanan data

Contoh `tag-resource` kode berikut menandai penyimpanan data.

```
aws medical-imaging tag-resource \
  --resource-arn "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012" \
  --tags '{"Deployment":"Development"}'
```

Perintah ini tidak menghasilkan output.

Contoh 2: Untuk menandai set gambar

Contoh `tag-resource` kode berikut menandai set gambar.

```
aws medical-imaging tag-resource \
    --resource-arn "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/18f88ac7870584f58d56256646b4d92b" \
    --tags '{"Deployment":"Development"}'
```

Perintah ini tidak menghasilkan output.

Untuk informasi selengkapnya, lihat Menandai sumber daya AWS HealthImaging di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

```
   public static void tagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
           String resourceArn,
           Map<String, String> tags) {
       try {
           TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                   .resourceArn(resourceArn)
                   .tags(tags)
                   .build();

           medicalImagingClient.tagResource(tagResourceRequest);

           System.out.println("Tags have been added to the resource.");
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Untuk detail API, lihat TagResourcedi Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

```
import { TagResourceCommand } from "@aws-sdk/client-medical-imaging";
```

```
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data
 store or image set.
 * @param {Record<string,string>} tags - The tags to add to the resource as JSON.
 *                     - For example: {"Deployment" : "Development"}
 */
export const tagResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:xxxxxx:datastore/xxxxx/
imageset/xxx",
  tags = {},
) => {
  const response = await medicalImagingClient.send(
    new TagResourceCommand({ resourceArn: resourceArn, tags: tags }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //        httpStatusCode: 204,
  //         requestId: '8a6de9a3-ec8e-47ef-8643-473518b19d45',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //    }
  // }

  return response;
};
```

- Untuk detail API, lihat TagResourcedi Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def tag_resource(self, resource_arn, tags):
        """
        Tag a resource.

        :param resource_arn: The ARN of the resource.
        :param tags: The tags to apply.
        """
        try:
            self.health_imaging_client.tag_resource(resourceArn=resource_arn,
 tags=tags)
        except ClientError as err:
            logger.error(
                "Couldn't tag resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- 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

```
    TRY.
        " iv_resource_arn = 'arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012'
        lo_mig->tagresource(
          iv_resourcearn = iv_resource_arn
          it_tags = it_tags ).
        MESSAGE 'Resource tagged successfully.' TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

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

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

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

Tag daftar untuk sumber daya

Gunakan `ListTagsForResource`tindakan untuk mencantumkan tag untuk penyimpanan data dan set gambar di AWS HealthImaging. Contoh kode berikut menjelaskan cara menggunakan `ListTagsForResource` tindakan dengan Konsol Manajemen AWS, AWS CLI, dan AWS SDKs. Untuk informasi selengkapnya, lihat Menandai AWS sumber daya Anda di Referensi Umum AWS Panduan.

Untuk daftar tag untuk sumber daya

Pilih menu berdasarkan preferensi akses Anda ke AWS HealthImaging.

AWS Konsol

1. Buka halaman penyimpanan data HealthImaging konsol.
2. Pilih penyimpanan data.

   Halaman detail penyimpanan data terbuka.
3. Pilih tab Detail.

   Di bawah bagian Tag, semua tag penyimpanan data terdaftar.

AWS CLI dan SDKs

CLI

AWS CLI

Contoh 1: Untuk daftar tag sumber daya untuk penyimpanan data

Contoh `list-tags-for-resource` kode berikut mencantumkan tag untuk penyimpanan data.

```
aws medical-imaging list-tags-for-resource \
    --resource-arn "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012"
```

Output:

```
{
```

```
    "tags":{
        "Deployment":"Development"
    }
}
```

Contoh 2: Untuk mencantumkan tag sumber daya untuk kumpulan gambar

Contoh `list-tags-for-resource` kode berikut mencantumkan tag untuk kumpulan gambar.

```
aws medical-imaging list-tags-for-resource \
    --resource-arn "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/18f88ac7870584f58d56256646b4d92b"
```

Output:

```
{
    "tags":{
        "Deployment":"Development"
    }
}
```

Untuk informasi selengkapnya, lihat Menandai sumber daya AWS HealthImaging di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

```
   public static ListTagsForResourceResponse
listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
           String resourceArn) {
       try {
           ListTagsForResourceRequest listTagsForResourceRequest =
ListTagsForResourceRequest.builder()
                   .resourceArn(resourceArn)
                   .build();
```

```
            return
 medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

- Untuk detail API, lihat ListTagsForResourcedi Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

```
import { ListTagsForResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data
 store or image set.
 */
export const listTagsForResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:abc:datastore/def/imageset/
ghi",
) => {
  const response = await medicalImagingClient.send(
    new ListTagsForResourceCommand({ resourceArn: resourceArn }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: '008fc6d3-abec-4870-a155-20fa3631e645',
```

```
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     tags: { Deployment: 'Development' }
  // }

  return response;
};
```

- Untuk detail API, lihat ListTagsForResourcedi Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def list_tags_for_resource(self, resource_arn):
        """
        List the tags for a resource.

        :param resource_arn: The ARN of the resource.
        :return: The list of tags.
        """
        try:
            tags = self.health_imaging_client.list_tags_for_resource(
                resourceArn=resource_arn
            )
        except ClientError as err:
            logger.error(
```

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

Kode berikut membuat instance objek. MedicalImagingWrapper

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- 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

```
    TRY.
        " iv_resource_arn = 'arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012'
        oo_result = lo_mig->listtagsforresource( iv_resourcearn =
 iv_resource_arn ).
        DATA(lt_tags) = oo_result->get_tags( ).
        DATA(lv_count) = lines( lt_tags ).
        MESSAGE |Found { lv_count } tags for resource.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
```

```
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

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

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

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

Membuka tag sumber daya

Gunakan `UntagResource`tindakan untuk menghapus tag penyimpanan data dan set gambar di AWS HealthImaging. Contoh kode berikut menjelaskan cara menggunakan `UntagResource` tindakan dengan Konsol Manajemen AWS, AWS CLI, dan AWS SDKs. Untuk informasi selengkapnya, lihat Menandai AWS sumber daya Anda di Referensi Umum AWS Panduan.

Untuk menghapus tag sumber daya

Pilih menu berdasarkan preferensi akses Anda ke AWS HealthImaging.

AWS Konsol

1. Buka halaman HealthImaging Console Data Stores.
2. Pilih penyimpanan data.

   Halaman detail penyimpanan data terbuka.
3. Pilih tab Detail.

4. Di bawah bagian Tag, pilih Kelola tag.

   Halaman Kelola tag terbuka.

5. Pilih Hapus di samping tag yang ingin Anda hapus.
6. Pilih Simpan perubahan.

AWS CLI dan SDKs

CLI

AWS CLI

Contoh 1: Untuk menghapus tag penyimpanan data

Contoh `untag-resource` kode berikut untags penyimpanan data.

```
aws medical-imaging untag-resource \
    --resource-arn "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012" \
    --tag-keys '["Deployment"]'
```

Perintah ini tidak menghasilkan output.

Contoh 2: Untuk menghapus tag set gambar

Contoh `untag-resource` kode berikut untag set gambar.

```
aws medical-imaging untag-resource \
    --resource-arn "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/18f88ac7870584f58d56256646b4d92b" \
    --tag-keys '["Deployment"]'
```

Perintah ini tidak menghasilkan output.

Untuk informasi selengkapnya, lihat Menandai sumber daya AWS HealthImaging di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

```
   public static void untagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
           String resourceArn,
           Collection<String> tagKeys) {
       try {
           UntagResourceRequest untagResourceRequest =
UntagResourceRequest.builder()
                   .resourceArn(resourceArn)
                   .tagKeys(tagKeys)
                   .build();

           medicalImagingClient.untagResource(untagResourceRequest);

           System.out.println("Tags have been removed from the resource.");
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Untuk detail API, lihat UntagResourcedi Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

```
import { UntagResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
```

```
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data
 store or image set.
 * @param {string[]} tagKeys - The keys of the tags to remove.
 */
export const untagResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:xxxxxx:datastore/xxxxx/
imageset/xxx",
  tagKeys = [],
) => {
  const response = await medicalImagingClient.send(
    new UntagResourceCommand({ resourceArn: resourceArn, tagKeys: tagKeys }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //        httpStatusCode: 204,
  //         requestId: '8a6de9a3-ec8e-47ef-8643-473518b19d45',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //    }
  // }

  return response;
};
```

- Untuk detail API, lihat UntagResourcedi Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
```

```
        self.health_imaging_client = health_imaging_client


    def untag_resource(self, resource_arn, tag_keys):
        """
        Untag a resource.

        :param resource_arn: The ARN of the resource.
        :param tag_keys: The tag keys to remove.
        """
        try:
            self.health_imaging_client.untag_resource(
                resourceArn=resource_arn, tagKeys=tag_keys
            )
        except ClientError as err:
            logger.error(
                "Couldn't untag resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- 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

```
    TRY.
        " iv_resource_arn = 'arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012'
        lo_mig->untagresource(
          iv_resourcearn = iv_resource_arn
          it_tagkeys = it_tag_keys ).
        MESSAGE 'Resource untagged successfully.' TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

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

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

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

Contoh kode untuk HealthImaging menggunakan AWS SDKs

Contoh kode berikut menunjukkan cara menggunakan HealthImaging 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.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

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

Contoh kode

- Contoh dasar untuk HealthImaging menggunakan AWS SDKs
  - Halo HealthImaging
  - Tindakan untuk HealthImaging menggunakan AWS SDKs
    - Gunakan CopyImageSet dengan AWS SDK atau CLI
    - Gunakan CreateDatastore dengan AWS SDK atau CLI
    - Gunakan DeleteDatastore dengan AWS SDK atau CLI
    - Gunakan DeleteImageSet dengan AWS SDK atau CLI
    - Gunakan GetDICOMImportJob dengan AWS SDK atau CLI
    - Gunakan GetDatastore dengan AWS SDK atau CLI
    - Gunakan GetImageFrame dengan AWS SDK atau CLI
    - Gunakan GetImageSet dengan AWS SDK atau CLI
    - Gunakan GetImageSetMetadata dengan AWS SDK atau CLI
    - Gunakan ListDICOMImportJobs dengan AWS SDK atau CLI
    - Gunakan ListDatastores dengan AWS SDK atau CLI
    - Gunakan ListImageSetVersions dengan AWS SDK atau CLI

        - Gunakan ListTagsForResource dengan AWS SDK atau CLI
        - Gunakan SearchImageSets dengan AWS SDK atau CLI
        - Gunakan StartDICOMImportJob dengan AWS SDK atau CLI
        - Gunakan TagResource dengan AWS SDK atau CLI
        - Gunakan UntagResource dengan AWS SDK atau CLI
        - Gunakan UpdateImageSetMetadata dengan AWS SDK atau CLI
- Skenario untuk HealthImaging menggunakan AWS SDKs
    - Memulai set HealthImaging gambar dan bingkai gambar menggunakan AWS SDK
    - Menandai penyimpanan HealthImaging data menggunakan SDK AWS
    - Menandai set HealthImaging gambar menggunakan SDK AWS

Contoh dasar untuk HealthImaging menggunakan AWS SDKs

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

Contoh

- Halo HealthImaging
- Tindakan untuk HealthImaging menggunakan AWS SDKs
    - Gunakan CopyImageSet dengan AWS SDK atau CLI
    - Gunakan CreateDatastore dengan AWS SDK atau CLI
    - Gunakan DeleteDatastore dengan AWS SDK atau CLI
    - Gunakan DeleteImageSet dengan AWS SDK atau CLI
    - Gunakan GetDICOMImportJob dengan AWS SDK atau CLI
    - Gunakan GetDatastore dengan AWS SDK atau CLI
    - Gunakan GetImageFrame dengan AWS SDK atau CLI
    - Gunakan GetImageSet dengan AWS SDK atau CLI
    - Gunakan GetImageSetMetadata dengan AWS SDK atau CLI
    - Gunakan ListDICOMImportJobs dengan AWS SDK atau CLI
    - Gunakan ListDatastores dengan AWS SDK atau CLI
    - Gunakan ListImageSetVersions dengan AWS SDK atau CLI

- Gunakan ListTagsForResource dengan AWS SDK atau CLI
- Gunakan SearchImageSets dengan AWS SDK atau CLI
- Gunakan StartDICOMImportJob dengan AWS SDK atau CLI
- Gunakan TagResource dengan AWS SDK atau CLI
- Gunakan UntagResource dengan AWS SDK atau CLI
- Gunakan UpdateImageSetMetadata dengan AWS SDK atau CLI

Halo HealthImaging

Contoh kode berikut menunjukkan cara untuk mulai menggunakan HealthImaging.

C++

SDK untuk C++

Kode untuk CMake file CMake Lists.txt.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS medical-imaging)

# Set this project's name.
project("hello_health-imaging")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()
```

```
# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
    # and set the proper subdirectory to the executable location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_health_imaging.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Kode untuk file sumber hello_health_imaging.cpp.

```
#include <aws/core/Aws.h>
#include <aws/medical-imaging/MedicalImagingClient.h>
#include <aws/medical-imaging/model/ListDatastoresRequest.h>

#include <iostream>

/*
 *  A "Hello HealthImaging" starter application which initializes an AWS
 HealthImaging (HealthImaging) client
 *  and lists the HealthImaging data stores in the current account.
 *
 *  main function
 *
 *  Usage: 'hello_health-imaging'
 *
 */
#include <aws/core/auth/AWSCredentialsProviderChain.h>
#include <aws/core/platform/Environment.h>
```

```
int main(int argc, char **argv) {
    (void) argc;
    (void) argv;
    Aws::SDKOptions options;
    //   Optional: change the log level for debugging.
    //   options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;

    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::MedicalImaging::MedicalImagingClient
 medicalImagingClient(clientConfig);
        Aws::MedicalImaging::Model::ListDatastoresRequest listDatastoresRequest;

        Aws::Vector<Aws::MedicalImaging::Model::DatastoreSummary>
 allDataStoreSummaries;
        Aws::String nextToken; // Used for paginated results.
        do {
            if (!nextToken.empty()) {
                listDatastoresRequest.SetNextToken(nextToken);
            }
            Aws::MedicalImaging::Model::ListDatastoresOutcome
 listDatastoresOutcome =
                    medicalImagingClient.ListDatastores(listDatastoresRequest);
            if (listDatastoresOutcome.IsSuccess()) {
                const Aws::Vector<Aws::MedicalImaging::Model::DatastoreSummary>
 &dataStoreSummaries =

 listDatastoresOutcome.GetResult().GetDatastoreSummaries();
                allDataStoreSummaries.insert(allDataStoreSummaries.cend(),
                                             dataStoreSummaries.cbegin(),
                                             dataStoreSummaries.cend());
                nextToken = listDatastoresOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "ListDatastores error: "
                          << listDatastoresOutcome.GetError().GetMessage() <<
 std::endl;
                break;
            }
        } while (!nextToken.empty());
```

```
        std::cout << allDataStoreSummaries.size() << " HealthImaging data "
                  << ((allDataStoreSummaries.size() == 1) ?
                      "store was retrieved." : "stores were retrieved.") <<
 std::endl;

        for (auto const &dataStoreSummary: allDataStoreSummaries) {
            std::cout << "  Datastore: " << dataStoreSummary.GetDatastoreName()
                      << std::endl;
            std::cout << "  Datastore ID: " << dataStoreSummary.GetDatastoreId()
                      << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Untuk detail API, lihat ListDatastoresdi Referensi AWS SDK untuk C++ API.

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

JavaScript

SDK untuk JavaScript (v3)

```
import {
  ListDatastoresCommand,
  MedicalImagingClient,
} from "@aws-sdk/client-medical-imaging";

// When no region or credentials are provided, the SDK will use the
// region and credentials from the local AWS config.
const client = new MedicalImagingClient({});

export const helloMedicalImaging = async () => {
  const command = new ListDatastoresCommand({});
```

```
  const { datastoreSummaries } = await client.send(command);
  console.log("Datastores: ");
  console.log(datastoreSummaries.map((item) => item.datastoreName).join("\n"));
  return datastoreSummaries;
};
```

- Untuk detail API, lihat ListDatastoresdi Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def hello_medical_imaging(medical_imaging_client):
    """
    Use the AWS SDK for Python (Boto3) to create an AWS HealthImaging
    client and list the data stores in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param medical_imaging_client: A Boto3 AWS HealthImaging Client object.
    """
    print("Hello, Amazon Health Imaging! Let's list some of your data stores:\n")
    try:
        paginator = medical_imaging_client.get_paginator("list_datastores")
        page_iterator = paginator.paginate()
        datastore_summaries = []
        for page in page_iterator:
```

```
            datastore_summaries.extend(page["datastoreSummaries"])
        print("\tData Stores:")
        for ds in datastore_summaries:
            print(f"\t\tDatastore: {ds['datastoreName']} ID {ds['datastoreId']}")
    except ClientError as err:
        logger.error(
            "Couldn't list data stores. Here's why: %s: %s",
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise


if __name__ == "__main__":
    hello_medical_imaging(boto3.client("medical-imaging"))
```

- Untuk detail API, lihat ListDatastoresdi 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.

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

Tindakan untuk HealthImaging menggunakan AWS SDKs

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

Kutipan ini memanggil HealthImaging API dan merupakan kutipan kode dari program yang lebih besar yang harus dijalankan dalam konteks. Anda dapat melihat tindakan dalam konteks diSkenario untuk HealthImaging menggunakan AWS SDKs.

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

Contoh

- Gunakan CopyImageSet dengan AWS SDK atau CLI
- Gunakan CreateDatastore dengan AWS SDK atau CLI
- Gunakan DeleteDatastore dengan AWS SDK atau CLI
- Gunakan DeleteImageSet dengan AWS SDK atau CLI
- Gunakan GetDICOMImportJob dengan AWS SDK atau CLI
- Gunakan GetDatastore dengan AWS SDK atau CLI
- Gunakan GetImageFrame dengan AWS SDK atau CLI
- Gunakan GetImageSet dengan AWS SDK atau CLI
- Gunakan GetImageSetMetadata dengan AWS SDK atau CLI
- Gunakan ListDICOMImportJobs dengan AWS SDK atau CLI
- Gunakan ListDatastores dengan AWS SDK atau CLI
- Gunakan ListImageSetVersions dengan AWS SDK atau CLI
- Gunakan ListTagsForResource dengan AWS SDK atau CLI
- Gunakan SearchImageSets dengan AWS SDK atau CLI
- Gunakan StartDICOMImportJob dengan AWS SDK atau CLI
- Gunakan TagResource dengan AWS SDK atau CLI
- Gunakan UntagResource dengan AWS SDK atau CLI
- Gunakan UpdateImageSetMetadata dengan AWS SDK atau CLI

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

Contoh kode berikut menunjukkan cara menggunakan`CopyImageSet`.

CLI

AWS CLI

Contoh 1: Untuk menyalin set gambar tanpa tujuan.

`copy-image-set`Contoh berikut membuat salinan duplikat dari gambar yang ditetapkan tanpa tujuan.

```
aws medical-imaging copy-image-set \
```

```
    --datastore-id 12345678901234567890123456789012 \
    --source-image-set-id ea92b0d8838c72a3f25d00d13616f87e \
    --copy-image-set-information '{"sourceImageSet": {"latestVersionId": "1" } }'
```

Output:

```
{
    "destinationImageSetProperties": {
        "latestVersionId": "2",
        "imageSetWorkflowStatus": "COPYING",
        "updatedAt": 1680042357.432,
        "imageSetId": "b9a06fef182a5f992842f77f8e0868e5",
        "imageSetState": "LOCKED",
        "createdAt": 1680042357.432
    },
    "sourceImageSetProperties": {
        "latestVersionId": "1",
        "imageSetWorkflowStatus": "COPYING_WITH_READ_ONLY_ACCESS",
        "updatedAt": 1680042357.432,
        "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
        "imageSetState": "LOCKED",
        "createdAt": 1680027126.436
    },
    "datastoreId": "12345678901234567890123456789012"
}
```

Contoh 2: Untuk menyalin gambar yang ditetapkan dengan tujuan.

`copy-image-set`Contoh berikut membuat salinan duplikat dari gambar yang ditetapkan dengan tujuan.

```
aws medical-imaging copy-image-set \
    --datastore-id 12345678901234567890123456789012 \
    --source-image-set-id ea92b0d8838c72a3f25d00d13616f87e \
    --copy-image-set-information '{"sourceImageSet": {"latestVersionId": "1" },
 "destinationImageSet": { "imageSetId": "b9a06fef182a5f992842f77f8e0868e5",
 "latestVersionId": "1"} }'
```

Output:

```
{
    "destinationImageSetProperties": {
```

```
        "latestVersionId": "2",
        "imageSetWorkflowStatus": "COPYING",
        "updatedAt": 1680042505.135,
        "imageSetId": "b9a06fef182a5f992842f77f8e0868e5",
        "imageSetState": "LOCKED",
        "createdAt": 1680042357.432
    },
    "sourceImageSetProperties": {
        "latestVersionId": "1",
        "imageSetWorkflowStatus": "COPYING_WITH_READ_ONLY_ACCESS",
        "updatedAt": 1680042505.135,
        "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
        "imageSetState": "LOCKED",
        "createdAt": 1680027126.436
    },
    "datastoreId": "12345678901234567890123456789012"
}
```

Contoh 3: Untuk menyalin subset instance dari gambar sumber yang disetel ke kumpulan gambar tujuan.

`copy-image-set`Contoh berikut menyalin satu contoh DICOM dari gambar sumber yang disetel ke set gambar tujuan. Parameter gaya disediakan untuk mengesampingkan inkonsistensi dalam atribut tingkat Pasien, Studi, dan Seri.

```
aws medical-imaging copy-image-set \
    --datastore-id 12345678901234567890123456789012 \
    --source-image-set-id ea92b0d8838c72a3f25d00d13616f87e \
    --copy-image-set-information '{"sourceImageSet":
 {"latestVersionId": "1","DICOMCopies": {"copiableAttributes":
 "{\"SchemaVersion\":\"1.1\",\"Study\":{\"Series\":
{\"1.3.6.1.4.1.5962.99.1.3673257865.2104868982.1369432891697.3666.0\":
{\"Instances\":
{\"1.3.6.1.4.1.5962.99.1.3673257865.2104868982.1369432891697.3669.0\":
{}}}}}}"}},"destinationImageSet": {"imageSetId":
 "b9eb50d8ee682eb9fcf4acbf92f62bb7","latestVersionId": "1"}}' \
    --force
```

Output:

```
{
    "destinationImageSetProperties": {
```

```
        "latestVersionId": "2",
        "imageSetWorkflowStatus": "COPYING",
        "updatedAt": 1680042505.135,
        "imageSetId": "b9eb50d8ee682eb9fcf4acbf92f62bb7",
        "imageSetState": "LOCKED",
        "createdAt": 1680042357.432
    },
    "sourceImageSetProperties": {
        "latestVersionId": "1",
        "imageSetWorkflowStatus": "COPYING_WITH_READ_ONLY_ACCESS",
        "updatedAt": 1680042505.135,
        "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
        "imageSetState": "LOCKED",
        "createdAt": 1680027126.436
    },
    "datastoreId": "12345678901234567890123456789012"
}
```

Untuk informasi selengkapnya, lihat Menyalin set gambar di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

```
    /**
     * Copy an AWS HealthImaging image set.
     *
     * @param medicalImagingClient  - The AWS HealthImaging client object.
     * @param datastoreId           - The datastore ID.
     * @param imageSetId            - The image set ID.
     * @param latestVersionId       - The version ID.
     * @param destinationImageSetId - The optional destination image set ID,
ignored if null.
     * @param destinationVersionId  - The optional destination version ID,
ignored if null.
     * @param force                 - The force flag.
     * @param subsets               - The optional subsets to copy, ignored if
null.
     * @return                      - The image set ID of the copy.
```

```
    * @throws MedicalImagingException - Base exception for all service
exceptions thrown by AWS HealthImaging.
    */
   public static String copyMedicalImageSet(MedicalImagingClient
medicalImagingClient,
                                            String datastoreId,
                                            String imageSetId,
                                            String latestVersionId,
                                            String destinationImageSetId,
                                            String destinationVersionId,
                                            boolean force,
                                            Vector<String> subsets) {

       try {
           CopySourceImageSetInformation.Builder copySourceImageSetInformation =
CopySourceImageSetInformation.builder()
                   .latestVersionId(latestVersionId);

           // Optionally copy a subset of image instances.
           if (subsets != null) {
               String subsetInstanceToCopy =
getCopiableAttributesJSON(imageSetId, subsets);

copySourceImageSetInformation.dicomCopies(MetadataCopies.builder()
                       .copiableAttributes(subsetInstanceToCopy)
                       .build());
           }

           CopyImageSetInformation.Builder copyImageSetBuilder =
CopyImageSetInformation.builder()
                   .sourceImageSet(copySourceImageSetInformation.build());

           // Optionally designate a destination image set.
           if (destinationImageSetId != null) {
               copyImageSetBuilder =
copyImageSetBuilder.destinationImageSet(CopyDestinationImageSet.builder()
                       .imageSetId(destinationImageSetId)
                       .latestVersionId(destinationVersionId)
                       .build());
           }

           CopyImageSetRequest copyImageSetRequest =
CopyImageSetRequest.builder()
                   .datastoreId(datastoreId)
```

```
                    .sourceImageSetId(imageSetId)
                    .copyImageSetInformation(copyImageSetBuilder.build())
                    .force(force)
                    .build();

            CopyImageSetResponse response =
medicalImagingClient.copyImageSet(copyImageSetRequest);

            return response.destinationImageSetProperties().imageSetId();
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            throw e;
        }
    }
```

Fungsi utilitas untuk membuat atribut yang dapat disalin.

```
    /**
     * Create a JSON string of copiable image instances.
     *
     * @param imageSetId - The image set ID.
     * @param subsets    - The subsets to copy.
     * @return A JSON string of copiable image instances.
     */
    private static String getCopiableAttributesJSON(String imageSetId,
Vector<String> subsets) {
        StringBuilder subsetInstanceToCopy = new StringBuilder(
                """
                        {
                          "SchemaVersion": 1.1,
                          "Study": {
                            "Series": {
                                "
                                 """
        );

        subsetInstanceToCopy.append(imageSetId);

        subsetInstanceToCopy.append(
                """
                                ": {
```

```
                                "Instances": {
                        """
        );

        for (String subset : subsets) {
            subsetInstanceToCopy.append('"' + subset + "\": {},");
        }
        subsetInstanceToCopy.deleteCharAt(subsetInstanceToCopy.length() - 1);
        subsetInstanceToCopy.append("""
                        }
                    }
                }
              }
            }
            """);
        return subsetInstanceToCopy.toString();
    }
```

- Untuk detail API, lihat CopyImageSetdi Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

Fungsi utilitas untuk menyalin set gambar.

```
import { CopyImageSetCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} imageSetId - The source image set ID.
 * @param {string} sourceVersionId - The source version ID.
 * @param {string} destinationImageSetId - The optional ID of the destination
 image set.
```

```
 * @param {string} destinationVersionId - The optional version ID of the
 destination image set.
 * @param {boolean} force - Force the copy action.
 * @param {[string]} copySubsets - A subset of instance IDs to copy.
 */
export const copyImageSet = async (
  datastoreId = "xxxxxxxxxx",
  imageSetId = "xxxxxxxxxxx",
  sourceVersionId = "1",
  destinationImageSetId = "",
  destinationVersionId = "",
  force = false,
  copySubsets = [],
) => {
  try {
    const params = {
      datastoreId: datastoreId,
      sourceImageSetId: imageSetId,
      copyImageSetInformation: {
        sourceImageSet: { latestVersionId: sourceVersionId },
      },
      force: force,
    };
    if (destinationImageSetId !== "" && destinationVersionId !== "") {
      params.copyImageSetInformation.destinationImageSet = {
        imageSetId: destinationImageSetId,
        latestVersionId: destinationVersionId,
      };
    }

    if (copySubsets.length > 0) {
      let copySubsetsJson;
      copySubsetsJson = {
        SchemaVersion: 1.1,
        Study: {
          Series: {
            imageSetId: {
              Instances: {},
            },
          },
        },
      };

      for (let i = 0; i < copySubsets.length; i++) {
```

```
        copySubsetsJson.Study.Series.imageSetId.Instances[copySubsets[i]] = {};
      }

      params.copyImageSetInformation.dicomCopies = copySubsetsJson;
    }

    const response = await medicalImagingClient.send(
      new CopyImageSetCommand(params),
    );
    console.log(response);
    // {
    //     '$metadata': {
    //         httpStatusCode: 200,
    //         requestId: 'd9b219ce-cc48-4a44-a5b2-c5c3068f1ee8',
    //         extendedRequestId: undefined,
    //         cfId: undefined,
    //         attempts: 1,
    //         totalRetryDelay: 0
    //      },
    //       datastoreId: 'xxxxxxxxxxxxx',
    //       destinationImageSetProperties: {
    //             createdAt: 2023-09-27T19:46:21.824Z,
    //             imageSetArn: 'arn:aws:medical-imaging:us-
east-1:xxxxxxxxxx:datastore/xxxxxxxxxxxx/imageset/xxxxxxxxxxxxxxxxx',
    //             imageSetId: 'xxxxxxxxxxxxx',
    //             imageSetState: 'LOCKED',
    //             imageSetWorkflowStatus: 'COPYING',
    //             latestVersionId: '1',
    //             updatedAt: 2023-09-27T19:46:21.824Z
    //       },
    //       sourceImageSetProperties: {
    //             createdAt: 2023-09-22T14:49:26.427Z,
    //             imageSetArn: 'arn:aws:medical-imaging:us-
east-1:xxxxxxxxxx:datastore/xxxxxxxxxxxx/imageset/xxxxxxxxxxxxxxx',
    //             imageSetId: 'xxxxxxxxxxxxxxx',
    //             imageSetState: 'LOCKED',
    //             imageSetWorkflowStatus: 'COPYING_WITH_READ_ONLY_ACCESS',
    //             latestVersionId: '4',
    //             updatedAt: 2023-09-27T19:46:21.824Z
    //      }
    // }
    return response;
  } catch (err) {
    console.error(err);
```

```
  }
};
```

Salin set gambar tanpa tujuan.

```
await copyImageSet(
  "12345678901234567890123456789012",
  "12345678901234567890123456789012",
  "1",
);
```

Salin set gambar dengan tujuan.

```
await copyImageSet(
  "12345678901234567890123456789012",
  "12345678901234567890123456789012",
  "1",
  "12345678901234567890123456789012",
  "1",
  false,
);
```

Salin subset dari kumpulan gambar dengan tujuan dan paksa salinannya.

```
await copyImageSet(
  "12345678901234567890123456789012",
  "12345678901234567890123456789012",
  "1",
  "12345678901234567890123456789012",
  "1",
  true,
  ["12345678901234567890123456789012", "11223344556677889900112233445566"],
);
```

- Untuk detail API, lihat CopyImageSetdi Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

Fungsi utilitas untuk menyalin set gambar.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def copy_image_set(
        self,
        datastore_id,
        image_set_id,
        version_id,
        destination_image_set_id=None,
        destination_version_id=None,
        force=False,
        subsets=[],
    ):
        """
        Copy an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The ID of the image set version.
        :param destination_image_set_id: The ID of the optional destination image
 set.
        :param destination_version_id: The ID of the optional destination image
 set version.
        :param force: Force the copy.
        :param subsets: The optional subsets to copy. For example:
["12345678901234567890123456789012"].
        :return: The copied image set ID.
```

```
        """
        try:
            copy_image_set_information = {
                "sourceImageSet": {"latestVersionId": version_id}
            }
            if destination_image_set_id and destination_version_id:
                copy_image_set_information["destinationImageSet"] = {
                    "imageSetId": destination_image_set_id,
                    "latestVersionId": destination_version_id,
                }
            if len(subsets) > 0:
                copySubsetsJson = {
                    "SchemaVersion": "1.1",
                    "Study": {"Series": {"imageSetId": {"Instances": {}}}},
                }

                for subset in subsets:
                    copySubsetsJson["Study"]["Series"]["imageSetId"]["Instances"]
[
                        subset
                    ] = {}

                copy_image_set_information["sourceImageSet"]["DICOMCopies"] = {
                    "copiableAttributes": json.dumps(copySubsetsJson)
                }
            copy_results = self.health_imaging_client.copy_image_set(
                datastoreId=datastore_id,
                sourceImageSetId=image_set_id,
                copyImageSetInformation=copy_image_set_information,
                force=force,
            )
        except ClientError as err:
            logger.error(
                "Couldn't copy image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return copy_results["destinationImageSetProperties"]["imageSetId"]
```

Salin set gambar tanpa tujuan.

```
            copy_image_set_information = {
                "sourceImageSet": {"latestVersionId": version_id}
            }

            copy_results = self.health_imaging_client.copy_image_set(
                datastoreId=datastore_id,
                sourceImageSetId=image_set_id,
                copyImageSetInformation=copy_image_set_information,
                force=force,
            )
```

Salin set gambar dengan tujuan.

```
            copy_image_set_information = {
                "sourceImageSet": {"latestVersionId": version_id}
            }

            if destination_image_set_id and destination_version_id:
                copy_image_set_information["destinationImageSet"] = {
                    "imageSetId": destination_image_set_id,
                    "latestVersionId": destination_version_id,
                }

            copy_results = self.health_imaging_client.copy_image_set(
                datastoreId=datastore_id,
                sourceImageSetId=image_set_id,
                copyImageSetInformation=copy_image_set_information,
                force=force,
            )
```

Salin subset dari kumpulan gambar.

```
            copy_image_set_information = {
                "sourceImageSet": {"latestVersionId": version_id}
            }

            if len(subsets) > 0:
                copySubsetsJson = {
                    "SchemaVersion": "1.1",
```

```
                    "Study": {"Series": {"imageSetId": {"Instances": {}}}},
                }

                for subset in subsets:
                    copySubsetsJson["Study"]["Series"]["imageSetId"]["Instances"]
[
                        subset
                    ] = {}

                copy_image_set_information["sourceImageSet"]["DICOMCopies"] = {
                    "copiableAttributes": json.dumps(copySubsetsJson)
                }

            copy_results = self.health_imaging_client.copy_image_set(
                datastoreId=datastore_id,
                sourceImageSetId=image_set_id,
                copyImageSetInformation=copy_image_set_information,
                force=force,
            )
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Untuk detail API, lihat CopyImageSetdi 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

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678901234567890'
        " iv_source_image_set_id = '12345678901234567890123456789012345678901234567890'
```

```
        " iv_source_version_id = '1'
        " iv_destination_image_set_id =
'12345678901234567890123456789012345678901234567890' (optional)
        " iv_destination_version_id = '1' (optional)
        " iv_force = abap_false
        DATA(lo_source_info) = NEW /aws1/cl_migcpsrcimagesetinf00(
          iv_latestversionid = iv_source_version_id ).
        DATA(lo_copy_info) = NEW /aws1/cl_migcpimagesetinfmtion(
          io_sourceimageset = lo_source_info ).
        IF iv_destination_image_set_id IS NOT INITIAL AND
           iv_destination_version_id IS NOT INITIAL.
          DATA(lo_dest_info) = NEW /aws1/cl_migcopydstimageset(
            iv_imagesetid = iv_destination_image_set_id
            iv_latestversionid = iv_destination_version_id ).
          lo_copy_info = NEW /aws1/cl_migcpimagesetinfmtion(
            io_sourceimageset = lo_source_info
            io_destinationimageset = lo_dest_info ).
        ENDIF.
        oo_result = lo_mig->copyimageset(
          iv_datastoreid = iv_datastore_id
          iv_sourceimagesetid = iv_source_image_set_id
          io_copyimagesetinformation = lo_copy_info
          iv_force = iv_force ).
        DATA(lo_dest_props) = oo_result->get_dstimagesetproperties( ).
        DATA(lv_new_id) = lo_dest_props->get_imagesetid( ).
        MESSAGE |Image set copied with new ID: { lv_new_id }.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Image set not found.' TYPE 'I'.
      CATCH /aws1/cx_migservicequotaexcdex.
        MESSAGE 'Service quota exceeded.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

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

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

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

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

Contoh kode berikut menunjukkan cara menggunakan`CreateDatastore`.

Bash

AWS CLI dengan skrip Bash

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function imaging_create_datastore
#
# This function creates an AWS HealthImaging data store for importing DICOM P10
 files.
#
# Parameters:
#       -n data_store_name - The name of the data store.
#
# Returns:
#       The datastore ID.
#    And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
```

```
function imaging_create_datastore() {
  local datastore_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function imaging_create_datastore"
    echo "Creates an AWS HealthImaging data store for importing DICOM P10 files."
    echo "  -n data_store_name - The name of the data store."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) datastore_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$datastore_name" ]]; then
    errecho "ERROR: You must provide a data store name with the -n parameter."
    usage
    return 1
  fi

  response=$(aws medical-imaging create-datastore \
    --datastore-name "$datastore_name" \
    --output text \
    --query 'datastoreId')

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
```

```
    errecho "ERROR: AWS reports medical-imaging create-datastore operation
 failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

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

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

CLI

AWS CLI

Contoh 1: Untuk membuat penyimpanan data

Contoh `create-datastore` kode berikut membuat penyimpanan data dengan namamy-datastore. Saat Anda membuat datastore tanpa menentukan a`--lossless-storage-format`, AWS HealthImaging default ke HTJ2 K (High Throughput JPEG 2000).

```
aws medical-imaging create-datastore \
    --datastore-name "my-datastore"
```

Output:

```
{
    "datastoreId": "12345678901234567890123456789012",
    "datastoreStatus": "CREATING"
}
```

Contoh 2: Untuk membuat penyimpanan data dengan format penyimpanan Lossless JPEG 2000

Penyimpanan data yang dikonfigurasi dengan format penyimpanan Lossless JPEG 2000 akan mentranskode dan mempertahankan bingkai gambar lossless dalam format JPEG 2000. Bingkai gambar kemudian dapat diambil dalam JPEG 2000 Lossless tanpa transcoding. Contoh `create-datastore` kode berikut membuat penyimpanan data dikonfigurasi untuk format penyimpanan Lossless JPEG 2000 dengan nama. `my-datastore`

```
aws medical-imaging create-datastore \
    --datastore-name "my-datastore" \
    --lossless-storage-format JPEG_2000_LOSSLESS
```

Output:

```
{
    "datastoreId": "12345678901234567890123456789012",
    "datastoreStatus": "CREATING"
}
```

Untuk informasi selengkapnya, lihat Membuat penyimpanan data di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

```
   public static String createMedicalImageDatastore(MedicalImagingClient
medicalImagingClient,
           String datastoreName) {
       try {
           CreateDatastoreRequest datastoreRequest =
CreateDatastoreRequest.builder()
                   .datastoreName(datastoreName)
                   .build();
           CreateDatastoreResponse response =
medicalImagingClient.createDatastore(datastoreRequest);
           return response.datastoreId();
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
```

```
        return "";
    }
```

- Untuk detail API, lihat CreateDatastoredi Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

```
import { CreateDatastoreCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreName - The name of the data store to create.
 */
export const createDatastore = async (datastoreName = "DATASTORE_NAME") => {
  const response = await medicalImagingClient.send(
    new CreateDatastoreCommand({ datastoreName: datastoreName }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //      httpStatusCode: 200,
  //      requestId: 'a71cd65f-2382-49bf-b682-f9209d8d399b',
  //      extendedRequestId: undefined,
  //      cfId: undefined,
  //      attempts: 1,
  //      totalRetryDelay: 0
  //    },
  //    datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //    datastoreStatus: 'CREATING'
  // }
  return response;
};
```

- Untuk detail API, lihat CreateDatastoredi Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def create_datastore(self, name):
        """
        Create a data store.

        :param name: The name of the data store to create.
        :return: The data store ID.
        """
        try:
            data_store =
 self.health_imaging_client.create_datastore(datastoreName=name)
        except ClientError as err:
            logger.error(
                "Couldn't create data store %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return data_store["datastoreId"]
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Untuk detail API, lihat CreateDatastoredi 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

```
    TRY.
        " iv_datastore_name = 'my-datastore-name'
        oo_result = lo_mig->createdatastore( iv_datastorename =
iv_datastore_name ).
        DATA(lv_datastore_id) = oo_result->get_datastoreid( ).
        MESSAGE 'Data store created.' TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict. Data store may already exist.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migservicequotaexcdex.
        MESSAGE 'Service quota exceeded.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

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

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

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

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

Contoh kode berikut menunjukkan cara menggunakan`DeleteDatastore`.

Bash

AWS CLI dengan skrip Bash

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function imaging_delete_datastore
#
# This function deletes an AWS HealthImaging data store.
#
# Parameters:
#       -i datastore_id - The ID of the data store.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function imaging_delete_datastore() {
  local datastore_id response
  local option OPTARG # Required to use getopts command in a function.
```

```
# bashsupport disable=BP5008
function usage() {
  echo "function imaging_delete_datastore"
  echo "Deletes an AWS HealthImaging data store."
  echo "  -i datastore_id - The ID of the data store."
  echo ""
}

# Retrieve the calling parameters.
while getopts "i:h" option; do
  case "${option}" in
    i) datastore_id="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$datastore_id" ]]; then
  errecho "ERROR: You must provide a data store ID with the -i parameter."
  usage
  return 1
fi

response=$(aws medical-imaging delete-datastore \
  --datastore-id "$datastore_id")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports medical-imaging delete-datastore operation
failed.$response"
  return 1
fi
```

```
  return 0
}
```

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

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

CLI

AWS CLI

Untuk menghapus penyimpanan data

Contoh `delete-datastore` kode berikut menghapus penyimpanan data.

```
aws medical-imaging delete-datastore \
    --datastore-id "12345678901234567890123456789012"
```

Output:

```
{
    "datastoreId": "12345678901234567890123456789012",
    "datastoreStatus": "DELETING"
}
```

Untuk informasi selengkapnya, lihat Menghapus penyimpanan data di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

```
    public static void deleteMedicalImagingDatastore(MedicalImagingClient
medicalImagingClient,
```

```
            String datastoreID) {
        try {
            DeleteDatastoreRequest datastoreRequest =
 DeleteDatastoreRequest.builder()
                    .datastoreId(datastoreID)
                    .build();
            medicalImagingClient.deleteDatastore(datastoreRequest);
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Untuk detail API, lihat DeleteDatastoredi Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

```
import { DeleteDatastoreCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store to delete.
 */
export const deleteDatastore = async (datastoreId = "DATASTORE_ID") => {
  const response = await medicalImagingClient.send(
    new DeleteDatastoreCommand({ datastoreId }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //           httpStatusCode: 200,
  //           requestId: 'f5beb409-678d-48c9-9173-9a001ee1ebb1',
  //           extendedRequestId: undefined,
```

```
  //           cfId: undefined,
  //           attempts: 1,
  //           totalRetryDelay: 0
  //         },
  //     datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //     datastoreStatus: 'DELETING'
  // }

  return response;
};
```

- Untuk detail API, lihat DeleteDatastoredi Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def delete_datastore(self, datastore_id):
        """
        Delete a data store.

        :param datastore_id: The ID of the data store.
        """
        try:
            self.health_imaging_client.delete_datastore(datastoreId=datastore_id)
        except ClientError as err:
            logger.error(
                "Couldn't delete data store %s. Here's why: %s: %s",
                datastore_id,
                err.response["Error"]["Code"],
```

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

Kode berikut membuat instance objek. MedicalImagingWrapper

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Untuk detail API, lihat DeleteDatastoredi 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

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678901234567890'
        oo_result = lo_mig->deletedatastore( iv_datastoreid = iv_datastore_id ).
        MESSAGE 'Data store deleted.' TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict. Data store may contain resources.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Data store not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

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

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

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

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

Contoh kode berikut menunjukkan cara menggunakan`DeleteImageSet`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- Memulai dengan set gambar dan bingkai gambar

C++

SDK untuk C++

```
//! Routine which deletes an AWS HealthImaging image set.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param imageSetID: The image set ID.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
  */
bool AwsDoc::Medical_Imaging::deleteImageSet(
        const Aws::String &dataStoreID, const Aws::String &imageSetID,
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::DeleteImageSetRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);
```

```
    Aws::MedicalImaging::Model::DeleteImageSetOutcome outcome =
 client.DeleteImageSet(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted image set " << imageSetID
                  << " from data store " << dataStoreID << std::endl;
    }
    else {
        std::cerr << "Error deleting image set " << imageSetID << " from data
 store "
                  << dataStoreID << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Untuk detail API, lihat DeleteImageSetdi Referensi AWS SDK untuk C++ API.

