aws

Guía para desarrolladores

AWS HealthLake

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

AWS HealthLake: Guía para desarrolladores

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

Las marcas comerciales y la imagen comercial de Amazon no se pueden utilizar en relación con ningún producto o servicio que no sea de Amazon, de ninguna manera que pueda causar confusión entre los clientes y que menosprecie o desacredite a Amazon. Todas las demás marcas registradas que no son propiedad de Amazon son propiedad de sus respectivos propietarios, que pueden o no estar afiliados, conectados o patrocinados por Amazon.

Table of Contents

¿Qué es AWS HealthLake? 1
Aviso importante 2
Características 2
Servicios relacionados 3
Acceso 4
HIPAA 4
Precios 5
Introducción 6
Conceptos 6
estrategia de autorización 7
PNL integrada 7
Analítica integrada 7
Configuración 7
Inscríbase en una Cuenta de AWS 8
Configure un usuario o un rol de IAM 8
Agregue un usuario o rol de administrador de Data Lake 10
Crear buckets de S3 12
Crear un almacén de datos 12
Configure los permisos de importación 13
Configure los permisos de exportación 15
Instala la AWS CLI 19
Tutorial 20
Administración de almacenes de datos 21
Creación de un almacén de datos 21
Obtención de propiedades de los almacenes de datos 29
Enumeración de almacenes de datos 33
Etiquetado de almacenes de datos 37
Etiquetar un almacén de datos 37
Listar etiquetas para un banco de datos 41
Eliminar el etiquetado de un banco de datos 44
Eliminación de almacenes de datos 47
Gestión de las suscripciones del FHIR 52
¿Cómo funcionan las suscripciones del FHIR 52
Componentes principales 53

Temas de suscripción 53
Suscripciones 53
Canales de notificación 54
Cargas útiles de notificación 54
Prácticas recomendadas 54
Ciclo de vida de 55
Crear una suscripción 57
Ejemplo de cargas útiles de suscripción 60
Ejemplo de cargas útiles de notificación 65
Buscando suscripciones 69
Filtrar notificaciones 72
Importación de datos del FHIR 76
Inicio de un trabajo de importación 78
Obtención de las propiedades de trabajos de importación 83
Enumeración de trabajos de importación 87
Administrar los recursos del FHIR 93
Creación de un recurso 95
Lectura de un recurso 98
Lectura del historial de recursos 100
Lectura del historial de versiones específicas 103
Actualización de un recurso 105
Actualización condicional 108
Configurar el nivel de validación para las actualizaciones de recursos 109
Modificación de un recurso 110
Formatos PATCH compatibles 110
De uso 111
Formato de parche JSON 111
FHIRPath Formato de parche 114
Encabezados de solicitud 116
Respuesta de ejemplo 116
Comportamiento 117
Gestión de errores 117
Resumen de capacidades 118
Limitaciones 118
Recursos adicionales 118
Agrupar recursos 119

Agrupa como entidades independientes 123
Condicional PUTs 127
Agrupar como una sola entidad 131
Configurar el nivel de validación de los paquetes 134
Soporte limitado para el tipo de paquete «mensaje» 135
Transacciones asíncronas 137
Eliminación de un recurso 146
Eliminación condicional para el FHIR 149
Idempotencia y concurrencia 151
Claves de idempotencia 151
ETag en AWS HealthLake 152
Buscando recursos del FHIR 155
Buscando con GET 155
Ejemplos de búsqueda GET 158
¿Buscando con POST 159
Ejemplos de búsqueda POST 162
Niveles de consistencia de búsqueda 164
Niveles de consistencia 165
Ejemplo de uso 165
Prácticas recomendadas 166
Exportación de datos del FHIR 167
Empezar un trabajo de exportación 167
Conseguir propiedades de trabajo de exportación 172
Listado de trabajos de exportación 176
Ejemplos de código 182
Conceptos básicos 182
Acciones 183
Integración 230
Procesamiento de lenguaje natural 230
Bibliotecas de PNL 231
Uso del FHIR APIs 233
Parámetros de búsqueda 233
Solicitudes de ejemplo 236
Índice y consulta de SQL 253
Introducción 254
Consultas con SQL 258

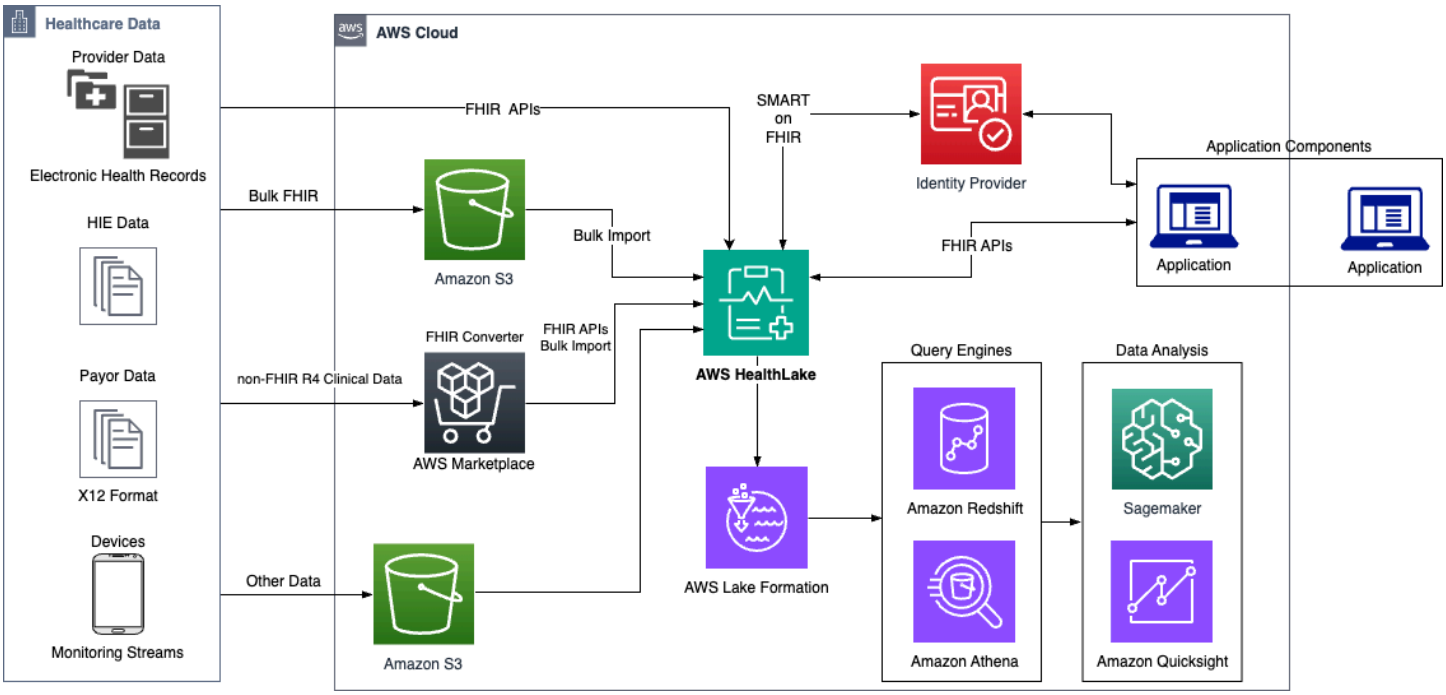
Consultas de ejemplo 265
Supervisión 272
CloudTrail (llamadas a la API) 273
AWS HealthLake Información en CloudTrail 273
Descripción de las entradas de los archivos de AWS HealthLake registro 275
CloudWatch (Métricas) 276
Visualización de HealthLake métricas 279
Crear una alarma de 279
EventBridge (Eventos) 280
HealthLake eventos enviados a EventBridge 281
HealthLake estructura de eventos 282
Seguridad 297
Protección de los datos 298
Cifrado en reposo 299
Clave KMS propiedad de AWS 299
Clave de KMS administradas por el cliente 299
Creación de una clave administrada por el cliente 300
Permisos de IAM necesarios cuando al utilizar una clave de KMS administrada por el cliente 302
Cifrado en tránsito 309
Identity and Access Management 309
Público 309
Autenticación con identidades 310
Administración del acceso con políticas 311
¿Cómo AWS HealthLake funciona con IAM 313
Ejemplos de políticas basadas en identidades 318
AWS políticas gestionadas 322
Resolución de problemas 326
Validación de conformidad 328
Seguridad de la infraestructura 329
Infraestructura como código 329
HealthLake y CloudFormation plantillas 329
Más información sobre CloudFormation 330
Puntos de conexión de VPC 330
Consideraciones sobre los puntos HealthLake finales de VPC 330
Crear un punto final de VPC de interfaz para; HealthLake 331

Crear una política de puntos de conexión de VPC para HealthLake 331
Prácticas recomendadas 332
Resiliencia 333
Referencia 334
SMART sobre el FHIR 334
Introducción 335
Autenticación 338
OAuth osciloscopios 2.0 340
Validación de tókenes 343
Autorización detallada 355
Documento de descubrimiento 356
Ejemplo de solicitud 358
FHIR R4 359
Declaración de capacidad 359
Validaciones de perfil 360
Tipos de recurso 368
Parámetros de búsqueda 370
$ Operaciones 384
Conformidad 536
CMS 537
HealthLake 543
Cuotas y puntos de conexión 543
Tipos de datos precargados 555
Proyectos de ejemplo 556
Resolución de problemas 556
Trabajando con AWS SDKs 565
Versiones 567
..... dxciv

¿Qué es AWS HealthLake?

AWS HealthLake es un servicio que cumple con los requisitos de la HIPAA y permite almacenar, analizar y compartir datos de salud en la nube mediante la especificación R4 de los Recursos de Interoperabilidad Sanitaria Rápida (FHIR). HealthLake los casos de uso incluyen:

- Datos de salud empresariales: administre y comparta directamente los datos de salud del FHIR R4 y, Nube de AWS al mismo tiempo, mantenga el alto rendimiento y la disponibilidad.
- Interoperabilidad de la atención médica: Support el cumplimiento de los clientes con la Ley de Curaciones del Siglo XXI para el acceso de los pacientes a través de un almacén de datos del FHIR totalmente gestionado.
- Procesamiento del lenguaje natural (PNL): utilice modelos de PNL integrados para extraer información médica significativa a partir de datos de salud no estructurados.
- Análisis multimodal: combine HealthLake datos con datos y AWS HealthImaging AWS HealthOmics datos para ofrecer información útil para la medicina de precisión.



Temas

- Aviso importante
- Características de AWS HealthLake

- Servicios relacionados AWS
- Acceder AWS HealthLake
- Cumplimiento con para la HIPAA y seguridad de la información
- Precios

Aviso importante

AWS HealthLake no sustituye el consejo, el diagnóstico o el tratamiento de un médico profesional, y no pretende curar, tratar, mitigar, prevenir o diagnosticar ninguna enfermedad o afección de salud. Usted es responsable de instituir la revisión humana como parte de cualquier uso de cualquier producto de terceros destinado a fundamentar la toma de decisiones clínicas AWS HealthLake, incluso en asociación con él. AWS HealthLake solo debe utilizarse en la atención de pacientes o en situaciones clínicas tras su revisión por parte de profesionales médicos cualificados que apliquen un criterio médico sólido.

Características de AWS HealthLake

AWS HealthLake proporciona las siguientes funciones.

Importe los datos de salud del FHIR R4

Con la acción de importación HealthLake nativa, puede migrar fácilmente sus datos del FHIR de un bucket de Amazon S3 a un almacén de HealthLake datos, incluidas notas clínicas, informes de laboratorio, reclamaciones de seguros y más. HealthLake es compatible con la especificación R4 del FHIR para el intercambio de datos de atención médica. Si es necesario, puede trabajar con un AWS HealthLake socio para convertir sus datos de salud al formato FHIR R4.

Almacene los datos de salud de forma segura, compatible y auditable

Un almacén HealthLake de datos ayuda a indexar los datos de salud para poder consultarlos. El almacén de datos crea una vista completa del historial médico de cada paciente en orden cronológico y facilita el intercambio de información mediante la especificación R4 del FHIR. Además, siempre está activo para mantener su índice actualizado, lo que le permite buscar la información en cualquier momento mediante las interacciones estándar del FHIR R4, con almacenamiento primario duradero y escalado de índices.

Aproveche el servidor FHIR transaccional

Aproveche el FHIR APIs para la validación de recursos estándar, la autorización SMART para el FHIR y la exportación masiva de datos a través de la API FHIR para unificar y analizar sus datos a fin de reducir los costes operativos y mejorar la toma de decisiones. HealthLake respalda la conformidad del cliente con las normas reglamentarias más recientes de la ONC y el CMS, incluidas las siguientes: HL7 FHIR R4 APIs, FHIR Bulk Data Access, US Core IG STU, HL7 SMART App Launch Framework IG, 2.0 OAuth y OpenID Connect.

Transforme los datos médicos no estructurados mediante la PNL

El procesamiento médico del lenguaje natural (PNL) integrado transforma todos los datos textuales médicos sin procesar en un almacén de HealthLake datos para comprender y extraer información significativa de los datos de salud no estructurados. Con la PNL médica integrada, puede extraer automáticamente las entidades, las relaciones entre las entidades, los rasgos de las entidades y la información de salud protegida (PHI) de su texto médico. Las entidades extraídas de la PNL se almacenan como recursos nativos de FHIR R4 en un almacén de HealthLake datos y se puede acceder a ellas a través de FHIR R4 o APIs Amazon Athena (SQL).

Servicios relacionados AWS

AWS HealthLake presenta una estrecha integración con otros AWS servicios. Es útil conocer los siguientes servicios para aprovecharlos al máximo HealthLake.

- AWS Identity and Access Management— Utilice la IAM para gestionar de forma segura las identidades y el acceso a HealthLake los recursos.
- Amazon Simple Storage Service: utilice Amazon S3 como área de almacenamiento provisional para importar datos DICOM. HealthLake
- AWS CloudTrail— Úselo CloudTrail para realizar un seguimiento de la actividad HealthLake de los usuarios y el uso de la API.
- Amazon CloudWatch: se usa CloudWatch para observar y monitorear HealthLake los recursos.
- AWS CloudFormation— Se utilizan CloudFormation para implementar plantillas de infraestructura como código (IaC) para crear recursos. HealthLake
- AWS PrivateLink— Utilice Amazon VPC para establecer la conectividad entre Amazon Virtual Private Cloud HealthLake y Amazon sin exponer los datos a Internet.
- Amazon EventBridge: utilícelo EventBridge para crear aplicaciones escalables y basadas en eventos mediante la creación de reglas que dirijan HealthLake los eventos a los objetivos.

- AWS Lake Formation— Utilice Lake Formation para gestionar, proteger y compartir HealthLake datos de forma centralizada para el análisis y el aprendizaje automático.
- Amazon Athena: utilice Athena para consultar HealthLake datos con SQL para permitir un análisis más profundo.

Acceder AWS HealthLake

Puede acceder AWS HealthLake mediante el Consola de administración de AWS, AWS Command Line Interface y el AWS SDKs. Esta guía proporciona instrucciones de procedimiento Consola de administración de AWS y ejemplos de código para AWS CLI y AWS SDKs.

AWS Command Line Interface (AWS CLI)

AWS CLI Proporciona comandos para un amplio conjunto de AWS productos y es compatible con Windows, Mac y Linux. Para obtener más información, consulte la Guía del usuario de AWS Command Line Interface .

AWS SDKs

AWS SDKs proporciona bibliotecas, ejemplos de código y otros recursos para los desarrolladores de software. Estas bibliotecas proporcionan funciones básicas que automatizan tareas como la firma criptográfica de las solicitudes, el reintento de las solicitudes o el tratamiento de las respuestas de error. Para obtener más información, consulte Herramientas sobre las que construir AWS.

Consola de administración de AWS

Consola de administración de AWS Proporciona una interfaz de usuario basada en la web para administrar los recursos HealthLake y sus recursos asociados. Si ha creado una AWS cuenta, puede iniciar sesión en la HealthLake consola.

Cumplimiento con para la HIPAA y seguridad de la información

Se trata de un servicio compatible con HIPAA. Para obtener más información sobre AWS la Ley de Portabilidad y Responsabilidad de los Seguros de Salud de los Estados Unidos de 1996 (HIPAA) y el uso de AWS los servicios para procesar, almacenar y transmitir información de salud protegida (PHI), consulte Descripción general de la HIPAA.

Las conexiones que HealthLake contienen la PHI y la información de identificación personal (PII) deben estar cifradas. De forma predeterminada, todas las conexiones deben HealthLake usar HTTPS a través de TLS. HealthLake almacena el contenido cifrado de los clientes y funciona de acuerdo con el modelo de responsabilidad AWS compartida.

Precios

Para obtener información sobre HealthLake precios, consulte AWS HealthLake precios. Para estimar los costos, usa la calculadora de HealthLake precios.

Empezar con AWS HealthLake

Para empezar a usarlo AWS HealthLake, configura una AWS cuenta y crea un AWS Identity and Access Management usuario. Para usar el AWS CLIo el AWS SDKs, debe instalarlos y configurarlos.

> **Note**
>
> El Referencia capítulo de esta guía proporciona contenido de apoyo para SMART en FHIR, FHIR R4 y. AWS HealthLake Por ejemplo, puede encontrar información sobre la configuración de SMART en el FHIR, las validaciones de perfil del FHIR compatibles y los puntos finales. HealthLake

Tras aprender HealthLake los conceptos y la configuración, dispondrá de un breve tutorial con ejemplos de código que le ayudará a empezar.

Temas

- AWS HealthLake conceptos
- Configuración AWS HealthLake
- AWS HealthLake tutorial

AWS HealthLake conceptos

La siguiente terminología y conceptos son fundamentales para su comprensión y uso de AWS HealthLake.

Conceptos

- Estrategia de autorización del almacén de datos
- PNL integrada
- Analítica integrada

Estrategia de autorización del almacén de datos

Un almacén HealthLake de datos es un repositorio de datos de salud del FHIR R4 que se encuentra dentro de una sola unidad. Región de AWS HealthLake admite las siguientes estrategias de autorización de almacenes de datos.

- Autorización SigV4: HealthLake autoriza las llamadas a la API del FHIR mediante la autorización AWS Signature Version 4 (SiGV4).
- Autorización SMART con autorización del FHIR: HealthLake autoriza las llamadas a la API del FHIR mediante aplicaciones médicas sustituibles y tecnologías reutilizables (SMART) con la autorización del FHIR.

Para obtener más información, consulte Creación de un almacén HealthLake de datos.

PNL integrada

AWS HealthLake se integra con las bibliotecas de procesamiento del lenguaje natural (PNL) aptas para la HIPAA para extraer datos de salud significativos a partir de textos médicos no estructurados. Las bibliotecas de PNL identifican entidades médicas, como afecciones, medicamentos, dosis, pruebas, tratamientos y procedimientos. Reconocen las relaciones entre las entidades y las vinculan con bibliotecas de ontología médica, como la ICD-10-CM y. RxNorm Para obtener más información, consulte Procesamiento de lenguaje natural (PNL) integrado para HealthLake.

Analítica integrada

AWS HealthLake va más allá del `bundle` APIs FHIR `search` y proporciona análisis integrados para consultar y analizar grandes volúmenes de datos de salud. Durante la importación, genera HealthLake automáticamente tablas para el índice y la consulta de SQL. Esto le permite obtener información útil a partir de datos sanitarios complejos sin necesidad de un trabajo exhaustivo de ingeniería de datos. Para obtener más información, consulte Consulta de HealthLake datos con Amazon Athena y AWS HealthLake proyectos de muestra.

Configuración AWS HealthLake

En este capítulo, se utilizan Consola de administración de AWS para configurar los permisos necesarios para empezar a utilizar AWS HealthLake y crear un almacén de datos. Para configurar los permisos para crear un banco de datos, debe crear un usuario o rol de IAM que sea administrador y

HealthLake administrador del lago de datos. Convierte a este usuario en administrador de lagos de datos en AWS Lake Formation. El administrador del lago de datos concede a Lake Formation acceso a los recursos necesarios para utilizar Amazon Athena para consultar un almacén de datos. Después de crear un almacén de HealthLake datos, puede configurar los permisos para importar y exportar archivos.

Temas

- Inscríbase en una Cuenta de AWS
- Configure el usuario o rol de IAM que desee utilizar HealthLake (administrador de IAM)
- Agregue un usuario o rol como administrador de Data Lake en Lake Formation (administrador de IAM)
- Crear buckets de S3
- Crear un almacén de datos
- Configurar los permisos para los trabajos de importación
- Configurar los permisos para los trabajos de exportación
- Instala la AWS CLI

Inscríbase en una Cuenta de AWS

Para empezar AWS, necesitas un Cuenta de AWS. Para obtener información sobre cómo crear un Cuenta de AWS, consulte Cómo empezar con un Cuenta de AWS en la Guía de AWS Account Management referencia.

Configure el usuario o rol de IAM que desee utilizar HealthLake (administrador de IAM)

> Perfil: administrador de IAM
>
> Un usuario que puede crear usuarios y funciones de IAM y añadir administradores de lagos de datos.

Los pasos de este tema los debe llevar a cabo un administrador de IAM.

Para conectar su almacén de HealthLake datos a Athena, debe crear un usuario o rol de IAM que sea administrador y administrador del lago de datos. HealthLake Este nuevo usuario o rol otorga

acceso a los recursos que se encuentran en un almacén de datos a través de AWS Lake Formation y tiene la política `AmazonHealthLakeFullAccess` AWS administrada agregada a su usuario o rol.

> **Important**
>
> Un usuario o rol de IAM que sea administrador de un lago de datos no puede crear nuevos administradores de lagos de datos. Para añadir un administrador de lago de datos adicional, debe utilizar un usuario o rol de IAM al que se le haya concedido `AdministratorAccess` acceso.

Para crear un administrador

1. Añada la política AWS gestionada de **`AmazonHealthlakeFullAccess`** IAM a un usuario o rol de su organización.

   Si no está familiarizado con la creación de un usuario de IAM, consulte Creación de un usuario de IAM y Descripción general de las políticas de AWS IAM en la Guía del usuario de IAM.
2. Conceda al usuario o rol de IAM acceso a AWS Lake Formation.

   - Añada la siguiente política AWS gestionada por IAM a un usuario o rol de su organización: **`AWSLakeFormationDataAdmin`**

     > **Note**
     >
     > La `AWSLakeFormationDataAdmin` política otorga acceso a todos los recursos de AWS Lake Formation. Le recomendamos que utilice siempre los permisos mínimos necesarios para realizar la tarea. Para obtener más información, consulte Prácticas recomendadas de IAM en la Guía del usuario de IAM.
3. Agregue la siguiente política en línea al usuario o rol. Para obtener más información, consulte las políticas integradas en la Guía del usuario de IAM.

   JSON

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

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

Para obtener más información sobre la `AWSLakeFormationDataAdmin` política, consulte la referencia sobre las personas de Lake Formation y los permisos de IAM en la Guía para desarrolladores de AWS Lake Formation.

Agregue un usuario o rol como administrador de Data Lake en Lake Formation (administrador de IAM)

> **Note**
>
> Este paso es obligatorio si va a realizar la integraciónÍndice y consulta de SQL.

A continuación, el administrador de IAM debe añadir el usuario o rol creado en el paso anterior como administrador del lago de datos en Lake Formation.

Para agregar un usuario o rol de IAM como administrador de un lago de datos

1. Abre la consola de AWS Lake Formation: https://console.aws.amazon.com/lakeformation/

   > **Note**
   >
   > Si es la primera vez que visita Lake Formation, aparecerá el cuadro de diálogo Bienvenido a Lake Formation pidiéndole que defina un administrador de Lake Formation.

2. Asigne al nuevo usuario o rol como administrador AWS del lago de datos de Lake Formation.
   - Opción 1: Si ha recibido el cuadro de diálogo Welcome to Lake Formation.
     1. Seleccione Añadir otros AWS usuarios o roles.
     2. Seleccione la flecha hacia abajo (▼).
     3. Elija el HealthLake administrador que le gustaría que también fuera administrador de Lake Formation.
     4. Elija Comenzar.
   - Opción 2: utilice el panel de navegación (☰).
     1. Elija el panel de navegación (☰).
     2. En Permisos, elija Funciones y tareas administrativas.
     3. En la sección Administradores de Data Lake, selecciona Elegir administradores.
     4. En el cuadro de diálogo Administrar administradores de lagos de datos, seleccione la flecha hacia abajo (▼).
     5. A continuación, seleccione o busque los HealthLake administradores, usuarios o roles que también desee que sean administradores de Lake Formation.
     6. Seleccione Save.
3. Cambie la configuración de seguridad predeterminada para que la administre Lake Formation. Los recursos del almacén de HealthLake datos deben ser gestionados por Lake Formation, no por IAM. Para actualizar, consulte Cambiar el modelo de permisos predeterminado en la Guía para desarrolladores de AWS Lake Formation.

Crear buckets de S3

Para importar datos de FHIR R4 AWS HealthLake, se recomiendan dos depósitos de Amazon S3. El depósito de entrada de Amazon S3 contiene los datos del FHIR que se van a importar y los HealthLake lee de este depósito. El bucket de salida de Amazon S3 almacena los resultados del procesamiento del trabajo de importación y HealthLake escribe (registra) en este bucket.

> **Note**
>
> Debido a la política AWS Identity and Access Management (IAM), los nombres de los buckets de Amazon S3 deben ser únicos. Para más información, consulte Reglas de nomenclatura de buckets en la Guía del usuario de Amazon Simple Storage Service.

A los efectos de esta guía, especificamos los siguientes buckets de entrada y salida de Amazon S3 al configurar los permisos de importación más adelante en esta sección.

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

Para más información, consulte Crear un bucket en la Guía del usuario de Amazon S3.

Crear un almacén de datos

Un almacén HealthLake de datos es un repositorio de datos del FHIR R4 que reside en una sola región. AWS Una AWS cuenta puede tener cero o varios almacenes de datos. HealthLake admite dos estrategias de autorización de almacenes de datos.

> **Importante**
>
> Antes de crear un banco de HealthLake datos, revise las políticas de control de servicios (SCP) de su AWS organización que puedan restringir la creación o la administración de HealthLake los recursos. Los SCP pueden impedir la creación correcta de almacenes de HealthLake datos, incluso si sus permisos de IAM están configurados correctamente.
> A `datastoreID` se genera al crear un banco de HealthLake datos. Debe utilizarla `datastoreID` al configurar los permisos de importación más adelante en esta sección.

Para crear un HealthLake banco de datos, consulteCreación de un almacén HealthLake de datos.

Configurar los permisos para los trabajos de importación

Antes de importar archivos a un almacén de datos, debe conceder HealthLake permiso para acceder a los depósitos de entrada y salida en Amazon S3. Para conceder el HealthLake acceso, debe crear un rol de servicio de IAM HealthLake, añadir una política de confianza al rol para conceder permisos de HealthLake asumir el rol y adjuntar una política de permisos al rol que le conceda acceso a sus buckets de Amazon S3.

Al crear un trabajo de importación, debe especificar el nombre de recurso de Amazon (ARN) de este rol para el. `DataAccessRoleArn` Para obtener más información sobre las funciones de IAM y las políticas de confianza, consulte Funciones de IAM.

Tras configurar el permiso, estará listo para importar archivos a su almacén de datos con un trabajo de importación. Para obtener más información, consulte Inicio de un trabajo de importación del FHIR.

Para configurar los permisos de importación

1. Si aún no lo ha hecho, cree un bucket de Amazon S3 de destino para los archivos de registro de salida. El bucket de Amazon S3 debe estar en la misma AWS región que el servicio y bloquear el acceso público debe estar activado en todas las opciones. Para obtener más información, consulte Uso de Amazon S3 para bloquear el acceso público. También se debe usar una Amazon-owned clave KMS propiedad del cliente para el cifrado. Para obtener más información sobre el uso de claves de KMS, consulte Amazon Key Management Service.
2. Cree una función de servicio de acceso a datos HealthLake y dé permiso al HealthLake servicio para que la asuma con la siguiente política de confianza. HealthLake usa esto para escribir el bucket Amazon S3 de salida.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "healthlake.amazonaws.com"
                ]
```

```
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "accountID"
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:healthlake:us-
west-2:111122223333:datastore/fhir/datastoreID"
                }
            }
        }
    ]
}
```

3. Añada una política de permisos a la función de acceso a los datos que le permita acceder al bucket de Amazon S3. `amzn-s3-demo-bucket`Sustitúyalo por el nombre de tu bucket.

   JSON

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

```
        "Action": [
            "kms:DescribeKey",
            "kms:GenerateDataKey*"
        ],
        "Resource": [
            "arn:aws:kms:us-east-1:012345678910:key/d330e7fc-b56c-4216-a250-
f4c43ef46e83"
        ],
        "Effect": "Allow"
    }]
}
```

Configurar los permisos para los trabajos de exportación

Antes de exportar archivos desde un almacén de datos, debe conceder HealthLake permiso para acceder al depósito de salida en Amazon S3. Para conceder el HealthLake acceso, debe crear un rol de servicio de IAMHealthLake, añadir una política de confianza al rol para conceder permisos de HealthLake asumir el rol y adjuntar una política de permisos al rol que le conceda acceso a su bucket de Amazon S3.

Si ya has creado un rol para HealthLake, puedes reutilizarlo y concederle los permisos adicionales para tu bucket de exportación de Amazon S3 que se indican en este tema. Para obtener más información sobre las políticas de confianza y los roles de IAM, consulte Políticas y permisos de IAM.

> **Importante**
>
> HealthLake admite tanto las solicitudes de exportación del SDK nativo como la operación FHIR R4 `$export`. Se deben proporcionar acciones de IAM independientes en función de la API de exportación que decida utilizar. Esto le permite gestionar `allow` los `deny` permisos por separado. Si desea restringir las exportaciones de la API REST HealthLake del SDK y del FHIR, debe aplicar los permisos de denegación a las distintas acciones de IAM. No es necesario cambiar los permisos de los usuarios de IAM si les das a los usuarios acceso total a ellos. HealthLake
>
> Utilización AWS CLI and AWS SDK:
>
> Las siguientes HealthLake acciones nativas están disponibles para exportar datos de un banco de datos mediante los AWS SDK AWS CLI y:
>
> - `StartFHIRExportJob`

- DescribeFHIRExportJob
- ListFHIRExportJobs

Uso de las API del FHIR:

Las siguientes acciones de IAM están disponibles para exportar datos de un banco de HealthLake datos y para cancelar (eliminar) un trabajo de exportación mediante la operación FHIR: $export
POST:

- StartFHIRExportJobWithPost

GET:

- StartFHIRExportJobWithGet
- DescribeFHIRExportJobWithGet
- GetExportedFile

DELETE:

- CancelFHIRExportJobWithDelete

El usuario o rol que configura los permisos debe tener permiso para crear roles, crear políticas y adjuntar políticas a los roles. La siguiente política de IAM concede estos permisos.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "iam:CreateRole",
                "iam:CreatePolicy",
                "iam:AttachRolePolicy"
            ],
```

```
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": "iam:PassRole",
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "healthlake.amazonaws.com"
                }
            }
        }
    ]
}
```

Para configurar los permisos de exportación

1. Si aún no lo ha hecho, cree un bucket de Amazon S3 de destino para los datos que va a exportar desde su almacén de datos. El bucket de Amazon S3 debe estar en la misma región de AWS que el servicio y bloquear el acceso público debe estar activado en todas las opciones. Para obtener más información, consulte Uso de Amazon S3 para bloquear el acceso público. También se debe usar una Amazon-owned clave KMS propiedad del cliente para el cifrado. Para obtener más información sobre el uso de claves de KMS, consulte Amazon Key Management Service.
2. Si aún no lo has hecho, crea un rol de servicio de acceso a datos HealthLake y dale permiso al HealthLake servicio para que lo asuma con la siguiente política de confianza. HealthLakeusa esto para escribir el bucket Amazon S3 de salida. Si ya has creado uno enConfigurar los permisos para los trabajos de importación, puedes reutilizarlo y concederle permisos para tu bucket de Amazon S3 en el siguiente paso.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
```

```
                "Service": [
                    "healthlake.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "accountID"
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:healthlake:us-
west-2:111122223333:datastore/fhir/data store ID"
                }
            }
        }
    ]
}
```

3. Añada una política de permisos a la función de acceso a los datos que le permita acceder a su bucket de Amazon S3 de salida. `amzn-s3-demo-bucket`Sustitúyalo por el nombre del bucket.

   JSON

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

```
        "Effect": "Allow"
    },
    {
        "Action": [
            "kms:DescribeKey",
            "kms:GenerateDataKey*"
        ],
        "Resource": [
            "arn:aws:kms:us-east-1:012345678910:key/d330e7fc-b56c-4216-a250-
f4c43ef46e83"
        ],
        "Effect": "Allow"
    }]
}
```

Instala la AWS CLI

AWS CLI Es obligatorio para describir y HealthLake enumerar las propiedades de los trabajos de importación y exportación. También puede solicitar esta información mediante HealthLake los SDK.

Para configurar el AWS CLI

1. Descargue y configure la AWS CLI. Para obtener instrucciones, consulte los siguientes temas en la Guía del usuario de AWS Command Line Interface .

   - Instalar o actualizar la última versión del AWS CLI
   - Cómo empezar con el AWS CLI

2. En el AWS CLI `config` archivo, añada un perfil con nombre para el administrador. Este perfil se utiliza al ejecutar los AWS CLI comandos. Según el principio de seguridad de privilegio mínimo, le recomendamos que cree un rol de IAM independiente con privilegios específicos para las tareas que se estén realizando. Para más información sobre los perfiles con nombre, consulte Opciones de los archivos de configuración y credenciales en la Guía del usuario de AWS Command Line Interface .

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

3. Verifique la configuración con el comando `help` siguiente.

```
aws healthlake help
```

Si AWS CLI está configurado correctamente, verá una breve descripción AWS HealthLake y una lista de los comandos disponibles.

AWS HealthLake tutorial

Objetivo

En este tutorial, importará los datos del FHIR R4 a un banco de HealthLake datos mediante acciones nativas HealthLake. A continuación, administrará (creará, leerá, actualizará, eliminará) un recurso del FHIR mediante el FHIR. RESTful APIs Para concluir el tutorial, exportará los datos del FHIR mediante acciones nativas. HealthLake

Requisitos previos

Todos los procedimientos que se enumeran en Configuración son necesarios para completar este tutorial.

Pasos del tutorial

1. Inicie el trabajo de importación del FHIR
2. Obtenga las propiedades del trabajo de importación del FHIR
3. Cree un recurso del FHIR
4. Lea el recurso del FHIR
5. Actualice el recurso del FHIR
6. Eliminar el recurso FHIR
7. Exportar datos del FHIR
8. Eliminación de almacenes de datos

Administrar almacenes de datos con AWS HealthLake

Con él AWS HealthLake, puede crear y administrar almacenes de datos para los recursos del FHIR R4. Al crear un banco de HealthLake datos, un repositorio de datos del FHIR está disponible a través de un punto final de API. RESTful Puede optar por importar (precargar) los datos de salud del FHIR R4 de código abierto de Synthea a su almacén de datos al crearlos. Para obtener más información, consulte Tipos de datos precargados.

> **Importante**
>
> HealthLake admite dos tipos de estrategias de autorización de almacenes de datos del FHIR: AWS SiGv4 o SMART on FHIR. Debe elegir una de las estrategias de autorización antes de crear un banco de datos del HealthLake FHIR. Para obtener más información, consulte Estrategia de autorización del almacén de datos.

Para encontrar las capacidades (comportamientos) relacionadas con el FHIR de un banco de HealthLake datos activo, consulte su declaración de capacidad.

En los siguientes temas se describe cómo utilizar las acciones nativas de HealthLake la nube para crear, describir, enumerar, etiquetar y eliminar almacenes de datos del FHIR mediante AWS CLI, AWS SDKs y. Consola de administración de AWS

Temas

- Creación de un almacén HealthLake de datos
- Obtener las propiedades del almacén de HealthLake datos
- Listado de almacenes HealthLake de datos
- Etiquetado de almacenes HealthLake de datos
- Eliminar un almacén HealthLake de datos

Creación de un almacén HealthLake de datos

Se utiliza `CreateFHIRDatastore` para crear un banco AWS HealthLake de datos que cumpla con la especificación FHIR R4. HealthLake Los bancos de datos se utilizan para importar, administrar, buscar y exportar datos del FHIR. Puede optar por importar (precargar) los datos de salud del FHIR

R4 de código abierto de Synthea a su almacén de datos al crearlos. Para obtener más información, consulte Tipos de datos precargados.

> **Importante**
>
> HealthLake admite dos tipos de estrategias de autorización de almacenes de datos del FHIR: AWS SiGv4 o SMART on FHIR. Debe elegir una de las estrategias de autorización antes de crear un banco de datos del HealthLake FHIR. Para obtener más información, consulte Estrategia de autorización del almacén de datos.

Al crear un banco de HealthLake datos, un repositorio de datos del FHIR está disponible a través de un punto final de la RESTful API. Una vez creado el banco de HealthLake datos, puede solicitar su declaración de capacidad para encontrar todas las capacidades (comportamientos) relacionadas con el FHIR asociadas.

Los siguientes menús proporcionan ejemplos de la AWS CLI AWS SDKs y un procedimiento para la. Consola de administración de AWS Para obtener más información, consulta `CreateFHIRDatastore` en la AWS HealthLake Referencia de la API de .

Para crear un banco HealthLake de datos

Elija un menú en función de sus preferencias de acceso a AWS HealthLake.

AWS CLI y SDKs

CLI

AWS CLI

Ejemplo 1: Crear un almacén de datos compatible con SIGv4 HealthLake

El siguiente `create-fhir-datastore` ejemplo muestra cómo crear un nuevo banco de datos en. AWS HealthLake

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

Salida:

```
{
    "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/datastore/
(Data store ID)/r4/",
    "DatastoreArn": "arn:aws:healthlake:us-east-1:(AWS Account ID):datastore/
(Data store ID)",
    "DatastoreStatus": "CREATING",
    "DatastoreId": "(Data store ID)"
}
```

Ejemplo 2: Crear un almacén de datos SMART en un almacén de datos compatible con FHIR HealthLake

El siguiente `create-fhir-datastore` ejemplo muestra cómo crear un nuevo almacén de datos compatible con SMART on FHIR en. AWS HealthLake

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

Contenido de `identity_provider_configuration.json`:

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

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

Salida:

```
{
    "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/datastore/
(Data store ID)/r4/",
    "DatastoreArn": "arn:aws:healthlake:us-east-1:(AWS Account ID):datastore/
(Data store ID)",
    "DatastoreStatus": "CREATING",
    "DatastoreId": "(Data store ID)"
}
```

- Para obtener más información sobre la API, consulte Create FHIRDatastore in AWS CLI Command Reference.

Python

SDK para Python (Boto3)

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

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


    def create_fhir_datastore(
        self,
        datastore_name: str,
        sse_configuration: dict[str, any] = None,
        identity_provider_configuration: dict[str, any] = None,
    ) -> dict[str, str]:
        """
```

```
        Creates a new HealthLake data store.
        When creating a SMART on FHIR data store, the following parameters are
required:
        - sse_configuration: The server-side encryption configuration for a SMART
on FHIR-enabled data store.
        - identity_provider_configuration: The identity provider configuration
for a SMART on FHIR-enabled data store.

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

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

En el siguiente código se muestra un ejemplo de parámetros para un almacén de datos compatible con SMART on FHIR HealthLake .

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

- Para obtener más información sobre la API, consulta la referencia de la API Create FHIRDatastore in AWS SDK for Python (Boto3).

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

```
    ENDTRY.
```

- Para obtener más información sobre la API, consulte Crear FHIRDatastore en el AWS SDK para la referencia sobre la API ABAP de SAP.

> **Ejemplo de disponibilidad**
>
> ¿No encuentra lo que necesita? Solicita un ejemplo de código mediante el enlace Enviar comentarios en la barra lateral derecha de esta página.

AWS Consola

> **Nota**
>
> El siguiente procedimiento crea un almacén de HealthLake datos con la autorización de AWS SiGv4. La HealthLake consola no admite la creación de un almacén de datos SMART en el FHIR.

Para crear un almacén de HealthLake datos con autorización AWS SiGv4

1. Inicie sesión en la página de creación de un almacén de datos de la HealthLake consola.
2. Selecciona Crear almacén de datos.
3. En la sección de configuración del almacén de datos, en Nombre del almacén de datos, especifique un nombre.
4. (Opcional) En la sección de configuración del almacén de datos, en Precargar datos de muestra, active la casilla de verificación para precargar los datos de Synthea. Los datos de Synthea son un conjunto de datos de muestra de código abierto. Para obtener más información, consulte Tipos de datos precargados de Synthea para HealthLake.
5. En la sección de cifrado del almacén de datos, elija Usar una clave propia de AWS (predeterminada) o Elegir una clave de AWS KMS diferente (avanzada).
6. En la sección Etiquetas (opcional), puede añadir etiquetas a su almacén de datos. Para obtener más información sobre cómo etiquetar el banco de datos, consulteEtiquetado de almacenes HealthLake de datos.

7. Elija Crear almacén de datos.

   El estado del banco de datos está disponible en la página Almacenes de datos.

Obtener las propiedades del almacén de HealthLake datos

Se utiliza `DescribeFHIRDatastore` para obtener las propiedades de un AWS HealthLake banco de datos. Los menús siguientes proporcionan un procedimiento Consola de administración de AWS y ejemplos de código para AWS CLI y AWS SDKs. Para obtener más información, consulta `DescribeFHIRDatastore` en la AWS HealthLake Referencia de la API de .

Para obtener las propiedades de un banco HealthLake de datos

Elija un menú en función de sus preferencias de acceso a AWS HealthLake.

AWS CLI y SDKs

CLI

AWS CLI

Descripción de un almacén de datos de FHIR

El siguiente `describe-fhir-datastore` ejemplo muestra cómo encontrar las propiedades de un almacén de datos en AWS HealthLake.

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

Salida:

```
{
    "DatastoreProperties": {
        "PreloadDataConfig": {
            "PreloadDataType": "SYNTHEA"
        },
        "SseConfiguration": {
            "KmsEncryptionConfig": {
                "CmkType": "CUSTOMER_MANAGED_KMS_KEY",
                "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

```
            }
        },
        "DatastoreName": "Demo",
        "DatastoreArn": "arn:aws:healthlake:us-east-1:<AWS Account ID>:datastore/
<Data store ID>",
        "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/
datastore/<Data store ID>/r4/",
        "DatastoreStatus": "ACTIVE",
        "DatastoreTypeVersion": "R4",
        "CreatedAt": 1603761064.881,
        "DatastoreId": "<Data store ID>",
        "IdentityProviderConfiguration": {
            "AuthorizationStrategy": "AWS_AUTH",
            "FineGrainedAuthorizationEnabled": false
        }
    }
}
```

- Para obtener más información sobre la API, consulte la sección Describe FHIRDatastore in AWS CLI Command Reference.

Python

SDK para Python (Boto3)

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

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


    def describe_fhir_datastore(self, datastore_id: str) -> dict[str, any]:
        """
        Describes a HealthLake data store.
        :param datastore_id: The data store ID.
```

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

- Para obtener más información sobre la API, consulta la descripción FHIRDatastore en la referencia de la API del AWS SDK for Python (Boto3).

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        oo_result = lo_hll->describefhirdatastore(
          iv_datastoreid = iv_datastore_id
        ).
```

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

- Para obtener más información sobre la API, consulta la sección Describe FHIRDatastore en el AWS SDK para ver la referencia sobre la API ABAP de SAP.

> **Ejemplo de disponibilidad**
>
> ¿No encuentra lo que necesita? Solicita un ejemplo de código mediante el enlace Enviar comentarios en la barra lateral derecha de esta página.

AWS Consola

1. Inicie sesión en la página de almacenes de datos de la HealthLake consola.
2. Elija un almacén de datos.

   Se abre la página de detalles del almacén de HealthLake datos y están disponibles todas las propiedades del almacén de datos.

Listado de almacenes HealthLake de datos

Se utiliza `ListFHIRDatastores` para enumerar todos los almacenes de HealthLake datos de la cuenta de un usuario, independientemente del estado del almacén de datos. Los siguientes menús proporcionan un procedimiento Consola de administración de AWS y ejemplos de código para las teclas AWS CLI y AWS SDKs. Para obtener más información, consulta `ListFHIRDatastores` en la AWS HealthLake Referencia de la API de .

Para enumerar todos los almacenes HealthLake de datos

Elija un menú en función de sus preferencias de acceso a AWS HealthLake.

AWS CLI y SDKs

CLI

AWS CLI

Muestra de los almacenes de datos de FHIR

En el siguiente `list-fhir-datastores` ejemplo, se muestra cómo utilizar el comando y cómo los usuarios pueden filtrar los resultados en función del estado del almacén de datos AWS HealthLake.

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

Salida:

```
{
    "DatastorePropertiesList": [
    {
        "PreloadDataConfig": {
            "PreloadDataType": "SYNTHEA"
        },
        "SseConfiguration": {
            "KmsEncryptionConfig": {
                "CmkType": "CUSTOMER_MANAGED_KMS_KEY",
                "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
            }
        },
```

```
        "DatastoreName": "Demo",
        "DatastoreArn": "arn:aws:healthlake:us-east-1:<AWS Account ID>:datastore/
<Data store ID>",
        "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/
datastore/<Data store ID>/r4/",
        "DatastoreStatus": "ACTIVE",
        "DatastoreTypeVersion": "R4",
        "CreatedAt": 1603761064.881,
        "DatastoreId": "<Data store ID>",
        "IdentityProviderConfiguration": {
            "AuthorizationStrategy": "AWS_AUTH",
            "FineGrainedAuthorizationEnabled": false
        }
    }
    ]
}
```

- Para obtener más información sobre la API, consulte la lista FHIRDatastores en la referencia de AWS CLI comandos.

Python

SDK para Python (Boto3)

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

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


    def list_fhir_datastores(self) -> list[dict[str, any]]:
        """
        Lists all HealthLake data stores.
        :return: A list of data store descriptions.
        """
```

```
        try:
            next_token = None
            datastores = []

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

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

- Para obtener más información sobre las API, consulta la lista FHIRDatastores en la referencia de API de AWS SDK for Python (Boto3).