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

CLI

AWS CLI

Untuk menghapus kumpulan gambar

Contoh `delete-image-set` kode berikut menghapus set gambar.

```
aws medical-imaging delete-image-set \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id ea92b0d8838c72a3f25d00d13616f87e
```

Output:

```
{
    "imageSetWorkflowStatus": "DELETING",
```

```
    "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
    "imageSetState": "LOCKED",
    "datastoreId": "12345678901234567890123456789012"
}
```

Untuk informasi selengkapnya, lihat Menghapus set gambar di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

```
   public static void deleteMedicalImageSet(MedicalImagingClient
medicalImagingClient,
           String datastoreId,
           String imagesetId) {
       try {
           DeleteImageSetRequest deleteImageSetRequest =
DeleteImageSetRequest.builder()
                   .datastoreId(datastoreId)
                   .imageSetId(imagesetId)
                   .build();

           medicalImagingClient.deleteImageSet(deleteImageSetRequest);

           System.out.println("The image set was deleted.");
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Untuk detail API, lihat DeleteImageSetdi Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

```
import { DeleteImageSetCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The data store ID.
 * @param {string} imageSetId - The image set ID.
 */
export const deleteImageSet = async (
  datastoreId = "xxxxxxxxxxxxxxx",
  imageSetId = "xxxxxxxxxxxxxxx",
) => {
  const response = await medicalImagingClient.send(
    new DeleteImageSetCommand({
      datastoreId: datastoreId,
      imageSetId: imageSetId,
    }),
  );
  console.log(response);
  // {
  //    '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: '6267bbd2-eaa5-4a50-8ee8-8fddf535cf73',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     datastoreId: 'xxxxxxxxxxxxxxx',
  //     imageSetId: 'xxxxxxxxxxxxxx',
  //     imageSetState: 'LOCKED',
  //     imageSetWorkflowStatus: 'DELETING'
  // }
  return response;
};
```

- Untuk detail API, lihat DeleteImageSetdi Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def delete_image_set(self, datastore_id, image_set_id):
        """
        Delete an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :return: The delete results.
        """
        try:
            delete_results = self.health_imaging_client.delete_image_set(
                imageSetId=image_set_id, datastoreId=datastore_id
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return delete_results
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
    client = boto3.client("medical-imaging")
```

```
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Untuk detail API, lihat DeleteImageSetdi 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

```
TRY.
    " iv_datastore_id = '12345678901234567890123456789012345678901234567890'
    " iv_image_set_id = '12345678901234567890123456789012345678901234567890'
    oo_result = lo_mig->deleteimageset(
      iv_datastoreid = iv_datastore_id
      iv_imagesetid = iv_image_set_id ).
    MESSAGE 'Image set deleted.' TYPE 'I'.
  CATCH /aws1/cx_migaccessdeniedex.
    MESSAGE 'Access denied.' TYPE 'I'.
  CATCH /aws1/cx_migconflictexception.
    MESSAGE 'Conflict error.' TYPE 'I'.
  CATCH /aws1/cx_miginternalserverex.
    MESSAGE 'Internal server error.' TYPE 'I'.
  CATCH /aws1/cx_migresourcenotfoundex.
    MESSAGE 'Image set not found.' TYPE 'I'.
  CATCH /aws1/cx_migthrottlingex.
    MESSAGE 'Request throttled.' TYPE 'I'.
  CATCH /aws1/cx_migvalidationex.
    MESSAGE 'Validation error.' TYPE 'I'.
ENDTRY.
```

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

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

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

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

Contoh kode berikut menunjukkan cara menggunakan`GetDICOMImportJob`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- Memulai dengan set gambar dan bingkai gambar

C++

SDK untuk C++

```
//! Routine which gets a HealthImaging DICOM import job's properties.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param importJobID: The DICOM import job ID
  \param clientConfig: Aws client configuration.
  \return GetDICOMImportJobOutcome: The import job outcome.
*/
Aws::MedicalImaging::Model::GetDICOMImportJobOutcome
AwsDoc::Medical_Imaging::getDICOMImportJob(const Aws::String &dataStoreID,
                                           const Aws::String &importJobID,
                                           const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::GetDICOMImportJobRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetJobId(importJobID);
    Aws::MedicalImaging::Model::GetDICOMImportJobOutcome outcome =
 client.GetDICOMImportJob(
```

```
            request);
    if (!outcome.IsSuccess()) {
        std::cerr << "GetDICOMImportJob error: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome;
}
```

- Untuk detail API, lihat Mendapatkan DICOMImport Job di Referensi AWS SDK untuk C++ API.

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

CLI

AWS CLI

Untuk mendapatkan properti pekerjaan impor dicom

Contoh `get-dicom-import-job` kode berikut mendapatkan properti pekerjaan dicom import.

```
aws medical-imaging get-dicom-import-job \
    --datastore-id "12345678901234567890123456789012" \
    --job-id "09876543210987654321098765432109"
```

Output:

```
{
    "jobProperties": {
        "jobId": "09876543210987654321098765432109",
        "jobName": "my-job",
        "jobStatus": "COMPLETED",
        "datastoreId": "12345678901234567890123456789012",
        "dataAccessRoleArn": "arn:aws:iam::123456789012:role/
ImportJobDataAccessRole",
```

```
        "endedAt": "2022-08-12T11:29:42.285000+00:00",
        "submittedAt": "2022-08-12T11:28:11.152000+00:00",
        "inputS3Uri": "s3://medical-imaging-dicom-input/dicom_input/",
        "outputS3Uri": "s3://medical-imaging-output/
job_output/123456789012345678901234567890 12-
DicomImport-09876543210987654321098765432109/"
    }
}
```

Untuk informasi selengkapnya, lihat Mendapatkan properti pekerjaan impor di Panduan AWS HealthImaging Pengembang.

- Untuk detail API, lihat Mendapatkan DICOMImport Job di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

```
   public static DICOMImportJobProperties getDicomImportJob(MedicalImagingClient
medicalImagingClient,
            String datastoreId,
            String jobId) {

        try {
            GetDicomImportJobRequest getDicomImportJobRequest =
GetDicomImportJobRequest.builder()
                    .datastoreId(datastoreId)
                    .jobId(jobId)
                    .build();
            GetDicomImportJobResponse response =
medicalImagingClient.getDICOMImportJob(getDicomImportJobRequest);
            return response.jobProperties();
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

- Untuk detail API, lihat Mendapatkan DICOMImport Job di Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

```
import { GetDICOMImportJobCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} jobId - The ID of the import job.
 */
export const getDICOMImportJob = async (
  datastoreId = "xxxxxxxxxxxxxxxxxxx",
  jobId = "xxxxxxxxxxxxxxxxxxx",
) => {
  const response = await medicalImagingClient.send(
    new GetDICOMImportJobCommand({ datastoreId: datastoreId, jobId: jobId }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //     httpStatusCode: 200,
  //         requestId: 'a2637936-78ea-44e7-98b8-7a87d95dfaee',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  // },
  //     jobProperties: {
  //         dataAccessRoleArn: 'arn:aws:iam::xxxxxxxxxxxx:role/dicom_import',
  //             datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxx',
  //             endedAt: 2023-09-19T17:29:21.753Z,
  //             inputS3Uri: 's3://healthimaging-source/CTStudy/',
```

```
  //             jobId: ''xxxxxxxxxxxxxxxxxxxxxxxx'',
  //             jobName: 'job_1',
  //             jobStatus: 'COMPLETED',
  //             outputS3Uri: 's3://health-imaging-dest/
ouput_ct/'xxxxxxxxxxxxxxxxxxxxxxxx'-DicomImport-'xxxxxxxxxxxxxxxxxxxxxxxx'/',
  //             submittedAt: 2023-09-19T17:27:25.143Z
  //     }
  // }

  return response;
};
```

- Untuk detail API, lihat Mendapatkan DICOMImport Job di Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def get_dicom_import_job(self, datastore_id, job_id):
        """
        Get the properties of a DICOM import job.

        :param datastore_id: The ID of the data store.
        :param job_id: The ID of the job.
        :return: The job properties.
        """
        try:
            job = self.health_imaging_client.get_dicom_import_job(
                jobId=job_id, datastoreId=datastore_id
```

```
            )
        except ClientError as err:
            logger.error(
                "Couldn't get DICOM import job. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return job["jobProperties"]
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
client = boto3.client("medical-imaging")
medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Untuk detail API, lihat Mendapatkan DICOMImport 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

```
TRY.
    " iv_datastore_id = '12345678901234567890123456789012345678901234567890'
    " iv_job_id = '12345678901234567890123456789012'
    oo_result = lo_mig->getdicomimportjob(
      iv_datastoreid = iv_datastore_id
      iv_jobid = iv_job_id ).
    DATA(lo_job_props) = oo_result->get_jobproperties( ).
    DATA(lv_job_status) = lo_job_props->get_jobstatus( ).
    MESSAGE |Job status: { lv_job_status }.| TYPE 'I'.
  CATCH /aws1/cx_migaccessdeniedex.
```

```
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Job not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Untuk detail API, lihat Mendapatkan DICOMImport Job di AWS SDK untuk referensi SAP ABAP API.

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

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

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

Contoh kode berikut menunjukkan cara menggunakan`GetDatastore`.

Bash

AWS CLI dengan skrip Bash

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
```

```
}

##############################################################################
# function imaging_get_datastore
#
# Get a data store's properties.
#
# Parameters:
#       -i data_store_id - The ID of the data store.
#
# Returns:
#       [datastore_name, datastore_id, datastore_status, datastore_arn,
 created_at, updated_at]
#    And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function imaging_get_datastore() {
  local datastore_id option OPTARG # Required to use getopts command in a
 function.
  local error_code
  # bashsupport disable=BP5008
  function usage() {
    echo "function imaging_get_datastore"
    echo "Gets a data store's properties."
    echo "  -i datastore_id - The ID of the data store."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) datastore_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
```

```
  export OPTIND=1

  if [[ -z "$datastore_id" ]]; then
    errecho "ERROR: You must provide a data store ID with the -i parameter."
    usage
    return 1
  fi

  local response

  response=$(
    aws medical-imaging get-datastore \
      --datastore-id "$datastore_id" \
      --output text \
      --query "[ datastoreProperties.datastoreName,
 datastoreProperties.datastoreId, datastoreProperties.datastoreStatus,
 datastoreProperties.datastoreArn,  datastoreProperties.createdAt,
 datastoreProperties.updatedAt]"
  )
  error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports list-datastores operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

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

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

CLI

AWS CLI

Contoh 1: Untuk mendapatkan properti penyimpanan data

Contoh `get-datastore` kode berikut mendapatkan properti penyimpanan data ini.

```
aws medical-imaging get-datastore \
    --datastore-id 12345678901234567890123456789012
```

Output:

```
{
    "datastoreProperties": {
        "datastoreId": "12345678901234567890123456789012",
        "datastoreName": "TestDatastore123",
        "datastoreStatus": "ACTIVE",
        "losslessStorageFormat": "HTJ2K"
        "datastoreArn": "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012",
        "createdAt": "2022-11-15T23:33:09.643000+00:00",
        "updatedAt": "2022-11-15T23:33:09.643000+00:00"
    }
}
```

Contoh 2: Untuk mendapatkan properti penyimpanan data yang dikonfigurasi untuk JPEG2000

Contoh `get-datastore` kode berikut mendapatkan properti penyimpanan data untuk penyimpanan data yang dikonfigurasi untuk format penyimpanan Lossless JPEG 2000.

```
aws medical-imaging get-datastore \
    --datastore-id 12345678901234567890123456789012
```

Output:

```
{
    "datastoreProperties": {
        "datastoreId": "12345678901234567890123456789012",
        "datastoreName": "TestDatastore123",
        "datastoreStatus": "ACTIVE",
        "losslessStorageFormat": "JPEG_2000_LOSSLESS",
```

```
        "datastoreArn": "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012",
        "createdAt": "2022-11-15T23:33:09.643000+00:00",
        "updatedAt": "2022-11-15T23:33:09.643000+00:00"
    }
}
```

Untuk informasi selengkapnya, lihat Mendapatkan properti penyimpanan data di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

```
   public static DatastoreProperties
getMedicalImageDatastore(MedicalImagingClient medicalImagingClient,
           String datastoreID) {
       try {
           GetDatastoreRequest datastoreRequest = GetDatastoreRequest.builder()
                   .datastoreId(datastoreID)
                   .build();
           GetDatastoreResponse response =
medicalImagingClient.getDatastore(datastoreRequest);
           return response.datastoreProperties();
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return null;
   }
```

- Untuk detail API, lihat GetDatastoredi Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

```
import { GetDatastoreCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreID - The ID of the data store.
 */
export const getDatastore = async (datastoreID = "DATASTORE_ID") => {
  const response = await medicalImagingClient.send(
    new GetDatastoreCommand({ datastoreId: datastoreID }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //       httpStatusCode: 200,
  //       requestId: '55ea7d2e-222c-4a6a-871e-4f591f40cadb',
  //       extendedRequestId: undefined,
  //       cfId: undefined,
  //       attempts: 1,
  //       totalRetryDelay: 0
  //    },
  //   datastoreProperties: {
  //        createdAt: 2023-08-04T18:50:36.239Z,
  //         datastoreArn: 'arn:aws:medical-imaging:us-
east-1:xxxxxxxxx:datastore/xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //         datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //         datastoreName: 'my_datastore',
  //         datastoreStatus: 'ACTIVE',
  //         updatedAt: 2023-08-04T18:50:36.239Z
  //   }
  // }
  return response.datastoreProperties;
};
```

- Untuk detail API, lihat GetDatastoredi Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def get_datastore_properties(self, datastore_id):
        """
        Get the properties of a data store.

        :param datastore_id: The ID of the data store.
        :return: The data store properties.
        """
        try:
            data_store = self.health_imaging_client.get_datastore(
                datastoreId=datastore_id
            )
        except ClientError as err:
            logger.error(
                "Couldn't get data store %s. Here's why: %s: %s",
                id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return data_store["datastoreProperties"]
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
    client = boto3.client("medical-imaging")
```

```
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Untuk detail API, lihat GetDatastoredi 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

```
TRY.
    " iv_datastore_id = '12345678901234567890123456789012345678901234567890'
    oo_result = lo_mig->getdatastore( iv_datastoreid = iv_datastore_id ).
    DATA(lo_properties) = oo_result->get_datastoreproperties( ).
    DATA(lv_name) = lo_properties->get_datastorename( ).
    DATA(lv_status) = lo_properties->get_datastorestatus( ).
    MESSAGE 'Data store properties retrieved.' TYPE 'I'.
  CATCH /aws1/cx_migaccessdeniedex.
    MESSAGE 'Access denied.' TYPE 'I'.
  CATCH /aws1/cx_miginternalserverex.
    MESSAGE 'Internal server error.' TYPE 'I'.
  CATCH /aws1/cx_migresourcenotfoundex.
    MESSAGE 'Data store not found.' TYPE 'I'.
  CATCH /aws1/cx_migthrottlingex.
    MESSAGE 'Request throttled.' TYPE 'I'.
  CATCH /aws1/cx_migvalidationex.
    MESSAGE 'Validation error.' TYPE 'I'.
ENDTRY.
```

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

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

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

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

Contoh kode berikut menunjukkan cara menggunakan`GetImageFrame`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- Memulai dengan set gambar dan bingkai gambar

C++

SDK untuk C++

```
//! Routine which downloads an AWS HealthImaging image frame.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param imageSetID: The image set ID.
  \param frameID: The image frame ID.
  \param jphFile: File to store the downloaded frame.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::Medical_Imaging::getImageFrame(const Aws::String &dataStoreID,
                                            const Aws::String &imageSetID,
                                            const Aws::String &frameID,
                                            const Aws::String &jphFile,
                                            const
 Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);

    Aws::MedicalImaging::Model::GetImageFrameRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);

    Aws::MedicalImaging::Model::ImageFrameInformation imageFrameInformation;
    imageFrameInformation.SetImageFrameId(frameID);
    request.SetImageFrameInformation(imageFrameInformation);
```

```
    Aws::MedicalImaging::Model::GetImageFrameOutcome outcome =
 client.GetImageFrame(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully retrieved image frame." << std::endl;
        auto &buffer = outcome.GetResult().GetImageFrameBlob();

        std::ofstream outfile(jphFile, std::ios::binary);
        outfile << buffer.rdbuf();
    }
    else {
        std::cout << "Error retrieving image frame." <<
 outcome.GetError().GetMessage()
                  << std::endl;

    }

    return outcome.IsSuccess();
}
```

- Untuk detail API, lihat GetImageFramedi Referensi AWS SDK untuk C++ API.

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

CLI

AWS CLI

Untuk mendapatkan data piksel set gambar

Contoh `get-image-frame` kode berikut mendapat bingkai gambar.

```
aws medical-imaging get-image-frame \
    --datastore-id "12345678901234567890123456789012" \
    --image-set-id "98765412345612345678907890789012" \
    --image-frame-information imageFrameId=3abf5d5d7ae72f80a0ec81b2c0de3ef4 \
```

```
    imageframe.jph
```

Catatan: Contoh kode ini tidak menyertakan output karena GetImageFrame tindakan mengembalikan aliran data piksel ke file imageframe.jph. Untuk informasi tentang decoding dan melihat bingkai gambar, lihat HTJ2 K decoding library.

Untuk informasi selengkapnya, lihat Mendapatkan data piksel yang disetel gambar di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

```
       public static void getMedicalImageSetFrame(MedicalImagingClient
medicalImagingClient,
                        String destinationPath,
                        String datastoreId,
                        String imagesetId,
                        String imageFrameId) {

                try {
                        GetImageFrameRequest getImageSetMetadataRequest =
GetImageFrameRequest.builder()
                                        .datastoreId(datastoreId)
                                        .imageSetId(imagesetId)

.imageFrameInformation(ImageFrameInformation.builder()

.imageFrameId(imageFrameId)
                                                        .build())
                                        .build();

medicalImagingClient.getImageFrame(getImageSetMetadataRequest,

FileSystems.getDefault().getPath(destinationPath));

                        System.out.println("Image frame downloaded to " +
destinationPath);
                } catch (MedicalImagingException e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
```

```
                }
        }
```

- Untuk detail API, lihat GetImageFramedi Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

```
import { GetImageFrameCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} imageFrameFileName - The name of the file for the HTJ2K-
encoded image frame.
 * @param {string} datastoreID - The data store's ID.
 * @param {string} imageSetID - The image set's ID.
 * @param {string} imageFrameID - The image frame's ID.
 */
export const getImageFrame = async (
  imageFrameFileName = "image.jph",
  datastoreID = "DATASTORE_ID",
  imageSetID = "IMAGE_SET_ID",
  imageFrameID = "IMAGE_FRAME_ID",
) => {
  const response = await medicalImagingClient.send(
    new GetImageFrameCommand({
      datastoreId: datastoreID,
      imageSetId: imageSetID,
      imageFrameInformation: { imageFrameId: imageFrameID },
    }),
  );
  const buffer = await response.imageFrameBlob.transformToByteArray();
  writeFileSync(imageFrameFileName, buffer);
```

```
  console.log(response);
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: 'e4ab42a5-25a3-4377-873f-374ecf4380e1',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     contentType: 'application/octet-stream',
  //     imageFrameBlob: <ref *1> IncomingMessage {}
  // }
  return response;
};
```

- Untuk detail API, lihat GetImageFramedi Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def get_pixel_data(
        self, file_path_to_write, datastore_id, image_set_id, image_frame_id
    ):
        """
        Get an image frame's pixel data.

        :param file_path_to_write: The path to write the image frame's HTJ2K
 encoded pixel data.
```

```
        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param image_frame_id: The ID of the image frame.
        """
        try:
            image_frame = self.health_imaging_client.get_image_frame(
                datastoreId=datastore_id,
                imageSetId=image_set_id,
                imageFrameInformation={"imageFrameId": image_frame_id},
            )
            with open(file_path_to_write, "wb") as f:
                for chunk in image_frame["imageFrameBlob"].iter_chunks():
                    if chunk:
                        f.write(chunk)
        except ClientError as err:
            logger.error(
                "Couldn't get image frame. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
client = boto3.client("medical-imaging")
medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Untuk detail API, lihat GetImageFramedi 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

```
TRY.
    " iv_datastore_id = '12345678901234567890123456789012345678901234567890'
    " iv_image_set_id = '12345678901234567890123456789012345678901234567890'
    " iv_image_frame_id = '12345678901234567890123456789012345678901234567890'
    oo_result = lo_mig->getimageframe(
      iv_datastoreid = iv_datastore_id
      iv_imagesetid = iv_image_set_id
      io_imageframeinformation = NEW /aws1/cl_migimageframeinfmtion(
        iv_imageframeid = iv_image_frame_id ) ).
    DATA(lv_frame_blob) = oo_result->get_imageframeblob( ).
    MESSAGE 'Image frame retrieved.' TYPE 'I'.
  CATCH /aws1/cx_migaccessdeniedex.
    MESSAGE 'Access denied.' TYPE 'I'.
  CATCH /aws1/cx_migconflictexception.
    MESSAGE 'Conflict error.' TYPE 'I'.
  CATCH /aws1/cx_miginternalserverex.
    MESSAGE 'Internal server error.' TYPE 'I'.
  CATCH /aws1/cx_migresourcenotfoundex.
    MESSAGE 'Image frame not found.' TYPE 'I'.
  CATCH /aws1/cx_migthrottlingex.
    MESSAGE 'Request throttled.' TYPE 'I'.
  CATCH /aws1/cx_migvalidationex.
    MESSAGE 'Validation error.' TYPE 'I'.
ENDTRY.
```

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

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

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

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

Contoh kode berikut menunjukkan cara menggunakan`GetImageSet`.

CLI

AWS CLI

Untuk mendapatkan properti set gambar

Contoh `get-image-set` kode berikut mendapatkan properti untuk set gambar.

```
aws medical-imaging get-image-set \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id 18f88ac7870584f58d56256646b4d92b \
    --version-id 1
```

Output:

```
{
    "versionId": "1",
    "imageSetWorkflowStatus": "COPIED",
    "updatedAt": 1680027253.471,
    "imageSetId": "18f88ac7870584f58d56256646b4d92b",
    "imageSetState": "ACTIVE",
    "createdAt": 1679592510.753,
    "datastoreId": "12345678901234567890123456789012"
}
```

Untuk informasi selengkapnya, lihat Mendapatkan properti set gambar di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

```
    public static GetImageSetResponse getMedicalImageSet(MedicalImagingClient
 medicalImagingClient,
            String datastoreId,
            String imagesetId,
```

```
            String versionId) {
        try {
            GetImageSetRequest.Builder getImageSetRequestBuilder =
GetImageSetRequest.builder()
                    .datastoreId(datastoreId)
                    .imageSetId(imagesetId);

            if (versionId != null) {
                getImageSetRequestBuilder =
getImageSetRequestBuilder.versionId(versionId);
            }

            return
medicalImagingClient.getImageSet(getImageSetRequestBuilder.build());
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

- Untuk detail API, lihat GetImageSetdi Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

```
import { GetImageSetCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} imageSetId - The ID of the image set.
 * @param {string} imageSetVersion - The optional version of the image set.
```

```
 *
 */
export const getImageSet = async (
  datastoreId = "xxxxxxxxxxxxxx",
  imageSetId = "xxxxxxxxxxxxxx",
  imageSetVersion = "",
) => {
  const params = { datastoreId: datastoreId, imageSetId: imageSetId };
  if (imageSetVersion !== "") {
    params.imageSetVersion = imageSetVersion;
  }
  const response = await medicalImagingClient.send(
    new GetImageSetCommand(params),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //     httpStatusCode: 200,
  //         requestId: '0615c161-410d-4d06-9d8c-6e1241bb0a5a',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  // },
  //     createdAt: 2023-09-22T14:49:26.427Z,
  //     datastoreId: 'xxxxxxxxxxxxxx',
  //     imageSetArn: 'arn:aws:medical-imaging:us-east-1:xxxxxxxxxx:datastore/
xxxxxxxxxxxxxxxxxx/imageset/xxxxxxxxxxxxxxxxxx',
  //     imageSetId: 'xxxxxxxxxxxxxx',
  //     imageSetState: 'ACTIVE',
  //     imageSetWorkflowStatus: 'CREATED',
  //     updatedAt: 2023-09-22T14:49:26.427Z,
  //     versionId: '1'
  // }

  return response;
};
```

- Untuk detail API, lihat GetImageSetdi Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def get_image_set(self, datastore_id, image_set_id, version_id=None):
        """
        Get the properties of an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The optional version of the image set.
        :return: The image set properties.
        """
        try:
            if version_id:
                image_set = self.health_imaging_client.get_image_set(
                    imageSetId=image_set_id,
                    datastoreId=datastore_id,
                    versionId=version_id,
                )
            else:
                image_set = self.health_imaging_client.get_image_set(
                    imageSetId=image_set_id, datastoreId=datastore_id
                )
        except ClientError as err:
            logger.error(
                "Couldn't get image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
        else:
            return image_set
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Untuk detail API, lihat GetImageSetdi 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

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678901234567890'
        " iv_image_set_id = '12345678901234567890123456789012345678901234567890'
        " iv_version_id = '1' (optional)
        IF iv_version_id IS NOT INITIAL.
          oo_result = lo_mig->getimageset(
            iv_datastoreid = iv_datastore_id
            iv_imagesetid = iv_image_set_id
            iv_versionid = iv_version_id ).
        ELSE.
          oo_result = lo_mig->getimageset(
            iv_datastoreid = iv_datastore_id
            iv_imagesetid = iv_image_set_id ).
        ENDIF.
        DATA(lv_state) = oo_result->get_imagesetstate( ).
        MESSAGE |Image set retrieved with state: { lv_state }.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
```

```
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Image set not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

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

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

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

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

Contoh kode berikut menunjukkan cara menggunakan`GetImageSetMetadata`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- Memulai dengan set gambar dan bingkai gambar

C++

SDK untuk C++

Fungsi utilitas untuk mendapatkan metadata set gambar.

```
//! Routine which gets a HealthImaging image set's metadata.
/*!
```

```
  \param dataStoreID: The HealthImaging data store ID.
  \param imageSetID: The HealthImaging image set ID.
  \param versionID: The HealthImaging image set version ID, ignored if empty.
  \param outputFilePath: The path where the metadata will be stored as gzipped
 json.
  \param clientConfig: Aws client configuration.
  \\return bool: Function succeeded.
*/
bool AwsDoc::Medical_Imaging::getImageSetMetadata(const Aws::String &dataStoreID,
                                                  const Aws::String &imageSetID,
                                                  const Aws::String &versionID,
                                                  const Aws::String
 &outputFilePath,
                                                  const
 Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::Model::GetImageSetMetadataRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);
    if (!versionID.empty()) {
        request.SetVersionId(versionID);
    }
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::GetImageSetMetadataOutcome outcome =
 client.GetImageSetMetadata(
            request);
    if (outcome.IsSuccess()) {
        std::ofstream file(outputFilePath, std::ios::binary);
        auto &metadata = outcome.GetResult().GetImageSetMetadataBlob();
        file << metadata.rdbuf();
    }
    else {
        std::cerr << "Failed to get image set metadata: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

Dapatkan metadata set gambar tanpa versi.

```
        if (AwsDoc::Medical_Imaging::getImageSetMetadata(dataStoreID, imageSetID,
 "", outputFilePath, clientConfig))
```

```
        {
            std::cout << "Successfully retrieved image set metadata." <<
std::endl;
            std::cout << "Metadata stored in: " << outputFilePath << std::endl;
        }
```

Dapatkan metadata set gambar dengan versi.

```
        if (AwsDoc::Medical_Imaging::getImageSetMetadata(dataStoreID, imageSetID,
versionID, outputFilePath, clientConfig))
        {
            std::cout << "Successfully retrieved image set metadata." <<
std::endl;
            std::cout << "Metadata stored in: " << outputFilePath << std::endl;
        }
```

- Untuk detail API, lihat GetImageSetMetadatadi Referensi AWS SDK untuk C++ API.

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

CLI

AWS CLI

Contoh 1: Untuk mendapatkan metadata set gambar tanpa versi

Contoh `get-image-set-metadata` kode berikut mendapatkan metadata untuk kumpulan gambar tanpa menentukan versi.

Catatan: `outfile` adalah parameter yang diperlukan

```
aws medical-imaging get-image-set-metadata \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id ea92b0d8838c72a3f25d00d13616f87e \
    studymetadata.json.gz
```

Metadata yang dikembalikan dikompresi dengan gzip dan disimpan dalam file studymetadata.json.gz. Untuk melihat isi objek JSON yang dikembalikan, Anda harus terlebih dahulu mendekompresnya.

Output:

```
{
    "contentType": "application/json",
    "contentEncoding": "gzip"
}
```

Contoh 2: Untuk mendapatkan metadata set gambar dengan versi

Contoh `get-image-set-metadata` kode berikut mendapatkan metadata untuk set gambar dengan versi tertentu.

Catatan: `outfile` adalah parameter yang diperlukan

```
aws medical-imaging get-image-set-metadata \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id ea92b0d8838c72a3f25d00d13616f87e \
    --version-id 1 \
    studymetadata.json.gz
```

Metadata yang dikembalikan dikompresi dengan gzip dan disimpan dalam file studymetadata.json.gz. Untuk melihat isi objek JSON yang dikembalikan, Anda harus terlebih dahulu mendekompresnya.

Output:

```
{
    "contentType": "application/json",
    "contentEncoding": "gzip"
}
```

Untuk informasi selengkapnya, lihat Mendapatkan metadata set gambar di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

```
   public static void getMedicalImageSetMetadata(MedicalImagingClient
medicalImagingClient,
           String destinationPath,
           String datastoreId,
           String imagesetId,
           String versionId) {

       try {
           GetImageSetMetadataRequest.Builder getImageSetMetadataRequestBuilder
= GetImageSetMetadataRequest.builder()
                   .datastoreId(datastoreId)
                   .imageSetId(imagesetId);

           if (versionId != null) {
               getImageSetMetadataRequestBuilder =
getImageSetMetadataRequestBuilder.versionId(versionId);
           }


medicalImagingClient.getImageSetMetadata(getImageSetMetadataRequestBuilder.build(),
                   FileSystems.getDefault().getPath(destinationPath));

           System.out.println("Metadata downloaded to " + destinationPath);
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Untuk detail API, lihat GetImageSetMetadatadi Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

Fungsi utilitas untuk mendapatkan metadata set gambar.

```
import { GetImageSetMetadataCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";
import { writeFileSync } from "node:fs";

/**
 * @param {string} metadataFileName - The name of the file for the gzipped
 metadata.
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} imagesetId - The ID of the image set.
 * @param {string} versionID - The optional version ID of the image set.
 */
export const getImageSetMetadata = async (
  metadataFileName = "metadata.json.gzip",
  datastoreId = "xxxxxxxxxxxxxx",
  imagesetId = "xxxxxxxxxxxxxx",
  versionID = "",
) => {
  const params = { datastoreId: datastoreId, imageSetId: imagesetId };

  if (versionID) {
    params.versionID = versionID;
  }

  const response = await medicalImagingClient.send(
    new GetImageSetMetadataCommand(params),
  );
  const buffer = await response.imageSetMetadataBlob.transformToByteArray();
  writeFileSync(metadataFileName, buffer);

  console.log(response);
  // {
  //     '$metadata': {
  //     httpStatusCode: 200,
  //         requestId: '5219b274-30ff-4986-8cab-48753de3a599',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
```

```
  // },
  //     contentType: 'application/json',
  //     contentEncoding: 'gzip',
  //     imageSetMetadataBlob: <ref *1> IncomingMessage {}
  // }

  return response;
};
```

Dapatkan metadata set gambar tanpa versi.

```
  try {
    await getImageSetMetadata(
      "metadata.json.gzip",
      "12345678901234567890123456789012",
      "12345678901234567890123456789012",
    );
  } catch (err) {
    console.log("Error", err);
  }
```

Dapatkan metadata set gambar dengan versi.

```
  try {
    await getImageSetMetadata(
      "metadata2.json.gzip",
      "12345678901234567890123456789012",
      "12345678901234567890123456789012",
      "1",
    );
  } catch (err) {
    console.log("Error", err);
  }
```

- Untuk detail API, lihat GetImageSetMetadatadi Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

Fungsi utilitas untuk mendapatkan metadata set gambar.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def get_image_set_metadata(
        self, metadata_file, datastore_id, image_set_id, version_id=None
    ):
        """
        Get the metadata of an image set.

        :param metadata_file: The file to store the JSON gzipped metadata.
        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The version of the image set.
        """
        try:
            if version_id:
                image_set_metadata =
 self.health_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id,
                    datastoreId=datastore_id,
                    versionId=version_id,
                )
            else:

                image_set_metadata =
 self.health_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id, datastoreId=datastore_id
                )
            print(image_set_metadata)
```

```
            with open(metadata_file, "wb") as f:
                for chunk in
image_set_metadata["imageSetMetadataBlob"].iter_chunks():
                    if chunk:
                        f.write(chunk)

        except ClientError as err:
            logger.error(
                "Couldn't get image metadata. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Dapatkan metadata set gambar tanpa versi.

```
                image_set_metadata =
self.health_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id, datastoreId=datastore_id
                )
```

Dapatkan metadata set gambar dengan versi.

```
                image_set_metadata =
self.health_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id,
                    datastoreId=datastore_id,
                    versionId=version_id,
                )
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Untuk detail API, lihat GetImageSetMetadatadi 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

```
TRY.
    " iv_datastore_id = '12345678901234567890123456789012345678901234567890'
    " iv_image_set_id = '12345678901234567890123456789012345678901234567890'
    " iv_version_id = '1' (optional)
    IF iv_version_id IS NOT INITIAL.
      oo_result = lo_mig->getimagesetmetadata(
        iv_datastoreid = iv_datastore_id
        iv_imagesetid = iv_image_set_id
        iv_versionid = iv_version_id ).
    ELSE.
      oo_result = lo_mig->getimagesetmetadata(
        iv_datastoreid = iv_datastore_id
        iv_imagesetid = iv_image_set_id ).
    ENDIF.
    DATA(lv_metadata_blob) = oo_result->get_imagesetmetadatablob( ).
    MESSAGE 'Image set metadata retrieved.' TYPE 'I'.
  CATCH /aws1/cx_migaccessdeniedex.
    MESSAGE 'Access denied.' TYPE 'I'.
  CATCH /aws1/cx_migconflictexception.
    MESSAGE 'Conflict error.' TYPE 'I'.
  CATCH /aws1/cx_miginternalserverex.
    MESSAGE 'Internal server error.' TYPE 'I'.
  CATCH /aws1/cx_migresourcenotfoundex.
    MESSAGE 'Image set not found.' TYPE 'I'.
  CATCH /aws1/cx_migthrottlingex.
    MESSAGE 'Request throttled.' TYPE 'I'.
  CATCH /aws1/cx_migvalidationex.
    MESSAGE 'Validation error.' TYPE 'I'.
ENDTRY.
```

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

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

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

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

Contoh kode berikut menunjukkan cara menggunakan`ListDICOMImportJobs`.

CLI

AWS CLI

Untuk daftar pekerjaan dicom import

Contoh `list-dicom-import-jobs` kode berikut mencantumkan pekerjaan impor dicom.

```
aws medical-imaging list-dicom-import-jobs \
    --datastore-id "12345678901234567890123456789012"
```

Output:

```
{
    "jobSummaries": [
        {
            "jobId": "09876543210987654321098765432109",
            "jobName": "my-job",
            "jobStatus": "COMPLETED",
            "datastoreId": "12345678901234567890123456789012",
            "dataAccessRoleArn": "arn:aws:iam::123456789012:role/
ImportJobDataAccessRole",
            "endedAt": "2022-08-12T11:21:56.504000+00:00",
            "submittedAt": "2022-08-12T11:20:21.734000+00:00"
        }
    ]
}
```

Untuk informasi selengkapnya, lihat Daftar pekerjaan impor di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

```
   public static List<DICOMImportJobSummary>
listDicomImportJobs(MedicalImagingClient medicalImagingClient,
           String datastoreId) {

       try {
           ListDicomImportJobsRequest listDicomImportJobsRequest =
ListDicomImportJobsRequest.builder()
                   .datastoreId(datastoreId)
                   .build();
           ListDicomImportJobsResponse response =
medicalImagingClient.listDICOMImportJobs(listDicomImportJobsRequest);
           return response.jobSummaries();
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return new ArrayList<>();
   }
```

- Untuk detail API, lihat Daftar DICOMImport Pekerjaan di Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

```
import { paginateListDICOMImportJobs } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store.
 */
export const listDICOMImportJobs = async (
  datastoreId = "xxxxxxxxxxxxxxxxx",
) => {
  const paginatorConfig = {
    client: medicalImagingClient,
    pageSize: 50,
  };

  const commandParams = { datastoreId: datastoreId };
  const paginator = paginateListDICOMImportJobs(paginatorConfig, commandParams);

  const jobSummaries = [];
  for await (const page of paginator) {
    // Each page contains a list of `jobSummaries`. The list is truncated if is
 larger than `pageSize`.
    jobSummaries.push(...page.jobSummaries);
    console.log(page);
  }
  // {
  //     '$metadata': {
  //     httpStatusCode: 200,
  //         requestId: '3c20c66e-0797-446a-a1d8-91b742fd15a0',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  // },
  //     jobSummaries: [
  //         {
  //             dataAccessRoleArn: 'arn:aws:iam::xxxxxxxxxxxx:role/
dicom_import',
  //             datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxx',
  //             endedAt: 2023-09-22T14:49:51.351Z,
  //             jobId: 'xxxxxxxxxxxxxxxxxxxxxxxxx',
```

```
  //             jobName: 'test-1',
  //             jobStatus: 'COMPLETED',
  //             submittedAt: 2023-09-22T14:48:45.767Z
  // }
  // ]}

  return jobSummaries;
};
```

- Untuk detail API, lihat Daftar DICOMImport Pekerjaan di Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def list_dicom_import_jobs(self, datastore_id):
        """
        List the DICOM import jobs.

        :param datastore_id: The ID of the data store.
        :return: The list of jobs.
        """
        try:
            paginator = self.health_imaging_client.get_paginator(
                "list_dicom_import_jobs"
            )
            page_iterator = paginator.paginate(datastoreId=datastore_id)
            job_summaries = []
            for page in page_iterator:
```

```
                job_summaries.extend(page["jobSummaries"])
        except ClientError as err:
            logger.error(
                "Couldn't list DICOM import jobs. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return job_summaries
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Untuk detail API, lihat Daftar DICOMImport 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

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678901234567890'
        oo_result = lo_mig->listdicomimportjobs( iv_datastoreid =
iv_datastore_id ).
        DATA(lt_jobs) = oo_result->get_jobsummaries( ).
        DATA(lv_count) = lines( lt_jobs ).
        MESSAGE |Found { lv_count } DICOM import jobs.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
```

```
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

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

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

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

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

Contoh kode berikut menunjukkan cara menggunakan`ListDatastores`.

Bash

AWS CLI dengan skrip Bash

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}
```

```
################################################################################
# function imaging_list_datastores
#
# List the HealthImaging data stores in the account.
#
# Returns:
#       [[datastore_name, datastore_id, datastore_status]]
#    And:
#       0 - If successful.
#       1 - If it fails.
################################################################################
function imaging_list_datastores() {
  local option OPTARG # Required to use getopts command in a function.
  local error_code
  # bashsupport disable=BP5008
  function usage() {
    echo "function imaging_list_datastores"
    echo "Lists the AWS HealthImaging data stores in the account."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "h" option; do
    case "${option}" in
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local response
  response=$(aws medical-imaging list-datastores \
    --output text \
    --query "datastoreSummaries[*][datastoreName, datastoreId, datastoreStatus]")
  error_code=${?}

  if [[ $error_code -ne 0 ]]; then
```

```
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports list-datastores operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

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

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

CLI

AWS CLI

Untuk daftar penyimpanan data

Contoh `list-datastores` kode berikut mencantumkan penyimpanan data yang tersedia.

```
aws medical-imaging list-datastores
```

Output:

```
{
    "datastoreSummaries": [
        {
            "datastoreId": "12345678901234567890123456789012",
            "datastoreName": "TestDatastore123",
            "datastoreStatus": "ACTIVE",
            "datastoreArn": "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012",
            "createdAt": "2022-11-15T23:33:09.643000+00:00",
            "updatedAt": "2022-11-15T23:33:09.643000+00:00"
        }
```

```
    ]
}
```

Untuk informasi selengkapnya, lihat Menyimpan penyimpanan data di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

```
   public static List<DatastoreSummary>
listMedicalImagingDatastores(MedicalImagingClient medicalImagingClient) {
       try {
           ListDatastoresRequest datastoreRequest =
ListDatastoresRequest.builder()
                   .build();
           ListDatastoresIterable responses =
medicalImagingClient.listDatastoresPaginator(datastoreRequest);
           List<DatastoreSummary> datastoreSummaries = new ArrayList<>();

           responses.stream().forEach(response ->
datastoreSummaries.addAll(response.datastoreSummaries()));

           return datastoreSummaries;
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return null;
   }
```

- Untuk detail API, lihat ListDatastoresdi Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

```
import { paginateListDatastores } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

export const listDatastores = async () => {
  const paginatorConfig = {
    client: medicalImagingClient,
    pageSize: 50,
  };

  const commandParams = {};
  const paginator = paginateListDatastores(paginatorConfig, commandParams);

  /**
   * @type {import("@aws-sdk/client-medical-imaging").DatastoreSummary[]}
   */
  const datastoreSummaries = [];
  for await (const page of paginator) {
    // Each page contains a list of `jobSummaries`. The list is truncated if is
 larger than `pageSize`.
    datastoreSummaries.push(...page.datastoreSummaries);
    console.log(page);
  }
  // {
  //   '$metadata': {
  //       httpStatusCode: 200,
  //       requestId: '6aa99231-d9c2-4716-a46e-edb830116fa3',
  //       extendedRequestId: undefined,
  //       cfId: undefined,
  //       attempts: 1,
  //       totalRetryDelay: 0
  //   },
  //   datastoreSummaries: [
  //     {
  //       createdAt: 2023-08-04T18:49:54.429Z,
  //       datastoreArn: 'arn:aws:medical-imaging:us-east-1:xxxxxxxxx:datastore/
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //       datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //       datastoreName: 'my_datastore',
  //       datastoreStatus: 'ACTIVE',
  //       updatedAt: 2023-08-04T18:49:54.429Z
```

```
  //     }
  //     ...
  //   ]
  // }

  return datastoreSummaries;
};
```

- Untuk detail API, lihat ListDatastoresdi Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def list_datastores(self):
        """
        List the data stores.

        :return: The list of data stores.
        """
        try:
            paginator =
 self.health_imaging_client.get_paginator("list_datastores")
            page_iterator = paginator.paginate()
            datastore_summaries = []
            for page in page_iterator:
                datastore_summaries.extend(page["datastoreSummaries"])
        except ClientError as err:
            logger.error(
                "Couldn't list data stores. Here's why: %s: %s",
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return datastore_summaries
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
client = boto3.client("medical-imaging")
medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Untuk detail API, lihat ListDatastoresdi 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

```
TRY.
    oo_result = lo_mig->listdatastores( ).
    DATA(lt_datastores) = oo_result->get_datastoresummaries( ).
    DATA(lv_count) = lines( lt_datastores ).
    MESSAGE |Found { lv_count } data stores.| TYPE 'I'.
  CATCH /aws1/cx_migaccessdeniedex.
    MESSAGE 'Access denied.' TYPE 'I'.
  CATCH /aws1/cx_miginternalserverex.
    MESSAGE 'Internal server error.' TYPE 'I'.
  CATCH /aws1/cx_migthrottlingex.
    MESSAGE 'Request throttled.' TYPE 'I'.
  CATCH /aws1/cx_migvalidationex.
    MESSAGE 'Validation error.' TYPE 'I'.
ENDTRY.
```

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

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

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

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

Contoh kode berikut menunjukkan cara menggunakan`ListImageSetVersions`.

CLI

AWS CLI

Untuk daftar versi set gambar

Contoh `list-image-set-versions` kode berikut mencantumkan riwayat versi untuk kumpulan gambar.

```
aws medical-imaging list-image-set-versions \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id ea92b0d8838c72a3f25d00d13616f87e
```

Output:

```
{
    "imageSetPropertiesList": [
        {
            "ImageSetWorkflowStatus": "UPDATED",
            "versionId": "4",
            "updatedAt": 1680029436.304,
            "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
            "imageSetState": "ACTIVE",
            "createdAt": 1680027126.436
```

```
        },
        {
            "ImageSetWorkflowStatus": "UPDATED",
            "versionId": "3",
            "updatedAt": 1680029163.325,
            "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
            "imageSetState": "ACTIVE",
            "createdAt": 1680027126.436
        },
        {
            "ImageSetWorkflowStatus": "COPY_FAILED",
            "versionId": "2",
            "updatedAt": 1680027455.944,
            "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
            "imageSetState": "ACTIVE",
            "message": "INVALID_REQUEST:  Series of SourceImageSet and
 DestinationImageSet don't match.",
            "createdAt": 1680027126.436
        },
        {
            "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
            "imageSetState": "ACTIVE",
            "versionId": "1",
            "ImageSetWorkflowStatus": "COPIED",
            "createdAt": 1680027126.436
        }
    ]
}
```

Untuk informasi selengkapnya, lihat Daftar versi kumpulan gambar di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

```
    public static List<ImageSetProperties>
 listMedicalImageSetVersions(MedicalImagingClient medicalImagingClient,
            String datastoreId,
            String imagesetId) {
        try {
```

```
            ListImageSetVersionsRequest getImageSetRequest =
 ListImageSetVersionsRequest.builder()
                    .datastoreId(datastoreId)
                    .imageSetId(imagesetId)
                    .build();

            ListImageSetVersionsIterable responses = medicalImagingClient
                    .listImageSetVersionsPaginator(getImageSetRequest);
            List<ImageSetProperties> imageSetProperties = new ArrayList<>();
            responses.stream().forEach(response ->
 imageSetProperties.addAll(response.imageSetPropertiesList()));

            return imageSetProperties;
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

- Untuk detail API, lihat ListImageSetVersionsdi Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

```
import { paginateListImageSetVersions } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} imageSetId - The ID of the image set.
 */
export const listImageSetVersions = async (
```

```
  datastoreId = "xxxxxxxxxxxx",
  imageSetId = "xxxxxxxxxxxx",
) => {
  const paginatorConfig = {
    client: medicalImagingClient,
    pageSize: 50,
  };

  const commandParams = { datastoreId, imageSetId };
  const paginator = paginateListImageSetVersions(
    paginatorConfig,
    commandParams,
  );

  const imageSetPropertiesList = [];
  for await (const page of paginator) {
    // Each page contains a list of `jobSummaries`. The list is truncated if is
 larger than `pageSize`.
    imageSetPropertiesList.push(...page.imageSetPropertiesList);
    console.log(page);
  }
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: '74590b37-a002-4827-83f2-3c590279c742',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     imageSetPropertiesList: [
  //         {
  //             ImageSetWorkflowStatus: 'CREATED',
  //             createdAt: 2023-09-22T14:49:26.427Z,
  //             imageSetId: 'xxxxxxxxxxxxxxxxxxxxxx',
  //             imageSetState: 'ACTIVE',
  //             versionId: '1'
  //         }]
  // }
  return imageSetPropertiesList;
};
```

- Untuk detail API, lihat ListImageSetVersionsdi Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def list_image_set_versions(self, datastore_id, image_set_id):
        """
        List the image set versions.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :return: The list of image set versions.
        """
        try:
            paginator = self.health_imaging_client.get_paginator(
                "list_image_set_versions"
            )
            page_iterator = paginator.paginate(
                imageSetId=image_set_id, datastoreId=datastore_id
            )
            image_set_properties_list = []
            for page in page_iterator:
                image_set_properties_list.extend(page["imageSetPropertiesList"])
        except ClientError as err:
            logger.error(
                "Couldn't list image set versions. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return image_set_properties_list
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
client = boto3.client("medical-imaging")
medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Untuk detail API, lihat ListImageSetVersionsdi 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

```
TRY.
    " iv_datastore_id = '12345678901234567890123456789012345678901234567890'
    " iv_image_set_id = '12345678901234567890123456789012345678901234567890'
    oo_result = lo_mig->listimagesetversions(
      iv_datastoreid = iv_datastore_id
      iv_imagesetid = iv_image_set_id ).
    DATA(lt_versions) = oo_result->get_imagesetpropertieslist( ).
    DATA(lv_count) = lines( lt_versions ).
    MESSAGE |Found { lv_count } image set versions.| TYPE 'I'.
  CATCH /aws1/cx_migaccessdeniedex.
    MESSAGE 'Access denied.' TYPE 'I'.
  CATCH /aws1/cx_migconflictexception.
    MESSAGE 'Conflict error.' TYPE 'I'.
  CATCH /aws1/cx_miginternalserverex.
    MESSAGE 'Internal server error.' TYPE 'I'.
  CATCH /aws1/cx_migresourcenotfoundex.
    MESSAGE 'Image set not found.' TYPE 'I'.
  CATCH /aws1/cx_migthrottlingex.
    MESSAGE 'Request throttled.' TYPE 'I'.
  CATCH /aws1/cx_migvalidationex.
    MESSAGE 'Validation error.' TYPE 'I'.
```

```
    ENDTRY.
```

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

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

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan layanan ini 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`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- Menandai penyimpanan data
- Menandai set gambar

CLI

AWS CLI

Contoh 1: Untuk daftar tag sumber daya untuk penyimpanan data

Contoh `list-tags-for-resource` kode berikut mencantumkan tag untuk penyimpanan data.

```
aws medical-imaging list-tags-for-resource \
    --resource-arn "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012"
```

Output:

```
{
```

```
    "tags":{
        "Deployment":"Development"
    }
}
```

Contoh 2: Untuk mencantumkan tag sumber daya untuk kumpulan gambar

Contoh `list-tags-for-resource` kode berikut mencantumkan tag untuk kumpulan gambar.

```
aws medical-imaging list-tags-for-resource \
    --resource-arn "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/18f88ac7870584f58d56256646b4d92b"
```

Output:

```
{
    "tags":{
        "Deployment":"Development"
    }
}
```

Untuk informasi selengkapnya, lihat Menandai sumber daya AWS HealthImaging di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

```
   public static ListTagsForResourceResponse
listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
           String resourceArn) {
       try {
           ListTagsForResourceRequest listTagsForResourceRequest =
ListTagsForResourceRequest.builder()
                   .resourceArn(resourceArn)
                   .build();
```

```
            return
 medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

- Untuk detail API, lihat ListTagsForResourcedi Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

```
import { ListTagsForResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data
 store or image set.
 */
export const listTagsForResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:abc:datastore/def/imageset/
ghi",
) => {
  const response = await medicalImagingClient.send(
    new ListTagsForResourceCommand({ resourceArn: resourceArn }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: '008fc6d3-abec-4870-a155-20fa3631e645',
```

```
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     tags: { Deployment: 'Development' }
  // }

  return response;
};
```

- Untuk detail API, lihat ListTagsForResourcedi Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def list_tags_for_resource(self, resource_arn):
        """
        List the tags for a resource.

        :param resource_arn: The ARN of the resource.
        :return: The list of tags.
        """
        try:
            tags = self.health_imaging_client.list_tags_for_resource(
                resourceArn=resource_arn
            )
        except ClientError as err:
            logger.error(
```

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

Kode berikut membuat instance objek. MedicalImagingWrapper

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- 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

```
    TRY.
        " iv_resource_arn = 'arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012'
        oo_result = lo_mig->listtagsforresource( iv_resourcearn =
 iv_resource_arn ).
        DATA(lt_tags) = oo_result->get_tags( ).
        DATA(lv_count) = lines( lt_tags ).
        MESSAGE |Found { lv_count } tags for resource.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
```

```
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

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

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

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

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

Contoh kode berikut menunjukkan cara menggunakan`SearchImageSets`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- Memulai dengan set gambar dan bingkai gambar

C++

SDK untuk C++

Fungsi utilitas untuk mencari set gambar.

```
//! Routine which searches for image sets based on defined input attributes.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param searchCriteria: A search criteria instance.
  \param imageSetResults: Vector to receive the image set IDs.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
```

```
  */
bool AwsDoc::Medical_Imaging::searchImageSets(const Aws::String &dataStoreID,
                                              const
 Aws::MedicalImaging::Model::SearchCriteria &searchCriteria,
                                              Aws::Vector<Aws::String>
 &imageSetResults,
                                              const
 Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::SearchImageSetsRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetSearchCriteria(searchCriteria);

    Aws::String nextToken; // Used for paginated results.
    bool result = true;
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::MedicalImaging::Model::SearchImageSetsOutcome outcome =
 client.SearchImageSets(
                request);
        if (outcome.IsSuccess()) {
            for (auto &imageSetMetadataSummary:
 outcome.GetResult().GetImageSetsMetadataSummaries()) {

 imageSetResults.push_back(imageSetMetadataSummary.GetImageSetId());
            }

            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cout << "Error: " << outcome.GetError().GetMessage() <<
 std::endl;
            result = false;
        }
    } while (!nextToken.empty());

    return result;
}
```

Kasus penggunaan #1: operator EQUAL.

```
       Aws::Vector<Aws::String> imageIDsForPatientID;
       Aws::MedicalImaging::Model::SearchCriteria searchCriteriaEqualsPatientID;
       Aws::Vector<Aws::MedicalImaging::Model::SearchFilter>
patientIDSearchFilters = {

Aws::MedicalImaging::Model::SearchFilter().WithOperator(Aws::MedicalImaging::Model::Oper

.WithValues({Aws::MedicalImaging::Model::SearchByAttributeValue().WithDICOMPatientId(pat
       };

       searchCriteriaEqualsPatientID.SetFilters(patientIDSearchFilters);
       bool result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,

searchCriteriaEqualsPatientID,

imageIDsForPatientID,
                                                              clientConfig);
       if (result) {
           std::cout << imageIDsForPatientID.size() << " image sets found for
the patient with ID '"
           <<  patientID << "'." << std::endl;
           for (auto &imageSetResult : imageIDsForPatientID) {
               std::cout << "  Image set with ID '" << imageSetResult <<
std::endl;
           }
       }
```

Kasus penggunaan #2: ANTARA operator menggunakan DICOMStudy Tanggal dan DICOMStudy Waktu.

```
        Aws::MedicalImaging::Model::SearchByAttributeValue useCase2StartDate;

useCase2StartDate.SetDICOMStudyDateAndTime(Aws::MedicalImaging::Model::DICOMStudyDateAnd
       .WithDICOMStudyDate("19990101")
       .WithDICOMStudyTime("000000.000"));

       Aws::MedicalImaging::Model::SearchByAttributeValue useCase2EndDate;

useCase2EndDate.SetDICOMStudyDateAndTime(Aws::MedicalImaging::Model::DICOMStudyDateAndTi
```

```
 .WithDICOMStudyDate(Aws::Utils::DateTime(std::chrono::system_clock::now()).ToLocalTimeSt
%m%d"))
        .WithDICOMStudyTime("000000.000"));

       Aws::MedicalImaging::Model::SearchFilter useCase2SearchFilter;
       useCase2SearchFilter.SetValues({useCase2StartDate, useCase2EndDate});

 useCase2SearchFilter.SetOperator(Aws::MedicalImaging::Model::Operator::BETWEEN);

       Aws::MedicalImaging::Model::SearchCriteria useCase2SearchCriteria;
       useCase2SearchCriteria.SetFilters({useCase2SearchFilter});

       Aws::Vector<Aws::String> usesCase2Results;
       result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,
                                                         useCase2SearchCriteria,
                                                         usesCase2Results,
                                                         clientConfig);
       if (result) {
           std::cout << usesCase2Results.size() << " image sets found for
 between 1999/01/01 and present."
                     <<  std::endl;
           for (auto &imageSetResult : usesCase2Results) {
               std::cout << "  Image set with ID '" << imageSetResult <<
 std::endl;
           }
       }
```

Kasus penggunaan #3: ANTARA operator menggunakan createDat. Studi waktu sebelumnya bertahan.

```
       Aws::MedicalImaging::Model::SearchByAttributeValue useCase3StartDate;

 useCase3StartDate.SetCreatedAt(Aws::Utils::DateTime("20231130T000000000Z",Aws::Utils::Da

       Aws::MedicalImaging::Model::SearchByAttributeValue useCase3EndDate;

 useCase3EndDate.SetCreatedAt(Aws::Utils::DateTime(std::chrono::system_clock::now()));

       Aws::MedicalImaging::Model::SearchFilter useCase3SearchFilter;
       useCase3SearchFilter.SetValues({useCase3StartDate, useCase3EndDate});
```

```
useCase3SearchFilter.SetOperator(Aws::MedicalImaging::Model::Operator::BETWEEN);

       Aws::MedicalImaging::Model::SearchCriteria useCase3SearchCriteria;
       useCase3SearchCriteria.SetFilters({useCase3SearchFilter});

       Aws::Vector<Aws::String> usesCase3Results;
       result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,
                                                         useCase3SearchCriteria,
                                                         usesCase3Results,
                                                         clientConfig);
       if (result) {
           std::cout << usesCase3Results.size() << " image sets found for
created between 2023/11/30 and present."
                     <<  std::endl;
           for (auto &imageSetResult : usesCase3Results) {
               std::cout << "  Image set with ID '" << imageSetResult <<
std::endl;
           }
       }
```

Kasus penggunaan #4: Operator EQUAL di DICOMSeries InstanceUID dan BETHER di UpdateDAT dan mengurutkan respons dalam urutan ASC di bidang UpdateDAT.

```
       Aws::MedicalImaging::Model::SearchByAttributeValue useCase4StartDate;

useCase4StartDate.SetUpdatedAt(Aws::Utils::DateTime("20231130T000000000Z",Aws::Utils::Da

       Aws::MedicalImaging::Model::SearchByAttributeValue useCase4EndDate;

useCase4EndDate.SetUpdatedAt(Aws::Utils::DateTime(std::chrono::system_clock::now()));

       Aws::MedicalImaging::Model::SearchFilter useCase4SearchFilterBetween;
       useCase4SearchFilterBetween.SetValues({useCase4StartDate,
useCase4EndDate});

useCase4SearchFilterBetween.SetOperator(Aws::MedicalImaging::Model::Operator::BETWEEN);

       Aws::MedicalImaging::Model::SearchByAttributeValue seriesInstanceUID;
       seriesInstanceUID.SetDICOMSeriesInstanceUID(dicomSeriesInstanceUID);

       Aws::MedicalImaging::Model::SearchFilter useCase4SearchFilterEqual;
```

```
        useCase4SearchFilterEqual.SetValues({seriesInstanceUID});

useCase4SearchFilterEqual.SetOperator(Aws::MedicalImaging::Model::Operator::EQUAL);

        Aws::MedicalImaging::Model::SearchCriteria useCase4SearchCriteria;
        useCase4SearchCriteria.SetFilters({useCase4SearchFilterBetween,
useCase4SearchFilterEqual});

        Aws::MedicalImaging::Model::Sort useCase4Sort;

useCase4Sort.SetSortField(Aws::MedicalImaging::Model::SortField::updatedAt);
        useCase4Sort.SetSortOrder(Aws::MedicalImaging::Model::SortOrder::ASC);

        useCase4SearchCriteria.SetSort(useCase4Sort);

        Aws::Vector<Aws::String> usesCase4Results;
        result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,
                                                          useCase4SearchCriteria,
                                                          usesCase4Results,
                                                          clientConfig);
        if (result) {
            std::cout << usesCase4Results.size() << " image sets found for EQUAL
operator "
            << "on DICOMSeriesInstanceUID and BETWEEN on updatedAt and sort
response\n"
            <<  "in ASC order on updatedAt field." <<  std::endl;
            for (auto &imageSetResult : usesCase4Results) {
                std::cout << "  Image set with ID '" << imageSetResult <<
std::endl;
            }
        }
```

- Untuk detail API, lihat SearchImageSetsdi Referensi AWS SDK untuk C++ API.

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

CLI

AWS CLI

Contoh 1: Untuk mencari set gambar dengan operator EQUAL

Contoh `search-image-sets` kode berikut menggunakan operator EQUAL untuk mencari set gambar berdasarkan nilai tertentu.

```
aws medical-imaging search-image-sets \
    --datastore-id 12345678901234567890123456789012 \
    --search-criteria file://search-criteria.json
```

Isi dari `search-criteria.json`

```
{
    "filters": [{
        "values": [{"DICOMPatientId" : "SUBJECT08701"}],
        "operator": "EQUAL"
    }]
}
```

Output:

```
{
    "imageSetsMetadataSummaries": [{
        "imageSetId": "09876543210987654321098765432109",
        "createdAt": "2022-12-06T21:40:59.429000+00:00",
        "version": 1,
        "DICOMTags": {
            "DICOMStudyId": "2011201407",
            "DICOMStudyDate": "19991122",
             "DICOMPatientSex": "F",
             "DICOMStudyInstanceUID": "1.2.840.99999999.84710745.943275268089",
             "DICOMPatientBirthDate": "19201120",
             "DICOMStudyDescription": "UNKNOWN",
             "DICOMPatientId": "SUBJECT08701",
             "DICOMPatientName": "Melissa844 Huel628",
             "DICOMNumberOfStudyRelatedInstances": 1,
             "DICOMStudyTime": "140728",
             "DICOMNumberOfStudyRelatedSeries": 1
            },
```

```
        "updatedAt": "2022-12-06T21:40:59.429000+00:00"
    }]
}
```

Contoh 2: Untuk mencari set gambar dengan operator ANTARA menggunakan DICOMStudy Tanggal dan DICOMStudy Waktu

Contoh `search-image-sets` kode berikut mencari kumpulan gambar dengan Studi DICOM yang dihasilkan antara 1 Januari 1990 (12:00 AM) dan 1 Januari 2023 (12:00 AM).

Catatan: DICOMStudy Waktu adalah opsional. Jika tidak ada, 12:00 AM (awal hari) adalah nilai waktu untuk tanggal yang disediakan untuk penyaringan.

```
aws medical-imaging search-image-sets \
    --datastore-id 12345678901234567890123456789012 \
    --search-criteria file://search-criteria.json
```

Isi dari `search-criteria.json`

```
{
    "filters": [{
        "values": [{
            "DICOMStudyDateAndTime": {
                "DICOMStudyDate": "19900101",
                "DICOMStudyTime": "000000"
            }
        },
        {
            "DICOMStudyDateAndTime": {
                "DICOMStudyDate": "20230101",
                "DICOMStudyTime": "000000"
            }
        }],
        "operator": "BETWEEN"
    }]
}
```

Output:

```
{
    "imageSetsMetadataSummaries": [{
```

```
        "imageSetId": "09876543210987654321098765432109",
        "createdAt": "2022-12-06T21:40:59.429000+00:00",
        "version": 1,
        "DICOMTags": {
            "DICOMStudyId": "2011201407",
            "DICOMStudyDate": "19991122",
            "DICOMPatientSex": "F",
            "DICOMStudyInstanceUID": "1.2.840.99999999.84710745.943275268089",
            "DICOMPatientBirthDate": "19201120",
            "DICOMStudyDescription": "UNKNOWN",
            "DICOMPatientId": "SUBJECT08701",
            "DICOMPatientName": "Melissa844 Huel628",
            "DICOMNumberOfStudyRelatedInstances": 1,
            "DICOMStudyTime": "140728",
            "DICOMNumberOfStudyRelatedSeries": 1
        },
        "updatedAt": "2022-12-06T21:40:59.429000+00:00"
    }]
}
```

Contoh 3: Untuk mencari set gambar dengan operator ANTARA menggunakan createDat (studi waktu sebelumnya dipertahankan)

Contoh `search-image-sets` kode berikut mencari set gambar dengan Studi DICOM bertahan di HealthImaging antara rentang waktu di zona waktu UTC.

Catatan: Berikan CreateDat dalam format contoh (“1985-04-12T 23:20:50.52 Z”).

```
aws medical-imaging search-image-sets \
    --datastore-id 12345678901234567890123456789012 \
    --search-criteria  file://search-criteria.json
```

Isi dari `search-criteria.json`

```
{
    "filters": [{
        "values": [{
            "createdAt": "1985-04-12T23:20:50.52Z"
        },
        {
            "createdAt": "2022-04-12T23:20:50.52Z"
        }],
```

```
        "operator": "BETWEEN"
    }]
}
```

Output:

```
{
    "imageSetsMetadataSummaries": [{
        "imageSetId": "09876543210987654321098765432109",
        "createdAt": "2022-12-06T21:40:59.429000+00:00",
        "version": 1,
        "DICOMTags": {
            "DICOMStudyId": "2011201407",
            "DICOMStudyDate": "19991122",
            "DICOMPatientSex": "F",
            "DICOMStudyInstanceUID": "1.2.840.99999999.84710745.943275268089",
            "DICOMPatientBirthDate": "19201120",
            "DICOMStudyDescription": "UNKNOWN",
            "DICOMPatientId": "SUBJECT08701",
            "DICOMPatientName": "Melissa844 Huel628",
            "DICOMNumberOfStudyRelatedInstances": 1,
            "DICOMStudyTime": "140728",
            "DICOMNumberOfStudyRelatedSeries": 1
        },
        "lastUpdatedAt": "2022-12-06T21:40:59.429000+00:00"
    }]
}
```

Contoh 4: Untuk mencari set gambar dengan operator EQUAL di DICOMSeries InstanceUID dan BETHER pada UpdateDat dan mengurutkan respons dalam urutan ASC di bidang UpdateDAT

Contoh `search-image-sets` kode berikut mencari kumpulan gambar dengan operator EQUAL di DICOMSeries InstanceUID dan BETHER pada UpdateDat dan mengurutkan respons dalam urutan ASC di bidang UpdateDAT.

Catatan: Berikan UpdateDat dalam format contoh (“1985-04-12T 23:20:50.52 Z”).

```
aws medical-imaging search-image-sets \
    --datastore-id 12345678901234567890123456789012 \
    --search-criteria  file://search-criteria.json
```

Isi dari `search-criteria.json`

```
{
    "filters": [{
        "values": [{
            "updatedAt": "2024-03-11T15:00:05.074000-07:00"
        }, {
            "updatedAt": "2024-03-11T16:00:05.074000-07:00"
        }],
        "operator": "BETWEEN"
    }, {
        "values": [{
            "DICOMSeriesInstanceUID": "1.2.840.99999999.84710745.943275268089"
        }],
        "operator": "EQUAL"
    }],
    "sort": {
        "sortField": "updatedAt",
        "sortOrder": "ASC"
    }
}
```

Output:

```
{
    "imageSetsMetadataSummaries": [{
        "imageSetId": "09876543210987654321098765432109",
        "createdAt": "2022-12-06T21:40:59.429000+00:00",
        "version": 1,
        "DICOMTags": {
            "DICOMStudyId": "2011201407",
            "DICOMStudyDate": "19991122",
            "DICOMPatientSex": "F",
            "DICOMStudyInstanceUID": "1.2.840.99999999.84710745.943275268089",
            "DICOMPatientBirthDate": "19201120",
            "DICOMStudyDescription": "UNKNOWN",
            "DICOMPatientId": "SUBJECT08701",
            "DICOMPatientName": "Melissa844 Huel628",
            "DICOMNumberOfStudyRelatedInstances": 1,
            "DICOMStudyTime": "140728",
            "DICOMNumberOfStudyRelatedSeries": 1
        },
        "lastUpdatedAt": "2022-12-06T21:40:59.429000+00:00"
```

```
    }]
}
```

Untuk informasi selengkapnya, lihat Mencari kumpulan gambar di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

Fungsi utilitas untuk mencari set gambar.

```
   public static List<ImageSetsMetadataSummary> searchMedicalImagingImageSets(
           MedicalImagingClient medicalImagingClient,
           String datastoreId, SearchCriteria searchCriteria) {
       try {
           SearchImageSetsRequest datastoreRequest =
SearchImageSetsRequest.builder()
                   .datastoreId(datastoreId)
                   .searchCriteria(searchCriteria)
                   .build();
           SearchImageSetsIterable responses = medicalImagingClient
                   .searchImageSetsPaginator(datastoreRequest);
           List<ImageSetsMetadataSummary> imageSetsMetadataSummaries = new
ArrayList<>();

           responses.stream().forEach(response -> imageSetsMetadataSummaries
                   .addAll(response.imageSetsMetadataSummaries()));

           return imageSetsMetadataSummaries;
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return null;
   }
```

Kasus penggunaan #1: operator EQUAL.

```
        List<SearchFilter> searchFilters =
 Collections.singletonList(SearchFilter.builder()
                .operator(Operator.EQUAL)
                .values(SearchByAttributeValue.builder()
                        .dicomPatientId(patientId)
                        .build())
                .build());

        SearchCriteria searchCriteria = SearchCriteria.builder()
                .filters(searchFilters)
                .build();

        List<ImageSetsMetadataSummary> imageSetsMetadataSummaries =
 searchMedicalImagingImageSets(
                medicalImagingClient,
                datastoreId, searchCriteria);
        if (imageSetsMetadataSummaries != null) {
            System.out.println("The image sets for patient " + patientId + " are:
\n"
                    + imageSetsMetadataSummaries);
            System.out.println();
        }
```

Kasus penggunaan #2: ANTARA operator menggunakan DICOMStudy Tanggal dan DICOMStudy Waktu.

```
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyyMMdd");
        searchFilters = Collections.singletonList(SearchFilter.builder()
                .operator(Operator.BETWEEN)
                .values(SearchByAttributeValue.builder()

 .dicomStudyDateAndTime(DICOMStudyDateAndTime.builder()
                                        .dicomStudyDate("19990101")
                                        .dicomStudyTime("000000.000")
                                        .build())
                                .build(),
                        SearchByAttributeValue.builder()

 .dicomStudyDateAndTime(DICOMStudyDateAndTime.builder()
                                        .dicomStudyDate((LocalDate.now()
                                                .format(formatter)))
                                        .dicomStudyTime("000000.000")
```

```
                                        .build())
                                .build())
                .build());

        searchCriteria = SearchCriteria.builder()
                .filters(searchFilters)
                .build();

        imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
                datastoreId, searchCriteria);
        if (imageSetsMetadataSummaries != null) {
            System.out.println(
                    "The image sets searched with BETWEEN operator using
DICOMStudyDate and DICOMStudyTime are:\n"
                            +
                            imageSetsMetadataSummaries);
            System.out.println();
        }
```

Kasus penggunaan #3: ANTARA operator menggunakan createDat. Studi waktu sebelumnya bertahan.

```
        searchFilters = Collections.singletonList(SearchFilter.builder()
                .operator(Operator.BETWEEN)
                .values(SearchByAttributeValue.builder()

.createdAt(Instant.parse("1985-04-12T23:20:50.52Z"))
                                .build(),
                        SearchByAttributeValue.builder()
                                .createdAt(Instant.now())
                                .build())
                .build());

        searchCriteria = SearchCriteria.builder()
                .filters(searchFilters)
                .build();
        imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
                datastoreId, searchCriteria);
        if (imageSetsMetadataSummaries != null) {
```

```
            System.out.println("The image sets searched with BETWEEN operator
using createdAt are:\n "
                    + imageSetsMetadataSummaries);
            System.out.println();
        }
```

Kasus penggunaan #4: Operator EQUAL di DICOMSeries InstanceUID dan BETHER di UpdateDAT dan mengurutkan respons dalam urutan ASC di bidang UpdateDAT.

```
        Instant startDate = Instant.parse("1985-04-12T23:20:50.52Z");
        Instant endDate = Instant.now();

        searchFilters = Arrays.asList(
                SearchFilter.builder()
                        .operator(Operator.EQUAL)
                        .values(SearchByAttributeValue.builder()
                                .dicomSeriesInstanceUID(seriesInstanceUID)
                                .build())
                        .build(),
                SearchFilter.builder()
                        .operator(Operator.BETWEEN)
                        .values(

SearchByAttributeValue.builder().updatedAt(startDate).build(),

SearchByAttributeValue.builder().updatedAt(endDate).build()
                        ).build());

        Sort sort =
Sort.builder().sortOrder(SortOrder.ASC).sortField(SortField.UPDATED_AT).build();

        searchCriteria = SearchCriteria.builder()
                .filters(searchFilters)
                .sort(sort)
                .build();

        imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
                datastoreId, searchCriteria);
        if (imageSetsMetadataSummaries != null) {
            System.out.println("The image sets searched with EQUAL operator on
DICOMSeriesInstanceUID and BETWEEN on updatedAt and sort response\n" +
```

```
                    "in ASC order on updatedAt field are:\n "
                    + imageSetsMetadataSummaries);
            System.out.println();
        }
```

- Untuk detail API, lihat SearchImageSetsdi Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

Fungsi utilitas untuk mencari set gambar.

```
import { paginateSearchImageSets } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The data store's ID.
 * @param { import('@aws-sdk/client-medical-imaging').SearchFilter[] } filters -
 The search criteria filters.
 * @param { import('@aws-sdk/client-medical-imaging').Sort } sort - The search
 criteria sort.
 */
export const searchImageSets = async (
  datastoreId = "xxxxxxxx",
  searchCriteria = {},
) => {
  const paginatorConfig = {
    client: medicalImagingClient,
    pageSize: 50,
  };

  const commandParams = {
    datastoreId: datastoreId,
    searchCriteria: searchCriteria,
  };
```

```
  const paginator = paginateSearchImageSets(paginatorConfig, commandParams);

  const imageSetsMetadataSummaries = [];
  for await (const page of paginator) {
    // Each page contains a list of `jobSummaries`. The list is truncated if is
 larger than `pageSize`.
    imageSetsMetadataSummaries.push(...page.imageSetsMetadataSummaries);
    console.log(page);
  }
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: 'f009ea9c-84ca-4749-b5b6-7164f00a5ada',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     imageSetsMetadataSummaries: [
  //         {
  //             DICOMTags: [Object],
  //             createdAt: "2023-09-19T16:59:40.551Z",
  //             imageSetId: '7f75e1b5c0f40eac2b24cf712f485f50',
  //             updatedAt: "2023-09-19T16:59:40.551Z",
  //             version: 1
  //         }]
  // }

  return imageSetsMetadataSummaries;
};
```

Kasus penggunaan #1: operator EQUAL.

```
  const datastoreId = "12345678901234567890123456789012";

  try {
    const searchCriteria = {
      filters: [
        {
          values: [{ DICOMPatientId: "1234567" }],
          operator: "EQUAL",
```

```
      },
    ],
  };

  await searchImageSets(datastoreId, searchCriteria);
} catch (err) {
  console.error(err);
}
```

Kasus penggunaan #2: ANTARA operator menggunakan DICOMStudy Tanggal dan DICOMStudy Waktu.

```
const datastoreId = "12345678901234567890123456789012";

try {
  const searchCriteria = {
    filters: [
      {
        values: [
          {
            DICOMStudyDateAndTime: {
              DICOMStudyDate: "19900101",
              DICOMStudyTime: "000000",
            },
          },
          {
            DICOMStudyDateAndTime: {
              DICOMStudyDate: "20230901",
              DICOMStudyTime: "000000",
            },
          },
        ],
        operator: "BETWEEN",
      },
    ],
  };

  await searchImageSets(datastoreId, searchCriteria);
} catch (err) {
  console.error(err);
}
```

Kasus penggunaan #3: ANTARA operator menggunakan createDat. Studi waktu sebelumnya bertahan.

```
const datastoreId = "12345678901234567890123456789012";

try {
  const searchCriteria = {
    filters: [
      {
        values: [
          { createdAt: new Date("1985-04-12T23:20:50.52Z") },
          { createdAt: new Date() },
        ],
        operator: "BETWEEN",
      },
    ],
  };

  await searchImageSets(datastoreId, searchCriteria);
} catch (err) {
  console.error(err);
}
```

Kasus penggunaan #4: Operator EQUAL di DICOMSeries InstanceUID dan BETHER di UpdateDAT dan mengurutkan respons dalam urutan ASC di bidang UpdateDAT.

```
const datastoreId = "12345678901234567890123456789012";

try {
  const searchCriteria = {
    filters: [
      {
        values: [
          { updatedAt: new Date("1985-04-12T23:20:50.52Z") },
          { updatedAt: new Date() },
        ],
        operator: "BETWEEN",
      },
      {
        values: [
          {
            DICOMSeriesInstanceUID:
```

```
                "1.1.123.123456.1.12.1.1234567890.1234.12345678.123",
            },
          ],
          operator: "EQUAL",
        },
      ],
      sort: {
        sortOrder: "ASC",
        sortField: "updatedAt",
      },
    };

    await searchImageSets(datastoreId, searchCriteria);
  } catch (err) {
    console.error(err);
  }
```

- Untuk detail API, lihat SearchImageSetsdi Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

Fungsi utilitas untuk mencari set gambar.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def search_image_sets(self, datastore_id, search_filter):
        """
        Search for image sets.

        :param datastore_id: The ID of the data store.
        :param search_filter: The search filter.
```

```
            For example: {"filters" : [{ "operator": "EQUAL", "values":
[{"DICOMPatientId": "3524578"}]}]}.
        :return: The list of image sets.
        """
        try:
            paginator =
self.health_imaging_client.get_paginator("search_image_sets")
            page_iterator = paginator.paginate(
                datastoreId=datastore_id, searchCriteria=search_filter
            )
            metadata_summaries = []
            for page in page_iterator:
                metadata_summaries.extend(page["imageSetsMetadataSummaries"])
        except ClientError as err:
            logger.error(
                "Couldn't search image sets. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return metadata_summaries
```

Kasus penggunaan #1: operator EQUAL.

```
        search_filter = {
            "filters": [
                {"operator": "EQUAL", "values": [{"DICOMPatientId": patient_id}]}
            ]
        }

        image_sets = self.search_image_sets(data_store_id, search_filter)
        print(f"Image sets found with EQUAL operator\n{image_sets}")
```

Kasus penggunaan #2: ANTARA operator menggunakan DICOMStudy Tanggal dan DICOMStudy Waktu.

```
        search_filter = {
            "filters": [
                {
```

```
                    "operator": "BETWEEN",
                    "values": [
                        {
                            "DICOMStudyDateAndTime": {
                                "DICOMStudyDate": "19900101",
                                "DICOMStudyTime": "000000",
                            }
                        },
                        {
                            "DICOMStudyDateAndTime": {
                                "DICOMStudyDate": "20230101",
                                "DICOMStudyTime": "000000",
                            }
                        },
                    ],
                }
            ]
        }

        image_sets = self.search_image_sets(data_store_id, search_filter)
        print(
            f"Image sets found with BETWEEN operator using DICOMStudyDate and
DICOMStudyTime\n{image_sets}"
        )
```

Kasus penggunaan #3: ANTARA operator menggunakan createDat. Studi waktu sebelumnya bertahan.

```
        search_filter = {
            "filters": [
                {
                    "values": [
                        {
                            "createdAt": datetime.datetime(
                                2021, 8, 4, 14, 49, 54, 429000
                            )
                        },
                        {
                            "createdAt": datetime.datetime.now()
                            + datetime.timedelta(days=1)
                        },
                    ],
```

```
                    "operator": "BETWEEN",
                }
            ]
        }

        recent_image_sets = self.search_image_sets(data_store_id, search_filter)
        print(
            f"Image sets found with with BETWEEN operator using createdAt
\n{recent_image_sets}"
        )
```

Kasus penggunaan #4: Operator EQUAL di DICOMSeries InstanceUID dan BETHER di UpdateDAT dan mengurutkan respons dalam urutan ASC di bidang UpdateDAT.

```
        search_filter = {
            "filters": [
                {
                    "values": [
                        {
                            "updatedAt": datetime.datetime(
                                2021, 8, 4, 14, 49, 54, 429000
                            )
                        },
                        {
                            "updatedAt": datetime.datetime.now()
                            + datetime.timedelta(days=1)
                        },
                    ],
                    "operator": "BETWEEN",
                },
                {
                    "values": [{"DICOMSeriesInstanceUID": series_instance_uid}],
                    "operator": "EQUAL",
                },
            ],
            "sort": {
                "sortOrder": "ASC",
                "sortField": "updatedAt",
            },
        }

        image_sets = self.search_image_sets(data_store_id, search_filter)
```

```
        print(
            "Image sets found with EQUAL operator on DICOMSeriesInstanceUID and
BETWEEN on updatedAt and"
        )
        print(f"sort response in ASC order on updatedAt field\n{image_sets}")
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Untuk detail API, lihat SearchImageSetsdi 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

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678901234567890'
        oo_result = lo_mig->searchimagesets(
          iv_datastoreid = iv_datastore_id
          io_searchcriteria = io_search_criteria ).
        DATA(lt_imagesets) = oo_result->get_imagesetsmetadatasums( ).
        DATA(lv_count) = lines( lt_imagesets ).
        MESSAGE |Found { lv_count } image sets.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
```

```
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

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

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

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

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

Contoh kode berikut menunjukkan cara menggunakan`StartDICOMImportJob`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- Memulai dengan set gambar dan bingkai gambar

C++

SDK untuk C++

```
//! Routine which starts a HealthImaging import job.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param inputBucketName: The name of the Amazon S3 bucket containing the DICOM
 files.
  \param inputDirectory: The directory in the S3 bucket containing the DICOM
 files.
  \param outputBucketName: The name of the S3 bucket for the output.
  \param outputDirectory: The directory in the S3 bucket to store the output.
```

```
  \param roleArn: The ARN of the IAM role with permissions for the import.
  \param importJobId: A string to receive the import job ID.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
  */
bool AwsDoc::Medical_Imaging::startDICOMImportJob(
        const Aws::String &dataStoreID, const Aws::String &inputBucketName,
        const Aws::String &inputDirectory, const Aws::String &outputBucketName,
        const Aws::String &outputDirectory, const Aws::String &roleArn,
        Aws::String &importJobId,
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient medicalImagingClient(clientConfig);
    Aws::String inputURI = "s3://" + inputBucketName + "/" + inputDirectory +
 "/";
    Aws::String outputURI = "s3://" + outputBucketName + "/" + outputDirectory +
 "/";
    Aws::MedicalImaging::Model::StartDICOMImportJobRequest
 startDICOMImportJobRequest;
    startDICOMImportJobRequest.SetDatastoreId(dataStoreID);
    startDICOMImportJobRequest.SetDataAccessRoleArn(roleArn);
    startDICOMImportJobRequest.SetInputS3Uri(inputURI);
    startDICOMImportJobRequest.SetOutputS3Uri(outputURI);

    Aws::MedicalImaging::Model::StartDICOMImportJobOutcome
 startDICOMImportJobOutcome = medicalImagingClient.StartDICOMImportJob(
            startDICOMImportJobRequest);

    if (startDICOMImportJobOutcome.IsSuccess()) {
        importJobId = startDICOMImportJobOutcome.GetResult().GetJobId();
    }
    else {
        std::cerr << "Failed to start DICOM import job because "
                  << startDICOMImportJobOutcome.GetError().GetMessage() <<
 std::endl;
    }

    return startDICOMImportJobOutcome.IsSuccess();
}
```

- Untuk detail API, lihat Memulai DICOMImport Job di Referensi AWS SDK untuk C++ API.

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

CLI

AWS CLI

Untuk memulai pekerjaan impor dicom

Contoh `start-dicom-import-job` kode berikut memulai pekerjaan impor dicom.

```
aws medical-imaging start-dicom-import-job \
    --job-name "my-job" \
    --datastore-id "12345678901234567890123456789012" \
    --input-s3-uri "s3://medical-imaging-dicom-input/dicom_input/" \
    --output-s3-uri "s3://medical-imaging-output/job_output/" \
    --data-access-role-arn "arn:aws:iam::123456789012:role/
ImportJobDataAccessRole"
```

Output:

```
{
    "datastoreId": "12345678901234567890123456789012",
    "jobId": "09876543210987654321098765432109",
    "jobStatus": "SUBMITTED",
    "submittedAt": "2022-08-12T11:28:11.152000+00:00"
}
```

Untuk informasi selengkapnya, lihat Memulai pekerjaan impor di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

```
   public static String startDicomImportJob(MedicalImagingClient
medicalImagingClient,
           String jobName,
           String datastoreId,
           String dataAccessRoleArn,
           String inputS3Uri,
           String outputS3Uri) {

       try {
           StartDicomImportJobRequest startDicomImportJobRequest =
StartDicomImportJobRequest.builder()
                   .jobName(jobName)
                   .datastoreId(datastoreId)
                   .dataAccessRoleArn(dataAccessRoleArn)
                   .inputS3Uri(inputS3Uri)
                   .outputS3Uri(outputS3Uri)
                   .build();
           StartDicomImportJobResponse response =
medicalImagingClient.startDICOMImportJob(startDicomImportJobRequest);
           return response.jobId();
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return "";
   }
```

- Untuk detail API, lihat Memulai DICOMImport Job di Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

```
import { StartDICOMImportJobCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} jobName - The name of the import job.
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} dataAccessRoleArn - The Amazon Resource Name (ARN) of the role
 that grants permission.
 * @param {string} inputS3Uri - The URI of the S3 bucket containing the input
 files.
 * @param {string} outputS3Uri - The URI of the S3 bucket where the output files
 are stored.
 */
export const startDicomImportJob = async (
  jobName = "test-1",
  datastoreId = "12345678901234567890123456789012",
  dataAccessRoleArn = "arn:aws:iam::xxxxxxxxxxxx:role/ImportJobDataAccessRole",
  inputS3Uri = "s3://medical-imaging-dicom-input/dicom_input/",
  outputS3Uri = "s3://medical-imaging-output/job_output/",
) => {
  const response = await medicalImagingClient.send(
    new StartDICOMImportJobCommand({
      jobName: jobName,
      datastoreId: datastoreId,
      dataAccessRoleArn: dataAccessRoleArn,
      inputS3Uri: inputS3Uri,
      outputS3Uri: outputS3Uri,
    }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //     httpStatusCode: 200,
  //         requestId: '6e81d191-d46b-4e48-a08a-cdcc7e11eb79',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  // },
  //     datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
```

```
  //     jobId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //     jobStatus: 'SUBMITTED',
  //     submittedAt: 2023-09-22T14:48:45.767Z
  // }
  return response;
};
```

- Untuk detail API, lihat Memulai DICOMImport Job di Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def start_dicom_import_job(
        self, job_name, datastore_id, role_arn, input_s3_uri, output_s3_uri
    ):
        """
        Start a DICOM import job.

        :param job_name: The name of the job.
        :param datastore_id: The ID of the data store.
        :param role_arn: The Amazon Resource Name (ARN) of the role to use for
 the job.
        :param input_s3_uri: The S3 bucket input prefix path containing the DICOM
 files.
        :param output_s3_uri: The S3 bucket output prefix path for the result.
        :return: The job ID.
        """
        try:
```

```
            job = self.health_imaging_client.start_dicom_import_job(
                jobName=job_name,
                datastoreId=datastore_id,
                dataAccessRoleArn=role_arn,
                inputS3Uri=input_s3_uri,
                outputS3Uri=output_s3_uri,
            )
        except ClientError as err:
            logger.error(
                "Couldn't start DICOM import job. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return job["jobId"]
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
client = boto3.client("medical-imaging")
medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Untuk detail API, lihat Memulai DICOMImport 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

```
TRY.
    " iv_job_name = 'import-job-1'
    " iv_datastore_id = '12345678901234567890123456789012345678901234567890'
```

```
        " iv_role_arn = 'arn:aws:iam::123456789012:role/ImportJobRole'
        " iv_input_s3_uri = 's3://my-bucket/input/'
        " iv_output_s3_uri = 's3://my-bucket/output/'
        oo_result = lo_mig->startdicomimportjob(
          iv_jobname = iv_job_name
          iv_datastoreid = iv_datastore_id
          iv_dataaccessrolearn = iv_role_arn
          iv_inputs3uri = iv_input_s3_uri
          iv_outputs3uri = iv_output_s3_uri ).
        DATA(lv_job_id) = oo_result->get_jobid( ).
        MESSAGE |DICOM import job started with ID: { lv_job_id }.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
      CATCH /aws1/cx_migservicequotaexcdex.
        MESSAGE 'Service quota exceeded.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

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

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

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan layanan ini 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`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- Menandai penyimpanan data
- Menandai set gambar

CLI

AWS CLI

Contoh 1: Untuk menandai penyimpanan data

Contoh `tag-resource` kode berikut menandai penyimpanan data.

```
aws medical-imaging tag-resource \
  --resource-arn "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012" \
  --tags '{"Deployment":"Development"}'
```

Perintah ini tidak menghasilkan output.

Contoh 2: Untuk menandai set gambar

Contoh `tag-resource` kode berikut menandai set gambar.

```
aws medical-imaging tag-resource \
    --resource-arn "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/18f88ac7870584f58d56256646b4d92b" \
    --tags '{"Deployment":"Development"}'
```

Perintah ini tidak menghasilkan output.

Untuk informasi selengkapnya, lihat Menandai sumber daya AWS HealthImaging di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

```
   public static void tagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
           String resourceArn,
           Map<String, String> tags) {
       try {
           TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                   .resourceArn(resourceArn)
                   .tags(tags)
                   .build();

           medicalImagingClient.tagResource(tagResourceRequest);

           System.out.println("Tags have been added to the resource.");
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Untuk detail API, lihat TagResourcedi Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

```
import { TagResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data
 store or image set.
```

```
 * @param {Record<string,string>} tags - The tags to add to the resource as JSON.
 *                     - For example: {"Deployment" : "Development"}
 */
export const tagResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:xxxxxx:datastore/xxxxx/
imageset/xxx",
  tags = {},
) => {
  const response = await medicalImagingClient.send(
    new TagResourceCommand({ resourceArn: resourceArn, tags: tags }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //        httpStatusCode: 204,
  //         requestId: '8a6de9a3-ec8e-47ef-8643-473518b19d45',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //    }
  // }

  return response;
};
```

- Untuk detail API, lihat TagResourcedi Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client
```

```
    def tag_resource(self, resource_arn, tags):
        """
        Tag a resource.

        :param resource_arn: The ARN of the resource.
        :param tags: The tags to apply.
        """
        try:
            self.health_imaging_client.tag_resource(resourceArn=resource_arn,
tags=tags)
        except ClientError as err:
            logger.error(
                "Couldn't tag resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
client = boto3.client("medical-imaging")
medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- 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

```
TRY.
```

```
        " iv_resource_arn = 'arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012'
        lo_mig->tagresource(
          iv_resourcearn = iv_resource_arn
          it_tags = it_tags ).
        MESSAGE 'Resource tagged successfully.' TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

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

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

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan layanan ini 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`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- Menandai penyimpanan data
- Menandai set gambar

CLI

AWS CLI

Contoh 1: Untuk menghapus tag penyimpanan data

Contoh `untag-resource` kode berikut untags penyimpanan data.

```
aws medical-imaging untag-resource \
    --resource-arn "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012" \
    --tag-keys '["Deployment"]'
```

Perintah ini tidak menghasilkan output.

Contoh 2: Untuk menghapus tag set gambar

Contoh `untag-resource` kode berikut untag set gambar.

```
aws medical-imaging untag-resource \
    --resource-arn "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/18f88ac7870584f58d56256646b4d92b" \
    --tag-keys '["Deployment"]'
```

Perintah ini tidak menghasilkan output.

Untuk informasi selengkapnya, lihat Menandai sumber daya AWS HealthImaging di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

```
    public static void untagMedicalImagingResource(MedicalImagingClient
 medicalImagingClient,
            String resourceArn,
            Collection<String> tagKeys) {
        try {
```

```
            UntagResourceRequest untagResourceRequest =
 UntagResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .tagKeys(tagKeys)
                    .build();

            medicalImagingClient.untagResource(untagResourceRequest);

            System.out.println("Tags have been removed from the resource.");
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Untuk detail API, lihat UntagResourcedi Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

```
import { UntagResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data
 store or image set.
 * @param {string[]} tagKeys - The keys of the tags to remove.
 */
export const untagResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:xxxxxx:datastore/xxxxx/
imageset/xxx",
  tagKeys = [],
) => {
  const response = await medicalImagingClient.send(
```

```
    new UntagResourceCommand({ resourceArn: resourceArn, tagKeys: tagKeys }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //        httpStatusCode: 204,
  //         requestId: '8a6de9a3-ec8e-47ef-8643-473518b19d45',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //    }
  // }

  return response;
};
```

- Untuk detail API, lihat UntagResourcedi Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def untag_resource(self, resource_arn, tag_keys):
        """
        Untag a resource.

        :param resource_arn: The ARN of the resource.
        :param tag_keys: The tag keys to remove.
        """
```

```
        try:
            self.health_imaging_client.untag_resource(
                resourceArn=resource_arn, tagKeys=tag_keys
            )
        except ClientError as err:
            logger.error(
                "Couldn't untag resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- 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

```
    TRY.
        " iv_resource_arn = 'arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012'
        lo_mig->untagresource(
          iv_resourcearn = iv_resource_arn
          it_tagkeys = it_tag_keys ).
        MESSAGE 'Resource untagged successfully.' TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
```

```
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

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

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

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

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

Contoh kode berikut menunjukkan cara menggunakan`UpdateImageSetMetadata`.