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

- Para obtener más información sobre las API, consulta la lista FHIRDatastores en el AWS SDK para obtener información sobre las API ABAP de SAP.

> **Ejemplo de disponibilidad**
>
> ¿No encuentra lo que necesita? Solicita un ejemplo de código mediante el enlace Enviar comentarios en la barra lateral derecha de esta página.

AWS Consola

- Inicie sesión en la página de almacenes de datos de la HealthLake consola.

Todos los almacenes de HealthLake datos aparecen en la sección Almacenes de datos.

Etiquetado de almacenes HealthLake de datos

Puede asignar metadatos a los almacenes HealthLake de datos en forma de etiquetas. Cada etiqueta es una marca que consta de una clave y un valor definidos por el usuario. Las etiquetas le ayudan a administrar, identificar, organizar, buscar y filtrar los almacenes de datos.

> **Importante**
>
> No almacene información de identificación personal (PII), información médica protegida (PHI) ni otra información confidencial en las etiquetas. Las etiquetas no se han diseñado para usarse con información privada o confidencial.

En los temas siguientes se describe cómo utilizar las operaciones de HealthLake etiquetado mediante las letras Consola de administración de AWS AWS CLI, y AWS SDKs. Para obtener más información, consulte Etiquetar AWS los recursos en la Referencia general de AWS Guía.

Temas

- Etiquetar un almacén de datos HealthLake
- Listar etiquetas para un almacén HealthLake de datos
- Eliminar el etiquetado de un almacén de datos HealthLake

Etiquetar un almacén de datos HealthLake

Se utiliza `TagResource` para etiquetar un banco HealthLake de datos. Los menús siguientes proporcionan un procedimiento Consola de administración de AWS y ejemplos de código para AWS CLI y AWS SDKs. Para obtener más información, consulta `TagResource` en la AWS HealthLake Referencia de la API de .

Para etiquetar un HealthLake banco de datos

Elija un menú en función de sus preferencias de acceso a AWS HealthLake.

AWS CLI y SDKs

CLI

AWS CLI

Agregación de una etiqueta al almacén de datos

En el siguiente ejemplo de `tag-resource`, se muestra cómo agregar una etiqueta a un almacén de datos.

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

Este comando no genera ninguna salida.

.

- Para obtener más información sobre la API, consulte TagResourcela Referencia de AWS CLI comandos.

Python

SDK para Python (Boto3)

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

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


   def tag_resource(self, resource_arn: str, tags: list[dict[str, str]]) ->
None:
```

```
        """
        Tags a HealthLake resource.
        :param resource_arn: The resource ARN.
        :param tags: The tags to add to the resource.
        """
        try:
            self.health_lake_client.tag_resource(ResourceARN=resource_arn,
Tags=tags)
        except ClientError as err:
            logger.exception(
                "Couldn't tag resource %s. Here's why %s",
                resource_arn,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obtener más información sobre la API, consulta TagResourcela AWS Referencia de API de SDK for Python (Boto3).

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    TRY.
        " iv_resource_arn = 'arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        lo_hll->tagresource(
```

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

- Para obtener más información sobre la API, consulte TagResourcela referencia sobre la API ABAP del AWS SDK para SAP.

> **Ejemplo de disponibilidad**
>
> ¿No encuentra lo que necesita? Solicita un ejemplo de código mediante el enlace Enviar comentarios que aparece en la barra lateral derecha de esta página.

AWS Consola

1. Inicie sesión en la página de almacenes de datos de la HealthLake consola.
2. Elija un almacén de datos.

   Se abrirá la página de Detalles del almacén de datos.
3. En la sección Etiquetas, elija Administrar etiquetas.

   Se abrirá la página de Administrar etiquetas.
4. Elija Añadir nueva etiqueta.
5. Ingrese una clave y un valor (opcional).
6. Seleccione Save.

Listar etiquetas para un almacén HealthLake de datos

Se utiliza `ListTagsForResource` para enumerar las etiquetas de un HealthLake banco de datos. Los menús siguientes proporcionan un procedimiento Consola de administración de AWS y ejemplos de código para AWS CLI y AWS SDKs. Para obtener más información, consulta `ListTagsForResource` en la AWS HealthLake Referencia de la API de .

Para enumerar las etiquetas de un HealthLake banco de datos

Elija un menú en función de sus preferencias de acceso a AWS HealthLake.

AWS CLI y SDKs

CLI

AWS CLI

Muestra de las etiquetas para un almacén de datos

En el siguiente ejemplo de `list-tags-for-resource`, se muestran las etiquetas asociadas al almacén de datos especificado:

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

Salida:

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

- Para obtener más información sobre la API, consulte ListTagsForResourcela Referencia de AWS CLI comandos.

Python

SDK para Python (Boto3)

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

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


   def list_tags_for_resource(self, resource_arn: str) -> dict[str, str]:
       """
       Lists the tags for a HealthLake resource.
       :param resource_arn: The resource ARN.
       :return: The tags for the resource.
       """
       try:
           response = self.health_lake_client.list_tags_for_resource(
               ResourceARN=resource_arn
           )
           return response["Tags"]
       except ClientError as err:
           logger.exception(
               "Couldn't list tags for resource %s. Here's why %s",
               resource_arn,
               err.response["Error"]["Message"],
           )
           raise
```

- Para obtener más información sobre la API, consulta ListTagsForResourcela AWS Referencia de API de SDK for Python (Boto3).

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

- Para obtener más información sobre la API, consulte ListTagsForResourcela referencia sobre la API ABAP del AWS SDK para SAP.

Ejemplo de disponibilidad

¿No encuentra lo que necesita? Solicita un ejemplo de código mediante el enlace Enviar comentarios que aparece en la barra lateral derecha de esta página.

AWS Consola

1. Inicie sesión en la página de almacenes de datos de la HealthLake consola.
2. Elija un almacén de datos.

   Se abrirá la página de Detalles del almacén de datos. En la sección Etiquetas, se muestran todas las etiquetas del almacén de datos.

Eliminar el etiquetado de un almacén de datos HealthLake

Se utiliza `UntagResource` para eliminar una etiqueta de un banco de HealthLake datos. Los menús siguientes proporcionan un procedimiento Consola de administración de AWS y ejemplos de código para AWS CLI y AWS SDKs. Para obtener más información, consulta `UntagResource` en la AWS HealthLake Referencia de la API de .

Para quitar la etiqueta de un banco HealthLake de datos

Elija un menú en función de sus preferencias de acceso a AWS HealthLake.

AWS CLI y SDKs

CLI

AWS CLI

Eliminación de etiquetas de un almacén de datos.

En el siguiente ejemplo de `untag-resource`, se muestra cómo eliminar etiquetas de un almacén de datos.

```
aws healthlake untag-resource \
    --resource-arn "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
b91723d65c6fdeb1d26543a49d2ed1fa" \
```

```
    --tag-keys '["key1"]'
```

Este comando no genera ninguna salida.

- Para obtener más información sobre la API, consulte UntagResourcela Referencia de AWS CLI comandos.

Python

SDK para Python (Boto3)

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

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


   def untag_resource(self, resource_arn: str, tag_keys: list[str]) -> None:
       """
       Untags a HealthLake resource.
       :param resource_arn: The resource ARN.
       :param tag_keys: The tag keys to remove from the resource.
       """
       try:
           self.health_lake_client.untag_resource(
               ResourceARN=resource_arn, TagKeys=tag_keys
           )
       except ClientError as err:
           logger.exception(
               "Couldn't untag resource %s. Here's why %s",
               resource_arn,
               err.response["Error"]["Message"],
           )
           raise
```

- Para obtener más información sobre la API, consulta UntagResourcela AWS Referencia de API de SDK for Python (Boto3).

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

- Para obtener más información sobre la API, consulte UntagResourcela referencia sobre la API ABAP del AWS SDK para SAP.

> **Ejemplo de disponibilidad**
>
> ¿No encuentra lo que necesita? Solicita un ejemplo de código mediante el enlace Enviar comentarios que aparece en la barra lateral derecha de esta página.

AWS Consola

1. Inicie sesión en la página de almacenes de datos de la HealthLake consola.
2. Elija un almacén de datos.

   Se abrirá la página de Detalles del almacén de datos.
3. En la sección Etiquetas, elija Administrar etiquetas.

   Se abrirá la página de Administrar etiquetas.
4. Elija Eliminar junto a la etiqueta que desea eliminar.
5. Seleccione Save.

Eliminar un almacén HealthLake de datos

Se utiliza `DeleteFHIRDatastore` para eliminar un almacén HealthLake de datos. Los menús siguientes proporcionan un procedimiento Consola de administración de AWS y ejemplos de código para AWS CLI y AWS SDKs. Para obtener más información, consulta `DeleteFHIRDatastore` en la AWS HealthLake Referencia de la API de .

Para eliminar un almacén HealthLake de datos

Elija un menú en función de sus preferencias de acceso a AWS HealthLake.

AWS CLI y SDKs

CLI

AWS CLI

Eliminación de un almacén de datos de FHIR

El siguiente `delete-fhir-datastore` ejemplo muestra cómo eliminar un almacén de datos y todo su contenido AWS HealthLake.

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

Salida:

```
{
    "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/datastore/
(Data store ID)/r4/",
    "DatastoreArn": "arn:aws:healthlake:us-east-1:(AWS Account ID):datastore/
(Data store ID)",
    "DatastoreStatus": "DELETING",
    "DatastoreId": "(Data store ID)"
}
```

- Para obtener más información sobre la API, consulte Eliminar en la referencia de comandos. FHIRDatastore AWS CLI

Python

SDK para Python (Boto3)

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

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

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


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

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

- Para obtener más información sobre la API, consulta Eliminar FHIRDatastore en la referencia de la API del AWS SDK for Python (Boto3).

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

- Para obtener más información sobre la API, consulta Eliminar FHIRDatastore en el AWS SDK para ver la referencia sobre la API ABAP de SAP.

> **Ejemplo de disponibilidad**
>
> ¿No encuentra lo que necesita? Solicita un ejemplo de código mediante el enlace Enviar comentarios en la barra lateral derecha de esta página.

AWS Consola

1. Inicie sesión en la página de almacenes de datos de la HealthLake consola.
2. Elija un almacén de datos.

   Se abrirá la página de Detalles del almacén de datos.
3. Elija Eliminar.

Se abrirá la página de Eliminar almacén de datos.

4. Para confirmar la eliminación del almacén de datos, escriba el nombre del almacén de datos en el campo de entrada de texto.
5. Elija Eliminar.

Administrar las suscripciones del FHIR en AWS HealthLake

AWS HealthLake es compatible con las suscripciones del FHIR, lo que le permite recibir notificaciones en tiempo real cuando se produzcan cambios específicos en los datos de atención médica. Esta capacidad implementa el modelo de suscripción basado en temas del FHIR R5 Backport, lo que proporciona una escalabilidad y flexibilidad mejoradas en comparación con el modelo de suscripción tradicional del FHIR R4.

Con las suscripciones al FHIR, puede crear aplicaciones de atención médica basadas en eventos que respondan inmediatamente a los cambios en los datos clínicos, lo que permite realizar intervenciones oportunas, automatizar los flujos de trabajo y mejorar la coordinación de la atención.

Temas

- Cómo funcionan las suscripciones al FHIR
- Componentes principales
- Prácticas recomendadas
- El ciclo de vida de la suscripción al FHIR con AWS HealthLake
- Crear una suscripción al FHIR con AWS HealthLake
- Búsqueda de suscripciones al FHIR con AWS HealthLake
- Filtrar notificaciones con AWS HealthLake

Cómo funcionan las suscripciones al FHIR

Las suscripciones del FHIR HealthLake funcionan según un modelo basado en temas en el que:

1. Cree temas para definir eventos: cree temas de suscripción que especifiquen los eventos que pueden activar notificaciones
2. Te suscribes: crea suscripciones a estos temas con criterios de filtrado específicos
3. HealthLake monitorea: el servicio monitorea continuamente los eventos que coincidan con sus criterios
4. Notificaciones entregadas: se producen eventos CWhen coincidentes y HealthLake envía las notificaciones a través del canal que elijas

Componentes principales

Las suscripciones a la FHIR constan de los siguientes componentes.

Temas de suscripción

Los temas de suscripción son la base del sistema de notificaciones y definen:

- Eventos desencadenantes: qué cambios activan las notificaciones (por ejemplo: creación de recursos, actualizaciones o eliminaciones)
- Filtros disponibles: ¿Qué opciones de filtrado están disponibles para los suscriptores
- Contenido de las notificaciones: ¿Qué datos se incluyen en las notificaciones

En la siguiente tabla se enumeran los tipos de temas más comunes.

Tipo de evento	Description (Descripción)	Casos de uso comunes
Creación de recursos	Se activa cuando se crean los recursos	Registro de un nuevo paciente, registro de una nueva observación
Actualizaciones de recursos	Se activa cuando se modifican los recursos	Cambios de estado, actualizaciones clínicas
Eliminación de recursos	Se activa cuando se eliminan los recursos	Seguimiento de auditoría y cumplimiento

Suscripciones

Una suscripción es su solicitud para recibir notificaciones sobre eventos específicos definidos por un tema de suscripción. Cada suscripción incluye:

- Referencia de tema: especifica el tema de suscripción al que se está suscribiendo
- Filtros: criterios para seleccionar qué eventos generan notificaciones
- Configuración del canal: dónde y cómo deben enviarse las notificaciones

- Preferencias de carga útil: ¿qué nivel de detalle se debe incluir en las notificaciones

Canales de notificación

HealthLake admite los siguientes canales de notificación:

Tipo de canal	Casos de uso	
EventBridge	Integraciones empresariales, flujos de trabajo sin servidor, organización multiservicio AWS	
REST Hook	Notificaciones directas de puntos finales, integración de sistemas de terceros	

Cargas útiles de notificación

Elija el tipo de carga útil adecuado en función de sus necesidades:

Tipo de carga útil	Description (Descripción)	Consideraciones de seguridad
Solo identificación	Contiene solo identificadores de recursos	Exposición mínima a la PHI
Recurso completo	Contiene el contenido completo de los recursos con un tamaño máximo de 256 KB. Si el tamaño es superior a 256 KB, volverá a ser solo para ID	Contiene PHI; verifique su manejo seguro

Prácticas recomendadas

Optimización del rendimiento

- Utilice filtros específicos: limite sus criterios para recibir solo las notificaciones esenciales
- Elija los tipos de carga útiles adecuados: utilice cargas útiles solo de identificación siempre que sea posible para obtener un mejor rendimiento
- Implemente receptores eficientes: asegúrese de que los receptores de notificaciones procesen los mensajes rápidamente

Consideraciones de seguridad

- Puntos de conexión seguros: Implemente una autenticación adecuada para los puntos de conexión REST Hook
- Protección de la PHI: tenga cuidado con las cargas útiles de recursos completos, ya que contienen PHI
- Control de acceso: restrinja la creación de suscripciones únicamente a usuarios autorizados

Excelencia operativa

- Establezca las fechas de finalización adecuadas: utilice las fechas de finalización para las suscripciones temporales
- Supervise el estado de las suscripciones: compruebe periódicamente el estado de sus suscripciones
- Implemente la gestión de errores: diseñe sus aplicaciones para gestionar los errores en la entrega de notificaciones

El ciclo de vida de la suscripción al FHIR con AWS HealthLake

Siga estos pasos para comprender el ciclo de vida de la suscripción al FHIR:

1. Creación de un `SubscriptionTopic`
   - Cree una `SubscriptionTopic` con estado `"unknown"`
2. Creación de un `Subscription`
   - Crea un estado `Subscription` con `"requested"`
   - HealthLake valida la configuración `Subscription`
   - `Subscription`debe hacer referencia a un tema ya existente (el tema debe estar en estadounknown,`draft`,`active`).

3. Activación
   - Si es válido, HealthLake actualiza el estado de `Subscription` a `"active"`
   - Al crear una`Subscription`, si el tema indicado estaba en estado`"unknown"`, HealthLake actualiza el estado a `"active"` una vez que la suscripción también esté activa
   - Las suscripciones suelen tardar entre 5 y 10 minutos en crearse correctamente
   - Si `Subscription` no se crea correctamente, el estado cambiará al estado en el `error` que debería realizar una operación de eliminación y, a continuación, volver a intentar crear una suscripción. Puede ver el `"error"` campo en el recurso de suscripción para ver por qué la suscripción no se creó correctamente.
4. Ingerir mientras la suscripción esté `active`
5. ¿Mientras `Subscription` es `active`
   - HealthLake monitorea los eventos que coincidan con sus criterios
   - Las notificaciones se envían al punto final configurado cuando se producen coincidencias
6. Gestión de errores
   - HealthLake intenta volver a intentarlo durante 14 días y, a continuación, deja de volver a intentarlo para esos eventos
7. Desactivación
   - A `Subscription` se puede desactivar de la siguiente manera:

     Establecer una fecha de finalización (desactivación automática)

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

```
        "url": "http://hl7.org/fhir/uv/subscriptions-backport/StructureDefinition/
backport-filter-criteria",
        "valueString": "Encounter?subject=Patient/<patient id>"
      }
    ]
  },
  "channel": {
    "type": "event-bridge",
    "endpoint": "<your event bus arn>",
    "payload": "application/fhir+json",
    "_payload": {
      "extension": [
        {
          "url": "http://hl7.org/fhir/uv/subscriptions-backport/
StructureDefinition/backport-payload-content",
          "valueCode": "id-only"
        }
      ]
    }
  }
}
```

Eliminar el recurso `Subscription`

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

Crear una suscripción al FHIR con AWS HealthLake

La siguiente guía le muestra cómo crear una suscripción al FHIR utilizando. AWS HealthLake

Para crear una suscripción al FHIR

1. Creación de una `SubscriptionTopic`.

   Ejemplo de recurso sobre temas de suscripción:

```
{
  "resourceType": "SubscriptionTopic",
  "url": "http://example.org/FHIR/SubscriptionTopic/encounter-create",
  "version": "1.0.0-fhir.r4b",
  "title": "encounter-create",
```

```
  "status": "unknown",
  "description": "Example topic for new encounters",
  "resourceTrigger": [
    {
      "description": "Encounter Create",
      "resource": "Encounter",
      "supportedInteraction": ["create", "update"]
    }
  ]
}
```

2. Prepara tu terminal de notificaciones (canal personalizado). Los siguientes pasos son necesarios para garantizar que el punto final reciba las notificaciones

   Cuando se utiliza REST Hook

   - Confíe `events.amazonaws.com` en su política de claves de KMS si utiliza el almacén de datos CM_CMK.
   - Si usa un almacén de datos CM_CMK, debe agregar la etiqueta a su clave KMS con el valor de `EventBridgeApiDestinations` `true`
   - HealthLake se utiliza para OAuth autenticar el punto final de REST Hook. Por lo tanto, al crear una suscripción a REST Hook, debes introducir un identificador de cliente, un secreto de cliente y el canal. oAuth-endpoint-url _type.extension [*].

   Ejemplo de política de claves de KMS si se utiliza el almacén de datos CM_CMK:

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

   Cuando se usa EventBridge

   - Confíe `events.amazonaws.com` en su política de claves de KMS si utiliza el almacén de datos CM_CMK.

- Compruebe que su política de EventBridge recursos confía en el principal de servicio. `healthlake.amazonaws.com`
- Cuando utilice CM_CMK y EventBridge sea el punto final, compruebe que está cifrando el EventBridge bus con la misma clave KMS que la clave KMS del almacén de datos.
- Compruebe que su EventBridge autobús tenga al menos una regla que coincida con los eventos generados por. HealthLake

Ejemplo de política de recursos para el bus EventBridge del canal:

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

Ejemplo de patrón de eventos de la EventBridge regla para recibir eventos: HealthLake

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

> **Note**
>
> HealthLake admite 2 fuentes:
>
> - “`healthlake`”: Solo para suscripciones.
> - “`aws.healthlake`”: Para recibir eventos HealthLake de servicio.

Úselo “`healthlake`” como fuente al crear una regla para los autobuses de eventos de suscripciones al FHIR.

3. Crear su `Subscription`.

   Envíe un recurso de suscripción con:

   - Estado: `"requested"`
   - Referencia a la `SubscriptionTopic` identificación elegida
   - Criterios de filtro. Para obtener más información, consulte Filtrar notificaciones para ver los filtros compatibles.
   - Configuración de canal

Ejemplo de cargas útiles de suscripción

Los siguientes ejemplos de código muestran cómo crear cargas útiles de suscripción.

EventBridge

Suscripción con tipo `event-bridge` de canal y `id-only` carga útil.

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

```
    ]
  },
  "channel": {
    "type": "event-bridge",
    "endpoint": "arn:aws:healthlake:eu-west-2:111122223333:event-bus/FhirSubscriptions-
bus",
    "payload": "application/fhir+json",
    "_payload": {
      "extension": [
        {
          "url": "http://hl7.org/fhir/uv/subscriptions-backport/StructureDefinition/
backport-payload-content",
          "valueCode": "id-only"
        }
      ]
    }
  }
}
```

Suscripción con tipo de `event-bridge` terminal y `full-resource` carga útil.

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

```
  "channel": {
    "type": "event-bridge",
    "endpoint": "<your event bus arn>",
    "payload": "application/fhir+json",
    "_payload": {
      "extension": [
        {
          "url": "http://hl7.org/fhir/uv/subscriptions-backport/StructureDefinition/
backport-payload-content",
          "valueCode": "full-resource"
        }
      ]
    }
  }
}
```

Rest Hook

Suscripción con `rest-hook` terminal y tipo `id-only` de carga útil.

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

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

Suscripción con tipo `rest-hook` de canal y `full-resource` carga útil.

```
{
  "resourceType": "Subscription",
  "meta": {
    "profile": [
      "http://hl7.org/fhir/uv/subscriptions-backport/StructureDefinition/backport-
subscription"
    ]
  },
  "status": "requested",
  "end": "2026-07-31T05:38:17.2404292+00:00",
  "reason": "Test subscription for walkthrough",
```

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

Ejemplo de cargas útiles de notificación

Mientras se crea una suscripción, envía un paquete de apretón de manos al canal que hayas configurado para HealthLake comprobar si la suscripción se ha configurado correctamente. La siguiente carga útil es un ejemplo del paquete de apretón de manos.

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

```
  }
}
```

Ejemplo de paquete de notificaciones solo con ID.

```
{
  "version": "0",
  "id": "<your-id>",
  "detail-type": "FHIR Subscription Notification",
  "source": "healthlake",
  "account": "436845984719",
  "time": "2025-09-05T00:18:43Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "subscriptionUrl": "https://healthlake.<AWS_REGION>.amazonaws.com/datastore/
<DS_ID>/r4/Subscription/<SUBSCRIPTION_ID>",
    "notificationBundlePayload": {
      "resourceType": "Bundle",
      "id": "c74ea02a-9c69-4e34-85d6-e72720189574",
      "type": "history",
      "timestamp": "2025-09-05T00:18:43.393688851Z",
      "status": "requested",
      "entry": [
        {
          "fullUrl": "urn:uuid:173135e3-3c80-4b90-a10a-e01a1420fdea",
          "resource": {
            "resourceType": "SubscriptionStatus",
            "id": "173135e3-3c80-4b90-a10a-e01a1420fdea",
            "status": "active",
            "type": "event-notification",
            "eventsSinceSubscriptionStart": "-1",
            "subscription": {
              "reference": "https://healthlake.<AWS_REGION>.amazonaws.com/datastore/
<DS_ID>/r4/Subscription/<SUBSCRIPTION_ID>"
            },
            "topic": "https://healthlake.<AWS_REGION>.amazonaws.com/datastore/<DS_ID>/
r4/SubscriptionTopic/<TOPIC_ID>",
            "notificationEvent": [
              {
                "eventNumber": "0",
                "timestamp": "2025-09-05T00:18:43.393775234Z",
                "focus": "Encounter/c5ae898f-bd96-44dd-a509-87fdbcf23b19",
```

```
                "additionalContext": "Encounter/c5ae898f-bd96-44dd-a509-87fdbcf23b19/
_history/1",
                "id": "8f4e9c1a-2b3d-4e5f-6a7b-8c9d0e1f2a3b"
              }
            ]
          }
        }
      ]
    }
  }
}
```

Ejemplo de paquete de notificaciones con todos los recursos.

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

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

```
  }
}
```

Control de versiones de eventos

HealthLake admite el historial del FHIR de forma predeterminada.

Para saber qué versión del recurso recibió en su paquete de notificaciones:

- recurso completo: dado que los paquetes de recursos completos incluyen todo el recurso, la versión se incluirá en el paquete `entry[*]` para cada recurso del paquete.
- Solo id: los paquetes no incluirán ninguna información sobre los recursos. HealthLakeincluye la versión coincidente e incluida en el paquete a través del `entry[0].notificationEvent[*].additionalContext` campo. Este campo tiene el formato`<ResourceType>/<ResourceId>/_history/<Version Id>`. Para obtener más información, consulta el campo AdditionalContext en el ejemplo de carga útil solo para ID.

Detección de duplicación de eventos

HealthLakeLa función de suscripción al FHIR garantiza al menos una entrega. Esto significa que puedes recibir el mismo evento varias veces, ya sea en el mismo paquete o en un paquete diferente. Para identificar los duplicados, HealthLake proporciona un identificador único para cada evento del paquete de notificaciones. `entry[0].notificationEvent[*].id`

Este identificador es exclusivo de la versión específica del evento que se comparó y se entregó. Por ejemplo, si el mismo encuentro se actualiza dos veces y ambas actualizaciones coinciden con los criterios del filtro, recibirás dos eventos distintos con la misma referencia de encuentro. Tendrán lo mismo`notificationEvent[*].focus`, pero tendrán uno único`notificationEvent[*].id`. Además, estos eventos pueden enviarse en paquetes separados o dentro del mismo paquete de notificaciones.

Búsqueda de suscripciones al FHIR con AWS HealthLake

`Subscription`y `SubscriptionTopic` los recursos también se pueden buscar. HealthLake admite todos los parámetros de búsqueda habituales para la suscripción y SubscriptionTopic los recursos.

Además, ofrecemos capacidades de búsqueda adicionales a través de los siguientes parámetros:

Suscripción

Parámetro de búsqueda	Description (Descripción)	Ejemplo
contacto	Busque en el campo Subscription.contact de la especificación básica del R4	`Subscription?contact=phone`
criterios	Busque en el campo Subscription.criteria de la especificación básica del R4	`Subscription?criteria=[baseUrl]/datastore/[datastoreId]/r4/SubscriptionTopic/[topicId]`
payload	Busque en el campo Subscription.channel.payload de la especificación principal del R4	`Subscription?payload=application/fhir+json`
status	Busque en el campo Subscription.status de la especificación principal del R4	`Subscription?status=error`
type	Busque en el campo Subscription.channel.type de la especificación principal del R4	`Subscription?topic=event-bridge`
url	Busque en el campo Subscription.channel.endpoint de la especificación principal del R4	`Subscription?url=[Subscription.channel.endpoint]`
criterios de filtrado	Busque en el campo de extensión de criterios con la URL "http://hl7. org/fhir/uv/subscriptions-backport/StructureDefinition/backport-filter-criteria» como cadena	`Subscription?filter-criteria=Encounter?`
canal personalizado	Busca en el campo de extensión del tipo de canal personalizado con la URL	`Subscription?custom-channel=[System]|[code]`

Parámetro de búsqueda	Description (Descripción)	Ejemplo
	"http://hl7. org/fhir/uv/subscriptions-backport/StructureDefinition/backport-channel-type» como código Ejemplo de carga útil de suscripción	
tipo de carga útil	Busca en el campo de extensión del tipo de carga útil, donde especificas el formato de la carga útil (uno o varios) `id-only` `full-resource`	`Subscription?payload-type=full-resource`
tema	Busque en el campo Subscription.criteria, donde se agrega el tema	`Subscription?topic=[topicId]`

SubscriptionTopic

Parámetro de búsqueda	Description (Descripción)	Ejemplo
date	Busque en el campo de fecha del SubscriptionTopic recurso	`SubscriptionTopic?date=[SubscriptionTopic.date]`
derived-or-self	Busque en los `derivedFrom` campos `url` o en el SubscriptionTopic recurso	`SubscriptionTopic?derived-or-self=[SubscriptionTopic.url | SubscriptionTopic.derivedFrom]`
identificador	Busque en el campo SubscriptoinTopic .identifier	`SubscriptionTopic?identifier=[Subscr`

Parámetro de búsqueda	Description (Descripción)	Ejemplo
		`iptionTopic.identifier]`
recurso	Busque en el campo SubscriptionTopic .resourceTrigger.resource	`SubscriptionTopic?resource=Encounter`
status	Busque en función del estado del SubscriptionTopic	`SubscriptionTopic?status=unknown`
título	Busque en el título un SubscriptionTopic	`SubscriptionTopic?title=admission`
descripción del disparador	Busque en .resourceTrigger.description SubscriptionTopic	`SubscriptionTopic.trigger-description=resource moving to state 'in-progress'`
url	Busque en la URL el SubscriptionTopic	`SubscriptionTopic?url=[SubscriptionTopic.url]`
versión	Busque en la versión del SubscriptionTopic	`SubscriptionTopic?version=1`

Filtrar notificaciones con AWS HealthLake

Refina tus notificaciones utilizando los parámetros de búsqueda estándar del FHIR `Subscription` según tus criterios.

Filtros compatibles

En la siguiente tabla se muestra una lista de los criterios de filtro compatibles HealthLake con las suscripciones.

Tipos de parámetros de búsqueda	Tipo de datos FHIR	Modifiers (Modificadores)	Prefijos compatibles
Cadena	cadena HumanName Dirección	exacto, contiene lo que falta	
Token	booleano code cadena Codificación CodeableConcept Identificador ContactPoint id	no, texto, falta	
Número	entero decimal PositiveInt INT sin firmar	desaparecido	«eq», «ne», «gt», «lt», «ge», «le», «sa», «eb», «ap»
Date	date dateTime instant Periodo Timing		«eq», «ne», «gt», «lt», «ge», «le», «sa», «eb», «ap»

Tipos de parámetros de búsqueda	Tipo de datos FHIR	Modifiers (Modificadores)	Prefijos compatibles
Cantidad	Cantidad ¿Dinero Range SimpleQuantity		«eq», «ne», «gt», «lt», «ge», «le», «sa», «eb», «ap»
Referencia	Referencia	falta, identificador, tipo	
URI	uri url canonical uid oid	desaparecido	

Ejemplos de filtros compatibles

En la siguiente tabla se muestran ejemplos de los criterios de filtro compatibles HealthLake con las suscripciones:

Finalidad	Criterios del filtro	Description (Descripción)
Observaciones específicas de cada paciente	`Observation?patient=Patient/[id]&status=final`	Reciba notificaciones cuando finalicen las observaciones de un paciente específico
Observaciones específicas de cada paciente	`Patient?birthdate=gt2021`	Reciba notificaciones cuando los pacientes nacidos después de 2021 se registren o actualicen

Finalidad	Criterios del filtro	Description (Descripción)
Observaciones específicas de cada paciente	`Condition?code=http://snomed.info/sct|39065001`	Reciba notificaciones sobre afecciones con códigos SNOMED específicos
Observaciones específicas de cada paciente	`Observation?code=http://loinc.org|8480-6&value-quantity=gt160`	Reciba notificaciones sobre las lecturas de la presión arterial sistólica alta

Importación de datos del FHIR con AWS HealthLake

Tras crear un almacén de HealthLake datos, el siguiente paso es importar los archivos de un bucket de Amazon Simple Storage Service (S3). Puede iniciar un trabajo de importación del FHIR con Consola de administración de AWS AWS CLI, o AWS SDKs. Utilice AWS HealthLake acciones nativas para iniciar, describir y enumerar los trabajos de importación del FHIR.

> **Importante**
>
> HealthLake es compatible con la especificación R4 del FHIR para el intercambio de datos de atención médica. Si es necesario, puede trabajar con un AWS HealthLake socio para convertir sus datos de salud al formato FHIR R4 antes de importarlos.

Al iniciar un trabajo de importación de FHIR, debe especificar una ubicación de entrada del bucket de Amazon S3, una ubicación de salida del bucket de Amazon S3 (para los resultados del procesamiento del trabajo), un rol de IAM que le concede HealthLake acceso a los buckets de Amazon S3 y una clave propiedad o AWS propiedad del cliente. AWS Key Management Service Para obtener más información, consulte Configurar los permisos para los trabajos de importación.

> **Note**
>
> Puede poner en cola los trabajos de importación. Los trabajos de importación asíncrona se procesan de forma FIFO (primero en entrar, primero en salir). Puede poner los trabajos en cola de la misma manera que se inician los trabajos de importación. Si hay alguno en curso, simplemente se pondrá en cola. Puede crear, leer, actualizar o eliminar los recursos del FHIR mientras se está realizando un trabajo de importación.

HealthLake genera un `manifest.json` archivo para cada trabajo de importación del FHIR. El archivo describe los éxitos y los fracasos de un trabajo de importación del FHIR. HealthLake envía el `manifest.json` archivo al bucket de Amazon S3 especificado al iniciar un trabajo de importación de FHIR. Los archivos de registro se organizan en dos carpetas, denominadas `SUCCESS` y`FAILURE`. Utilice el `manifest.json` archivo como primer paso para solucionar un error en un trabajo de importación, ya que proporciona detalles sobre cada archivo.

```
{
```

```
    "inputDataConfig": {
        "s3Uri": "s3://amzn-s3-demo-source-bucket/healthlake-input/invalidInput/"
    },
    "outputDataConfig": {
        "s3Uri": "s3://amzn-s3-demo-logging-bucket/32839038a2f47f17c2fe0f53f0c3a0ba-
FHIR_IMPORT-19dd7bb7bcc8ee12a09bf6d322744a3d/",
        "encryptionKeyID": "arn:aws:kms:us-west-2:123456789012:key/fbbbfee3-20b3-42a5-
a99d-c48c655ed545"
    },
    "successOutput": {
        "successOutputS3Uri": "s3://amzn-s3-demo-logging-
bucket/32839038a2f47f17c2fe0f53f0c3a0ba-FHIR_IMPORT-19dd7bb7bcc8ee12a09bf6d322744a3d/
SUCCESS/"
    },
    "failureOutput": {
        "failureOutputS3Uri": "s3://amzn-s3-demo-logging-
bucket/32839038a2f47f17c2fe0f53f0c3a0ba-FHIR_IMPORT-19dd7bb7bcc8ee12a09bf6d322744a3d/
FAILURE/"
    },
    "numberOfScannedFiles": 1,
    "numberOfFilesImported": 1,
    "sizeOfScannedFilesInMB": 0.023627,
    "sizeOfDataImportedSuccessfullyInMB": 0.011232,
    "numberOfResourcesScanned": 9,
    "numberOfResourcesImportedSuccessfully": 4,
    "numberOfResourcesWithCustomerError": 5,
    "numberOfResourcesWithServerError": 0
}
```


Configurar el nivel de validación para las importaciones

Al iniciar un trabajo de importación del FHIR, si lo desea, puede especificar un valor `ValidationLevel` para aplicarlo a cada recurso. AWS HealthLake actualmente admite los siguientes niveles de validación:

- `strict`: Los recursos se validan según el elemento de perfil del recurso o según la especificación R4 si no hay ningún perfil presente. Este es el nivel de validación predeterminado para AWS HealthLake.
- `structure-only`: Los recursos se validan con el R4 e ignoran los perfiles a los que se hace referencia.

- `minimal`: Los recursos se validan mínimamente, ignorando ciertas reglas de R4. Los recursos que no superen las comprobaciones de estructura requeridas se search/analytics actualizarán para incluir una advertencia para la auditoría.

Al importar mediante el nivel de `minimal` validación, es posible que se generen archivos de registro adicionales en una carpeta denominada`SUCCESS_WITH_SEARCH_VALIDATION_FAILURES`. Los recursos de los archivos de registro de esta carpeta se incorporaron al almacén de datos a pesar de no superar las comprobaciones de validación relacionadas con la búsqueda. Esto implica que algunos aspectos de su recurso del FHIR no eran válidos, según el FHIR, y es posible que los campos con un formato incorrecto no se puedan buscar. Estos recursos tendrán un `extension` anexo que describa dicha falla.

Temas

- Inicio de un trabajo de importación del FHIR
- Conseguir que el FHIR importe propiedades de trabajo
- Listado de trabajos de importación del FHIR

Inicio de un trabajo de importación del FHIR

Se utiliza `StartFHIRImportJob` para iniciar un trabajo de importación del FHIR a un almacén de HealthLake datos. Los siguientes menús proporcionan un procedimiento Consola de administración de AWS y ejemplos de código para AWS CLI y AWS SDKs. Para obtener más información, consulta `StartFHIRImportJob` en la AWS HealthLake Referencia de la API de .

> **Importante**
>
> HealthLake es compatible con la especificación R4 del FHIR para el intercambio de datos de atención médica. Si es necesario, puede trabajar con un AWS HealthLake socio para convertir sus datos de salud al formato FHIR R4 antes de importarlos.

Para iniciar un trabajo de importación de FHIR

Elija un menú según sus preferencias de acceso a. AWS HealthLake

AWS CLI y SDKs

CLI

AWS CLI

Para iniciar un trabajo de importación de FHIR

El siguiente `start-fhir-import-job` ejemplo muestra cómo iniciar un trabajo de importación del FHIR utilizando AWS HealthLake.

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

Salida:

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

- Para obtener información sobre la API, consulte Start FHIRImport Job in AWS CLI Command Reference.

Python

SDK para Python (Boto3)

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

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


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

```
            )
            raise
```

- Para obtener más información sobre la API, consulta la referencia de la API Start FHIRImport Job in AWS SDK for Python (Boto3).

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    TRY.
        " iv_job_name = 'MyImportJob'
        " iv_input_s3_uri = 's3://my-bucket/import/data.ndjson'
        " iv_job_output_s3_uri = 's3://my-bucket/import/output/'
        " iv_kms_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012'
        " iv_data_access_role_arn = 'arn:aws:iam::123456789012:role/
HealthLakeImportRole'
        oo_result = lo_hll->startfhirimportjob(
          iv_jobname = iv_job_name
          io_inputdataconfig = NEW /aws1/cl_hllinputdataconfig( iv_s3uri =
 iv_input_s3_uri )
          io_joboutputdataconfig = NEW /aws1/cl_hlloutputdataconfig(
            io_s3configuration = NEW /aws1/cl_hlls3configuration(
              iv_s3uri = iv_job_output_s3_uri
              iv_kmskeyid = iv_kms_key_id
            )
```

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

- Para obtener más información sobre la API, consulte Start FHIRImport Job in AWS SDK para obtener una referencia sobre la API ABAP de SAP.

> **Ejemplo de disponibilidad**
>
> ¿No encuentra lo que necesita? Solicita un ejemplo de código mediante el enlace Enviar comentarios en la barra lateral derecha de esta página.

AWS Consola

1. Inicie sesión en la página de almacenes de datos de la HealthLake consola.
2. Elija un almacén de datos.
3. Seleccione Importar.

   Se abre la página de importación.

4. En la sección Datos de entrada, introduce la siguiente información:
   - Ubicación de datos de entrada en Amazon S3
5. En la sección Importar archivos de salida, introduzca la siguiente información:
   - Importar la ubicación de los archivos de salida en Amazon S3
   - Cifrado de archivos de salida de importación
6. En la sección Permisos de acceso, elija Usar un rol de servicio de IAM existente y seleccione el rol en el menú del nombre del rol de servicio o elija Crear un rol de IAM.
7. Elija Importar datos.

> **Note**
>
> Durante la importación, seleccione Copiar el ID del trabajo en el banner de la parte superior de la página. Puede utilizar el `JobID`para solicitar la importación de propiedades del trabajo mediante el AWS CLI. Para obtener más información, consulte Conseguir que el FHIR importe propiedades de trabajo.

Conseguir que el FHIR importe propiedades de trabajo

Úselo `DescribeFHIRImportJob` para obtener las propiedades de los trabajos de importación del FHIR. Los siguientes menús proporcionan un procedimiento Consola de administración de AWS y ejemplos de código para las teclas AWS CLI y AWS SDKs. Para obtener más información, consulta `DescribeFHIRImportJob` en la AWS HealthLake Referencia de la API de .

Para obtener las propiedades del trabajo de importación FHIR

Elija un menú en función de sus preferencias de acceso a AWS HealthLake.

AWS CLI y SDKs

CLI

AWS CLI

Para describir un trabajo de importación de FHIR

El siguiente `describe-fhir-import-job` ejemplo muestra cómo aprender las propiedades de un trabajo de importación FHIR utilizando AWS HealthLake.

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

Salida:

```
{
    "ImportJobProperties": {
    "InputDataConfig": {
        "S3Uri": "s3://(Bucket Name)/(Prefix Name)/"
        { "arrayitem2": 2 }
    },
    "DataAccessRoleArn": "arn:aws:iam::(AWS Account ID):role/(Role Name)",
    "JobStatus": "COMPLETED",
    "JobId": "c145fbb27b192af392f8ce6e7838e34f",
    "SubmitTime": 1606272542.161,
    "EndTime": 1606272609.497,
    "DatastoreId": "(Data store ID)"
    }
}
```

- Para obtener más información sobre la API, consulte Describe FHIRImport Job in AWS CLI Command Reference.

Python

SDK para Python (Boto3)

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

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

```
        return cls(health_lake_client)


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

- Para obtener más información sobre la API, consulta Describe FHIRImport Job in AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

- Para obtener más información sobre la API, consulte Describe FHIRImport Job en el AWS SDK para la referencia de la API ABAP de SAP.

> **Ejemplo de disponibilidad**
>
> ¿No encuentra lo que necesita? Solicita un ejemplo de código mediante el enlace Enviar comentarios que aparece en la barra lateral derecha de esta página.

AWS Consola

> **Note**
>
> La información sobre los trabajos de importación del FHIR no está disponible en la HealthLake consola. En su lugar, utilice el AWS CLI comando with `DescribeFHIRImportJob` para solicitar propiedades del trabajo de importación, como `JobStatus`. Para obtener más información, consulte el AWS CLI ejemplo de esta página.

Listado de trabajos de importación del FHIR

Se usa `ListFHIRImportJobs` para enumerar los trabajos de importación del FHIR para un almacén de HealthLake datos activo. Los siguientes menús proporcionan un procedimiento Consola de administración de AWS y ejemplos de código para AWS CLI y AWS SDKs. Para obtener más información, consulta `ListFHIRImportJobs` en la AWS HealthLake Referencia de la API de .

Para enumerar los trabajos de importación del FHIR

Elija un menú en función de sus preferencias de acceso a AWS HealthLake.

AWS CLI y SDKs

CLI

AWS CLI

Para enumerar todos los trabajos de importación de FHIR

En el siguiente ejemplo de `list-fhir-import-jobs`, se muestra cómo utilizar el comando para ver una lista de todos los trabajos de importación asociados a una cuenta.