CLI

AWS CLI

Contoh 1: Untuk menyisipkan atau memperbarui atribut dalam metadata set gambar

`update-image-set-metadata`Contoh berikut menyisipkan atau memperbarui atribut dalam metadata set gambar.

```
aws medical-imaging update-image-set-metadata \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id ea92b0d8838c72a3f25d00d13616f87e \
    --latest-version-id 1 \
    --cli-binary-format raw-in-base64-out \
    --update-image-set-metadata-updates file://metadata-updates.json
```

Isi dari `metadata-updates.json`

```
{
    "DICOMUpdates": {
        "updatableAttributes": "{\"SchemaVersion\":1.1,\"Patient\":{\"DICOM\":
{\"PatientName\":\"MX^MX\"}}}"
    }
}
```

Output:

```
{
    "latestVersionId": "2",
    "imageSetWorkflowStatus": "UPDATING",
    "updatedAt": 1680042257.908,
    "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
    "imageSetState": "LOCKED",
    "createdAt": 1680027126.436,
    "datastoreId": "12345678901234567890123456789012"
}
```

Contoh 2: Untuk menghapus atribut dari metadata set gambar

`update-image-set-metadata`Contoh berikut menghapus atribut dari metadata set gambar.

```
aws medical-imaging update-image-set-metadata \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id ea92b0d8838c72a3f25d00d13616f87e \
    --latest-version-id 1 \
    --cli-binary-format raw-in-base64-out \
    --update-image-set-metadata-updates file://metadata-updates.json
```

Isi dari `metadata-updates.json`

```
{
    "DICOMUpdates": {
        "removableAttributes": "{\"SchemaVersion\":1.1,\"Study\":{\"DICOM\":
{\"StudyDescription\":\"CHEST\"}}}"
    }
}
```

Output:

```
{
    "latestVersionId": "2",
    "imageSetWorkflowStatus": "UPDATING",
    "updatedAt": 1680042257.908,
    "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
    "imageSetState": "LOCKED",
    "createdAt": 1680027126.436,
    "datastoreId": "12345678901234567890123456789012"
}
```

Contoh 3: Untuk menghapus instance dari metadata set gambar

`update-image-set-metadata`Contoh berikut menghapus instance dari metadata set gambar.

```
aws medical-imaging update-image-set-metadata \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id ea92b0d8838c72a3f25d00d13616f87e \
    --latest-version-id 1 \
    --cli-binary-format raw-in-base64-out \
    --update-image-set-metadata-updates file://metadata-updates.json \
    --force
```

Isi dari `metadata-updates.json`

```
{
    "DICOMUpdates": {
        "removableAttributes": "{\"SchemaVersion\": 1.1,\"Study\": {\"Series
\": {\"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1\": {\"Instances\":
 {\"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1\": {}}}}}}"
    }
}
```

Output:

```
{
    "latestVersionId": "2",
    "imageSetWorkflowStatus": "UPDATING",
    "updatedAt": 1680042257.908,
    "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
    "imageSetState": "LOCKED",
    "createdAt": 1680027126.436,
```

```
    "datastoreId": "12345678901234567890123456789012"
}
```

Contoh 4: Untuk mengembalikan gambar yang disetel ke versi sebelumnya

`update-image-set-metadata`Contoh berikut menunjukkan cara mengembalikan gambar yang disetel ke versi sebelumnya. CopyImageSet dan UpdateImageSetMetadata tindakan membuat versi baru dari set gambar.

```
aws medical-imaging update-image-set-metadata \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id 53d5fdb05ca4d46ac7ca64b06545c66e \
    --latest-version-id 3 \
    --cli-binary-format raw-in-base64-out \
    --update-image-set-metadata-updates '{"revertToVersionId": "1"}'
```

Output:

```
{
    "datastoreId": "12345678901234567890123456789012",
    "imageSetId": "53d5fdb05ca4d46ac7ca64b06545c66e",
    "latestVersionId": "4",
    "imageSetState": "LOCKED",
    "imageSetWorkflowStatus": "UPDATING",
    "createdAt": 1680027126.436,
    "updatedAt": 1680042257.908
}
```

Contoh 5: Untuk menambahkan elemen data DICOM pribadi ke sebuah instance

`update-image-set-metadata`Contoh berikut menunjukkan bagaimana menambahkan elemen pribadi untuk contoh tertentu dalam set gambar. Standar DICOM memungkinkan elemen data pribadi untuk komunikasi informasi yang tidak dapat terkandung dalam elemen data standar. Anda dapat membuat, memperbarui, dan menghapus elemen data pribadi dengan UpdateImageSetMetadata tindakan.

```
aws medical-imaging update-image-set-metadata \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id 53d5fdb05ca4d46ac7ca64b06545c66e \
    --latest-version-id 1 \
    --cli-binary-format raw-in-base64-out \
    --force \
```

```
    --update-image-set-metadata-updates file://metadata-updates.json
```

Isi dari `metadata-updates.json`

```
{
    "DICOMUpdates": {
        "updatableAttributes": "{\"SchemaVersion\": 1.1,\"Study\": {\"Series
\": {\"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1\": {\"Instances
\": {\"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1\": {\"DICOM\":
 {\"001910F9\": \"97\"},\"DICOMVRs\": {\"001910F9\": \"DS\"}}}}}}"
    }
}
```

Output:

```
{
    "latestVersionId": "2",
    "imageSetWorkflowStatus": "UPDATING",
    "updatedAt": 1680042257.908,
    "imageSetId": "53d5fdb05ca4d46ac7ca64b06545c66e",
    "imageSetState": "LOCKED",
    "createdAt": 1680027126.436,
    "datastoreId": "12345678901234567890123456789012"
}
```

Contoh 6: Untuk memperbarui elemen data DICOM pribadi ke sebuah instance

`update-image-set-metadata`Contoh berikut menunjukkan cara memperbarui nilai elemen data pribadi milik sebuah instance dalam kumpulan gambar.

```
aws medical-imaging update-image-set-metadata \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id 53d5fdb05ca4d46ac7ca64b06545c66e \
    --latest-version-id 1 \
    --cli-binary-format raw-in-base64-out \
    --force \
    --update-image-set-metadata-updates file://metadata-updates.json
```

Isi dari `metadata-updates.json`

```
{
    "DICOMUpdates": {
```

```
        "updatableAttributes": "{\"SchemaVersion\": 1.1,\"Study\": {\"Series
\": {\"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1\": {\"Instances
\": {\"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1\": {\"DICOM\":
 {\"00091001\": \"GE_GENESIS_DD\"}}}}}}}"
    }
}
```

Output:

```
{
    "latestVersionId": "2",
    "imageSetWorkflowStatus": "UPDATING",
    "updatedAt": 1680042257.908,
    "imageSetId": "53d5fdb05ca4d46ac7ca64b06545c66e",
    "imageSetState": "LOCKED",
    "createdAt": 1680027126.436,
    "datastoreId": "12345678901234567890123456789012"
}
```

Contoh 7: Untuk memperbarui SOPInstance UID dengan parameter gaya

`update-image-set-metadata`Contoh berikut menunjukkan cara memperbarui SOPInstance UID, menggunakan parameter gaya untuk mengganti kendala metadata DICOM.

```
aws medical-imaging update-image-set-metadata \
        --datastore-id 12345678901234567890123456789012 \
        --image-set-id 53d5fdb05ca4d46ac7ca64b06545c66e \
        --latest-version-id 1 \
        --cli-binary-format raw-in-base64-out \
        --force \
        --update-image-set-metadata-updates file://metadata-updates.json
```

Isi dari `metadata-updates.json`

```
{
    "DICOMUpdates": {
        "updatableAttributes": "{\"SchemaVersion\":1.1,\"Study\":{\"Series
\":{\"1.3.6.1.4.1.5962.99.1.3633258862.2104868982.1369432891697.3656.0\":
{\"Instances\":
{\"1.3.6.1.4.1.5962.99.1.3633258862.2104868982.1369432891697.3659.0\":{\"DICOM\":
{\"SOPInstanceUID\":
\"1.3.6.1.4.1.5962.99.1.3633258862.2104868982.1369432891697.3659.9\"}}}}}}"
```

```
    }
}
```

Output:

```
{
    "latestVersionId": "2",
    "imageSetWorkflowStatus": "UPDATING",
    "updatedAt": 1680042257.908,
    "imageSetId": "53d5fdb05ca4d46ac7ca64b06545c66e",
    "imageSetState": "LOCKED",
    "createdAt": 1680027126.436,
    "datastoreId": "12345678901234567890123456789012"
}
```

Untuk informasi selengkapnya, lihat Memperbarui metadata set gambar di Panduan AWS HealthImaging Pengembang.

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

Java

SDK untuk Java 2.x

```
   /**
    * Update the metadata of an AWS HealthImaging image set.
    *
    * @param medicalImagingClient - The AWS HealthImaging client object.
    * @param datastoreId          - The datastore ID.
    * @param imageSetId           - The image set ID.
    * @param versionId            - The version ID.
    * @param metadataUpdates      - A MetadataUpdates object containing the
updates.
    * @param force                - The force flag.
    * @throws MedicalImagingException - Base exception for all service
exceptions thrown by AWS HealthImaging.
    */
   public static void updateMedicalImageSetMetadata(MedicalImagingClient
medicalImagingClient,
                                                    String datastoreId,
                                                    String imageSetId,
```

```
                                                    String versionId,
                                                    MetadataUpdates
metadataUpdates,
                                                    boolean force) {
        try {
            UpdateImageSetMetadataRequest updateImageSetMetadataRequest =
UpdateImageSetMetadataRequest
                    .builder()
                    .datastoreId(datastoreId)
                    .imageSetId(imageSetId)
                    .latestVersionId(versionId)
                    .updateImageSetMetadataUpdates(metadataUpdates)
                    .force(force)
                    .build();

            UpdateImageSetMetadataResponse response =
medicalImagingClient.updateImageSetMetadata(updateImageSetMetadataRequest);

            System.out.println("The image set metadata was updated" + response);
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            throw e;
        }
    }
```

Kasus penggunaan #1: Menyisipkan atau memperbarui atribut.

```
                final String insertAttributes = """
                        {
                          "SchemaVersion": 1.1,
                          "Study": {
                            "DICOM": {
                              "StudyDescription": "CT CHEST"
                            }
                          }
                        }
                        """;
                MetadataUpdates metadataInsertUpdates = MetadataUpdates.builder()
                        .dicomUpdates(DICOMUpdates.builder()
                                .updatableAttributes(SdkBytes.fromByteBuffer(
                                        ByteBuffer.wrap(insertAttributes
```

```
.getBytes(StandardCharsets.UTF_8))))
                                .build())
                        .build();

                updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
imagesetId,
                        versionid, metadataInsertUpdates, force);
```

Use case #2: Hapus atribut.

```
                final String removeAttributes = """
                        {
                          "SchemaVersion": 1.1,
                          "Study": {
                            "DICOM": {
                              "StudyDescription": "CT CHEST"
                            }
                          }
                        }
                        """;
                MetadataUpdates metadataRemoveUpdates = MetadataUpdates.builder()
                        .dicomUpdates(DICOMUpdates.builder()
                                .removableAttributes(SdkBytes.fromByteBuffer(
                                        ByteBuffer.wrap(removeAttributes

.getBytes(StandardCharsets.UTF_8))))
                                .build())
                        .build();

                updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
imagesetId,
                        versionid, metadataRemoveUpdates, force);
```

Use case #3: Hapus sebuah instance.

```
                final String removeInstance = """
                        {
                          "SchemaVersion": 1.1,
                          "Study": {
                            "Series": {
```

```
"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {
                                "Instances": {

"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {}
                                }
                              }
                            }
                          }
                        }
                        """;
                MetadataUpdates metadataRemoveUpdates = MetadataUpdates.builder()
                        .dicomUpdates(DICOMUpdates.builder()
                                .removableAttributes(SdkBytes.fromByteBuffer(
                                        ByteBuffer.wrap(removeInstance

.getBytes(StandardCharsets.UTF_8))))
                                .build())
                        .build();

                updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
imagesetId,
                        versionid, metadataRemoveUpdates, force);
```

Kasus penggunaan #4: Kembalikan ke versi sebelumnya.

```
                // In this case, revert to previous version.
                String revertVersionId =
Integer.toString(Integer.parseInt(versionid) - 1);
                MetadataUpdates metadataRemoveUpdates = MetadataUpdates.builder()
                        .revertToVersionId(revertVersionId)
                        .build();
                updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
imagesetId,
                        versionid, metadataRemoveUpdates, force);
```

- Untuk detail API, lihat UpdateImageSetMetadatadi Referensi AWS SDK for Java 2.x API.

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

JavaScript

SDK untuk JavaScript (v3)

```
import { UpdateImageSetMetadataCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the HealthImaging data store.
 * @param {string} imageSetId - The ID of the HealthImaging image set.
 * @param {string} latestVersionId - The ID of the HealthImaging image set
 version.
 * @param {{}} updateMetadata - The metadata to update.
 * @param {boolean} force - Force the update.
 */
export const updateImageSetMetadata = async (
  datastoreId = "xxxxxxxxxx",
  imageSetId = "xxxxxxxxxx",
  latestVersionId = "1",
  updateMetadata = "{}",
  force = false,
) => {
  try {
    const response = await medicalImagingClient.send(
      new UpdateImageSetMetadataCommand({
        datastoreId: datastoreId,
        imageSetId: imageSetId,
        latestVersionId: latestVersionId,
        updateImageSetMetadataUpdates: updateMetadata,
        force: force,
      }),
    );
    console.log(response);
    // {
    //     '$metadata': {
    //     httpStatusCode: 200,
```

```
    //         requestId: '7966e869-e311-4bff-92ec-56a61d3003ea',
    //         extendedRequestId: undefined,
    //         cfId: undefined,
    //         attempts: 1,
    //         totalRetryDelay: 0
    // },
    //     createdAt: 2023-09-22T14:49:26.427Z,
    //     datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
    //     imageSetId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
    //     imageSetState: 'LOCKED',
    //     imageSetWorkflowStatus: 'UPDATING',
    //     latestVersionId: '4',
    //     updatedAt: 2023-09-27T19:41:43.494Z
    // }
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

Kasus penggunaan #1: Menyisipkan atau memperbarui atribut dan memaksa pembaruan.

```
    const insertAttributes = JSON.stringify({
      SchemaVersion: 1.1,
      Study: {
        DICOM: {
          StudyDescription: "CT CHEST",
        },
      },
    });

    const updateMetadata = {
      DICOMUpdates: {
        updatableAttributes: new TextEncoder().encode(insertAttributes),
      },
    };

    await updateImageSetMetadata(
      datastoreID,
      imageSetID,
      versionID,
      updateMetadata,
```

```
  true,
);
```

Kasus penggunaan #2: Hapus atribut.

```
// Attribute key and value must match the existing attribute.
const remove_attribute = JSON.stringify({
  SchemaVersion: 1.1,
  Study: {
    DICOM: {
      StudyDescription: "CT CHEST",
    },
  },
});

const updateMetadata = {
  DICOMUpdates: {
    removableAttributes: new TextEncoder().encode(remove_attribute),
  },
};

await updateImageSetMetadata(
  datastoreID,
  imageSetID,
  versionID,
  updateMetadata,
);
```

Use case #3: Hapus sebuah instance.

```
const remove_instance = JSON.stringify({
  SchemaVersion: 1.1,
  Study: {
    Series: {
      "1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {
        Instances: {
          "1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {},
        },
      },
    },
  },
```

```
});

const updateMetadata = {
  DICOMUpdates: {
    removableAttributes: new TextEncoder().encode(remove_instance),
  },
};

await updateImageSetMetadata(
  datastoreID,
  imageSetID,
  versionID,
  updateMetadata,
);
```

Kasus penggunaan #4: Kembalikan ke versi sebelumnya.

```
const updateMetadata = {
  revertToVersionId: "1",
};

await updateImageSetMetadata(
  datastoreID,
  imageSetID,
  versionID,
  updateMetadata,
);
```

- Untuk detail API, lihat UpdateImageSetMetadatadi Referensi AWS SDK untuk JavaScript API.

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

Python

SDK untuk Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def update_image_set_metadata(
        self, datastore_id, image_set_id, version_id, metadata, force=False
    ):
        """
        Update the metadata of an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The ID of the image set version.
        :param metadata: The image set metadata as a dictionary.
            For example {"DICOMUpdates": {"updatableAttributes":
            "{\"SchemaVersion\":1.1,\"Patient\":{\"DICOM\":{\"PatientName\":
\"Garcia^Gloria\"}}}"}}
        :param: force: Force the update.
        :return: The updated image set metadata.
        """
        try:
            updated_metadata =
 self.health_imaging_client.update_image_set_metadata(
                imageSetId=image_set_id,
                datastoreId=datastore_id,
                latestVersionId=version_id,
                updateImageSetMetadataUpdates=metadata,
                force=force,
            )
        except ClientError as err:
            logger.error(
                "Couldn't update image set metadata. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return updated_metadata
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

Kasus penggunaan #1: Menyisipkan atau memperbarui atribut.

```
            attributes = """{
                    "SchemaVersion": 1.1,
                    "Study": {
                        "DICOM": {
                            "StudyDescription": "CT CHEST"
                        }
                    }
                }"""
            metadata = {"DICOMUpdates": {"updatableAttributes": attributes}}

            self.update_image_set_metadata(
                data_store_id, image_set_id, version_id, metadata, force
            )
```

Kasus penggunaan #2: Hapus atribut.

```
            # Attribute key and value must match the existing attribute.
            attributes = """{
                    "SchemaVersion": 1.1,
                    "Study": {
                        "DICOM": {
                            "StudyDescription": "CT CHEST"
                        }
                    }
                }"""
            metadata = {"DICOMUpdates": {"removableAttributes": attributes}}

            self.update_image_set_metadata(
                data_store_id, image_set_id, version_id, metadata, force
            )
```

Use case #3: Hapus sebuah instance.

```
            attributes = """{
                    "SchemaVersion": 1.1,
                    "Study": {
                        "Series": {

"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {
                                "Instances": {

"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {}
                                }
                            }
                        }
                    }
                }"""
            metadata = {"DICOMUpdates": {"removableAttributes": attributes}}

            self.update_image_set_metadata(
                data_store_id, image_set_id, version_id, metadata, force
            )
```

Kasus penggunaan #4: Kembalikan ke versi sebelumnya.

```
            metadata = {"revertToVersionId": "1"}

            self.update_image_set_metadata(
                data_store_id, image_set_id, version_id, metadata, force
            )
```

- Untuk detail API, lihat UpdateImageSetMetadatadi 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

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678901234567890'
        " iv_image_set_id = '12345678901234567890123456789012345678901234567890'
        " iv_latest_version_id = '1'
        " iv_force = abap_false
        oo_result = lo_mig->updateimagesetmetadata(
          iv_datastoreid = iv_datastore_id
          iv_imagesetid = iv_image_set_id
          iv_latestversionid = iv_latest_version_id
          io_updateimagesetmetupdates = io_metadata_updates
          iv_force = iv_force ).
        DATA(lv_new_version) = oo_result->get_latestversionid( ).
        MESSAGE |Image set metadata updated to version: { lv_new_version }.| TYPE
'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Image set not found.' TYPE 'I'.
      CATCH /aws1/cx_migservicequotaexcdex.
        MESSAGE 'Service quota exceeded.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

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

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

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

Skenario untuk HealthImaging menggunakan AWS SDKs

Contoh kode berikut menunjukkan kepada Anda bagaimana menerapkan skenario umum HealthImaging dengan AWS SDKs. Skenario ini menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi di dalam HealthImaging atau dikombinasikan dengan yang lain Layanan AWS. Setiap skenario menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode.

Skenario menargetkan tingkat pengalaman menengah untuk membantu Anda memahami tindakan layanan dalam konteks.

Contoh

- Memulai set HealthImaging gambar dan bingkai gambar menggunakan AWS SDK
- Menandai penyimpanan HealthImaging data menggunakan SDK AWS
- Menandai set HealthImaging gambar menggunakan SDK AWS

Memulai set HealthImaging gambar dan bingkai gambar menggunakan AWS SDK

Contoh kode berikut menunjukkan cara mengimpor file DICOM dan mengunduh bingkai gambar di HealthImaging.

Implementasinya disusun sebagai aplikasi baris perintah.

- Siapkan sumber daya untuk impor DICOM.
- Impor file DICOM ke penyimpanan data.
- Ambil gambar yang ditetapkan IDs untuk pekerjaan impor.
- Ambil bingkai gambar IDs untuk set gambar.
- Unduh, dekode, dan verifikasi bingkai gambar.
- Pembersihan sumber daya

C++

SDK untuk C++

Buat CloudFormation tumpukan dengan sumber daya yang diperlukan.

```
   Aws::String inputBucketName;
   Aws::String outputBucketName;
   Aws::String dataStoreId;
   Aws::String roleArn;
   Aws::String stackName;

   if (askYesNoQuestion(
           "Would you like to let this workflow create the resources for you?
(y/n) ")) {
       stackName = askQuestion(
               "Enter a name for the AWS CloudFormation stack to create. ");
       Aws::String dataStoreName = askQuestion(
               "Enter a name for the HealthImaging datastore to create. ");

       Aws::Map<Aws::String, Aws::String> outputs = createCloudFormationStack(
               stackName,
               dataStoreName,
               clientConfiguration);

       if (!retrieveOutputs(outputs, dataStoreId, inputBucketName,
outputBucketName,
                            roleArn)) {
           return false;
       }

       std::cout << "The following resources have been created." << std::endl;
       std::cout << "A HealthImaging datastore with ID: " << dataStoreId << "."
                 << std::endl;
       std::cout << "An Amazon S3 input bucket named: " << inputBucketName <<
"."
                 << std::endl;
       std::cout << "An Amazon S3 output bucket named: " << outputBucketName <<
"."
                 << std::endl;
       std::cout << "An IAM role with the ARN: " << roleArn << "." << std::endl;
       askQuestion("Enter return to continue.", alwaysTrueTest);
   }
   else {
```

```
        std::cout << "You have chosen to use preexisting resources:" <<
std::endl;
        dataStoreId = askQuestion(
                "Enter the data store ID of the HealthImaging datastore you wish
to use: ");
        inputBucketName = askQuestion(
                "Enter the name of the S3 input bucket you wish to use: ");
        outputBucketName = askQuestion(
                "Enter the name of the S3 output bucket you wish to use: ");
        roleArn = askQuestion(
                "Enter the ARN for the IAM role with the proper permissions to
import a DICOM series: ");
    }
```

Salin file DICOM ke bucket impor Amazon S3.

```
    std::cout
            << "This workflow uses DICOM files from the National Cancer Institute
Imaging Data\n"
            << "Commons (IDC) Collections." << std::endl;
    std::cout << "Here is the link to their website." << std::endl;
    std::cout << "https://registry.opendata.aws/nci-imaging-data-commons/" <<
std::endl;
    std::cout << "We will use DICOM files stored in an S3 bucket managed by the
IDC."
              << std::endl;
    std::cout
            << "First one of the DICOM folders in the IDC collection must be
copied to your\n"
               "input S3 bucket."
            << std::endl;
    std::cout << "You have the choice of one of the following "
              << IDC_ImageChoices.size() << " folders to copy." << std::endl;

    int index = 1;
    for (auto &idcChoice: IDC_ImageChoices) {
        std::cout << index << " - " << idcChoice.mDescription << std::endl;
        index++;
    }
    int choice = askQuestionForIntRange("Choose DICOM files to import: ", 1, 4);

    Aws::String fromDirectory = IDC_ImageChoices[choice - 1].mDirectory;
```

```
   Aws::String inputDirectory = "input";

   std::cout << "The files in the directory '" << fromDirectory << "' in the
bucket '"
             << IDC_S3_BucketName << "' will be copied " << std::endl;
   std::cout << "to the folder '" << inputDirectory << "/" << fromDirectory
             << "' in the bucket '" << inputBucketName << "'." << std::endl;
   askQuestion("Enter return to start the copy.", alwaysTrueTest);

   if (!AwsDoc::Medical_Imaging::copySeriesBetweenBuckets(
           IDC_S3_BucketName,
           fromDirectory,
           inputBucketName,
           inputDirectory, clientConfiguration)) {
       std::cerr << "This workflow will exit because of an error." << std::endl;
       cleanup(stackName, dataStoreId, clientConfiguration);
       return false;
   }
```

Impor file DICOM ke penyimpanan data Amazon S3.

```
bool AwsDoc::Medical_Imaging::startDicomImport(const Aws::String &dataStoreID,
                                               const Aws::String
 &inputBucketName,
                                               const Aws::String &inputDirectory,
                                               const Aws::String
 &outputBucketName,
                                               const Aws::String
 &outputDirectory,
                                               const Aws::String &roleArn,
                                               Aws::String &importJobId,
                                               const
 Aws::Client::ClientConfiguration &clientConfiguration) {
   bool result = false;
   if (startDICOMImportJob(dataStoreID, inputBucketName, inputDirectory,
                           outputBucketName, outputDirectory, roleArn,
 importJobId,
                           clientConfiguration)) {
       std::cout << "DICOM import job started with job ID " << importJobId <<
 "."
                 << std::endl;
```

```
        result = waitImportJobCompleted(dataStoreID, importJobId,
 clientConfiguration);
        if (result) {
            std::cout << "DICOM import job completed." << std::endl;

        }
    }

    return result;
}

//! Routine which starts a HealthImaging import job.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param inputBucketName: The name of the Amazon S3 bucket containing the DICOM
 files.
  \param inputDirectory: The directory in the S3 bucket containing the DICOM
 files.
  \param outputBucketName: The name of the S3 bucket for the output.
  \param outputDirectory: The directory in the S3 bucket to store the output.
  \param roleArn: The ARN of the IAM role with permissions for the import.
  \param importJobId: A string to receive the import job ID.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
  */
bool AwsDoc::Medical_Imaging::startDICOMImportJob(
        const Aws::String &dataStoreID, const Aws::String &inputBucketName,
        const Aws::String &inputDirectory, const Aws::String &outputBucketName,
        const Aws::String &outputDirectory, const Aws::String &roleArn,
        Aws::String &importJobId,
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient medicalImagingClient(clientConfig);
    Aws::String inputURI = "s3://" + inputBucketName + "/" + inputDirectory +
 "/";
    Aws::String outputURI = "s3://" + outputBucketName + "/" + outputDirectory +
 "/";
    Aws::MedicalImaging::Model::StartDICOMImportJobRequest
 startDICOMImportJobRequest;
    startDICOMImportJobRequest.SetDatastoreId(dataStoreID);
    startDICOMImportJobRequest.SetDataAccessRoleArn(roleArn);
    startDICOMImportJobRequest.SetInputS3Uri(inputURI);
    startDICOMImportJobRequest.SetOutputS3Uri(outputURI);
```

```
    Aws::MedicalImaging::Model::StartDICOMImportJobOutcome
 startDICOMImportJobOutcome = medicalImagingClient.StartDICOMImportJob(
            startDICOMImportJobRequest);

    if (startDICOMImportJobOutcome.IsSuccess()) {
        importJobId = startDICOMImportJobOutcome.GetResult().GetJobId();
    }
    else {
        std::cerr << "Failed to start DICOM import job because "
                  << startDICOMImportJobOutcome.GetError().GetMessage() <<
 std::endl;
    }

    return startDICOMImportJobOutcome.IsSuccess();
}


//! Routine which waits for a DICOM import job to complete.
/*!
 * @param dataStoreID: The HealthImaging data store ID.
 * @param importJobId: The import job ID.
 * @param clientConfiguration : Aws client configuration.
 * @return  bool: Function succeeded.
 */
bool AwsDoc::Medical_Imaging::waitImportJobCompleted(const Aws::String
 &datastoreID,
                                                     const Aws::String
 &importJobId,
                                                     const
 Aws::Client::ClientConfiguration &clientConfiguration) {

    Aws::MedicalImaging::Model::JobStatus jobStatus =
 Aws::MedicalImaging::Model::JobStatus::IN_PROGRESS;
    while (jobStatus == Aws::MedicalImaging::Model::JobStatus::IN_PROGRESS) {
        std::this_thread::sleep_for(std::chrono::seconds(1));

        Aws::MedicalImaging::Model::GetDICOMImportJobOutcome
 getDicomImportJobOutcome = getDICOMImportJob(
                datastoreID, importJobId,
                clientConfiguration);

        if (getDicomImportJobOutcome.IsSuccess()) {
            jobStatus =
 getDicomImportJobOutcome.GetResult().GetJobProperties().GetJobStatus();
```

```
            std::cout << "DICOM import job status: " <<

 Aws::MedicalImaging::Model::JobStatusMapper::GetNameForJobStatus(
                               jobStatus) << std::endl;
        }
        else {
            std::cerr << "Failed to get import job status because "
                      << getDicomImportJobOutcome.GetError().GetMessage() <<
 std::endl;
            return false;
        }
    }

    return jobStatus == Aws::MedicalImaging::Model::JobStatus::COMPLETED;
}

//! Routine which gets a HealthImaging DICOM import job's properties.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param importJobID: The DICOM import job ID
  \param clientConfig: Aws client configuration.
  \return GetDICOMImportJobOutcome: The import job outcome.
*/
Aws::MedicalImaging::Model::GetDICOMImportJobOutcome
AwsDoc::Medical_Imaging::getDICOMImportJob(const Aws::String &dataStoreID,
                                           const Aws::String &importJobID,
                                           const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::GetDICOMImportJobRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetJobId(importJobID);
    Aws::MedicalImaging::Model::GetDICOMImportJobOutcome outcome =
 client.GetDICOMImportJob(
            request);
    if (!outcome.IsSuccess()) {
        std::cerr << "GetDICOMImportJob error: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome;
}
```

Dapatkan set gambar yang dibuat oleh pekerjaan impor DICOM.

```
bool
AwsDoc::Medical_Imaging::getImageSetsForDicomImportJob(const Aws::String
 &datastoreID,
                                                       const Aws::String
 &importJobId,
                                                       Aws::Vector<Aws::String>
 &imageSets,
                                                       const
 Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::MedicalImaging::Model::GetDICOMImportJobOutcome getDicomImportJobOutcome
 = getDICOMImportJob(
            datastoreID, importJobId, clientConfiguration);
    bool result = false;
    if (getDicomImportJobOutcome.IsSuccess()) {
        auto outputURI =
 getDicomImportJobOutcome.GetResult().GetJobProperties().GetOutputS3Uri();
        Aws::Http::URI uri(outputURI);
        const Aws::String &bucket = uri.GetAuthority();
        Aws::String key = uri.GetPath();

        Aws::S3::S3Client s3Client(clientConfiguration);
        Aws::S3::Model::GetObjectRequest objectRequest;
        objectRequest.SetBucket(bucket);
        objectRequest.SetKey(key + "/" + IMPORT_JOB_MANIFEST_FILE_NAME);

        auto getObjectOutcome = s3Client.GetObject(objectRequest);
        if (getObjectOutcome.IsSuccess()) {
            auto &data = getObjectOutcome.GetResult().GetBody();

            std::stringstream stringStream;
            stringStream << data.rdbuf();

            try {
                // Use JMESPath to extract the image set IDs.
                // https://jmespath.org/specification.html
                std::string jmesPathExpression =
 "jobSummary.imageSetsSummary[].imageSetId";
                jsoncons::json doc = jsoncons::json::parse(stringStream.str());
```

```
                jsoncons::json imageSetsJson = jsoncons::jmespath::search(doc,

 jmesPathExpression);\
                for (auto &imageSet: imageSetsJson.array_range()) {
                    imageSets.push_back(imageSet.as_string());
                }

                result = true;
            }
            catch (const std::exception &e) {
                std::cerr << e.what() << '\n';
            }

        }
        else {
            std::cerr << "Failed to get object because "
                      << getObjectOutcome.GetError().GetMessage() << std::endl;
        }

    }
    else {
        std::cerr << "Failed to get import job status because "
                  << getDicomImportJobOutcome.GetError().GetMessage() <<
 std::endl;
    }

    return result;
}
```

Dapatkan informasi bingkai gambar untuk set gambar.

```
bool AwsDoc::Medical_Imaging::getImageFramesForImageSet(const Aws::String
 &dataStoreID,
                                                        const Aws::String
 &imageSetID,
                                                        const Aws::String
 &outDirectory,

 Aws::Vector<ImageFrameInfo> &imageFrames,
                                                        const
 Aws::Client::ClientConfiguration &clientConfiguration) {
```

```
   Aws::String fileName = outDirectory + "/" + imageSetID +
"_metadata.json.gzip";
   bool result = false;
   if (getImageSetMetadata(dataStoreID, imageSetID, "", // Empty string for
version ID.
                           fileName, clientConfiguration)) {
       try {
           std::string metadataGZip;
           {
               std::ifstream inFileStream(fileName.c_str(), std::ios::binary);
               if (!inFileStream) {
                   throw std::runtime_error("Failed to open file " + fileName);
               }

               std::stringstream stringStream;
               stringStream << inFileStream.rdbuf();
               metadataGZip = stringStream.str();
           }
           std::string metadataJson = gzip::decompress(metadataGZip.data(),
                                                       metadataGZip.size());
           // Use JMESPath to extract the image set IDs.
           // https://jmespath.org/specification.html
           jsoncons::json doc = jsoncons::json::parse(metadataJson);
           std::string jmesPathExpression = "Study.Series.*.Instances[].*[]";
           jsoncons::json instances = jsoncons::jmespath::search(doc,

jmesPathExpression);
           for (auto &instance: instances.array_range()) {
               jmesPathExpression = "DICOM.RescaleSlope";
               std::string rescaleSlope = jsoncons::jmespath::search(instance,

jmesPathExpression).to_string();
               jmesPathExpression = "DICOM.RescaleIntercept";
               std::string rescaleIntercept =
jsoncons::jmespath::search(instance,

jmesPathExpression).to_string();

               jmesPathExpression = "ImageFrames[][]";
               jsoncons::json imageFramesJson =
jsoncons::jmespath::search(instance,

jmesPathExpression);
```

```
                for (auto &imageFrame: imageFramesJson.array_range()) {
                    ImageFrameInfo imageFrameIDs;
                    imageFrameIDs.mImageSetId = imageSetID;
                    imageFrameIDs.mImageFrameId = imageFrame.find(
                            "ID")->value().as_string();
                    imageFrameIDs.mRescaleIntercept = rescaleIntercept;
                    imageFrameIDs.mRescaleSlope = rescaleSlope;
                    imageFrameIDs.MinPixelValue = imageFrame.find(
                            "MinPixelValue")->value().as_string();
                    imageFrameIDs.MaxPixelValue = imageFrame.find(
                            "MaxPixelValue")->value().as_string();

                    jmesPathExpression =
 "max_by(PixelDataChecksumFromBaseToFullResolution, &Width).Checksum";
                    jsoncons::json checksumJson =
 jsoncons::jmespath::search(imageFrame,

 jmesPathExpression);
                    imageFrameIDs.mFullResolutionChecksum =
 checksumJson.as_integer<uint32_t>();

                    imageFrames.emplace_back(imageFrameIDs);
                }
            }

            result = true;
        }
        catch (const std::exception &e) {
            std::cerr << "getImageFramesForImageSet failed because " << e.what()
                      << std::endl;
        }
    }

    return result;
}

//! Routine which gets a HealthImaging image set's metadata.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param imageSetID: The HealthImaging image set ID.
  \param versionID: The HealthImaging image set version ID, ignored if empty.
  \param outputFilePath: The path where the metadata will be stored as gzipped
 json.
  \param clientConfig: Aws client configuration.
```

```
  \\return bool: Function succeeded.
*/
bool AwsDoc::Medical_Imaging::getImageSetMetadata(const Aws::String &dataStoreID,
                                                  const Aws::String &imageSetID,
                                                  const Aws::String &versionID,
                                                  const Aws::String
 &outputFilePath,
                                                  const
 Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::Model::GetImageSetMetadataRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);
    if (!versionID.empty()) {
        request.SetVersionId(versionID);
    }
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::GetImageSetMetadataOutcome outcome =
 client.GetImageSetMetadata(
            request);
    if (outcome.IsSuccess()) {
        std::ofstream file(outputFilePath, std::ios::binary);
        auto &metadata = outcome.GetResult().GetImageSetMetadataBlob();
        file << metadata.rdbuf();
    }
    else {
        std::cerr << "Failed to get image set metadata: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

Unduh, dekode, dan verifikasi bingkai gambar.

```
bool AwsDoc::Medical_Imaging::downloadDecodeAndCheckImageFrames(
        const Aws::String &dataStoreID,
        const Aws::Vector<ImageFrameInfo> &imageFrames,
        const Aws::String &outDirectory,
        const Aws::Client::ClientConfiguration &clientConfiguration) {

    Aws::Client::ClientConfiguration clientConfiguration1(clientConfiguration);
```

```
   clientConfiguration1.executor =
Aws::MakeShared<Aws::Utils::Threading::PooledThreadExecutor>(
           "executor", 25);
   Aws::MedicalImaging::MedicalImagingClient medicalImagingClient(
           clientConfiguration1);

   Aws::Utils::Threading::Semaphore semaphore(0, 1);
   std::atomic<size_t> count(imageFrames.size());

   bool result = true;
   for (auto &imageFrame: imageFrames) {
       Aws::MedicalImaging::Model::GetImageFrameRequest getImageFrameRequest;
       getImageFrameRequest.SetDatastoreId(dataStoreID);
       getImageFrameRequest.SetImageSetId(imageFrame.mImageSetId);

       Aws::MedicalImaging::Model::ImageFrameInformation imageFrameInformation;
       imageFrameInformation.SetImageFrameId(imageFrame.mImageFrameId);
       getImageFrameRequest.SetImageFrameInformation(imageFrameInformation);

       auto getImageFrameAsyncLambda = [&semaphore, &result, &count, imageFrame,
outDirectory](
               const Aws::MedicalImaging::MedicalImagingClient *client,
               const Aws::MedicalImaging::Model::GetImageFrameRequest &request,
               Aws::MedicalImaging::Model::GetImageFrameOutcome outcome,
               const std::shared_ptr<const Aws::Client::AsyncCallerContext>
&context) {

               if (!handleGetImageFrameResult(outcome, outDirectory,
imageFrame)) {
                   std::cerr << "Failed to download and convert image frame: "
                             << imageFrame.mImageFrameId << " from image set: "
                             << imageFrame.mImageSetId << std::endl;
                   result = false;
               }

               count--;
               if (count <= 0) {

                   semaphore.ReleaseAll();
               }
       }; // End of 'getImageFrameAsyncLambda' lambda.

       medicalImagingClient.GetImageFrameAsync(getImageFrameRequest,
                                               getImageFrameAsyncLambda);
```

```
    }

    if (count > 0) {
        semaphore.WaitOne();
    }

    if (result) {
        std::cout << imageFrames.size() << " image files were downloaded."
                  << std::endl;
    }

    return result;
}

bool AwsDoc::Medical_Imaging::decodeJPHFileAndValidateWithChecksum(
        const Aws::String &jphFile,
        uint32_t crc32Checksum) {
    opj_image_t *outputImage = jphImageToOpjBitmap(jphFile);
    if (!outputImage) {
        return false;
    }

    bool result = true;
    if (!verifyChecksumForImage(outputImage, crc32Checksum)) {
        std::cerr << "The checksum for the image does not match the expected
 value."
                  << std::endl;
        std::cerr << "File :" << jphFile << std::endl;
        result = false;
    }

    opj_image_destroy(outputImage);

    return result;
}

opj_image *
AwsDoc::Medical_Imaging::jphImageToOpjBitmap(const Aws::String &jphFile) {
    opj_stream_t *inFileStream = nullptr;
    opj_codec_t *decompressorCodec = nullptr;
    opj_image_t *outputImage = nullptr;
    try {
        std::shared_ptr<opj_dparameters> decodeParameters =
 std::make_shared<opj_dparameters>();
```

```
        memset(decodeParameters.get(), 0, sizeof(opj_dparameters));

        opj_set_default_decoder_parameters(decodeParameters.get());

        decodeParameters->decod_format = 1; // JP2 image format.
        decodeParameters->cod_format = 2; // BMP image format.

        std::strncpy(decodeParameters->infile, jphFile.c_str(),
                     OPJ_PATH_LEN);

        inFileStream = opj_stream_create_default_file_stream(
                decodeParameters->infile, true);
        if (!inFileStream) {
            throw std::runtime_error(
                    "Unable to create input file stream for file '" + jphFile +
"'.");
        }

        decompressorCodec = opj_create_decompress(OPJ_CODEC_JP2);
        if (!decompressorCodec) {
            throw std::runtime_error("Failed to create decompression codec.");
        }

        int decodeMessageLevel = 1;
        if (!setupCodecLogging(decompressorCodec, &decodeMessageLevel)) {
            std::cerr << "Failed to setup codec logging." << std::endl;
        }

        if (!opj_setup_decoder(decompressorCodec, decodeParameters.get())) {
            throw std::runtime_error("Failed to setup decompression codec.");
        }
        if (!opj_codec_set_threads(decompressorCodec, 4)) {
            throw std::runtime_error("Failed to set decompression codec
threads.");
        }

        if (!opj_read_header(inFileStream, decompressorCodec, &outputImage)) {
            throw std::runtime_error("Failed to read header.");
        }

        if (!opj_decode(decompressorCodec, inFileStream,
                        outputImage)) {
            throw std::runtime_error("Failed to decode.");
        }
```

```
        if (DEBUGGING) {
            std::cout << "image width : " << outputImage->x1 - outputImage->x0
                      << std::endl;
            std::cout << "image height : " << outputImage->y1 - outputImage->y0
                      << std::endl;
            std::cout << "number of channels: " << outputImage->numcomps
                      << std::endl;
            std::cout << "colorspace : " << outputImage->color_space <<
 std::endl;
        }

    } catch (const std::exception &e) {
        std::cerr << e.what() << std::endl;
        if (outputImage) {
            opj_image_destroy(outputImage);
            outputImage = nullptr;
        }
    }
    if (inFileStream) {
        opj_stream_destroy(inFileStream);
    }
    if (decompressorCodec) {
        opj_destroy_codec(decompressorCodec);
    }

    return outputImage;
}

//! Template function which converts a planar image bitmap to an interleaved
 image bitmap and
//! then verifies the checksum of the bitmap.
/*!
 * @param image: The OpenJPEG image struct.
 * @param crc32Checksum: The CRC32 checksum.
 * @return  bool: Function succeeded.
 */
template<class myType>
bool verifyChecksumForImageForType(opj_image_t *image, uint32_t crc32Checksum) {
    uint32_t width = image->x1 - image->x0;
    uint32_t height = image->y1 - image->y0;
    uint32_t numOfChannels = image->numcomps;

    // Buffer for interleaved bitmap.
```

```
    std::vector<myType> buffer(width * height * numOfChannels);

    // Convert planar bitmap to interleaved bitmap.
    for (uint32_t channel = 0; channel < numOfChannels; channel++) {
        for (uint32_t row = 0; row < height; row++) {
            uint32_t fromRowStart = row / image->comps[channel].dy * width /
                                    image->comps[channel].dx;
            uint32_t toIndex = (row * width) * numOfChannels + channel;

            for (uint32_t col = 0; col < width; col++) {
                uint32_t fromIndex = fromRowStart + col / image-
>comps[channel].dx;

                buffer[toIndex] = static_cast<myType>(image-
>comps[channel].data[fromIndex]);

                toIndex += numOfChannels;
            }
        }
    }

    // Verify checksum.
    boost::crc_32_type crc32;
    crc32.process_bytes(reinterpret_cast<char *>(buffer.data()),
                        buffer.size() * sizeof(myType));

    bool result = crc32.checksum() == crc32Checksum;
    if (!result) {
        std::cerr << "verifyChecksumForImage, checksum mismatch, expected - "
                  << crc32Checksum << ", actual - " << crc32.checksum()
                  << std::endl;
    }

    return result;
}

//! Routine which verifies the checksum of an OpenJPEG image struct.
/*!
 * @param image: The OpenJPEG image struct.
 * @param crc32Checksum: The CRC32 checksum.
 * @return  bool: Function succeeded.
 */
bool AwsDoc::Medical_Imaging::verifyChecksumForImage(opj_image_t *image,
                                                     uint32_t crc32Checksum) {
```

```
   uint32_t channels = image->numcomps;
   bool result = false;
   if (0 < channels) {
       // Assume the precision is the same for all channels.
       uint32_t precision = image->comps[0].prec;
       bool signedData = image->comps[0].sgnd;
       uint32_t bytes = (precision + 7) / 8;

       if (signedData) {
           switch (bytes) {
               case 1 :
                   result = verifyChecksumForImageForType<int8_t>(image,

crc32Checksum);
                   break;
               case 2 :
                   result = verifyChecksumForImageForType<int16_t>(image,

crc32Checksum);
                   break;
               case 4 :
                   result = verifyChecksumForImageForType<int32_t>(image,

crc32Checksum);
                   break;
               default:
                   std::cerr
                           << "verifyChecksumForImage, unsupported data type,
signed bytes - "
                           << bytes << std::endl;
                   break;
           }
       }
       else {
           switch (bytes) {
               case 1 :
                   result = verifyChecksumForImageForType<uint8_t>(image,

crc32Checksum);
                   break;
               case 2 :
                   result = verifyChecksumForImageForType<uint16_t>(image,

crc32Checksum);
```

```
                    break;
                case 4 :
                    result = verifyChecksumForImageForType<uint32_t>(image,

 crc32Checksum);
                    break;
                default:
                    std::cerr
                            << "verifyChecksumForImage, unsupported data type,
 unsigned bytes - "
                            << bytes << std::endl;
                    break;
            }
        }

        if (!result) {
            std::cerr << "verifyChecksumForImage, error bytes " << bytes
                      << " signed "
                      << signedData << std::endl;
        }
    }
    else {
        std::cerr << "'verifyChecksumForImage', no channels in the image."
                  << std::endl;
    }
    return result;
}
```

Pembersihan sumber daya

```
bool AwsDoc::Medical_Imaging::cleanup(const Aws::String &stackName,
                                      const Aws::String &dataStoreId,
                                      const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    bool result = true;

    if (!stackName.empty() && askYesNoQuestion(
            "Would you like to delete the stack " + stackName + "? (y/n)")) {
        std::cout << "Deleting the image sets in the stack." << std::endl;
        result &= emptyDatastore(dataStoreId, clientConfiguration);
        printAsterisksLine();
        std::cout << "Deleting the stack." << std::endl;
```

```
        result &= deleteStack(stackName, clientConfiguration);
    }
    return result;
}

bool AwsDoc::Medical_Imaging::emptyDatastore(const Aws::String &datastoreID,
                                             const
 Aws::Client::ClientConfiguration &clientConfiguration) {

    Aws::MedicalImaging::Model::SearchCriteria emptyCriteria;
    Aws::Vector<Aws::String> imageSetIDs;
    bool result = false;
    if (searchImageSets(datastoreID, emptyCriteria, imageSetIDs,
                        clientConfiguration)) {
        result = true;
        for (auto &imageSetID: imageSetIDs) {
            result &= deleteImageSet(datastoreID, imageSetID,
 clientConfiguration);
        }
    }

    return result;
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK untuk C++ .
  - DeleteImageSet
  - Dapatkan DICOMImport Job
  - GetImageFrame
  - GetImageSetMetadata
  - SearchImageSets
  - Mulai DICOMImport Job

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

JavaScript

SDK untuk JavaScript (v3)

Mengatur langkah (). index.js

```
import {
  parseScenarioArgs,
  Scenario,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  saveState,
  loadState,
} from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";

import {
  createStack,
  deployStack,
  getAccountId,
  getDatastoreName,
  getStackName,
  outputState,
  waitForStackCreation,
} from "./deploy-steps.js";
import {
  doCopy,
  selectDataset,
  copyDataset,
  outputCopiedObjects,
} from "./dataset-steps.js";
import {
  doImport,
  outputImportJobStatus,
  startDICOMImport,
  waitForImportJobCompletion,
} from "./import-steps.js";
import {
  getManifestFile,
  outputImageSetIds,
  parseManifestFile,
} from "./image-set-steps.js";
import {
  getImageSetMetadata,
```

```
  outputImageFrameIds,
} from "./image-frame-steps.js";
import { decodeAndVerifyImages, doVerify } from "./verify-steps.js";
import {
  confirmCleanup,
  deleteImageSets,
  deleteStack,
} from "./clean-up-steps.js";

const context = {};

const scenarios = {
  deploy: new Scenario(
    "Deploy Resources",
    [
      deployStack,
      getStackName,
      getDatastoreName,
      getAccountId,
      createStack,
      waitForStackCreation,
      outputState,
      saveState,
    ],
    context,
  ),
  demo: new Scenario(
    "Run Demo",
    [
      loadState,
      doCopy,
      selectDataset,
      copyDataset,
      outputCopiedObjects,
      doImport,
      startDICOMImport,
      waitForImportJobCompletion,
      outputImportJobStatus,
      getManifestFile,
      parseManifestFile,
      outputImageSetIds,
      getImageSetMetadata,
      outputImageFrameIds,
      doVerify,
```

```
      decodeAndVerifyImages,
      saveState,
    ],
    context,
  ),
  destroy: new Scenario(
    "Clean Up Resources",
    [loadState, confirmCleanup, deleteImageSets, deleteStack],
    context,
  ),
};

// Call function if run directly
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Health Imaging Workflow",
    description:
      "Work with DICOM images using an AWS Health Imaging data store.",
    synopsis:
      "node index.js --scenario <deploy | demo | destroy> [-h|--help] [-y|--yes]
 [-v|--verbose]",
  });
}
```

Menyebarkan sumber daya (deploy-steps.js).