```
aws healthlake list-fhir-import-jobs \
    --datastore-id (Data store ID) \
```

```
    --submitted-before (DATE like 2024-10-13T19:00:00Z) \
    --submitted-after (DATE like 2020-10-13T19:00:00Z ) \
    --job-name "FHIR-IMPORT" \
    --job-status SUBMITTED  \
    -max-results (Integer between 1 and 500)
```

Salida:

```
{
    "ImportJobPropertiesList": [
        {
            "JobId": "c0fddbf76f238297632d4aebdbfc9ddf",
            "JobStatus": "COMPLETED",
            "SubmitTime": "2024-11-20T10:08:46.813000-05:00",
            "EndTime": "2024-11-20T10:10:09.093000-05:00",
            "DatastoreId": "(Data store ID)",
            "InputDataConfig": {
                "S3Uri": "s3://(Bucket Name)/(Prefix Name)/"
            },
            "JobOutputDataConfig": {
                "S3Configuration": {
                    "S3Uri": "s3://(Bucket Name)/
import/6407b9ae4c2def3cb6f1a46a0c599ec0-FHIR_IMPORT-
c0fddbf76f238297632d4aebdbfc9ddf/",
                    "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/b7f645cb-
e564-4981-8672-9e012d1ff1a0"
                }
            },
            "JobProgressReport": {
                "TotalNumberOfScannedFiles": 1,
                "TotalSizeOfScannedFilesInMB": 0.001798,
                "TotalNumberOfImportedFiles": 1,
                "TotalNumberOfResourcesScanned": 1,
                "TotalNumberOfResourcesImported": 1,
                "TotalNumberOfResourcesWithCustomerError": 0,
                "TotalNumberOfFilesReadWithCustomerError": 0,
                "Throughput": 0.0
            },
            "DataAccessRoleArn": "arn:aws:iam::(AWS Account ID):role/(Role Name)"
        }
    ]
}
```

- Para obtener más información sobre la API, consulte Listar FHIRImport trabajos en la referencia de AWS CLI comandos.

Python

SDK para Python (Boto3)

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

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


   def list_fhir_import_jobs(
       self,
       datastore_id: str,
       job_name: str = None,
       job_status: str = None,
       submitted_before: datetime = None,
       submitted_after: datetime = None,
   ) -> list[dict[str, any]]:
       """
       Lists HealthLake import jobs satisfying the conditions.
       :param datastore_id: The data store ID.
       :param job_name: The import job name.
       :param job_status: The import job status.
       :param submitted_before: The import job submitted before the specified
date.
       :param submitted_after: The import job submitted after the specified
date.
       :return: A list of import jobs.
       """
       try:
           parameters = {"DatastoreId": datastore_id}
           if job_name is not None:
               parameters["JobName"] = job_name
```

```
            if job_status is not None:
                parameters["JobStatus"] = job_status
            if submitted_before is not None:
                parameters["SubmittedBefore"] = submitted_before
            if submitted_after is not None:
                parameters["SubmittedAfter"] = submitted_after
            next_token = None
            jobs = []
            # Loop through paginated results.
            while True:
                if next_token is not None:
                    parameters["NextToken"] = next_token
                response =
self.health_lake_client.list_fhir_import_jobs(**parameters)
                jobs.extend(response["ImportJobPropertiesList"])
                if "NextToken" in response:
                    next_token = response["NextToken"]
                else:
                    break
            return jobs
        except ClientError as err:
            logger.exception(
                "Couldn't list import jobs. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Para obtener más información sobre la API, consulta la sección Lista de FHIRImport trabajos en la referencia de la API del AWS SDK for Python (Boto3).

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

- Para obtener más información sobre la API, consulte Listar FHIRImport trabajos en el AWS SDK para obtener una referencia sobre la API ABAP de SAP.

Ejemplo de disponibilidad

¿No encuentra lo que necesita? Solicita un ejemplo de código mediante el enlace Enviar comentarios en la barra lateral derecha de esta página.

AWS Consola

Note

La información sobre los trabajos de importación del FHIR no está disponible en la HealthLake consola. En su lugar, utilice AWS CLI with `ListFHIRImportJobs` para enumerar todos los trabajos de importación del FHIR. Para obtener más información, consulte el AWS CLI ejemplo de esta página.

Administrar los recursos del FHIR en AWS HealthLake

Utilice las interacciones de la RESTful API R4 del FHIR para gestionar los recursos del FHIR en un almacén de datos. HealthLake En las siguientes secciones se describen todas las interacciones de la RESTful API R4 del FHIR HealthLake compatibles y disponibles para la gestión de recursos del FHIR. Para obtener información sobre las capacidades del almacén de HealthLake datos y las partes de la especificación del FHIR compatibles, consulte. Declaración de capacidad del FHIR R4 para AWS HealthLake

> **Note**
>
> Las interacciones del FHIR que se enumeran en este capítulo están diseñadas de conformidad con el estándar R4 del HL7 FHIR para el intercambio de datos de atención médica. Como son representaciones de los servicios del HL7 FHIR, no se ofrecen a través de y. AWS CLI AWS SDKs Para obtener más información, consulte la RESTful API en la documentación de la API R4 RESTful del FHIR.

En la siguiente tabla se enumeran las interacciones del FHIR R4 compatibles con. AWS HealthLake Para obtener información sobre los tipos de recursos del FHIR compatibles HealthLake, consulte. Tipos de recurso

Las interacciones FHIR R4 son compatibles con AWS HealthLake

Interacción	Description (Descripción)
Interacciones de todo el sistema	
`capabilities`	Obtenga una declaración de capacidad del sistema. Consulte Declaración de capacidad del FHIR R4 para AWS HealthLake.
`batch`	Actualice, cree o elimine un conjunto de recursos en una sola interacción. Consulte Agrupación de recursos del FHIR.
Interacciones a nivel de tipo	
`create`	Cree un nuevo recurso con un ID asignado por el servidor. Consulte Creación de un recurso FHIR.

Interacción	Description (Descripción)
`search`	Busque un tipo de recurso en función de algunos criterios de filtrado. Consulte Buscando recursos del FHIR.
`history`	Recupera el historial de cambios de un tipo de recurso concreto. Consulte Lectura del historial de recursos del FHIR.
Interacciones a nivel de instancia	
`read`	Lee el estado actual de un recurso. Consulte Lectura de un recurso del FHIR.
`history`	Lea el historial de cambios de un recurso en particular. Consulte Lectura del historial de recursos del FHIR.
`vread`	Lea el estado de una versión específica del recurso. Consulte Lectura del historial de recursos del FHIR específico de cada versión.
`update`	Actualiza un recurso por su ID (o créalo si es nuevo). Consulte Actualización de un recurso del FHIR.
`delete`	Elimina un recurso. Consulte Eliminar un recurso del FHIR.

Temas

- Creación de un recurso FHIR
- Lectura de un recurso del FHIR
- Lectura del historial de recursos del FHIR
- Actualización de un recurso del FHIR
- Modificación de recursos con la operación PATCH
- Agrupación de recursos del FHIR
- Eliminar un recurso del FHIR
- Idempotencia y concurrencia

Creación de un recurso FHIR

La `create` interacción del FHIR crea un nuevo recurso del FHIR en un HealthLake almacén de datos. Para obtener información adicional, consulte la documentación de la `create`API RESTful R4 del FHIR.

Para crear un recurso del FHIR

1. HealthLake `region`Recopila y `datastoreId` valora. Para obtener más información, consulte Obtención de propiedades de los almacenes de datos.
2. Determine el tipo de FHIR `Resource` que desea crear. Para obtener más información, consulte Tipos de recurso.
3. Cree una URL para la solicitud utilizando los valores recopilados para HealthLake `region` y`datastoreId`. Incluya también el `Resource` tipo FHIR que desee crear. Para ver la ruta URL completa en el siguiente ejemplo, desplázate sobre el botón Copiar.

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

4. Cree un cuerpo JSON para la solicitud y especifique los datos del FHIR del nuevo recurso. Para este procedimiento, utilizamos un `Patient` recurso del FHIR, así que guarde el archivo como. `create-patient.json`

```
{
    "resourceType": "Patient",
    "identifier": [
        {
            "system": "urn:oid:1.2.36.146.595.217.0.1",
            "value": "12345"
        }
    ],
    "name": [
        {
            "family": "Silva",
            "given": [
                "Ana",
                "Carolina"
            ]
        }
    ],
```

```
    "gender": "female",
    "birthDate": "1992-02-10"
}
```

5. Envíe la solicitud . La `create` interacción del FHIR utiliza una `POST` solicitud con la versión 4 de la AWS firma o SMART previa autorización del FHIR. En los siguientes ejemplos se crea un `Patient` recurso del FHIR HealthLake utilizando curl o la consola. HealthLake Para ver un ejemplo completo, desplázate sobre el botón Copiar.

   SigV4

   Autorización SigV4

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

   SMART on FHIR

   Ejemplo de autorización SMART on FHIR para el tipo de `IdentityProviderConfiguration`datos.

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

```
\"capabilities\":[\"launch-ehr\",\"sso-openid-connect\",\"client-public\",
\"permission-v2\"]}"
}
```

La persona que llama puede asignar permisos en la lambda de autorización. Para obtener más información, consulte OAuth osciloscopios 2.0.

AWS Console

> **Note**
>
> La HealthLake consola solo admite la autorización AWS SiGv4.

1. Inicie sesión en la página Ejecutar consultas de la HealthLake consola.
2. En la sección de configuración de consultas, realice las siguientes selecciones.
   - ID de almacén de datos: elija un ID de almacén de datos para generar una cadena de consulta.
   - Tipo de consulta: elija`Create`.
   - Tipo de recurso: elija el tipo de recurso FHIR que desee crear.
   - Cuerpo de la solicitud: cree un cuerpo JSON para la solicitud y especifique los datos del FHIR del nuevo recurso.
3. Elija Ejecutar consulta.

Configurar el nivel de validación para la creación de recursos

Al crear un recurso FHIR, puede especificar opcionalmente un encabezado `x-amzn-healthlake-fhir-validation-level` HTTP para configurar un nivel de validación para el recurso. AWS HealthLake actualmente admite los siguientes niveles de validación:

- `strict`: Los recursos se validan según el elemento de perfil del recurso o según la especificación R4 si no hay ningún perfil presente. Este es el nivel de validación predeterminado para AWS HealthLake.

- `structure-only`: Los recursos se validan con el R4 e ignoran los perfiles a los que se hace referencia.
- `minimal`: Los recursos se validan mínimamente, ignorando ciertas reglas de R4. Los recursos que no superen las comprobaciones de estructura requeridas se search/analytics actualizarán para incluir una advertencia para la auditoría.

Los recursos creados con el nivel de validación mínimo pueden incorporarse a un almacén de datos a pesar de no pasar la validación necesaria para la indexación de las búsquedas. En este caso, los recursos se actualizarán para incluir una extensión específica de Healthlake para documentar dichos errores:

```
{
    "url": "http://healthlake.amazonaws.com/fhir/StructureDefinition/validation-issue",
    "valueString": "{\"resourceType\":\"OperationOutcome\",\"issue\":[{\"severity\":
\"error\",\"code\":\"processing\",\"details\":{\"text\":\"FHIR resource in payload
 failed FHIR validation rules.\"},\"diagnostics\":\"FHIR resource in payload failed
 FHIR validation rules.\"}]}"
}
```

Además, se incluirá el siguiente encabezado de respuesta HTTP con el valor «true»:

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

> **Note**
>
> Es posible que los datos ingresados que tengan un formato incorrecto según la especificación R4 no se puedan buscar como se esperaba si se presentan estos errores.

Lectura de un recurso del FHIR

La `read` interacción del FHIR lee el estado actual de un recurso en un almacén de HealthLake datos. Para obtener información adicional, consulte la documentación de `read`la API R4 RESTful del FHIR.

Para leer un recurso del FHIR

1. Colecciona HealthLake `region` y `datastoreId` valora. Para obtener más información, consulte Obtención de propiedades de los almacenes de datos.

2. Determine el tipo de FHIR `Resource` para leer y recopilar el `id` valor asociado. Para obtener más información, consulte Tipos de recurso.
3. Cree una URL para la solicitud utilizando los valores recopilados para HealthLake `region` y`datastoreId`. Incluya también el `Resource` tipo FHIR y su correspondiente`id`. Para ver la ruta URL completa en el siguiente ejemplo, desplázate sobre el botón Copiar.

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

4. Envíe la solicitud . La `read` interacción del FHIR utiliza una `GET` solicitud con la versión 4 de la AWS firma o SMART con la autorización del FHIR. En el siguiente `curl` ejemplo, se lee el estado actual de un recurso del FHIR `Patient` en. HealthLake Para ver el ejemplo completo, desplázate sobre el botón Copiar.

   SigV4

   Autorización SigV4

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

   SMART on FHIR

   Ejemplo de autorización SMART on FHIR para el tipo de `IdentityProviderConfiguration`datos.

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

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

La persona que llama puede asignar permisos en la lambda de autorización. Para obtener más información, consulte OAuth osciloscopios 2.0.

AWS Console

1. Inicie sesión en la página Ejecutar consultas de la HealthLake consola.
2. En la sección de configuración de consultas, realice las siguientes selecciones.
   - ID de almacén de datos: elija un ID de almacén de datos para generar una cadena de consulta.
   - Tipo de consulta: elija`Read`.
   - Tipo de recurso: elija el tipo de recurso FHIR que desee leer.
   - ID de recurso: introduzca el ID de recurso del FHIR.
3. Elija Ejecutar consulta.

Lectura del historial de recursos del FHIR

La `history` interacción del FHIR recupera el historial de un recurso del FHIR en particular en un almacén de datos. HealthLake Con esta interacción, puede determinar cómo ha cambiado el contenido de un recurso del FHIR a lo largo del tiempo. También resulta útil, en coordinación con los registros de auditoría, para ver el estado de un recurso antes y después de la modificación. El FHIR `create` interactúa `update` y da `delete` como resultado una versión histórica del recurso que se va a guardar. Para obtener información adicional, consulte la documentación de `history`la API R4 RESTful del FHIR.

> **Note**
>
> Puede optar por no utilizar tipos de recursos `history` específicos del FHIR. Para excluirse, cree un caso utilizando AWS Support Center Console. Para crear tu caso, inicia sesión en tu cuenta Cuenta de AWS y selecciona Crear caso.

Para leer el historial de recursos del FHIR

1. HealthLake `region`Recopila y `datastoreId` valora. Para obtener más información, consulte Obtención de propiedades de los almacenes de datos.
2. Determine el tipo de FHIR `Resource` para leer y recopilar el `id` valor asociado. Para obtener más información, consulte Tipos de recurso.
3. Cree una URL para la solicitud utilizando los valores recopilados para HealthLake `region` y`datastoreId`. Incluya también el `Resource` tipo de FHIR, sus parámetros de búsqueda asociados `id` y opcionales. Para ver la ruta URL completa en el siguiente ejemplo, desplázate sobre el botón Copiar.

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

HealthLake parámetros de búsqueda compatibles para la interacción FHIR **`history`**

Parámetro	Description (Descripción)
`_count : integer`	El número máximo de resultados de búsqueda en una página. El servidor devolverá el número solicitado o el número máximo de resultados de búsqueda permitido de forma predeterminada para el almacén de datos, lo que sea inferior.
`_since : instant`	Incluya únicamente las versiones de recursos que se hayan creado en un instante determinado o con posterioridad a él.

Parámetro	Description (Descripción)
`_at : date(Time)`	Incluya únicamente las versiones de recursos que estuvieran actualizadas en algún momento durante el período de tiempo especificado en el valor de fecha y hora. Para obtener más información, consulte la documentación *date*de la RESTful API del HL7 FHIR.

4. Envíe la solicitud . La `history` interacción del FHIR utiliza una `GET` solicitud con la versión 4 de la AWS firma o SMART previa autorización del FHIR. En el siguiente `curl` ejemplo, se utiliza el parámetro de `_count` búsqueda para devolver 100 resultados de búsqueda históricos por página para un recurso del FHIR `Patient` en. HealthLake Para ver el ejemplo completo, desplázate sobre el botón Copiar.

   SigV4

   Autorización SigV4

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

   SMART on FHIR

   Ejemplo de autorización SMART on FHIR para el tipo de `IdentityProviderConfiguration`datos.

   ```
   {
       "AuthorizationStrategy": "SMART_ON_FHIR",
       "FineGrainedAuthorizationEnabled": true,
       "IdpLambdaArn": "arn:aws:lambda:your-region:your-account-id:function:your-
   lambda-name",
       "Metadata": "{\"issuer\":\"https://ehr.example.com\", \"jwks_uri\":
   \"https://ehr.example.com/.well-known/jwks.json\",\"authorization_endpoint
   ```

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

La persona que llama puede asignar permisos en la lambda de autorización. Para obtener más información, consulte OAuth osciloscopios 2.0.

El contenido devuelto de una `history` interacción está contenido en un recurso FHIR`Bundle`, con el tipo establecido en. `history` Contiene el historial de versiones especificado, ordenado en último lugar por las versiones más antiguas, e incluye los recursos eliminados. Para obtener más información, consulte `Resource Bundle`la documentación del FHIR R4.

Lectura del historial de recursos del FHIR específico de cada versión

La `vread` interacción del FHIR realiza una lectura específica de la versión de un recurso en un almacén de datos. HealthLake Con esta interacción, puede ver el contenido de un recurso del FHIR tal como estaba en un momento determinado del pasado.

> **Note**
>
> Si utiliza la `history` interacción del FHIR sin `vread` ella, HealthLake siempre devuelve la última versión de los metadatos del recurso.

HealthLake declara que admite el control de versiones `CapabilityStatement.rest.resource.versioning`para cada recurso compatible. Todos los almacenes de HealthLake datos incluyen `Resource.meta.versionId` (`vid`) en todos los recursos.

Cuando `history` la interacción FHIR está habilitada (de forma predeterminada para los almacenes de datos creados después del 25 de octubre de 2024 o cuando se solicita para los almacenes de datos más antiguos), la `Bundle` respuesta los incluye `vid` como parte del elemento. `location` En el siguiente ejemplo, se `vid` muestra como un número. 1 Para ver el ejemplo completo, consulta Example Bundle/Bundle-Response (JSON).

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

Para leer el historial de recursos del FHIR específico de cada versión

1. HealthLake `region`Recopila y valora. `datastoreId` Para obtener más información, consulte Obtención de propiedades de los almacenes de datos.
2. Determine el `Resource` tipo de FHIR que se va a leer y recopilar `vid` los valores `id` y valores asociados. Para obtener más información, consulte Tipos de recurso.
3. Cree una URL para la solicitud utilizando los valores recopilados para el HealthLake FHIR. Para ver la ruta URL completa en el siguiente ejemplo, desplázate sobre el botón Copiar.

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

4. Envíe la solicitud . La `history` interacción del FHIR utiliza una `GET` solicitud con la versión 4 de la AWS firma o SMART con la autorización del FHIR. La siguiente `vread` interacción devuelve una sola instancia con el contenido especificado para el `Patient` recurso de la FHIR para la versión de los metadatos del recurso especificada por la. `vid` Para ver la ruta URL completa en el siguiente ejemplo, desplázate sobre el botón Copiar.

   SigV4

   Autorización SigV4

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

```
  --header 'Accept: application/json'
```

SMART on FHIR

Ejemplo de autorización SMART on FHIR para el tipo de `IdentityProviderConfiguration`datos.

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

La persona que llama puede asignar permisos en la lambda de autorización. Para obtener más información, consulte OAuth osciloscopios 2.0.

Actualización de un recurso del FHIR

La `update` interacción del FHIR crea una nueva versión actual para un recurso existente o crea una versión inicial si aún no existe ningún recurso para ese recurso. `id` Para obtener información adicional, consulte la documentación de `update`la API R4 RESTful del FHIR.

Para actualizar un recurso del FHIR

1. HealthLake `region`Recopila y `datastoreId` valora. Para obtener más información, consulte Obtención de propiedades de los almacenes de datos.

2. Determine el tipo de `Resource` FHIR que desea actualizar y recopile el `id` valor asociado. Para obtener más información, consulte Tipos de recurso.

3. Cree una URL para la solicitud utilizando los valores recopilados para HealthLake `region` y`datastoreId`. Incluya también el `Resource` tipo FHIR y su correspondiente`id`. Para ver la ruta URL completa en el siguiente ejemplo, desplázate sobre el botón Copiar.

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

4. Cree un `JSON` cuerpo para la solicitud, especificando las actualizaciones de datos del FHIR que se van a realizar. Para este procedimiento, guarde el archivo como`update-patient.json`.

```
{
    "id": "2de04858-ba65-44c1-8af1-f2fe69a977d9",
    "resourceType": "Patient",
    "active": true,
    "name": [
        {
            "use": "official",
            "family": "Doe",
            "given": [
                "Jane"
            ]
        },
        {
            "use": "usual",
            "given": [
                "Jane"
            ]
        }
    ],
    "gender": "female",
    "birthDate": "1985-12-31"
}
```

5. Envíe la solicitud . La `update` interacción con el FHIR utiliza una `PUT` solicitud con la versión 4 de AWS Signature o SMART previa autorización del FHIR. En el siguiente `curl` ejemplo, se actualiza un `Patient` recurso en. HealthLake Para ver el ejemplo completo, desplázate sobre el botón Copiar.

SigV4

Autorización SigV4

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

Su solicitud devolverá un código de estado `200` HTTP si se actualiza un recurso existente o un código de estado `201` HTTP si se crea un recurso nuevo.

SMART on FHIR

Ejemplo de autorización SMART on FHIR para el tipo `IdentityProviderConfiguration`de datos.

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

La persona que llama puede asignar permisos en la lambda de autorización. Para obtener más información, consulte OAuth osciloscopios 2.0.

AWS Console

1. Inicie sesión en la página Ejecutar consultas de la HealthLake consola.
2. En la sección de configuración de consultas, realice las siguientes selecciones.
   - ID de almacén de datos: elija un ID de almacén de datos para generar una cadena de consulta.
   - Tipo de consulta: elija`Update (PUT)`.
   - Tipo de recurso: elija el tipo de recurso FHIR que desee actualizar o crear.
   - Cuerpo de la solicitud: cree un cuerpo JSON para la solicitud y especifique los datos del FHIR con los que se va a actualizar el recurso.
3. Elija Ejecutar consulta.

Actualización de los recursos del FHIR en función de las condiciones

La actualización condicional permite actualizar un recurso existente en función de algunos criterios de búsqueda de identificación, en lugar de hacerlo mediante el FHIR `id` lógico. Cuando el servidor procesa la actualización, realiza una búsqueda utilizando sus capacidades de búsqueda estándar para el tipo de recurso, con el objetivo de resolver una única lógica `id` para la solicitud.

La acción que realice el servidor depende del número de coincidencias que encuentre:

- No hay coincidencias, no **`id`** se proporciona en el cuerpo de la solicitud: el servidor crea el recurso FHIR.
- No hay coincidencias, **`id`** se ha proporcionado y el recurso aún no existe con el **`id`**: el servidor trata la interacción como una interacción entre actualizar y crear.
- No hay coincidencias, **`id`** se ha proporcionado y ya existe: el servidor rechaza la actualización con un `409 Conflict` error.
- Una coincidencia, no **`id`** se proporcionó ningún recurso O (recurso **`id`** proporcionado y coincide con el recurso encontrado): el servidor realiza la actualización con el recurso coincidente como se indica anteriormente, donde, si el recurso se actualizó, el servidor DEVOLVERÁ un`200 OK`.

- Una coincidencia, recurso **`id`** proporcionado pero no coincide con el recurso encontrado: el servidor devuelve un `409 Conflict` error que indica que la especificación del identificador de cliente era un problema, preferiblemente con un `OperationOutcome`
- Varias coincidencias: el servidor devuelve un `412 Precondition Failed` error que indica que los criterios del cliente no fueron lo suficientemente selectivos, preferiblemente con un OperationOutcome

En el siguiente ejemplo, se actualiza un `Patient` recurso cuyo nombre es peter, su fecha de nacimiento es el 1 de enero de 2000 y su número de teléfono es 1234567890.

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

Configurar el nivel de validación para las actualizaciones de recursos

Al actualizar un recurso del FHIR, puede especificar opcionalmente un encabezado `x-amzn-healthlake-fhir-validation-level` HTTP para configurar un nivel de validación para el recurso. AWS HealthLake actualmente admite los siguientes niveles de validación:

- `strict`: Los recursos se validan según el elemento de perfil del recurso o según la especificación R4 si no hay ningún perfil presente. Este es el nivel de validación predeterminado para AWS HealthLake.
- `structure-only`: Los recursos se validan con el R4 e ignoran los perfiles a los que se hace referencia.
- `minimal`: Los recursos se validan mínimamente, ignorando ciertas reglas de R4. Los recursos que no superen las comprobaciones de estructura requeridas se search/analytics actualizarán para incluir una advertencia para la auditoría.

Los recursos actualizados con el nivel de validación mínimo pueden incorporarse a un almacén de datos a pesar de no pasar la validación necesaria para la indexación de las búsquedas. En este caso, los recursos se actualizarán para incluir una extensión específica de Healthlake para documentar dichos errores:

```
{
    "url": "http://healthlake.amazonaws.com/fhir/StructureDefinition/validation-issue",
    "valueString": "{\"resourceType\":\"OperationOutcome\",\"issue\":[{\"severity\":
\"error\",\"code\":\"processing\",\"details\":{\"text\":\"FHIR resource in payload
```

```
 failed FHIR validation rules.\"},\"diagnostics\":\"FHIR resource in payload failed
 FHIR validation rules.\"}]}"
}
```

Además, se incluirá el siguiente encabezado de respuesta HTTP con el valor «true»:

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

> **Note**
>
> Tenga en cuenta que es posible que los datos ingresados que tengan un formato incorrecto según la especificación R4 no se puedan buscar como se esperaba si se presentan estos errores.

Modificación de recursos con la operación PATCH

AWS HealthLake admite la operación PATCH para los recursos del FHIR, lo que le permite modificar los recursos seleccionando elementos específicos para añadirlos, sustituirlos o eliminarlos sin necesidad de actualizar todo el recurso. Esta operación resulta especialmente útil cuando se necesita:

- Realice actualizaciones específicas en recursos de gran tamaño
- Reduzca el uso del ancho de banda de
- Realice modificaciones atómicas en elementos de recursos específicos
- Minimice el riesgo de sobrescribir los cambios simultáneos
- Actualice los recursos como parte de los flujos de trabajo por lotes y transacciones

Formatos PATCH compatibles

AWS HealthLake admite dos formatos PATCH estándar:

Parche JSON (RFC 6902)

Utiliza la sintaxis JSON Pointer para seleccionar los elementos según su posición en la estructura de recursos.

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

FHIRPath Parche (especificación FHIR R4)

Utiliza FHIRPath expresiones para seleccionar los elementos según su contenido y sus relaciones, lo que proporciona un enfoque de aplicación de parches nativo del FHIR.

Tipo de contenido: `application/fhir+json`

De uso

Operaciones directas de PATCH

La operación PATCH se puede invocar directamente en los recursos del FHIR mediante el método HTTP PATCH:

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

PATCH en paquetes

Las operaciones PATCH se pueden incluir como entradas en los paquetes del FHIR `batch` o bien`transaction`, lo que permite combinar las operaciones de parche con otras interacciones del FHIR (crear, leer, actualizar, eliminar) en una sola solicitud.

- Paquetes de transacciones: todas las entradas se realizan correctamente o fallan de forma atómica
- Paquetes de lotes: cada entrada se procesa de forma independiente

Formato de parche JSON

Operaciones admitidas

Operación	Description (Descripción)
`add`	Agregue un nuevo valor al recurso
`remove`	Eliminar un valor del recurso
`replace`	Reemplace un valor existente en el recurso
`move`	Elimine un valor de una ubicación y agréguelo a otra

Operación	Description (Descripción)
`copy`	Copie un valor de una ubicación a otra
`test`	Compruebe que un valor en la ubicación de destino es igual a un valor especificado

Sintaxis de ruta

El parche JSON utiliza la sintaxis JSON Pointer (RFC 6901):

Ejemplo de ruta	Description (Descripción)
`/name/0/family`	Elemento familiar del nombre
`/telecom/-`	Añadir a la matriz de telecomunicaciones
`/active`	Elemento activo a nivel de raíz
`/address/0/line/1`	Segunda línea de la primera dirección

Ejemplos

Solicitud directa de parches JSON con múltiples operaciones

```
PATCH [base]/Patient/example
Content-Type: application/json-patch+json
If-Match: W/"1"

[
  {
    "op": "replace",
    "path": "/name/0/family",
    "value": "Smith"
  },
  {
    "op": "add",
    "path": "/telecom/-",
```

```
    "value": {
      "system": "phone",
      "value": "555-555-5555",
      "use": "home"
    }
  },
  {
    "op": "remove",
    "path": "/address/0"
  },
  {
    "op": "move",
    "from": "/name/0/family",
    "path": "/name/1/family"
  },
  {
    "op": "test",
    "path": "/gender",
    "value": "male"
  },
  {
    "op": "copy",
    "from": "/name/0",
    "path": "/name/1"
  }
]
```

Solicitud directa de parches JSON con una sola operación

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

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

Parche JSON en paquete

Utilice un recurso binario que contenga la carga útil del parche JSON codificada en base64:

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

FHIRPath Formato de parche

Operaciones admitidas

Operación	Description (Descripción)
`add`	Añadir un elemento nuevo a un recurso
`insert`	Inserte un elemento en una posición específica de una lista
`delete`	Eliminar un elemento de un recurso
`replace`	Reemplace el valor de un elemento existente
`move`	Reordenar los elementos de una lista

Sintaxis de ruta

FHIRPath El parche usa FHIRPath expresiones que admiten:

- Acceso basado en índices: `Patient.name[0]`
- Filtrar con: `where()` `Patient.name.where(use = 'official')`

- Lógica booleana: `Patient.telecom.where(system = 'phone' and use = 'work')`
- Funciones de subconjuntos:, `first()` `last()`
- Comprobaciones de existencia:`exists()`, `count()`
- Navegación polimórfica: `Observation.value`

Ejemplos

Solicitud de parche directa FHIRPath

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

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

FHIRPath Parche en paquete

Utilice un recurso de parámetros como recurso de entrada con`method: PATCH`:

```
{
  "resourceType": "Bundle",
  "type": "transaction",
  "entry": [{
    "resource": {
      "resourceType": "Parameters",
      "parameter": [{
        "name": "operation",
        "part": [
          { "name": "type", "valueCode": "add" },
```

```
          { "name": "path", "valueString": "Patient" },
          { "name": "name", "valueString": "birthDate" },
          { "name": "value", "valueDate": "1990-01-01" }
        ]
      }]
    },
    "request": {
      "method": "PATCH",
      "url": "Patient/123"
    }
  }]
}
```

Encabezados de solicitud

Encabezado	Obligatorio	Descripción
`Content-Type`	Sí	`application/json-patch+json` para JSON Patch o `application/fhir+json` para FHIRPath Patch
`If-Match`	No	Actualización condicional específica de la versión mediante ETag

Respuesta de ejemplo

La operación devuelve el recurso actualizado con la información de la nueva versión:

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

{
  "resourceType": "Patient",
  "id": "example",
  "active": true,
  "name": [
    {
      "family": "Smith",
```

```
      "given": ["John"]
    }
  ],
  "telecom": [
    {
      "system": "phone",
      "value": "555-555-5555",
      "use": "home"
    }
  ],
  "meta": {
    "versionId": "2",
    "lastUpdated": "2025-05-05T10:10:10Z"
  }
}
```

Comportamiento

La operación PATCH:

- Valida la sintaxis del parche según la especificación correspondiente (RFC 6902 para JSON Patch, FHIR R4 para Patch) FHIRPath
- Aplica las operaciones de forma atómica: todas las operaciones se realizan correctamente o todas fallan
- Actualiza el ID de versión del recurso y crea una nueva entrada en el historial
- Conserva el recurso original en el historial antes de aplicar los cambios
- Valida las restricciones de recursos del FHIR después de aplicar los parches
- Admite actualizaciones condicionales mediante el encabezado If-Match con ETag

Gestión de errores

La operación gestiona las siguientes condiciones de error:

- 400 Solicitud errónea: sintaxis de parche no válida (solicitud no conforme o documento de parche mal formado)
- 404 No encontrado: no se encontró el recurso (el ID especificado no existe)
- 409 Conflicto: conflicto de versiones (se proporciona un identificador de versión no actualizado o actualizaciones simultáneas)

- 422 Entidad inprocesable: las operaciones de parche no se pueden aplicar a los elementos de recursos especificados

Resumen de capacidades

Funcionalidad	Parche JSON	FHIRPath Parche
Tipo de contenido	`application/json-patch+json`	`application/fhir+json`
Formato de ruta	Puntero JSON (RFC 6901)	FHIRPath expresiones
API DIRECT PATCH	Soportado	compatible
Batch de paquetes	Compatible (mediante binario)	Compatible (mediante parámetros)
Transacción de paquete	Compatible (mediante binario)	Compatible (mediante parámetros)
Operaciones	añadir, eliminar, reemplazar, mover, copiar, probar	añadir, insertar, eliminar, reemplazar, mover

Limitaciones

- No se admiten las operaciones PATCH condicionales que utilizan condiciones de búsqueda
- Los paquetes de parches JSON deben utilizar recursos binarios con contenido codificado en base64
- FHIRPath Los parches incluidos en los paquetes deben usar recursos de parámetros

Recursos adicionales

Para obtener más información sobre las operaciones PATCH, consulte:

- Documentación del PARCHE FHIR R4
- Especificación del parche FHIR R4 FHIRPath

- RFC 6902: parche JSON
- RFC 6901: puntero JSON

Agrupación de recursos del FHIR

Un FHIR `Bundle` es un contenedor que contiene una colección de recursos del FHIR. AWS HealthLake AWS HealthLake admite dos tipos de paquetes con diferentes comportamientos de procesamiento.

`Batch`los paquetes procesan cada recurso de forma independiente. Si un recurso falla, los recursos restantes aún pueden funcionar correctamente. Cada operación se procesa de forma individual y el procesamiento continúa incluso cuando algunas operaciones fallan. Utilice paquetes por lotes para operaciones masivas en las que sea aceptable un éxito parcial, como cargar varios historiales de pacientes no relacionados.

`Transaction`los paquetes procesan todos los recursos de forma atómica como una sola unidad. Todas las operaciones de recursos se realizan correctamente o no AWS HealthLake se compromete ninguna de ellas. Utilice los paquetes de transacciones cuando necesite garantizar la integridad referencial en todos los recursos relacionados, por ejemplo, si crea un paciente con observaciones y afecciones relacionadas, en el que todos los datos deben registrarse juntos.

Diferencias entre lotes y paquetes de transacciones

Característica	Lote	Transacción
Modelo de procesamiento	Cada operación se realiza correctamente o fracasa de forma independiente.	Todas las operaciones tienen éxito o fallan como una sola unidad atómica.
Administración de errores	El procesamiento continúa incluso si las operaciones individuales fallan.	Se produce un error en todo el paquete si se produce un error en una sola operación.
Orden de ejecución	La orden de ejecución no está garantizada.	Las operaciones se procesan en el orden especificado.
Integridad referenci al	No se aplica en todas las operacion es.	Se aplica a los recursos referenci ados localmente dentro del paquete.

Característica	Lote	Transacción
Más adecuado para	Operaciones masivas en las que es aceptable un éxito parcial.	Recursos relacionados que deben crearse o actualizarse juntos.

Puede agrupar recursos del FHIR del mismo tipo o de tipos diferentes, y estos pueden incluir una combinación de operaciones del FHIR`create`, como, `read` `update``delete`, y. `patch` Para obtener información adicional, consulte el paquete de recursos en la documentación del FHIR R4.

A continuación, se muestran ejemplos de casos de uso para cada tipo de paquete.

Paquetes por lotes

- Cargue varios registros de pacientes no relacionados de diferentes centros durante la sincronización de datos nocturna.
- Cargue de forma masiva los registros históricos de medicamentos cuando algunos registros puedan tener problemas de validación.
- Carga datos de referencia, como organizaciones y profesionales, de forma que los errores individuales no afecten a otras entradas.

Paquetes de transacciones

- Cree un paciente con las observaciones y afecciones relacionadas durante un ingreso al servicio de urgencias, donde todos los datos deben registrarse juntos.
- Actualice la lista de medicamentos del paciente y la información relacionada con las alergias, que debe ser coherente.
- Registre un encuentro completo con el paciente, las observaciones, los procedimientos y la información de facturación como una sola unidad atómica.

> **Important**
>
> Tanto los paquetes de lotes como los de transacciones utilizan la misma estructura `Bundle` de recursos. La única diferencia es el valor del `type` campo.

El siguiente ejemplo muestra un paquete de transacciones con varios tipos de recursos y operaciones.

```
{
  "resourceType": "Bundle",
  "type": "transaction",
  "entry": [
    {
      "fullUrl": "urn:uuid:4f6a30fb-cd3c-4ab6-8757-532101f72065",
      "resource": {
        "resourceType": "Patient",
        "id": "new-patient",
        "active": true,
        "name": [
          {
            "family": "Johnson",
            "given": [
              "Sarah"
            ]
          }
        ],
        "gender": "female",
        "birthDate": "1985-08-12",
        "telecom": [
          {
            "system": "phone",
            "value": "555-123-4567",
            "use": "home"
          }
        ]
      },
      "request": {
        "method": "POST",
        "url": "Patient"
      }
    },
    {
      "fullUrl": "urn:uuid:7f83f473-d8cc-4a8d-86d3-9d9876a3248b",
      "resource": {
        "resourceType": "Observation",
        "id": "blood-pressure",
        "status": "final",
        "code": {
          "coding": [
            {
              "system": "http://loinc.org",
```

```
            "code": "85354-9",
            "display": "Blood pressure panel"
          }
        ],
        "text": "Blood pressure panel"
      },
      "subject": {
        "reference": "urn:uuid:4f6a30fb-cd3c-4ab6-8757-532101f72065"
      },
      "effectiveDateTime": "2023-10-15T09:30:00Z",
      "component": [
        {
          "code": {
            "coding": [
              {
                "system": "http://loinc.org",
                "code": "8480-6",
                "display": "Systolic blood pressure"
              }
            ]
          },
          "valueQuantity": {
            "value": 120,
            "unit": "mmHg",
            "system": "http://unitsofmeasure.org",
            "code": "mm[Hg]"
          }
        },
        {
          "code": {
            "coding": [
              {
                "system": "http://loinc.org",
                "code": "8462-4",
                "display": "Diastolic blood pressure"
              }
            ]
          },
          "valueQuantity": {
            "value": 80,
            "unit": "mmHg",
            "system": "http://unitsofmeasure.org",
            "code": "mm[Hg]"
          }
```

```
          }
        ]
      },
      "request": {
        "method": "POST",
        "url": "Observation"
      }
    },
    {
      "resource": {
        "resourceType": "Appointment",
        "id": "appointment-123",
        "status": "booked",
        "description": "Annual physical examination",
        "start": "2023-11-15T09:00:00Z",
        "end": "2023-11-15T09:30:00Z",
        "participant": [
          {
            "actor": {
              "reference": "urn:uuid:4f6a30fb-cd3c-4ab6-8757-532101f72065"
            },
            "status": "accepted"
          }
        ]
      },
      "request": {
        "method": "PUT",
        "url": "Appointment/appointment-123"
      }
    },
    {
      "request": {
        "method": "DELETE",
        "url": "MedicationRequest/med-request-456"
      }
    }
  ]
}
```

Agrupar los recursos del FHIR como entidades independientes

Para agrupar los recursos del FHIR como entidades independientes

1. Recolecta HealthLake `region` y `datastoreId` valora. Para obtener más información, consulte Obtención de propiedades de los almacenes de datos.
2. Cree una URL para la solicitud utilizando los valores recopilados para HealthLake `region` y`datastoreId`. No especifique un tipo de recurso FHIR en la URL. Para ver la ruta URL completa en el siguiente ejemplo, desplázate sobre el botón Copiar.

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

3. Crea un cuerpo JSON para la solicitud y especifica cada verbo HTTP como parte de los `method` elementos. En el siguiente ejemplo, se utiliza una interacción de `batch` tipos con el `Bundle` recurso para crear `Medication` recursos `Patient` y nuevos. Todas las secciones obligatorias se comentan en consecuencia. Para realizar este procedimiento, guarde el archivo como`batch-independent.json`.

```
{
    "resourceType": "Bundle",
    "id": "bundle-batch",
    "meta": {
        "lastUpdated": "2014-08-18T01:43:30Z"
    },
    "type": "batch",
    "entry": [
        {
            "resource": {
                "resourceType": "Patient",
                "meta": {
                    "lastUpdated": "2022-06-03T17:53:36.724Z"
                },
                "text": {
                    "status": "generated",
                    "div": "Some narrative"
                },
                "active": true,
                "name": [
                    {
                        "use": "official",
                        "family": "Jackson",
                        "given": [
                            "Mateo",
                            "James"
                        ]
```

```
                    }
                ],
                "gender": "male",
                "birthDate": "1974-12-25"
            },
            "request": {
                "method": "POST",
                "url": "Patient"
            }
        },
        {
            "resource": {
                "resourceType": "Medication",
                "id": "med0310",
                "contained": [
                    {
                        "resourceType": "Substance",
                        "id": "sub03",
                        "code": {
                            "coding": [
                                {
                                    "system": "http://snomed.info/sct",
                                    "code": "55452001",
                                    "display": "Oxycodone (substance)"
                                }
                            ]
                        }
                    }
                ],
                "code": {
                    "coding": [
                        {
                            "system": "http://snomed.info/sct",
                            "code": "430127000",
                            "display": "Oral Form Oxycodone (product)"
                        }
                    ]
                },
                "form": {
                    "coding": [
                        {
                            "system": "http://snomed.info/sct",
                            "code": "385055001",
                            "display": "Tablet dose form (qualifier value)"
```

```
                        }
                    ]
                },
                "ingredient": [
                    {
                        "itemReference": {
                            "reference": "#sub03"
                        },
                        "strength": {
                            "numerator": {
                                "value": 5,
                                "system": "http://unitsofmeasure.org",
                                "code": "mg"
                            },
                            "denominator": {
                                "value": 1,
                                "system": "http://terminology.hl7.org/CodeSystem/
v3-orderableDrugForm",
                                "code": "TAB"
                            }
                        }
                    }
                ]
            },
            "request": {
                "method": "POST",
                "url": "Medication"
            }
        }
    ]
}
```

4. Envíe la solicitud . El tipo de `Bundle` lote del FHIR utiliza una `POST` solicitud con la versión 4 de AWS Signature o SMART previa autorización del FHIR. El siguiente ejemplo de código utiliza la herramienta de línea de `curl` comandos con fines de demostración.

   SigV4

   autorización SigV4

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

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

SMART on FHIR

Ejemplo de autorización SMART on FHIR para el tipo de `IdentityProviderConfiguration`datos.

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

La persona que llama puede asignar permisos en la lambda de autorización. Para obtener más información, consulte OAuth osciloscopios 2.0.

El servidor devuelve una respuesta que muestra los `Medication` recursos `Patient` y los recursos creados como resultado de la solicitud de tipo `Bundle` lote.

Condicional PUTs en paquetes

AWS HealthLake admite las actualizaciones condicionales dentro de los paquetes mediante los siguientes parámetros de consulta:

> **Note**
>
> PUTs Las condicionales solo se admiten en `batch` paquetes. `Transaction`los paquetes no admiten las condiciones condicionales. PUTs

- `_id`(independiente)
- `_id`en combinación con uno de los siguientes:
  - `_tag`
  - `_createdAt`
  - `_lastUpdated`

Cuando utilizas la opción condicional PUTs en paquetes, AWS HealthLake evalúa los parámetros de la consulta comparándolos con los recursos existentes y toma medidas en función de los resultados de las coincidencias.

Comportamiento de actualización condicional

Escenario	Estado HTTP	Acción emprendida
Recurso sin ID proporcionado	201 Creado	Siempre crea un recurso nuevo.
Recurso con un nuevo identific ador (no coincide)	201 Creado	Crea un nuevo recurso con el ID especificado.
Recurso con un ID existente (coincidencia única)	200 OK	Actualiza el recurso coincidente.
Recurso con un identificador existente (se detectó un conflicto)	Conflicto, 409	Devuelve un error. No se realiza ningún cambio.
Recurso con un identificador existente (los identificadores no coinciden)	400: solicitud maligna	Devuelve un error. No se realiza ningún cambio.
Varios recursos cumplen las condiciones	Condición previa con error, 412	Devuelve un error. No se realiza ningún cambio.

En el siguiente paquete de ejemplo con una actualización condicional, el `Patient` recurso con el identificador FHIR se `476` actualiza solo si `_lastUpdated=lt2025-04-20` se cumple la condición.

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

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

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

Agrupar los recursos del FHIR como una sola entidad

Para agrupar los recursos del FHIR como una sola entidad

1. HealthLake `region`Recopila y `datastoreId` valora. Para obtener más información, consulte Obtención de propiedades de los almacenes de datos.
2. Cree una URL para la solicitud utilizando los valores recopilados para HealthLake `region` y`datastoreId`. Incluya el tipo de recurso FHIR `Bundle` como parte de la URL. Para ver la ruta URL completa en el siguiente ejemplo, desplázate sobre el botón Copiar.

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

3. Cree un cuerpo JSON para la solicitud, especificando los recursos del FHIR que desea agrupar. En el siguiente ejemplo, se agrupan dos `Patient` recursos. HealthLake Para realizar este procedimiento, guarde el archivo como`batch-single.json`.

   ```
   {
       "resourceType": "Bundle",
       "id": "bundle-minimal",
       "language": "en-US",
       "identifier": {
           "system": "urn:oid:1.2.3.4.5",
           "value": "28b95815-76ce-457b-b7ae-a972e527db4f"
       },
       "type": "document",
   ```

```
    "timestamp": "2020-12-11T14:30:00+01:00",
    "entry": [
        {
            "fullUrl": "urn:uuid:f40b07e3-37e8-48c3-bf1c-ae70fe12dabf",
            "resource": {
                "resourceType": "Composition",
                "id": "f40b07e3-37e8-48c3-bf1c-ae70fe12dabf",
                "status": "final",
                "type": {
                    "coding": [
                        {
                            "system": "http://loinc.org",
                            "code": "60591-5",
                            "display": "Patient summary Document"
                        }
                    ]
                },
                "date": "2020-12-11T14:30:00+01:00",
                "author": [
                    {
                        "reference":
"urn:uuid:45271f7f-63ab-4946-970f-3daaaa0663ff"
                    }
                ],
                "title": "Patient Summary as of December 7, 2020 14:30"
            }
        },
        {
            "fullUrl": "urn:uuid:45271f7f-63ab-4946-970f-3daaaa0663ff",
            "resource": {
                "resourceType": "Practitioner",
                "id": "45271f7f-63ab-4946-970f-3daaaa0663ff",

                "active": true,
                "name": [
                    {
                        "family": "Doe",
                        "given": [
                            "John"
                        ]
                    }
                ]
            }
        }
```

```
    ]
}
```

4. Envíe la solicitud . El tipo de `Bundle` documento FHIR utiliza una `POST` solicitud con el protocolo de AWS firma Signature versión 4. El siguiente ejemplo de código utiliza la herramienta de línea de `curl` comandos con fines de demostración.

   SigV4

   autorización SigV4

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

   SMART on FHIR

   Ejemplo de autorización SMART on FHIR para el tipo de `IdentityProviderConfiguration`datos.

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

```
}
```

La persona que llama puede asignar permisos en la lambda de autorización. Para obtener más información, consulte OAuth osciloscopios 2.0.

El servidor devuelve una respuesta que muestra dos `Patient` recursos creados como resultado de la solicitud de tipo de `Bundle` documento.

Configurar el nivel de validación de los paquetes

Al agrupar los recursos del FHIR, puede especificar opcionalmente un encabezado `x-amzn-healthlake-fhir-validation-level` HTTP para configurar un nivel de validación para el recurso. Este nivel de validación se establecerá para todas las solicitudes de creación y actualización del paquete. AWS HealthLake actualmente admite los siguientes niveles de validación:

- `strict`: Los recursos se validan según el elemento de perfil del recurso o según la especificación R4 si no hay ningún perfil presente. Este es el nivel de validación predeterminado para AWS HealthLake.
- `structure-only`: Los recursos se validan con el R4 e ignoran los perfiles a los que se hace referencia.
- `minimal`: Los recursos se validan mínimamente, ignorando ciertas reglas de R4. Los recursos que no superen las comprobaciones de estructura requeridas se search/analytics actualizarán para incluir una advertencia para la auditoría.

Los recursos incluidos con un nivel de validación mínimo pueden incorporarse a un almacén de datos a pesar de no pasar la validación necesaria para la indexación de las búsquedas. En este caso, los recursos se actualizarán para incluir una extensión específica de Healthlake para documentar dichos errores, y las entradas de la respuesta del paquete incluirán los siguientes recursos: OperationOutcome

```
{
    "resourceType": "Bundle",
    "type": "batch-response",
    "timestamp": "2025-08-25T22:58:48.846287342Z",
    "entry": [
        {
            "response": {
```

```
                "status": "201",
                "location": "Patient/195abc49-ba8e-4c8b-95c2-abc88fef7544/_history/1",
                "etag": "W/\"1\"",
                "lastModified": "2025-08-25T22:58:48.801245445Z",
                "outcome": {
                    "resourceType": "OperationOutcome",
                    "issue": [
                        {
                            "severity": "error",
                            "code": "processing",
                            "details": {
                                "text": "FHIR resource in payload failed FHIR
 validation rules."
                            },
                            "diagnostics": "FHIR resource in payload failed FHIR
 validation rules."
                        }
                    ]
                }
            }
        }
    ]
}
```

Además, se incluirá el siguiente encabezado de respuesta HTTP con el valor «true»:

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

> **Note**
>
> Tenga en cuenta que es posible que los datos ingresados que tengan un formato incorrecto según la especificación R4 no se puedan buscar como se esperaba si se presentan estos errores.

Soporte limitado para el tipo de paquete «mensaje»

HealthLake proporciona un soporte limitado para el tipo de paquete FHIR `message` mediante un proceso de conversión interno. Este soporte está diseñado para situaciones en las que los paquetes de mensajes no se pueden volver a formatear en su origen, por ejemplo, cuando se ingieren ficheros ADT (admisión, alta, transferencia) de sistemas hospitalarios antiguos.

> **Warning**
>
> Esta función requiere la inclusión explícita de las AWS cuentas en la lista y no impone la semántica de los mensajes R4 ni la integridad referencial del FHIR. Póngase en contacto con AWS Support para solicitar la activación de su cuenta antes de usar paquetes de mensajes.

Diferencias clave con respecto al procesamiento de mensajes estándar

- Paquetes de mensajes (especificación FHIR): la primera entrada debe ser una `MessageHeader` que haga referencia a otros recursos. Los recursos carecen de `request` objetos individuales y el MessageHeader evento determina las acciones de procesamiento.
- HealthLake Procesamiento: convierte los paquetes de mensajes en paquetes por lotes mediante la asignación automática de operaciones PUT a cada entrada de recursos. Los recursos se procesan de forma independiente sin imponer la semántica de los mensajes ni la integridad referencial.

Limitaciones importantes

- No se aplican las reglas de procesamiento específicas de los mensajes del FHIR R4
- No hay integridad transaccional en todos los recursos
- Las referencias entre recursos no están validadas
- Requiere la inclusión explícita de cuentas

Ejemplo de estructura de paquete de mensajes

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

```
            "focus": [{"reference": "Encounter/example-id"}]
          }
        },
        {
          "resource": {"resourceType": "Patient", "id": "example-id"}
        },
        {
          "resource": {"resourceType": "Encounter", "id": "example-id"}
        }
      ]
    }
```

> **Note**
>
> Cada recurso se almacena de forma independiente, como si se hubiera enviado mediante operaciones PUT individuales. Si se requiere una validación completa de la semántica de los mensajes o de la integridad referencial del FHIR, procese previamente los paquetes de mensajes o implemente una validación a nivel de aplicación antes de enviarlos.

Transacciones de paquetes asíncronas

AWS HealthLake admite el `Bundle` tipo asíncrono `transaction` que le permite enviar transacciones con hasta 500 recursos. Al enviar una transacción asíncrona, la pone en HealthLake cola para su procesamiento e inmediatamente devuelve una URL de sondeo. Puedes usar esta URL para comprobar el estado y recuperar la respuesta. Sigue el patrón de paquetes asíncronos del FHIR.

Cuándo usar transacciones asíncronas

- Debe enviar más de 100 recursos (límite sincrónico) en una sola transacción.
- Desea evitar bloquear su solicitud mientras espera a que se complete el procesamiento de la transacción.
- Necesita procesar grandes volúmenes de recursos relacionados con un mejor rendimiento.

> **Important**
>
> Los resultados de la encuesta están disponibles durante 90 días después de que se complete la transacción. Después de este período de 90 días, la URL de la encuesta ya no devuelve los resultados. Diseñe su integración para recuperar y almacenar los resultados en esta ventana.

> **Note**
>
> El `Bundle` tipo síncrono `transaction` sigue admitiendo hasta 100 recursos y es el modo de procesamiento predeterminado. Si envía un `Bundle` tipo `transaction` con más de 100 recursos sin el `Prefer: respond-async` encabezado, HealthLake devuelve un `422 Unprocessable Entity` error. Los paquetes con tipo no `batch` se admiten para el procesamiento asíncrono; solo los `Bundle` tipos se `transaction` pueden enviar de forma asíncrona (con un máximo de 500 operaciones).

> **Note**
>
> `PATCH`las operaciones y las condicionales no se admiten en las transacciones de paquetes asíncronas. `PUT`s

Envío de una transacción asíncrona

Para enviar una transacción asíncrona, envíe una `POST` solicitud al punto final del almacén de datos con el encabezado. `Prefer: respond-async` El paquete debe tener un tipo. `transaction` Los paquetes con un tipo no `batch` se admiten para el procesamiento asíncrono de paquetes.

HealthLake realiza las validaciones iniciales del paquete en el momento de su envío. Si la validación se realiza correctamente, HealthLake devuelve HTTP 202 Accepted con un encabezado de `content-location` respuesta que contiene la URL de la encuesta.

Para enviar un tipo **`Bundle`** asíncrono **`transaction`**

1. Envíe una `POST` solicitud al punto final del almacén de HealthLake datos.

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

2. Cree un cuerpo JSON para la solicitud con el tipo de paquete`transaction`. Para realizar este procedimiento, guarde el archivo como`async-transaction.json`.

```
{
    "resourceType": "Bundle",
    "type": "transaction",
    "entry": [
        {
            "resource": {
                "resourceType": "Patient",
                "active": true,
                "name": [
                    {
                        "use": "official",
                        "family": "Smith",
                        "given": ["Jane"]
                    }
                ],
                "gender": "female",
                "birthDate": "1990-01-15"
            },
            "request": {
                "method": "POST",
                "url": "Patient"
            }
        },
        {
            "resource": {
                "resourceType": "Observation",
                "status": "final",
                "code": {
                    "coding": [
                        {
                            "system": "http://loinc.org",
                            "code": "85354-9",
                            "display": "Blood pressure panel"
                        }
                    ]
                },
                "subject": {
```

```
                    "reference": "urn:uuid:example-patient-id"
                }
            },
            "request": {
                "method": "POST",
                "url": "Observation"
            }
        }
    ]
}
```

3. Envíe la solicitud con el `Prefer: respond-async` encabezado. El tipo de `Bundle` transacción del FHIR utiliza una `POST` solicitud con la versión 4 de la AWS firma o SMART con la autorización del FHIR. El siguiente ejemplo de código utiliza la herramienta de línea de `curl` comandos con fines de demostración.

   SigV4

   autorización SigV4

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

   SMART on FHIR

   Ejemplo de autorización SMART on FHIR para el tipo de `IdentityProviderConfiguration`datos.

   ```
   {
       "AuthorizationStrategy": "SMART_ON_FHIR",
       "FineGrainedAuthorizationEnabled": true,
       "IdpLambdaArn": "arn:aws:lambda:your-region:your-account-id:function:your-
   lambda-name",
       "Metadata": "{\"issuer\":\"https://ehr.example.com\", \"jwks_uri\":
   \"https://ehr.example.com/.well-known/jwks.json\",\"authorization_endpoint
   \":\"https://ehr.example.com/auth/authorize\",\"token_endpoint\":\"https://
   ```

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

La persona que llama puede asignar permisos en la lambda de autorización. Para obtener más información, consulte OAuth osciloscopios 2.0.

4. Si el envío se realiza correctamente, el servidor devuelve HTTP 202 Accepted. El encabezado de la `content-location` respuesta contiene la URL de la encuesta. El cuerpo de la respuesta es un `OperationOutcome` recurso.

```
HTTP/1.1 202 Accepted
content-location: https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
Transaction/transactionId
```

```
{
    "resourceType": "OperationOutcome",
    "issue": [
        {
            "severity": "information",
            "code": "informational",
            "diagnostics": "Submitted Asynchronous Bundle Transaction",
            "location": [
                "https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
Transaction/transactionId"
            ]
        }
    ]
}
```

Sondeo del estado de la transacción

Después de enviar una transacción asíncrona, usa la URL de sondeo del encabezado de `content-location` respuesta para comprobar el estado de la transacción. Envía una `GET` solicitud a la URL de la encuesta.

> **Note**
>
> En el caso de los almacenes de datos habilitados para SMART o FHIR, el token de autorización debe incluir `read` permisos sobre el tipo de `Transaction` recurso para comprobar el estado de la transacción. Para obtener más información sobre SMART en los ámbitos del FHIR, consulte. Los osciloscopios SMART on FHIR OAuth 2.0 son compatibles con HealthLake

Envíe una `GET` solicitud a la URL de la encuesta. En el siguiente ejemplo, se utiliza la herramienta de línea de `curl` comandos.

SigV4

Autorización SigV4

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

SMART on FHIR

Autorización SMART sobre la FHIR. El token de autorización debe incluir `read` los permisos correspondientes al tipo `Transaction` de recurso.

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

En la siguiente tabla se describen las posibles respuestas.

Códigos de respuesta al sondeo

Estado HTTP	Significado	Cuerpo de respuesta
202: aceptada	La transacción está en cola	`OperationOutcome` con el diagnóstico «ENVIADO»
202: aceptada	La transacción se está procesando	`OperationOutcome` con el diagnóstico «IN_PROGRESS»
200 OK	La transacción se completó correctamente	`Bundle`con el tipo `transaction-response`
4xx/5xx	Fallo en la transacción	`OperationOutcome` con detalles del error

Los siguientes ejemplos muestran cada tipo de respuesta.

Transacción en cola (202)

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

Procesamiento de transacciones (202)

```
{
    "resourceType": "OperationOutcome",
    "id": "transactionId",
    "issue": [
        {
```

```
            "severity": "information",
            "code": "informational",
            "diagnostics": "IN_PROGRESS"
        }
    ]
}
```

Transacción completada (200)

```
{
    "resourceType": "Bundle",
    "type": "transaction-response",
    "entry": [
        {
            "response": {
                "status": "201",
                "location": "Patient/example-id/_history/1",
                "etag": "W/\"1\"",
                "lastModified": "2024-01-15T10:30:00.000Z"
            }
        },
        {
            "response": {
                "status": "201",
                "location": "Observation/example-id/_history/1",
                "etag": "W/\"1\"",
                "lastModified": "2024-01-15T10:30:00.000Z"
            }
        }
    ]
}
```

Transacción fallida (4xx/5xx)

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

```
        }
    ]
}
```

Procesando el pedido

Los paquetes de tipo asíncrono `transaction` están en cola, pero no se procesan siguiendo un orden de envío estricto. HealthLake optimiza el procesamiento en función de la capacidad disponible y la carga del sistema.

> **Important**
>
> No dependa de que las transacciones se procesen en el orden en que se enviaron. Por ejemplo, si envía la transacción A a las 10:00 a.m. y la transacción B a las 10:01 a.m., la transacción B podría completarse antes que la transacción A. Diseñe su solicitud para:
>
> - Gestione la out-of-order finalización.
> - Utilice la URL de la encuesta para realizar un seguimiento de cada transacción de forma independiente.
> - Implemente la secuenciación a nivel de aplicación si el orden es importante para su caso de uso.

Cuotas y estrangulamiento

Las siguientes cuotas y límites de tasa se aplican a las transacciones asincrónicas.

Cuotas de transacciones asincrónicas

Cuota	Valor	Ajustable
Máximo de operaciones por transacción asíncrona	500	No
Máximo de transacciones pendientes por almacén de datos	500	Sí

- Las transacciones asincrónicas comparten los mismos límites de velocidad de la API definidos en. Cuotas de servicio

- El sondeo del estado de las transacciones comparte los mismos límites de velocidad de la API que las operaciones de lectura (`GET`) en los recursos del FHIR.
- Si se alcanza el límite de transacciones pendientes, los envíos posteriores devuelven un error hasta que se completen las transacciones existentes.

Gestión de errores

En el caso de un paquete de «transacciones», todos los recursos del FHIR contenidos en el paquete se procesan como una operación atómica. Todos los recursos de la operación deben realizarse correctamente o no se procesará ninguna operación del paquete.

Los errores se dividen en dos categorías: los errores de envío, que se HealthLake devuelven de forma sincrónica, y los errores de procesamiento, que se recuperan mediante un sondeo.

Errores de envío

HealthLake valida el paquete en el momento del envío y devuelve los errores de forma sincrónica antes de que la transacción se ponga en cola. Los errores de envío incluyen los errores de validación de recursos del FHIR no válidos, los tipos de recursos no admitidos, la superación del límite de 500 operaciones y el uso del `Prefer: respond-async` encabezado con paquetes de lotes. Si se ha alcanzado el límite de transacciones pendientes del almacén de datos, HealthLake devuelve un. `ThrottlingException` Cuando se produce un error de envío, la transacción no se pondrá en cola.

Errores de procesamiento

Los errores de procesamiento se producen una vez que la transacción se ha puesto en cola y se devuelven a través de la URL de consulta. Estos incluyen los conflictos de transacciones, en los que otra operación modificó un recurso que forma parte de la transacción, y los errores del servidor durante el procesamiento. Cuando se produce un error de procesamiento, no se produce ninguna mutación de recursos para los recursos de la transacción. La URL del sondeo devolverá una `OperationOutcome` con los detalles del error.

Eliminar un recurso del FHIR

La `delete` interacción del FHIR elimina un recurso del FHIR existente de un HealthLake almacén de datos. Para obtener información adicional, consulte la documentación de la `delete`API RESTful R4 del FHIR.

Para eliminar un recurso del FHIR

1. HealthLake `region`Recopila y `datastoreId` valora. Para obtener más información, consulte Obtención de propiedades de los almacenes de datos.
2. Determine el tipo de `Resource` FHIR que desea eliminar y recopile el `id` valor asociado. Para obtener más información, consulte Tipos de recurso.
3. Cree una URL para la solicitud utilizando los valores recopilados para HealthLake `region` y`datastoreId`. Incluya también el `Resource` tipo FHIR y su correspondiente`id`. Para ver la ruta URL completa en el siguiente ejemplo, desplázate sobre el botón Copiar.

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

4. Envíe la solicitud . La `delete` interacción del FHIR utiliza una `DELETE` solicitud con la versión 4 de la AWS firma o SMART con la autorización del FHIR. En el siguiente `curl` ejemplo, se elimina un `Patient` recurso del FHIR existente de un HealthLake banco de datos. Para ver el ejemplo completo, desplácese sobre el botón Copiar.

   SigV4

   Autorización SigV4

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

   El servidor devuelve un código de estado `204` HTTP que confirma que el recurso se ha eliminado del almacén de HealthLake datos. Si se produce un error en una solicitud de eliminación, recibirá una `400` serie de códigos de estado HTTP que indicará el motivo del error en la solicitud.

   SMART on FHIR

   Ejemplo de autorización SMART on FHIR para el tipo `IdentityProviderConfiguration`de datos.

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

La persona que llama puede asignar permisos en la lambda de autorización. Para obtener más información, consulte OAuth osciloscopios 2.0.

AWS Console

1. Inicie sesión en la página Ejecutar consultas de la HealthLake consola.

2. En la sección de configuración de consultas, realice las siguientes selecciones.

- ID de almacén de datos: elija un ID de almacén de datos para generar una cadena de consulta.
- Tipo de consulta: elija`Delete`.
- Tipo de recurso: elija el tipo de recurso FHIR que desee eliminar.
- ID de recurso: introduzca el ID de recurso del FHIR.

3. Elija Ejecutar consulta.

Eliminar los recursos del FHIR en función de las condiciones

La eliminación condicional resulta especialmente útil cuando no se conoce el identificador específico del recurso del FHIR, pero se dispone de otra información de identificación sobre el recurso que se desea eliminar.

La eliminación condicional le permite eliminar un recurso existente en función de los criterios de búsqueda y no de un ID de FHIR lógico. Cuando el servidor procesa la solicitud de eliminación, realiza una búsqueda utilizando las capacidades de búsqueda estándar para el tipo de recurso a fin de resolver un único identificador lógico para la solicitud.

Cómo funciona la eliminación condicional

La acción del servidor depende del número de coincidencias que encuentre:

1. No hay coincidencias: el servidor intenta realizar una eliminación normal y responde adecuadamente (404 No se ha encontrado ningún recurso, 204 no se ha encontrado contenido para un recurso ya eliminado)
2. Una coincidencia: el servidor realiza una eliminación normal del recurso coincidente
3. Coincidencias múltiples: devuelve el error 412: no se pudo cumplir la condición previa, lo que indica que los criterios del cliente no eran lo suficientemente selectivos

Escenarios de respuesta

AWS HealthLake gestiona las operaciones de borrado condicional con los siguientes patrones de respuesta:

Operaciones exitosas

- Cuando los criterios de búsqueda identifican correctamente un único recurso activo, el sistema devuelve 204 Sin contenido tras completar la eliminación, al igual que en las operaciones de eliminación estándar.

Eliminación condicional basada en ID

Al realizar una eliminación condicional basada en `id` parámetros adicionales (`createdAt`,`tag`, `o_lastUpdated`):

- 204 Sin contenido: el recurso ya se ha eliminado
- 404 No encontrado: el recurso no existe
- 409 Conflicto: el identificador coincide pero otros parámetros no coinciden

Non-ID-Based Eliminación condicional

Cuando no `id` se proporciona o cuando se utilizan parámetros distintos de `createdAttag`, `o_lastUpdated`:

- 404 No se encontró: no se encontraron coincidencias

Situaciones de conflicto

Varios escenarios dan como resultado 412 respuestas fallidas a una condición previa:

- Varios recursos coinciden con sus criterios de búsqueda (los criterios no son lo suficientemente específicos)
- La versión entra en conflicto cuando se utilizan ETag encabezados con `If-Match`
- Las actualizaciones de recursos se producen entre las operaciones de búsqueda y eliminación

Ejemplo de una eliminación condicional correcta

En el siguiente ejemplo, se elimina un recurso para pacientes en función de criterios específicos:

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

Esta solicitud elimina un recurso para pacientes en el que:

- El nombre es «peter»
- La fecha de nacimiento es el 1 de enero de 2000
- El número de teléfono es 1234567890

Prácticas recomendadas

1. Utilice criterios de búsqueda específicos para evitar múltiples coincidencias y evitar errores 412.

2. Tenga en cuenta ETag los encabezados para el control de versiones cuando sea necesario para gestionar las modificaciones simultáneas.
3. Gestione las respuestas de error de forma adecuada:
   - Para 404: Refina tus criterios de búsqueda
   - Para 412: especifique los criterios o resuelva los conflictos de versiones
4. Prepárese para los conflictos de temporización en entornos de alta concurrencia, en los que los recursos pueden modificarse entre las operaciones de búsqueda y eliminación.

Idempotencia y concurrencia

Claves de idempotencia

AWS HealthLake admite claves de idempotencia para las `POST` operaciones del FHIR, lo que proporciona un mecanismo sólido para garantizar la integridad de los datos durante la creación de los recursos. Al incluir un UUID único como clave de idempotencia en el encabezado de la solicitud, las aplicaciones sanitarias pueden garantizar que cada recurso del FHIR se cree exactamente una vez, incluso en escenarios que impliquen inestabilidad de la red o reintentos automáticos.

Esta característica es particularmente crucial para los sistemas de salud, donde la duplicación de registros médicos podría tener graves consecuencias. Cuando se reciba una solicitud con la misma clave de idempotencia que una solicitud anterior, HealthLake devolverá el recurso original en lugar de crear un duplicado. Por ejemplo, esto podría ocurrir durante un ciclo de reintento o debido a canalizaciones de solicitudes redundantes. El uso de la clave de idempotencia permite mantener la coherencia de los datos y, HealthLake al mismo tiempo, proporciona una experiencia perfecta para las aplicaciones cliente que gestionan problemas de conectividad intermitentes.

Implementación

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

Escenarios de respuesta

Primera solicitud (201 creada)

- El nuevo recurso se creó correctamente
- La respuesta incluye el identificador del recurso

Solicitud duplicada (409 Conflicto)

- Se detectó la misma clave de idempotencia
- Recurso original devuelto
- No se ha creado ningún recurso nuevo

Solicitud no válida (400 solicitudes incorrectas)

- UUID mal formado
- Faltan campos obligatorios

Prácticas recomendadas

- Genera un UUID único para cada nueva creación de recursos
- Almacene las claves de idempotencia para la lógica de reintento
- Utilice un formato de clave coherente: se recomienda el UUID v4
- Implételo en las aplicaciones cliente que gestionen la creación de recursos

> **Note**
>
> Esta función es particularmente valiosa para los sistemas de salud que requieren una precisión de datos estricta y evitan la duplicación de registros médicos.

ETag en AWS HealthLake

AWS HealthLake utiliza un control ETags de simultaneidad optimista en los recursos del FHIR, proporcionando un mecanismo fiable para gestionar las modificaciones simultáneas y mantener la coherencia de los datos. Un ETag es un identificador único que representa una versión específica de un recurso y funciona como un sistema de control de versiones a través de encabezados HTTP. Al leer o modificar los recursos, las aplicaciones pueden utilizarlos ETags para evitar sobrescrituras

no deseadas y garantizar la integridad de los datos, especialmente en escenarios con posibles actualizaciones simultáneas.

Ejemplo de implementación

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

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

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

Escenarios de respuesta

Funcionamiento exitoso (200 OK o 204 sin contenido)

- ETag coincide con la versión actual
- La operación se desarrolla según lo previsto

Conflicto de versión (error en la condición previa 412)

- ETag no coincide con la versión actual
- Actualización rechazada para evitar la pérdida de datos

Prácticas recomendadas

- Incluir ETags en todas las operaciones de actualización y eliminación
- Implemente la lógica de reintento para gestionar los conflictos de versiones
- Utilice If-None-Match: * para escenarios create-if-not-exists
- Compruebe siempre la ETag frescura antes de realizar modificaciones

Este sistema de control de simultaneidad es esencial para mantener la integridad de los datos de atención médica, especialmente en entornos con varios usuarios o sistemas que acceden a los mismos recursos y los modifican.

Búsqueda de recursos del FHIR en AWS HealthLake

Utilice la `search`interacción del FHIR para buscar un conjunto de recursos del FHIR en un banco de HealthLake datos en función de algunos criterios de filtrado. La `search` interacción se puede realizar mediante una solicitud `GET` o`POST`. Para las búsquedas que incluyan información de identificación personal (PII) o información de salud protegida (PHI), se recomienda utilizar `POST` solicitudes, ya que la PII y la PHI se agregan como parte del cuerpo de la solicitud y se cifran durante el tránsito.

> **Note**
>
> La `search` interacción entre el FHIR que se describe en este capítulo se basa en el estándar R4 del HL7 FHIR para el intercambio de datos de atención médica. Como es una representación de un servicio de la HL7 FHIR, no se ofrece a través de y. AWS CLI AWS SDKs Para obtener más información, consulte la documentación de `search`la API R4 RESTful del FHIR.

HealthLake admite un subconjunto de parámetros de búsqueda del FHIR R4. Para obtener más información, consulte Parámetros de búsqueda FHIR R4 para HealthLake.

Temas

- Búsqueda de recursos del FHIR con GET
- Búsqueda de recursos del FHIR con POST
- Niveles de consistencia de búsqueda del FHIR

Búsqueda de recursos del FHIR con GET

Puede utilizar `GET` las solicitudes para buscar en un banco HealthLake de datos. Cuando se usa`GET`, HealthLake permite proporcionar parámetros de búsqueda como parte de la URL, pero no como parte del cuerpo de la solicitud. Para obtener más información, consulte Parámetros de búsqueda FHIR R4 para HealthLake.

> **Importante**
>
> Para las búsquedas que incluyen información de identificación personal (PII) o información de salud protegida (PHI), las mejores prácticas de seguridad recomiendan utilizar `POST`

solicitudes, ya que la PII y la PHI se añaden como parte del cuerpo de la solicitud y se cifran en tránsito. Para obtener más información, consulte Búsqueda de recursos del FHIR con POST.

El siguiente procedimiento va seguido de ejemplos que se utilizan GET para buscar en un almacén de HealthLake datos.

Para buscar en un HealthLake banco de datos con **GET**

1. Recopilar HealthLake `region` y `datastoreId` valorar. Para obtener más información, consulte Obtención de propiedades de los almacenes de datos.
2. Determine el tipo de recurso del FHIR para buscar y recopilar el `id` valor asociado. Para obtener más información, consulte Tipos de recurso.
3. Cree una URL para la solicitud utilizando los valores recopilados para HealthLake `region` y`datastoreId`. Incluya también el `Resource` tipo de FHIR y los parámetros de búsqueda compatibles. Para ver la ruta URL completa en el siguiente ejemplo, desplázate sobre el botón Copiar.

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

4. Envíe la `GET` solicitud con la versión 4 de AWS Signature o SMART con la autorización del FHIR. El siguiente `curl` ejemplo devuelve el número total de `Patient` recursos de un almacén de HealthLake datos. Para ver el ejemplo completo, desplácese sobre el botón Copiar.

   SigV4

   Autorización SigV4

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

SMART on FHIR

Ejemplo de autorización SMART on FHIR para el tipo de `IdentityProviderConfiguration`datos.

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

La persona que llama puede asignar permisos en la lambda de autorización. Para obtener más información, consulte OAuth osciloscopios 2.0.

AWS Console

> **Note**
>
> La HealthLake consola solo admite la autorización SiGv4. La autorización SMART on FHIR se admite mediante AWS CLI y. AWS SDKs

1. Inicie sesión en la página Ejecutar consultas de la HealthLake consola.

2. En la sección de configuración de consultas, realice las siguientes selecciones.

- ID de almacén de datos: elija un ID de almacén de datos para generar una cadena de consulta.
- Tipo de consulta: elija`Search with GET`.
- Tipo de recurso: elija el tipo de recurso del FHIR en el que desea buscar.
- Parámetros de búsqueda: seleccione un parámetro de búsqueda o una combinación de parámetros de búsqueda para centrar la consulta en registros específicos.

3. Elija Ejecutar consulta.

Ejemplos: busque con GET

Las siguientes pestañas proporcionan ejemplos para buscar tipos de recursos específicos del FHIR con. `GET` Los ejemplos muestran cómo especificar los parámetros de búsqueda en la solicitud URLs.

> **Note**
>
> La HealthLake consola solo admite la autorización SiGv4. La autorización SMART on FHIR se admite mediante AWS CLI y. AWS SDKs
> HealthLake admite un subconjunto de parámetros de búsqueda del FHIR R4. Para obtener más información, consulte Parámetros de búsqueda.

Patient (age)

Si bien la edad no es un tipo de recurso definido en el FHIR, se captura como un elemento del tipo de recurso. `Patient` Utilice el siguiente ejemplo para realizar una solicitud de búsqueda `GET` basada en tipos de `Patient`recursos mediante el elemento BirthDate y el comparador de `eq` búsquedas para buscar personas nacidas en el año 1997.

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

Condition

Utilice el siguiente ejemplo para realizar una `GET` solicitud sobre el tipo de `Condition`recurso. La búsqueda busca condiciones en el almacén de HealthLake datos que contienen el código médico de SNOMED`72892002`, que se traduce como. `Normal pregnancy`

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

DocumentationReference

En el siguiente ejemplo, se muestra cómo crear una GET solicitud sobre el tipo de DocumentReferencerecurso para Patient personas con diagnóstico de estreptococos y a quienes también se les ha recetado amoxicilina.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
DocumentReference?_lastUpdated=le2021-12-19&infer-icd10cm-entity-text-concept-
score;=streptococcal|0.6&infer-rxnorm-entity-text-concept-score=Amoxicillin|0.8
```

Location

Utilice el siguiente ejemplo para realizar una solicitud sobre el tipo de recursoGET. Location La siguiente búsqueda busca ubicaciones en el banco de HealthLake datos que contienen el nombre de la ciudad Boston como parte de la dirección.

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

Observation

Utilice el siguiente ejemplo para realizar una solicitud de búsqueda GET basada en el tipo Observationde recurso. Esta búsqueda utiliza el parámetro value-concept de búsqueda para buscar el código médico266919005, que se traduce comoNever smoker.

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

Búsqueda de recursos del FHIR con POST

Puede utilizar la searchinteracción del FHIR con POST las solicitudes para buscar en un banco de HealthLake datos. Cuando se usaPOST, HealthLake admite los parámetros de búsqueda en la URL o en el cuerpo de una solicitud, pero no puede usar ambos en una sola solicitud.

> **Importante**
>
> Para las búsquedas que incluyen información de identificación personal (PII) o información de salud protegida (PHI), las mejores prácticas de seguridad recomiendan el uso de `POST` solicitudes, ya que la PII y la PHI se agregan como parte del cuerpo de la solicitud y se cifran durante el tránsito.

El siguiente procedimiento va seguido de ejemplos en los que se utiliza la `search` interacción del FHIR R4 `POST` para buscar en un almacén de datos. HealthLake Los ejemplos muestran cómo especificar los parámetros de búsqueda en el cuerpo de la solicitud JSON.

Para buscar en un HealthLake banco de datos con **`POST`**

1. Recopile HealthLake `region` y `datastoreId` valore. Para obtener más información, consulte Obtención de propiedades de los almacenes de datos.
2. Determine el tipo de recurso del FHIR para buscar y recopilar el `id` valor asociado. Para obtener más información, consulte Tipos de recurso.
3. Cree una URL para la solicitud utilizando los valores recopilados para HealthLake `region` y`datastoreId`. Incluya también el `Resource` tipo y la `_search` interacción del FHIR. Para ver la ruta URL completa en el siguiente ejemplo, desplázate sobre el botón Copiar.

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

4. Cree un cuerpo JSON para la solicitud y especifique los datos del FHIR que desee buscar. Para realizar este procedimiento, buscará `Observation` recursos para encontrar pacientes que nunca hayan fumado. Para especificar el estado del código médico`Never smoker`, configúrelo `value-concept=266919005` en el cuerpo de la solicitud en JSON. Guarde el archivo como `search-observation.json`.

   ```
   value-concept=266919005
   ```

5. Envíe la solicitud . La `search` interacción con el FHIR utiliza la `GET` solicitud con la versión 4 de AWS Signature o SMART si cuenta con la autorización del FHIR.

> **Note**
>
> Al realizar una POST solicitud con parámetros de búsqueda en el cuerpo de la solicitud, utilícelos `Content-Type: application/x-www-form-urlencoded` como parte del encabezado.

En el siguiente `curl` ejemplo, se realiza una solicitud de búsqueda basada en POST para el tipo `Observation` de recurso. La solicitud utiliza el parámetro `value-concept`de búsqueda para buscar un código médico `266919005` que indique un valor`Never smoker`. Para ver el ejemplo completo, desplázate sobre el botón Copiar.

SigV4

Autorización SigV4

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

SMART on FHIR

Ejemplo de autorización SMART on FHIR para el tipo de `IdentityProviderConfiguration`datos.

```
{
    "AuthorizationStrategy": "SMART_ON_FHIR",
    "FineGrainedAuthorizationEnabled": true,
    "IdpLambdaArn": "arn:aws:lambda:your-region:your-account-id:function:your-
lambda-name",
    "Metadata": "{\"issuer\":\"https://ehr.example.com\", \"jwks_uri\":
\"https://ehr.example.com/.well-known/jwks.json\",\"authorization_endpoint
\":\"https://ehr.example.com/auth/authorize\",\"token_endpoint\":\"https://
ehr.token.com/auth/token\",\"token_endpoint_auth_methods_supported\":
```

```
[\"client_secret_basic\",\"foo\"],\"grant_types_supported\":[\"client_credential
\",\"foo\"],\"registration_endpoint\":\"https://ehr.example.com/auth/
register\",\"scopes_supported\":[\"openId\",\"profile\",\"launch\"],
\"response_types_supported\":[\"code\"],\"management_endpoint\":\"https://
ehr.example.com/user/manage\",\"introspection_endpoint\":\"https://
ehr.example.com/user/introspect\",\"revocation_endpoint\":\"https://
ehr.example.com/user/revoke\",\"code_challenge_methods_supported\":[\"S256\"],
\"capabilities\":[\"launch-ehr\",\"sso-openid-connect\",\"client-public\",
\"permission-v2\"]}"
}
```

La persona que llama puede asignar permisos en la lambda de autorización. Para obtener más información, consulte OAuth osciloscopios 2.0.

Ejemplos: buscar con POST

Las siguientes pestañas proporcionan ejemplos para buscar tipos de recursos específicos del FHIR con. `POST` Los ejemplos muestran cómo especificar una solicitud en el URLs.

> **Note**
>
> La HealthLake consola solo admite la autorización SiGv4. La autorización SMART on FHIR se admite mediante AWS CLI y. AWS SDKs
> HealthLake admite un subconjunto de parámetros de búsqueda del FHIR R4. Para obtener más información, consulte Parámetros de búsqueda.

Patient (age)

Si bien la edad no es un tipo de recurso definido en el FHIR, se captura como un elemento del tipo de recurso. `Patient` Utilice el siguiente ejemplo para realizar una solicitud de búsqueda `POST` basada en el tipo de `Patient` recurso. En el siguiente ejemplo de búsqueda, se utiliza el `eq` comparador de búsquedas para buscar personas nacidas en 1997.

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

Para especificar el año 1997 en la búsqueda, añada el siguiente elemento al cuerpo de la solicitud.

```
birthdate=eq1997
```

Condition

Utilice lo siguiente para realizar una `POST` solicitud sobre el tipo `Condition` de recurso. Esta búsqueda busca ubicaciones en su almacén de HealthLake datos que contienen el código médico`72892002`.

Debe especificar la URL de la solicitud y el cuerpo de la solicitud. A continuación, se muestra un ejemplo de la URL de la solicitud.

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

Para especificar el código médico que desea buscar, añada el siguiente elemento JSON al cuerpo de la solicitud.

```
code=72892002
```

DocumentReference

Para ver los resultados del procesamiento HealthLake del lenguaje natural (NLP) integrado al realizar una `POST` solicitud sobre el tipo de `DocumentReference` recurso, formatee la solicitud de la siguiente manera.

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

Para especificar los parámetros de `DocumentReference` búsqueda a los que se debe hacer referencia, consulteTipos de parámetros de búsqueda. La siguiente cadena de consulta utiliza varios parámetros de búsqueda para buscar en las operaciones de la API Amazon Comprehend Medical utilizadas para generar los resultados de la PNL integrada.

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

Location

Utilice el siguiente ejemplo para realizar una `POST` solicitud sobre el tipo de `Location` recurso. La búsqueda busca ubicaciones en el almacén de HealthLake datos que contienen el nombre de la ciudad Boston como parte de la dirección.

Debe especificar la URL y el cuerpo de la solicitud. A continuación, se muestra un ejemplo de la URL de la solicitud.

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

Para `Boston` especificarlo en la búsqueda, agrega el siguiente elemento al cuerpo de la solicitud:

```
address=Boston
```

Observation

Utilice el siguiente ejemplo para realizar una solicitud de búsqueda `POST` basada en el tipo `Observation` de recurso. La búsqueda utiliza el parámetro de `value-concept` búsqueda para buscar el código médico `266919005` que indica`Never smoker`.

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

Para especificar el estado`Never smoker`, `value-concept=266919005` establézcalo en el cuerpo del JSON.

```
value-concept=266919005
```

Niveles de consistencia de búsqueda del FHIR

El índice HealthLake de búsqueda de AWS funciona según un modelo de coherencia eventual para `GET` y `POST` con las operaciones de BÚSQUEDA. En aras de la coherencia, si hay una actualización pendiente del índice de búsqueda de un recurso, los resultados de la búsqueda excluyen la versión N-1 del recurso hasta que se complete la actualización del índice.

AWS HealthLake ahora incluye la posibilidad de seleccionar cómo se comportará el modelo de coherencia para los recursos actualizados. Los desarrolladores pueden incluir «coherencia

eventual», el comportamiento predeterminado descrito anteriormente, o «coherencia sólida». Strong Consistency permitirá incluir en los resultados de la búsqueda la versión N-1 del recurso para los recursos con actualizaciones pendientes de los índices de búsqueda. Esto se puede utilizar en casos de uso en los que se requieran todos los recursos en el resultado, incluso cuando la actualización del índice de búsqueda aún no se haya completado. Los clientes pueden controlar este comportamiento mediante el encabezado de la `x-amz-fhir-history-consistency-level` solicitud.

Niveles de consistencia

Consistencia sólida

`x-amz-fhir-history-consistency-level: strong`Configúrelo para que devuelva todos los registros coincidentes, incluidos aquellos con actualizaciones pendientes en el índice de búsqueda. Utilice esta opción cuando necesite buscar recursos inmediatamente después de las actualizaciones.

Consistencia final

`x-amz-fhir-history-consistency-level: eventual`Configúrelo para que se devuelvan solo los registros que hayan completado las actualizaciones del índice de búsqueda. Este es el comportamiento predeterminado si no se especifica ningún nivel de coherencia.

Ejemplo de uso

1. Al actualizar un recurso:

```
POST <baseURL>/Patient
Content-Type: application/fhir+json
x-amz-fhir-history-consistency-level: strong

{
  "resourceType": "Patient",
  "id": "123",
  "meta": {
    "profile": ["http://hl7.org/fhir/us/core/StructureDefinition/us-core-patient"]
  },
  "identifier": [
    {
      "system": "http://example.org/identifiers",
      "value": "123"
    }
```

```
  ],
  "active": true,
  "name": [
    {
      "family": "Smith",
      "given": ["John"]
    }
  ],
  "gender": "male",
  "birthDate": "1970-01-01"
}
```

2. Búsqueda posterior:

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

Prácticas recomendadas

- Utilice una coherencia sólida cuando necesite buscar inmediatamente recursos actualizados recientemente
- Utilice la coherencia eventual para las consultas generales en las que la visibilidad inmediata no sea fundamental
- Tenga en cuenta el equilibrio entre la visibilidad inmediata y el posible impacto en el rendimiento

> **Note**
>
> La configuración del nivel de coherencia afecta a la rapidez con la que aparecen los recursos actualizados en los resultados de búsqueda, pero no afecta al almacenamiento real de los recursos.
> Si se establece el `x-amz-fhir-history-consistency-level` encabezado opcional en «fuerte», se duplica el consumo de capacidad de escritura por recurso.
> Esta función solo se aplica a los almacenes de datos que tienen activado el historial de versiones (todos los almacenes de datos creados después del 25 de octubre de 2024 lo tienen activado de forma predeterminada).

Exportación de datos del FHIR con AWS HealthLake

Utilice AWS HealthLake acciones nativas para iniciar, describir y enumerar los trabajos de exportación del FHIR. Puede poner en cola los trabajos de exportación. Los trabajos de exportación asíncronos se procesan de forma FIFO (primero en entrar, primero en salir). Puede poner los trabajos en cola de la misma manera que se inician los trabajos de exportación. Si hay alguno en curso, simplemente se pondrá en cola. Puede crear, leer, actualizar o eliminar los recursos del FHIR mientras se está realizando un trabajo de exportación.

> **Note**
>
> También puede exportar los datos del FHIR desde un banco de HealthLake datos mediante la operación FHIR R4. `$export` Para obtener más información, consulte Exportación HealthLake de datos con FHIR `$export`.

Temas

- Inicio de un trabajo de exportación del FHIR
- Conseguir que el FHIR exporte propiedades laborales
- Listado de trabajos de exportación del FHIR

Inicio de un trabajo de exportación del FHIR

Se utiliza `StartFHIRExportJob` para iniciar un trabajo de exportación del FHIR desde un almacén de HealthLake datos. Los siguientes menús proporcionan un procedimiento Consola de administración de AWS y ejemplos de código para AWS CLI y AWS SDKs. Para obtener más información, consulta `StartFHIRExportJob` en la AWS HealthLake Referencia de la API de .

> **Nota**
>
> HealthLake es compatible con la especificación R4 del FHIR para el intercambio de datos de atención médica. Por lo tanto, todos los datos de salud se exportan en formato FHIR R4.

Para iniciar un trabajo de exportación de FHIR

Elija un menú en función de sus preferencias de acceso a. AWS HealthLake

AWS CLI y SDKs

CLI

AWS CLI

Para iniciar un trabajo de exportación de FHIR

El siguiente `start-fhir-export-job` ejemplo muestra cómo iniciar un trabajo de exportación del FHIR utilizando AWS HealthLake.

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

Salida:

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

- Para obtener información sobre la API, consulte Start FHIRExport Job in AWS CLI Command Reference.

Python

SDK para Python (Boto3)

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

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


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

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

- Para obtener más información sobre la API, consulta la referencia de la API Start FHIRExport Job in AWS SDK for Python (Boto3).

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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

- Para obtener más información sobre la API, consulte Start FHIRExport Job in AWS SDK para obtener una referencia sobre la API ABAP de SAP.

> **Ejemplo de disponibilidad**
>
> ¿No encuentra lo que necesita? Solicita un ejemplo de código mediante el enlace Enviar comentarios que aparece en la barra lateral derecha de esta página.

AWS Consola

1. Inicie sesión en la página de almacenes de datos de la HealthLake consola.
2. Elija un almacén de datos.
3. Seleccione Exportar.

   Se abre la página de exportación.
4. En la sección Datos de salida, introduce la siguiente información:
   - Ubicación de los datos de salida en Amazon S3
   - Cifrado de salida

5. En la sección Permisos de acceso, elija Usar un rol de servicio de IAM existente y seleccione el rol en el menú Nombre del rol o elija Crear un rol de IAM.
6. Elija Exportar datos.

> **Note**
>
> Durante la exportación, seleccione Copiar el identificador del trabajo en el banner de la parte superior de la página. Puede utilizar el `JobID`para solicitar la exportación de las propiedades del trabajo mediante el AWS CLI. Para obtener más información, consulte Conseguir que el FHIR exporte propiedades laborales.

Conseguir que el FHIR exporte propiedades laborales

Se utiliza `DescribeFHIRExportJob` para obtener las propiedades de los trabajos de exportación desde un banco de HealthLake datos. Los siguientes menús proporcionan un procedimiento Consola de administración de AWS y ejemplos de código para AWS CLI y AWS SDKs. Para obtener más información, consulta `DescribeFHIRExportJob` en la AWS HealthLake Referencia de la API de .

> **Nota**
>
> HealthLake es compatible con la especificación R4 del FHIR para el intercambio de datos de atención médica. Por lo tanto, todos los datos de salud se exportan en formato FHIR R4.

Para describir un trabajo de exportación de FHIR

Elija un menú en función de sus preferencias de acceso a. AWS HealthLake

AWS CLI y SDKs

CLI

AWS CLI

Para describir un trabajo de exportación de FHIR

El siguiente `describe-fhir-export-job` ejemplo muestra cómo encontrar las propiedades de un trabajo de exportación FHIR en AWS HealthLake.

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

Salida:

```
{
    "ExportJobProperties": {
        "DataAccessRoleArn": "arn:aws:iam::(AWS Account ID):role/(Role Name)",
        "JobStatus": "IN_PROGRESS",
        "JobId": "9009813e9d69ba7cf79bcb3468780f16",
        "SubmitTime": "2024-11-20T11:31:46.672000-05:00",
        "EndTime": "2024-11-20T11:34:01.636000-05:00",
        "OutputDataConfig": {
            "S3Configuration": {
            "S3Uri": "s3://(Bucket Name)/(Prefix Name)/",
            "KmsKeyId": "arn:aws:kms:us-east-1:012345678910:key/d330e7fc-
b56c-4216-a250-f4c43ef46e83"
        }

        },
        "DatastoreId": "(Data store ID)"
    }
}
```

- Para obtener más información sobre la API, consulte Describe FHIRExport Job in AWS CLI Command Reference.

Python

SDK para Python (Boto3)

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

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

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


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

- Para obtener más información sobre la API, consulta Describe FHIRExport Job in AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

- Para obtener más información sobre la API, consulte Describe FHIRExport Job en el AWS SDK para la referencia de la API ABAP de SAP.

> **Ejemplo de disponibilidad**
>
> ¿No encuentra lo que necesita? Solicita un ejemplo de código mediante el enlace Enviar comentarios en la barra lateral derecha de esta página.

AWS Consola

> **Note**
>
> La información sobre los trabajos de exportación del FHIR no está disponible en la HealthLake consola. En su lugar, utilice el comando AWS CLI with `DescribeFHIRExportJob` para solicitar la exportación de propiedades del trabajo, como `JobStatus`. Para obtener más información, consulta el AWS CLI ejemplo de esta página.

Listado de trabajos de exportación del FHIR

Se usa `ListFHIRExportJobs` para enumerar los trabajos de exportación del FHIR para un almacén de HealthLake datos. Los siguientes menús proporcionan un procedimiento Consola de administración de AWS y ejemplos de código para AWS CLI y AWS SDKs. Para obtener más información, consulta `ListFHIRExportJobs` en la AWS HealthLake Referencia de la API de .

> **Nota**
>
> HealthLake es compatible con la especificación R4 del FHIR para el intercambio de datos de atención médica. Por lo tanto, todos los datos de salud se exportan en formato FHIR R4.

Para enumerar los trabajos de exportación del FHIR

Elija un menú en función de sus preferencias de acceso a AWS HealthLake.

AWS CLI y SDKs

CLI

AWS CLI

Para enumerar todos los trabajos de exportación de FHIR

En el siguiente ejemplo de `list-fhir-export-jobs`, se muestra cómo utilizar el comando para ver una lista de trabajos de exportación asociados a una cuenta.