```
import fs from "node:fs/promises";
import path from "node:path";

import {
  CloudFormationClient,
  CreateStackCommand,
  DescribeStacksCommand,
} from "@aws-sdk/client-cloudformation";
import { STSClient, GetCallerIdentityCommand } from "@aws-sdk/client-sts";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
```

```
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

const cfnClient = new CloudFormationClient({});
const stsClient = new STSClient({});

const __dirname = path.dirname(new URL(import.meta.url).pathname);
const cfnTemplatePath = path.join(
  __dirname,
  "../../../../../scenarios/features/healthimaging_image_sets/resources/
cfn_template.yaml",
);

export const deployStack = new ScenarioInput(
  "deployStack",
  "Do you want to deploy the CloudFormation stack?",
  { type: "confirm" },
);

export const getStackName = new ScenarioInput(
  "getStackName",
  "Enter a name for the CloudFormation stack:",
  { type: "input", skipWhen: (/** @type {{}} */ state) => !state.deployStack },
);

export const getDatastoreName = new ScenarioInput(
  "getDatastoreName",
  "Enter a name for the HealthImaging datastore:",
  { type: "input", skipWhen: (/** @type {{}} */ state) => !state.deployStack },
);

export const getAccountId = new ScenarioAction(
  "getAccountId",
  async (/** @type {{}} */ state) => {
    const command = new GetCallerIdentityCommand({});
    const response = await stsClient.send(command);
    state.accountId = response.Account;
  },
  {
    skipWhen: (/** @type {{}} */ state) => !state.deployStack,
  },
);

export const createStack = new ScenarioAction(
  "createStack",
```

```
  async (/** @type {{}} */ state) => {
    const stackName = state.getStackName;
    const datastoreName = state.getDatastoreName;
    const accountId = state.accountId;

    const command = new CreateStackCommand({
      StackName: stackName,
      TemplateBody: await fs.readFile(cfnTemplatePath, "utf8"),
      Capabilities: ["CAPABILITY_IAM"],
      Parameters: [
        {
          ParameterKey: "datastoreName",
          ParameterValue: datastoreName,
        },
        {
          ParameterKey: "userAccountID",
          ParameterValue: accountId,
        },
      ],
    });

    const response = await cfnClient.send(command);
    state.stackId = response.StackId;
  },
  { skipWhen: (/** @type {{}} */ state) => !state.deployStack },
);

export const waitForStackCreation = new ScenarioAction(
  "waitForStackCreation",
  async (/** @type {{}} */ state) => {
    const command = new DescribeStacksCommand({
      StackName: state.stackId,
    });

    await retry({ intervalInMs: 10000, maxRetries: 60 }, async () => {
      const response = await cfnClient.send(command);
      const stack = response.Stacks?.find(
        (s) => s.StackName === state.getStackName,
      );
      if (!stack || stack.StackStatus === "CREATE_IN_PROGRESS") {
        throw new Error("Stack creation is still in progress");
      }
      if (stack.StackStatus === "CREATE_COMPLETE") {
        state.stackOutputs = stack.Outputs?.reduce((acc, output) => {
```

```
          acc[output.OutputKey] = output.OutputValue;
          return acc;
        }, {});
      } else {
        throw new Error(
          `Stack creation failed with status: ${stack.StackStatus}`,
        );
      }
    });
  },
  {
    skipWhen: (/** @type {{}} */ state) => !state.deployStack,
  },
);

export const outputState = new ScenarioOutput(
  "outputState",
  (/** @type {{}} */ state) => {
    /**
     * @type {{ stackOutputs: { DatastoreID: string, BucketName: string, RoleArn:
 string }}}
     */
    const { stackOutputs } = state;
    return `Stack creation completed. Output values:
Datastore ID: ${stackOutputs?.DatastoreID}
Bucket Name: ${stackOutputs?.BucketName}
Role ARN: ${stackOutputs?.RoleArn}
    `;
  },
  { skipWhen: (/** @type {{}} */ state) => !state.deployStack },
);
```

Salin file DICOM (dataset-steps.js).

```
import {
  S3Client,
  CopyObjectCommand,
  ListObjectsV2Command,
} from "@aws-sdk/client-s3";

import {
  ScenarioAction,
```

```
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

const s3Client = new S3Client({});

const datasetOptions = [
  {
    name: "CT of chest (2 images)",
    value: "00029d25-fb18-4d42-aaa5-a0897d1ac8f7",
  },
  {
    name: "CT of pelvis (57 images)",
    value: "00025d30-ef8f-4135-a35a-d83eff264fc1",
  },
  {
    name: "MRI of head (192 images)",
    value: "0002d261-8a5d-4e63-8e2e-0cbfac87b904",
  },
  {
    name: "MRI of breast (92 images)",
    value: "0002dd07-0b7f-4a68-a655-44461ca34096",
  },
];

/**
 * @typedef {{ stackOutputs: {
 *   BucketName: string,
 *   DatastoreID: string,
 *   doCopy: boolean
 * }}} State
 */

export const selectDataset = new ScenarioInput(
  "selectDataset",
  (state) => {
    if (!state.doCopy) {
      process.exit(0);
    }
    return "Select a DICOM dataset to import:";
  },
  {
    type: "select",
    choices: datasetOptions,
```

```
  },
);

export const doCopy = new ScenarioInput(
  "doCopy",
  "Do you want to copy images from the public dataset into your bucket?",
  {
    type: "confirm",
  },
);

export const copyDataset = new ScenarioAction(
  "copyDataset",
  async (/** @type { State } */ state) => {
    const inputBucket = state.stackOutputs.BucketName;
    const inputPrefix = "input/";
    const selectedDatasetId = state.selectDataset;

    const sourceBucket = "idc-open-data";
    const sourcePrefix = `${selectedDatasetId}`;

    const listObjectsCommand = new ListObjectsV2Command({
      Bucket: sourceBucket,
      Prefix: sourcePrefix,
    });

    const objects = await s3Client.send(listObjectsCommand);

    const copyPromises = objects.Contents.map((object) => {
      const sourceKey = object.Key;
      const destinationKey = `${inputPrefix}${sourceKey
        .split("/")
        .slice(1)
        .join("/")}`;

      const copyCommand = new CopyObjectCommand({
        Bucket: inputBucket,
        CopySource: `/${sourceBucket}/${sourceKey}`,
        Key: destinationKey,
      });

      return s3Client.send(copyCommand);
    });
```

```
    const results = await Promise.all(copyPromises);
    state.copiedObjects = results.length;
  },
);

export const outputCopiedObjects = new ScenarioOutput(
  "outputCopiedObjects",
  (state) => `${state.copiedObjects} DICOM files were copied.`,
);
```

Mulai impor ke datastore ()import-steps.js.

```
import {
  MedicalImagingClient,
  StartDICOMImportJobCommand,
  GetDICOMImportJobCommand,
} from "@aws-sdk/client-medical-imaging";

import {
  ScenarioAction,
  ScenarioOutput,
  ScenarioInput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

/**
 * @typedef {{ stackOutputs: {
 *   BucketName: string,
 *   DatastoreID: string,
 *   RoleArn: string
 * }}} State
 */

export const doImport = new ScenarioInput(
  "doImport",
  "Do you want to import DICOM images into your datastore?",
  {
    type: "confirm",
    default: true,
  },
);
```

```
export const startDICOMImport = new ScenarioAction(
  "startDICOMImport",
  async (/** @type {State} */ state) => {
    if (!state.doImport) {
      process.exit(0);
    }
    const medicalImagingClient = new MedicalImagingClient({});
    const inputS3Uri = `s3://${state.stackOutputs.BucketName}/input/`;
    const outputS3Uri = `s3://${state.stackOutputs.BucketName}/output/`;

    const command = new StartDICOMImportJobCommand({
      dataAccessRoleArn: state.stackOutputs.RoleArn,
      datastoreId: state.stackOutputs.DatastoreID,
      inputS3Uri,
      outputS3Uri,
    });

    const response = await medicalImagingClient.send(command);
    state.importJobId = response.jobId;
  },
);

export const waitForImportJobCompletion = new ScenarioAction(
  "waitForImportJobCompletion",
  async (/** @type {State} */ state) => {
    const medicalImagingClient = new MedicalImagingClient({});
    const command = new GetDICOMImportJobCommand({
      datastoreId: state.stackOutputs.DatastoreID,
      jobId: state.importJobId,
    });

    await retry({ intervalInMs: 10000, maxRetries: 60 }, async () => {
      const response = await medicalImagingClient.send(command);
      const jobStatus = response.jobProperties?.jobStatus;
      if (!jobStatus || jobStatus === "IN_PROGRESS") {
        throw new Error("Import job is still in progress");
      }
      if (jobStatus === "COMPLETED") {
        state.importJobOutputS3Uri = response.jobProperties.outputS3Uri;
      } else {
        throw new Error(`Import job failed with status: ${jobStatus}`);
      }
    });
  },
```

```
);

export const outputImportJobStatus = new ScenarioOutput(
  "outputImportJobStatus",
  (state) =>
    `DICOM import job completed. Output location: ${state.importJobOutputS3Uri}`,
);
```

Dapatkan set gambar IDs (image-set-steps.js-).

```
import { S3Client, GetObjectCommand } from "@aws-sdk/client-s3";

import {
  ScenarioAction,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * @typedef {{ stackOutputs: {
 *   BucketName: string,
 *   DatastoreID: string,
 *   RoleArn: string
 * }, importJobId: string,
 * importJobOutputS3Uri: string,
 * imageSetIds: string[],
 * manifestContent: { jobSummary: { imageSetsSummary: { imageSetId: string }
[] } }
 * }} State
 */

const s3Client = new S3Client({});

export const getManifestFile = new ScenarioAction(
  "getManifestFile",
  async (/** @type {State} */ state) => {
    const bucket = state.stackOutputs.BucketName;
    const prefix = `output/${state.stackOutputs.DatastoreID}-DicomImport-
${state.importJobId}/`;
    const key = `${prefix}job-output-manifest.json`;

    const command = new GetObjectCommand({
      Bucket: bucket,
```

```
      Key: key,
    });

    const response = await s3Client.send(command);
    const manifestContent = await response.Body.transformToString();
    state.manifestContent = JSON.parse(manifestContent);
  },
);

export const parseManifestFile = new ScenarioAction(
  "parseManifestFile",
  (/** @type {State} */ state) => {
    const imageSetIds =
      state.manifestContent.jobSummary.imageSetsSummary.reduce((ids, next) => {
        return Object.assign({}, ids, {
          [next.imageSetId]: next.imageSetId,
        });
      }, {});
    state.imageSetIds = Object.keys(imageSetIds);
  },
);

export const outputImageSetIds = new ScenarioOutput(
  "outputImageSetIds",
  (/** @type {State} */ state) =>
    `The image sets created by this import job are: \n${state.imageSetIds
      .map((id) => `Image set: ${id}`)
      .join("\n")}`,
);
```

Dapatkan bingkai gambar IDs (image-frame-steps.js).

```
import {
  MedicalImagingClient,
  GetImageSetMetadataCommand,
} from "@aws-sdk/client-medical-imaging";
import { gunzip } from "node:zlib";
import { promisify } from "node:util";

import {
  ScenarioAction,
  ScenarioOutput,
```

```
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

const gunzipAsync = promisify(gunzip);

/**
 * @typedef {Object} DICOMValueRepresentation
 * @property {string} name
 * @property {string} type
 * @property {string} value
 */

/**
 * @typedef {Object} ImageFrameInformation
 * @property {string} ID
 * @property {Array<{ Checksum: number, Height: number, Width: number }>}
 PixelDataChecksumFromBaseToFullResolution
 * @property {number} MinPixelValue
 * @property {number} MaxPixelValue
 * @property {number} FrameSizeInBytes
 */

/**
 * @typedef {Object} DICOMMetadata
 * @property {Object} DICOM
 * @property {DICOMValueRepresentation[]} DICOMVRs
 * @property {ImageFrameInformation[]} ImageFrames
 */

/**
 * @typedef {Object} Series
 * @property {{ [key: string]: DICOMMetadata }} Instances
 */

/**
 * @typedef {Object} Study
 * @property {Object} DICOM
 * @property {Series[]} Series
 */

/**
 * @typedef {Object} Patient
 * @property {Object} DICOM
 */
```

```
/**
 * @typedef {{
 *  SchemaVersion: string,
 *  DatastoreID: string,
 *  ImageSetID: string,
 *  Patient: Patient,
 *  Study: Study
 * }} ImageSetMetadata
 */

/**
 * @typedef {{ stackOutputs: {
 *   BucketName: string,
 *   DatastoreID: string,
 *   RoleArn: string
 * }, imageSetIds: string[] }} State
 */

const medicalImagingClient = new MedicalImagingClient({});

export const getImageSetMetadata = new ScenarioAction(
  "getImageSetMetadata",
  async (/** @type {State} */ state) => {
    const outputMetadata = [];

    for (const imageSetId of state.imageSetIds) {
      const command = new GetImageSetMetadataCommand({
        datastoreId: state.stackOutputs.DatastoreID,
        imageSetId,
      });

      const response = await medicalImagingClient.send(command);
      const compressedMetadataBlob =
        await response.imageSetMetadataBlob.transformToByteArray();
      const decompressedMetadata = await gunzipAsync(compressedMetadataBlob);
      const imageSetMetadata = JSON.parse(decompressedMetadata.toString());

      outputMetadata.push(imageSetMetadata);
    }

    state.imageSetMetadata = outputMetadata;
  },
);
```

```
export const outputImageFrameIds = new ScenarioOutput(
  "outputImageFrameIds",
  (/** @type {State & { imageSetMetadata: ImageSetMetadata[] }} */ state) => {
    let output = "";

    for (const metadata of state.imageSetMetadata) {
      const imageSetId = metadata.ImageSetID;
      /** @type {DICOMMetadata[]} */
      const instances = Object.values(metadata.Study.Series).flatMap(
        (series) => {
          return Object.values(series.Instances);
        },
      );
      const imageFrameIds = instances.flatMap((instance) =>
        instance.ImageFrames.map((frame) => frame.ID),
      );

      output += `Image set ID: ${imageSetId}\nImage frame IDs:\n
${imageFrameIds.join(
        "\n",
      )}\n\n`;
    }

    return output;
  },
);
```

Verifikasi bingkai gambar (verify-steps.js). Pustaka Verifikasi Data AWS HealthImaging Pixel digunakan untuk verifikasi.

```
import { spawn } from "node:child_process";

import {
  ScenarioAction,
  ScenarioInput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * @typedef {Object} DICOMValueRepresentation
 * @property {string} name
 * @property {string} type
 * @property {string} value
```

```
 */

/**
 * @typedef {Object} ImageFrameInformation
 * @property {string} ID
 * @property {Array<{ Checksum: number, Height: number, Width: number }>}
 PixelDataChecksumFromBaseToFullResolution
 * @property {number} MinPixelValue
 * @property {number} MaxPixelValue
 * @property {number} FrameSizeInBytes
 */

/**
 * @typedef {Object} DICOMMetadata
 * @property {Object} DICOM
 * @property {DICOMValueRepresentation[]} DICOMVRs
 * @property {ImageFrameInformation[]} ImageFrames
 */

/**
 * @typedef {Object} Series
 * @property {{ [key: string]: DICOMMetadata }} Instances
 */

/**
 * @typedef {Object} Study
 * @property {Object} DICOM
 * @property {Series[]} Series
 */

/**
 * @typedef {Object} Patient
 * @property {Object} DICOM
 */

/**
 * @typedef {{
 *  SchemaVersion: string,
 *  DatastoreID: string,
 *  ImageSetID: string,
 *  Patient: Patient,
 *  Study: Study
 * }} ImageSetMetadata
 */
```

```
/**
 * @typedef {{ stackOutputs: {
 *   BucketName: string,
 *   DatastoreID: string,
 *   RoleArn: string
 * }, imageSetMetadata: ImageSetMetadata[] }} State
 */

export const doVerify = new ScenarioInput(
  "doVerify",
  "Do you want to verify the imported images?",
  {
    type: "confirm",
    default: true,
  },
);

export const decodeAndVerifyImages = new ScenarioAction(
  "decodeAndVerifyImages",
  async (/** @type {State} */ state) => {
    if (!state.doVerify) {
      process.exit(0);
    }
    const verificationTool = "./pixel-data-verification/index.js";

    for (const metadata of state.imageSetMetadata) {
      const datastoreId = state.stackOutputs.DatastoreID;
      const imageSetId = metadata.ImageSetID;

      for (const [seriesInstanceUid, series] of Object.entries(
        metadata.Study.Series,
      )) {
        for (const [sopInstanceUid, _] of Object.entries(series.Instances)) {
          console.log(
            `Verifying image set ${imageSetId} with series ${seriesInstanceUid}
 and sop ${sopInstanceUid}`,
          );
          const child = spawn(
            "node",
            [
              verificationTool,
              datastoreId,
              imageSetId,
```

```
              seriesInstanceUid,
              sopInstanceUid,
            ],
            { stdio: "inherit" },
          );

          await new Promise((resolve, reject) => {
            child.on("exit", (code) => {
              if (code === 0) {
                resolve();
              } else {
                reject(
                  new Error(
                    `Verification tool exited with code ${code} for image set
 ${imageSetId}`,
                  ),
                );
              }
            });
          });
        }
      }
    }
  },
);
```

Hancurkan sumber daya (clean-up-steps.js).

```
import {
  CloudFormationClient,
  DeleteStackCommand,
} from "@aws-sdk/client-cloudformation";
import {
  MedicalImagingClient,
  DeleteImageSetCommand,
} from "@aws-sdk/client-medical-imaging";

import {
  ScenarioAction,
  ScenarioInput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
```

```
/**
 * @typedef {Object} DICOMValueRepresentation
 * @property {string} name
 * @property {string} type
 * @property {string} value
 */

/**
 * @typedef {Object} ImageFrameInformation
 * @property {string} ID
 * @property {Array<{ Checksum: number, Height: number, Width: number }>}
 PixelDataChecksumFromBaseToFullResolution
 * @property {number} MinPixelValue
 * @property {number} MaxPixelValue
 * @property {number} FrameSizeInBytes
 */

/**
 * @typedef {Object} DICOMMetadata
 * @property {Object} DICOM
 * @property {DICOMValueRepresentation[]} DICOMVRs
 * @property {ImageFrameInformation[]} ImageFrames
 */

/**
 * @typedef {Object} Series
 * @property {{ [key: string]: DICOMMetadata }} Instances
 */

/**
 * @typedef {Object} Study
 * @property {Object} DICOM
 * @property {Series[]} Series
 */

/**
 * @typedef {Object} Patient
 * @property {Object} DICOM
 */

/**
 * @typedef {{
 *  SchemaVersion: string,
 *  DatastoreID: string,
```

```
 *  ImageSetID: string,
 *  Patient: Patient,
 *  Study: Study
 * }} ImageSetMetadata
 */

/**
 * @typedef {{ stackOutputs: {
 *   BucketName: string,
 *   DatastoreID: string,
 *   RoleArn: string
 * }, imageSetMetadata: ImageSetMetadata[] }} State
 */

const cfnClient = new CloudFormationClient({});
const medicalImagingClient = new MedicalImagingClient({});

export const confirmCleanup = new ScenarioInput(
  "confirmCleanup",
  "Do you want to delete the created resources?",
  { type: "confirm" },
);

export const deleteImageSets = new ScenarioAction(
  "deleteImageSets",
  async (/** @type {State} */ state) => {
    const datastoreId = state.stackOutputs.DatastoreID;

    for (const metadata of state.imageSetMetadata) {
      const command = new DeleteImageSetCommand({
        datastoreId,
        imageSetId: metadata.ImageSetID,
      });

      try {
        await medicalImagingClient.send(command);
        console.log(`Successfully deleted image set ${metadata.ImageSetID}`);
      } catch (e) {
        if (e instanceof Error) {
          if (e.name === "ConflictException") {
            console.log(`Image set ${metadata.ImageSetID} already deleted`);
          }
        }
      }
```

```
    }
  },
  {
    skipWhen: (/** @type {{}} */ state) => !state.confirmCleanup,
  },
);

export const deleteStack = new ScenarioAction(
  "deleteStack",
  async (/** @type {State} */ state) => {
    const stackName = state.getStackName;

    const command = new DeleteStackCommand({
      StackName: stackName,
    });

    await cfnClient.send(command);
    console.log(`Stack ${stackName} deletion initiated`);
  },
  {
    skipWhen: (/** @type {{}} */ state) => !state.confirmCleanup,
  },
);
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK untuk JavaScript .
  - DeleteImageSet
  - Dapatkan DICOMImport Job
  - GetImageFrame
  - GetImageSetMetadata
  - SearchImageSets
  - Mulai DICOMImport Job

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

Python

SDK untuk Python (Boto3)

Buat CloudFormation tumpukan dengan sumber daya yang diperlukan.

```
    def deploy(self):
        """
        Deploys prerequisite resources used by the scenario. The resources are
        defined in the associated `setup.yaml` AWS CloudFormation script and are
 deployed
        as a CloudFormation stack, so they can be easily managed and destroyed.
        """

        print("\t\tLet's deploy the stack for resource creation.")
        stack_name = q.ask("\t\tEnter a name for the stack: ", q.non_empty)

        data_store_name = q.ask(
            "\t\tEnter a name for the Health Imaging Data Store: ", q.non_empty
        )

        account_id = boto3.client("sts").get_caller_identity()["Account"]

        with open(
            "../../../../scenarios/features/healthimaging_image_sets/resources/
cfn_template.yaml"
        ) as setup_file:
            setup_template = setup_file.read()
        print(f"\t\tCreating {stack_name}.")
        stack = self.cf_resource.create_stack(
            StackName=stack_name,
            TemplateBody=setup_template,
            Capabilities=["CAPABILITY_NAMED_IAM"],
            Parameters=[
                {
                    "ParameterKey": "datastoreName",
                    "ParameterValue": data_store_name,
                },
                {
                    "ParameterKey": "userAccountID",
                    "ParameterValue": account_id,
                },
            ],
        )
```

```
        print("\t\tWaiting for stack to deploy. This typically takes a minute or
two.")
        waiter = self.cf_resource.meta.client.get_waiter("stack_create_complete")
        waiter.wait(StackName=stack.name)
        stack.load()
        print(f"\t\tStack status: {stack.stack_status}")

        outputs_dictionary = {
            output["OutputKey"]: output["OutputValue"] for output in
stack.outputs
        }
        self.input_bucket_name = outputs_dictionary["BucketName"]
        self.output_bucket_name = outputs_dictionary["BucketName"]
        self.role_arn = outputs_dictionary["RoleArn"]
        self.data_store_id = outputs_dictionary["DatastoreID"]
        return stack
```

Salin file DICOM ke bucket impor Amazon S3.

```
    def copy_single_object(self, key, source_bucket, target_bucket,
target_directory):
        """
        Copies a single object from a source to a target bucket.

        :param key: The key of the object to copy.
        :param source_bucket: The source bucket for the copy.
        :param target_bucket: The target bucket for the copy.
        :param target_directory: The target directory for the copy.
        """
        new_key = target_directory + "/" + key
        copy_source = {"Bucket": source_bucket, "Key": key}
        self.s3_client.copy_object(
            CopySource=copy_source, Bucket=target_bucket, Key=new_key
        )
        print(f"\n\t\tCopying {key}.")

    def copy_images(
        self, source_bucket, source_directory, target_bucket, target_directory
    ):
        """
```

```
        Copies the images from the source to the target bucket using multiple
 threads.

        :param source_bucket: The source bucket for the images.
        :param source_directory: Directory within the source bucket.
        :param target_bucket: The target bucket for the images.
        :param target_directory: Directory within the target bucket.
        """

        # Get list of all objects in source bucket.
        list_response = self.s3_client.list_objects_v2(
            Bucket=source_bucket, Prefix=source_directory
        )
        objs = list_response["Contents"]
        keys = [obj["Key"] for obj in objs]

        # Copy the objects in the bucket.
        for key in keys:
            self.copy_single_object(key, source_bucket, target_bucket,
 target_directory)

        print("\t\tDone copying all objects.")
```

Impor file DICOM ke penyimpanan data Amazon S3.

```
class MedicalImagingWrapper:
    """Encapsulates AWS HealthImaging functionality."""

    def __init__(self, medical_imaging_client, s3_client):
        """
        :param medical_imaging_client: A Boto3 Amazon MedicalImaging client.
        :param s3_client: A Boto3 S3 client.
        """
        self.medical_imaging_client = medical_imaging_client
        self.s3_client = s3_client

    @classmethod
    def from_client(cls):
        medical_imaging_client = boto3.client("medical-imaging")
```

```
        s3_client = boto3.client("s3")
        return cls(medical_imaging_client, s3_client)


    def start_dicom_import_job(
        self,
        data_store_id,
        input_bucket_name,
        input_directory,
        output_bucket_name,
        output_directory,
        role_arn,
    ):
        """
        Routine which starts a HealthImaging import job.

        :param data_store_id: The HealthImaging data store ID.
        :param input_bucket_name: The name of the Amazon S3 bucket containing the
DICOM files.
        :param input_directory: The directory in the S3 bucket containing the
DICOM files.
        :param output_bucket_name: The name of the S3 bucket for the output.
        :param output_directory: The directory in the S3 bucket to store the
output.
        :param role_arn: The ARN of the IAM role with permissions for the import.
        :return: The job ID of the import.
        """

        input_uri = f"s3://{input_bucket_name}/{input_directory}/"
        output_uri = f"s3://{output_bucket_name}/{output_directory}/"
        try:
            job = self.medical_imaging_client.start_dicom_import_job(
                jobName="examplejob",
                datastoreId=data_store_id,
                dataAccessRoleArn=role_arn,
                inputS3Uri=input_uri,
                outputS3Uri=output_uri,
            )
        except ClientError as err:
            logger.error(
                "Couldn't start DICOM import job. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
```

```
            raise
        else:
            return job["jobId"]
```

Dapatkan set gambar yang dibuat oleh pekerjaan impor DICOM.

```
class MedicalImagingWrapper:
    """Encapsulates AWS HealthImaging functionality."""

    def __init__(self, medical_imaging_client, s3_client):
        """
        :param medical_imaging_client: A Boto3 Amazon MedicalImaging client.
        :param s3_client: A Boto3 S3 client.
        """
        self.medical_imaging_client = medical_imaging_client
        self.s3_client = s3_client

    @classmethod
    def from_client(cls):
        medical_imaging_client = boto3.client("medical-imaging")
        s3_client = boto3.client("s3")
        return cls(medical_imaging_client, s3_client)


    def get_image_sets_for_dicom_import_job(self, datastore_id, import_job_id):
        """
        Retrieves the image sets created for an import job.

        :param datastore_id: The HealthImaging data store ID
        :param import_job_id: The import job ID
        :return: List of image set IDs
        """

        import_job = self.medical_imaging_client.get_dicom_import_job(
            datastoreId=datastore_id, jobId=import_job_id
        )

        output_uri = import_job["jobProperties"]["outputS3Uri"]
```

```
        bucket = output_uri.split("/")[2]
        key = "/".join(output_uri.split("/")[3:])

        # Try to get the manifest.
        retries = 3
        while retries > 0:
            try:
                obj = self.s3_client.get_object(
                    Bucket=bucket, Key=key + "job-output-manifest.json"
                )
                body = obj["Body"]
                break
            except ClientError as error:
                retries = retries - 1
                time.sleep(3)
        try:
            data = json.load(body)
            expression =
jmespath.compile("jobSummary.imageSetsSummary[].imageSetId")
            image_sets = expression.search(data)
        except json.decoder.JSONDecodeError as error:
            image_sets = import_job["jobProperties"]

        return image_sets


    def get_image_set(self, datastore_id, image_set_id, version_id=None):
        """
        Get the properties of an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The optional version of the image set.
        :return: The image set properties.
        """
        try:
            if version_id:
                image_set = self.medical_imaging_client.get_image_set(
                    imageSetId=image_set_id,
                    datastoreId=datastore_id,
                    versionId=version_id,
                )
            else:
                image_set = self.medical_imaging_client.get_image_set(
```

```
                    imageSetId=image_set_id, datastoreId=datastore_id
                )
        except ClientError as err:
            logger.error(
                "Couldn't get image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return image_set
```

Dapatkan informasi bingkai gambar untuk set gambar.

```
class MedicalImagingWrapper:
    """Encapsulates AWS HealthImaging functionality."""

    def __init__(self, medical_imaging_client, s3_client):
        """
        :param medical_imaging_client: A Boto3 Amazon MedicalImaging client.
        :param s3_client: A Boto3 S3 client.
        """
        self.medical_imaging_client = medical_imaging_client
        self.s3_client = s3_client

    @classmethod
    def from_client(cls):
        medical_imaging_client = boto3.client("medical-imaging")
        s3_client = boto3.client("s3")
        return cls(medical_imaging_client, s3_client)


    def get_image_frames_for_image_set(self, datastore_id, image_set_id,
 out_directory):
        """
        Get the image frames for an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
```

```
        :param out_directory: The directory to save the file.
        :return: The image frames.
        """
        image_frames = []
        file_name = os.path.join(out_directory,
f"{image_set_id}_metadata.json.gzip")
        file_name = file_name.replace("/", "\\\\")
        self.get_image_set_metadata(file_name, datastore_id, image_set_id)
        try:
            with gzip.open(file_name, "rb") as f_in:
                doc = json.load(f_in)
            instances = jmespath.search("Study.Series.*.Instances[].*[]", doc)
            for instance in instances:
                rescale_slope = jmespath.search("DICOM.RescaleSlope", instance)
                rescale_intercept = jmespath.search("DICOM.RescaleIntercept",
instance)
                image_frames_json = jmespath.search("ImageFrames[][]", instance)
                for image_frame in image_frames_json:
                    checksum_json = jmespath.search(
                        "max_by(PixelDataChecksumFromBaseToFullResolution,
&Width)",
                        image_frame,
                    )
                    image_frame_info = {
                        "imageSetId": image_set_id,
                        "imageFrameId": image_frame["ID"],
                        "rescaleIntercept": rescale_intercept,
                        "rescaleSlope": rescale_slope,
                        "minPixelValue": image_frame["MinPixelValue"],
                        "maxPixelValue": image_frame["MaxPixelValue"],
                        "fullResolutionChecksum": checksum_json["Checksum"],
                    }
                    image_frames.append(image_frame_info)
            return image_frames
        except TypeError:
            return {}
        except ClientError as err:
            logger.error(
                "Couldn't get image frames for image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        return image_frames
```

```
    def get_image_set_metadata(
        self, metadata_file, datastore_id, image_set_id, version_id=None
    ):
        """
        Get the metadata of an image set.

        :param metadata_file: The file to store the JSON gzipped metadata.
        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The version of the image set.
        """

        try:
            if version_id:
                image_set_metadata =
self.medical_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id,
                    datastoreId=datastore_id,
                    versionId=version_id,
                )
            else:
                image_set_metadata =
self.medical_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id, datastoreId=datastore_id
                )
            with open(metadata_file, "wb") as f:
                for chunk in
image_set_metadata["imageSetMetadataBlob"].iter_chunks():
                    if chunk:
                        f.write(chunk)

        except ClientError as err:
            logger.error(
                "Couldn't get image metadata. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Unduh, dekode, dan verifikasi bingkai gambar.

```
class MedicalImagingWrapper:
    """Encapsulates AWS HealthImaging functionality."""

    def __init__(self, medical_imaging_client, s3_client):
        """
        :param medical_imaging_client: A Boto3 Amazon MedicalImaging client.
        :param s3_client: A Boto3 S3 client.
        """
        self.medical_imaging_client = medical_imaging_client
        self.s3_client = s3_client

    @classmethod
    def from_client(cls):
        medical_imaging_client = boto3.client("medical-imaging")
        s3_client = boto3.client("s3")
        return cls(medical_imaging_client, s3_client)


    def get_pixel_data(
        self, file_path_to_write, datastore_id, image_set_id, image_frame_id
    ):
        """
        Get an image frame's pixel data.

        :param file_path_to_write: The path to write the image frame's HTJ2K
 encoded pixel data.
        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param image_frame_id: The ID of the image frame.
        """
        try:
            image_frame = self.medical_imaging_client.get_image_frame(
                datastoreId=datastore_id,
                imageSetId=image_set_id,
                imageFrameInformation={"imageFrameId": image_frame_id},
            )
            with open(file_path_to_write, "wb") as f:
                for chunk in image_frame["imageFrameBlob"].iter_chunks():
                    f.write(chunk)
        except ClientError as err:
```

```
            logger.error(
                "Couldn't get image frame. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def download_decode_and_check_image_frames(
        self, data_store_id, image_frames, out_directory
    ):
        """
        Downloads image frames, decodes them, and uses the checksum to validate
        the decoded images.

        :param data_store_id: The HealthImaging data store ID.
        :param image_frames: A list of dicts containing image frame information.
        :param out_directory: A directory for the downloaded images.
        :return: True if the function succeeded; otherwise, False.
        """
        total_result = True
        for image_frame in image_frames:
            image_file_path = f"{out_directory}/
image_{image_frame['imageFrameId']}.jph"
            self.get_pixel_data(
                image_file_path,
                data_store_id,
                image_frame["imageSetId"],
                image_frame["imageFrameId"],
            )

            image_array = self.jph_image_to_opj_bitmap(image_file_path)
            crc32_checksum = image_frame["fullResolutionChecksum"]
            # Verify checksum.
            crc32_calculated = zlib.crc32(image_array)
            image_result = crc32_checksum == crc32_calculated
            print(
                f"\t\tImage checksum verified for {image_frame['imageFrameId']}:
 {image_result }"
            )
            total_result = total_result and image_result
        return total_result

    @staticmethod
```

```
    def jph_image_to_opj_bitmap(jph_file):
        """
        Decode the image to a bitmap using an OPENJPEG library.
        :param jph_file: The file to decode.
        :return: The decoded bitmap as an array.
        """
        # Use format 2 for the JPH file.
        params = openjpeg.utils.get_parameters(jph_file, 2)
        print(f"\n\t\tImage parameters for {jph_file}: \n\t\t{params}")

        image_array = openjpeg.utils.decode(jph_file, 2)

        return image_array
```

Pembersihan sumber daya

```
    def destroy(self, stack):
        """
        Destroys the resources managed by the CloudFormation stack, and the
CloudFormation
        stack itself.

        :param stack: The CloudFormation stack that manages the example
resources.
        """

        print(f"\t\tCleaning up resources and {stack.name}.")
        data_store_id = None
        for oput in stack.outputs:
            if oput["OutputKey"] == "DatastoreID":
                data_store_id = oput["OutputValue"]
        if data_store_id is not None:
            print(f"\t\tDeleting image sets in data store {data_store_id}.")
            image_sets = self.medical_imaging_wrapper.search_image_sets(
                data_store_id, {}
            )
            image_set_ids = [image_set["imageSetId"] for image_set in image_sets]

            for image_set_id in image_set_ids:
                self.medical_imaging_wrapper.delete_image_set(
                    data_store_id, image_set_id
```

```
                )
                print(f"\t\tDeleted image set with id : {image_set_id}")

        print(f"\t\tDeleting {stack.name}.")
        stack.delete()
        print("\t\tWaiting for stack removal. This may take a few minutes.")
        waiter = self.cf_resource.meta.client.get_waiter("stack_delete_complete")
        waiter.wait(StackName=stack.name)
        print("\t\tStack delete complete.")




class MedicalImagingWrapper:
    """Encapsulates AWS HealthImaging functionality."""

    def __init__(self, medical_imaging_client, s3_client):
        """
        :param medical_imaging_client: A Boto3 Amazon MedicalImaging client.
        :param s3_client: A Boto3 S3 client.
        """
        self.medical_imaging_client = medical_imaging_client
        self.s3_client = s3_client

    @classmethod
    def from_client(cls):
        medical_imaging_client = boto3.client("medical-imaging")
        s3_client = boto3.client("s3")
        return cls(medical_imaging_client, s3_client)


    def search_image_sets(self, datastore_id, search_filter):
        """
        Search for image sets.

        :param datastore_id: The ID of the data store.
        :param search_filter: The search filter.
            For example: {"filters" : [{ "operator": "EQUAL", "values":
 [{"DICOMPatientId": "3524578"}]}]}.
        :return: The list of image sets.
        """
        try:
            paginator =
 self.medical_imaging_client.get_paginator("search_image_sets")
```

```
            page_iterator = paginator.paginate(
                datastoreId=datastore_id, searchCriteria=search_filter
            )
            metadata_summaries = []
            for page in page_iterator:
                metadata_summaries.extend(page["imageSetsMetadataSummaries"])
        except ClientError as err:
            logger.error(
                "Couldn't search image sets. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return metadata_summaries


    def delete_image_set(self, datastore_id, image_set_id):
        """
        Delete an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        """
        try:
            delete_results = self.medical_imaging_client.delete_image_set(
                imageSetId=image_set_id, datastoreId=datastore_id
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK untuk Python (Boto3).
  - DeleteImageSet
  - Dapatkan DICOMImport Job
  - GetImageFrame

- GetImageSetMetadata
- SearchImageSets
- Mulai DICOMImport Job

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

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

Menandai penyimpanan HealthImaging data menggunakan SDK AWS

Contoh kode berikut menunjukkan cara menandai penyimpanan HealthImaging data.

Java

SDK untuk Java 2.x

Untuk menandai penyimpanan data.

```
                final String datastoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

                TagResource.tagMedicalImagingResource(medicalImagingClient,
 datastoreArn,
                                ImmutableMap.of("Deployment", "Development"));
```

Fungsi utilitas untuk menandai sumber daya.

```
   public static void tagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
           String resourceArn,
           Map<String, String> tags) {
       try {
           TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                   .resourceArn(resourceArn)
```

```
                    .tags(tags)
                    .build();

            medicalImagingClient.tagResource(tagResourceRequest);

            System.out.println("Tags have been added to the resource.");
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

Untuk daftar tag untuk penyimpanan data.

```
                final String datastoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

                ListTagsForResourceResponse result =
 ListTagsForResource.listMedicalImagingResourceTags(
                                medicalImagingClient,
                                datastoreArn);
                if (result != null) {
                        System.out.println("Tags for resource: " +
 result.tags());
                }
```

Fungsi utilitas untuk daftar tag sumber daya.

```
    public static ListTagsForResourceResponse
 listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
            String resourceArn) {
        try {
            ListTagsForResourceRequest listTagsForResourceRequest =
 ListTagsForResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .build();

            return
 medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }

        return null;
    }
```

Untuk menghapus tag penyimpanan data.

```
                final String datastoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

                UntagResource.untagMedicalImagingResource(medicalImagingClient,
 datastoreArn,
                                Collections.singletonList("Deployment"));
```

Fungsi utilitas untuk membuka tag sumber daya.

```
   public static void untagMedicalImagingResource(MedicalImagingClient
 medicalImagingClient,
            String resourceArn,
            Collection<String> tagKeys) {
        try {
            UntagResourceRequest untagResourceRequest =
 UntagResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .tagKeys(tagKeys)
                    .build();

            medicalImagingClient.untagResource(untagResourceRequest);

            System.out.println("Tags have been removed from the resource.");
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Java 2.x .
  - ListTagsForResource

- TagResource
- UntagResource

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

JavaScript

SDK untuk JavaScript (v3)

Untuk menandai penyimpanan data.

```
  try {
    const datastoreArn =
      "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";
    const tags = {
      Deployment: "Development",
    };
    await tagResource(datastoreArn, tags);
  } catch (e) {
    console.log(e);
  }
```

Fungsi utilitas untuk menandai sumber daya.

```
import { TagResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data
 store or image set.
 * @param {Record<string,string>} tags - The tags to add to the resource as JSON.
 *                     - For example: {"Deployment" : "Development"}
 */
export const tagResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:xxxxxx:datastore/xxxxx/
imageset/xxx",
```

```
  tags = {},
) => {
  const response = await medicalImagingClient.send(
    new TagResourceCommand({ resourceArn: resourceArn, tags: tags }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //        httpStatusCode: 204,
  //         requestId: '8a6de9a3-ec8e-47ef-8643-473518b19d45',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //    }
  // }

  return response;
};
```

Untuk daftar tag untuk penyimpanan data.

```
  try {
    const datastoreArn =
      "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";
    const { tags } = await listTagsForResource(datastoreArn);
    console.log(tags);
  } catch (e) {
    console.log(e);
  }
```

Fungsi utilitas untuk daftar tag sumber daya.

```
import { ListTagsForResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data
 store or image set.
 */
```

```
export const listTagsForResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:abc:datastore/def/imageset/
ghi",
) => {
  const response = await medicalImagingClient.send(
    new ListTagsForResourceCommand({ resourceArn: resourceArn }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: '008fc6d3-abec-4870-a155-20fa3631e645',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     tags: { Deployment: 'Development' }
  // }

  return response;
};
```

Untuk menghapus tag penyimpanan data.

```
  try {
    const datastoreArn =
      "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";
    const keys = ["Deployment"];
    await untagResource(datastoreArn, keys);
  } catch (e) {
    console.log(e);
  }
```

Fungsi utilitas untuk membuka tag sumber daya.

```
import { UntagResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
```

```
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data
 store or image set.
 * @param {string[]} tagKeys - The keys of the tags to remove.
 */
export const untagResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:xxxxxx:datastore/xxxxx/
imageset/xxx",
  tagKeys = [],
) => {
  const response = await medicalImagingClient.send(
    new UntagResourceCommand({ resourceArn: resourceArn, tagKeys: tagKeys }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //        httpStatusCode: 204,
  //         requestId: '8a6de9a3-ec8e-47ef-8643-473518b19d45',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //    }
  // }

  return response;
};
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK untuk JavaScript .
  - ListTagsForResource
  - TagResource
  - UntagResource

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

Python

SDK untuk Python (Boto3)

Untuk menandai penyimpanan data.

```
    a_data_store_arn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012"

    medical_imaging_wrapper.tag_resource(data_store_arn, {"Deployment":
 "Development"})
```

Fungsi utilitas untuk menandai sumber daya.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def tag_resource(self, resource_arn, tags):
        """
        Tag a resource.

        :param resource_arn: The ARN of the resource.
        :param tags: The tags to apply.
        """
        try:
            self.health_imaging_client.tag_resource(resourceArn=resource_arn,
 tags=tags)
        except ClientError as err:
            logger.error(
                "Couldn't tag resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Untuk daftar tag untuk penyimpanan data.

```
    a_data_store_arn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012"

    medical_imaging_wrapper.list_tags_for_resource(data_store_arn)
```

Fungsi utilitas untuk daftar tag sumber daya.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def list_tags_for_resource(self, resource_arn):
        """
        List the tags for a resource.

        :param resource_arn: The ARN of the resource.
        :return: The list of tags.
        """
        try:
            tags = self.health_imaging_client.list_tags_for_resource(
                resourceArn=resource_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't list tags for resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return tags["tags"]
```

Untuk menghapus tag penyimpanan data.

```
    a_data_store_arn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012"

    medical_imaging_wrapper.untag_resource(data_store_arn, ["Deployment"])
```

Fungsi utilitas untuk membuka tag sumber daya.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def untag_resource(self, resource_arn, tag_keys):
        """
        Untag a resource.

        :param resource_arn: The ARN of the resource.
        :param tag_keys: The tag keys to remove.
        """
        try:
            self.health_imaging_client.untag_resource(
                resourceArn=resource_arn, tagKeys=tag_keys
            )
        except ClientError as err:
            logger.error(
                "Couldn't untag resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK untuk Python (Boto3).
  - ListTagsForResource
  - TagResource
  - UntagResource

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

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

Menandai set HealthImaging gambar menggunakan SDK AWS

Contoh kode berikut menunjukkan cara menandai set HealthImaging gambar.

Java

SDK untuk Java 2.x

Untuk menandai set gambar.

```
                final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";

                TagResource.tagMedicalImagingResource(medicalImagingClient,
 imageSetArn,
                                ImmutableMap.of("Deployment", "Development"));
```

Fungsi utilitas untuk menandai sumber daya.

```
   public static void tagMedicalImagingResource(MedicalImagingClient
 medicalImagingClient,
            String resourceArn,
            Map<String, String> tags) {
        try {
            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .tags(tags)
                    .build();
```

```
            medicalImagingClient.tagResource(tagResourceRequest);

            System.out.println("Tags have been added to the resource.");
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

Untuk mencantumkan tag untuk kumpulan gambar.

```
                final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";

                ListTagsForResourceResponse result =
 ListTagsForResource.listMedicalImagingResourceTags(
                                medicalImagingClient,
                                imageSetArn);
                if (result != null) {
                        System.out.println("Tags for resource: " +
 result.tags());
                }
```

Fungsi utilitas untuk daftar tag sumber daya.

```
    public static ListTagsForResourceResponse
 listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
            String resourceArn) {
        try {
            ListTagsForResourceRequest listTagsForResourceRequest =
 ListTagsForResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .build();

            return
 medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
        return null;
    }
```

Untuk menghapus tag set gambar.

```
                final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";

                UntagResource.untagMedicalImagingResource(medicalImagingClient,
 imageSetArn,
                                Collections.singletonList("Deployment"));
```

Fungsi utilitas untuk membuka tag sumber daya.

```
   public static void untagMedicalImagingResource(MedicalImagingClient
 medicalImagingClient,
            String resourceArn,
            Collection<String> tagKeys) {
        try {
            UntagResourceRequest untagResourceRequest =
 UntagResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .tagKeys(tagKeys)
                    .build();

            medicalImagingClient.untagResource(untagResourceRequest);

            System.out.println("Tags have been removed from the resource.");
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Java 2.x .
  - ListTagsForResource
  - TagResource

- UntagResource

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

JavaScript

SDK untuk JavaScript (v3)

Untuk menandai set gambar.

```
  try {
    const imagesetArn =
      "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";
    const tags = {
      Deployment: "Development",
    };
    await tagResource(imagesetArn, tags);
  } catch (e) {
    console.log(e);
  }
```

Fungsi utilitas untuk menandai sumber daya.

```
import { TagResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data
 store or image set.
 * @param {Record<string,string>} tags - The tags to add to the resource as JSON.
 *                     - For example: {"Deployment" : "Development"}
 */
export const tagResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:xxxxxx:datastore/xxxxx/
imageset/xxx",
```

```
  tags = {},
) => {
  const response = await medicalImagingClient.send(
    new TagResourceCommand({ resourceArn: resourceArn, tags: tags }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //        httpStatusCode: 204,
  //         requestId: '8a6de9a3-ec8e-47ef-8643-473518b19d45',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //    }
  // }

  return response;
};
```

Untuk mencantumkan tag untuk kumpulan gambar.

```
  try {
    const imagesetArn =
      "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";
    const { tags } = await listTagsForResource(imagesetArn);
    console.log(tags);
  } catch (e) {
    console.log(e);
  }
```

Fungsi utilitas untuk daftar tag sumber daya.

```
import { ListTagsForResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data
 store or image set.
```

```
 */
export const listTagsForResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:abc:datastore/def/imageset/
ghi",
) => {
  const response = await medicalImagingClient.send(
    new ListTagsForResourceCommand({ resourceArn: resourceArn }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: '008fc6d3-abec-4870-a155-20fa3631e645',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     tags: { Deployment: 'Development' }
  // }

  return response;
};
```

Untuk menghapus tag set gambar.

```
  try {
    const imagesetArn =
      "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";
    const keys = ["Deployment"];
    await untagResource(imagesetArn, keys);
  } catch (e) {
    console.log(e);
  }
```

Fungsi utilitas untuk membuka tag sumber daya.

```
import { UntagResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";
```

```
/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data
 store or image set.
 * @param {string[]} tagKeys - The keys of the tags to remove.
 */
export const untagResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:xxxxxx:datastore/xxxxx/
imageset/xxx",
  tagKeys = [],
) => {
  const response = await medicalImagingClient.send(
    new UntagResourceCommand({ resourceArn: resourceArn, tagKeys: tagKeys }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //        httpStatusCode: 204,
  //         requestId: '8a6de9a3-ec8e-47ef-8643-473518b19d45',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //    }
  // }

  return response;
};
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK untuk JavaScript .
  - ListTagsForResource
  - TagResource
  - UntagResource

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

Python

SDK untuk Python (Boto3)

Untuk menandai set gambar.