```
aws healthlake list-fhir-export-jobs \
    --datastore-id (Data store ID) \
    --submitted-before (DATE like 2024-10-13T19:00:00Z)\
    --submitted-after (DATE like 2020-10-13T19:00:00Z )\
    --job-name "FHIR-EXPORT" \
    --job-status SUBMITTED  \
    --max-results (Integer between 1 and 500)
```

Salida:

```
{
    "ExportJobPropertiesList": [
        {
            "ExportJobProperties": {
                "OutputDataConfig": {
                    "S3Uri": "s3://(Bucket Name)/(Prefix Name)/",
                    "S3Configuration": {
                        "S3Uri": "s3://(Bucket Name)/(Prefix Name)/",
                        "KmsKeyId": "(KmsKey Id)"
                    }
                },
                "DataAccessRoleArn": "arn:aws:iam::(AWS Account ID):role/(Role
 Name)",
                "JobStatus": "COMPLETED",
                "JobId": "c145fbb27b192af392f8ce6e7838e34f",
                "JobName": "FHIR-EXPORT",
                "SubmitTime": "2024-11-20T11:31:46.672000-05:00",
                "EndTime": "2024-11-20T11:34:01.636000-05:00",
                "DatastoreId": "(Data store ID)"
            }
        }
    ]
```

```
}
```

- Para obtener más información sobre la API, consulte Listar FHIRExport trabajos en la referencia de AWS CLI comandos.

Python

SDK para Python (Boto3)

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

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


   def list_fhir_export_jobs(
       self,
       datastore_id: str,
       job_name: str = None,
       job_status: str = None,
       submitted_before: datetime = None,
       submitted_after: datetime = None,
   ) -> list[dict[str, any]]:
       """
       Lists HealthLake export jobs satisfying the conditions.
       :param datastore_id: The data store ID.
       :param job_name: The export job name.
       :param job_status: The export job status.
       :param submitted_before: The export job submitted before the specified
date.
       :param submitted_after: The export job submitted after the specified
date.
       :return: A list of export jobs.
       """
       try:
```

```
            parameters = {"DatastoreId": datastore_id}
            if job_name is not None:
                parameters["JobName"] = job_name
            if job_status is not None:
                parameters["JobStatus"] = job_status
            if submitted_before is not None:
                parameters["SubmittedBefore"] = submitted_before
            if submitted_after is not None:
                parameters["SubmittedAfter"] = submitted_after
            next_token = None
            jobs = []
            # Loop through paginated results.
            while True:
                if next_token is not None:
                    parameters["NextToken"] = next_token
                response =
self.health_lake_client.list_fhir_export_jobs(**parameters)
                jobs.extend(response["ExportJobPropertiesList"])
                if "NextToken" in response:
                    next_token = response["NextToken"]
                else:
                    break
            return jobs
        except ClientError as err:
            logger.exception(
                "Couldn't list export jobs. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Para obtener más información sobre la API, consulta la sección Lista de FHIRExport trabajos en la referencia de la API del AWS SDK for Python (Boto3).

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

- Para obtener más información sobre la API, consulte Listar FHIRExport trabajos en el AWS SDK para obtener una referencia sobre la API ABAP de SAP.

Ejemplo de disponibilidad

¿No encuentra lo que necesita? Solicita un ejemplo de código mediante el enlace Enviar comentarios en la barra lateral derecha de esta página.

AWS Consola

Note

La información sobre los trabajos de exportación del FHIR no está disponible en la HealthLake consola. En su lugar, utilice AWS CLI with `ListFHIRExportJobs` para enumerar todos los trabajos de exportación del FHIR. Para obtener más información, consulte el AWS CLI ejemplo de esta página.

Ejemplos de código para HealthLake usar AWS SDKs

Los siguientes ejemplos de código muestran cómo usarlo HealthLake con un kit de desarrollo de AWS software (SDK).

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, consulteHealthLake Utilización con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Ejemplos de código

- Ejemplos básicos de HealthLake uso AWS SDKs
  - Acciones para HealthLake usar AWS SDKs
    - Úselo CreateFHIRDatastore con un AWS SDK o CLI
    - Úselo DeleteFHIRDatastore con un AWS SDK o CLI
    - Úselo DescribeFHIRDatastore con un AWS SDK o CLI
    - Úselo DescribeFHIRExportJob con un AWS SDK o CLI
    - Úselo DescribeFHIRImportJob con un AWS SDK o CLI
    - Úselo ListFHIRDatastores con un AWS SDK o CLI
    - Úselo ListFHIRExportJobs con un AWS SDK o CLI
    - Úselo ListFHIRImportJobs con un AWS SDK o CLI
    - Úselo ListTagsForResource con un AWS SDK o CLI
    - Úselo StartFHIRExportJob con un AWS SDK o CLI
    - Úselo StartFHIRImportJob con un AWS SDK o CLI
    - Úselo TagResource con un AWS SDK o CLI
    - Úselo UntagResource con un AWS SDK o CLI

Ejemplos básicos de HealthLake uso AWS SDKs

Los siguientes ejemplos de código muestran cómo utilizar los conceptos básicos de AWS HealthLake with AWS SDKs.

Ejemplos

- Acciones para HealthLake usar AWS SDKs
  - Úselo CreateFHIRDatastore con un AWS SDK o CLI
  - Úselo DeleteFHIRDatastore con un AWS SDK o CLI
  - Úselo DescribeFHIRDatastore con un AWS SDK o CLI
  - Úselo DescribeFHIRExportJob con un AWS SDK o CLI
  - Úselo DescribeFHIRImportJob con un AWS SDK o CLI
  - Úselo ListFHIRDatastores con un AWS SDK o CLI
  - Úselo ListFHIRExportJobs con un AWS SDK o CLI
  - Úselo ListFHIRImportJobs con un AWS SDK o CLI
  - Úselo ListTagsForResource con un AWS SDK o CLI
  - Úselo StartFHIRExportJob con un AWS SDK o CLI
  - Úselo StartFHIRImportJob con un AWS SDK o CLI
  - Úselo TagResource con un AWS SDK o CLI
  - Úselo UntagResource con un AWS SDK o CLI

Acciones para HealthLake usar AWS SDKs

Los siguientes ejemplos de código muestran cómo realizar HealthLake acciones individuales con AWS SDKs. Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones para configurar y ejecutar el código.

Los siguientes ejemplos incluyen solo las acciones que se utilizan con mayor frecuencia. Para ver una lista completa, consulte la Referencia de la API de AWS HealthLake.

Ejemplos

- Úselo CreateFHIRDatastore con un AWS SDK o CLI
- Úselo DeleteFHIRDatastore con un AWS SDK o CLI
- Úselo DescribeFHIRDatastore con un AWS SDK o CLI
- Úselo DescribeFHIRExportJob con un AWS SDK o CLI
- Úselo DescribeFHIRImportJob con un AWS SDK o CLI
- Úselo ListFHIRDatastores con un AWS SDK o CLI
- Úselo ListFHIRExportJobs con un AWS SDK o CLI

- Úselo ListFHIRImportJobs con un AWS SDK o CLI
- Úselo ListTagsForResource con un AWS SDK o CLI
- Úselo StartFHIRExportJob con un AWS SDK o CLI
- Úselo StartFHIRImportJob con un AWS SDK o CLI
- Úselo TagResource con un AWS SDK o CLI
- Úselo UntagResource con un AWS SDK o CLI

Úselo **`CreateFHIRDatastore`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `CreateFHIRDatastore`.

CLI

AWS CLI

Ejemplo 1: Crear un banco de datos compatible con SIGv4 HealthLake

El siguiente `create-fhir-datastore` ejemplo muestra cómo crear un nuevo banco de datos en. AWS HealthLake

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

Salida:

```
{
    "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/datastore/
(Data store ID)/r4/",
    "DatastoreArn": "arn:aws:healthlake:us-east-1:(AWS Account ID):datastore/
(Data store ID)",
    "DatastoreStatus": "CREATING",
    "DatastoreId": "(Data store ID)"
}
```

Ejemplo 2: Crear un almacén de datos SMART en un almacén de datos compatible con FHIR HealthLake

El siguiente `create-fhir-datastore` ejemplo muestra cómo crear un nuevo almacén de datos compatible con SMART on FHIR en. AWS HealthLake

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

Contenido de `identity_provider_configuration.json`:

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

Salida:

```
{
    "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/datastore/
(Data store ID)/r4/",
    "DatastoreArn": "arn:aws:healthlake:us-east-1:(AWS Account ID):datastore/
(Data store ID)",
    "DatastoreStatus": "CREATING",
    "DatastoreId": "(Data store ID)"
}
```

- Para obtener más información sobre la API, consulte Create FHIRDatastore in AWS CLI Command Reference.

Python

SDK para Python (Boto3)

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

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


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

       :param datastore_name: The name of the data store.
       :param sse_configuration: The server-side encryption configuration for a
SMART on FHIR-enabled data store.
       :param identity_provider_configuration: The identity provider
configuration for a SMART on FHIR-enabled data store.
       :return: A dictionary containing the data store information.
       """
       try:
```

```
            parameters = {"DatastoreName": datastore_name,
 "DatastoreTypeVersion": "R4"}
            if (
                sse_configuration is not None
                and identity_provider_configuration is not None
            ):
                # Creating a SMART on FHIR-enabled data store
                parameters["SseConfiguration"] = sse_configuration
                parameters[
                    "IdentityProviderConfiguration"
                ] = identity_provider_configuration

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

En el siguiente código se muestra un ejemplo de parámetros para un almacén de datos compatible con SMART on FHIR HealthLake .

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

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

- Para obtener más información sobre la API, consulta la referencia de la API Create FHIRDatastore in AWS SDK for Python (Boto3).

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

- Para obtener más información sobre la API, consulte Crear FHIRDatastore en el AWS SDK para la referencia sobre la API ABAP de SAP.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteHealthLake Utilización con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`DeleteFHIRDatastore`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `DeleteFHIRDatastore`.

CLI

AWS CLI

Eliminación de un almacén de datos de FHIR

El siguiente `delete-fhir-datastore` ejemplo muestra cómo eliminar un banco de datos y todo su contenido AWS HealthLake.

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

Salida:

```
{
    "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/datastore/
(Data store ID)/r4/",
    "DatastoreArn": "arn:aws:healthlake:us-east-1:(AWS Account ID):datastore/
(Data store ID)",
    "DatastoreStatus": "DELETING",
    "DatastoreId": "(Data store ID)"
}
```

- Para obtener más información sobre la API, consulte Eliminar en la referencia de comandos. FHIRDatastore AWS CLI

Python

SDK para Python (Boto3)

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

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


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

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

- Para obtener más información sobre la API, consulta Eliminar FHIRDatastore en la referencia de la API del AWS SDK for Python (Boto3).

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

- Para obtener más información sobre la API, consulta Eliminar FHIRDatastore en el AWS SDK para ver la referencia sobre la API ABAP de SAP.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteHealthLake Utilización con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`DescribeFHIRDatastore`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `DescribeFHIRDatastore`.

CLI

AWS CLI

Descripción de un almacén de datos de FHIR

El siguiente `describe-fhir-datastore` ejemplo muestra cómo encontrar las propiedades de un almacén de datos en AWS HealthLake.

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

Salida:

```
{
    "DatastoreProperties": {
        "PreloadDataConfig": {
            "PreloadDataType": "SYNTHEA"
        },
        "SseConfiguration": {
            "KmsEncryptionConfig": {
                "CmkType": "CUSTOMER_MANAGED_KMS_KEY",
                "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
            }
        },
        "DatastoreName": "Demo",
        "DatastoreArn": "arn:aws:healthlake:us-east-1:<AWS Account ID>:datastore/
<Data store ID>",
        "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/
datastore/<Data store ID>/r4/",
        "DatastoreStatus": "ACTIVE",
        "DatastoreTypeVersion": "R4",
        "CreatedAt": 1603761064.881,
        "DatastoreId": "<Data store ID>",
        "IdentityProviderConfiguration": {
            "AuthorizationStrategy": "AWS_AUTH",
            "FineGrainedAuthorizationEnabled": false
```

```
        }
    }
}
```

- Para obtener más información sobre la API, consulte la sección Describe FHIRDatastore in AWS CLI Command Reference.

Python

SDK para Python (Boto3)

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

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


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

- Para obtener más información sobre la API, consulta la descripción FHIRDatastore en la referencia de la API del AWS SDK for Python (Boto3).

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

```
        lv_error = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
    ENDTRY.
```

- Para obtener más información sobre la API, consulta la sección Describe FHIRDatastore en el AWS SDK para ver la referencia sobre la API ABAP de SAP.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteHealthLake Utilización con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`DescribeFHIRExportJob`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `DescribeFHIRExportJob`.

CLI

AWS CLI

Para describir un trabajo de exportación de FHIR

El siguiente `describe-fhir-export-job` ejemplo muestra cómo encontrar las propiedades de un trabajo de exportación FHIR en AWS HealthLake.

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

Salida:

```
{
    "ExportJobProperties": {
        "DataAccessRoleArn": "arn:aws:iam::(AWS Account ID):role/(Role Name)",
        "JobStatus": "IN_PROGRESS",
        "JobId": "9009813e9d69ba7cf79bcb3468780f16",
        "SubmitTime": "2024-11-20T11:31:46.672000-05:00",
        "EndTime": "2024-11-20T11:34:01.636000-05:00",
        "OutputDataConfig": {
            "S3Configuration": {
```

```
            "S3Uri": "s3://(Bucket Name)/(Prefix Name)/",
            "KmsKeyId": "arn:aws:kms:us-east-1:012345678910:key/d330e7fc-
b56c-4216-a250-f4c43ef46e83"
        }

        },
        "DatastoreId": "(Data store ID)"
    }
}
```

- Para obtener más información sobre la API, consulte Describe FHIRExport Job in AWS CLI Command Reference.

Python

SDK para Python (Boto3)

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

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


    def describe_fhir_export_job(
        self, datastore_id: str, job_id: str
    ) -> dict[str, any]:
        """
        Describes a HealthLake export job.
        :param datastore_id: The data store ID.
        :param job_id: The export job ID.
        :return: The export job description.
        """
        try:
            response = self.health_lake_client.describe_fhir_export_job(
                DatastoreId=datastore_id, JobId=job_id
```

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

- Para obtener más información sobre la API, consulta Describe FHIRExport Job in AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
TRY.
    " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
    " iv_job_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
    oo_result = lo_hll->describefhirexportjob(
      iv_datastoreid = iv_datastore_id
      iv_jobid = iv_job_id
    ).
    DATA(lo_export_job_properties) = oo_result->get_exportjobproperties( ).
    IF lo_export_job_properties IS BOUND.
      DATA(lv_job_status) = lo_export_job_properties->get_jobstatus( ).
```

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

- Para obtener más información sobre la API, consulte Describe FHIRExport Job en el AWS SDK para la referencia de la API ABAP de SAP.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteHealthLake Utilización con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`DescribeFHIRImportJob`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `DescribeFHIRImportJob`.

CLI

AWS CLI

Para describir un trabajo de importación de FHIR

El siguiente `describe-fhir-import-job` ejemplo muestra cómo aprender las propiedades de un trabajo de importación del FHIR utilizando AWS HealthLake.

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

Salida:

```
{
```

```
    "ImportJobProperties": {
    "InputDataConfig": {
        "S3Uri": "s3://(Bucket Name)/(Prefix Name)/"
        { "arrayitem2": 2 }
    },
    "DataAccessRoleArn": "arn:aws:iam::(AWS Account ID):role/(Role Name)",
    "JobStatus": "COMPLETED",
    "JobId": "c145fbb27b192af392f8ce6e7838e34f",
    "SubmitTime": 1606272542.161,
    "EndTime": 1606272609.497,
    "DatastoreId": "(Data store ID)"
    }
}
```

- Para obtener más información sobre la API, consulte Describe FHIRImport Job in AWS CLI Command Reference.

Python

SDK para Python (Boto3)

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

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


    def describe_fhir_import_job(
        self, datastore_id: str, job_id: str
    ) -> dict[str, any]:
        """
        Describes a HealthLake import job.
        :param datastore_id: The data store ID.
        :param job_id: The import job ID.
        :return: The import job description.
```

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

- Para obtener más información sobre la API, consulta Describe FHIRImport Job in AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        " iv_job_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        oo_result = lo_hll->describefhirimportjob(
          iv_datastoreid = iv_datastore_id
          iv_jobid = iv_job_id
```

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

- Para obtener más información sobre la API, consulte Describe FHIRImport Job en el AWS SDK para la referencia de la API ABAP de SAP.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteHealthLake Utilización con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`ListFHIRDatastores`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `ListFHIRDatastores`.

CLI

AWS CLI

Muestra de los almacenes de datos de FHIR

El siguiente `list-fhir-datastores` ejemplo muestra cómo utilizar el comando y cómo los usuarios pueden filtrar los resultados en función del estado del almacén de datos AWS HealthLake.

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

Salida:

```
{
    "DatastorePropertiesList": [
    {
        "PreloadDataConfig": {
            "PreloadDataType": "SYNTHEA"
        },
        "SseConfiguration": {
            "KmsEncryptionConfig": {
                "CmkType": "CUSTOMER_MANAGED_KMS_KEY",
                "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
            }
        },
        "DatastoreName": "Demo",
        "DatastoreArn": "arn:aws:healthlake:us-east-1:<AWS Account ID>:datastore/
<Data store ID>",
        "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/
datastore/<Data store ID>/r4/",
        "DatastoreStatus": "ACTIVE",
        "DatastoreTypeVersion": "R4",
        "CreatedAt": 1603761064.881,
        "DatastoreId": "<Data store ID>",
        "IdentityProviderConfiguration": {
            "AuthorizationStrategy": "AWS_AUTH",
            "FineGrainedAuthorizationEnabled": false
        }
    }
    ]
}
```

- Para obtener más información sobre la API, consulte la lista FHIRDatastores en la referencia de AWS CLI comandos.

Python

SDK para Python (Boto3)

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

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

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


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

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

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

- Para obtener más información sobre las API, consulta la lista FHIRDatastores en la referencia de API de AWS SDK for Python (Boto3).

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

- Para obtener más información sobre las API, consulta la lista FHIRDatastores en el AWS SDK para obtener información sobre las API ABAP de SAP.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteHealthLake Utilización con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`ListFHIRExportJobs`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `ListFHIRExportJobs`.

CLI

AWS CLI

Para enumerar todos los trabajos de exportación de FHIR

En el siguiente ejemplo de `list-fhir-export-jobs`, se muestra cómo utilizar el comando para ver una lista de trabajos de exportación asociados a una cuenta.

```
aws healthlake list-fhir-export-jobs \
    --datastore-id (Data store ID) \
    --submitted-before (DATE like 2024-10-13T19:00:00Z)\
    --submitted-after (DATE like 2020-10-13T19:00:00Z )\
    --job-name "FHIR-EXPORT" \
    --job-status SUBMITTED  \
    --max-results (Integer between 1 and 500)
```

Salida:

```
{
    "ExportJobPropertiesList": [
        {
            "ExportJobProperties": {
                "OutputDataConfig": {
                    "S3Uri": "s3://(Bucket Name)/(Prefix Name)/",
                    "S3Configuration": {
                        "S3Uri": "s3://(Bucket Name)/(Prefix Name)/",
                        "KmsKeyId": "(KmsKey Id)"
                    }
                },
                "DataAccessRoleArn": "arn:aws:iam::(AWS Account ID):role/(Role
 Name)",
                "JobStatus": "COMPLETED",
                "JobId": "c145fbb27b192af392f8ce6e7838e34f",
                "JobName": "FHIR-EXPORT",
                "SubmitTime": "2024-11-20T11:31:46.672000-05:00",
                "EndTime": "2024-11-20T11:34:01.636000-05:00",
                "DatastoreId": "(Data store ID)"
            }
```

```
        }
    ]
}
```

- Para obtener más información sobre la API, consulte Listar FHIRExport trabajos en la referencia de AWS CLI comandos.

Python

SDK para Python (Boto3)

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

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


    def list_fhir_export_jobs(
        self,
        datastore_id: str,
        job_name: str = None,
        job_status: str = None,
        submitted_before: datetime = None,
        submitted_after: datetime = None,
    ) -> list[dict[str, any]]:
        """
        Lists HealthLake export jobs satisfying the conditions.
        :param datastore_id: The data store ID.
        :param job_name: The export job name.
        :param job_status: The export job status.
        :param submitted_before: The export job submitted before the specified
date.
        :param submitted_after: The export job submitted after the specified
date.
        :return: A list of export jobs.
```

```
        """
        try:
            parameters = {"DatastoreId": datastore_id}
            if job_name is not None:
                parameters["JobName"] = job_name
            if job_status is not None:
                parameters["JobStatus"] = job_status
            if submitted_before is not None:
                parameters["SubmittedBefore"] = submitted_before
            if submitted_after is not None:
                parameters["SubmittedAfter"] = submitted_after
            next_token = None
            jobs = []
            # Loop through paginated results.
            while True:
                if next_token is not None:
                    parameters["NextToken"] = next_token
                response =
self.health_lake_client.list_fhir_export_jobs(**parameters)
                jobs.extend(response["ExportJobPropertiesList"])
                if "NextToken" in response:
                    next_token = response["NextToken"]
                else:
                    break
            return jobs
        except ClientError as err:
            logger.exception(
                "Couldn't list export jobs. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Para obtener más información sobre la API, consulta la sección Lista de FHIRExport trabajos en la referencia de la API del AWS SDK for Python (Boto3).

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

- Para obtener más información sobre la API, consulte Listar FHIRExport trabajos en el AWS SDK para obtener una referencia sobre la API ABAP de SAP.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteHealthLake Utilización con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`ListFHIRImportJobs`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `ListFHIRImportJobs`.

CLI

AWS CLI

Para enumerar todos los trabajos de importación de FHIR

En el siguiente ejemplo de `list-fhir-import-jobs`, se muestra cómo utilizar el comando para ver una lista de todos los trabajos de importación asociados a una cuenta.

```
aws healthlake list-fhir-import-jobs \
    --datastore-id (Data store ID) \
    --submitted-before (DATE like 2024-10-13T19:00:00Z) \
    --submitted-after (DATE like 2020-10-13T19:00:00Z ) \
    --job-name "FHIR-IMPORT" \
    --job-status SUBMITTED  \
    -max-results (Integer between 1 and 500)
```

Salida:

```
{
    "ImportJobPropertiesList": [
        {
            "JobId": "c0fddbf76f238297632d4aebdbfc9ddf",
            "JobStatus": "COMPLETED",
            "SubmitTime": "2024-11-20T10:08:46.813000-05:00",
            "EndTime": "2024-11-20T10:10:09.093000-05:00",
            "DatastoreId": "(Data store ID)",
            "InputDataConfig": {
                "S3Uri": "s3://(Bucket Name)/(Prefix Name)/"
            },
            "JobOutputDataConfig": {
                "S3Configuration": {
                    "S3Uri": "s3://(Bucket Name)/
import/6407b9ae4c2def3cb6f1a46a0c599ec0-FHIR_IMPORT-
c0fddbf76f238297632d4aebdbfc9ddf/",
```

```
                    "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/b7f645cb-
e564-4981-8672-9e012d1ff1a0"
                }
            },
            "JobProgressReport": {
                "TotalNumberOfScannedFiles": 1,
                "TotalSizeOfScannedFilesInMB": 0.001798,
                "TotalNumberOfImportedFiles": 1,
                "TotalNumberOfResourcesScanned": 1,
                "TotalNumberOfResourcesImported": 1,
                "TotalNumberOfResourcesWithCustomerError": 0,
                "TotalNumberOfFilesReadWithCustomerError": 0,
                "Throughput": 0.0
            },
            "DataAccessRoleArn": "arn:aws:iam::(AWS Account ID):role/(Role Name)"
        }
    ]
}
```

- Para obtener más información sobre la API, consulte Listar FHIRImport trabajos en la referencia de AWS CLI comandos.

Python

SDK para Python (Boto3)

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

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


    def list_fhir_import_jobs(
        self,
        datastore_id: str,
```

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

```
            )
            raise
```

- Para obtener más información sobre la API, consulta la sección Lista de FHIRImport trabajos en la referencia de la API del AWS SDK for Python (Boto3).

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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

- Para obtener más información sobre la API, consulte Listar FHIRImport trabajos en el AWS SDK para obtener una referencia sobre la API ABAP de SAP.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteHealthLake Utilización con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`ListTagsForResource`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `ListTagsForResource`.

CLI

AWS CLI

Muestra de las etiquetas para un almacén de datos

En el siguiente ejemplo de `list-tags-for-resource`, se muestran las etiquetas asociadas al almacén de datos especificado:

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

Salida:

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

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

- Para obtener más información sobre la API, consulte ListTagsForResourcela Referencia de AWS CLI comandos.

Python

SDK para Python (Boto3)

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

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


    def list_tags_for_resource(self, resource_arn: str) -> dict[str, str]:
        """
        Lists the tags for a HealthLake resource.
        :param resource_arn: The resource ARN.
        :return: The tags for the resource.
        """
        try:
            response = self.health_lake_client.list_tags_for_resource(
                ResourceARN=resource_arn
            )
            return response["Tags"]
        except ClientError as err:
            logger.exception(
                "Couldn't list tags for resource %s. Here's why %s",
                resource_arn,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obtener más información sobre la API, consulta ListTagsForResourcela AWS Referencia de API de SDK for Python (Boto3).

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

```
    ENDTRY.
```

- Para obtener más información sobre la API, consulte ListTagsForResourcela referencia sobre la API ABAP del AWS SDK para SAP.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteHealthLake Utilización con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`StartFHIRExportJob`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `StartFHIRExportJob`.

CLI

AWS CLI

Para iniciar un trabajo de exportación de FHIR

El siguiente `start-fhir-export-job` ejemplo muestra cómo iniciar un trabajo de exportación del FHIR utilizando AWS HealthLake.

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

Salida:

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

- Para obtener información sobre la API, consulte Start FHIRExport Job in AWS CLI Command Reference.

Python

SDK para Python (Boto3)

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

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


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

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

- Para obtener más información sobre la API, consulta la referencia de la API Start FHIRExport Job in AWS SDK for Python (Boto3).

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

```
              iv_s3uri = iv_output_s3_uri
              iv_kmskeyid = iv_kms_key_id
            )
          )
          iv_dataaccessrolearn = iv_data_access_role_arn
          iv_datastoreid = iv_datastore_id
        ).
        DATA(lv_job_id) = oo_result->get_jobid( ).
        MESSAGE |Export job started with ID { lv_job_id }.| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllthrottlingex INTO DATA(lo_throttling_ex).
        lv_error = |Throttling error: { lo_throttling_ex->av_err_code }-
{ lo_throttling_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_throttling_ex.
      CATCH /aws1/cx_hllaccessdeniedex INTO DATA(lo_access_ex).
        lv_error = |Access denied: { lo_access_ex->av_err_code }-{ lo_access_ex-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_access_ex.
    ENDTRY.
```

- Para obtener más información sobre la API, consulte Start FHIRExport Job in AWS SDK para obtener una referencia sobre la API ABAP de SAP.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteHealthLake Utilización con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`StartFHIRImportJob`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `StartFHIRImportJob`.

CLI

AWS CLI

Para iniciar un trabajo de importación de FHIR

El siguiente `start-fhir-import-job` ejemplo muestra cómo iniciar un trabajo de importación del FHIR utilizando AWS HealthLake.

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

Salida:

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

- Para obtener información sobre la API, consulte Start FHIRImport Job in AWS CLI Command Reference.

Python

SDK para Python (Boto3)

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

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

```
        return cls(health_lake_client)


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

- Para obtener más información sobre la API, consulta la referencia de la API Start FHIRImport Job in AWS SDK for Python (Boto3).

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    TRY.
        " iv_job_name = 'MyImportJob'
        " iv_input_s3_uri = 's3://my-bucket/import/data.ndjson'
        " iv_job_output_s3_uri = 's3://my-bucket/import/output/'
        " iv_kms_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012'
        " iv_data_access_role_arn = 'arn:aws:iam::123456789012:role/
HealthLakeImportRole'
        oo_result = lo_hll->startfhirimportjob(
          iv_jobname = iv_job_name
          io_inputdataconfig = NEW /aws1/cl_hllinputdataconfig( iv_s3uri =
 iv_input_s3_uri )
          io_joboutputdataconfig = NEW /aws1/cl_hlloutputdataconfig(
            io_s3configuration = NEW /aws1/cl_hlls3configuration(
              iv_s3uri = iv_job_output_s3_uri
              iv_kmskeyid = iv_kms_key_id
            )
          )
          iv_dataaccessrolearn = iv_data_access_role_arn
          iv_datastoreid = iv_datastore_id
```

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

- Para obtener más información sobre la API, consulte Start FHIRImport Job in AWS SDK para obtener una referencia sobre la API ABAP de SAP.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteHealthLake Utilización con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`TagResource`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `TagResource`.

CLI

AWS CLI

Agregación de una etiqueta al almacén de datos

En el siguiente ejemplo de `tag-resource`, se muestra cómo agregar una etiqueta a un almacén de datos.

```
aws healthlake tag-resource \
```

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

Este comando no genera ninguna salida.

.

- Para obtener más información sobre la API, consulte TagResourcela Referencia de AWS CLI comandos.

Python

SDK para Python (Boto3)

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

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


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

```
            )
            raise
```

- Para obtener más información sobre la API, consulta TagResourcela AWS Referencia de API de SDK for Python (Boto3).

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

```
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- Para obtener más información sobre la API, consulte TagResourcela referencia sobre la API ABAP del AWS SDK para SAP.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteHealthLake Utilización con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`UntagResource`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `UntagResource`.

CLI

AWS CLI

Eliminación de etiquetas de un almacén de datos.

En el siguiente ejemplo de `untag-resource`, se muestra cómo eliminar etiquetas de un almacén de datos.

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

Este comando no genera ninguna salida.

- Para obtener más información sobre la API, consulte UntagResourcela Referencia de AWS CLI comandos.

Python

SDK para Python (Boto3)

```
    @classmethod
```

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

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


    def untag_resource(self, resource_arn: str, tag_keys: list[str]) -> None:
        """
        Untags a HealthLake resource.
        :param resource_arn: The resource ARN.
        :param tag_keys: The tag keys to remove from the resource.
        """
        try:
            self.health_lake_client.untag_resource(
                ResourceARN=resource_arn, TagKeys=tag_keys
            )
        except ClientError as err:
            logger.exception(
                "Couldn't untag resource %s. Here's why %s",
                resource_arn,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obtener más información sobre la API, consulta UntagResourcela AWS Referencia de API de SDK for Python (Boto3).

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

- Para obtener más información sobre la API, consulte UntagResourcela referencia sobre la API ABAP del AWS SDK para SAP.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteHealthLake Utilización con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Integrando AWS HealthLake

Los siguientes AWS servicios se integran directamente AWS HealthLake para permitir el procesamiento integrado del lenguaje natural, las consultas SQL y el almacenamiento de datos.

- Amazon Comprehend Medical es un servicio de procesamiento del lenguaje natural (NLP) que cumple con los requisitos de la HIPAA y que utiliza bibliotecas de aprendizaje automático para extraer datos de salud significativos de textos médicos no estructurados. HealthLake Para obtener más información, consulte la Guía para desarrolladores de Amazon Comprehend Medical.
- Amazon Athena es un servicio de consultas interactivo que le permite analizar HealthLake los datos directamente en los depósitos de Amazon Simple Storage Service (Amazon S3) mediante SQL estándar. Para obtener más información, consulte la Guía para desarrolladores de Amazon Athena.

Temas

- Procesamiento de lenguaje natural (PNL) integrado para HealthLake
- Consulta de HealthLake datos con Amazon Athena

Procesamiento de lenguaje natural (PNL) integrado para HealthLake

AWS HealthLake proporciona bibliotecas integradas de procesamiento del lenguaje natural (NLP) para analizar, identificar y mapear los datos no estructurados almacenados en los tipos de recursos del FHIR. `DocumentReference`

> **Importante**
>
> La PNL integrada está desactivada de forma HealthLake predeterminada. Para activarlo, envíe un caso de soporte utilizando AWS Support Center Console. Para crear tu caso, inicia sesión en tu cuenta Cuenta de AWS y selecciona Crear caso.

HealthLake La PNL integrada funciona denominando a las acciones Amazon Comprehend Medical `DetectEntities-V2` y `InferICD10-CM` `InferRxNorm` API. Las acciones anexan sus resultados

como una extensión del recurso. `DocumentReference` Cuando las acciones de la API Amazon Comprehend Medical detectan rasgos que `SIGN` son `SYMPTOM` `DIAGNOSIS` y/o generan un recurso `Linkage`FHIR. Los `Observation` recursos nuevos `Condition` y los recursos se crean a partir de entidades identificadas con las características de `SIGNSYMPTOM`, o, `DIAGNOSIS` y se vinculan al documento fuente mediante el `Linkage` recurso.

> **Note**
>
> Si bien se admiten `GET` solicitudes de recursos del FHIR generados por la PNL HealthLake integrada, la `search` funcionalidad de la API del FHIR no lo es. Para obtener más información sobre la búsqueda de extensiones de PNL mediante HealthLake la integración con Athena, consulte. Índice y consulta de SQL

Contenido

- HealthLake Bibliotecas de PNL integradas
- Uso de las interacciones de la API REST del FHIR
- Parámetros de búsqueda para HealthLake la PNL integrada
- HealthLake solicitudes de ejemplo de PNL integradas

HealthLake Bibliotecas de PNL integradas

HealthLake deduce los datos encontrados en el tipo de `DocumentReference` recurso mediante las bibliotecas de Amazon Comprehend Medical. La API Amazon Comprehend Medical `DetectEntities-V2` opera `InferICD10-CM` `InferRxNorm` y detecta afecciones médicas como rasgos. Cada operación proporciona información diferente.

> **Lenguajes compatibles**
>
> Las operaciones de la API Amazon Comprehend Medical solo detectan entidades médicas en textos en inglés.

- DetectEntities-V2: inspecciona el texto clínico en busca de diversas entidades médicas y devuelve información específica sobre ellas, como la categoría de la entidad, la ubicación y el puntaje de confianza.

- Inferir ICD1 0-CM: detecta las afecciones médicas en el registro de un paciente como entidades y vincula esas entidades con los identificadores de conceptos normalizados de la base de conocimientos ICD-10-CM del Centro Nacional de Estadísticas de Salud de los CDC, con la autorización de la Organización Mundial de la Salud (OMS).
- InferRxNorm: Detecta los medicamentos como entidades incluidas en la historia clínica de un paciente y los vincula con los identificadores conceptuales normalizados de la base de datos de la Biblioteca Nacional de Medicina. RxNorm

Las características compatibles para cada operación de la API son `SIGNSYMPTOM`, y. `DIAGNOSIS` Si se detectan rasgos, se agregan como extensiones compatibles con la FHIR a diferentes ubicaciones del almacén de datos. HealthLake

Ubicaciones donde se agregan las extensiones.

- `DocumentReference`: Los resultados de las operaciones de la API Amazon Comprehend Medical se añaden como `extension` a cada documento que se encuentra dentro `DocumentReference` del tipo de recurso. Los resultados de la extensión se dividen en dos grupos. Puede encontrarlos en los resultados en función de sus`URL`.
  - `http://healthlake.amazonaws.com/system-generated-resources/`
    - Estos son tipos de recursos que han sido creados o añadidos por HealthLake.
  - `http://healthlake.amazonaws.com/aws-cm/`
    - Donde el resultado bruto de las operaciones de la API Amazon Comprehend Medical se añade a HealthLake su almacén de datos.
- `Linkage`: Este tipo de recurso se agrega o se crea como resultado de la PNL integrada. Una `GET` solicitud sobre un objeto específico `Linkage` devuelve una lista de recursos enlazados. Para identificar si a `Linkage` fue agregado por HealthLake, busca el par `"tag": [{"display": "SYSTEM_GENERATED"}]` clave-valor agregado. Para obtener más información sobre las especificaciones del FHIR para Linkage, consulte la documentación del FHIR `Linkage`R4.
- Tipos de recursos del FHIR generados como resultado de las operaciones de Amazon Comprehend Medical.
  - `Observation`: incluye los resultados de las `DetectEntities-V2` acciones de la API Amazon Comprehend Medical `InferICD10-CM` y cuándo las características `SIGN` son `SYMPTOM` o.
  - `Condition`: incluye los resultados de las `DetectEntities-V2` acciones de la API Amazon Comprehend Medical `InferICD10-CM` y cuándo se encuentra la característica`DIAGNOSIS`.

- `MedicationStatement`: incluye los resultados de las acciones de la API `InferRxNorm` Amazon Comprehend Medical.

Uso de las interacciones de la API REST del FHIR

De forma predeterminada, las características detectadas por las operaciones de la API Amazon Comprehend Medical no se devuelven al realizar `GET` una solicitud. Para ver los resultados de las operaciones de PNL integradas, debe especificar un recurso conocido `ID` para los siguientes tipos de recursos del FHIR.

- `Linkage`
- `Observation`
- `Condition`
- `MedicationStatement`

Los resultados de las acciones de PNL HealthLake integradas ajenas al tipo de `DocumentReference` recurso están disponibles mediante una `GET` solicitud en la que `ID` se sabe que lo especificado contiene resultados de las operaciones de la API Amazon Comprehend Medical.

Parámetros de búsqueda para HealthLake la PNL integrada

En la siguiente tabla se enumeran los atributos que se pueden buscar para la HealthLake PNL integrada.

Parámetros de búsqueda para la PNL HealthLake

Parámetros de búsqueda	Busca coincidencias para
Detectar ntities-entity-category	Categoría de entidad dentro de la DetectEntities subextensión de la AWS extensión CM
Detecta ntities-entity-text	Texto de entidad dentro de la DetectEntities subextensión dentro de la AWS extensión CM
Detecta ntities-entity-type	Tipo de entidad dentro de la DetectEntities subextens ión dentro de la AWS extensión CM

Parámetros de búsqueda	Busca coincidencias para
Detecta ntities-entity-score	Entity Score dentro de la DetectEntities subextensión de la AWS extensión CM
infer-icd10 cm-entity-text	Texto de entidad dentro de la subextensión Infer ICD1 0CM dentro de la extensión CM AWS
infer-icd10 cm-entity-score	Puntuación de entidad dentro de la subextensión Infer ICD1 0CM dentro de la extensión CM AWS
infer-icd10 cm-entity-concept-code	Código conceptual de entidad dentro de la subextensión Infer ICD1 0CM dentro de la extensión CM AWS
infer-icd10 cm-entity-concept-description	Descripción del concepto de entidad dentro de la subextensión Infer ICD1 0CM dentro de la extensión CM AWS
infer-icd10 cm-entity-concept-score	Puntuación del concepto de entidad dentro de la subextensión Infer ICD1 0CM dentro de la extensión CM AWS
infer-rxnorm-entity-score	Puntuación de entidad dentro de la InferRxNorm subextensión de la extensión CM AWS
infer-rxnorm-entity-text	Texto de entidad dentro de la InferRxNorm subextensión dentro de la AWS extensión CM
infer-rxnorm-entity-concept-código	Código conceptual de entidad dentro de la InferRxNorm subextensión de la AWS extensión CM
infer-rxnorm-entity-concept-descripción	Descripción del concepto de entidad dentro de la InferRxNorm subextensión de la AWS extensión CM
infer-rxnorm-entity-concept-puntuación	Entity Concept Score dentro de la InferRxNorm subextensión de la AWS extensión CM

HealthLake proporciona una búsqueda especial para que coincidan con los criterios en los que la misma entidad `EntityCategory` sea parte de ella `EntityText` y sea parte de ella. En la siguiente tabla se describen los parámetros de búsqueda especiales que admite HealthLake.

Parámetros de búsqueda

Parámetros de búsqueda	Coincidencias devueltas
Detecta ntities-entity-text-category	Si hay al menos una entidad en la DetectEntities subextensión que coincida con EntityText y EntityCategory.
Detecta ntities-entity-type-score	Si hay al menos una entidad en la DetectEnt ities subextensión que coincida con EntityType y EntityScore.
Detecta ntities-entity-text-score	Si hay al menos una entidad en la DetectEntities subextensión que coincida con EntityText y EntityScore.
Detecta ntities-entity-text-type	Si hay al menos una entidad en la DetectEntities subextensión que coincida con EntityText y EntityType.
Detecta ntities-entity-category-score	Si hay al menos una entidad que coincida con EntityCategory y EntityScore.
infer-icd10 -code cm-entity-text-concept	Si hay al menos una entidad en la subextensión Infer ICD1 0CM que coincide con el EntityText y hay al menos un ConceptCode para esa entidad que coincide con el código.
infer-icd10 -score cm-entity-text-concept	Si hay al menos una entidad en la subextensión Infer ICD1 0CM que coincide con la EntityText y hay al menos un ConceptScore para esa entidad que coincide con la puntuación.

Parámetros de búsqueda	Coincidencias devueltas
infer-icd10 -concept-score cm-entity-concept-description	Si hay al menos un concepto dentro de la entidad en la subextensión Infer ICD1 0CM que coincida con la descripción del concepto y con el ConceptScore.
infer-rxnorm-entity-text-código conceptual	Si hay al menos una entidad en la InferRxNorm subextensión que coincide con el EntityText y hay al menos un ConceptCode para esa entidad que coincide con el código.
infer-rxnorm-entity-text-puntuación conceptual	Si hay al menos una entidad en la InferRxNorm subextensión que coincide con la puntuación de EntityText y hay al menos un ConceptScore para esa entidad que coincide con la puntuación.
infer-rxnorm-entity-concept-description-concept-score	Si hay al menos un concepto dentro de la entidad de la InferRxNorm subextensión que coincida con la descripción del concepto y el ConceptScore.

HealthLake solicitudes de ejemplo de PNL integradas

Ejemplo 1: **`Patient`** registro ingerido en un HealthLake almacén de datos

El siguiente es un ejemplo de una nota clínica basada en un `Patient` encuentro con un profesional de la salud.

> **Datos sintéticos**
>
> El texto del siguiente ejemplo es de contenido sintético y no contiene información de salud protegida (PHI).

```
1991-08-31

# Chief Complaint
- Headache
- Sinus Pain
```

```
- Nasal Congestion
- Sore Throat
- Pain with Bright Lights
- Nasal Discharge
- Cough

# History of Present Illness
Jerónimo599 is a 4 month-old non-hispanic white male.

# Social History
Patient has never smoked.

Patient comes from a middle socioeconomic background.

Patient currently has Aetna.

# Allergies
No Known Allergies.

# Medications
No Active Medications.

# Assessment and Plan
Patient is presenting with bee venom (substance), mold (organism), house dust
 mite (organism), animal dander (substance), grass pollen (substance), tree pollen
 (substance), lisinopril, sulfamethoxazole / trimethoprim, fish (substance).

## Plan

The patient was prescribed the following medications:
- astemizole 10 mg oral tablet
- nda020800 0.3 ml epinephrine 1 mg/ml auto-injector
The patient was placed on a careplan:
- self-care interventions (procedure)
```

Como recordatorio, esta información está codificada en formato base64 en el `DocumentReference` recurso. Cuando haya ingerido este documento HealthLake y se hayan completado las operaciones de la API Amazon Comprehend Medical, para ver los resultados, puede empezar con la `GET` solicitud del tipo de recurso. `DocumentReference`

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

Cuando las operaciones de la API Amazon Comprehend Medical se realicen correctamente, busque estos pares clave-valor en `extension` el enlace a lo siguiente `"url": "http://healthlake.amazonaws.com/aws-cm/"`

```
{
  "url": "http://healthlake.amazonaws.com/aws-cm/status/",
  "valueString": "SUCCESS"
 },
 {
  "url": "http://healthlake.amazonaws.com/aws-cm/message/",
  "valueString": "The AWS HealthLake integrated medical NLP operation was successful."
 }
```

En las siguientes pestañas, se muestra cómo se registra la historia clínica ingresada en su almacén de HealthLake datos en función del tipo de recurso.

DocumentReference

Para ver los resultados de un solo tipo de `DocumentReference` recurso, realice una `GET` solicitud en la que se proporcione el `id` de un recurso específico.

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

Si tiene éxito, obtendrá un código de respuesta `200` HTTP y la siguiente respuesta JSON (que se ha truncado para mayor claridad).

Esta es la parte. `http://healthlake.amazonaws.com/system-generated-resources/` Puede ver que se `Linkage/e366d29f-2c22-4c19-866e-09603937935a` ha agregado una nueva. También puede ver dónde se HealthLake han agregado los hallazgos basados en inferencias a tipos específicos `Observation` y de `Condition` recursos.

Para ver cómo se han modificado estos tipos de recursos, selecciona las pestañas relacionadas.

```
{
  "extension": [
   {
    "url": "http://healthlake.amazonaws.com/linkage",
    "valueReference": {
```

```
      "reference": "Linkage/e366d29f-2c22-4c19-866e-09603937935a"
     }
    },
    {
     "url": "http://healthlake.amazonaws.com/nlp-entity",
     "valueReference": {
      "reference": "Observation/c6e0a3ff-7a17-4d8b-bfd0-d02d7da090c5"
     }
    },
    {
     "url": "http://healthlake.amazonaws.com/nlp-entity",
     "valueReference": {
      "reference": "Condition/0854e1f3-894d-448e-a8d9-3af5b9902baf"
     }
    }
   ],
   "url": "http://healthlake.amazonaws.com/system-generated-resources/"
  }
```

Linkage

Para ver los resultados de un solo tipo de `Linkage` recurso, realiza una `GET` solicitud en la que se proporcione el `ID` de un recurso específico.

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

Si tiene éxito, obtendrá un código de respuesta `200` HTTP y la siguiente respuesta JSON truncada.

La respuesta contiene el `item` elemento. En él, el par clave-valor `"type": "source"` indica la `DocumentReference` entrada específica utilizada para modificar el par clave-valor `Condition` y que `Observations` aparece debajo del par `"type": "alternate"` clave-valor.

También puede ver el *`meta`* elemento y el par clave-valor correspondiente, lo que indica que estos recursos fueron *`"tag": [{"display": "SYSTEM_GENERATED"}]`* creados por. HealthLake

```
{
  "resourceType": "Linkage",
  "id": "e366d29f-2c22-4c19-866e-09603937935a",
  "active": true,
```

```
  "item":
  [
   {
    "type": "alternate",
    "resource": {
     "reference": "Observation/c6e0a3ff-7a17-4d8b-bfd0-d02d7da090c5",
     "type": "Observation"
    }
   },
   {
    "type": "alternate",
    "resource": {
     "reference": "Condition/9d5c1ef6-f822-4faf-b55f-7c70f2a4aa8d",
     "type": "Condition"
    }
   },
   {
    "type": "source",
    "resource": {
     "reference": "DocumentReference/0e938f03-da7f-4178-acd8-eea9586c46ed",
     "type": "DocumentReference"
    }
   }
  ],
  "meta": {
   "lastUpdated": "2022-10-21T19:38:31.327Z",
   "tag": [{
    "display": "SYSTEM_GENERATED"
   }]
  }
 }
```

Resource type: Observation

Para ver los resultados de un solo tipo de `Observation` recurso, realiza una `GET` solicitud en la que se proporcione el `ID` de un recurso específico.

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

Los resultados de las operaciones de la API Amazon Comprehend Medical se modifican para incluir los siguientes elementos`code`:`meta`, `modifierExtension` y.

`code`

Un elemento de tipo`CodeableConcept`. Para obtener más información, consulte `CodeableConcept`la documentación del FHIR R4.

HealthLake agrega los tres pares clave-valor siguientes.

- `"system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/"`: donde la URL hace referencia a una operación específica de la API Amazon Comprehend Medical. En este caso, infiera 0 cm. ICD1
- `"code": "A52.06"`: ¿Dónde `A52.06` está el código ICD-10-CM que identifica el concepto que se encuentra en la base de conocimientos de los Centros para el Control de Enfermedades?
- `"display": "Other syphilitic heart involvement"`: ¿Dónde `"Other syphilitic heart involvement"` está la descripción larga del código ICD-10-CM en la ontología?

La siguiente respuesta JSON truncada contiene solo el elemento. `code`

```
"code": {
  "coding":
  [
    {
   "system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/",
   "code": "A52.06",
   "display": "Other syphilitic heart involvement"
    }
  ],
  "text": "Other syphilitic heart involvement"
 }
```

Para entender si el modelo confía en que el código ICD-10-CM asignado es correcto, utilice el elemento. `modifierExtension`

`meta`

El `meta` elemento contiene metadatos que indican si el `code` elemento contiene detalles añadidos por las operaciones de la API Amazon Comprehend Medical.

La siguiente respuesta JSON truncada contiene solo el elemento. `meta`

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

modifierExtension

El `modifierExtension` elemento contiene más detalles sobre el nivel de confianza de los códigos asignados que se encuentran en el `code` elemento. También tiene pares clave-valor que proporcionan un enlace al original DocumentReference utilizado para generar los resultados y al tipo de recurso de enlace relacionado.

Para cada `coding` elemento agregado, verá una `entity-score` y una `entity-Concept-Score` agregada a la ModifierExtension. Para cada valor del par clave-valor, verá una puntuación. Pues`entity-score`, esta puntuación es el nivel de confianza que Amazon Comprehend Medical tiene en la precisión de la detección. `entity-Concept-Score`Pues esta puntuación es el nivel de confianza que Amazon Comprehend Medical tiene en cuanto a que la entidad está vinculada con precisión al concepto ICD-10-CM.

La siguiente respuesta JSON truncada contiene solo el elemento. `modifierExtension`

```
"modifierExtension": [{
   "url": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/aws-cm-icd10-entity-
score",
   "valueDecimal": 0.45005733
  },
  {
   "url": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/aws-cm-icd10-entity-
Concept-Score",
   "valueDecimal": 0.1111792
  },
  {
   "url": "http://healthlake.amazonaws.com/system-generated-linkage",
   "valueReference": {
    "reference": "Linkage/e366d29f-2c22-4c19-866e-09603937935a"
   }
  },
  {
   "url": "http://healthlake.amazonaws.com/source-document-reference",
```

```
   "valueReference": {
    "reference": "DocumentReference/0e938f03-da7f-4178-acd8-eea9586c46ed"
   }
  }
 ]
```

Respuesta JSON completa

```
{
  "subject": {
   "reference": "Patient/0679b7b7-937d-488a-b48d-6315b8e7003b"
  },
  "resourceType": "Observation",
  "status": "unknown",
  "code": {
   "coding": [{
    "system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/",
    "code": "A52.06",
    "display": "Other syphilitic heart involvement"
   }],
   "text": "Other syphilitic heart involvement"
  },
  "meta": {
   "lastUpdated": "2022-10-21T19:38:30.879Z",
   "tag": [{
    "display": "SYSTEM_GENERATED"
   }]
  },
  "modifierExtension": [{
    "url": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/aws-cm-icd10-entity-
score",
    "valueDecimal": 0.45005733
   },
   {
    "url": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/aws-cm-icd10-entity-
Concept-Score",
    "valueDecimal": 0.1111792
   },
   {
    "url": "http://healthlake.amazonaws.com/system-generated-linkage",
    "valueReference": {
     "reference": "Linkage/e366d29f-2c22-4c19-866e-09603937935a"
    }
```

```
   },
   {
    "url": "http://healthlake.amazonaws.com/source-document-reference",
    "valueReference": {
     "reference": "DocumentReference/0e938f03-da7f-4178-acd8-eea9586c46ed"
    }
   }
  ],
  "id": "7e88c7c5-21a5-4dd7-8fc2-a02474fba583"
 }
```

Condition

Para ver los resultados de un solo tipo de `Condition` recurso, realiza una `GET` solicitud en la que se proporcione el `ID` de un recurso específico.

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

Los resultados de las operaciones de la API Amazon Comprehend Medical se modifican para incluir los siguientes elementos`code`:`meta`, `modifierExtension` y.

`code`

Un elemento de tipo`CodeableConcept`. Para obtener más información, consulte `CodeableConcept`la documentación del FHIR R4.

HealthLake agrega los tres pares clave-valor siguientes.

- `"system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/"`: donde la URL hace referencia a una operación específica de la API Amazon Comprehend Medical. En este caso, infiera 0 cm. ICD1
- `"code": "I70.0"`: ¿Dónde `A52.06` está el código ICD-10-CM que identifica el concepto que se encuentra en la base de conocimientos de los Centros para el Control de Enfermedades?
- `"display": "Atherosclerosis of aorta"`: ¿Dónde `"Other syphilitic heart involvement"` está la descripción larga del código ICD-10-CM en la ontología?

La siguiente respuesta JSON truncada contiene solo el elemento. `code`

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

Para entender si el modelo confía en que el código ICD-10-CM asignado es correcto, utilice el elemento. `modifierExtension`

`meta`

El `meta` elemento contiene metadatos que indican si el `code` elemento contiene detalles añadidos por las operaciones de la API Amazon Comprehend Medical.

La siguiente respuesta JSON truncada contiene solo el elemento. `meta`

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

`modifierExtension`

El `modifierExtension` elemento contiene más detalles sobre el nivel de confianza de los códigos asignados que se encuentran en el `code` elemento. También tiene pares clave-valor que proporcionan un enlace al original DocumentReference utilizado para generar los resultados y al tipo de recurso de enlace relacionado.

Para cada `coding` elemento agregado, verá una `entity-score` y una `entity-Concept-Score` agregada a la ModifierExtension. Para cada valor del par clave-valor, verá una puntuación. Pues`entity-score`, esta puntuación es el nivel de confianza que Amazon Comprehend Medical tiene en la precisión de la detección. `entity-Concept-Score`Pues esta puntuación es el nivel

de confianza que Amazon Comprehend Medical tiene en cuanto a que la entidad está vinculada con precisión al concepto ICD-10-CM.

La siguiente respuesta JSON truncada contiene solo el elemento. `modifierExtension`

```
"modifierExtension": [{
   "url": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/aws-cm-icd10-entity-
score",
   "valueDecimal": 0.94417894
  },
  {
   "url": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/aws-cm-icd10-entity-
Concept-Score",
   "valueDecimal": 0.8458298
  },
  {
   "url": "http://healthlake.amazonaws.com/system-generated-linkage",
   "valueReference": {
    "reference": "Linkage/e366d29f-2c22-4c19-866e-09603937935a"
   }
  },
  {
   "url": "http://healthlake.amazonaws.com/source-document-reference",
   "valueReference": {
    "reference": "DocumentReference/0e938f03-da7f-4178-acd8-eea9586c46ed"
   }
  }
 ]
```

Respuesta JSON completa

```
{
  "subject": {
   "reference": "Patient/0679b7b7-937d-488a-b48d-6315b8e7003b"
  },
  "resourceType": "Condition",
  "code": {
   "coding": [{
    "system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/",
    "code": "I70.0",
    "display": "Atherosclerosis of aorta"
   }],
   "text": "Atherosclerosis of aorta"
```

```
  },
  "meta": {
   "lastUpdated": "2022-10-21T19:38:30.877Z",
   "tag": [{
    "display": "SYSTEM_GENERATED"
   }]
  },
  "modifierExtension": [{
    "url": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/aws-cm-icd10-entity-
score",
    "valueDecimal": 0.94417894
   },
   {
    "url": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/aws-cm-icd10-entity-
Concept-Score",
    "valueDecimal": 0.8458298
   },
   {
    "url": "http://healthlake.amazonaws.com/system-generated-linkage",
    "valueReference": {
     "reference": "Linkage/e366d29f-2c22-4c19-866e-09603937935a"
    }
   },
   {
    "url": "http://healthlake.amazonaws.com/source-document-reference",
    "valueReference": {
     "reference": "DocumentReference/0e938f03-da7f-4178-acd8-eea9586c46ed"
    }
   }
  ],
  "id": "b06d343d-ddb8-4f36-82cb-853fcd434dfd"
 }
```


Ejemplo 2: Una **`DocumentReference`** que contiene un tipo **`MedicationStatement`** de recurso

Este es un ejemplo de una nota clínica basada en el encuentro de un paciente con un profesional médico.

> **⚠ Datos sintéticos**
>
> El texto de este ejemplo es de contenido sintético y no contiene información de salud protegida (PHI).

```
Tom is not prescribed Advil
```

Las siguientes pestañas muestran cómo se registra el registro médico ingerido en el almacén de HealthLake datos en función del tipo de recurso.

DocumentReference

Para ver los resultados de un solo tipo de `DocumentReference` recurso, realice una `GET` solicitud en la que se proporcione el `ID` de un recurso específico.

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

Si tiene éxito, obtendrá un código de respuesta `200` HTTP y la siguiente respuesta JSON truncada.

El par clave-valor,`"url": "http://healthlake.amazonaws.com/system-generated-resources/"`, indica que los tipos de recursos que contiene los `extension` han agregado las operaciones de la API Amazon Comprehend Medical. Puede ver el nuevo tipo de `Linkage` recurso y varios recursos. `MedicationStatement`

```
"extension": [{
   "extension": [{
     "url": "http://healthlake.amazonaws.com/linkage",
     "valueReference": {
      "reference": "Linkage/394bb244-177b-4409-8657-26b20ed56dd7"
     }
    },
    {
     "url": "http://healthlake.amazonaws.com/nlp-entity",
     "valueReference": {
      "reference": "MedicationStatement/cbf6af10-b0b9-451c-bdde-99611e3498a8"
     }
```

```
    },
    {
     "url": "http://healthlake.amazonaws.com/nlp-entity",
     "valueReference": {
      "reference": "MedicationStatement/9a89b0d3-6681-45ca-9926-27951edce5c7"
     }
    },
    {
     "url": "http://healthlake.amazonaws.com/nlp-entity",
     "valueReference": {
      "reference": "MedicationStatement/4a01f6c8-5f3a-4122-80ab-405312f96aa2"
     }
    },
    {
     "url": "http://healthlake.amazonaws.com/nlp-entity",
     "valueReference": {
      "reference": "MedicationStatement/fbfb77d8-70cf-4579-b4c0-d6fe3c01656b"
     }
    },
    {
     "url": "http://healthlake.amazonaws.com/nlp-entity",
     "valueReference": {
      "reference": "MedicationStatement/1340c9ce-9c48-4bf9-9b2f-d0ab027f5e0b"
     }
    }
   ],
   "url": "http://healthlake.amazonaws.com/system-generated-resources/"
  }
```

Linkage

Para ver los resultados de un solo tipo de `Linkage` recurso, realiza una `GET` solicitud en la que se proporcione el `ID` de un recurso específico.

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

Si tiene éxito, obtendrá un código de respuesta `200` HTTP y la siguiente respuesta JSON.

La respuesta contiene el `item` elemento. En él, el par clave-valor `"type": "source"` indica la `DocumentReference` entrada específica utilizada para modificar los tipos de `MedicationStatement` recursos.

También puede ver el `meta` elemento y el par clave-valor correspondiente`"tag": [{"display": "SYSTEM_GENERATED"}]`, lo que indica que estos recursos fueron creados por. HealthLake


```
{
  "resourceType": "Linkage",
  "id": "394bb244-177b-4409-8657-26b20ed56dd7",
  "active": true,
  "item": [{
    "type": "alternate",
    "resource": {
     "reference": "MedicationStatement/cbf6af10-b0b9-451c-bdde-99611e3498a8",
     "type": "MedicationStatement"
    }
   },
   {
    "type": "alternate",
    "resource": {
     "reference": "MedicationStatement/9a89b0d3-6681-45ca-9926-27951edce5c7",
     "type": "MedicationStatement"
    }
   },
   {
    "type": "alternate",
    "resource": {
     "reference": "MedicationStatement/4a01f6c8-5f3a-4122-80ab-405312f96aa2",
     "type": "MedicationStatement"
    }
   },
   {
    "type": "alternate",
    "resource": {
     "reference": "MedicationStatement/fbfb77d8-70cf-4579-b4c0-d6fe3c01656b",
     "type": "MedicationStatement"
    }
   },
   {
    "type": "alternate",
    "resource": {
     "reference": "MedicationStatement/1340c9ce-9c48-4bf9-9b2f-d0ab027f5e0b",
     "type": "MedicationStatement"
    }
   },
```

```
   {
    "type": "source",
    "resource": {
     "reference": "DocumentReference/c549125d-a218-421f-b8bf-23614c5e796c",
     "type": "DocumentReference"
    }
   }
  ],
  "meta": {
   "lastUpdated": "2022-10-24T20:05:03.501Z",
   "tag": [{
    "display": "SYSTEM_GENERATED"
   }]
  }
 }
```

MedicationStatement

Para ver los resultados de un solo tipo de `MedicationStatement` recurso, realiza una `GET` solicitud en la que se proporcione el `ID` de un recurso específico.

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

El tipo de `MedicationStatement` recurso es donde se encuentran los resultados de la operación de la API Amazon Comprehend InferRxNorm Medical. Los resultados se modifican para incluir los siguientes elementos:`medicationCodeableConcept`,`meta`, y`modifierExtension`.

medicationCodeableConcept

Un elemento de tipo`CodeableConcept`. Para obtener más información, consulte `CodeableConcept`la documentación del FHIR R4.

HealthLake agrega los tres pares clave-valor siguientes.

- `"system": ""http://healthlake.amazonaws.com/aws-cm/infer-rxnorm/`: donde la URL hace referencia a una operación específica de la API Amazon Comprehend Medical. En este caso, InferRxNorm.
- `"code": "731533"`: ¿Dónde `731533` está un identificador RxNorm conceptual, también conocido como RxCUI?

- `"display": "ibuprofen 200 MG Oral Capsule [Advil]"`: ¿Dónde `ibuprofen 200 MG Oral Capsule [Advil]` está la descripción del concepto? RxNorm

La siguiente respuesta JSON truncada contiene solo el `MedicationStatement` elemento.

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

`meta`

El `meta` elemento contiene metadatos que indican si el `code` elemento contiene detalles añadidos por las operaciones de la API Amazon Comprehend Medical.

La siguiente respuesta JSON truncada contiene solo el elemento. `meta`

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

`modifierExtension`

El `modifierExtension` elemento contiene pares clave-valor que proporcionan un enlace al original `DocumentReference` utilizado para generar los resultados y al tipo de recurso de enlace relacionado.

```
"modifierExtension": [
   {
    "url": "http://healthlake.amazonaws.com/system-generated-linkage",
```

```
    "valueReference": {
     "reference": "Linkage/394bb244-177b-4409-8657-26b20ed56dd7"
    }
   },
   {
    "url": "http://healthlake.amazonaws.com/source-document-reference",
    "valueReference": {
     "reference": "DocumentReference/c549125d-a218-421f-b8bf-23614c5e796c"
    }
   }
]
```

Consulta de HealthLake datos con Amazon Athena

Durante un trabajo de HealthLake importación, los datos JSON anidados del FHIR se someten a un proceso ETL y se almacenan en el formato de tabla abierta Apache Iceberg, donde cada tipo de recurso del FHIR se representa como una tabla individual en Athena. Esto permite a los usuarios consultar los datos del FHIR mediante SQL, pero sin tener que exportarlos primero. Esto es valioso, ya que permite a los médicos y científicos consultar los datos del FHIR para validar sus decisiones o avanzar en sus investigaciones. Para obtener más información sobre el funcionamiento de las tablas de Iceberg de Apache en Athena, consulte Consulte las tablas de Apache Iceberg en la Guía del usuario de Athena.

> **Note**
>
> HealthLake admite la `read` interacción FHIR R4 con sus HealthLake datos en Athena. Para obtener más información, consulte Lectura de un recurso del FHIR.

En los temas de esta sección se describe cómo conectar el HealthLake banco de datos a Athena, cómo consultarlo mediante SQL y cómo conectar los resultados con otros AWS servicios para su posterior análisis.

Temas

- Cómo empezar a usar Amazon Athena
- Consulta de HealthLake datos con SQL
- Ejemplos de consultas SQL con filtrado complejo

Cómo empezar a usar Amazon Athena

Para integrarse HealthLake con Amazon Athena, debe configurar los permisos. Para ello, creará un usuario, grupo o rol de Athena y le concederá acceso a los recursos del FHIR ubicados en un HealthLake almacén de datos.

- Otorgar a un usuario, grupo o rol acceso a un almacén de HealthLake datos (AWS Lake Formation Console)
- Configuración de una cuenta de Athena

Otorgar a un usuario, grupo o rol acceso a un almacén de HealthLake datos (AWS Lake Formation Console)

> Persona: HealthLake administrador
>
> La persona HealthLake administradora es un administrador de un lago de datos en AWS Lake Formation. Permiten el acceso a HealthLake los almacenes de datos en Lake Formation.

Para cada banco de datos creado, hay dos entradas visibles en la consola de AWS Lake Formation. Una entrada es un enlace a un recurso. Los nombres de los enlaces de recursos siempre se muestran en cursiva. Cada enlace a un recurso se muestra con el nombre y el propietario del recurso compartido vinculado. En todos los almacenes de HealthLake datos, el propietario del recurso compartido es la cuenta HealthLake de servicio. La otra entrada es el almacén HealthLake de datos de la cuenta HealthLake de servicio. En los pasos de este procedimiento se utiliza el banco de datos que es el enlace de recursos.

Para obtener más información sobre los enlaces de recursos, consulte Cómo funcionan los enlaces de recursos en Lake Formation en la Guía para desarrolladores de AWS Lake Formation.

Para que un usuario, grupo o rol pueda consultar datos en Athena, debe conceder el permiso Describe en la base de datos de recursos. A continuación, debe conceder las opciones Seleccionar y Describir en las tablas.

PASO 1: Otorgar los permisos DESCRIBE en una base de HealthLake datos de enlaces a recursos del almacén de datos

1. Abre la consola de AWS Lake Formation: https://console.aws.amazon.com/lakeformation/
2. En la barra de navegación principal, selecciona Bases de datos.
3. En la página Bases de datos, pulse el botón de opción situado junto al nombre del banco de datos que aparece en cursiva.
4. Elija Acciones (▼).
5. Elija Conceder.
6. En la página Otorgar permisos de datos, en Directores, selecciona usuarios o roles de IAM.
7. En Usuarios o roles de IAM, utilice la flecha hacia abajo (▼) o busque el usuario, rol o grupo de IAM sobre el que desee realizar consultas en Athena.
8. En las etiquetas LF o en la tarjeta de recursos del catálogo, seleccione la opción Recursos del catálogo de datos con nombre asignado.
9. En Bases de datos, utilice la flecha hacia abajo (▼) para elegir la base de HealthLake datos del almacén de datos a la que desea compartir el acceso.
10. En la tarjeta de permisos de enlace de recursos, en Permisos de enlace de recursos, elija Describir.

Cuando la concesión se haya concedido correctamente, aparecerá el aviso de autorización de concesión correcta. Para ver el permiso que acabas de conceder, selecciona Permisos de Data Lake. Busque el usuario, el grupo y el rol en la tabla. En la columna Permisos, verá la lista Describe.

Ahora debe usar Grant on target para conceder las opciones Seleccionar y Describir en todas las tablas de la base de datos.

PASO 2: Otorgue acceso a todas las tablas de un enlace de recursos del almacén de HealthLake datos

1. Abre la consola de AWS Lake Formation: https://console.aws.amazon.com/lakeformation/
2. En la barra de navegación principal, selecciona Bases de datos.
3. En la página Bases de datos, pulse el botón de opción situado junto al nombre del banco de datos que aparece en cursiva.
4. Elija Acciones (▼).

5. Elige Subvencionar según el objetivo.
6. En la página Otorgar permisos de datos, en Directores, selecciona usuarios o roles de IAM.
7. En Usuarios o roles de IAM, utilice la flecha hacia abajo (▼) o busque el usuario, grupo o rol de IAM sobre el que desee realizar consultas en Athena.
8. En las etiquetas LF o en la tarjeta de recursos del catálogo, seleccione la opción Recursos del catálogo de datos con nombre asignado.
9. En Bases de datos, utilice la flecha hacia abajo (▼) para elegir la base de HealthLake datos del almacén de datos a la que desea conceder acceso.
10. En Tablas, elija Todas las tablas para compartir todas las tablas con un HealthLake usuario.
11. En la tarjeta de permisos de tabla, en Permisos de tabla, elija Describir y seleccionar.
12. Elija Conceder.

Tras seleccionar la concesión, aparece un aviso de autorización de autorización de concesión de permisos. El usuario especificado ahora puede realizar consultas en un almacén de HealthLake datos de Athena.

Cómo empezar con Athena

> **HealthLake usuario**
>
> El HealthLake usuario utilizará la consola de Athena o AWS SDKs consultará un almacén de HealthLake datos compartido con él por el HealthLake administrador. AWS CLI

Para consultar un banco de datos con Athena, debe hacer las tres cosas siguientes.

- Conceda al usuario o rol de IAM acceso al almacén de HealthLake datos a través de Lake Formation. Para obtener más información, consulte Otorgar a un usuario, grupo o rol acceso a un almacén de HealthLake datos (AWS Lake Formation Console).
- Cree un grupo de trabajo para su almacén de HealthLake datos.
- Designe un bucket de Amazon S3 para almacenar los resultados de las consultas.

Para empezar a utilizar Athena, añada las políticas gestionadas por Amazon S3 AmazonAthenaFullAccessy las políticas FullAccess AWS gestionadas por Amazon S3 a su usuario, grupo o función. El uso de una política AWS administrada es una excelente forma de empezar a

utilizar un nuevo servicio. Tenga en cuenta que es posible que las políticas administradas por AWS no otorguen permisos de privilegios mínimos para sus casos de uso específicos, ya que están disponibles para que los usen todos los clientes de AWS. Cuando establezca permisos con políticas de IAM, conceda solo los permisos necesarios para llevar a cabo una tarea. Para obtener más información sobre la IAM y la aplicación de los privilegios mínimos, consulte Aplicar los permisos con los privilegios mínimos en la Guía del usuario de IAM.

> **Important**
>
> Para consultar un banco HealthLake de datos en Athena, debe utilizar la versión 3 del motor Athena.

Los grupos de trabajo son recursos y, por lo tanto, puede utilizar políticas basadas en la IAM para controlar el acceso a grupos de trabajo específicos. Para obtener más información, consulte Uso de grupos de trabajo para controlar el acceso y los costos de las consultas en la Guía del usuario de Athena.

Para obtener más información sobre la configuración de grupos de trabajo, consulte la https://docs.aws.amazon.com/athena/latest/ug/workgroups-procedure.html Guía del usuario de Athena.

> **Note**
>
> La región en la que se encuentra su bucket de Amazon S3 y la consola Athena deben coincidir.

Antes de poder ejecutar una consulta, se debe especificar una ubicación de bucket de resultado de consulta en Amazon S3 o debe utilizar un grupo de trabajo que tenga especificado un bucket y cuya configuración sobrescriba la configuración del cliente. Los archivos de salida se guardan automáticamente por cada consulta que se hace.

Para obtener más información sobre cómo especificar las ubicaciones de los resultados de las consultas en la consola de Athena, consulte Especificar una ubicación de resultados de consulta mediante la consola de Athena en la Guía del usuario de Amazon Athena.

Para ver ejemplos de cómo consultar el banco de HealthLake datos en Athena, consulte. Consulta de HealthLake datos con SQL

Consulta de HealthLake datos con SQL

Al importar los datos del FHIR al almacén de HealthLake datos, los datos FHIR de JSON anidados se someten simultáneamente a un proceso de ETL y se almacenan en formato de tabla abierta Apache Iceberg en Amazon S3. Cada tipo de recurso FHIR del almacén de HealthLake datos se convierte en una tabla, donde se puede consultar mediante Amazon Athena. Las tablas se pueden consultar individualmente o en grupo mediante consultas basadas en SQL. Debido a la estructura de los almacenes de datos, los datos se importan a Athena como varios tipos de datos diferentes. Para obtener más información sobre la creación de consultas SQL que puedan acceder a estos tipos de datos, consulte Matrices de consultas con tipos complejos y estructuras anidadas en la Guía del usuario de Amazon Athena.

> **Note**
>
> Todos los ejemplos de este tema utilizan datos ficticios creados con Synthea. Para obtener más información sobre cómo crear un banco de datos precargado con datos de Synthea, consulte. Creación de un almacén HealthLake de datos

Para cada elemento de un tipo de recurso, la especificación FHIR define una cardinalidad. La cardinalidad de un elemento define los límites inferior y superior del número de veces que puede aparecer este elemento. Al crear una consulta SQL, debe tener esto en cuenta. Por ejemplo, veamos algunos elementos de Tipo de recurso: paciente.

- Elemento: nombre La especificación del FHIR establece la cardinalidad como. `0..*`

  El elemento se captura como una matriz.

```
[{
 id = null,
 extension = null,
 use = official,
 _use = null,
 text = null,
 _text = null,
 family = Wolf938,
 _family = null,
 given = [Noel608],
 _given = null,
 prefix = null,
```

```
 _prefix = null,
 suffix = null,
 _suffix = null,
 period = null
}]
```

En Athena, para ver cómo se ha ingerido un tipo de recurso, búsquelo en Tablas y vistas. Para acceder a los elementos de esta matriz, puede utilizar la notación de puntos. A continuación, se muestra un ejemplo sencillo que permite acceder a los valores de `given` y`family`.

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

- Elemento: MaritalStatus La especificación del FHIR establece la cardinalidad como. `0..1`

  Este elemento se captura como JSON.

```
{
 id = null,
 extension = null,
 coding = [
  {
   id = null,
   extension = null,
   system = http: //terminology.hl7.org/CodeSystem/v3-MaritalStatus,
    _system = null,
   version = null,
   _version = null,
   code = S,
   _code = null,
   display = Never Married,
   _display = null,
   userSelected = null,
   _userSelected = null
  }

 ],
 text = Never Married,
 _text = null
}
```

En Athena, para ver cómo se ha ingerido un tipo de recurso, búsquelo en Tablas y vistas. Para acceder a los pares clave-valor en el JSON, puede utilizar la notación de puntos. Como no es una matriz, no se requiere ningún índice de matriz. A continuación, se muestra un ejemplo sencillo con el que se accedería al valor `detext`.

```
SELECT
    maritalstatus.text as MaritalStatus
FROM Patient
```

Para obtener más información sobre cómo acceder y buscar en JSON, consulte Consultas de JSON en la Guía del usuario de Athena.

Las declaraciones de consulta del Lenguaje de Manipulación de Datos (DML) de Athena se basan en Trino. Athena no es compatible con todas las funciones de Trino y existen diferencias significativas. Para obtener más información, consulte las consultas, las funciones y los operadores de DML en la Guía del usuario de Amazon Athena.

Además, Athena admite varios tipos de datos que puede encontrar al crear consultas en su banco de HealthLake datos. Para obtener más información sobre los tipos de datos en Athena, consulte Tipos de datos en Amazon Athena en la Guía del usuario de Amazon Athena.

Para obtener más información sobre cómo funcionan las consultas SQL en Athena, consulte la referencia de SQL para Amazon Athena en la Guía del usuario de Amazon Athena.

Cada pestaña muestra ejemplos de cómo buscar en los tipos de recursos especificados y los elementos asociados con Athena.

Element: Extension

El elemento `extension` se utiliza para crear campos personalizados en un banco de datos.

En este ejemplo, se muestra cómo acceder a las funciones del `extension` elemento que se encuentra en el tipo de `Patient` recurso.

Cuando el banco de HealthLake datos se importa a Athena, los elementos de un tipo de recurso se analizan de forma diferente. Como la estructura de `element` es variable, no se puede especificar completamente en el esquema. Para gestionar esa variabilidad, los elementos de la matriz se pasan como cadenas.

En la descripción de la tablaPatient, puede ver el elemento extension descrito comoarray<string>, lo que significa que puede acceder a los elementos de la matriz mediante un valor de índice. Sin embargo, para acceder a los elementos de la cadena, debe utilizarjson_extract.

Esta es una única entrada del extension elemento que se encuentra en la tabla de pacientes.

```
[{
  "valueString": "Kerry175 Cummerata161",
  "url": "http://hl7.org/fhir/StructureDefinition/patient-mothersMaidenName"
 },
 {
  "valueAddress": {
   "country": "DE",
   "city": "Hamburg",
   "state": "Hamburg"
  },
  "url": "http://hl7.org/fhir/StructureDefinition/patient-birthPlace"
 },
 {
  "valueDecimal": 0.0,
  "url": "http://synthetichealth.github.io/synthea/disability-adjusted-life-years"
 },
 {
  "valueDecimal": 5.0,
  "url": "http://synthetichealth.github.io/synthea/quality-adjusted-life-years"
 }
]
```

Aunque se trata de un JSON válido, Athena lo trata como una cadena.

En este ejemplo de consulta SQL se muestra cómo se puede crear una tabla que contenga los patient-birthPlace elementos patient-mothersMaidenName y. Para acceder a estos elementos, debe utilizar diferentes índices de matriz y json_extract.

```
SELECT
    extension[1],
    json_extract(extension[1], '$.valueString') AS MothersMaidenName,
    extension[2],
    json_extract(extension[2], '$.valueAddress.city') AS birthPlace
FROM patient
```

Para obtener más información sobre las consultas que implican JSON, consulte Extracción de datos de JSON en la Guía del usuario de Amazon Athena.

Element: birthDate (Age)

La edad no es un elemento del tipo de recurso para pacientes en el FHIR. Estos son dos ejemplos de búsquedas que se filtran según la edad.

Como la edad no es un elemento, utilizamos la `birthDate` para las consultas SQL. Para ver cómo se ha incorporado un elemento al FHIR, busque el nombre de la tabla en Tablas y vistas. Puede ver que es de tipo cadena.

Ejemplo 1: Calcular un valor para la edad

En este ejemplo de consulta SQL, utilizamos una herramienta SQL integrada `year` para extraer esos componentes. `current_date` Luego, los restamos para obtener la edad real del paciente, como se llama `age` una columna.

```
SELECT
 (year(current_date) - year(date(birthdate))) as age
FROM patient
```

Ejemplo 2: Filtrar los pacientes que nacieron antes `2019-01-01` y que lo son`male`.

La consulta SQL muestra cómo utilizar la `CAST` función para convertir el `birthDate` elemento en un tipo `DATE` y cómo filtrar en función de dos criterios de la `WHERE` cláusula. Como el elemento se ingiere como cadena de texto de forma predeterminada, debemos `CAST` incorporarlo como tipo`DATE`. Luego puede usar el < operador para compararlo con una fecha diferente,`2019-01-01`. Al usar`AND`, puede agregar un segundo criterio a la `WHERE` cláusula.

```
SELECT birthdate
FROM patient
-- we convert birthdate (varchar) to date  > cast that as date too
WHERE CAST(birthdate AS DATE) < CAST('2019-01-01' AS DATE) AND gender = 'male'
```

Resource type: Location

En este ejemplo, se muestran las búsquedas de ubicaciones dentro del tipo de recurso Ubicación donde el nombre de la ciudad es Attleboro.

```
SELECT *
```

```
FROM Location
WHERE address.city='ATTLEBORO'
LIMIT 10;
```

Element: Age

```
SELECT birthdate
FROM patient
-- we convert birthdate (varchar) to date  > cast that as date too
WHERE CAST(birthdate AS DATE) < CAST('2019-01-01' AS DATE) AND gender = 'male'
```

Resource type: Condition

La condición del tipo de recurso almacena datos de diagnóstico relacionados con problemas que han llegado a ser preocupantes. HealthLakeEl procesamiento médico del lenguaje natural (PNL) integrado genera nuevos `Condition` recursos en función de los detalles que se encuentran en el tipo de DocumentReference recurso. Cuando se genera un nuevo recurso, HealthLake añade la etiqueta `SYSTEM_GENERATED` al `meta` elemento. En este ejemplo de consulta SQL se muestra cómo se puede buscar en la tabla de condiciones y devolver los `SYSTEM_GENERATED` resultados cuando se han eliminado.

Para obtener más información sobre HealthLake el procesamiento del lenguaje natural (NLP) integrado, consulteProcesamiento de lenguaje natural (PNL) integrado para HealthLake.

```
SELECT *
FROM condition
WHERE meta.tag[1] is NULL
```

También puede buscar dentro de un elemento de cadena específico para filtrar aún más la consulta. El `modifierextension` elemento contiene detalles sobre qué `DocumentReference` recurso se utilizó para generar un conjunto de condiciones. De nuevo, debe utilizarlos `json_extract` para acceder a los elementos JSON anidados que se traen a Athena como una cadena.

En este ejemplo de consulta SQL se muestra cómo buscar todo lo `Condition` que se ha generado a partir de una consulta específica. `DocumentReference` Se utiliza `CAST` para establecer el elemento JSON como una cadena que se puede utilizar `LIKE` para comparar.

```
SELECT
    meta.tag[1].display as SystemGenerated,
```

```
    json_extract(modifierextension[4], '$.valueReference.reference') as
 DocumentReference
FROM condition
WHERE meta.tag[1].display = 'SYSTEM_GENERATED'

AND CAST(json_extract(modifierextension[4], '$.valueReference.reference') as
 VARCHAR) LIKE '%DocumentReference/67aa0278-8111-40d0-8adc-43055eb9d18d%'
```

Resource type: Observation

El tipo de recurso, Observation, almacena las mediciones y las afirmaciones simples realizadas sobre un paciente, un dispositivo u otro tema. HealthLakeEl procesamiento del lenguaje natural (PNL) integrado genera nuevos `Observation` recursos en función de los detalles que se encuentran en un `DocumentReference` recurso. Este ejemplo de consulta SQL incluye `WHERE meta.tag[1] is NULL` comentarios, lo que significa que se incluyen los `SYSTEM_GENERATED` resultados.

```
SELECT valueCodeableConcept.coding[1].code
FROM Observation
WHERE  valueCodeableConcept.coding[1].code = '266919005'
-- WHERE meta.tag[1] is NULL
```

Esta columna se importó como `struct`. Por lo tanto, puede acceder a los elementos que contiene mediante la notación de puntos.

Resource type: MedicationStatement

MedicationStatement es un tipo de recurso del FHIR que puede utilizar para almacenar detalles sobre los medicamentos que un paciente ha tomado, está tomando o tomará en el futuro. HealthLakeEl procesamiento médico del lenguaje natural (PNL) integrado genera nuevos MedicationStatement recursos a partir de los documentos que se encuentran en ese tipo de DocumentReference recurso. Cuando se generan nuevos recursos, HealthLake añade la etiqueta `SYSTEM_GENERATED` al `meta` elemento. Este ejemplo de consulta SQL muestra cómo crear una consulta que filtre en función de un solo paciente mediante su identificador y busque los recursos que se han agregado mediante HealthLake la PNL integrada.

```
SELECT *
FROM medicationstatement
WHERE meta.tag[1].display = 'SYSTEM_GENERATED' AND subject.reference =
 'Patient/0679b7b7-937d-488a-b48d-6315b8e7003b';
```

Para obtener más información sobre HealthLake el procesamiento del lenguaje natural (NLP) integrado, consulte. Procesamiento de lenguaje natural (PNL) integrado para HealthLake

Ejemplos de consultas SQL con filtrado complejo

Los siguientes ejemplos muestran cómo utilizar las consultas SQL de Amazon Athena con filtrado complejo para localizar los datos del FHIR de un HealthLake almacén de datos.

Example Cree criterios de filtrado basados en datos demográficos

Es importante identificar los datos demográficos correctos de los pacientes a la hora de crear una cohorte de pacientes. Este ejemplo de consulta demuestra cómo puede utilizar la notación de puntos de Trino y filtrar `json_extract` los datos del banco de HealthLake datos.

```
SELECT
    id
    , CONCAT(name[1].family, ' ', name[1].given[1]) as name
    , (year(current_date) - year(date(birthdate))) as age
    , gender as gender
    , json_extract(extension[1], '$.valueString') as MothersMaidenName
    , json_extract(extension[2], '$.valueAddress.city') as birthPlace
    , maritalstatus.coding[1].display as maritalstatus
    , address[1].line[1] as addressline
    , address[1].city as city
    , address[1].district as district
    , address[1].state as state
    , address[1].postalcode as postalcode
    , address[1].country as country
    , json_extract(address[1].extension[1], '$.extension[0].valueDecimal') as latitude
    , json_extract(address[1].extension[1], '$.extension[1].valueDecimal') as longitude
    , telecom[1].value as telNumber
    , deceasedboolean as deceasedIndicator
    , deceaseddatetime
FROM database.patient;
```

Con la consola Athena, puede ordenar y descargar aún más los resultados.

Example Cree filtros para un paciente y sus afecciones relacionadas

La siguiente consulta de ejemplo muestra cómo encontrar y ordenar todas las afecciones relacionadas con los pacientes que se encuentran en un banco de HealthLake datos.

```
SELECT
 patient.id as patientId
    , condition.id  as conditionId
    , CONCAT(name[1].family, ' ', name[1].given[1]) as name
    , condition.meta.tag[1].display
    , json_extract(condition.modifierextension[1], '$.valueDecimal') AS confidenceScore
    , category[1].coding[1].code as categoryCode
    , category[1].coding[1].display as categoryDescription
    , code.coding[1].code as diagnosisCode
    , code.coding[1].display as diagnosisDescription
    , onsetdatetime
    , severity.coding[1].code as severityCode
    , severity.coding[1].display as severityDescription
    , verificationstatus.coding[1].display as verificationStatus
    , clinicalstatus.coding[1].display as clinicalStatus
    , encounter.reference as encounterId
    , encounter.type as encountertype
FROM database.patient, condition
WHERE CONCAT('Patient/', patient.id) = condition.subject.reference
ORDER BY name;
```

Puede utilizar la consola Athena para ordenar aún más los resultados o descargarlos para analizarlos más a fondo.

Example Cree filtros para los pacientes y sus observaciones relacionadas

La siguiente consulta de ejemplo muestra cómo buscar y ordenar todas las observaciones relacionadas de los pacientes que se encuentran en un HealthLake banco de datos.

```
SELECT
 patient.id as patientId
    , observation.id as observationId
    , CONCAT(name[1].family, ' ', name[1].given[1]) as name
    , meta.tag[1].display
    , json_extract(modifierextension[1], '$.valueDecimal') AS confidenceScore
    , status
    , category[1].coding[1].code as categoryCode
    , category[1].coding[1].display as categoryDescription
    , code.coding[1].code as observationCode
    , code.coding[1].display as observationDescription
    , effectivedatetime
    , CASE
```

```
  WHEN valuequantity.value IS NOT NULL THEN CONCAT(CAST(valuequantity.value AS
 VARCHAR),' ',valuequantity.unit)
       WHEN valueCodeableConcept.coding [ 1 ].code IS NOT NULL THEN
 CAST(valueCodeableConcept.coding [ 1 ].code AS VARCHAR)
       WHEN valuestring IS NOT NULL THEN CAST(valuestring AS VARCHAR)
       WHEN valueboolean IS NOT NULL THEN CAST(valueboolean AS VARCHAR)
       WHEN valueinteger IS NOT NULL THEN CAST(valueinteger AS VARCHAR)
       WHEN valueratio IS NOT NULL THEN CONCAT(CAST(valueratio.numerator.value AS
 VARCHAR),'/',CAST(valueratio.denominator.value AS VARCHAR))
       WHEN valuerange IS NOT NULL THEN CONCAT(CAST(valuerange.low.value AS
 VARCHAR),'-',CAST(valuerange.high.value AS VARCHAR))
       WHEN valueSampledData IS NOT NULL THEN CAST(valueSampledData.data AS VARCHAR)
       WHEN valueTime IS NOT NULL THEN CAST(valueTime AS VARCHAR)
       WHEN valueDateTime IS NOT NULL THEN CAST(valueDateTime AS VARCHAR)
       WHEN valuePeriod IS NOT NULL THEN valuePeriod.start
       WHEN component[1] IS NOT NULL THEN CONCAT(CAST(component[2].valuequantity.value
 AS VARCHAR),' ',CAST(component[2].valuequantity.unit AS VARCHAR),
 '/', CAST(component[1].valuequantity.value AS VARCHAR),'
 ',CAST(component[1].valuequantity.unit AS VARCHAR))
    END AS observationvalue
 , encounter.reference as encounterId
    , encounter.type as encountertype
FROM database.patient, observation
WHERE CONCAT('Patient/', patient.id) = observation.subject.reference
ORDER BY name;
```

Example Cree condiciones de filtrado para un paciente y sus procedimientos relacionados

Conectar los procedimientos con los pacientes es un aspecto importante de la atención médica. La siguiente consulta de ejemplo de SQL muestra cómo utilizar el FHIR `Patient` y `Procedure` los tipos de recursos para lograrlo. La siguiente consulta SQL devolverá todos los pacientes y sus procedimientos relacionados que se encuentran en el almacén de HealthLake datos.

```
SELECT
 patient.id  as patientId
 , PROCEDURE.id as procedureId
 , CONCAT(name[1].family, ' ', name[1].given[1]) as name
 , status
 , category.coding[1].code as categoryCode
 , category.coding[1].display as categoryDescription
 , code.coding[1].code as procedureCode
 , code.coding[1].display as procedureDescription
 , performeddatetime
```

```
 , performer[1]
 , encounter.reference as encounterId
 , encounter.type as encountertype
FROM database.patient, procedure
WHERE CONCAT('Patient/', patient.id) = procedure.subject.reference
ORDER BY name;
```

Puede utilizar la consola Athena para descargar los resultados para analizarlos más a fondo o ordenarlos para comprenderlos mejor.

Example Cree condiciones de filtrado para un paciente y sus recetas relacionadas

Es importante ver una lista actualizada de los medicamentos que toman los pacientes. Con Athena, puede escribir una consulta SQL que utilice tanto los tipos de `MedicationRequest` recursos como los `Patient` que se encuentran en su almacén de HealthLake datos.

La siguiente consulta SQL une las `MedicationRequest` tablas `Patient` y importadas a Athena. También organiza las recetas en sus entradas individuales mediante la notación de puntos.

```
SELECT
 patient.id  as patientId
 , medicationrequest.id  as medicationrequestid
 , CONCAT(name[1].family, ' ', name[1].given[1]) as name
 , status
 , statusreason.coding[1].code as categoryCode
 , statusreason.coding[1].display as categoryDescription
 , category[1].coding[1].code as categoryCode
 , category[1].coding[1].display as categoryDescription
 , priority
 , donotperform
 , encounter.reference as encounterId
 , encounter.type as encountertype
 , medicationcodeableconcept.coding[1].code as medicationCode
 , medicationcodeableconcept.coding[1].display as medicationDescription
 , dosageinstruction[1].text as dosage
FROM database.patient, medicationrequest
WHERE CONCAT('Patient/', patient.id ) = medicationrequest.subject.reference
ORDER BY name
```

Puede utilizar la consola Athena para ordenar los resultados o descargarlos para analizarlos más a fondo.

Example Consulte los medicamentos que se encuentran en el tipo de **`MedicationStatement`** recurso

La siguiente consulta de ejemplo muestra cómo organizar el JSON anidado importado a Athena mediante SQL. La consulta utiliza el `meta` elemento FHIR para indicar cuándo se ha agregado un medicamento mediante el procesamiento HealthLake del lenguaje natural (PNL) integrado. También se utiliza `json_extract` para buscar datos dentro de la matriz de cadenas JSON. Para obtener más información, consulte Procesamiento de lenguaje natural.

```
SELECT
 medicationcodeableconcept.coding[1].code as medicationCode
 , medicationcodeableconcept.coding[1].display as medicationDescription
 , meta.tag[1].display
 , json_extract(modifierextension[1], '$.valueDecimal') AS confidenceScore
FROM medicationstatement;
```

Puede utilizar la consola de Athena para descargar estos resultados u ordenarlos.

Example Filtra por un tipo de enfermedad específico

El ejemplo muestra cómo encontrar un grupo de pacientes de entre 18 y 75 años a los que se les haya diagnosticado diabetes.