```
    an_image_set_arn = (
        "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/"
        "imageset/12345678901234567890123456789012"
    )

    medical_imaging_wrapper.tag_resource(image_set_arn, {"Deployment":
 "Development"})
```

Fungsi utilitas untuk menandai sumber daya.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def tag_resource(self, resource_arn, tags):
        """
        Tag a resource.

        :param resource_arn: The ARN of the resource.
        :param tags: The tags to apply.
        """
        try:
            self.health_imaging_client.tag_resource(resourceArn=resource_arn,
 tags=tags)
        except ClientError as err:
            logger.error(
                "Couldn't tag resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Untuk mencantumkan tag untuk kumpulan gambar.

```
    an_image_set_arn = (
        "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/"
        "imageset/12345678901234567890123456789012"
    )

    medical_imaging_wrapper.list_tags_for_resource(image_set_arn)
```

Fungsi utilitas untuk daftar tag sumber daya.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def list_tags_for_resource(self, resource_arn):
        """
        List the tags for a resource.

        :param resource_arn: The ARN of the resource.
        :return: The list of tags.
        """
        try:
            tags = self.health_imaging_client.list_tags_for_resource(
                resourceArn=resource_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't list tags for resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return tags["tags"]
```

Untuk menghapus tag set gambar.

```
    an_image_set_arn = (
        "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/"
        "imageset/12345678901234567890123456789012"
    )

    medical_imaging_wrapper.untag_resource(image_set_arn, ["Deployment"])
```

Fungsi utilitas untuk membuka tag sumber daya.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def untag_resource(self, resource_arn, tag_keys):
        """
        Untag a resource.

        :param resource_arn: The ARN of the resource.
        :param tag_keys: The tag keys to remove.
        """
        try:
            self.health_imaging_client.untag_resource(
                resourceArn=resource_arn, tagKeys=tag_keys
            )
        except ClientError as err:
            logger.error(
                "Couldn't untag resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Kode berikut membuat instance objek. MedicalImagingWrapper

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK untuk Python (Boto3).
  - ListTagsForResource
  - TagResource
  - UntagResource

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

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

Pemantauan AWS HealthImaging

Pemantauan dan pencatatan adalah bagian penting dalam menjaga keamanan, keandalan, ketersediaan, dan kinerja AWS HealthImaging. AWS menyediakan alat pencatatan dan pemantauan berikut untuk menonton HealthImaging, 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

Menggunakan AWS CloudTrail dengan HealthImaging

HealthImaging AWS terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di HealthImaging. CloudTrail menangkap semua panggilan API untuk HealthImaging sebagai peristiwa. Panggilan yang diambil termasuk panggilan dari HealthImaging konsol dan panggilan kode ke operasi HealthImaging API. Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman CloudTrail acara secara berkelanjutan ke bucket Amazon S3, termasuk acara untuk. HealthImaging 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 HealthImaging, alamat IP dari mana permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan.

Untuk mempelajari selengkapnya CloudTrail, lihat Panduan AWS CloudTrail Pengguna.

Membuat jejak

CloudTrail diaktifkan untuk Anda Akun AWS saat Anda membuat akun. Ketika aktivitas terjadi di HealthImaging, 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 Anda. Untuk informasi selengkapnya, lihat Melihat peristiwa dengan Riwayat CloudTrail acara.

> **Note**
>
> Untuk melihat riwayat CloudTrail peristiwa AWS HealthImaging di Konsol Manajemen AWS, buka menu atribut Pencarian, pilih Sumber peristiwa, dan pilih`medical-imaging.amazonaws.com`.

Untuk catatan acara yang sedang berlangsung di Anda Akun AWS, termasuk acara untuk HealthImaging, buat jejak. Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon S3. Secara default, saat Anda membuat jejak di konsol, jejak tersebut berlaku untuk 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
- CloudTraillayanan 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

> **Note**
>
> AWS HealthImaging mendukung dua jenis CloudTrail peristiwa — peristiwa manajemen dan peristiwa data. Peristiwa manajemen adalah peristiwa umum yang dihasilkan oleh setiap AWS layanan, termasuk HealthImaging. Secara default, logging diterapkan ke peristiwa manajemen untuk setiap panggilan HealthImaging API yang mengaktifkannya. Peristiwa data dapat ditagih dan umumnya dicadangkan untuk APIs yang memiliki transaksi per detik (tps) tinggi, sehingga Anda dapat memilih untuk tidak memiliki CloudTrail log untuk tujuan biaya. Dengan HealthImaging, semua tindakan API asli yang tercantum dalam AWS HealthImaging API Referensi diklasifikasikan sebagai peristiwa manajemen dengan pengecualian `GetImageFrame`. `GetImageFrame`Tindakan ini dionboard dengan CloudTrail sebagai peristiwa data dan oleh karena itu harus diaktifkan. Untuk informasi selengkapnya, lihat Mencatat peristiwa data dalam AWS CloudTrail Panduan Pengguna.
> DICOMweb Tindakan API WADO-RS diklasifikasikan sebagai peristiwa data di CloudTrail, oleh karena itu, Anda harus ikut serta. Untuk informasi selengkapnya, lihat Mengambil data DICOM dari HealthImaging dan Mencatat peristiwa data di Panduan AWS CloudTrail Pengguna.

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 selengkapnya, lihat elemen CloudTrail `userIdentity`.

Memahami entri 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 manapun 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 untuk HealthImaging yang menunjukkan `GetDICOMImportJob` tindakan.


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "XXXXXXXXXXXXXXXXXXXXX:ce6d90ba-5fba-4456-a7bc-f9bc877597c3",
        "arn": "arn:aws:sts::123456789012:assumed-role/TestAccessRole/
ce6d90ba-5fba-4456-a7bc-f9bc877597c3"
        "accountId": "123456789012",
        "accessKeyId": "XXXXXXXXXXXXXXXXXXXX",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "XXXXXXXXXXXXXXXXXXXXX",
                "arn": "arn:aws:iam::123456789012:role/TestAccessRole",
                "accountId": "123456789012",
                "userName": "TestAccessRole"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-10-28T15:52:42Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-10-28T16:02:30Z",
    "eventSource": "medical-imaging.amazonaws.com",
    "eventName": "GetDICOMImportJob",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
```

```
    "userAgent": "aws-sdk-java/2.18.1 Linux/5.4.209-129.367.amzn2int.x86_64 OpenJDK_64-
Bit_Server_VM/11.0.17+9-LTS Java/11.0.17 vendor/Amazon.com_Inc. md/internal io/sync
 http/Apache cfg/retry-mode/standard",
    "requestParameters": {
        "jobId": "5d08d05d6aab2a27922d6260926077d4",
        "datastoreId": "12345678901234567890123456789012"
    },
    "responseElements": null,
    "requestID": "922f5304-b39f-4034-9d2e-f062de092a44",
    "eventID": "26307f73-07f4-4276-b379-d362aa303b22",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "824333766656",
    "eventCategory": "Management"
}
```


Menggunakan Amazon CloudWatch dengan HealthImaging

Anda dapat memantau HealthImaging penggunaan AWS 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.

HealthImaging menerbitkan jenis metrik berikut ke CloudWatch, di namespace: `AWS/HealthImaging`

- Metrik API - Jumlah panggilan untuk HealthImaging operasi API
- HealthImaging metrik - Penyimpanan data dan penggunaan sumber daya tingkat akun

> **Note**
>
> - Metrik dilaporkan untuk sebagian besar HealthImaging APIs.
> - HealthImaging metrik hanya tersedia untuk penyimpanan data yang dibuat setelah 9 Februari 2026, atau dengan mengajukan Kasus dukungan.

Metrik AWS HealthImaging API

Tabel berikut mencantumkan metrik dan dimensi untuk HealthImaging. Masing-masing disajikan sebagai jumlah frekuensi untuk rentang data yang ditentukan pengguna.

Metrik

Metrik-metrik	Deskripsi
`CallCount`	Jumlah panggilan ke APIs. Ini dapat dilaporkan baik untuk akun atau penyimpanan data tertentu. Unit: Hitungan Statistik yang Valid: Jumlah, Hitung Dimensi: Operasi, ID penyimpanan data, tipe penyimpanan data

Anda bisa mendapatkan metrik HealthImaging dengan Konsol Manajemen AWS, 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. HealthImaging Konsol menampilkan grafik berdasarkan data mentah dari CloudWatch API.

Anda harus memiliki CloudWatch izin yang sesuai untuk dipantau HealthImaging . CloudWatch Untuk informasi selengkapnya, lihat Identitas dan manajemen akses CloudWatch di Panduan CloudWatch Pengguna.

HealthImaging Metrik AWS

`AWS/HealthImaging`Namespace mencakup metrik berikut di tingkat akun dan penyimpanan data.

Metrik Tingkat Akun

Metrik tingkat akun memberikan visibilitas gabungan di semua penyimpanan data di akun Anda.

Metrik Tingkat Akun

Metrik	Deskripsi
`DataStoreCount`	Jumlah penyimpanan data dengan status aktif. Unit: Count (Jumlah) Statistik yang valid: Jumlah, Rata-rata
`ImageSetCount`	Jumlah total kumpulan gambar di semua penyimpanan data. Unit: Count (Jumlah) Statistik yang valid: Jumlah, Rata-rata
`StorageBytes`	Jumlah data dalam byte yang disimpan di semua penyimpanan data dalam tingkatan penyimpanan berikut: • Akses yang Sering (`FrequentAccessStorage`) • Arsipkan Akses Instan (`ArchiveInstantAccessStorage`) Nilai ini dihitung dengan menjumlahkan ukuran semua set gambar di semua penyimpanan data. Filter tingkat penyimpanan yang valid (Lihat `StorageTier` dimensinya): • `FrequentAccessStorage` • `ArchiveInstantAccessStorage` • `AllStorage` Unit: Byte

Metrik	Deskripsi
	Statistik yang valid: Jumlah, Rata-rata

Metrik Tingkat Penyimpanan Data

Metrik tingkat penyimpanan data memberikan visibilitas terperinci ke penyimpanan data individual.

Metrik Tingkat Penyimpanan Data

Metrik	Deskripsi
`TotalImageSetCount`	Jumlah total set gambar di penyimpanan data. Unit: Count (Jumlah) Statistik yang valid: Jumlah, Rata-rata
`PrimaryImageSetCount`	Jumlah set gambar utama di datastore. Unit: Count (Jumlah) Statistik yang valid: Jumlah, Rata-rata
`SmallImageSetCount`	Jumlah set gambar kurang dari 5MB di datastore. Unit: Count (Jumlah) Statistik yang valid: Jumlah, Rata-rata
`StorageBytes`	Jumlah data dalam byte yang disimpan di semua penyimpanan data dalam tingkatan penyimpanan berikut: • Akses yang Sering (`FrequentAccessStorage`) • Arsipkan Akses Instan (`ArchiveInstantAccessStorage`)

Metrik	Deskripsi
	Nilai ini dihitung dengan menjumlahkan ukuran semua set gambar di penyimpanan data. Filter tingkat penyimpanan yang valid (Lihat `StorageTier` dimensinya): • FrequentAccessStorage • ArchiveInstantAccessStorage • AllStorage Unit: Byte Statistik yang valid: Jumlah, Rata-rata
`DICOMStudyCount`	Jumlah studi DICOM di datastore. Unit: Count (Jumlah) Statistik yang valid: Jumlah, Rata-rata
`DICOMSeriesCount`	Jumlah seri DICOM di datastore. Unit: Count (Jumlah) Statistik yang valid: Jumlah, Rata-rata
`DICOMInstanceCount`	Jumlah instance DICOM di datastore. Unit: Count (Jumlah) Statistik yang valid: Jumlah, Rata-rata
`StructuredStorageBytes`	Jumlah penyimpanan terstruktur dalam byte yang diindeks oleh penyimpanan data. Unit: Byte Statistik yang valid: Jumlah, Rata-rata

HealthImaging Dimensi di CloudWatch

Dimensi berikut digunakan untuk memfilter HealthImaging metrik.

Dimensi

Dimensi	Deskripsi
`AccountId`	Dimensi ini menyaring data untuk AWS akun yang diidentifikasi.
`DatastoreId`	Dimensi ini menyaring data untuk penyimpanan data yang diidentifikasi saja.
`StorageTier`	Dimensi ini menyaring data dengan tingkatan penyimpanan berikut: • `FrequentAccessStorage` — Jumlah byte yang digunakan untuk set gambar di penyimpanan Frequent Access. • `ArchiveInstantAccessStorage` — Jumlah byte yang digunakan untuk set gambar di penyimpanan Akses Instan Arsip. • `AllStorage` — Jumlah total byte di semua tingkatan penyimpanan.

Mengakses Metrik HealthImaging

Anda dapat mengakses HealthImaging metrik menggunakan:

- Konsol Manajemen AWS- Lihat metrik di konsol CloudWatch
- AWS CLI- Gunakan perintah CloudWatch CLI
- CloudWatch API - Akses melalui AWS SDKs atau alat CloudWatch API

Anda harus memiliki izin yang sesuai untuk dipantau HealthImaging . CloudWatch Untuk informasi selengkapnya, lihat Identitas dan manajemen akses CloudWatch di Panduan CloudWatch Pengguna.

Menyiapkan HealthImaging Metrik

Untuk menerima HealthImaging metrik di CloudWatch akun Anda, Anda perlu membuat peran terkait layanan yang memungkinkan HealthImaging untuk mempublikasikan metrik atas nama Anda. Lihat Menggunakan peran terkait layanan HealthImaging untuk mengetahui detail tentang cara membuat peran terkait layanan.

Melihat HealthImaging Metrik

Untuk melihat metrik (CloudWatch konsol)

1. Masuk ke Konsol Manajemen AWS dan buka CloudWatchkonsol.
2. Pilih Metrik, pilih Semua Metrik, lalu pilih. **`AWS/HealthImaging`**
3. Pilih dimensi:
   - Berdasarkan Id Akun - Lihat metrik tingkat akun
   - By Datastore Id - Lihat metrik tingkat penyimpanan data
4. Pilih nama metrik, lalu pilih Tambahkan ke grafik.
5. Pilih nilai untuk rentang tanggal, dan jumlah metrik akan ditampilkan.

Membuat alarm menggunakan CloudWatch

CloudWatch Alarm mengawasi satu metrik selama periode waktu tertentu, dan melakukan satu atau beberapa tindakan: mengirim pemberitahuan ke topik Simple Notification Service Amazon (Amazon 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 Amazon SNS saat alarm berubah status.

CloudWatch alarm memanggil tindakan hanya ketika status berubah dan telah bertahan selama periode yang Anda tentukan. Untuk informasi selengkapnya, lihat Menggunakan CloudWatch alarm.

Menggunakan Amazon EventBridge dengan HealthImaging

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 merutekan peristiwa ke target. AWS HealthImaging menyediakan pengiriman perubahan status yang tahan lama ke EventBridge. Untuk informasi selengkapnya, lihat Apa itu Amazon EventBridge? di Panduan EventBridge Pengguna Amazon.

Topik

- HealthImaging peristiwa dikirim ke EventBridge
- HealthImaging struktur acara dan contoh

HealthImaging peristiwa dikirim ke EventBridge

Tabel berikut mencantumkan semua HealthImaging peristiwa yang dikirim EventBridge untuk diproses.

HealthImaging jenis acara	Status
Acara penyimpanan data	
Membuat Toko Data	`CREATING`
Pembuatan Penyimpanan Data Gagal	`CREATE_FAILED`
Toko Data Dibuat	`ACTIVE`
Penghapusan Toko Data	`DELETING`
Toko Data Dihapus	`DELETED`
Untuk informasi selengkapnya, lihat DataStorestatus di AWS API Referensi. HealthImaging	
Impor acara pekerjaan	
Impor Job yang Dikirim	`SUBMITTED`
Impor Job Sedang Berlangsung	`IN_PROGRESS`
Impor Job Selesai	`COMPLETED`
Impor Job Gagal	`FAILED`
Untuk informasi selengkapnya, lihat JobStatus di AWS HealthImaging API Referensi.	

HealthImaging jenis acara	Status
Acara set gambar	
Set Gambar Dibuat	`CREATED`
Menyalin Set Gambar	`COPYING`
Set Gambar Menyalin Dengan Akses Hanya Baca	`COPYING_WITH_READ_ONLY_ACCESS`
Set Gambar Disalin	`COPIED`
Salinan Set Gambar Gagal	`COPY_FAILED`
Pembaruan Set Gambar	`UPDATING`
Set Gambar Diperbarui	`UPDATED`
Pembaruan Set Gambar Gagal	`UPDATE_FAILED`
Menghapus Set Gambar	`DELETING`
Set Gambar Dihapus	`DELETED`
Untuk informasi selengkapnya, lihat ImageSetWorkflowStatusdi AWS HealthImaging API Referensi.	

HealthImaging struktur acara dan contoh

HealthImaging 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 Referensi struktur acara di Panduan EventBridge Pengguna Amazon.

> **Note**
>
> `source`Atribut untuk struktur HealthImaging acara adalah`aws.medical-imaging`.

Acara penyimpanan data

Data Store Creating

Negara - **CREATING**

```
{
    "version": "0",
    "id": "7cf0fb1c-8720-4d48-baa3-6eb97b35a10e",
    "detail-type": "Data Store Creating",
    "source": "aws.medical-imaging",
    "account": "111122223333",
    "time": "2024-03-14T00:01:00Z",
    "region": "us-west-2",
    "resources": ["arn:aws:medical-imaging:us-west-2:111122223333:datastore/
bbc4f3cccbae4095a34170fddc19b13d"],
    "detail": {
        "imagingVersion": "1.0",
        "datastoreId" : "bbc4f3cccbae4095a34170fddc19b13d",
        "datastoreName": "test",
        "datastoreStatus": "CREATING"
    }
}
```

Data Store Creation Failed

Negara - **CREATE_FAILED**

```
{
    "version": "0",
    "id": "7cf0fb1c-8720-4d48-baa3-6eb97b35a10e",
    "detail-type": "Data Store Creation Failed",
    "source": "aws.medical-imaging",
    "account": "111122223333",
    "time": "2024-03-14T00:01:00Z",
    "region": "us-west-2",
    "resources": ["arn:aws:medical-imaging:us-west-2:111122223333:datastore/
bbc4f3cccbae4095a34170fddc19b13d"],
    "detail": {
        "imagingVersion": "1.0",
        "datastoreId" : "bbc4f3cccbae4095a34170fddc19b13d",
        "datastoreName": "test",
        "datastoreStatus": "CREATE_FAILED"
```

```
    }
}
```

Data Store Created

Negara - **ACTIVE**

```
{
    "version": "0",
    "id": "7cf0fb1c-8720-4d48-baa3-6eb97b35a10e",
    "detail-type": "Data Store Created",
    "source": "aws.medical-imaging",
    "account": "111122223333",
    "time": "2024-03-14T00:01:00Z",
    "region": "us-west-2",
    "resources": ["arn:aws:medical-imaging:us-west-2:111122223333:datastore/
bbc4f3cccbae4095a34170fddc19b13d"],
    "detail": {
        "imagingVersion": "1.0",
        "datastoreId" : "bbc4f3cccbae4095a34170fddc19b13d",
        "datastoreName": "test",
        "datastoreStatus": "ACTIVE"
    }
}
```

Data Store Deleting

Negara - **DELETING**

```
{
    "version": "0",
    "id": "7cf0fb1c-8720-4d48-baa3-6eb97b35a10e",
    "detail-type": "Data Store Deleting",
    "source": "aws.medical-imaging",
    "account": "111122223333",
    "time": "2024-03-14T00:01:00Z",
    "region": "us-west-2",
    "resources": ["arn:aws:medical-imaging:us-west-2:111122223333:datastore/
bbc4f3cccbae4095a34170fddc19b13d"],
    "detail": {
        "imagingVersion": "1.0",
        "datastoreId" : "bbc4f3cccbae4095a34170fddc19b13d",
        "datastoreName": "test",
```

```
        "datastoreStatus": "DELETING"
    }
}
```

Data Store Deleted

Negara - **DELETED**

```
{
    "version": "0",
    "id": "7cf0fb1c-8720-4d48-baa3-6eb97b35a10e",
    "detail-type": "Data Store Deleted",
    "source": "aws.medical-imaging",
    "account": "111122223333",
    "time": "2024-03-14T00:01:00Z",
    "region": "us-west-2",
    "resources": ["arn:aws:medical-imaging:us-west-2:111122223333:datastore/
bbc4f3cccbae4095a34170fddc19b13d"],
    "detail": {
        "imagingVersion": "1.0",
        "datastoreId" : "bbc4f3cccbae4095a34170fddc19b13d",
        "datastoreName": "test",
        "datastoreStatus": "DELETED"
    }
}
```

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.

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.imagingVersion`	string	ID versi yang melacak perubahan skema detail acara. HealthImaging
`detail.datastoreId`	string	ID penyimpanan data yang terkait dengan peristiwa perubahan status.
`detail.datastoreName`	string	Nama penyimpanan data.
`detail.datastoreStatus`	string	Status penyimpanan data saat ini.

Impor acara pekerjaan

Import Job Submitted

Negara - **`SUBMITTED`**

```
{
    "version": "0",
    "id": "7cf0fb1c-8720-4d48-baa3-6eb97b35a10e",
    "detail-type": "Import Job Submitted",
```

```
    "source": "aws.medical-imaging",
    "account": "111122223333",
    "time": "2024-03-14T00:01:00Z",
    "region": "us-west-2",
    "resources": ["arn:aws:medical-imaging:us-west-2:111122223333:datastore/
bbc4f3cccbae4095a34170fddc19b13d"],
    "detail": {
        "imagingVersion": "1.0",
        "datastoreId" : "bbc4f3cccbae4095a34170fddc19b13d",
        "jobId": "a6a1d220f152e7aab6d8925d995d8b76",
        "jobName": "test_only_1",
        "jobStatus": "SUBMITTED",
        "inputS3Uri": "s3://healthimaging-test-bucket/input/",
        "outputS3Uri": "s3://healthimaging-test-bucket/output/"
    }
}
```

Import Job In Progress

Negara - **IN_PROGRESS**

```
{
    "version": "0",
    "id": "7cf0fb1c-8720-4d48-baa3-6eb97b35a10e",
    "detail-type": "Import Job In Progress",
    "source": "aws.medical-imaging",
    "account": "111122223333",
    "time": "2024-03-14T00:01:00Z",
    "region": "us-west-2",
    "resources": ["arn:aws:medical-imaging:us-west-2:111122223333:datastore/
bbc4f3cccbae4095a34170fddc19b13d"],
    "detail": {
        "imagingVersion": "1.0",
        "datastoreId" : "bbc4f3cccbae4095a34170fddc19b13d",
        "jobId": "a6a1d220f152e7aab6d8925d995d8b76",
        "jobName": "test_only_1",
        "jobStatus": "IN_PROGRESS",
        "inputS3Uri": "s3://healthimaging-test-bucket/input/",
        "outputS3Uri": "s3://healthimaging-test-bucket/output/"
    }
}
```

Import Job Completed

Negara - **COMPLETED**

```
{
    "version": "0",
    "id": "7cf0fb1c-8720-4d48-baa3-6eb97b35a10e",
    "detail-type": "Import Job Completed",
    "source": "aws.medical-imaging",
    "account": "111122223333",
    "time": "2024-03-14T00:01:00Z",
    "region": "us-west-2",
    "resources": ["arn:aws:medical-imaging:us-west-2:111122223333:datastore/
bbc4f3cccbae4095a34170fddc19b13d"],
    "detail": {
        "imagingVersion": "1.0",
        "datastoreId" : "bbc4f3cccbae4095a34170fddc19b13d",
        "jobId": "a6a1d220f152e7aab6d8925d995d8b76",
        "jobName": "test_only_1",
        "jobStatus": "COMPLETED",
        "inputS3Uri": "s3://healthimaging-test-bucket/input/",
        "outputS3Uri": "s3://healthimaging-test-bucket/output/"
    }
}
```

Import Job Failed

Negara - **FAILED**

```
{
    "version": "0",
    "id": "7cf0fb1c-8720-4d48-baa3-6eb97b35a10e",
    "detail-type": "Import Job Failed",
    "source": "aws.medical-imaging",
    "account": "111122223333",
    "time": "2024-03-14T00:01:00Z",
    "region": "us-west-2",
    "resources": ["arn:aws:medical-imaging:us-west-2:111122223333:datastore/
bbc4f3cccbae4095a34170fddc19b13d"],
    "detail": {
        "imagingVersion": "1.0",
        "datastoreId" : "bbc4f3cccbae4095a34170fddc19b13d",
        "jobId": "a6a1d220f152e7aab6d8925d995d8b76",
        "jobName": "test_only_1",
```

```
        "jobStatus": "FAILED",
        "inputS3Uri": "s3://healthimaging-test-bucket/input/",
        "outputS3Uri": "s3://healthimaging-test-bucket/output/"
    }
}
```

Impor 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.
`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.imagingVersion`	string	ID versi yang melacak perubahan skema detail acara. HealthImaging

Nama	Tipe	Deskripsi
`detail.datastoreId`	string	Penyimpanan data yang menghasilkan peristiwa perubahan status.
`detail.jobId`	string	ID pekerjaan impor yang terkait dengan peristiwa perubahan status.
`detail.jobName`	string	Nama pekerjaan impor.
`detail.jobStatus`	string	Status pekerjaan saat ini.
`detail.inputS3Uri`	string	Jalur awalan input untuk bucket S3 yang berisi file DICOM yang akan diimpor.
`detail.outputS3Uri`	string	Awalan keluaran bucket S3 tempat hasil pekerjaan impor DICOM akan diunggah.

Acara set gambar

Image Set Created

Negara - **`CREATED`**


```
{
   "version": "0",
   "id": "7cf0fb1c-8720-4d48-baa3-6eb97b35a10e",
   "detail-type": "Image Set Created",
   "source": "aws.medical-imaging",
   "account": "111122223333",
   "time": "2024-03-14T00:01:00Z",
   "region": "us-west-2",
   "resources": ["arn:aws:medical-imaging:us-west-2:846006145877:datastore/
bbc4f3cccbae4095a34170fddc19b13d/imageset/207284eef860ac01c4b2a8de27a6fc11"],
   "detail": {
      "imagingVersion": "1.0",
```

```
      "isPrimary": true,
      "imageSetVersion": "1",
      "datastoreId": "bbc4f3cccbae4095a34170fddc19b13d",
      "imagesetId": "5b3a711878c34d40e888253319388649",
      "imageSetState": "ACTIVE",
      "imageSetWorkflowStatus": "CREATED"
   }
}
```

Image Set Copying

Negara - **COPYING**

```
{
   "version": "0",
   "id": "7cf0fb1c-8720-4d48-baa3-6eb97b35a10e",
   "detail-type": "Image Set Copying",
   "source": "aws.medical-imaging",
   "account": "111122223333",
   "time": "2024-03-14T00:01:00Z",
   "region": "us-west-2",
   "resources": ["arn:aws:medical-imaging:us-west-2:846006145877:datastore/
bbc4f3cccbae4095a34170fddc19b13d/imageset/207284eef860ac01c4b2a8de27a6fc11"],
   "detail": {
      "imagingVersion": "1.0",
      "isPrimary": true,
      "imageSetVersion": "1",
      "datastoreId": "bbc4f3cccbae4095a34170fddc19b13d",
      "imagesetId": "5b3a711878c34d40e888253319388649",
      "imageSetState": "LOCKED",
      "imageSetWorkflowStatus": "COPYING",
      "sourceImageSetArn": "arn:aws:medical-imaging:us-
west-2:147997158357:datastore/c381ee9b9ef34902a45b476dd7be068b/
imageset/0309de3674fd551fa7ddd2880b21f990"
   }
}
```

Image Set Copying With Read Only Access

Negara - **COPYING_WITH_READ_ONLY_ACCESS**

```
{
   "version": "0",
```

```
   "id": "7cf0fb1c-8720-4d48-baa3-6eb97b35a10e",
   "detail-type": "Image Set Copying With Read Only Access",
   "source": "aws.medical-imaging",
   "account": "111122223333",
   "time": "2024-03-14T00:01:00Z",
   "region": "us-west-2",
   "resources": ["arn:aws:medical-imaging:us-west-2:846006145877:datastore/
bbc4f3cccbae4095a34170fddc19b13d/imageset/207284eef860ac01c4b2a8de27a6fc11"],
   "detail": {
      "imagingVersion": "1.0",
      "isPrimary": true,
      "imageSetVersion": "1",
      "datastoreId": "bbc4f3cccbae4095a34170fddc19b13d",
      "imagesetId": "5b3a711878c34d40e888253319388649",
      "imageSetState": "LOCKED",
      "imageSetWorkflowStatus": "COPYING_WITH_READ_ONLY_ACCESS"
   }
}
```

Image Set Copied

Negara - **COPIED**

```
{
   "version": "0",
   "id": "7cf0fb1c-8720-4d48-baa3-6eb97b35a10e",
   "detail-type": "Image Set Copied",
   "source": "aws.medical-imaging",
   "account": "111122223333",
   "time": "2024-03-14T00:01:00Z",
   "region": "us-west-2",
   "resources": ["arn:aws:medical-imaging:us-west-2:846006145877:datastore/
bbc4f3cccbae4095a34170fddc19b13d/imageset/207284eef860ac01c4b2a8de27a6fc11"],
   "detail": {
      "imagingVersion": "1.0",
      "isPrimary": true,
      "imageSetVersion": "1",
      "datastoreId": "bbc4f3cccbae4095a34170fddc19b13d",
      "imagesetId": "5b3a711878c34d40e888253319388649",
      "imageSetState": "ACTIVE",
      "imageSetWorkflowStatus": "COPIED"
   }
}
```

Image Set Copy Failed

Negara - **COPY_FAILED**

```
{
   "version": "0",
   "id": "7cf0fb1c-8720-4d48-baa3-6eb97b35a10e",
   "detail-type": "Image Set Copy Failed",
   "source": "aws.medical-imaging",
   "account": "111122223333",
   "time": "2024-03-14T00:01:00Z",
   "region": "us-west-2",
   "resources": ["arn:aws:medical-imaging:us-west-2:846006145877:datastore/
bbc4f3cccbae4095a34170fddc19b13d/imageset/207284eef860ac01c4b2a8de27a6fc11"],
   "detail": {
      "imagingVersion": "1.0",
      "isPrimary": true,
      "imageSetVersion": "1",
      "datastoreId": "bbc4f3cccbae4095a34170fddc19b13d",
      "imagesetId": "5b3a711878c34d40e8882533193886 49",
      "imageSetState": "ACTIVE",
      "imageSetWorkflowStatus": "COPY_FAILED"
   }
}
```

Image Set Updating

Negara - **UPDATING**

```
{
   "version": "0",
   "id": "7cf0fb1c-8720-4d48-baa3-6eb97b35a10e",
   "detail-type": "Image Set Updating",
   "source": "aws.medical-imaging",
   "account": "111122223333",
   "time": "2024-03-14T00:01:00Z",
   "region": "us-west-2",
   "resources": ["arn:aws:medical-imaging:us-west-2:846006145877:datastore/
bbc4f3cccbae4095a34170fddc19b13d/imageset/207284eef860ac01c4b2a8de27a6fc11"],
   "detail": {
      "imagingVersion": "1.0",
      "isPrimary": true,
      "imageSetVersion": "1",
      "datastoreId": "bbc4f3cccbae4095a34170fddc19b13d",
```

```
      "imagesetId": "5b3a711878c34d40e888253319388649",
      "imageSetState": "LOCKED",
      "imageSetWorkflowStatus": "UPDATING"
   }
}
```

Image Set Updated

Negara - **UPDATED**

```
{
   "version": "0",
   "id": "7cf0fb1c-8720-4d48-baa3-6eb97b35a10e",
   "detail-type": "Image Set Updated",
   "source": "aws.medical-imaging",
   "account": "111122223333",
   "time": "2024-03-14T00:01:00Z",
   "region": "us-west-2",
   "resources": ["arn:aws:medical-imaging:us-west-2:846006145877:datastore/
bbc4f3cccbae4095a34170fddc19b13d/imageset/207284eef860ac01c4b2a8de27a6fc11"],
   "detail": {
      "imagingVersion": "1.0",
      "isPrimary": true,
      "imageSetVersion": "1",
      "datastoreId": "bbc4f3cccbae4095a34170fddc19b13d",
      "imagesetId": "5b3a711878c34d40e888253319388649",
      "imageSetState": "ACTIVE",
      "imageSetWorkflowStatus": "UPDATED"
   }
}
```

Image Set Update Failed

Negara - **UPDATE_FAILED**

```
{
   "version": "0",
   "id": "7cf0fb1c-8720-4d48-baa3-6eb97b35a10e",
   "detail-type": "Image Set Update Failed",
   "source": "aws.medical-imaging",
   "account": "111122223333",
   "time": "2024-03-14T00:01:00Z",
   "region": "us-west-2",
```

```
   "resources": ["arn:aws:medical-imaging:us-west-2:846006145877:datastore/
bbc4f3cccbae4095a34170fddc19b13d/imageset/207284eef860ac01c4b2a8de27a6fc11"],
   "detail": {
      "imagingVersion": "1.0",
      "isPrimary": true,
      "imageSetVersion": "1",
      "datastoreId": "bbc4f3cccbae4095a34170fddc19b13d",
      "imagesetId": "5b3a711878c34d40e888253319388649",
      "imageSetState": "ACTIVE",
      "imageSetWorkflowStatus": "UPDATE_FAILED"
   }
}
```

Image Set Deleting

Negara - **DELETING**

```
{
   "version": "0",
   "id": "7cf0fb1c-8720-4d48-baa3-6eb97b35a10e",
   "detail-type": "Image Set Deleting",
   "source": "aws.medical-imaging",
   "account": "111122223333",
   "time": "2024-03-14T00:01:00Z",
   "region": "us-west-2",
   "resources": ["arn:aws:medical-imaging:us-west-2:846006145877:datastore/
bbc4f3cccbae4095a34170fddc19b13d/imageset/207284eef860ac01c4b2a8de27a6fc11"],
   "detail": {
      "imagingVersion": "1.0",
      "isPrimary": true,
      "imageSetVersion": "1",
      "datastoreId": "bbc4f3cccbae4095a34170fddc19b13d",
      "imagesetId": "5b3a711878c34d40e888253319388649",
      "imageSetState": "LOCKED",
      "imageSetWorkflowStatus": "DELETING"
   }
}
```

Image Set Deleted

Negara - **DELETED**

```
{
```

```
   "version": "0",
   "id": "7cf0fb1c-8720-4d48-baa3-6eb97b35a10e",
   "detail-type": "Image Set Deleted",
   "source": "aws.medical-imaging",
   "account": "111122223333",
   "time": "2024-03-14T00:01:00Z",
   "region": "us-west-2",
   "resources": ["arn:aws:medical-imaging:us-west-2:846006145877:datastore/
bbc4f3cccbae4095a34170fddc19b13d/imageset/207284eef860ac01c4b2a8de27a6fc11"],
   "detail": {
      "imagingVersion": "1.0",
      "isPrimary": true,
      "imageSetVersion": "1",
      "datastoreId": "bbc4f3cccbae4095a34170fddc19b13d",
      "imagesetId": "5b3a711878c34d40e888253319388649",
      "imageSetState": "DELETED",
      "imageSetWorkflowStatus": "DELETED"
   }
}
```


Acara set gambar - 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.

Nama	Tipe	Deskripsi
`region`	string	Mengidentifikasi AWS Wilayah penyimpanan data.
`resources`	array (string)	Sebuah array JSON yang berisi ARN dari set gambar.
`detail`	object	Objek JSON yang berisi informasi tentang peristiwa.
`detail.imagingVersion`	string	ID versi yang melacak perubahan skema detail acara. HealthImaging
`detail.isPrimary`	boolean	Menunjukkan apakah data yang diimpor berhasil diatur ke dalam hierarki terkelola atau jika ada konflik metadata yang perlu diselesaikan.
`detail.imageSetVersion`	string	Versi set gambar akan bertambah ketika sebuah instance diimpor lebih dari sekali. Versi terbaru akan menimpa versi lama yang disimpan dalam kumpulan gambar utama.
`detail.datastoreId`	string	ID penyimpanan data yang menghasilkan peristiwa perubahan status.
`detail.imagesetId`	string	ID set gambar yang terkait dengan peristiwa perubahan status.
`detail.imageSetState`	string	Status set gambar saat ini.

Nama	Tipe	Deskripsi
`detail.imageSetWorkflowStatus`	string	Gambar saat ini mengatur status alur kerja.

Keamanan di AWS HealthImaging

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. Auditor pihak ketiga secara teratur menguji dan memverifikasi efektivitas keamanan kami sebagai bagian dari Program AWS Kepatuhan Program AWS Kepatuhan . Untuk mempelajari tentang program kepatuhan yang berlaku AWS HealthImaging, lihat AWS Layanan dalam Lingkup oleh AWS Layanan Program Kepatuhan .
- 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 HealthImaging. Topik berikut menunjukkan cara mengonfigurasi HealthImaging untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga belajar cara menggunakan AWS layanan lain yang membantu Anda memantau dan mengamankan HealthImaging sumber daya Anda.

Topik

Perlindungan data di AWS HealthImaging

Model tanggung jawab AWS bersama model tanggung berlaku untuk perlindungan data di AWS HealthImaging. 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. Lihat informasi yang lebih lengkap tentang privasi data dalam Pertanyaan Umum Privasi Data. Lihat informasi tentang perlindungan data di Eropa di pos blog Model Tanggung Jawab Bersama dan GDPR AWS di Blog Keamanan AWS .

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 HealthImaging atau lainnya Layanan AWS menggunakan konsol, API AWS CLI, atau AWS SDKs. 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.

Topik

Enkripsi data

Dengan AWS HealthImaging, Anda dapat menambahkan lapisan keamanan ke data Anda saat diam di cloud, menyediakan fitur enkripsi yang dapat diskalakan dan efisien. Ini termasuk:

- Kemampuan enkripsi data saat istirahat tersedia di sebagian besar AWS layanan
- Opsi manajemen kunci yang fleksibel, termasuk AWS Key Management Service, yang dengannya Anda dapat memilih apakah akan AWS mengelola kunci enkripsi atau untuk menjaga kendali penuh atas kunci Anda sendiri.
- AWS kunci AWS KMS enkripsi yang dimiliki
- Antrian pesan terenkripsi untuk transmisi data sensitif menggunakan enkripsi sisi server (SSE) untuk Amazon SQS

Selain itu, AWS menyediakan APIs bagi Anda untuk mengintegrasikan enkripsi dan perlindungan data dengan salah satu layanan yang Anda kembangkan atau terapkan di AWS lingkungan.

Enkripsi saat diam

Secara default, HealthImaging mengenkripsi data pelanggan saat istirahat menggunakan kunci milik layanan AWS Key Management Service . Secara opsional, Anda dapat mengonfigurasi HealthImaging untuk mengenkripsi data saat istirahat menggunakan AWS KMS kunci yang dikelola pelanggan simetris yang Anda buat, miliki, dan kelola. Untuk informasi selengkapnya, lihat Membuat kunci KMS enkripsi simetris di Panduan AWS Key Management Service Pengembang.

Enkripsi saat bergerak

HealthImaging menggunakan TLS 1.2 untuk mengenkripsi data dalam perjalanan melalui titik akhir publik dan melalui layanan backend.

Manajemen kunci

AWS KMS kunci (kunci KMS) adalah sumber daya utama di AWS Key Management Service. Anda juga dapat menghasilkan kunci data untuk digunakan di luar AWS KMS.

AWS kunci KMS yang dimiliki

HealthImaging 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. AWS Kunci KMS yang dimiliki tidak disimpan di akun Anda. Mereka adalah bagian dari kumpulan kunci KMS yang AWS memiliki dan mengelola untuk digunakan di beberapa AWS akun. AWS Layanan dapat menggunakan kunci KMS yang AWS dimiliki untuk melindungi data Anda. Anda tidak dapat melihat, mengelola, menggunakan kunci KMS yang AWS dimiliki, 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 yang AWS dimiliki, dan mereka tidak dihitung terhadap AWS KMS kuota untuk akun Anda. Untuk informasi selengkapnya, lihat kunci yang dimiliki AWS di Panduan AWS Key Management Service Pengembang.

Kunci KMS yang dikelola pelanggan

Jika Anda ingin kontrol penuh atas AWS KMS siklus hidup dan penggunaan, HealthImaging mendukung penggunaan kunci KMS yang dikelola pelanggan simetris yang Anda buat, miliki, dan kelola. 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 HealthImaging dikirim AWS KMS atas nama Anda. AWS KMS Biaya tambahan berlaku. Untuk informasi selengkapnya, lihat Kunci terkelola pelanggan di Panduan AWS Key Management Service Pengembang.

Membuat kunci yang dikelola pelanggan

Anda dapat membuat kunci yang dikelola pelanggan simetris dengan menggunakan Konsol Manajemen AWS atau. AWS KMS APIs Untuk informasi selengkapnya, lihat Membuat kunci KMS enkripsi simetris di Panduan AWS Key Management Service Pengembang.

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 yang dikelola pelanggan di Panduan AWS Key Management Service Pengembang.

Untuk menggunakan kunci yang dikelola pelanggan dengan HealthImaging 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 operasi Hibah yang HealthImaging diperlukan. Untuk informasi selengkapnya, lihat Hibah AWS KMS di Panduan AWS Key Management Service Pengembang.

Untuk menggunakan kunci KMS yang dikelola pelanggan dengan HealthImaging sumber daya Anda, operasi API berikut harus diizinkan dalam kebijakan kunci:

- `kms:DescribeKey`memberikan rincian 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 ke operasi membaca atau mencari sumber daya terenkripsi.
- `kms:ReEncrypt*`menyediakan akses untuk mengenkripsi ulang sumber daya.

Berikut ini adalah contoh pernyataan kebijakan yang memungkinkan pengguna untuk membuat dan berinteraksi dengan penyimpanan data HealthImaging yang dienkripsi oleh kunci tersebut:

```
{
    "Sid": "Allow access to create data stores and perform CRUD and search in
 HealthImaging",
    "Effect": "Allow",
    "Principal": {
        "Service": [
```

```
            "medical-imaging.amazonaws.com"
        ]
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey",
        "kms:GenerateDataKeyWithoutPlaintext"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "kms:EncryptionContext:kms-arn": "arn:aws:kms:us-east-1:123456789012:key/
bec71d48-3462-4cdd-9514-77a7226e001f",
            "kms:EncryptionContext:aws:medical-imaging:datastoreId": "datastoreId"
        }
    }
}
```

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. HealthImaging

Untuk informasi selengkapnya tentang kebijakan utama, lihat Mengaktifkan kebijakan IAM di Panduan AWS Key Management Service Pengembang.

Pengguna IAM, peran IAM, atau AWS akun yang membuat repositori Anda harus memiliki izin untuk kebijakan di bawah ini, ditambah izin yang diperlukan untuk AWS. HealthImaging

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:CreateGrant",
        "kms:GenerateDataKey",
        "kms:RetireGrant",
        "kms:Decrypt",
```

```
        "kms:ReEncrypt*"
      ],
      "Resource": "arn:aws:kms:us-east-1:123456789012:key/
bec71d48-3462-4cdd-9514-77a7226e001f"
    }
  ]
}
```

Bagaimana HealthImaging menggunakan hibah di AWS KMS

HealthImaging membutuhkan hibah untuk menggunakan kunci KMS yang dikelola pelanggan Anda. Saat Anda membuat penyimpanan data yang dienkripsi dengan kunci KMS yang dikelola pelanggan, HealthImaging buat hibah atas nama Anda dengan mengirimkan permintaan ke. CreateGrant AWS KMS Hibah AWS KMS digunakan untuk memberikan HealthImaging akses ke kunci KMS di akun pelanggan.

Hibah yang HealthImaging dibuat atas nama Anda tidak boleh dicabut atau pensiun. Jika Anda mencabut atau menghentikan hibah yang memberikan HealthImaging izin untuk menggunakan AWS KMS kunci di akun Anda, HealthImaging tidak dapat mengakses data ini, mengenkripsi sumber pencitraan baru yang didorong ke penyimpanan data, atau mendekripsi ketika ditarik. Ketika Anda mencabut atau pensiun hibah untuk HealthImaging, perubahan terjadi segera. Untuk mencabut hak akses, Anda harus menghapus penyimpanan data daripada mencabut hibah. Ketika penyimpanan data dihapus, HealthImaging pensiun hibah atas nama Anda.

Memantau kunci enkripsi Anda untuk HealthImaging

Anda dapat menggunakan CloudTrail untuk melacak permintaan yang HealthImaging dikirim atas nama Anda saat menggunakan kunci KMS yang dikelola pelanggan. AWS KMS Entri log di CloudTrail log ditampilkan `medical-imaging.amazonaws.com` di `userAgent` bidang untuk membedakan dengan jelas permintaan yang dibuat oleh HealthImaging.

Contoh berikut adalah CloudTrail peristiwa untuk`CreateGrant`,, `GenerateDataKeyDecrypt`, dan `DescribeKey` untuk memantau AWS KMS operasi yang dipanggil oleh HealthImaging untuk mengakses data yang dienkripsi oleh kunci yang dikelola pelanggan Anda.