```
SELECT patient.id as patientId,
 condition.id as conditionId,
 CONCAT(name [ 1 ].family, ' ', name [ 1 ].given [ 1 ]) as name,
 (year(current_date) - year(date(birthdate))) AS age,
 CASE
  WHEN condition.encounter.reference IS NOT NULL THEN condition.encounter.reference
  WHEN observation.encounter.reference IS NOT NULL THEN observation.encounter.reference
 END as encounterId,
 CASE
  WHEN condition.encounter.type IS NOT NULL THEN observation.encounter.type
  WHEN observation.encounter.type IS NOT NULL THEN observation.encounter.type
 END AS encountertype,
 condition.code.coding [ 1 ].code as diagnosisCode,
 condition.code.coding [ 1 ].display as diagnosisDescription,
 observation.category [ 1 ].coding [ 1 ].code as categoryCode,
 observation.category [ 1 ].coding [ 1 ].display as categoryDescription,
 observation.code.coding [ 1 ].code as observationCode,
 observation.code.coding [ 1 ].display as observationDescription,
 effectivedatetime AS observationDateTime,
```

```
 CASE
      WHEN valuequantity.value IS NOT NULL THEN CONCAT(CAST(valuequantity.value AS
 VARCHAR),' ',valuequantity.unit)
      WHEN valueCodeableConcept.coding [ 1 ].code IS NOT NULL THEN
 CAST(valueCodeableConcept.coding [ 1 ].code AS VARCHAR)
      WHEN valuestring IS NOT NULL THEN CAST(valuestring AS VARCHAR)
      WHEN valueboolean IS NOT NULL THEN CAST(valueboolean AS VARCHAR)
      WHEN valueinteger IS NOT NULL THEN CAST(valueinteger AS VARCHAR)
      WHEN valueratio IS NOT NULL THEN CONCAT(CAST(valueratio.numerator.value AS
 VARCHAR),'/',CAST(valueratio.denominator.value AS VARCHAR))
      WHEN valuerange IS NOT NULL THEN CONCAT(CAST(valuerange.low.value AS
 VARCHAR),'-',CAST(valuerange.high.value AS VARCHAR))
      WHEN valueSampledData IS NOT NULL THEN CAST(valueSampledData.data AS VARCHAR)
      WHEN valueTime IS NOT NULL THEN CAST(valueTime AS VARCHAR)
      WHEN valueDateTime IS NOT NULL THEN CAST(valueDateTime AS VARCHAR)
      WHEN valuePeriod IS NOT NULL THEN valuePeriod.start
      WHEN component[1] IS NOT NULL THEN CONCAT(CAST(component[2].valuequantity.value
 AS VARCHAR),' ',CAST(component[2].valuequantity.unit AS VARCHAR),
 '/', CAST(component[1].valuequantity.value AS VARCHAR),'
 ',CAST(component[1].valuequantity.unit AS VARCHAR))
    END AS observationvalue,
 CASE
  WHEN condition.meta.tag [ 1 ].display = 'SYSTEM GENERATED' THEN 'YES'
  WHEN condition.meta.tag [ 1 ].display IS NULL THEN 'NO'
  WHEN observation.meta.tag [ 1 ].display = 'SYSTEM GENERATED' THEN 'YES'
  WHEN observation.meta.tag [ 1 ].display IS NULL THEN 'NO'
   END AS IsSystemGenerated,
  CAST(
    json_extract(
      condition.modifierextension [ 1 ],
      '$.valueDecimal'
    ) AS int
  ) AS confidenceScore
FROM database.patient,
 database.condition,
 database.observation
WHERE CONCAT('Patient/', patient.id) = condition.subject.reference
 AND CONCAT('Patient/', patient.id) = observation.subject.reference
   AND (year(current_date) - year(date(birthdate))) >= 18
   AND (year(current_date) - year(date(birthdate))) <= 75
   AND condition.code.coding [ 1 ].display like ('%diabetes%');
```

Ahora puede usar la consola Athena para ordenar los resultados o descargarlos para analizarlos más a fondo.

Monitorización AWS HealthLake

El monitoreo y el registro son partes importantes para mantener la seguridad, la confiabilidad, la disponibilidad y el rendimiento de AWS HealthLake. AWS proporciona los siguientes servicios para observar HealthLake, informar cuando algo anda mal y tomar medidas automáticas cuando sea apropiado.

- AWS CloudTrailcaptura las llamadas a la API y los eventos relacionados realizados por su AWS cuenta o en su nombre y entrega los archivos de registro a un bucket de Amazon S3 que especifique. Puede identificar qué usuarios y cuentas llamaron AWS, la dirección IP de origen desde la que se realizaron las llamadas y cuándo se produjeron. Para obtener más información, consulte la Guía del usuario de AWS CloudTrail.
- Amazon CloudWatch monitorea tus AWS recursos y las aplicaciones en las que AWS ejecutas en tiempo real. Puede recopilar métricas y realizar un seguimiento de las métricas, crear paneles personalizados y definir alarmas que le advierten o que toman medidas cuando una métrica determinada alcanza el umbral que se especifique. Por ejemplo, puedes CloudWatch hacer un seguimiento del uso de la CPU u otras métricas de tus EC2 instancias de Amazon y lanzar automáticamente nuevas instancias cuando sea necesario. Para obtener más información, consulta la Guía del CloudWatch usuario de Amazon.
- Amazon EventBridge es un servicio de bus de eventos sin servidor que facilita la conexión de sus aplicaciones con datos de diversas fuentes. EventBridge ofrece un flujo de datos en tiempo real desde sus propias aplicaciones, aplicaciones Software-as-a-Service (SaaS) y AWS servicios, y dirige esos datos a destinos como Lambda. Esto le permite monitorear los eventos que ocurren en los servicios y crear arquitecturas basadas en eventos. Para obtener más información, consulta la Guía del EventBridge usuario de Amazon.

Temas

- Registro de llamadas a la HealthLake API mediante AWS CloudTrail
- Monitorización HealthLake de métricas con Amazon CloudWatch
- Monitorización de HealthLake eventos con Amazon EventBridge

Registro de llamadas a la HealthLake API mediante AWS CloudTrail

AWS HealthLake está integrado con AWS CloudTrail un servicio que proporciona un registro de las acciones realizadas por un usuario, un rol o un AWS servicio en HealthLake. CloudTrail captura todas las llamadas a la API HealthLake como eventos. Las llamadas capturadas incluyen llamadas desde la HealthLake consola y llamadas en código a las operaciones de la HealthLake API. Si crea una ruta, puede habilitar la entrega continua de CloudTrail eventos a un bucket de Amazon S3, incluidos los eventos para HealthLake. Si no configura una ruta, podrá ver los eventos más recientes en la CloudTrail consola, en el historial de eventos. Con la información recopilada por usted CloudTrail, puede determinar a HealthLake qué dirección IP se realizó la solicitud, quién la realizó, cuándo se realizó y detalles adicionales.

Para obtener más información CloudTrail, consulte la Guía AWS CloudTrail del usuario.

AWS HealthLake Información en CloudTrail

CloudTrail está habilitada en su AWS cuenta al crear la cuenta. Cuando se produce una actividad en HealthLake, esa actividad se registra en un CloudTrail evento junto con otros eventos de AWS servicio en el historial de eventos. Se puede ver, buscar y descargar los últimos eventos de la cuenta de AWS . Para obtener más información, consulte Visualización de eventos con el historial de CloudTrail eventos.

Para tener un registro continuo de los eventos de tu AWS cuenta, incluidos los eventos de tu cuenta HealthLake, crea una ruta. Un rastro permite CloudTrail entregar archivos de registro a un bucket de Amazon S3. De forma predeterminada, cuando se crea un registro de seguimiento en la consola, el registro de seguimiento se aplica a todas las regiones de AWS. La ruta registra los eventos de todas las regiones de la AWS partición y envía los archivos de registro al bucket de Amazon S3 que especifique. Además, puede configurar otros AWS servicios para analizar más a fondo los datos de eventos recopilados en los CloudTrail registros y actuar en función de ellos. Para más información, consulte los siguientes temas:

- Introducción a la creación de registros de seguimiento
- CloudTrail Integraciones y servicios compatibles
- Configuración de las notificaciones de Amazon SNS para CloudTrail
- Recibir archivos de CloudTrail registro de varias regiones y recibir archivos de CloudTrail registro de varias cuentas

Todas HealthLake las acciones se registran CloudTrail y se documentan en la referencia de la HealthLake API y en esta guía para desarrolladores para las acciones realizadas con la API REST del FHIR. Por ejemplo, las llamadas a las siguientes acciones generan entradas en los archivos de CloudTrail registro:

- `DescribeFHIRImportJob`
- `DescribeFHIRExportJob`
- `StartFHIRImportJob`
- `ListFHIRImportJobs`
- `StartFHIRExportJob`
- `ListFHIRExportJobs`
- `CreateFHIRDatastore`
- `ListFHIRDatastores`
- `DeleteFHIRDatastore`
- `DescribeFHIRDatastore`
- `UpdateResource`
- `CreateResource`
- `DeleteResource`
- `ReadResource`
- `GetCapabilities`
- `SearchWithGet`
- `SearchWithPost`

Cada entrada de registro o evento contiene información sobre quién generó la solicitud. La información de identidad del usuario le ayuda a determinar lo siguiente:

- Si la solicitud se realizó con credenciales de usuario root o AWS Identity and Access Management (IAM).
- Si la solicitud se realizó con credenciales de seguridad temporales de un rol o fue un usuario federado.
- Si la solicitud la realizó otro AWS servicio.

Para obtener más información, consulte el Elemento userIdentity de CloudTrail .

Descripción de las entradas de los archivos de AWS HealthLake registro

Un rastro es una configuración que permite la entrega de eventos como archivos de registro a un bucket de Amazon S3 que especifique. CloudTrail Los archivos de registro contienen una o más entradas de registro. Un evento representa una solicitud única de cualquier fuente e incluye información sobre la acción solicitada, la fecha y la hora de la acción, los parámetros de la solicitud, etc. CloudTrail Los archivos de registro no son un registro ordenado de las llamadas a la API pública, por lo que no aparecen en ningún orden específico.

En el siguiente ejemplo, se muestra una entrada de CloudTrail registro que demuestra la `CreateFHIRDatastore` acción.


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA2B3ZHOADD2OJ4AHJX:git
 full_access_iam_role580074395690222150",
        "arn": "arn:aws:sts::691207106566:assumed-role/
colossusfrontend_full_access_iam_role/_iam_role580074395690222150",
        "accountId": "AccountID",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROA2B3ZHOADD2OJ4AHJX",
                "arn": "arn:aws:iam::691207106566:role/full_access_iam_role",
                "accountId": "AccountID",
                "userName": "full_access_iam_role"
            },
            "webIdFederationData": {

            },
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-11-20T00:08:15Z"
            }
        }
    },
    "eventTime": "2020-11-20T00:08:16Z",
    "eventSource": "healthlake.amazonaws.com",
    "eventName": "CreateFHIRDatastore",
```

```
    "awsRegion": "us-east-1",
    "sourceIPAddress": "3.213.247.1",
    "userAgent": "Coral/Netty4",
    "requestParameters": {
        "datastoreName":
 "testCreateFHIRDatastore_GBYAZFCLLBLSUTOYYFQZRLBLQJNFOYQVRPZBOJAIIUAHICAEAGIWLNVQEYAMSXVWMBLXC
        "datastoreTypeVersion": "R4",
        "clientToken": "d737ffe0-14dd-44cc-9f0a-fdf59b26c66b"
    },
    "responseElements": {
        "datastoreId": "datastoreID",
        "datastoreArn": "arn:aws:healthlake:us-
east-1:691207106566:datastore/55576c487ff4975262b10d1d65eb4509",
        "datastoreStatus": "CREATING",
        "datastoreEndpoint": "datastore_endpoint/"
    },
    "requestID": "68e62bdd-d2d4-44c1-af69-e6f055a69f99",
    "eventID": "7ef483dc-5dca-469e-823a-7d9e3a7fe924",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "691207106566"
}
```


Monitorización HealthLake de métricas con Amazon CloudWatch

Puedes monitorizar HealthLake con Amazon CloudWatch, que recopila datos sin procesar y los procesa para convertirlos en métricas legibles prácticamente en tiempo real. Estas estadísticas se mantienen durante 15 meses, de modo que pueda consultar información histórica y disponer de una mejor perspectiva sobre el desempeño de su aplicación web o servicio. También puede establecer alarmas que vigilen determinados umbrales y enviar notificaciones o realizar acciones cuando se cumplan dichos umbrales. Para obtener más información, consulta la Guía del CloudWatch usuario de Amazon.

> **Note**
>
> Se muestran las métricas de todas las HealthLakeacciones nativas.

En las siguientes tablas se enumeran las HealthLake métricas y las dimensiones a las que se ha informado CloudWatch. Cada una se presenta como un recuento de frecuencias para un rango de datos especificado por el usuario.

Se informa a las siguientes HealthLake métricas CloudWatch.

HealthLake métricas notificadas a CloudWatch

Métrica	Description (Descripción)
Conteo de llamadas	El número de llamadas a APIs. Se puede informar de ello para la cuenta o para un almacén de datos específico. Unidades: recuento Estadísticas válidas: Sum, Count Dimensiones: funcionamiento, ID del almacén de datos, tipo de almacén de datos
Solicitudes satisfactorias	Número de solicitudes de la API realizadas correctamente. Unidades: recuento Estadísticas válidas: Sum, Average Dimensiones: funcionamiento, ID del almacén de datos, tipo de almacén de datos
Errores de usuario	El número de solicitudes que fallaron debido a un error del usuario. Unidades: recuento Estadísticas válidas: Sum, Average Dimensiones: funcionamiento, ID del almacén de datos, tipo de almacén de datos

Métrica	Description (Descripción)
Errores de servidor	El número de solicitudes que fallaron debido a un error del servidor. Unidades: recuento Estadísticas válidas: Sum, Average Dimensiones: funcionamiento, ID del almacén de datos, tipo de almacén de datos
Solicitudes limitadas	El número de solicitudes que se han limitado. Esta métrica no se incluye en los recuentos de errores del usuario o del servidor. Unidades: recuento Estadísticas válidas: Sum, Average Dimensiones: funcionamiento, ID del almacén de datos, tipo de almacén de datos
Latencia	El tiempo en milisegundos que se tardó en procesar la solicitud del usuario. Unidad: milisegundos Estadísticas válidas: mínimo, máximo, promedio Dimensiones: funcionamiento, ID del almacén de datos, tipo de almacén de datos

Se informa de las siguientes HealthLake dimensiones CloudWatch.

HealthLake Dimensiones notificadas a CloudWatch

Dimensión	Description (Descripción)
Operación	La operación de API utilizada en la solicitud
DataStoreID	El ID del almacén de datos utilizado en la solicitud
DataStoreType	El tipo de almacén de datos utilizado en la solicitud

Puede obtener métricas HealthLake con la consola de administración de AWS AWS CLI, la o la CloudWatch API. Puede usar la CloudWatch API a través de uno de los kits de desarrollo de software de Amazon AWS (SDKs) o las herramientas de la CloudWatch API. La HealthLake consola muestra gráficos basados en los datos sin procesar de la CloudWatch API.

Debe tener los CloudWatch permisos adecuados para poder realizar la supervisión HealthLake CloudWatch. Para obtener más información, consulta Gestión de identidad y acceso para Amazon CloudWatch en la Guía del CloudWatch usuario de Amazon.

Visualización de HealthLake las métricas

Para ver las métricas (CloudWatch consola)

1. Inicie sesión en la CloudWatchconsola Consola de administración de AWS y ábrala.
2. Elija Métricas, elija Todas las métricas y, a continuación, elija AWS/ HealthLake.
3. Elija la dimensión, un nombre de métrica y, a continuación, Add to graph (Añadir al gráfico).
4. Elija un valor para el intervalo de fechas. El recuento de las métricas del intervalo de fechas seleccionado se muestra en el gráfico.

Crear una alarma mediante CloudWatch

Una CloudWatch alarma vigila una única métrica durante un período de tiempo específico y realiza una o más acciones: enviar una notificación a un tema del Amazon Simple Notification Service (SNS) o a una política de Auto Scaling. La acción o las acciones se basan en el valor de la métrica en relación con un umbral determinado durante un número de períodos de tiempo que usted

especifique. CloudWatch también puede enviarle un mensaje de SNS cuando la alarma cambie de estado.

> **Note**
>
> CloudWatch las alarmas invocan acciones solo cuando el estado cambia y ha persistido durante el período que usted especifique.

Para ver las métricas (consola) CloudWatch

1. Inicie sesión en la consola de CloudWatch.
2. Elija Alarms y, a continuación, seleccione Create Alarm.
3. Elija AWS/ yHealthLake, a continuación, elija una métrica.
4. En Time Range, elija un intervalo de tiempo para monitorizar y, a continuación, elija Next.
5. Introduzca un Nombre y una Descripción.
6. En Whenever, elija >= e introduzca un valor máximo.
7. Si quieres CloudWatch enviar un correo electrónico cuando se alcance el estado de alarma, en la sección Acciones, en Siempre que aparezca esta alarma, selecciona State is ALARM. En Enviar notificación a, selecciona una lista de correo o selecciona Nueva lista y crea una nueva lista de correo.
8. Obtenga una vista previa de la alarma en la sección Vista previa de la alarma. Si está satisfecho con la alarma, elija Create Alarm.

Monitorización de HealthLake eventos con Amazon EventBridge

Amazon EventBridge es un servicio sin servidor que utiliza eventos para conectar los componentes de la aplicación entre sí, lo que facilita la creación de aplicaciones escalables basadas en eventos. La base EventBridge es crear reglas que dirijan los eventos a los objetivos. AWS HealthLake proporciona una entrega duradera de los cambios de estado a EventBridge. Para obtener más información, consulta ¿Qué es Amazon EventBridge? en la Guía del EventBridge usuario de Amazon.

> **Note**
>
> Para obtener información sobre cómo enviar HealthLake eventos a Amazon EventBridge, consulta la EventBridge integración con Amazon AWS HealthLake en el blog AWS for Industries.

Temas

- HealthLake eventos enviados a EventBridge
- HealthLake estructura de eventos

HealthLake eventos enviados a EventBridge

En la siguiente tabla se enumeran todos los HealthLake eventos enviados a EventBridge para su procesamiento.

HealthLake tipo de evento	Estado
eventos del almacén de datos	
Creación de un almacén de datos	`CREATING`
Almacén de datos activo	`ACTIVE`
Eliminación del almacén de datos	`DELETING`
Almacén de datos eliminado	`DELETED`
Falló la creación del almacén de datos	`CREATE_FAILED`
Para obtener más información, consulta `datastoreStatus` en la AWS HealthLake Referencia de la API de .	
Importación de eventos de trabajo	
Trabajo de importación enviado	`SUBMITTED`
Importar trabajo en curso	`IN_PROGRESS`

HealthLake tipo de evento	Estado
Job de importación completado con errores	`COMPLETED_WITH_ERRORS`
Trabajo de importación completado	`COMPLETED`
No se pudo importar el trabajo	`FAILED`
Para obtener más información, consulta `jobStatus` en la AWS HealthLake Referencia de la API de .	
Exportación de eventos de trabajo	
Exportar trabajo enviado	`SUBMITTED`
Trabajo de exportación en curso	`IN_PROGRESS`
Job de exportación completado con errores	`COMPLETED_WITH_ERRORS`
Trabajo de exportación completado	`COMPLETED`
No se pudo exportar el trabajo	`FAILED`
Para obtener más información, consulta `jobStatus` en la AWS HealthLake Referencia de la API de .	

HealthLake estructura de eventos

HealthLake los eventos son objetos con estructura JSON que también contienen detalles de metadatos. Puede utilizar los metadatos como entrada para recrear un evento o para obtener más información. Todos los campos de metadatos asociados se muestran en una tabla bajo los ejemplos de código de los siguientes menús. Para obtener más información, consulta los metadatos de los eventos de AWS servicio en la Guía del EventBridge usuario de Amazon.

> **Note**
>
> Para obtener información sobre cómo enviar HealthLake eventos a Amazon EventBridge, consulta la EventBridge integración con Amazon AWS HealthLake en el blog AWS for Industries.

Eventos del almacén de datos

Data Store Creating

Estado - **CREATING**

```
{
    "version": "0",
    "id": "514ad836-bb8a-4523-a10b-fa2756c3bdb0",
    "detail-type": "Data Store Creating",
    "source": "aws.healthlake",
    "account": "123456789012",
    "time": "2023-12-08T08:58:12Z",
    "region": "us-east-1",
    "resources":
    [
        "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
eeb8005725ae22b35b4edbdc68cf2dfd"
    ],
    "detail":
    {
        "datastoreId": "eeb8005725ae22b35b4edbdc68cf2dfd",
        "datastoreName": "your-data-store-name",
        "datastoreTypeVersion": "R4",
        "datastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/datastore/
eeb8005725ae22b35b4edbdc68cf2dfd/r4/"
    }
}
```

Data Store Active

Estado - **ACTIVE**

```
{
    "version": "0",
    "id": "d57105bc-0d2d-4009-b34d-453e2567c599",
    "detail-type": "Data Store Active",
    "source": "aws.healthlake",
    "account": "123456789012",
    "time": "2023-12-08T09:16:51Z",
    "region": "us-east-1",
    "resources":
    [
```

```
        "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
eeb8005725ae22b35b4edbdc68cf2dfd"
    ],
    "detail":
    {
        "datastoreId": "eeb8005725ae22b35b4edbdc68cf2dfd",
        "datastoreName": "your-data-store-name",
        "datastoreTypeVersion": "R4",
        "datastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/datastore/
eeb8005725ae22b35b4edbdc68cf2dfd/r4/"
    }
}
```

Data Store Deleting

Estado - **DELETING**

```
{
    "version": "0",
    "id": "d135ee1f-e14a-4730-8766-7b98f822c94a",
    "detail-type": "Data Store Deleting",
    "source": "aws.healthlake",
    "account": "123456789012",
    "time": "2023-12-08T12:44:47Z",
    "region": "us-east-1",
    "resources":
    [
        "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
eeb8005725ae22b35b4edbdc68cf2dfd"
    ],
    "detail":
    {
        "datastoreId": "eeb8005725ae22b35b4edbdc68cf2dfd",
        "datastoreName": "your-data-store-name",
        "datastoreTypeVersion": "R4",
        "datastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/datastore/
eeb8005725ae22b35b4edbdc68cf2dfd/r4/"
    }
}
```

Data Store Deleted

Estado - **DELETED**

```
{
    "version": "0",
    "id": "6d880b86-e115-4947-81a9-494db704571a",
    "detail-type": "Data Store Deleted",
    "source": "aws.healthlake",
    "account": "123456789012",
    "time": "2023-05-12T12:58:03Z",
    "region": "us-east-1",
    "resources":
    [
        "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
eeb8005725ae22b35b4edbdc68cf2dfd"
    ],
    "detail":
    {
        "datastoreId": "eeb8005725ae22b35b4edbdc68cf2dfd",
        "datastoreName": "your-data-store-name",
        "datastoreTypeVersion": "R4",
        "datastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/datastore/
eeb8005725ae22b35b4edbdc68cf2dfd/r4/"
    }
}
```

Eventos del almacén de datos: descripciones de metadatos

Name	Tipo	Description (Descripción)
`version`	cadena	La versión EventBridge del esquema de eventos.
`id`	cadena	El UUID de la versión 4 generado para cada evento.
`detail-type`	cadena	El tipo de evento que se envía.
`source`	cadena	Identifica el servicio que generó el evento.

Name	Tipo	Description (Descripción)
`account`	cadena	El ID de cuenta de AWS de 12 dígitos del propietario del almacén de datos.
`time`	cadena	La hora a la que ocurrió el evento.
`region`	cadena	Identifica la AWS región del banco de datos.
`resources`	matriz (cadena)	Una matriz JSON que contiene el ARN del almacén de datos.
`detail`	objeto	Un objeto JSON que contiene información sobre el evento.
`detail.datastoreId`	cadena	El ID del almacén de datos asociado al evento de cambio de estado.
`detail.datastoreName`	cadena	El nombre del almacén de datos.
`detail.datastoreTypeVersion`	cadena	La versión FHIR del almacén de datos.
`detail.datastoreEndpoint`	cadena	El punto final del almacén de datos.

Importar eventos de trabajo

Import Job Submitted

Estado - **`SUBMITTED`**

```
{
```

```
    "version": "0",
    "id": "25e606f7-800c-da41-45df-0e68587250c9",
    "detail-type": "Import Job Submitted",
    "source": "aws.healthlake",
    "account": "123456789012",
    "time": "2023-12-08T01:50:51Z",
    "region": "us-east-1",
    "resources":
    [
        "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
eeb8005725ae22b35b4edbdc68cf2dfd"
    ],
    "detail":
    {
        "jobId": "08c60716d6321710893ff88410e902c2",
        "submitTime": "2023-12-08T01:50:50.986Z",
        "datastoreId": "eeb8005725ae22b35b4edbdc68cf2dfd",
        "inputDataConfig":
        {
            "s3Uri": "s3://amzn-s3-demo-source-bucket/input/"
        }
    }
}
```

Import Job In Progress

Estado - **IN_PROGRESS**

```
{
    "version": "0",
    "id": "cc886b49-2737-19c4-7c4e-84ac9429ab73",
    "detail-type": "Import Job In Progress",
    "source": "aws.healthlake",
    "account": "123456789012",
    "time": "2023-12-08T01:51:23Z",
    "region": "us-east-1",
    "resources":
    [
        "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
eeb8005725ae22b35b4edbdc68cf2dfd"
    ],
    "detail":
    {
        "jobId": "08c60716d6321710893ff88410e902c2",
```

```
        "submitTime": "2023-12-08T01:50:50.986Z",
        "datastoreId": "eeb8005725ae22b35b4edbdc68cf2dfd",
        "inputDataConfig":
        {
            "s3Uri": "s3://amzn-s3-demo-source-bucket/input/"
        }
    }
}
```

Import Job Completed

Estado - **COMPLETED**

```
{
    "version": "0",
    "id": "36c865ef-da41-76ef-c882-3ba8dad8656b",
    "detail-type": "Import Job Completed",
    "source": "aws.healthlake",
    "account": "123456789012",
    "time": "2023-12-08T02:14:42Z",
    "region": "us-east-1",
    "resources":
    [
        "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
eeb8005725ae22b35b4edbdc68cf2dfd"
    ],
    "detail":
    {
        "jobId": "08c60716d6321710893ff88410e902c2",
        "submitTime": "2023-12-08T01:50:50.986Z",
        "datastoreId": "eeb8005725ae22b35b4edbdc68cf2dfd",
        "inputDataConfig":
        {
            "s3Uri": "s3://amzn-s3-demo-source-bucket/input/"
        }
    }
}
```

Import Job Completed With Errors

Estado - **COMPLETED_WITH_ERRORS**

```
{
```

```
    "version": "0",
    "id": "b61387d7-bffe-4f01-8291-65dc4be52cc1",
    "detail-type": "Import Job Completed With Errors",
    "source": "aws.healthlake",
    "account": "123456789012",
    "time": "2023-12-08T02:14:42Z",
    "region": "us-east-1",
    "resources":
    [
        "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
eeb8005725ae22b35b4edbdc68cf2dfd"
    ],
    "detail":
    {
        "jobId": "08c60716d6321710893ff88410e902c2",
        "submitTime": "2023-12-08T01:50:50.986Z",
        "datastoreId": "eeb8005725ae22b35b4edbdc68cf2dfd",
        "inputDataConfig":
        {
            "s3Uri": "s3://amzn-s3-demo-source-bucket/input/"
        }
    }
}
```

Import Job Failed

Estado - **FAILED**

```
{
    "version": "0",
    "id": "c4d65575-d1a7-4040-9c6c-c225bf6723c5",
    "detail-type": "Import Job Failed",
    "source": "aws.healthlake",
    "account": "123456789012",
    "time": "2023-12-08T02:14:42Z",
    "region": "us-east-1",
    "resources":
    [
        "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
eeb8005725ae22b35b4edbdc68cf2dfd"
    ],
    "detail":
    {
        "jobId": "08c60716d6321710893ff88410e902c2",
```

```
        "submitTime": "2023-12-08T01:50:50.986Z",
        "datastoreId": "eeb8005725ae22b35b4edbdc68cf2dfd",
        "inputDataConfig":
        {
            "s3Uri": "s3://amzn-s3-demo-source-bucket/input/"
        }
    }
}
```

Importar eventos de trabajo: descripciones de metadatos

Name	Tipo	Description (Descripción)
`version`	cadena	La versión EventBridge del esquema de eventos.
`id`	cadena	El UUID de la versión 4 generado para cada evento.
`detail-type`	cadena	El tipo de evento que se envía.
`source`	cadena	Identifica el servicio que generó el evento.
`account`	cadena	El ID de cuenta de AWS de 12 dígitos del propietario del almacén de datos.
`time`	cadena	La hora a la que ocurrió el evento.
`region`	cadena	Identifica la AWS región del banco de datos.
`resources`	matriz (cadena)	Una matriz JSON que contiene el ARN del almacén de datos.

Name	Tipo	Description (Descripción)
`detail`	objeto	Un objeto JSON que contiene información sobre el evento.
`detail.jobId`	cadena	El ID del trabajo de importación asociado al evento de cambio de estado.
`detail.submitTime`	cadena	La hora en que se envió el trabajo de importación.
`detail.datastoreId`	cadena	El banco de datos que generó el evento de cambio de estado.
`detail.inputDataConfig`	cadena	La ruta del prefijo de entrada para el depósito de Amazon S3 que contiene los archivos FHIR que se van a importar.

Exporta eventos de trabajo

Export Job Submitted

Estado - **`SUBMITTED`**

```
{
    "version": "0",
    "id": "f8af7d04-2221-4f02-a01a-6fc3ae403bab",
    "detail-type": "Export Job Submitted",
    "source": "aws.healthlake",
    "account": "123456789012",
    "time": "2023-12-08T01:50:51Z",
    "region": "us-east-1",
    "resources":
    [
        "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
eeb8005725ae22b35b4edbdc68cf2dfd"
    ],
```

```
    "detail":
    {
        "jobId": "45e899e545bf774710388260fc60b143",
        "submitTime": "2023-12-08T01:50:50.986Z",
        "datastoreId": "eeb8005725ae22b35b4edbdc68cf2dfd",
        "outputDataConfig":
        {
            "s3Uri": "s3://amzn-s3-demo-source-bucket/output/"
        }
    }
}
```

Export Job In Progress

Estado - **IN_PROGRESS**

```
{
    "version": "0",
    "id": "7bb7e39c-707d-4a83-8532-cee015299100",
    "detail-type": "Export Job In Progress",
    "source": "aws.healthlake",
    "account": "123456789012",
    "time": "2023-12-08T01:51:23Z",
    "region": "us-east-1",
    "resources":
    [
        "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
eeb8005725ae22b35b4edbdc68cf2dfd"
    ],
    "detail":
    {
        "jobId": "45e899e545bf774710388260fc60b143",
        "submitTime": "2023-12-08T01:50:50.986Z",
        "datastoreId": "eeb8005725ae22b35b4edbdc68cf2dfd",
        "outputDataConfig":
        {
            "s3Uri": "s3://amzn-s3-demo-source-bucket/output/"
        }
    }
}
```

Export Job Completed

Estado - **COMPLETED**

```
{
    "version": "0",
    "id": "d7629aa7-e63a-4b84-858c-96a62b57ebc8",
    "detail-type": "Export Job Completed",
    "source": "aws.healthlake",
    "account": "123456789012",
    "time": "2023-12-08T02:14:42Z",
    "region": "us-east-1",
    "resources":
    [
        "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
eeb8005725ae22b35b4edbdc68cf2dfd"
    ],
    "detail":
    {
        "jobId": "45e899e545bf774710388260fc60b143",
        "submitTime": "2023-12-08T01:50:50.986Z",
        "datastoreId": "eeb8005725ae22b35b4edbdc68cf2dfd",
        "outputDataConfig":
        {
            "s3Uri": "s3://amzn-s3-demo-source-bucket/output/"
        }
    }
}
```

Export Job Completed With Errors

Estado - **COMPLETED_WITH_ERRORS**

```
{
    "version": "0",
    "id": "5fa50bc5-50e3-4bc4-b66a-1b1d2f7b07a7",
    "detail-type": "Export Job Completed With Errors",
    "source": "aws.healthlake",
    "account": "123456789012",
    "time": "2023-12-08T02:14:42Z",
    "region": "us-east-1",
    "resources":
    [
        "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
eeb8005725ae22b35b4edbdc68cf2dfd"
    ],
    "detail":
```

```
    {
        "jobId": "45e899e545bf774710388260fc60b143",
        "submitTime": "2023-12-08T01:50:50.986Z",
        "datastoreId": "eeb8005725ae22b35b4edbdc68cf2dfd",
        "outputDataConfig":
        {
            "s3Uri": "s3://amzn-s3-demo-source-bucket/output/"
        }
    }
}
```

Export Job Failed

Estado - **FAILED**

```
{
    "version": "0",
    "id": "49fce45e-7e02-4846-8582-e7f19ca039cb",
    "detail-type": "Export Job Failed",
    "source": "aws.healthlake",
    "account": "123456789012",
    "time": "2023-12-08T02:14:42Z",
    "region": "us-east-1",
    "resources":
    [
        "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
eeb8005725ae22b35b4edbdc68cf2dfd"
    ],
    "detail":
    {
        "jobId": "45e899e545bf774710388260fc60b143",
        "submitTime": "2023-12-08T01:50:50.986Z",
        "datastoreId": "eeb8005725ae22b35b4edbdc68cf2dfd",
        "outputDataConfig":
        {
            "s3Uri": "s3://amzn-s3-demo-source-bucket/output/"
        }
    }
}
```

Exportar eventos de trabajo: descripciones de metadatos

Name	Tipo	Description (Descripción)
`version`	cadena	La versión EventBridge del esquema de eventos.
`id`	cadena	El UUID de la versión 4 generado para cada evento.
`detail-type`	cadena	El tipo de evento que se envía.
`source`	cadena	Identifica el servicio que generó el evento.
`account`	cadena	El ID de cuenta de AWS de 12 dígitos del propietario del almacén de datos.
`time`	cadena	La hora a la que ocurrió el evento.
`region`	cadena	Identifica la AWS región del banco de datos.
`resources`	matriz (cadena)	Una matriz JSON que contiene el ARN del almacén de datos.
`detail`	objeto	Un objeto JSON que contiene información sobre el evento.
`detail.jobId`	cadena	El ID del trabajo de exportación asociado al evento de cambio de estado.
`detail.submitTime`	cadena	La hora en que se envió el trabajo de exportación.

Name	Tipo	Description (Descripción)
`detail.datastoreId`	cadena	El banco de datos que generó el evento de cambio de estado.
`detail.outputDataConfig`	cadena	La ruta del prefijo de salida del depósito de Amazon S3 que contiene los archivos FHIR que se van a exportar.

Seguridad en AWS HealthLake

La seguridad en la nube AWS es la máxima prioridad. Como AWS cliente, usted se beneficia de una arquitectura de centro de datos y red diseñada para cumplir con los requisitos de las organizaciones más sensibles a la seguridad.

La seguridad es una responsabilidad compartida entre usted AWS y usted. El https://aws.amazon.com/compliance/shared-responsibility-model/ se refiere a estos conceptos como seguridad de la nube y seguridad en la nube:

- La seguridad de la nube AWS es responsable de proteger la infraestructura que ejecuta AWS los servicios en la AWS nube. AWS también le proporciona servicios que puede utilizar de forma segura. Third-party los auditores comprueban y verifican periódicamente la eficacia de nuestra seguridad como parte de los AWS programas de de . Para obtener más información sobre los programas de conformidad aplicables HealthLake, consulte Servicios de AWS dentro del alcance por programa de conformidad .
- Seguridad en la nube: su responsabilidad viene determinada por el AWS servicio que utilice. También es responsable de otros factores, incluida la confidencialidad de los datos, los requisitos de la empresa y la legislación y la normativa aplicables.

Esta documentación le ayuda a comprender cómo aplicar el modelo de responsabilidad compartida cuando se utiliza HealthLake. Los siguientes temas muestran cómo configurarlo HealthLake para cumplir sus objetivos de seguridad y conformidad. También aprenderá a utilizar otros servicios de AWS que le ayudan a supervisar y proteger sus HealthLake recursos.

Temas

- Protección de datos en AWS HealthLake
- Cifrado en REST para AWS HealthLake
- Cifrado en tránsito para AWS HealthLake
- Gestión de identidad y acceso para AWS HealthLake
- Validación de conformidad para AWS HealthLake
- Seguridad de la infraestructura en AWS HealthLake
- Creación de AWS HealthLake recursos con AWS CloudFormation
- AWS HealthLake y puntos finales de VPC de interfaz ()AWS PrivateLink
- Mejores prácticas de seguridad en AWS HealthLake

- Resiliencia en AWS HealthLake

Protección de datos en AWS HealthLake

El modelo de responsabilidad compartida de AWS se aplica a la protección de datos en AWS HealthLake. Como se describe en este modelo, AWS es responsable de proteger la infraestructura global que ejecuta todos los Nube de AWS. Eres responsable de mantener el control sobre el contenido alojado en esta infraestructura. También eres responsable de las tareas de administración y configuración de seguridad para los Servicios de AWS que utiliza. Para obtener más información sobre la privacidad de los datos, consulte las Preguntas frecuentes sobre privacidad de datos y los . Para obtener más información sobre la protección de datos en Europa, consulte el Centro del Reglamento General de Protección de Datos (RGPD).

Para fines de protección de datos, le recomendamos que proteja Cuenta de AWS las credenciales y configure usuarios individuales con AWS IAM Identity Center o AWS Identity and Access Management (IAM). De esta manera, solo se otorgan a cada usuario los permisos necesarios para cumplir sus obligaciones laborales. También recomendamos proteger sus datos de la siguiente manera:

- Utiliza la autenticación multifactor (MFA) en cada cuenta.
- Se utiliza SSL/TLS para comunicarse con AWS los recursos. Exigimos TLS 1.2 y recomendamos TLS 1.3.
- Configure la API y el registro de actividad de los usuarios con AWS CloudTrail. Para obtener información sobre el uso de CloudTrail senderos para capturar AWS actividades, consulte Cómo trabajar con CloudTrail senderos en la Guía del AWS CloudTrail usuario.
- Utilice soluciones de AWS cifrado, junto con todos los controles de seguridad predeterminados Servicios de AWS.
- Utiliza servicios de seguridad administrados avanzados, como Amazon Macie, que lo ayuden a detectar y proteger la información confidencial almacenada en Amazon S3.
- Si necesita módulos criptográficos validados por FIPS 140-3 para acceder a AWS través de una interfaz de línea de comandos o una API, utilice un punto final FIPS. Para obtener más información sobre los puntos de conexión de FIPS disponibles, consulte Estándar de procesamiento de la información federal (FIPS) 140-3.

Se recomienda encarecidamente no introducir nunca información confidencial o sensible, como por ejemplo, direcciones de correo electrónico de clientes, en etiquetas o campos de formato libre,

tales como el campo Nombre. Esto incluye cuando trabaja HealthLake o Servicios de AWS utiliza la consola, la API o los SDK. AWS CLI AWS Cualquier dato que introduzca en etiquetas o campos de formato libre utilizados para los nombres se pueden emplear para los registros de facturación o diagnóstico. Si proporciona una URL a un servidor externo, recomendamos encarecidamente que no incluya información de credenciales en la URL a fin de validar la solicitud para ese servidor.

Cifrado en REST para AWS HealthLake

HealthLake proporciona cifrado de forma predeterminada para proteger los datos confidenciales de los clientes en reposo mediante una clave propiedad del servicio AWS Key Management Service (AWS KMS). Customer-managed También se admiten las claves KMS, que son necesarias tanto para la importación como para la exportación de archivos de un almacén de datos. Para obtener más información sobre Customer-managed KMS Key, consulte Amazon Key Management Service. Los clientes pueden elegir una clave de KMS o una clave de Customer-managed KMS propiedad de AWS al crear un almacén de datos. La configuración de cifrado no se puede cambiar una vez creado un almacén de datos. Si un almacén de datos utiliza una clave KMS propiedad de AWS, se indicará como AWS_OWNED_KMS_KEY y no verá la clave específica utilizada para el cifrado en reposo.

Clave KMS propiedad de AWS

HealthLake utiliza estas claves de forma predeterminada para cifrar automáticamente la información potencialmente confidencial, como los datos de identificación personal o de información de salud privada (PHI) en reposo. Las claves de KMS propiedad de AWS no se almacenan en su cuenta. Forman parte de un conjunto de claves de KMS que AWS posee y administra para su uso en varias cuentas de AWS. Los servicios de AWS pueden usar claves de KMS propiedad de AWS para proteger sus datos. No puede ver, administrar, usar las claves de KMS propiedad de AWS ni auditar su uso. Sin embargo, no es necesario que realice ninguna acción ni que cambie programas para proteger las claves que cifran sus datos.

No se le cobrará una cuota mensual ni una cuota de uso si utiliza claves de KMS propiedad de AWS, y no se tienen en cuenta para las cuotas de AWS KMS de su cuenta. Para obtener más información, consulte Claves propiedad de AWS.

Clave de KMS administradas por el cliente

HealthLake admite el uso de una clave KMS simétrica administrada por el cliente que usted crea, posee y administra para agregar una segunda capa de cifrado sobre el cifrado existente

propiedad de AWS. Como usted tiene el control total de dicho cifrado, puede realizar tareas como las siguientes:

- Establecer y mantener políticas de claves, políticas de IAM y concesiones
- Rotar el material criptográfico
- Habilitar y deshabilitar políticas de claves
- Adición de etiquetas de
- Crear alias de clave
- Programar la eliminación de claves

También puede utilizarla CloudTrail para realizar un seguimiento de las solicitudes que se HealthLake envían AWS KMS en su nombre. Se aplican AWS KMS cargos adicionales. Para obtener más información, consulta las claves propiedad del cliente.

Creación de una clave administrada por el cliente

Puede crear una clave simétrica administrada por el cliente mediante la consola de administración de AWS o las AWS KMS API.

Siga los pasos para crear una clave simétrica gestionada por el cliente de la Guía para desarrolladores de AWS Key Management Service.

Las políticas de clave controlan el acceso a la clave administrada por el cliente. Cada clave administrada por el cliente debe tener exactamente una política de clave, que contiene instrucciones que determinan quién puede usar la clave y cómo puede utilizarla. Cuando crea la clave administrada por el cliente, puede especificar una política de clave. Para obtener más información, consulte Administrar el acceso a las claves administradas por el cliente en la Guía para desarrolladores de AWS Key Management Service.

Para utilizar la clave gestionada por el cliente con sus HealthLake recursos, la política de claves debe permitir las CreateGrant operaciones. Esto añade una concesión a una clave gestionada por el cliente que controla el acceso a una clave de KMS específica, lo que permite al usuario acceder a los kms:grant que requieren las operaciones. HealthLake Consulte Uso de subvenciones para obtener más información.

Para usar la clave de KMS administrada por el cliente con sus HealthLake recursos, la política de claves debe permitir las siguientes operaciones de API:

- kms: CreateGrant añade concesiones a una clave de KMS específica administrada por el cliente que permite el acceso a las operaciones de concesión.
- kms: DescribeKey proporciona los detalles de la clave administrada por el cliente necesarios para validar la clave. Esto es necesario para todas las operaciones.
- kms: GenerateDataKey proporciona acceso a los recursos de cifrado en reposo para todas las operaciones de escritura.
- KMS:Decrypt proporciona acceso a las operaciones de lectura o búsqueda de recursos cifrados.

A continuación se muestra un ejemplo de declaración de política que permite a un usuario crear un almacén de datos cifrado con esa clave e interactuar con él: AWS HealthLake

```
"Statement": [
    {
        "Sid": "Allow access to create data stores and do CRUD/search in AWS
 HealthLake",
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::111122223333:HealthLakeFullAccessRole"
        },
        "Action": [
            "kms:DescribeKey",
            "kms:CreateGrant",
            "kms:GenerateDataKey",
            "kms:Decrypt"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "kms:ViaService": "healthlake.amazonaws.com",
                "kms:CallerAccount": "111122223333"
            }
        }
    }
]
```

Permisos de IAM necesarios cuando al utilizar una clave de KMS administrada por el cliente

Al crear un banco de datos con el AWS KMS cifrado habilitado mediante una clave de KMS administrada por el cliente, se requieren permisos tanto para la política de claves como para la política de IAM para el usuario o rol que crea el banco de HealthLake datos.

Puede usar la clave de ViaService condición kms: para limitar el uso de la clave KMS únicamente a las solicitudes que se HealthLake originen en.

Para obtener más información sobre las políticas clave, consulte Habilitar las políticas de IAM en la Guía para desarrolladores de AWS Key Management Service.

El usuario de IAM, el rol de IAM o la cuenta de AWS que crea los repositorios debe tener los permisos kms:CreateGrant, kms: y kms:GenerateDataKey, además de DescribeKey los permisos necesarios. HealthLake

Cómo se HealthLake utilizan las subvenciones en AWS KMS

HealthLake requiere una concesión para usar la clave de KMS administrada por el cliente. Cuando crea un almacén de datos cifrado con una clave de KMS administrada por el cliente, HealthLake crea una concesión en su nombre mediante el envío de una CreateGrantsolicitud a AWS KMS. Las subvenciones en AWS KMS se utilizan para dar HealthLake acceso a una clave de KMS en una cuenta de cliente.

Las subvenciones que se HealthLake crean en su nombre no deben revocarse ni retirarse. Si revoca o retira la concesión que le da HealthLake permiso para usar las claves de AWS KMS de su cuenta, HealthLake no puede acceder a estos datos, cifra los nuevos recursos del FHIR enviados al almacén de datos o los descifra cuando se extraen. Cuando revoca o retira una subvención HealthLake, el cambio se produce inmediatamente. Para revocar los derechos de acceso, debe eliminar el almacén de datos en lugar de revocar la concesión. Cuando se elimina un almacén de datos, HealthLake retira las concesiones en tu nombre.

Supervisar sus claves de cifrado para HealthLake

Puede utilizarla CloudTrail para realizar un seguimiento de las solicitudes que se HealthLake envían AWS KMS en su nombre cuando utiliza una clave KMS administrada por el cliente. Las entradas del CloudTrail registro muestran healthlake.amazonaws.com en el campo UserAgent para distinguir claramente las solicitudes realizadas por. HealthLake

Los siguientes ejemplos son CloudTrail eventos para CreateGrant descifrar y supervisar las AWS KMS operaciones solicitadas para acceder DescribeKey a los datos cifrados por la clave gestionada por HealthLake el cliente. GenerateDataKey

A continuación, se muestra cómo se utiliza CreateGrant para permitir el acceso HealthLake a una clave de KMS proporcionada por el cliente, lo que HealthLake permite utilizar esa clave de KMS para cifrar todos los datos inactivos del cliente.