Berikut ini menunjukkan cara menggunakan untuk memungkinkan `CreateGrant` HealthImaging untuk mengakses kunci KMS yang disediakan pelanggan, memungkinkan HealthImaging untuk menggunakan kunci KMS itu untuk mengenkripsi semua data pelanggan saat istirahat.

Pengguna tidak diharuskan untuk membuat hibah mereka sendiri. HealthImaging membuat hibah atas nama Anda dengan mengirimkan `CreateGrant` permintaan ke AWS KMS. Hibah AWS KMS digunakan untuk memberikan HealthImaging akses ke AWS KMS kunci di akun pelanggan.


```
{
            "KeyId": "arn:aws:kms:us-
east-1:147997158357:key/8e1c34df-5fd2-49fa-8986-4618c9829a8c",
            "GrantId":
 "44e88bc45b769499ce5ec4abd5ecb27eeb3b178a4782452aae65fe885ee5ba20",
            "Name": "MedicalImagingGrantForQIDO_ebff634a-2d16-4046-9238-e3dc4ab54d29",
            "CreationDate": "2025-04-17T20:12:49+00:00",
            "GranteePrincipal": "AWS Internal",
            "RetiringPrincipal": "medical-imaging.us-east-1.amazonaws.com",
            "IssuingAccount": "medical-imaging.us-east-1.amazonaws.com",
            "Operations": [
                "Decrypt",
                "Encrypt",
                "GenerateDataKey",
                "GenerateDataKeyWithoutPlaintext",
                "ReEncryptFrom",
                "ReEncryptTo",
                "CreateGrant",
                "RetireGrant",
                "DescribeKey"
            ]
        },
        {
            "KeyId": "arn:aws:kms:us-
east-1:147997158357:key/8e1c34df-5fd2-49fa-8986-4618c9829a8c",
            "GrantId":
 "9e5fd5ba7812daf75be4a86efb2b1920d6c0c9c0b19781549556bf2ff98953a1",
            "Name": "2025-04-17T20:12:38",
            "CreationDate": "2025-04-17T20:12:38+00:00",
            "GranteePrincipal": "medical-imaging.us-east-1.amazonaws.com",
            "RetiringPrincipal": "medical-imaging.us-east-1.amazonaws.com",
            "IssuingAccount": "AWS Internal",
            "Operations": [
                "Decrypt",
                "Encrypt",
                "GenerateDataKey",
                "GenerateDataKeyWithoutPlaintext",
                "ReEncryptFrom",
                "ReEncryptTo",
```

```
                "CreateGrant",
                "RetireGrant",
                "DescribeKey"
            ]
        },
        {
            "KeyId": "arn:aws:kms:us-
east-1:147997158357:key/8e1c34df-5fd2-49fa-8986-4618c9829a8c",
            "GrantId":
 "ab4a9b919f6ca8eb2bd08ee72475658ee76cfc639f721c9caaa3a148941bcd16",
            "Name": "9d060e5b5d4144a895e9b24901088ca5",
            "CreationDate": "2025-04-17T20:12:39+00:00",
            "GranteePrincipal": "AWS Internal",
            "RetiringPrincipal": "medical-imaging.us-east-1.amazonaws.com",
            "IssuingAccount": "medical-imaging.us-east-1.amazonaws.com",
            "Operations": [
                "Decrypt",
                "Encrypt",
                "GenerateDataKey",
                "GenerateDataKeyWithoutPlaintext",
                "ReEncryptFrom",
                "ReEncryptTo",
                "DescribeKey"
            ],
            "Constraints": {
                "EncryptionContextSubset": {
                    "kms-arn": "arn:aws:kms:us-
east-1:147997158357:key/8e1c34df-5fd2-49fa-8986-4618c9829a8c"
                }
            }
        }
```

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": "medical-imaging.amazonaws.com"
    },
    "eventTime": "2021-06-30T21:17:37Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "medical-imaging.amazonaws.com",
    "userAgent": "medical-imaging.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 HealthImaging memanggil `Decrypt` operasi 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": "medical-imaging.amazonaws.com"
    },
    "eventTime": "2021-06-30T21:21:59Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "medical-imaging.amazonaws.com",
    "userAgent": "medical-imaging.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 HealthImaging 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": "medical-imaging.amazonaws.com"
    },
    "eventTime": "2021-07-01T18:36:36Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DescribeKey",
    "awsRegion": "us-east-1",
```

```
    "sourceIPAddress": "medical-imaging.amazonaws.com",
    "userAgent": "medical-imaging.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 dan terletak di Panduan AWS Key Management Service Pengembang.

- Konsep AWS KMS
- Praktik terbaik keamanan untuk AWS KMS

Privasi lalu lintas jaringan

Lalu lintas dilindungi baik antara HealthImaging dan aplikasi lokal dan antara HealthImaging dan Amazon S3. Lalu lintas antara HealthImaging dan AWS Key Management Service menggunakan HTTPS secara default.

- AWS HealthImaging adalah layanan regional yang tersedia di Wilayah AS Timur (Virginia N.), AS Barat (Oregon), Eropa (Irlandia), dan Asia Pasifik (Sydney).
- Untuk traffic antara HealthImaging dan bucket Amazon S3, Transport Layer Security (TLS) mengenkripsi objek dalam perjalanan antara dan HealthImaging Amazon S3, dan antara dan aplikasi pelanggan yang mengaksesnya, Anda hanya boleh mengizinkan koneksi terenkripsi

melalui HTTPS (TLS) menggunakan kebijakan IAM bucket Amazon S3 di Amazon S3. HealthImaging `aws:SecureTransport condition` Meskipun HealthImaging saat ini menggunakan titik akhir publik untuk mengakses data di bucket Amazon S3, ini tidak berarti bahwa data tersebut melintasi internet publik. Semua lalu lintas antara HealthImaging dan Amazon S3 dirutekan melalui AWS jaringan dan dienkripsi menggunakan TLS.

Identity and Access Management untuk AWS HealthImaging

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. HealthImaging IAM adalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Topik

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 HealthImaging identitas dan akses AWS)
- Administrator layanan - tentukan akses pengguna dan mengirimkan permintaan izin (lihat Bagaimana AWS HealthImaging bekerja dengan IAM)

- Administrator IAM - tulis kebijakan untuk mengelola akses (lihat Contoh kebijakan berbasis identitas untuk AWS HealthImaging)

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), otentikasi 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 HealthImaging bekerja dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses HealthImaging, pelajari fitur IAM yang tersedia untuk digunakan. HealthImaging

Fitur IAM yang dapat Anda gunakan dengan AWS HealthImaging

Fitur IAM	HealthImaging dukungan
Kebijakan berbasis identitas	Ya
Kebijakan berbasis sumber daya	Tidak
Tindakan kebijakan	Ya
Sumber daya kebijakan	Ya
kunci-kunci persyaratan kebijakan (spesifik layanan)	Ya
ACLs	Tidak
ABAC (tanda dalam kebijakan)	Parsial
Kredensial sementara	Ya
Izin principal	Ya
Peran layanan	Ya

Fitur IAM	HealthImaging dukungan
Peran terkait layanan	Tidak

Untuk mendapatkan tampilan tingkat tinggi tentang cara HealthImaging 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 HealthImaging

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 HealthImaging

Untuk melihat contoh kebijakan HealthImaging berbasis identitas, lihat. Contoh kebijakan berbasis identitas untuk AWS HealthImaging

Kebijakan berbasis sumber daya dalam HealthImaging

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 HealthImaging

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 HealthImaging tindakan, lihat Tindakan yang ditentukan oleh AWS HealthImaging di Referensi Otorisasi Layanan.

Tindakan kebijakan HealthImaging menggunakan awalan berikut sebelum tindakan:

```
AWS
```

Untuk menetapkan secara spesifik beberapa tindakan dalam satu pernyataan, pisahkan tindakan tersebut dengan koma.

```
"Action": [
      "AWS:action1",
      "AWS:action2"
         ]
```

Untuk melihat contoh kebijakan HealthImaging berbasis identitas, lihat. Contoh kebijakan berbasis identitas untuk AWS HealthImaging

Sumber daya kebijakan untuk HealthImaging

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 HealthImaging sumber daya dan jenisnya ARNs, lihat Jenis sumber daya yang ditentukan oleh AWS HealthImaging di Referensi Otorisasi Layanan. Untuk mempelajari tindakan dan sumber daya yang dapat Anda gunakan ARN, lihat Tindakan yang ditentukan oleh AWS. HealthImaging

Untuk melihat contoh kebijakan HealthImaging berbasis identitas, lihat. Contoh kebijakan berbasis identitas untuk AWS HealthImaging

Kunci kondisi kebijakan untuk HealthImaging

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 HealthImaging kondisi, lihat Kunci kondisi untuk AWS HealthImaging di Referensi Otorisasi Layanan. Untuk mempelajari tindakan dan sumber daya yang dapat Anda gunakan kunci kondisi, lihat Tindakan yang ditentukan oleh AWS HealthImaging.

Untuk melihat contoh kebijakan HealthImaging berbasis identitas, lihat. Contoh kebijakan berbasis identitas untuk AWS HealthImaging

ACLs di HealthImaging

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.

RBAC dengan HealthImaging

Mendukung RBAC	Ya

Model otorisasi tradisional yang digunakan di IAM disebut kontrol akses berbasis peran (RBAC). RBAC mendefinisikan izin berdasarkan fungsi tugas seseorang, yang dikenal di luar AWS sebagai peran. Untuk informasi selengkapnya, lihat Membandingkan ABAC dengan model RBAC tradisional di Panduan Pengguna IAM.

ABAC dengan HealthImaging

Mendukung ABAC (tag dalam kebijakan): Sebagian

> **Warning**
>
> ABAC tidak diberlakukan melalui tindakan `SearchImageSets` API. Siapa pun yang memiliki akses ke `SearchImageSets` tindakan dapat mengakses semua metadata untuk kumpulan gambar di penyimpanan data.

> **Note**
>
> Kumpulan gambar adalah sumber daya anak dari penyimpanan data. Untuk menggunakan ABAC, kumpulan gambar harus memiliki tag yang sama dengan penyimpanan data. Untuk informasi lebih lanjut, lihatMenandai sumber daya dengan AWS HealthImaging.

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 HealthImaging

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 HealthImaging

Mendukung sesi akses terusan (FAS): Ya

Saat Anda menggunakan pengguna atau peran IAM untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Kebijakan memberikan izin kepada principal. Saat Anda menggunakan beberapa layanan, Anda mungkin melakukan tindakan yang kemudian memicu tindakan lain di layanan yang berbeda. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk melihat apakah suatu tindakan memerlukan tindakan dependen tambahan dalam kebijakan, lihat Kunci tindakan, sumber daya, dan kondisi AWS HealthImaging di Referensi Otorisasi Layanan.

Peran layanan untuk HealthImaging

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.

> ⚠ Warning
>
> Mengubah izin untuk peran layanan dapat merusak HealthImaging fungsionalitas. Edit peran layanan hanya jika HealthImaging memberikan panduan untuk melakukannya.

Peran terkait layanan untuk HealthImaging

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 HealthImaging

Secara default, pengguna dan peran tidak memiliki izin untuk membuat atau mengubah sumber daya HealthImaging. 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 Awesome, termasuk format ARNs untuk setiap jenis sumber daya, lihat Tindakan, Sumber Daya, dan Kunci Kondisi untuk AWS Keren di Referensi Otorisasi Layanan.

Topik

- Praktik terbaik kebijakan
- Menggunakan HealthImaging konsol
- Mengizinkan pengguna melihat izin mereka sendiri

Praktik terbaik kebijakan

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus HealthImaging 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 HealthImaging konsol

Untuk mengakses HealthImaging konsol AWS, Anda harus memiliki set izin minimum. Izin ini harus memungkinkan Anda untuk membuat daftar dan melihat detail tentang HealthImaging 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 memastikan bahwa pengguna dan peran masih dapat menggunakan HealthImaging konsol, lampirkan juga kebijakan HealthImaging *ConsoleAccess* atau *ReadOnly* AWS terkelola ke entitas. Untuk informasi selengkapnya, lihat Menambah izin untuk pengguna dalam Panduan Pengguna IAM.

Mengizinkan pengguna 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 HealthImaging

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, pembaruan 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.

Topik

- AWS kebijakan terkelola: AWSHealth ImagingServiceRolePolicy
- AWS kebijakan terkelola: AWSHealth ImagingFullAccess
- AWS kebijakan terkelola: AWSHealth ImagingReadOnlyAccess
- HealthImaging pembaruan kebijakan AWS terkelola

AWS kebijakan terkelola: AWSHealth ImagingServiceRolePolicy

Kebijakan ini dilampirkan pada peran terkait layanan. `AWSServiceRoleForHealthImaging` Ini memberikan izin untuk HealthImaging mengelola operasi layanan dan mempublikasikan metrik layanan.

Untuk informasi selengkapnya tentang kebijakan ini, termasuk dokumen kebijakan JSON, lihat AWSHealthImagingServiceRolePolicydi Panduan Referensi Kebijakan Terkelola AWS.

AWS kebijakan terkelola: AWSHealth ImagingFullAccess

Anda dapat melampirkan kebijakan `AWSHealthImagingFullAccess` ke identitas IAM Anda.

Kebijakan ini memberikan izin administratif untuk semua HealthImaging tindakan.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "medical-imaging:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "medical-imaging.amazonaws.com"
            }
        }
        }
    ]
}
```

AWS kebijakan terkelola: AWSHealth ImagingReadOnlyAccess

Anda dapat melampirkan kebijakan `AWSHealthImagingReadOnlyAccess` ke identitas IAM Anda.

Kebijakan ini memberikan izin hanya-baca untuk tindakan AWS tertentu. HealthImaging

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "medical-imaging:GetDICOMImportJob",
            "medical-imaging:GetDatastore",
            "medical-imaging:GetImageFrame",
            "medical-imaging:GetImageSet",
            "medical-imaging:GetImageSetMetadata",
            "medical-imaging:ListDICOMImportJobs",
            "medical-imaging:ListDatastores",
```

```
            "medical-imaging:ListImageSetVersions",
            "medical-imaging:ListTagsForResource",
            "medical-imaging:SearchImageSets"
        ],
        "Resource": "*"
    }]
}
```

HealthImaging pembaruan kebijakan AWS terkelola

Lihat detail tentang pembaruan kebijakan AWS terkelola HealthImaging sejak layanan ini mulai melacak perubahan ini. Untuk peringatan otomatis tentang perubahan pada halaman ini, berlangganan umpan RSS di halaman Rilis.

Ubah	Deskripsi	Date
AWSHealthImagingSe rviceRolePolicy	AWS HealthImaging menambahkan kebijakan terkelola baru untuk peran terkait layanan yang menyediakan izin untuk HealthImaging mengelola operasi layanan dan mempublikasikan metrik layanan.	Februari 9, 2026
HealthImaging mulai melacak perubahan	HealthImaging mulai melacak perubahan untuk kebijakan yang AWS dikelola.	Juli 19, 2023

Cross-service membingungkan pencegahan deputi di HealthImaging

Masalah "confused deputy" adalah masalah keamanan di mana entitas yang tidak memiliki izin untuk melakukan tindakan dapat memengaruhi entitas yang memiliki hak akses lebih tinggi untuk

melakukan tindakan. Di AWS, peniruan identitas lintas layanan dapat mengakibatkan masalah deputi yang membingungkan. Peniruan identitas lintas layanan dapat terjadi ketika satu layanan (layanan yang dipanggil) memanggil layanan lain (layanan yang dipanggil). Layanan pemanggilan dapat dimanipulasi menggunakan izinnya untuk bertindak pada sumber daya pelanggan lain dengan cara yang seharusnya tidak dilakukannya kecuali bila memiliki izin untuk mengakses. Untuk mencegah hal ini, AWS menyediakan alat yang membantu Anda melindungi data Anda untuk semua layanan dengan prinsip layanan yang telah diberi akses ke sumber daya di akun Anda.

Sebaiknya gunakan kunci konteks kondisi `aws:SourceAccount`global `aws:SourceArn`dan global dalam kebijakan hubungan kepercayaan peran `ImportJobDataAccessRole` IAM Anda untuk membatasi izin yang HealthImaging diberikan AWS kepada layanan lain ke sumber daya Anda. Gunakan `aws:SourceArn` untuk mengaitkan hanya satu sumber daya dengan akses lintas layanan. Gunakan `aws:SourceAccount` untuk membiarkan sumber daya apa pun di akun itu dikaitkan dengan penggunaan lintas layanan. Jika Anda menggunakan kedua kunci konteks kondisi global, `aws:SourceAccount` nilai dan akun yang direferensikan dalam `aws:SourceArn` nilai harus menggunakan ID akun yang sama saat digunakan dalam pernyataan kebijakan yang sama.

Nilai `aws:SourceArn` harus ARN dari penyimpanan data yang terpengaruh. Jika Anda tidak mengetahui ARN lengkap dari penyimpanan data, atau jika Anda menentukan beberapa penyimpanan data, gunakan kunci kondisi konteks `aws:SourceArn` global dengan wildcard * untuk bagian ARN yang tidak diketahui. Misalnya, Anda dapat mengatur `aws:SourceArn` `kearn:aws:medical-imaging:us-west-2:111122223333:datastore/*`.

Dalam contoh kebijakan kepercayaan berikut, kami menggunakan kunci `aws:SourceArn` dan `aws:SourceAccount` kondisi untuk membatasi akses ke prinsipal layanan berdasarkan ARN penyimpanan data untuk mencegah masalah wakil yang membingungkan.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "medical-imaging.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
```

```
        "ArnLike": {
          "aws:SourceArn": "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/*"
        },
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        }
      }
    }
  ]
}
```

Menggunakan peran terkait layanan untuk HealthImaging

AWS HealthImaging menggunakan peran terkait layanan AWS Identity and Access Management (IAM) yang telah ditentukan sebelumnya oleh layanan dan menyertakan semua izin yang diperlukan layanan untuk memanggil layanan AWS lain atas nama Anda. Untuk informasi selengkapnya, lihat Izin peran terkait layanan di Panduan Pengguna IAM.

Izin peran terkait layanan untuk HealthImaging

HealthImaging menggunakan peran terkait layanan bernama `AWSServiceRoleForHealthImaging` untuk melakukan operasi di akun Anda AWS . Anda perlu membuat peran terkait layanan ini jika HealthImaging ingin mempublikasikan metrik tentang penyimpanan data Anda. CloudWatch

Kebijakan izin peran bernama `AWSHealthImagingServiceRolePolicy` memberikan izin untuk HealthImaging mengelola operasi layanan dan mempublikasikan metrik layanan.

Untuk pembaruan kebijakan terkelola, lihat kebijakan HealthImaging terkelola.

Membuat peran terkait layanan untuk HealthImaging

Membuat peran terkait layanan dengan Konsol IAM

Anda dapat membuat peran terkait layanan menggunakan Konsol IAM dengan Memilih `AWS Service` sebagai jenis entitas Tepercaya, lalu `HealthImaging` di menu tarik-turun Kasus penggunaan.

Buat peran terkait layanan dengan AWS CLI

Di AWS CLI, jalankan `aws iam create-service-linked-role –aws-service-name medical-imaging.amazonaws.com`

Menghapus peran terkait layanan untuk HealthImaging

Anda dapat menghapus peran yang ditautkan layanan kapan saja, tetapi melakukannya akan memblokir HealthImaging dari melakukan tindakan di AWS akun Anda, seperti mempublikasikan metrik penyimpanan data ke. CloudWatch

Untuk menghapus peran tertaut layanan secara manual menggunakan IAM

Anda dapat menggunakan konsol IAM, AWS CLI, atau AWS API untuk menghapus peran terkait layanan`AWSServiceRoleForHealthImaging`. Untuk informasi selengkapnya, lihat Menghapus peran terkait layanan dalam Panduan Pengguna IAM. Jika Anda menghapus peran terkait layanan, Anda dapat menggunakan proses pembuatan peran untuk membuat peran baru.

Memecahkan masalah HealthImaging identitas dan akses AWS

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan HealthImaging dan IAM.

Topik

- Saya tidak berwenang untuk melakukan tindakan di HealthImaging
- Saya tidak berwenang untuk melakukan iam: PassRole
- Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses HealthImaging sumber daya saya

Saya tidak berwenang untuk melakukan tindakan di HealthImaging

Jika Anda menerima pesan kesalahan bahwa Anda tidak memiliki otorisasi untuk melakukan tindakan, kebijakan Anda harus diperbarui agar Anda dapat melakukan tindakan tersebut.

Contoh kesalahan berikut terjadi ketika pengguna IAM `mateojackson` mencoba menggunakan konsol untuk melihat detail tentang suatu sumber daya *`my-example-widget`* rekaan, tetapi tidak memiliki izin `AWS:`*`GetWidget`* rekaan.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 AWS:GetWidget on resource: my-example-widget
```

Dalam hal ini, kebijakan untuk pengguna `mateojackson` harus diperbarui untuk mengizinkan akses ke sumber daya *`my-example-widget`* dengan menggunakan tindakan `AWS:GetWidget`.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

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 HealthImaging.

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 HealthImaging. 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 saya Akun AWS untuk mengakses HealthImaging 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 HealthImaging mendukung fitur-fitur ini, lihatBagaimana AWS HealthImaging 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 HealthImaging

Auditor pihak ketiga menilai keamanan dan kepatuhan AWS HealthImaging sebagai bagian dari beberapa program AWS kepatuhan. Sebab HealthImaging, ini termasuk HIPAA.

Untuk daftar AWS layanan dalam lingkup program kepatuhan tertentu, lihat AWS Services in Scope by Compliance Program . 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 AWS HealthImaging ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, serta undang-undang dan peraturan yang berlaku. AWS menyediakan sumber daya berikut untuk membantu kepatuhan:

- AWS Solusi Mitra — Panduan penerapan referensi otomatis untuk Keamanan dan Kepatuhan membahas pertimbangan arsitektur dan memberikan langkah-langkah untuk menerapkan keamanan dan lingkungan dasar yang berfokus pada kepatuhan. AWS
- Arsitektur untuk Whitepaper Keamanan dan Kepatuhan HIPAA — Whitepaper ini menjelaskan bagaimana perusahaan dapat menggunakan untuk membuat aplikasi yang sesuai dengan HIPAA. AWS
- Sistem GxP aktif AWS— Whitepaper ini memberikan informasi tentang cara AWS mendekati kepatuhan dan keamanan terkait GXP dan memberikan panduan tentang penggunaan AWS layanan dalam konteks GxP.

- AWS Sumber Daya Kepatuhan — Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.
- Mengevaluasi Sumber Daya dengan Aturan — AWS Config menilai seberapa baik konfigurasi sumber daya Anda mematuhi praktik internal, pedoman industri, dan peraturan.
- AWS Security Hub CSPM— AWS Layanan ini memberikan pandangan komprehensif tentang keadaan keamanan Anda di dalamnya AWS yang membantu Anda memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik.

Keamanan infrastruktur di AWS HealthImaging

Sebagai layanan terkelola, AWS HealthImaging 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 HealthImaging melalui jaringan. Klien harus mendukung Transport Layer Security (TLS) 1.3 atau yang lebih baru. Klien juga harus mendukung cipher suite dengan perfect forward secrecy (PFS) seperti Ephemeral Diffie-Hellman (DHE) atau Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Sebagian besar sistem-sistem modern seperti Java 7 dan versi yang lebih baru mendukung mode-mode ini.

Selain itu, permintaan harus ditandatangani menggunakan ID kunci akses dan kunci akses rahasia yang dikaitkan dengan pengguna utama IAM. Anda juga dapat menggunakan AWS Security Token Service(AWS STS) untuk menghasilkan kredensyal keamanan sementara untuk menandatangani permintaan.

Membuat HealthImaging sumber daya AWS dengan AWS CloudFormation

AWS HealthImaging terintegrasi dengan AWS CloudFormation, layanan yang membantu Anda memodelkan dan menyiapkan AWS sumber daya 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 HealthImaging 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.

HealthImaging dan CloudFormation template

Untuk menyediakan dan mengonfigurasi sumber daya untuk HealthImaging 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 .

AWS HealthImaging mendukung pembuatan penyimpanan data dengan CloudFormation. Untuk informasi selengkapnya, termasuk contoh templat JSON dan YAMAL untuk menyediakan penyimpanan HealthImaging data, lihat referensi jenis HealthImaging sumber daya AWS 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 HealthImaging dan antarmuka titik akhir VPC ()AWS PrivateLink

Anda dapat membuat koneksi pribadi antara VPC Anda dan AWS HealthImaging dengan membuat antarmuka VPC endpoint. Endpoint antarmuka didukung oleh AWS PrivateLink, teknologi yang dapat Anda gunakan untuk mengakses secara pribadi HealthImaging APIs tanpa gateway internet, perangkat NAT, koneksi VPN, atau koneksi AWS Direct Connect. Instans di VPC Anda tidak memerlukan alamat IP publik untuk berkomunikasi. HealthImaging APIs Lalu lintas antara VPC Anda dan HealthImaging 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.

Topik

- Pertimbangan untuk titik akhir HealthImaging VPC
- Membuat titik akhir VPC antarmuka untuk HealthImaging
- Membuat kebijakan titik akhir VPC untuk HealthImaging

Pertimbangan untuk titik akhir HealthImaging VPC

Sebelum menyiapkan titik akhir VPC antarmuka HealthImaging, pastikan Anda meninjau properti dan batasan titik akhir Antarmuka di Panduan Pengguna Amazon VPC.

HealthImaging mendukung panggilan ke semua AWS HealthImaging tindakan dari VPC Anda.

Membuat titik akhir VPC antarmuka untuk HealthImaging

Anda dapat membuat titik akhir VPC untuk HealthImaging layanan 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 HealthImaging menggunakan nama layanan berikut:

- com.amazonaws. *region*.pencitraan medis
- com.amazonaws. *region*. runtime-medical-imaging
- com.amazonaws. *region*. dicom-medical-imaging

> **Note**
>
> DNS pribadi harus diaktifkan untuk digunakan PrivateLink.

Anda dapat membuat permintaan API untuk HealthImaging menggunakan nama DNS default untuk Wilayah, misalnya,`medical-imaging.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 HealthImaging

Anda dapat melampirkan kebijakan titik akhir ke VPC endpoint yang mengendalikan akses ke HealthImaging. Kebijakan titik akhir menentukan informasi berikut:

- Prinsip-prinsip yang dapat melakukan tindakan.
- Tindakan yang dapat dilakukan
- Sumber daya yang dapat digunakan untuk mengambil tindakan

Untuk informasi selengkapnya, lihat Mengendalikan akses ke layanan dengan VPC endpoint di Panduan Pengguna Amazon VPC.

Contoh: Kebijakan titik akhir VPC untuk tindakan HealthImaging

Berikut ini adalah contoh kebijakan endpoint untuk HealthImaging. Saat dilampirkan ke titik akhir, kebijakan ini memberikan akses ke HealthImaging tindakan untuk semua prinsipal di semua sumber daya.

API

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "medical-imaging:*"
         ],
         "Resource":"*"
      }
   ]
}
```

CLI

```
aws ec2 modify-vpc-endpoint \
    --vpc-endpoint-id vpce-id
    --region us-west-2 \
    --private-dns-enabled \
    --policy-document \
```

```
    "{\"Statement\":[{\"Principal\":\"*\",\"Effect\":\"Allow\",\"Action\":
[\"medical-imaging:*\"],\"Resource\":\"*\"}]}"
```

Impor lintas akun untuk AWS HealthImaging

Dengan impor lintas akun/lintas wilayah, Anda dapat mengimpor data ke penyimpanan data dari bucket HealthImaging Amazon S3 yang terletak di Wilayah lain yang didukung. Anda dapat mengimpor data di seluruh AWS akun, akun yang dimiliki oleh AWS Organizations lain, dan dari sumber data terbuka seperti Imaging Data Commons (IDC) yang terletak di Registry of Open Data on. AWS

HealthImaging Kasus penggunaan impor lintas akun/lintas wilayah meliputi:

- Pencitraan medis Produk SaaS mengimpor data DICOM dari akun pelanggan
- Organisasi besar yang mengisi satu penyimpanan HealthImaging data dari banyak bucket input Amazon S3
- Para peneliti dengan aman berbagi data di seluruh studi klinis multi-institusi

Untuk menggunakan impor lintas akun

1. Pemilik bucket masukan (sumber) Amazon S3 harus memberikan pemilik penyimpanan HealthImaging data `s3:ListBucket` dan `s3:GetObject` izin.
2. Pemilik penyimpanan HealthImaging data harus menambahkan bucket Amazon S3 ke IAM mereka. `ImportJobDataAccessRole` Lihat Buat peran IAM untuk impor.
3. Pemilik penyimpanan HealthImaging data harus menyediakan bucket masukan Amazon S3 `inputOwnerAccountId`untuk memulai pekerjaan impor.

> **Note**
>
> Dengan menyediakan`inputOwnerAccountId`, pemilik penyimpanan data memvalidasi masukan bucket Amazon S3 milik akun yang ditentukan untuk menjaga kepatuhan terhadap standar industri dan mengurangi potensi risiko keamanan.

Contoh `startDICOMImportJob` kode berikut mencakup `inputOwnerAccountId` parameter opsional, yang dapat diterapkan ke semua AWS CLI dan contoh kode SDK di Memulai pekerjaan impor bagian.

Java

```
public static String startDicomImportJob(MedicalImagingClient
 medicalImagingClient,
        String jobName,
        String datastoreId,
        String dataAccessRoleArn,
        String inputS3Uri,
        String outputS3Uri,
        String inputOwnerAccountId) {

    try {
        StartDicomImportJobRequest startDicomImportJobRequest =
 StartDicomImportJobRequest.builder()
                .jobName(jobName)
                .datastoreId(datastoreId)
                .dataAccessRoleArn(dataAccessRoleArn)
                .inputS3Uri(inputS3Uri)
                .outputS3Uri(outputS3Uri)
                .inputOwnerAccountId(inputOwnerAccountId)
                .build();
        StartDicomImportJobResponse response =
 medicalImagingClient.startDICOMImportJob(startDicomImportJobRequest);
        return response.jobId();
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
```

Ketahanan di AWS HealthImaging

Infrastruktur AWS global dibangun di sekitar Wilayah AWS dan Availability Zones. Wilayah AWS menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan latensi rendah, throughput tinggi, dan jaringan yang sangat redundan. Dengan Zona Ketersediaan, Anda dapat merancang serta mengoperasikan aplikasi dan basis data yang secara otomatis melakukan fail over 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 Wilayah AWS, Wilayah AWS Lokal benar-benar terisolasi dari yang lain Wilayah AWS.

Untuk informasi selengkapnya tentang Wilayah AWS dan Availability Zone, lihat Infrastruktur AWS Global.

Bahan HealthImaging referensi AWS

> **Note**
>
> Semua tindakan HealthImaging API asli dan tipe data dijelaskan dalam AWS HealthImaging API Referensi.

Topik

Dukungan DICOM untuk AWS HealthImaging

AWS HealthImaging mendukung elemen DICOM tertentu dan sintaks transfer. Biasakan diri Anda dengan elemen data DICOM tingkat Pasien, Studi, dan Seri yang didukung, karena kunci HealthImaging metadata didasarkan padanya. Sebelum memulai impor, verifikasi bahwa data pencitraan medis Anda sesuai dengan HealthImaging sintaks transfer yang didukung dan batasan elemen DICOM.

> **Note**
>
> AWS saat ini HealthImaging tidak mendukung data piksel Gambar Segmentasi Biner atau Urutan Gambar Ikon.

Topik

Kelas SOP yang didukung

Dengan AWS HealthImaging, Anda dapat mengimpor instans DICOM P10 Service-Object Pair (SOP) yang dikodekan dengan UID kelas SOP apa pun, termasuk pensiunan dan pribadi. Semua atribut pribadi dipertahankan, juga.

Normalisasi metadata

Saat Anda mengimpor data DICOM P10 Anda ke AWS HealthImaging, data tersebut diubah menjadi kumpulan gambar yang terdiri dari metadata dan bingkai gambar (data piksel). Selama proses transformasi, kunci HealthImaging metadata dihasilkan berdasarkan versi tertentu dari standar DICOM. HealthImaging saat ini menghasilkan dan mendukung kunci metadata berdasarkan Kamus Data DICOM PS3 .6 2022b.

AWS HealthImaging mendukung elemen data DICOM berikut di tingkat Pasien, Studi, dan Seri.

Elemen tingkat pasien

> **Note**
>
> Untuk penjelasan rinci dari setiap elemen tingkat Pasien, lihat Registry of DICOM Data Elements.

AWS HealthImaging mendukung elemen tingkat Pasien berikut:

```
Patient Module Elements

  (0010,0010) - Patient's Name
  (0010,0020) - Patient ID

Issuer of Patient ID Macro Elements

  (0010,0021) - Issuer of Patient ID
  (0010,0024) - Issuer of Patient ID Qualifiers Sequence
  (0010,0022) - Type of Patient ID
  (0010,0030) - Patient's Birth Date
  (0010,0033) - Patient's Birth Date in Alternative Calendar
  (0010,0034) - Patient's Death Date in Alternative Calendar
  (0010,0035) - Patient's Alternative Calendar Attribute
  (0010,0040) - Patient's Sex
```

```
  (0010,1100) - Referenced Patient Photo Sequence
  (0010,0200) - Quality Control Subject
  (0008,1120) - Referenced Patient Sequence
  (0010,0032) - Patient's Birth Time
  (0010,1002) - Other Patient IDs Sequence
  (0010,1001) - Other Patient Names
  (0010,2160) - Ethnic Group
  (0010,4000) - Patient Comments
  (0010,2201) - Patient Species Description
  (0010,2202) - Patient Species Code Sequence Attribute
  (0010,2292) - Patient Breed Description
  (0010,2293) - Patient Breed Code Sequence
  (0010,2294) - Breed Registration Sequence Attribute
  (0010,0212) - Strain Description
  (0010,0213) - Strain Nomenclature Attribute
  (0010,0219) - Strain Code Sequence
  (0010,0218) - Strain Additional Information Attribute
  (0010,0216) - Strain Stock Sequence
  (0010,0221) - Genetic Modifications Sequence Attribute
  (0010,2297) - Responsible Person
  (0010,2298) - Responsible Person Role Attribute
  (0010,2299) - Responsible Organization
  (0012,0062) - Patient Identity Removed
  (0012,0063) - De-identification Method
  (0012,0064) - De-identification Method Code Sequence

Patient Group Macro Elements

  (0010,0026) - Source Patient Group Identification Sequence
  (0010,0027) - Group of Patients Identification Sequence

Clinical Trial Subject Module

  (0012,0010) - Clinical Trial Sponsor Name
  (0012,0020) - Clinical Trial Protocol ID
  (0012,0021) - Clinical Trial Protocol Name Attribute
  (0012,0030) - Clinical Trial Site ID
  (0012,0031) - Clinical Trial Site Name
  (0012,0040) - Clinical Trial Subject ID
  (0012,0042) - Clinical Trial Subject Reading ID
  (0012,0081) - Clinical Trial Protocol Ethics Committee Name
  (0012,0082) - Clinical Trial Protocol Ethics Committee Approval Number
```

Elemen tingkat studi

> **Note**
>
> Untuk penjelasan rinci dari setiap elemen tingkat Studi, lihat Registri Elemen Data DICOM.

AWS HealthImaging mendukung elemen tingkat Studi berikut:

```
General Study Module

  (0020,000D) - Study Instance UID
  (0008,0020) - Study Date
  (0008,0030) - Study Time
  (0008,0090) - Referring Physician's Name
  (0008,0096) - Referring Physician Identification Sequence
  (0008,009C) - Consulting Physician's Name
  (0008,009D) - Consulting Physician Identification Sequence
  (0020,0010) - Study ID
  (0008,0050) - Accession Number
  (0008,0051) - Issuer of Accession Number Sequence
  (0008,1030) - Study Description
  (0008,1048) - Physician(s) of Record
  (0008,1049) - Physician(s) of Record Identification Sequence
  (0008,1060) - Name of Physician(s) Reading Study
  (0008,1062) - Physician(s) Reading Study Identification Sequence
  (0032,1033) - Requesting Service
  (0032,1034) - Requesting Service Code Sequence
  (0008,1110) - Referenced Study Sequence
  (0008,1032) - Procedure Code Sequence
  (0040,1012) - Reason For Performed Procedure Code Sequence

Patient Study Module

  (0008,1080) - Admitting Diagnoses Description
  (0008,1084) - Admitting Diagnoses Code Sequence
  (0010,1010) - Patient's Age
  (0010,1020) - Patient's Size
  (0010,1030) - Patient's Weight
  (0010,1022) - Patient's Body Mass Index
  (0010,1023) - Measured AP Dimension
  (0010,1024) - Measured Lateral Dimension
  (0010,1021) - Patient's Size Code Sequence
```

```
  (0010,2000) - Medical Alerts
  (0010,2110) - Allergies
  (0010,21A0) - Smoking Status
  (0010,21C0) - Pregnancy Status
  (0010,21D0) - Last Menstrual Date
  (0038,0500) - Patient State
  (0010,2180) - Occupation
  (0010,21B0) - Additional Patient History
  (0038,0010) - Admission ID
  (0038,0014) - Issuer of Admission ID Sequence
  (0032,1066) - Reason for Visit
  (0032,1067) - Reason for Visit Code Sequence
  (0038,0060) - Service Episode ID
  (0038,0064) - Issuer of Service Episode ID Sequence
  (0038,0062) - Service Episode Description
  (0010,2203) - Patient's Sex Neutered

Clinical Trial Study Module

  (0012,0050) - Clinical Trial Time Point ID
  (0012,0051) - Clinical Trial Time Point Description
  (0012,0052) - Longitudinal Temporal Offset from Event
  (0012,0053) - Longitudinal Temporal Event Type
  (0012,0083) - Consent for Clinical Trial Use Sequence
```

Elemen tingkat seri

> **Note**
>
> Untuk penjelasan rinci dari setiap elemen tingkat Seri, lihat Registri Elemen Data DICOM.

AWS HealthImaging mendukung elemen level Seri berikut:

```
General Series Module

  (0008,0060) - Modality
  (0020,000E) - Series Instance UID
  (0020,0011) - Series Number
  (0020,0060) - Laterality
  (0008,0021) - Series Date
  (0008,0031) - Series Time
  (0008,1050) - Performing Physician's Name
```

```
  (0008,1052) - Performing Physician Identification Sequence
  (0018,1030) - Protocol Name
  (0008,103E) - Series Description
  (0008,103F) - Series Description Code Sequence
  (0008,1070) - Operators' Name
  (0008,1072) - Operator Identification Sequence
  (0008,1111) - Referenced Performed Procedure Step Sequence
  (0008,1250) - Related Series Sequence
  (0018,0015) - Body Part Examined
  (0018,5100) - Patient Position
  (0028,0108) - Smallest Pixel Value in Series
  (0028,0109) - Largest Pixel Value in Series
  (0040,0275) - Request Attributes Sequence
  (0010,2210) - Anatomical Orientation Type
  (300A,0700) - Treatment Session UID

Clinical Trial Series Module

  (0012,0060) - Clinical Trial Coordinating Center Name
  (0012,0071) - Clinical Trial Series ID
  (0012,0072) - Clinical Trial Series Description

General Equipment Module

  (0008,0070) - Manufacturer
  (0008,0080) - Institution Name
  (0008,0081) - Institution Address
  (0008,1010) - Station Name
  (0008,1040) - Institutional Department Name
  (0008,1041) - Institutional Department Type Code Sequence
  (0008,1090) - Manufacturer's Model Name
  (0018,100B) - Manufacturer's Device Class UID
  (0018,1000) - Device Serial Number
  (0018,1020) - Software Versions
  (0018,1008) - Gantry ID
  (0018,100A) - UDI Sequence
  (0018,1002) - Device UID
  (0018,1050) - Spatial Resolution
  (0018,1200) - Date of Last Calibration
  (0018,1201) - Time of Last Calibration
  (0028,0120) - Pixel Padding Value

Frame of Reference Module
```

```
  (0020,0052) - Frame of Reference UID
  (0020,1040) - Position Reference Indicator
```

Sintaks transfer yang didukung

AWS HealthImaging mendukung pengimporan file DICOM P10 dengan sintaks transfer yang berbeda. Beberapa file mempertahankan pengkodean sintaks transfer aslinya selama impor, sementara sebagian besar bingkai gambar lossless ditranskode selama impor. Perilaku transcoding tergantung pada konfigurasi datastore Anda. Penyimpanan data menggunakan HTJ2 K sebagai format penyimpanan secara default, tetapi dapat dikonfigurasi pada waktu pembuatan untuk menggunakan JPEG 2000 Lossless. Contoh berikut menunjukkan bagaimana HealthImaging catatan `StoredTransferSyntaxUID` untuk setiap instance dalam metadata dikembalikan oleh. `GetImageSetMetadata`

```
"Instances": {
   "999.999.2.19941105.134500.2.101": {
       "StoredTransferSyntaxUID": "1.2.840.10008.1.2.4.90",
       "ImageFrames": [{ ...
```

> **Note**
>
> Ingatlah poin-poin ini saat melihat tabel berikut:
>
> - Entri UID sintaks transfer yang ditandai dengan tanda bintang (*) menunjukkan file disimpan dalam format asli yang dikodekan selama impor. Untuk file-file ini, metadata yang `StoredTransferSyntaxUID` terletak di instance cocok dengan sintaks transfer asli.
> - Entri UID sintaks transfer yang ditandai tanpa tanda bintang menunjukkan file ditranskode ke HTJ2 K lossless selama impor dan disimpan di. HealthImaging `StoredTransferSyntaxUID`Elemen metadata instance akan diatur format penyimpanan High-Throughput JPEG 2000 dengan RPCL Options Image Compression—Lossless Only (1.2.840.10008.1.2.4.202).
> - Jika `StoredTransferSyntaxUID` kunci tidak ada atau diatur ke`null`, Anda dapat mengasumsikan itu dikodekan ke format penyimpanan penyimpanan data yang dikonfigurasi.

HealthImaging sintaks transfer yang didukung

Transfer sintaks UID	Transfer nama sintaks
1.2.840.10008.1.2	Implicit VR Endian: Sintaks Transfer Default untuk DICOM
1.2.840.10008.1.2.1* (segmentasi biner mempertahankan pengkodean asli, sedangkan , segmentasi non-biner ditranskode ke K Lossless RPCL) HTJ2	Eksplisit VR Little Endian
1.2.840.10008.1.2.1.99	Kempes Eksplisit VR Little Endian
1.2.840.10008.1.2.2	Eksplisit VR Big Endian
1.2.840.10008.1.2.4.50*	JPEG Baseline (Proses 1): Sintaks Transfer Default untuk Kompresi Gambar 8-bit Lossy JPEG
1.2.840.10008.1.2.4.51	JPEG Baseline (Proses 2 & 4): Sintaks Transfer Default untuk Kompresi Gambar 12-bit Lossy JPEG (Hanya Proses 4)
1.2.840.10008.1.2.4.57	JPEG Lossless Non-Hierarkis (Proses 14)
1.2.840.10008.1.2.4.70	JPEG Lossless, Nonhierarkis, Prediksi Urutan Pertama (Proses 14 [Nilai Seleksi 1]): Sintaks Transfer Default untuk Kompresi Gambar JPEG Lossless
1.2.840.10008.1.2.4.80	Kompresi Gambar Lossless JPEG-LS
1.2.840.10008.1.2.4.81	Kompresi Gambar JPEG-LS Lossy (Near-Los sless)
1.2.840.10008.1.2.4.90	Kompresi Gambar JPEG 2000 (Hanya Lossless)
1.2.840.10008.1.2.4.91*	Kompresi Gambar JPEG 2000

Transfer sintaks UID	Transfer nama sintaks
1.2.840.10008.1.2.4.92	JPEG 2000 Bagian 2 Kompresi Gambar Multikomponen (Hanya Lossless)
1.2.840.10008.1.2.4.93	JPEG 2000 Bagian 2 Kompresi Gambar Multikomponen
1.2.840.10008.1.2.4.112*	JPEG XL
1.2.840.10008.1.2.4.201	Kompresi Gambar JPEG 2000 High-Thro ughput (Hanya Lossless)
1.2.840.10008.1.2.4.202	High-Throughput JPEG 2000 dengan Kompresi Gambar Opsi RPCL (Hanya Lossless)
1.2.840.10008.1.2.4.203*	Kompresi Gambar JPEG 2000 Throughput Tinggi
1.2.840.10008.1.2.5	RLE Tanpa Kehilangan
1.2.840.10008.1.2.4.100*, 1.2.840.10008.1.2. 4.100.1*	MPEG2 Profil Utama Tingkat Utama
1.2.840.10008.1.2.4.101*, 1.2.840.10008.1.2. 4.101.1*	MPEG2 Profil Utama di Tingkat Tinggi
1.2.840.10008.1.2.4.102*, 1.2.840.10008.1.2. 4.102.1*	MPEG-4 AVC/H.264 Profil Tinggi/Level 4.1
1.2.840.10008.1.2.4.103*, 1.2.840.10008.1.2. 4.103.1*	Profil Tinggi Kompatibel dengan MPEG-4 AVC/ H.264 BD/Level 4.1
1.2.840.10008.1.2.4.104*, 1.2.840.10008.1.2. 4.104.1*	MPEG-4 AVC/H.264 Profil Tinggi/Level 4.2 untuk Video 2D
1.2.840.10008.1.2.4.105*, 1.2.840.10008.1.2. 4.105.1*	MPEG-4 AVC/H.264 Profil Tinggi/Level 4.2 dari ITU-T H.264 Video

Transfer sintaks UID	Transfer nama sintaks
1.2.840.10008.1.2.4.106*, 1.2.840.10008.1.2.4.106.1*	MPEG-4 AVC/H.264 Stereo Profil Tinggi/Level 4.2 dari ITU-T H.264 Video
1.2.840.10008.1.2.4.107*	HEVC/H.265 Profil Utama/Level 5.1 Video
1.2.840.10008.1.2.4.108*	HEVC/H.265 Utama 10 Profil/Level 5.1

* Mempertahankan pengkodean sintaks transfer asli selama impor

Kendala elemen DICOM

Saat mengimpor data pencitraan medis Anda ke AWS HealthImaging, batasan panjang maksimal diterapkan ke elemen DICOM berikut. Untuk mencapai impor yang berhasil, pastikan bahwa data Anda tidak melebihi batasan panjang maksimal.

Kendala elemen DICOM selama impor

DICOM kata kunci	Tag DICOM	Panjang maks
PatientName	(0010.0010)	256
PatientID	(0010.0020)	256
PatientBirthDate	(0010.0030)	18
PatientSex	(0010.0040)	16
StudyInstanceUID	(0020,000D)	256
StudyID	(0020,0010)	256
StudyDescription	(0008,1030)	256
NumberOfStudyRelatedSeries	(0020,1206)	1.000.000
NumberOfStudyRelatedInstances	(0020,1208)	1.000.000
AccessionNumber	(0008,0050)	256

DICOM kata kunci	Tag DICOM	Panjang maks
StudyDate	(0008,0020)	18
StudyTime	(0008,0030)	28
SOPInstanceUID	(0008,0018)	256
SeriesInstanceUID	(0020,000E)	256

Kendala metadata DICOM

Saat Anda menggunakan `UpdateImageSetMetadata` untuk memperbarui atribut HealthImaging metadata, batasan DICOM berikut diterapkan.

- Tidak dapat memperbarui atau menghapus atribut pribadi pada atribut Patient/Study/Series/Instance level kecuali kendala pembaruan berlaku untuk keduanya dan `updatableAttributes` `removableAttributes`
- Tidak dapat memperbarui atribut HealthImaging yang dihasilkan AWS berikut: `SchemaVersionDatastoreID`,`ImageSetID`,`PixelData`,`Checksum`,`Width`,`Height`,`MinPixelValue`, `FrameSizeInBytes`
- Tidak dapat memperbarui atribut DICOM berikut kecuali `force` bendera disetel:`Tag.PixelData`,,`Tag.StudyInstanceUID`,`Tag.SeriesInstanceUID`, `Tag.SOPInstanceUID` `Tag.StudyID`
- Tidak dapat memperbarui atribut dengan tipe VR SQ (atribut bersarang) kecuali `force` bendera disetel
- Tidak dapat memperbarui atribut multivaluasi kecuali `force` tanda disetel
- Tidak dapat memperbarui atribut dengan nilai yang tidak kompatibel dengan atribut tipe VR kecuali `force` tanda disetel
- Tidak dapat memperbarui atribut yang tidak dianggap sebagai atribut valid sesuai dengan standar DICOM kecuali `force` bendera disetel
- Tidak dapat memperbarui atribut di seluruh modul. Misalnya, jika atribut tingkat Pasien diberikan pada tingkat Studi dalam permintaan muatan pelanggan, permintaan dapat dibatalkan.
- Tidak dapat memperbarui atribut jika modul atribut terkait tidak ada di yang sudah ada`ImageSetMetadata`. Misalnya, Anda tidak diizinkan untuk memperbarui atribut

`seriesInstanceUID` jika Seri dengan tidak `seriesInstanceUID` ada dalam metadata kumpulan gambar yang ada.

HealthImaging Referensi AWS

Bagian ini berisi data pendukung yang relevan dengan AWS HealthImaging.

Topik

- HealthImaging Titik akhir dan kuota AWS
- Batas HealthImaging pelambatan AWS
- Verifikasi data HealthImaging piksel AWS
- HealthImaging Kode Peringatan
- Pustaka decoding bingkai gambar untuk AWS HealthImaging
- Proyek HealthImaging sampel AWS
- Menggunakan layanan ini dengan AWS SDK

HealthImaging Titik akhir dan kuota AWS

Topik berikut berisi informasi tentang titik akhir dan kuota HealthImaging layanan AWS.

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 HealthImaging.

Nama Wilayah	Wilayah	Titik Akhir	Protokol	
US East (N. Virginia)	us-east-1	medical-imaging.us-east-1.amazonaws.com medical-imaging-fips.us-east-1.amazonaws.com	HTTPS HTTPS	
US West (Oregon)	us-west-2	medical-imaging.us-west-2.amazonaws.com medical-imaging-fips.us-west-2.amazonaws.com	HTTPS HTTPS	
Asia Pacific (Sydney)	ap-southeast-2	medical-imaging.ap-southeast-2.amazonaws.com	HTTPS	
Eropa (Irlandia)	eu-west-1	medical-imaging.eu-west-1.amazonaws.com	HTTPS	

Jika Anda menggunakan permintaan HTTP untuk memanggil HealthImaging tindakan AWS, Anda harus menggunakan titik akhir yang berbeda tergantung pada tindakan yang dipanggil. Menu berikut mencantumkan endpoint layanan yang tersedia untuk permintaan HTTP dan tindakan yang mereka dukung.

Tindakan API yang didukung untuk permintaan HTTP

data store, import, tagging

Tindakan penyimpanan, impor, dan penandaan data berikut dapat diakses melalui titik akhir:

`https://medical-imaging.region.amazonaws.com`

- CreateDatastore
- GetDatastore
- ListDatastores

- DeleteDatastore
- Mulai DICOMImport Job
- Dapatkan DICOMImport Job
- Daftar DICOMImport Lowongan
- TagResource
- ListTagsForResource
- UntagResource

image set

Tindakan kumpulan gambar berikut dapat diakses melalui titik akhir:

```
https://runtime-medical-imaging.region.amazonaws.com
```

- SearchImageSets
- GetImageSet
- GetImageSetMetadata
- GetImageFrame
- ListImageSetVersions
- UpdateImageSetMetadata
- CopyImageSet

- DeleteImageSet

DICOMweb

HealthImaging menawarkan representasi dari layanan DICOMweb Retrieve WADO-RS. Untuk informasi selengkapnya, lihat Mengambil data DICOM dari HealthImaging.

DICOMweb Layanan berikut dapat diakses melalui endpoint:

```
https://dicom-medical-imaging.region.amazonaws.com
```

- `GetDICOMInstance`
- `GetDICOMInstanceMetadata`
- `GetDICOMInstanceFrames`

Kuota layanan

Kuota layanan didefinisikan sebagai nilai maksimum untuk sumber daya, tindakan, dan item Anda 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.

Tabel berikut mencantumkan kuota default untuk AWS HealthImaging.

Nama	Default	Dapat disesuaikan	Deskripsi
CopyImageSet Permintaan bersamaan maksimum per penyimpanan data	Setiap Wilayah yang didukung: 100	Ya	CopyImageSet Permintaan bersamaan maksimum per penyimpanan data di Wilayah saat ini AWS

Nama	Default	Dapat disesu an	Deskripsi
DeleteImageSet Permintaan bersamaan maksimum per penyimpanan data	Setiap Wilayah yang didukung: 100	Ya	DeleteImageSet Permintaan bersamaan maksimum per penyimpanan data di Wilayah saat ini AWS
UpdateImageSetMetadata Permintaan bersamaan maksimum per penyimpan an data	Setiap Wilayah yang didukung: 100	Ya	UpdateImageSetMeta data Permintaan bersamaan maksimum per penyimpanan data di Wilayah saat ini AWS
Pekerjaan impor bersamaan maksimum per penyimpanan data	ap-southeast-2:20 Masing-masing Wilayah yang didukung lainnya: 100	Ya	Jumlah maksimum pekerjaan impor bersamaan per penyimpanan data di Wilayah saat ini AWS
Penyimpanan data maksimum	Setiap Wilayah yang didukung: 10	Ya	Jumlah maksimum penyimpanan data aktif di AWS Wilayah saat ini
Jumlah maksimum yang ImageFrames diizinkan untuk disalin per permintaan CopyImageSet	Setiap Wilayah yang didukung: 1.000	Ya	Jumlah maksimum yang ImageFrames diizinkan untuk disalin per CopyImageSet permintaa n di Wilayah saat ini AWS
Jumlah maksimum file dalam pekerjaan impor DICOM	Setiap Wilayah yang didukung: 5.000	Ya	Jumlah maksimum file dalam pekerjaan impor DICOM di Wilayah saat ini AWS

Nama	Default	Dapat disesu an	Deskripsi
Jumlah maksimum folder bersarang dalam pekerjaan impor DICOM	Setiap Wilayah yang didukung: 10.000	Tidak	Jumlah maksimum folder bersarang dalam pekerjaan impor DICOM di Wilayah saat ini AWS
Batas ukuran muatan maksimum (dalam KB) diterima oleh UpdateImageSetMeta data	Setiap Wilayah yang didukung: 10 Kilobyte	Ya	Batas ukuran muatan maksimum (dalam KB) yang diterima oleh UpdateImageSetMeta data di Wilayah saat ini AWS
Ukuran maksimum (dalam GB) dari semua file dalam pekerjaan impor DICOM	Setiap Wilayah yang didukung: 10 Gigabytes	Tidak	Ukuran maksimum (dalam GB) semua file dalam pekerjaan impor DICOM di Wilayah saat ini AWS
Ukuran maksimum (dalam GB) setiap file DICOM P10 dalam pekerjaan impor DICOM	Setiap Wilayah yang didukung: 4 Gigabytes	Tidak	Ukuran maksimum (dalam GB) dari setiap file DICOM P10 dalam pekerjaan impor DICOM di Wilayah saat ini AWS
Batas ukuran maksimum (dalam MB) ImageSetMetadata per Impor, Salin, dan UpdateImageSet	Setiap Wilayah yang didukung: 50 Megabyte	Ya	Batas ukuran maksimum (dalam MB) ImageSetM etadata per Impor, Salin, dan UpdateImageSet di AWS Wilayah saat ini

Batas HealthImaging pelambatan AWS

AWS Akun Anda memiliki batas pembatasan yang berlaku untuk tindakan AWS HealthImaging API. Untuk semua tindakan, `ThrottlingException` kesalahan dilemparkan jika batas pelambatan terlampaui. Untuk informasi selengkapnya, lihat AWS HealthImaging API Referensi.

> **Note**
>
> Batas pelambatan dapat disesuaikan untuk semua tindakan HealthImaging API. Untuk meminta penyesuaian batas pelambatan, hubungi Pusat AWS Dukungan. Untuk membuat kasus, masuk ke AWS akun Anda dan pilih Buat kasus.

Tabel berikut mencantumkan batas pembatasan untuk HealthImaging tindakan asli dan representasi layanan. DICOMweb

Batas HealthImaging pelambatan AWS

Tindakan	Tingkat throttle	Throttle meledak
CreateDatastore	0,085 tps	1 tps
GetDatastore	10 tps	20 tps
ListDatastores	5 tps	10 tps
DeleteDatastore	0,085 tps	1 tps
Mulai DICOMImport Job	1 tps	2 tps
Dapatkan DICOMImport Job	25 tps	50 tps
Daftar DICOMImport Lowongan	10 tps	20 tps
SearchImageSets	25 tps	50 tps
GetImageSet	25 tps	50 tps
GetImageSetMetadata	50 tps	100 tps

Tindakan	Tingkat throttle	Throttle meledak
GetImageFrame	1000 tps	2000 tps
ListImageSetVersions	25 tps	50 tps
UpdateImageSetMetadata	0,25 tps	1 tps
CopyImageSet	0,25 tps	1 tps
DeleteImageSet	0,25 tps	1 tps
TagResource	10 tps	20 tps
ListTagsForResource	10 tps	20 tps
UntagResource	10 tps	20 tps
DICOMInstanceMendapatkan*	50 tps	100 tps
Dapatkan DICOMInstance Metadata*	50 tps	100 tps
Dapatkan DICOMInstance Bingkai*	50 tps	100 tps
Dapatkan DICOMSeries Metadata	50 tps	100 tps

* Representasi layanan DICOMweb

Verifikasi data HealthImaging piksel AWS

Selama impor, HealthImaging berikan verifikasi data piksel bawaan dengan memeriksa status encoding dan decoding lossless dari setiap gambar. Fitur ini memastikan bahwa gambar yang didekodekan menggunakan pustaka decoding HTJ2 K selalu cocok dengan gambar DICOM P10 asli yang diimpor. HealthImaging

- Proses orientasi gambar dimulai ketika pekerjaan impor menangkap status kualitas piksel asli gambar DICOM P10 sebelum diimpor. Image Frame Resolution Checksum (IFRC) unik yang tidak

dapat diubah dihasilkan untuk setiap gambar menggunakan algoritme. CRC32 Nilai checksum IFRC disajikan dalam dokumen metadata. `job-output-manifest.json` Untuk informasi selengkapnya, lihat Memahami pekerjaan impor.

- Setelah gambar diimpor ke penyimpanan HealthImaging data dan diubah menjadi kumpulan gambar, bingkai gambar yang HTJ2 disandikan K segera diterjemahkan dan yang baru dihitung. IFRCs HealthImaging kemudian membandingkan resolusi penuh IFRCs dari gambar asli dengan yang baru IFRCs dari bingkai gambar yang diimpor untuk memverifikasi akurasi.
- Kondisi kesalahan deskriptif per gambar yang sesuai ditangkap dalam log keluaran pekerjaan impor (`job-output-manifest.json`) untuk Anda tinjau dan verifikasi.

Untuk memverifikasi data piksel

1. Setelah mengimpor data pencitraan medis Anda, lihat keberhasilan deskriptif set per gambar (atau kondisi kesalahan) yang ditangkap dalam log keluaran pekerjaan impor,. `job-output-manifest.json` Untuk informasi selengkapnya, lihat Memahami pekerjaan impor.
2. Kumpulan gambar terdiri dari metadata dan bingkai gambar (data piksel). Metadata set gambar berisi informasi tentang bingkai gambar terkait. Gunakan `GetImageSetMetadata` tindakan untuk mendapatkan metadata untuk kumpulan gambar. Untuk informasi selengkapnya, lihat Mendapatkan metadata set gambar.
3. `PixelDataChecksumFromBaseToFullResolution`Berisi IFRC (checksum) untuk gambar resolusi penuh. Untuk gambar yang disimpan dalam sintaks transfer aslinya 1.2.840.10008.1.2.4.203, 1.2.840.10008.1.2.4.91, 1.2.840.10008.1.2.4.50, dan 1.2.840.10008.1.2.1 (Hanya Segmentasi Biner) checksum dihitung pada gambar asli. Untuk gambar yang disimpan dalam HTJ2 K Lossless dengan RPCL, checksum dihitung pada gambar resolusi penuh yang diterjemahkan. Untuk informasi selengkapnya, lihat Sintaks transfer yang didukung.

   Berikut ini adalah contoh keluaran metadata untuk IFRC yang dihasilkan sebagai bagian dari proses pekerjaan impor dan direkam ke. `job-output-manifest.json`

```
"ImageFrames": [{
"ID": "678906789067890123451234512",
"PixelDataChecksumFromBaseToFullResolution": [
{
    "Width": 512,
    "Height": 512,
    "Checksum": 2510355201
```

```
}
]
```

Untuk gambar yang disimpan dalam sintaks transfer aslinya 1.2.840.10008.1.2.4.203, 1.2.840.10008.1.2.4.91, 1.2.840.10008.1.2.4.50, dan 1.2.840.10008.1.2.1 (Hanya Segmentasi Biner) dan tidak akan tersedia. `MinPixelValue` `MaxPixelValue` `FrameSizeInBytes`Ini menunjukkan ukuran bingkai asli.

```
"PixelDataChecksumFromBaseToFullResolution": [
   {"Width": 512, "Height": 512, "Checksum": 1379921327 }
],
"MinPixelValue": null,
"MaxPixelValue": null,
"FrameSizeInBytes": 429
```

Untuk gambar yang disimpan dalam HTJ2 K Lossless dengan RPCL, `FrameSizeInBytes` menunjukkan ukuran bingkai gambar yang diterjemahkan.

```
"PixelDataChecksumFromBaseToFullResolution": [
   {"Width": 512, "Height": 512, "Checksum": 1379921327 }
],
"MinPixelValue": 11,
"MaxPixelValue": 11,
"FrameSizeInBytes": 1652
```

4. Untuk instance yang berisi video, HealthImaging lakukan validasi codec ringan yang memverifikasi sintaks transfer yang ditentukan dalam metadata DICOM cocok dengan codec video.

   HealthImaging menghitung nilai checksum IFRC pada objek video asli menggunakan algoritma. CRC32 Nilai checksum IFRC dicatat `kejob-output-manifest.json`, dan bertahan dalam metadata. HealthImaging Seperti gambar yang disimpan dalam sintaks transfer aslinya (dijelaskan di atas), `MinPixelValue` dan tidak `MaxPixelValue` akan tersedia. FrameSizeInBytes Ini menunjukkan ukuran bingkai asli.

5. HealthImaging verifikasi data piksel, akses prosedur verifikasi data Pixel GitHub dan ikuti instruksi dalam `README.md` file untuk memverifikasi pemrosesan gambar lossless secara independen oleh berbagai Pustaka decoding bingkai gambar yang digunakan oleh. HealthImaging Setelah gambar penuh dimuat, Anda dapat menghitung IFRC untuk data input

mentah di pihak Anda dan membandingkannya dengan nilai IFRC yang disediakan dalam HealthImaging metadata untuk memverifikasi data piksel.

HealthImaging Kode Peringatan

HealthImaging mencoba untuk mengimpor semua data pencitraan medis Anda. Jika ketidaksesuaian data atau elemen data yang tidak dikenal ditemui selama impor, HealthImaging akan menambahkan salah satu peringatan berikut ke file. `warning.ndjson` Peringatan yang terkait dengan instance yang diimpor juga akan dapat dicari melalui `SearchDICOMInstances` tindakan dengan elemen. `WarningReason` Instans yang diimpor dengan peringatan mungkin telah mengurangi dukungan melalui HealthImaging APIs, seperti yang dijelaskan di bawah ini.

HealthImaging Kode Peringatan Impor

Alasan Peringatan (Heksadesimal)	Alasan Peringatan (Desimal)	Jenis Peringatan (Enum)	Detail Peringatan	Perilaku yang dihasilkan
Alasan Peringatan Standar DICOM				
0xB000	45056	PAKSAAN_OF_DATA_ELEMENTS	Konsumsi memodifikasi satu atau lebih elemen data selama penyimpanan instance. Lihat Bagian 6.6.1.3.	T/A
0xB006	45062	ELEMENTS_DISCARDED	Konsumsi membuang beberapa elemen data selama penyimpanan instance. Lihat Bagian 6.6.1.3.	T/A
0xB007	45063	SOP_CLASS_DATA_KETIDAKCOCOKAN	`StoreDICOM` Tindakan mengamati bahwa Kumpulan Data tidak cocok dengan kendala	T/A

Alasan Peringatan (Heksadesimal)	Alasan Peringatan (Desimal)	Jenis Peringatan (Enum)	Detail Peringatan	Perilaku yang dihasilkan
			Kelas SOP selama penyimpanan instance.	
Alasan HealthImaging Peringatan AWS				
0xB100	45312	TRANSCODI NG_EXCEPT ION	Peringatan ini terjadi ketika HealthImaging tidak dapat mentransk ode instance PixelData ke HTJ2 K (format penyimpanan default), atau jika PixelData tidak dapat ditranskode karena alasan lain (yaitu validasi gagal, atribut data piksel yang salah, dll). Dalam hal ini, data piksel akan disimpan sebagai gumpalan.	Data piksel mungkin masih dapat diambil sebagai gumpalan tunggal jika diperlukan, meneruskan wildcard “*” karena header `transfer-syntax` dalam Terima akan mengembalikannya dalam format yang disimpan.
0xB110	45328	FRAMES_EX TRACTION_ FAILURE	Peringatan ini terjadi ketika ada masalah mengurai frame individu dari PixelData berdasark an metadata DICOM yang diberikan.	Data piksel salah bentuk dan tidak dapat diambil, gunakan `GetDICOMI nstance` untuk mengambil seluruh instance

Alasan Peringatan (Heksadesimal)	Alasan Peringatan (Desimal)	Jenis Peringatan (Enum)	Detail Peringatan	Perilaku yang dihasilkan
0xB111	45329	FRAME_NUMBER_MISMATCH	Peringatan ini terjadi ketika elemen DICOM NumberOfFrames "" tidak cocok dengan jumlah sebenarnya dari gambar “fragmen” dalam file DICOM input.	Data piksel salah bentuk dan tidak dapat diambil, gunakan `GetDICOMInstance` untuk mengambil seluruh instance
0xB112	45330	INVALID_OFFSET_TABLE	Peringatan ini terjadi ketika tabel offset dalam fragmen file DICOM input tidak cocok dengan panjang bingkai yang sebenarnya dan dapat mengakibatkan frame cacat tergantung pada tingkat keparahannya.	Data piksel salah bentuk dan tidak dapat diambil, gunakan `GetDICOMInstance` untuk mengambil seluruh instance
0xB120	45344	UNSUPPORTED_TRANSFER_SYNTAX	Peringatan ini terjadi ketika HealthImaging menemukan sintaks transfer yang tidak dikenal atau tidak didukung. Ketika ini terjadi, HealthImaging akan menyimpan data piksel sebagai gumpalan.	Data piksel mungkin masih dapat diambil sebagai gumpalan tunggal jika diperlukan, meneruskan wildcard “*” karena header `transfer-syntax` dalam Terima akan mengembalikannya dalam format yang disimpan.

Alasan Peringatan (Heksadesimal)	Alasan Peringatan (Desimal)	Jenis Peringatan (Enum)	Detail Peringatan	Perilaku yang dihasilkan
0xB201	45570	FORMAT TIDAK VALID	Peringatan ini terjadi ketika satu atau lebih elemen UID melanggar DICOM Value Representation (mis.) `1.2.3..4`	T/A
0xB202	45571	INVALID_DICOM_VALUE_LENGTH	Peringatan ini terjadi ketika elemen DICOM memiliki panjang lebih panjang dari yang didukung oleh Representasi Nilai DICOM, berpotensi memperkenalkan perilaku yang tidak valid untuk tindakan. search/retrieve	Beberapa bidang mungkin tidak dapat diuraikan dan oleh karena itu tidak dapat dicari di (yaitu atau) `StudyDate` `StudyTime`
0xBFFF	47513	LAINNYA	Peringatan ini terjadi ketika ada peringatan yang tidak tertangkap yang HealthImaging tidak menangkap sebagai kode peringatan tertentu.	Data piksel salah bentuk dan tidak dapat diambil, gunakan `GetDICOMInstance` untuk mengambil seluruh instance

Pustaka decoding bingkai gambar untuk AWS HealthImaging

Selama impor, beberapa sintaks transfer mempertahankan pengkodean aslinya, sementara yang lain ditranskode ke High-Throughput JPEG 2000 (HTJ2K) lossless secara default atau JPEG 2000 Lossless (saat dikonfigurasi) tergantung pada konfigurasi datastore Anda. HTJ2K memberikan tampilan gambar yang cepat secara konsisten dan akses universal ke fitur-fitur canggih HTJ2 K. Karena bingkai gambar dikodekan dalam HTJ2 K atau JPEG 2000 Lossless selama impor, mereka

harus diterjemahkan sebelum dilihat di penampil gambar. Untuk informasi tentang menentukan sintaks transfer, lihat Sintaks transfer yang didukung. Untuk informasi tentang menentukan sintaks transfer, lihatSintaks transfer yang didukung.

> **Note**
>
> HTJ2K didefinisikan dalam Bagian 15 dari standar JPEG2 000 (ISO/IEC 15444-15:2019). HTJ2K mempertahankan fitur-fitur canggih dari JPEG2 000 seperti skalabilitas resolusi, kantor polisi, ubin, kedalaman bit tinggi, beberapa saluran, dan dukungan ruang warna.

Topik

- Pustaka decoding bingkai gambar
- Penampil gambar

Pustaka decoding bingkai gambar

Bergantung pada bahasa pemrograman Anda, kami merekomendasikan pustaka decoding berikut untuk memecahkan kode bingkai gambar.

- NVIDIA nv JPEG2 000 - Komersil, dipercepat GPU
- Perangkat Lunak Kakadu - Komersil, C ++ dengan binding Java dan .NET
- OpenJPH — Sumber terbuka, C ++ dan WASM
- OpenJPEG - Sumber terbuka, C/C ++, Java
- openjphpy - Sumber terbuka, Python
- pylibjpeg-openjpeg - Sumber terbuka, Python

Penampil gambar

Anda dapat melihat bingkai gambar setelah Anda mendekodekannya. Tindakan AWS HealthImaging API mendukung berbagai pemirsa gambar sumber terbuka, termasuk:

- Yayasan Pencitraan Kesehatan Terbuka (OHIF)
- Cornerstone.js

Proyek HealthImaging sampel AWS

AWS HealthImaging menyediakan contoh proyek berikut pada GitHub.

OHIF Viewer terintegrasi ke HealthImaging AWS melalui OIDC

AWS Cloud Development Kit (AWS CDK)Proyek ini menyebarkan penampil OHIF di Amazon. CloudFront Penampil terintegrasi ke datastore Amazon Web Services sebagai sumber DICOMWeb data, dan dengan Amazon Cognito sebagai penyedia identitas untuk otentikasi melalui OIDC.

Penyerapan DICOM Dari Lokal ke AWS HealthImaging

Proyek AWS tanpa server untuk menerapkan solusi IoT edge yang menerima file DICOM dari sumber DICOM DIMSE (PACS, VNA, CT scanner) dan menyimpannya dalam bucket Amazon S3 yang aman. Solusi mengindeks file DICOM dalam database dan mengantri setiap seri DICOM untuk diimpor di AWS. HealthImaging Ini terdiri dari komponen yang berjalan di tepi yang dikelola oleh AWS IoT Greengrass, dan pipa konsumsi DICOM yang berjalan di Cloud. AWS

Proksi Penanda Tingkat Ubin (TLM)

AWS Cloud Development Kit (AWS CDK)Proyek untuk mengambil bingkai gambar dari AWS HealthImaging dengan menggunakan penanda tingkat ubin (TLM), fitur High-Throughput JPEG 2000 (K). HTJ2 Ini menghasilkan waktu pengambilan yang lebih cepat dengan gambar beresolusi lebih rendah. Alur kerja potensial termasuk menghasilkan thumbnail dan pemuatan gambar secara progresif.

CloudFront Pengiriman Amazon

Proyek AWS tanpa server untuk membuat CloudFront distribusi Amazon dengan titik akhir HTTPS yang di-cache (dengan menggunakan GET) dan mengirimkan bingkai gambar dari tepi. Secara default, titik akhir mengautentikasi permintaan dengan token web Amazon Cognito JSON (JWT). Otentikasi dan penandatanganan permintaan dilakukan di tepi menggunakan Lambda @Edge. Layanan ini adalah fitur Amazon CloudFront yang memungkinkan Anda menjalankan kode lebih dekat ke pengguna aplikasi Anda, yang meningkatkan kinerja dan mengurangi latensi. Tidak ada infrastuktur untuk dikelola.

AWS HealthImaging Viewer UI

AWS AmplifyProyek untuk menerapkan UI frontend dengan otentikasi backend yang dengannya Anda dapat melihat atribut metadata set gambar dan bingkai gambar (data piksel) yang disimpan di AWS menggunakan decoding progresif. HealthImaging Anda dapat secara opsional

mengintegrasikan proyek CloudFront Pengiriman and/or Amazon Proxy Tile Level Marker (TLM) di atas untuk memuat bingkai gambar menggunakan metode alternatif.

HealthImaging DICOMweb Proksi AWS

Proyek berbasis Python untuk mengaktifkan titik akhir DICOMweb WADO-RS dan QIDO-RS di penyimpanan HealthImaging data untuk mendukung pemirsa pencitraan medis berbasis web dan aplikasi lain yang kompatibel. DICOMweb

> **Note**
>
> Proyek ini tidak menggunakan HealthImaging representasi yang DICOMweb APIs dijelaskan dalamMenggunakan DICOMweb dengan AWS HealthImaging.

Untuk melihat proyek sampel tambahan, lihat AWS HealthImaging Sampel aktif GitHub.

Menggunakan layanan ini 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

Dokumentasi SDK	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 untuk Rust	AWS SDK untuk Rust contoh kode
AWS SDK untuk SAP ABAP	AWS SDK untuk SAP ABAP contoh kode
AWS SDK untuk Swift	AWS SDK untuk 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.

Pengoptimalan Biaya

HealthImaging dirancang untuk mengoptimalkan biaya penyimpanan dengan memindahkan data secara otomatis ke tingkat akses yang paling hemat biaya ketika pola akses berubah. Karena pola penggunaan berubah seiring waktu, tingkatan cerdas menyediakan manajemen siklus hidup otomatis untuk data DICOM tanpa overhead operasional apa pun. HealthImaging memiliki dua tingkatan penyimpanan:

- Penyimpanan tingkat Akses Sering untuk data DICOM yang baru diimpor atau diakses secara teratur.
- Arsipkan penyimpanan tingkat Akses Instan untuk data DICOM yang belum diakses baru-baru ini. Memberikan pengurangan biaya penyimpanan untuk arsip jangka panjang sambil mempertahankan latensi akses milidetik.

Anda ditagih untuk ukuran penyimpanan agregat dari semua set gambar di penyimpanan data. Kedua tingkatan penyimpanan ditagih per GB per bulan, dan tidak ada biaya per set gambar. Juga tidak ada biaya pengambilan untuk memindahkan data antar tingkatan penyimpanan.

> **Note**
>
> Set gambar ditagih dengan ukuran minimal 5 MB. HealthImaging menerima set gambar yang lebih kecil dari 5 MB, tetapi objek yang lebih kecil ini dibebankan pada tingkat 5 MB.

Cara Kerja Tiering Cerdas

Data Anda disimpan di tingkat Akses Sering saat kumpulan gambar baru dibuat. Kumpulan gambar dapat dibuat dengan mengimpor data baru (melalui Impor Pekerjaan atau DICOMweb STOW-RS) atau dengan memanggil. CopyImageSet HealthImaging secara otomatis memindahkan kumpulan gambar yang belum diakses selama 30 hari berturut-turut ke tingkat Akses Instan Arsip, dan kumpulan gambar tetap berada di tingkat Akses Instan Arsip hingga diakses kembali.

Berikut ini adalah tindakan yang merupakan akses ke data DICOM yang secara otomatis memindahkan kumpulan gambar dari tingkat Akses Instan Arsip kembali ke tingkat Akses Sering:

- Memohon GetImageSetMetadata, GetImageFrame, atau tindakan DICOMweb WADO-RS apa pun.

- Memohon CopyImageSetatau UpdateImageSetMetadata. Dalam kasus operasi Copy, hanya kumpulan gambar yang direplikasi yang berjenjang hingga Frequent Access. Dalam kasus Salin dengan tujuan, kumpulan gambar tujuan berjenjang.
- Melihat dan mengunduh data kumpulan gambar melalui konsol AWS HealthImaging manajemen.

Tindakan di atas mempromosikan set gambar ke tingkat Akses Sering dan mencegah set gambar di tingkat Akses Sering agar tidak diturunkan ke tingkat Akses Instan Arsip selama 30 hari lagi. Akses ke set gambar dapat terjadi melalui Konsol AWS Manajemen atau melalui antarmuka terprogram seperti atau. AWS CLI AWS SDKs Tindakan lain bukan merupakan akses dan oleh karena itu tidak akan secara otomatis memindahkan objek dari tingkat Akses Instan Arsip kembali ke tingkat Akses Sering. Berikut ini adalah contoh, bukan daftar definitif, dari tindakan tersebut:

- Tindakan pada penyimpanan data (CreateDatastoredan GetDatastore)
- Daftar tindakan (Daftar DICOMImport Pekerjaan, ListImageSetVersions, dan ListTagsForResource)
- Tindakan pencarian (SearchImageSetsdan kueri DICOMweb QIDO-RS)
- Memohon GetImageSet, TagResource, atau UntagResource
- Hapus operasi

Data yang diimpor ke HealthImaging dikenakan biaya untuk durasi penyimpanan minimum 30 hari. Anda dapat menghapus data Anda kapan saja, tetapi setiap gambar yang dihapus sebelum 30 hari setelah impor akan dikenakan biaya untuk sisa durasi minimum.

Memperkirakan Penyimpanan Data Terstruktur

HealthImaging menyimpan beberapa informasi header DICOM dalam penyimpanan yang diindeks. Anda dikenakan biaya untuk konsumsi penyimpanan terstruktur ini terlepas dari tingkat penyimpanan set gambar, dan biaya ini bersifat aditif. Anda dapat memperkirakan konsumsi penyimpanan terstruktur dengan mengalikan jumlah sumber daya DICOM di penyimpanan data Anda dengan ukuran rekaman yang diindeks per sumber daya. Nilai-nilai berikut adalah perkiraan.

Sumber Daya DICOM	Ukuran Terindeks (byte)
Belajar	1024
Seri	830

Sumber Daya DICOM	Ukuran Terindeks (byte)
Instans	680

AWS HealthImaging rilis

Tabel berikut menunjukkan kapan fitur dan pembaruan dirilis untuk HealthImaging layanan dan dokumentasi AWS. Untuk informasi selengkapnya tentang rilis, lihat topik tertaut.

Perubahan	Deskripsi	Tanggal
CloudWatch Metrik Amazon untuk HealthImaging	HealthImaging menerbitkan metrik tambahan ke CloudWatch dalam `AWS/HealthImaging` namespace, termasuk metrik penggunaan sumber daya di tingkat akun dan penyimpanan data. Untuk informasi selengkapnya, lihat Menggunakan Amazon CloudWatch dengan HealthImaging.	Februari 12, 2026
Peran terkait layanan untuk HealthImaging	HealthImaging sekarang mendukung peran terkait layanan yang memungkinkan layanan untuk mempublikasikan metrik datastore untuk atas nama Anda. CloudWatch `AWSServiceRoleForHealthImaging` Peran dapat dibuat menggunakan konsol IAM atau AWS CLI. Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk. HealthImaging	Februari 9, 2026

AWSHealthImagingServiceRolePolicy kebijakan terkelola	HealthImaging menambahkan kebijakan terkelola baru `AWSHealthImagingServiceRolePolicy` untuk peran terkait layanan yang menyediakan izin untuk mengelola operasi layanan dan mempublikasikan metrik layanan. CloudWatch Untuk informasi selengkapnya, lihat AWSHealthImagingServiceRolePolicy.	Februari 9, 2026
Dukungan JPEG XL untuk penyimpanan data AWS HealthImaging	HealthImaging mendukung mengimpor dan menyimpan file lossy DICOM dalam sintaks transfer JPEG XL (1.2.840.10008.1.2.4.112). Untuk informasi selengkapnya, lihat Sintaks transfer yang didukung.	Februari 2, 2026

Impor DICOM yang disempurnakan dengan dukungan peringatan untuk data yang tidak sesuai	HealthImaging sekarang mengimpor data DICOM yang sebelumnya ditolak dengan menerima file yang tidak sesuai sambil memberikan peringatan terperinci melalui peristiwa. EventBridge Impor pekerjaan sekarang menghasilkan folder PERINGATAN yang berisi `warning.ndjson` file dengan kode alasan peringatan khusus untuk data yang tidak sesuai. Pembaruan menambahkan dukungan elemen DICOM yang dapat dicari WarningReason (0008,1196) dan memperkenalkan `transfer-syntax=*` parameter untuk mengambil instance dalam format tersimpan saat transcoding tidak layak. Lihat Memahami pekerjaan impor dan kode HealthImaging peringatan untuk informasi selengkapnya.	Desember 8, 2025

Dukungan Lossless JPEG 2000 untuk penyimpanan data AWS HealthImaging	AWS HealthImaging sekarang mendukung pengkodean Lossless JPEG 2000 asli untuk gambar medis, memungkinkan Anda membuat datastores yang bertahan dan mengambil bingkai gambar lossless dalam format JPEG 2000 tanpa transcoding. Ini memungkinkan pengambilan latensi yang lebih rendah untuk aplikasi yang memerlukan format Lossless JPEG 2000 saat Anda menentukan `--lossless-storage-format JPEG_2000_LOSSLESS` saat membuat penyimpanan data.	November 25, 2025

Desain Perangkat Tambahan Pencarian QIDO-RS	HealthImaging sekarang mendukung pencarian wildcard pada atribut DICOM utama (Nama Pasien, Nama Dokter Merujuk, ID Pasien, Deskripsi Studi, Modalitas , dan Nomor Aksesi) dan kemampuan pencarian fuzzy untuk Nama Patient/Referring Dokter untuk mengakomo dasi istilah yang tidak lengkap atau salah eja. Pembaruan ini juga memperluas kemampuan kueri QIDO-RS API untuk menyertakan penelusuran Tanggal Lahir Pasien dan Deskripsi Studi, meningkat kan kemampuan untuk menemukan studi dan seri yang relevan. Lihat Mencari data DICOM HealthImaging untuk informasi selengkapnya.	September 15, 2025

Otorisasi OpenID Connect (OIDC) untuk DICOMweb APIs	HealthImaging sekarang mendukung otorisasi token pembawa OpenID Connect (OIDC) di semua. DICOMweb APIs Ini menambahkan interoperabilitas OAuth 2.0 berbasis standar dengan pemirsa umum dan toolkit (misalnya, OHIF, SLIM, MONAI) dengan menerima IDP yang dikeluarkan di header. JWTs `Authorization` Lihat otentikasi OIDC DICOMweb APIs untuk informasi selengkapnya.	September 3, 2025
Support untuk impor DICOMweb data dan DICOMweb Bulkdata	HealthImaging HealthImaging mendukung penyimpanan file DICOM P10 melalui protokol DICOMweb STOW-RS. Lihat Mengimpor data untuk detail selengkapnya. Selain itu, HealthImaging mendukung DICOM Bulkdata untuk memastikan pengambilan metadata latensi rendah yang konsisten. Lihat Mendapatkan data bulkdata DICOM untuk informasi lebih lanjut.	Juni 30, 2025

Organisasi data otomatis, pencarian DICOMweb QIDO-RS, dan DICOMweb peningkatan WADO-RS	HealthImaging menyediakan organisasi otomatis data DICOM P10 pada impor sesuai dengan hierarki tingkat Pasien, Studi, dan Seri dari standar DICOM. Untuk informasi selengkapnya lihat Memahami pekerjaan impor. HealthImaging mendukung pencarian kaya, sesuai standar DICOMweb QIDO-RS. Lihat Mencari data DICOM HealthImaging untuk informasi selengkapnya. HealthImaging mendukung pengambilan metadata untuk semua instance DICOM dalam satu seri melalui satu tindakan API, seperti yang dijelaskan dalam Mendapatkan metadata seri DICOM dari. HealthImaging	22 Mei 2025
Pembuatan set gambar menggunakan lebih sedikit elemen DICOM	HealthImaging mengurangi jumlah elemen yang digunakan saat mengelompokkan objek DICOM P10 yang masuk ke dalam kumpulan gambar. Untuk informasi selengkapnya, lihat Apa yang dimaksud dengan kumpulan gambar? .	Januari 27, 2025

Dukungan lossy untuk Start Job DICOMImport	HealthImaging mendukung mengimpor dan menyimpan file LOSSY DICOM (1.2.840.10008.1.2.4.203, 1.2.840.10008.1.2.4.91, 1.2.840.10008.1.2.4.50) dan file segmentasi biner dalam format aslinya. Untuk informasi selengkapnya, lihat Sintaks transfer yang didukung.	November 1, 2024
Dukungan lossy untuk pengambilan DICOMweb APIs	HealthImaging mendukung pengambilan gambar dan instance yang disimpan di 1.2.840.10008.1.2.4.203, 1.2.840.10008.1.2.4.91, 1.2.840.10008.1.2.4.50, dan 1.2.840.10008.1.2.1 (hanya segmentasi biner) dalam format aslinya atau di Explicit VR Little Endian (1.2.840.10008.1.2.1). Untuk informasi selengkapnya, lihat Sintaks transfer yang didukung dan Mengambil data DICOM.	November 1, 2024
Impor lebih cepat untuk patologi digital	HealthImaging mendukung hingga 6x pekerjaan impor lebih cepat untuk DICOM digital pathology (WSI).	November 1, 2024
Dukungan segmentasi biner	HealthImaging mendukung baik konsumsi dan pengambilan file segmentasi biner DICOM. Untuk informasi selengkapnya, lihat Sintaks transfer yang didukung.	November 1, 2024

Kembalikan ke ID versi set gambar sebelumnya	HealthImaging menyediakan `revertToVersionId` parameter untuk kembali ke ID versi set gambar sebelumnya. Untuk informasi selengkapnya, lihat `revertToVersionId` di AWS HealthImaging API Referensi.	Juli 24, 2024

Fungsionalitas paksa untuk modifikasi set gambar	HealthImaging menyediakan tipe `Overrides` data dengan parameter `forced` permintaan opsional. Pengaturan parameter ini memaksa `UpdateImageSetMetadata` dan `CopyImageSet` tindakan, bahkan jika metadata tingkat Pasien, Studi, atau Seri tidak cocok. Untuk informasi selengkapnya, lihat Mengganti di Referensi AWS HealthImaging API.
	• `UpdateImageSetMetadata` fungsi paksa-HealthImaging memperkenalkan parameter `force` permintaan opsional untuk memperbarui atribut berikut:
	  • `Tag.StudyInstanceUID` , `Tag.SeriesInstanceUID` , `Tag.SOPInstanceUID` , dan `Tag.StudyID`
	  • Menambahkan, menghapus, atau memperbarui elemen data DICOM pribadi tingkat instans
	Untuk informasi selengkapnya, lihat UpdateImageSetMetadatadi AWS HealthImaging API Referensi.
		Juli 24, 2024

	• `CopyImageSet` fungsi paksa- HealthImaging memperkenalkan parameter `force` permintaan opsional untuk menyalin set gambar. Menyetel parameter ini memaksa `CopyImageSet` tindakan, bahkan jika metadata tingkat Pasien, Studi, atau Seri tidak cocok di seluruh dan. `sourceImageSet` `destinationImageSet` Dalam kasus ini, metadata yang tidak konsisten tetap tidak berubah di. `destinationImageSet` Untuk informasi selengkapnya, lihat CopyImageSetdi AWS HealthImaging API Referensi.	
Salin subset dari Contoh SOP	HealthImaging meningkatkan `CopyImageSet` tindakan sehingga Anda dapat memilih satu atau lebih Instans SOP dari `sourceImageSet` untuk menyalin ke. `destinationImageSet` Untuk informasi selengkapnya, lihat Menyalin kumpulan gambar.	Juli 24, 2024

GetDICOMInstanceMetadata untuk mengembalikan metadata instance DICOM	HealthImaging menyediakan GetDICOMInstanceMetadata API untuk mengembalikan metadata DICOM Bagian 10 (.jsonfile). Untuk informasi selengkapnya, lihat Mendapatkan metadata instance.	Juli 11, 2024
GetDICOMInstanceFrames untuk mengembalikan bingkai instance DICOM (data piksel)	HealthImaging menyediakan GetDICOMInstanceFrames API untuk mengembalikan bingkai DICOM Bagian 10 (multipart permintaan). Untuk informasi selengkapnya, lihat Mendapatkan bingkai instans.	Juli 11, 2024

Dukungan yang ditingkatkan untuk impor data DICOM non-standar	HealthImaging memberikan dukungan untuk impor data yang mencakup penyimpan gan dari standar DICOM. Untuk informasi selengkapnya, lihat kendala elemen DICOM.
	• Elemen data DICOM berikut dapat mencapai 256 karakter dalam panjang maksimal:
	  • `Patient's Name (0010,0010)`
	  • `Patient ID (0010,0020)`
	  • `Accession Number (0008,0050)`
	• Variasi sintaks berikut diizinkan untuk`Study Instance UID`,,`Series Instance UID`,`Treatment Session UID`, `Manufactu rer's Device Class UIDDevice UID`, dan`Acquisition UID` :
	  • Elemen pertama dari setiap UID bisa nol
	  • UIDs dapat dimulai dengan satu atau lebih angka nol terkemuka
	  • UIDs bisa sampai 256 karakter panjangnya
		Juni 28, 2024

Pemberitahuan acara	HealthImaging terintegrasi dengan Amazon EventBridge untuk mendukung aplikasi berbasis peristiwa. Untuk informasi selengkapnya, lihat Menggunakan EventBridge.	Juni 5, 2024
`GetDICOMInstance` untuk mengembalikan data instans DICOM	HealthImaging menyediakan `GetDICOMInstance` layanan untuk mengembalikan data instance DICOM Bagian 10 (`.dcm`file). Untuk informasi selengkapnya, lihat Mendapatkan instance.	15 Mei 2024
Impor lintas akun	HealthImaging menyediakan dukungan untuk impor data dari bucket Amazon S3 yang terletak di Wilayah lain yang didukung. Untuk informasi selengkapnya, lihat Impor lintas akun.	15 Mei 2024

Perangkat tambahan pencarian untuk set gambar	HealthImaging `SearchImageSets` tindakan mendukung penyempurnaan pencarian berikut. Untuk informasi selengkapnya, lihat Mencari kumpulan gambar. • Dukungan tambahan untuk mencari di `UpdatedAt` dan `SeriesInstanceUID` • Cari antara waktu mulai dan waktu akhir • Urutkan hasil pencarian berdasarkan `Ascending` atau `Descending` • Parameter Seri DICOM dikembalikan dalam tanggapan	April 3, 2024
Ukuran file maksimum untuk impor meningkat	HealthImaging mendukung ukuran file maksimum 4 GB untuk setiap file DICOM P10 dalam pekerjaan impor. Untuk informasi selengkapnya, lihat Kuota layanan.	6 Maret 2024

Sintaks transfer untuk JPEG Lossless dan K HTJ2	HealthImaging menyediakan dukungan untuk sintaks transfer berikut untuk impor pekerjaan. Untuk informasi selengkapnya, lihat Sintaks transfer yang didukung. • 1.2.840.10008.1.2.4.57 - JPEG Lossless Non-Hierarkis (Proses 14) • 1.2.840.10008.1.2.4.201 - Kompresi Gambar JPEG 2000 Throughput Tinggi (Hanya Lossless) • 1.2.840.10008.1.2.4.202 - High-Throughput JPEG 2000 dengan Kompresi Gambar Opsi RPCL (Hanya Lossless) • 1.2.840.10008.1.2.4.203 - Kompresi Gambar JPEG 2000 Throughput Tinggi	Februari 16, 2024
Contoh kode yang diuji	HealthImaging dokumentasi menyediakan contoh kode yang diuji untuk AWS CLI dan AWS SDKs untuk Python,, Java JavaScript, dan C ++. Untuk informasi selengkapnya, lihat Contoh kode.	Desember 19, 2023
Jumlah file maksimum untuk impor meningkat	HealthImaging mendukung hingga 5.000 file untuk satu pekerjaan impor. Untuk informasi selengkapnya, lihat Kuota layanan.	Desember 19, 2023

Folder bersarang untuk impor	HealthImaging mendukung hingga 10.000 folder bersarang untuk satu pekerjaan impor. Untuk informasi lebih lanjut, lihat Kuota layanan.	1 Desember 2023
Impor lebih cepat	HealthImaging menyediakan impor 20X lebih cepat di semua Wilayah yang didukung. Untuk informasi selengkapnya, lihat Titik akhir layanan.	1 Desember 2023
CloudFormation dukungan	HealthImaging mendukung infrastruktur sebagai kode (IAc) untuk penyediaan penyimpanan data. Untuk informasi selengkapnya, lihat Membuat HealthImaging sumber daya dengan CloudFormation.	21 September 2023
Ketersediaan umum	AWS HealthImaging tersedia untuk semua pelanggan di Wilayah AS Timur (Virginia N.), AS Barat (Oregon), Eropa (Irlandia), dan Asia Pasifik (Sydney). Untuk informasi selengkapnya, lihat Titik akhir layanan.	26 Juli 2023

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.