Los usuarios no están obligados a crear sus propias subvenciones. HealthLake crea una subvención en su nombre mediante el envío de una CreateGrant solicitud a AWS KMS. Las concesiones in se AWS KMS utilizan para dar HealthLake acceso a una AWS KMS clave de la cuenta de un cliente.


```
         {
"eventVersion": "1.08",
"userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLEROLE:Sampleuser01",
    "arn": "arn:aws:sts::111122223333:assumed-role/Sampleuser01,
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLEKEYID",
    "sessionContext": {
        "sessionIssuer": {
            "type": "Role",
            "principalId": "EXAMPLEROLE",
            "arn": "arn:aws:iam::111122223333:role/Sampleuser01",
            "accountId": "111122223333",
            "userName": "Sampleuser01"
        },
        "webIdFederationData": {},
        "attributes": {
            "creationDate": "2021-06-30T19:33:37Z",
            "mfaAuthenticated": "false"
        }
    },
    "invokedBy": "healthlake.amazonaws.com"
},
"eventTime": "2021-06-30T20:31:15Z",
"eventSource": "kms.amazonaws.com",
"eventName": "CreateGrant",
"awsRegion": "us-east-1",
"sourceIPAddress": "healthlake.amazonaws.com",
"userAgent": "healthlake.amazonaws.com",
```

```
    "requestParameters": {
        "operations": [
            "CreateGrant",
            "Decrypt",
            "DescribeKey",
            "Encrypt",
            "GenerateDataKey",
            "GenerateDataKeyWithoutPlaintext",
            "ReEncryptFrom",
            "ReEncryptTo",
            "RetireGrant"
        ],
        "granteePrincipal": "healthlake.us-east-1.amazonaws.com",
        "keyId": "arn:aws:kms:us-east-1:111122223333:key/EXAMPLE_KEY_ARN",
        "retiringPrincipal": "healthlake.us-east-1.amazonaws.com"
    },
    "responseElements": {
        "grantId": "EXAMPLE_ID_01"
    },
    "requestID": "EXAMPLE_ID_02",
    "eventID": "EXAMPLE_ID_03",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-east-1:111122223333:key/EXAMPLE_KEY_ARN"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

Los siguientes ejemplos muestran cómo se utilizan GenerateDataKey para garantizar que el usuario tenga los permisos necesarios para cifrar los datos antes de almacenarlos.

```
            {
    "eventVersion": "1.08",
    "userIdentity": {
```

```
        "type": "AssumedRole",
        "principalId": "EXAMPLEUSER",
        "arn": "arn:aws:sts::111122223333:assumed-role/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLEKEYID",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "EXAMPLEROLE",
                "arn": "arn:aws:iam::111122223333:role/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Sampleuser01"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2021-06-30T21:17:06Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "healthlake.amazonaws.com"
    },
    "eventTime": "2021-06-30T21:17:37Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "healthlake.amazonaws.com",
    "userAgent": "healthlake.amazonaws.com",
    "requestParameters": {
        "keySpec": "AES_256",
        "keyId": "arn:aws:kms:us-east-1:111122223333:key/EXAMPLE_KEY_ARN"
    },
    "responseElements": null,
    "requestID": "EXAMPLE_ID_01",
    "eventID": "EXAMPLE_ID_02",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-east-1:111122223333:key/EXAMPLE_KEY_ARN"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
```

```
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

El siguiente ejemplo muestra cómo se HealthLake llama a la operación de descifrado para utilizar la clave de datos cifrados almacenada para acceder a los datos cifrados.

```
         {
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EXAMPLEUSER",
        "arn": "arn:aws:sts::111122223333:assumed-role/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLEKEYID",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "EXAMPLEROLE",
                "arn": "arn:aws:iam::111122223333:role/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Sampleuser01"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2021-06-30T21:17:06Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "healthlake.amazonaws.com"
    },
    "eventTime": "2021-06-30T21:21:59Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "healthlake.amazonaws.com",
    "userAgent": "healthlake.amazonaws.com",
    "requestParameters": {
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "keyId": "arn:aws:kms:us-east-1:111122223333:key/EXAMPLE_KEY_ARN"
    },
```

```
    "responseElements": null,
    "requestID": "EXAMPLE_ID_01",
    "eventID": "EXAMPLE_ID_02",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-east-1:111122223333:key/EXAMPLE_KEY_ARN"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

En el siguiente ejemplo, se muestra cómo se HealthLake utiliza la DescribeKey operación para comprobar si la AWS KMS clave propiedad del AWS KMS cliente se encuentra en un estado utilizable y para ayudar al usuario a solucionar problemas si no funciona.

```
         {
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EXAMPLEUSER",
        "arn": "arn:aws:sts::111122223333:assumed-role/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLEKEYID",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "EXAMPLEROLE",
                "arn": "arn:aws:iam::111122223333:role/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Sampleuser01"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2021-07-01T18:36:14Z",
                "mfaAuthenticated": "false"
```

```
            }
        },
        "invokedBy": "healthlake.amazonaws.com"
    },
    "eventTime": "2021-07-01T18:36:36Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DescribeKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "healthlake.amazonaws.com",
    "userAgent": "healthlake.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-east-1:111122223333:key/EXAMPLE_KEY_ARN"
    },
    "responseElements": null,
    "requestID": "EXAMPLE_ID_01",
    "eventID": "EXAMPLE_ID_02",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-east-1:111122223333:key/EXAMPLE_KEY_ARN"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```


Más información

Los siguientes recursos proporcionan más información sobre el cifrado de datos en reposo.

Para obtener más información sobre los conceptos básicos de AWS Key Management Service, consulte la AWS KMS documentación.

Para obtener más información sobre las prácticas recomendadas de seguridad, AWS KMS consulte la documentación.

Cifrado en tránsito para AWS HealthLake

AWS HealthLake utiliza TLS 1.2 para cifrar los datos en tránsito a través del punto final público y a través de los servicios de backend.

Gestión de identidad y acceso para AWS HealthLake

AWS Identity and Access Management (IAM) es una herramienta Servicio de AWS que ayuda al administrador a controlar de forma segura el acceso a los AWS recursos. Los administradores de IAM controlan quién puede autenticarse (iniciar sesión) y quién puede autorizarse (tener permisos) para usar los recursos. HealthLake La IAM es una Servicio de AWS opción que puede utilizar sin coste adicional.

Temas

- Público
- Autenticación con identidades
- Administración del acceso con políticas
- ¿Cómo AWS HealthLake funciona con IAM
- Ejemplos de políticas basadas en la identidad para AWS HealthLake
- AWS políticas gestionadas para AWS HealthLake
- Solución de problemas de AWS HealthLake identidad y acceso

Público

La forma de usar AWS Identity and Access Management (IAM) varía según la función que desempeñes:

- Usuario del servicio: solicite permisos al administrador si no puede acceder a las características (consulte Solución de problemas de AWS HealthLake identidad y acceso).
- Administrador del servicio: determine el acceso de los usuarios y envíe las solicitudes de permiso (consulte ¿Cómo AWS HealthLake funciona con IAM).
- Administrador de IAM: escribe las políticas para administrar el acceso (consulte Ejemplos de políticas basadas en la identidad para AWS HealthLake).

Autenticación con identidades

La autenticación es la forma en que inicias sesión AWS con tus credenciales de identidad. Debe autenticarse como usuario de Usuario raíz de la cuenta de AWS IAM o asumir una función de IAM.

Puede iniciar sesión como una identidad federada con las credenciales de una fuente de identidad, como AWS IAM Identity Center (IAM Identity Center), la autenticación de inicio de sesión único o las credenciales. Google/Facebook Para obtener más información sobre el inicio de sesión, consulte Cómo iniciar sesión en la Cuenta de AWS en la Guía del usuario de AWS Sign-In .

Para el acceso programático, AWS proporciona un SDK y una CLI para firmar criptográficamente las solicitudes. Para obtener más información, consulte AWS Signature Version 4 para solicitudes de API en la Guía del usuario de IAM.

Cuenta de AWS usuario root

Al crear un Cuenta de AWS, se comienza con una identidad de inicio de sesión denominada usuario Cuenta de AWS raíz que tiene acceso completo a todos Servicios de AWS los recursos. Se recomiendaencarecidamente que no utilice el usuario raíz para las tareas diarias. Para ver las tareas que requieren credenciales de usuario raíz, consulte Tareas que requieren credenciales de usuario raíz en la Guía del usuario de IAM.

Identidad federada

Como práctica recomendada, exija a los usuarios humanos que utilicen la federación con un proveedor de identidades para acceder Servicios de AWS mediante credenciales temporales.

Una identidad federada es un usuario del directorio empresarial, del proveedor de identidades web o al Directory Service que se accede Servicios de AWS mediante credenciales de una fuente de identidad. Las identidades federadas asumen roles que proporcionan credenciales temporales.

Para una administración de acceso centralizada, se recomienda AWS IAM Identity Center. Para obtener más información, consulte ¿Qué es el Centro de identidades de IAM? en la Guía del usuario de AWS IAM Identity Center .

Usuarios y grupos de IAM

Un usuario de IAM es una identidad con permisos específicos para una sola persona o aplicación. Recomendamos el uso de credenciales temporales en lugar de usuarios de IAM con credenciales de larga duración. Para obtener más información, consulte Exigir a los usuarios humanos que utilicen la

federación con un proveedor de identidad para acceder AWS mediante credenciales temporales en la Guía del usuario de IAM.

Un grupo de IAM especifica un conjunto de usuarios de IAM y facilita la administración de los permisos para grupos grandes de usuarios. Para obtener más información, consulte Casos de uso para usuarios de IAM en la Guía del usuario de IAM.

Roles de IAM

Un Rol de IAM es una identidad con permisos específicos que proporciona credenciales temporales. Puede asumir un rol cambiando de un rol de usuario a uno de IAM (consola) o llamando a una AWS CLI operación de AWS API. Para obtener más información, consulte Métodos para asumir un rol en la Guía del usuario de IAM.

Los roles de IAM son útiles para el acceso de usuario federado, los permisos de usuario de IAM temporales, el acceso entre cuentas, el acceso entre servicios y las aplicaciones que se ejecutan en Amazon EC2. Para obtener más información, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

Administración del acceso con políticas

AWS Para controlar el acceso, puede crear políticas y adjuntarlas a AWS identidades o recursos. Una política define los permisos cuando están asociados a una identidad o un recurso. AWS evalúa estas políticas cuando un director hace una solicitud. La mayoría de las políticas se almacenan AWS como documentos JSON. Para obtener más información sobre los documentos de políticas de JSON, consulte Información general de políticas de JSON en la Guía del usuario de IAM.

Mediante las políticas, los administradores especifican quién tiene acceso a qué, definiendo qué entidad principal puede realizar acciones sobre qué recursos y en qué condiciones.

De forma predeterminada, los usuarios y los roles no tienen permisos. Un administrador de IAM crea políticas de IAM y las agrega a roles, que los usuarios pueden asumir posteriormente. Las políticas de IAM definen permisos independientemente del método que se utilice para realizar la operación.

Políticas basadas en identidades

Las políticas basadas en identidad son documentos de política de permisos JSON que asocia a una identidad (usuario, grupo o rol). Estas políticas controlan qué acciones pueden realizar las identidades, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear

una política basada en la identidad, consulte Definición de permisos de IAM personalizados con políticas administradas por el cliente en la Guía del usuario de IAM.

Las políticas basadas en identidad pueden ser políticas insertadas (incrustadas directamente en una sola identidad) o políticas administradas (políticas independientes asociadas a varias identidades). Para obtener información sobre cómo elegir entre políticas administradas e insertadas, consulte Selección entre políticas administradas y políticas insertadas en la Guía del usuario de IAM.

Políticas basadas en recursos

Las políticas basadas en recursos son documentos de políticas JSON que se asocian a un recurso. Los ejemplos incluyen las Políticas de confianza de roles de IAM y las Políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Debe especificar una entidad principal en una política basada en recursos.

Las políticas basadas en recursos son políticas insertadas que se encuentran en ese servicio. No puedes usar políticas AWS gestionadas de IAM en una política basada en recursos.

Otros tipos de políticas

AWS admite tipos de políticas adicionales que pueden establecer los permisos máximos que conceden los tipos de políticas más comunes:

- Límites de permisos: establecen los permisos máximos que una política basada en identidad puede conceder a una entidad de IAM. Para obtener más información, consulte Límites de permisos para las entidades de IAM en la Guía del usuario de IAM.
- Políticas de control de servicios (SCPs): especifican los permisos máximos para una organización o unidad organizativa en AWS Organizations. Para obtener más información, consulte Políticas de control de servicios en la Guía del usuario de AWS Organizations .
- Políticas de control de recursos (RCPs): establece los permisos máximos disponibles para los recursos de tus cuentas. Para obtener más información, consulte Políticas de control de recursos (RCPs) en la Guía del AWS Organizations usuario.
- Políticas de sesión: políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal para un rol o un usuario federado. Para obtener más información, consulte Políticas de sesión en la Guía del usuario de IAM.

Varios tipos de políticas

Cuando se aplican varios tipos de políticas a una solicitud, los permisos resultantes son más complicados de entender. Para saber cómo se AWS determina si se debe permitir una solicitud cuando se trata de varios tipos de políticas, consulte la lógica de evaluación de políticas en la Guía del usuario de IAM.

¿Cómo AWS HealthLake funciona con IAM

Antes de utilizar IAM para gestionar el acceso HealthLake, infórmese sobre las funciones de IAM disponibles para su uso. HealthLake

Funciones de IAM que puede utilizar con AWS HealthLake

Característica de IAM	HealthLake soporte
Políticas basadas en identidades	Sí
Políticas basadas en recursos	No
Acciones de políticas	Sí
Recursos de políticas	Sí
Claves de condición de política	Sí
ACLs	No
ABAC (etiquetas en políticas)	Sí
Credenciales temporales	Sí
Permisos de entidades principales	Sí
Roles de servicio	Sí
Roles vinculados al servicio	No

Para obtener una visión general de cómo HealthLake funcionan otros AWS servicios con la mayoría de las funciones de IAM, consulte AWS los servicios que funcionan con IAM en la Guía del usuario de IAM.

Políticas basadas en la identidad para AWS HealthLake

Compatibilidad con las políticas basadas en identidad: sí

Las políticas basadas en identidad son documentos de políticas de permisos JSON que puede asociar a una identidad, como un usuario de IAM, un grupo de usuarios o un rol. Estas políticas controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en la identidad, consulte Definición de permisos de IAM personalizados con políticas administradas por el cliente en la Guía del usuario de IAM.

Con las políticas basadas en identidades de IAM, puede especificar las acciones y los recursos permitidos o denegados, así como las condiciones en las que se permiten o deniegan las acciones. Para obtener más información sobre los elementos que puede utilizar en una política de JSON, consulte Referencia de los elementos de la política de JSON de IAM en la Guía del usuario de IAM.

Ejemplos de políticas basadas en la identidad para AWS HealthLake

Para ver ejemplos de políticas HealthLake basadas en la identidad, consulte. Ejemplos de políticas basadas en la identidad para AWS HealthLake

Políticas basadas en recursos dentro de AWS HealthLake

Admite políticas basadas en recursos: no

Las políticas basadas en recursos son documentos de política JSON que se asocian a un recurso. Los ejemplos de políticas basadas en recursos son las políticas de confianza de roles de IAM y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Para el recurso al que se asocia la política, la política define qué acciones puede realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe especificar una entidad principal en una política basada en recursos. Los principales pueden incluir cuentas, usuarios, roles, usuarios federados o. Servicios de AWS

Para habilitar el acceso entre cuentas, puede especificar toda una cuenta o entidades de IAM de otra cuenta como la entidad principal de una política en función de recursos. Para obtener más información, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

Acciones políticas para AWS HealthLake

Compatibilidad con las acciones de políticas: sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Action` de una política JSON describe las acciones que puede utilizar para conceder o denegar el acceso en una política. Incluya acciones en una política para conceder permisos y así llevar a cabo la operación asociada.

Para ver una lista de HealthLake acciones, consulte las acciones definidas AWS HealthLake en la Referencia de autorización del servicio.

Las acciones políticas HealthLake utilizan el siguiente prefijo antes de la acción:

```
healthlake
```

Para especificar varias acciones en una única sentencia, separe cada acción con una coma.

```
"Action": [
      "healthlake:action1",
      "healthlake:action2"
         ]
```

Para ver ejemplos de políticas HealthLake basadas en la identidad, consulte. Ejemplos de políticas basadas en la identidad para AWS HealthLake

Recursos de políticas para AWS HealthLake

Compatibilidad con los recursos de políticas: sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Resource` de la política JSON especifica el objeto u objetos a los que se aplica la acción. Como práctica recomendada, especifique un recurso utilizando el Nombre de recurso de Amazon (ARN). En el caso de las acciones que no admiten permisos por recurso, utilice un carácter comodín (*) para indicar que la instrucción se aplica a todos los recursos.

```
"Resource": "*"
```

Para ver una lista de los tipos de HealthLake recursos y sus tipos ARNs, consulte los recursos definidos AWS HealthLake en la Referencia de autorización de servicios. Para conocer las acciones con las que puede especificar el ARN de cada recurso, consulte Acciones definidas por. AWS HealthLake

Para ver ejemplos de políticas HealthLake basadas en la identidad, consulte. Ejemplos de políticas basadas en la identidad para AWS HealthLake

Claves de condición de la política para AWS HealthLake

Compatibilidad con claves de condición de políticas específicas del servicio: sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Condition` especifica cuándo se ejecutan las instrucciones en función de criterios definidos. Puede crear expresiones condicionales que utilizan operadores de condición, tales como igual o menor que, para que la condición de la política coincida con los valores de la solicitud. Para ver todas las claves de condición AWS globales, consulte las claves de contexto de condición AWS globales en la Guía del usuario de IAM.

Para ver una lista de claves de HealthLake condición, consulte las claves de condición AWS HealthLake en la Referencia de autorización de servicio. Para obtener información sobre las acciones y los recursos con los que puede utilizar una clave de condición, consulte Acciones definidas por AWS HealthLake.

Para ver ejemplos de políticas HealthLake basadas en la identidad, consulte. Ejemplos de políticas basadas en la identidad para AWS HealthLake

Listas de control de acceso () ACLs en AWS HealthLake

Soporta ACLs: No

Las listas de control de acceso (ACLs) controlan qué directores (miembros de la cuenta, usuarios o roles) tienen permisos para acceder a un recurso. ACLs son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de políticas JSON.

Control de acceso basado en atributos (ABAC) con AWS HealthLake

Admite ABAC (etiquetas en las políticas): sí

El control de acceso basado en atributos (ABAC) es una estrategia de autorización que define permisos en función de atributos denominados etiquetas. Puede adjuntar etiquetas a las entidades y AWS los recursos de IAM y, a continuación, diseñar políticas de ABAC para permitir las operaciones cuando la etiqueta del principal coincida con la etiqueta del recurso.

Para controlar el acceso en función de etiquetas, debe proporcionar información de las etiquetas en el elemento de condición de una política utilizando las claves de condición `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` o `aws:TagKeys`.

Si un servicio admite las tres claves de condición para cada tipo de recurso, el valor es Sí para el servicio. Si un servicio admite las tres claves de condición solo para algunos tipos de recursos, el valor es Parcial.

Para obtener más información sobre ABAC, consulte Definición de permisos con la autorización de ABAC en la Guía del usuario de IAM. Para ver un tutorial con los pasos para configurar ABAC, consulte Uso del control de acceso basado en atributos (ABAC) en la Guía del usuario de IAM.

Utilizar credenciales temporales con AWS HealthLake

Compatibilidad con credenciales temporales: sí

Las credenciales temporales proporcionan acceso a AWS los recursos a corto plazo y se crean automáticamente al utilizar la federación o al cambiar de función. AWS recomienda generar credenciales temporales de forma dinámica en lugar de utilizar claves de acceso a largo plazo. Para obtener más información, consulte Credenciales de seguridad temporales en IAM y Servicios de AWS que funcionan con IAM en la Guía del usuario de IAM.

Permisos principales entre servicios para AWS HealthLake

Admite sesiones de acceso directo (FAS): sí

Las sesiones de acceso directo (FAS) utilizan los permisos de la persona principal que llama Servicio de AWS, junto con la solicitud, Servicio de AWS para realizar solicitudes a los servicios descendentes. Para obtener información sobre las políticas a la hora de realizar solicitudes de FAS, consulte Sesiones de acceso directo.

Funciones de servicio para AWS HealthLake

Compatible con roles de servicio: sí

Un rol de servicio es un rol de IAM que asume un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio desde IAM. Para obtener más información, consulte Crear un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM.

Para obtener información sobre las funciones de servicio y la política en línea necesaria para el acceso completo a ellas AWS HealthLake, consulteConfiguración AWS HealthLake.

> **Warning**
>
> Cambiar los permisos de un rol de servicio podría afectar a la HealthLake funcionalidad. Edite las funciones de servicio solo cuando se HealthLake proporcionen instrucciones para hacerlo.

Funciones vinculadas al servicio para AWS HealthLake

Compatibilidad con roles vinculados al servicio: no

Un rol vinculado a un servicio es un tipo de rol de servicio que está vinculado a un. Servicio de AWS El servicio puede asumir el rol para realizar una acción en su nombre. Los roles vinculados al servicio aparecen en usted Cuenta de AWS y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.

Para más información sobre cómo crear o administrar roles vinculados a servicios, consulta Servicios de AWS que funcionan con IAM. Busque un servicio en la tabla que incluya `Yes` en la columna Rol vinculado a un servicio. Seleccione el vínculo Sí para ver la documentación acerca del rol vinculado a servicios para ese servicio.

Ejemplos de políticas basadas en la identidad para AWS HealthLake

De forma predeterminada, los usuarios y roles no tienen permiso para crear, ver ni modificar recursos de HealthLake. Un administrador de IAM puede crear políticas de IAM para conceder permisos a los usuarios para realizar acciones en los recursos que necesitan.

Para obtener información acerca de cómo crear una política basada en identidades de IAM mediante el uso de estos documentos de políticas JSON de ejemplo, consulte Creación de políticas de IAM (consola) en la Guía del usuario de IAM.

Para obtener más información sobre las acciones y los tipos de recursos definidos por ellos HealthLake, incluido el ARNs formato de cada uno de los tipos de recursos, consulte las claves de acción, recursos y condición de la Referencia AWS HealthLake de autorización de servicios.

Temas

- Prácticas recomendadas sobre las políticas
- Uso de la consola AWS HealthLake
- Acceder a un almacén de AWS HealthLake datos en Amazon Athena
- Cómo permitir a los usuarios que vean sus propios permisos

Prácticas recomendadas sobre las políticas

Las políticas basadas en la identidad determinan si alguien puede crear HealthLake recursos de tu cuenta, acceder a ellos o eliminarlos. Estas acciones pueden generar costos adicionales para su Cuenta de AWS. Siga estas directrices y recomendaciones al crear o editar políticas basadas en identidades:

- Comience con las políticas AWS administradas y avance hacia los permisos con privilegios mínimos: para empezar a conceder permisos a sus usuarios y cargas de trabajo, utilice las políticas AWS administradas que otorgan permisos para muchos casos de uso comunes. Están disponibles en su. Cuenta de AWS Le recomendamos que reduzca aún más los permisos definiendo políticas administradas por el AWS cliente que sean específicas para sus casos de uso. Con el fin de obtener más información, consulte las políticas administradas por AWS o las políticas administradas por AWS para funciones de tarea en la Guía de usuario de IAM.
- Aplique permisos de privilegio mínimo: cuando establezca permisos con políticas de IAM, conceda solo los permisos necesarios para realizar una tarea. Para ello, debe definir las acciones que se pueden llevar a cabo en determinados recursos en condiciones específicas, también conocidos como permisos de privilegios mínimos. Con el fin de obtener más información sobre el uso de IAM para aplicar permisos, consulte Políticas y permisos en IAM en la Guía del usuario de IAM.
- Utilice condiciones en las políticas de IAM para restringir aún más el acceso: puede agregar una condición a sus políticas para limitar el acceso a las acciones y los recursos. Por ejemplo, puede escribir una condición de políticas para especificar que todas las solicitudes deben enviarse utilizando SSL. También puedes usar condiciones para conceder el acceso a las acciones del servicio si se utilizan a través de una acción específica Servicio de AWS, por ejemplo CloudFormation. Para obtener más información, consulte Elementos de la política de JSON de IAM: Condición en la Guía del usuario de IAM.

- Utiliza el analizador de acceso de IAM para validar las políticas de IAM con el fin de garantizar la seguridad y funcionalidad de los permisos: el analizador de acceso de IAM valida políticas nuevas y existentes para que respeten el lenguaje (JSON) de las políticas de IAM y las prácticas recomendadas de IAM. El analizador de acceso de IAM proporciona más de 100 verificaciones de políticas y recomendaciones procesables para ayudar a crear políticas seguras y funcionales. Para más información, consulte Validación de políticas con el Analizador de acceso de IAM en la Guía del usuario de IAM.
- Requerir autenticación multifactor (MFA): si tiene un escenario que requiere usuarios de IAM o un usuario raíz en Cuenta de AWS su cuenta, active la MFA para mayor seguridad. Para exigir la MFA cuando se invoquen las operaciones de la API, añada condiciones de MFA a sus políticas. Para más información, consulte Acceso seguro a la API con MFA en la Guía del usuario de IAM.

Para obtener más información sobre las prácticas recomendadas de IAM, consulte Prácticas recomendadas de seguridad en IAM en la Guía del usuario de IAM.

Uso de la consola AWS HealthLake

Para acceder a la AWS HealthLake consola, debe tener un conjunto mínimo de permisos. Estos permisos deben permitirle enumerar y ver detalles sobre los HealthLake recursos de su cuenta Cuenta de AWS. Si crea una política basada en identidades que sea más restrictiva que el mínimo de permisos necesarios, la consola no funcionará del modo esperado para las entidades (usuarios o roles) que tengan esa política.

No es necesario que concedas permisos mínimos de consola a los usuarios que solo realicen llamadas a la API AWS CLI o a la AWS API. En su lugar, permita el acceso únicamente a las acciones que coincidan con la operación de API que intentan realizar.

Para tener acceso completo a HealthLake, asocie las siguientes políticas a un usuario o rol de IAM: `AmazonHealthLakeFullAccess` y`AWSLakeFormationDataAdmin`. También debe adjuntar la política HealthLake en línea, que es una función de servicio. Un rol de servicio es un rol de IAM que asume un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio desde IAM. Para obtener más información, consulte Crear un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM. Para obtener información sobre la política en línea que crea el rol de servicio requerido, consulte. Configuración AWS HealthLake También debe usar la AWS Lake Formation consola o la CLI para asignar a su HealthLake administrador como administrador de AWS Lake Formation Data Lake. Para obtener más información, consulte Configuración AWS HealthLake.

Acceder a un almacén de AWS HealthLake datos en Amazon Athena

Si desea proporcionar a los usuarios y roles acceso a los almacenes de HealthLake datos Amazon Athena, adjunte las siguientes políticas de IAM al rol o usuario: `AmazonAthenaFullAccess` y`AmazonS3FullAccess`. `Select`y también se requieren `Describe` permisos en las tablas administradas por AWS Lake Formation. AWS Lake Formation los permisos de tabla los concede un AWS Lake Formation administrador en la AWS Lake Formation consola o mediante la CLI. Para obtener más información, consulte Configuración AWS HealthLake

Cómo permitir a los usuarios que vean sus propios permisos

En este ejemplo, se muestra cómo podría crear una política que permita a los usuarios de IAM ver las políticas administradas e insertadas que se asocian a la identidad de sus usuarios. Esta política incluye permisos para completar esta acción en la consola o mediante programación mediante la API AWS CLI o AWS .

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
```

```
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS políticas gestionadas para AWS HealthLake

Una política AWS administrada es una política independiente creada y administrada por AWS. AWS Las políticas administradas están diseñadas para proporcionar permisos para muchos casos de uso comunes, de modo que pueda empezar a asignar permisos a usuarios, grupos y funciones.

Ten en cuenta que es posible que las políticas AWS administradas no otorguen permisos con privilegios mínimos para tus casos de uso específicos, ya que están disponibles para que los usen todos los AWS clientes. Se recomienda definir políticas administradas por el cliente específicas para sus casos de uso a fin de reducir aún más los permisos.

No puedes cambiar los permisos definidos en AWS las políticas administradas. Si AWS actualiza los permisos definidos en una política AWS administrada, la actualización afecta a todas las identidades principales (usuarios, grupos y roles) a las que está asociada la política. AWS es más probable que actualice una política AWS administrada cuando Servicio de AWS se lance una nueva o cuando estén disponibles nuevas operaciones de API para los servicios existentes.

Para obtener más información, consulte Políticas administradas por AWS en la Guía del usuario de IAM.

AWS política gestionada: AmazonHealthLakeFullAccess

La `AmazonHealthLakeFullAccess` política proporciona acceso completo a HealthLake. Con esta política asociada a su usuario o función, los usuarios pueden utilizarla HealthLake para acceder a los datos, consultarlos, importarlos y exportarlos HealthLake. Para realizar muchas de las acciones habituales HealthLake, debe añadir políticas adicionales al usuario o al rol. Para obtener más información, consulte HealthLake operaciones Configuración AWS HealthLake y permisos.

Puede asociar la política `AmazonHealthLakeFullAccess` a las identidades de IAM.

Esta política concede permisos de administrador y de colaborador que permiten a los usuarios y a los roles consultar, buscar HealthLake, importar y exportar, y también permite HealthLake realizar acciones en nombre de los usuarios y roles que tienen estos permisos.

Detalles de los permisos

Esta política incluye la siguiente declaración.

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Action": [
    "healthlake:*",
    "s3:ListAllMyBuckets",
    "s3:ListBucket",
    "s3:GetBucketLocation",
    "iam:ListRoles"
   ],
   "Resource": "*",
   "Effect": "Allow"
  },
  {
   "Effect": "Allow",
   "Action": "iam:PassRole",
   "Resource": "*",
   "Condition": {
    "StringEquals": {
     "iam:PassedToService": "healthlake.amazonaws.com"
    }
   }
  }
 ]
}
```

AWS política gestionada: AmazonHealthLakeReadOnlyAccess

`AmazonHealthLakeReadOnlyAccess`la política otorga acceso y permisos de solo lectura a otros servicios HealthLake y a los recursos relacionados en otros AWS servicios. Aplique esta política a los usuarios a los que desee conceder la posibilidad de consultar y ver el almacén de HealthLake datos, pero no la posibilidad de crearlos o realizar cambios en ellos.

Puede asociar la política `AmazonHealthLakeReadOnlyAccess` a las identidades de IAM.

Esta política otorga *`read-only`* permisos que permiten a los usuarios y a los roles realizar consultas HealthLake.

Detalles de los permisos

Esta política incluye la siguiente declaración.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "healthlake:ListFHIRDatastores",
                "healthlake:DescribeFHIRDatastore",
                "healthlake:DescribeFHIRImportJob",
                "healthlake:DescribeFHIRExportJob",
                "healthlake:GetCapabilities",
                "healthlake:ReadResource",
                "healthlake:SearchWithGet",
                "healthlake:SearchWithPost",
                "healthlake:SearchEverything"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

HealthLake operaciones y permisos

En la siguiente tabla se enumeran las operaciones típicas HealthLake y los permisos necesarios para realizarlas.

HealthLake operaciones	Permisos necesarios
Cree un almacén de datos en HealthLake	`AmazonHealthLakeFullAccess` `AmazonLakeFormationDataAdmin` , la política en línea y los permisos de AWS Lake Formation administrador gestionados por AWS Lake Formation
Eliminar un almacén de datos en HealthLake	`AmazonHealthLakeFullAccess` `AmazonLakeFormationDataAdmin` , la política en línea y los permisos de AWS Lake Formation administrador gestionados por AWS Lake Formation
Enumere, busque o consulte un almacén de datos en HealthLake	`AmazonHealthLakeReadOnlyAccess`
Consulte un banco de datos mediante Amazon Athena	`AmazonAthenaFullAccess` `AmazonS3FullAccess` , AWS Lake Formation `Select` y `Describe` permisos en las tablas administradas por AWS Lake Formation
Importar datos de HealthLake	Consulte Configurar los permisos para los trabajos de importación.
Exportar datos desde HealthLake	Consulte Configurar los permisos para los trabajos de exportación.

HealthLake actualizaciones de las políticas AWS gestionadas

Vea los detalles sobre las actualizaciones de las políticas AWS administradas HealthLake desde el momento en que este servicio comenzó a rastrear estos cambios. Para recibir alertas automáticas

sobre los cambios en esta página, suscríbase a la fuente RSS de la página del historial del HealthLake documento.

Cambio	Descripción	Fecha
AmazonHealthLakeFullAccess	`AmazonHealthLakeFullAccess` política necesaria para permitir el acceso completo a HealthLake.	14 de noviembre de 2022
AmazonHealthLakeReadOnlyAccess	`AmazonHealthLakeReadOnlyAccess` política requerida para el acceso de solo lectura a. HealthLake	14 de noviembre de 2022
HealthLake comenzó a rastrear los cambios	HealthLake comenzó a realizar un seguimiento de los cambios de sus políticas AWS gestionadas.	14 de noviembre de 2022

Solución de problemas de AWS HealthLake identidad y acceso

Utilice la siguiente información como ayuda para diagnosticar y solucionar los problemas habituales que pueden surgir al trabajar con un HealthLake IAM.

Temas

- No estoy autorizado a realizar ninguna acción en AWS HealthLake
- No estoy autorizado a realizar lo siguiente: PassRole
- Quiero permitir que personas ajenas a mi AWS cuenta accedan a mis AWS HealthLake recursos

No estoy autorizado a realizar ninguna acción en AWS HealthLake

Si Consola de administración de AWS le indica que no está autorizado a realizar una acción, debe ponerse en contacto con su administrador para obtener ayuda. El administrador es la persona que le facilitó el nombre de usuario y la contraseña.

En el siguiente ejemplo, el error se produce cuando el usuario de IAM `mateojackson` intenta utilizar la consola para consultar los detalles acerca de un recurso ficticio *`my-example-widget`*, pero no tiene los permisos ficticios `healthlake:`*`GetWidget`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 healthlake:GetWidget on resource: my-example-widget
```

En este caso, Mateo pide a su administrador que actualice sus políticas de forma que pueda obtener acceso al recurso *`my-example-widget`* mediante la acción `healthlake:`*`GetWidget`*.

No estoy autorizado a realizar lo siguiente: PassRole

Si recibe un error que indica que no tiene autorización para realizar la acción `iam:PassRole`, las políticas deben actualizarse a fin de permitirle pasar un rol a HealthLake.

Algunas Servicios de AWS permiten transferir una función existente a ese servicio en lugar de crear una nueva función de servicio o una función vinculada a un servicio. Para ello, debe tener permisos para transferir la función al servicio.

En el siguiente ejemplo, el error se produce cuando un usuario de IAM denominado `marymajor` intenta utilizar la consola para realizar una acción en HealthLake. Sin embargo, la acción requiere que el servicio cuente con permisos que otorguen un rol de servicio. Mary no tiene permisos para transferir el rol al servicio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

En este caso, las políticas de Mary se deben actualizar para permitirle realizar la acción `iam:PassRole`.

Si necesita ayuda, póngase en contacto con su AWS administrador. El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

Quiero permitir que personas ajenas a mi AWS cuenta accedan a mis AWS HealthLake recursos

Se puede crear un rol que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder a sus recursos. Se puede especificar una persona de confianza para

que asuma el rol. En el caso de los servicios que admiten políticas basadas en recursos o listas de control de acceso (ACLs), puedes usar esas políticas para permitir que las personas accedan a tus recursos.

Para obtener más información, consulte lo siguiente:

- Para saber si HealthLake es compatible con estas funciones, consulte. ¿Cómo AWS HealthLake funciona con IAM
- Para obtener información sobre cómo proporcionar acceso a los recursos de su Cuentas de AWS propiedad, consulte Proporcionar acceso a un usuario de IAM en otro usuario de su propiedad Cuenta de AWS en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso a tus recursos a terceros Cuentas de AWS, consulta Cómo proporcionar acceso a recursos que Cuentas de AWS son propiedad de terceros en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso mediante una federación de identidades, consulte Proporcionar acceso a usuarios autenticados externamente (identidad federada) en la Guía del usuario de IAM.
- Para conocer sobre la diferencia entre las políticas basadas en roles y en recursos para el acceso entre cuentas, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

Validación de conformidad para AWS HealthLake

Third-party los auditores evalúan la seguridad y el cumplimiento AWS HealthLake como parte de varios programas de AWS cumplimiento. Para HealthLake ello incluye la HIPAA.

Para saber si un programa de cumplimiento Servicio de AWS está dentro del ámbito de aplicación de programas de cumplimiento específicos, consulte Servicios de AWS Alcance por programa de cumplimiento Servicios de AWS y elija el programa de cumplimiento que le interese. Para obtener información general, consulte Programas de AWS cumplimiento > Programas AWS .

Puede descargar informes de auditoría de terceros utilizando AWS Artifact. Para obtener más información, consulte Descarga de informes en AWS Artifact .

Su responsabilidad de cumplimiento al Servicios de AWS utilizarlos viene determinada por la confidencialidad de sus datos, los objetivos de cumplimiento de su empresa y las leyes y reglamentos aplicables. Para obtener más información sobre su responsabilidad de conformidad al utilizarlos Servicios de AWS, consulte AWS la documentación de seguridad.

Seguridad de la infraestructura en AWS HealthLake

Como servicio gestionado, AWS HealthLake está protegido por los procedimientos de seguridad de red AWS global que se describen en el documento técnico Amazon Web Services: Overview of Security Processes.

Utiliza las llamadas a la API AWS publicadas para acceder a HealthLake través de la red. Los clientes deben ser compatibles con la seguridad de la capa de transporte (TLS) 1.0 o una versión posterior. Recomendamos TLS 1.2 o una versión posterior. Los clientes también deben admitir conjuntos de cifrado con perfecto secreto directo (PFS), como Ephemeral (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Diffie-Hellman La mayoría de los sistemas modernos como Java 7 y posteriores son compatibles con estos modos.

Además, las solicitudes deben estar firmadas mediante un ID de clave de acceso y una clave de acceso secreta que esté asociada a una entidad principal de IAM. También puedes utilizar AWS Security Token Service (AWS STS) para generar credenciales de seguridad temporales para firmar solicitudes.

Creación de AWS HealthLake recursos con AWS CloudFormation

AWS HealthLake está integrado con AWS CloudFormation un servicio que le ayuda a modelar y configurar sus AWS recursos para que pueda dedicar menos tiempo a crear y administrar sus recursos e infraestructura. Crea una plantilla que describe todos los AWS recursos que desea y CloudFormation aprovisiona y configura esos recursos por usted.

Cuando la utilice CloudFormation, podrá reutilizar la plantilla para configurar los HealthLake recursos de forma coherente y repetida. Describa sus recursos una vez y, a continuación, aprovisione los mismos recursos una y otra vez en varias Cuentas de AWS regiones.

HealthLake y CloudFormation plantillas

Para aprovisionar y configurar los recursos HealthLake y los servicios relacionados, debe conocer CloudFormation las plantillas. Las plantillas son archivos de texto con formato JSON o YAML. Estas plantillas describen los recursos que desea aprovisionar en sus CloudFormation pilas. Si no estás familiarizado con JSON o YAML, puedes usar CloudFormation Designer para ayudarte a empezar con CloudFormation las plantillas. Para obtener más información, consulte ¿Qué es Designer de CloudFormation ? en la Guía del usuario de AWS CloudFormation .

> **Note**
>
> AWS HealthLake admite la creación de almacenes de datos con CloudFormation. Para obtener más información, incluidos ejemplos de plantillas JSON y YAML para el aprovisionamiento de almacenes de HealthLake datos, consulta la referencia sobre los tipos de AWS HealthLake recursos en la Guía del AWS CloudFormation usuario.

Más información sobre CloudFormation

Para obtener más información CloudFormation, consulte los siguientes recursos:

- AWS CloudFormation
- AWS CloudFormation Guía del usuario
- Referencia de la API de CloudFormation
- AWS CloudFormation Guía del usuario de la interfaz de línea de comandos

AWS HealthLake y puntos finales de VPC de interfaz ()AWS PrivateLink

Puede establecer una conexión privada entre su VPC y crear un punto final de AWS HealthLake la VPC de interfaz. Los puntos finales de VPC de interfaz funcionan con una tecnología que puede utilizar para acceder de forma privada HealthLake; APIs sin una puerta de enlace a Internet, un dispositivo NAT, una conexión VPN o una conexión. AWS PrivateLink Direct Connect Las instancias de su VPC no necesitan direcciones IP públicas para comunicarse con HealthLake ellas;. APIs El tráfico entre tu VPC y HealthLake; no sale de la red de Amazon.

Cada punto de conexión de la interfaz está representado por una o más interfaces de red elásticas en las subredes.

Para obtener más información, consulte Interface VPC Endpoints (AWS PrivateLink) en la Guía del usuario de Amazon VPC.

Consideraciones sobre los puntos HealthLake finales de VPC

Antes de configurar un punto de enlace de VPC de interfaz HealthLake, asegúrese de revisar las propiedades y limitaciones del punto de enlace de interfaz en la Guía del usuario de Amazon VPC.

HealthLake admite realizar llamadas a todas sus acciones de API desde su VPC.

Crear un punto final de VPC de interfaz para; HealthLake

Puede crear un punto de enlace de VPC para el servicio HealthLake; mediante la consola de Amazon VPC o el (). AWS Command Line Interface AWS CLI Para obtener más información, consulte Creación de un punto de conexión de interfaz en la Guía del usuario de Amazon VPC.

Cree un punto final de VPC para HealthLake; con el siguiente nombre de servicio:

- com.amazonaws. *region*.healthlake

Si activas el DNS privado para el punto final, puedes realizar solicitudes a la API para HealthLake utilizar su nombre de DNS predeterminado para la región. Por ejemplo, *healthlake.us-east-1.amazonaws.com*.

Para más información, consulte Acceso a un servicio a través de un punto de conexión de interfaz en la Guía del usuario de Amazon VPC.

Crear una política de puntos de conexión de VPC para HealthLake

Puede asociar una política de punto de conexión con su punto de conexión de VPC que controla el acceso a HealthLake. La política especifica la siguiente información:

- La entidad principal que puede realizar acciones.
- Las acciones que se pueden realizar.
- Los recursos en los que se pueden llevar a cabo las acciones.

Para más información, consulte Control del acceso a los servicios con puntos de enlace de la VPC en la Guía del usuario de Amazon VPC.

Ejemplo: política de puntos finales de VPC para acciones HealthLake

El siguiente es un ejemplo de una política de puntos finales para HealthLake. Cuando se adjunta a un punto final, esta política otorga acceso a la HealthLake `CreateFHIRDatastore` acción a todos los directores de todos los recursos.

```
{
```

```
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "healthlake:create-fhir-datastore"
         ],
         "Resource":"*"
      }
   ]
}
```

Mejores prácticas de seguridad en AWS HealthLake

AWS HealthLake proporciona una serie de características de seguridad que debe tener en cuenta a la hora de desarrollar e implementar sus propias políticas de seguridad. Las siguientes prácticas recomendadas son directrices generales y no constituyen una solución de seguridad completa. Puesto que es posible que estas prácticas recomendadas no sean adecuadas o suficientes para el entorno, considérelas como consideraciones útiles en lugar de como normas.

- Implemente el acceso a los privilegios mínimos.
- Siempre que sea posible, utilice Customer-Managed-Keys (CMK) para cifrar sus datos. Para obtener más información sobre las CMK, consulte Amazon Key Management Service.
- Utilice Buscar con POST, no con GET, cuando busque PHI o PII en su almacén de datos.
- Limite el acceso a las funciones de auditoría importantes y confidenciales.
- Al crear recursos mediante las API de actualización o importación masiva, no utilice la PHI ni la PII, incluidos los nombres de los almacenes de datos y los trabajos, en ningún campo visible o en el identificador lógico del FHIR (LID).
- Al enviar solicitudes de creación, lectura, actualización, eliminación o búsqueda, no utilice la PHI en el encabezado HTTP.
- AWS CloudTrail Actívela para auditar el AWS HealthLake uso y garantizar que no haya actividad inesperada.
- Revise las prácticas recomendadas para utilizar los buckets de Amazon S3 de forma segura. Para obtener más información, consulte las prácticas recomendadas de seguridad en la guía del usuario de Amazon S3.

Resiliencia en AWS HealthLake

La infraestructura AWS global se basa en AWS regiones y zonas de disponibilidad. AWS Las regiones proporcionan varias zonas de disponibilidad aisladas y separadas físicamente, que están conectadas mediante redes de baja latencia, alto rendimiento y alta redundancia. Con las zonas de disponibilidad, puede diseñar y utilizar aplicaciones y bases de datos que realizan una conmutación por error automática entre las zonas sin interrupciones. Las zonas de disponibilidad tienen una mayor disponibilidad, tolerancia a errores y escalabilidad que las infraestructuras tradicionales de uno o varios centros de datos.

Si necesita replicar sus datos o aplicaciones sobre grandes distancias geográficas, utilice las regiones locales de AWS . Una región AWS local es un centro de datos único diseñado para complementar una región existente. AWS Como todas AWS las regiones, las regiones AWS locales están completamente aisladas de otras AWS regiones.

Para obtener más información sobre AWS las regiones y las zonas de disponibilidad, consulte Infraestructura AWS global.

AWS HealthLake referencia

El siguiente material de referencia de apoyo está disponible para SMART sobre el FHIR, el FHIR y. AWS HealthLake

> **Note**
>
> Todas las HealthLake acciones y tipos de datos nativos se describen en una referencia independiente. Para obtener más información, consulte la Referencia de la API de AWS HealthLake .

Temas

- SMART sobre el apoyo del FHIR a AWS HealthLake
- Soporte FHIR R4 para AWS HealthLake
- Referencia de cumplimiento para AWS HealthLake
- Referencia de soporte para AWS HealthLake

SMART sobre el apoyo del FHIR a AWS HealthLake

Un almacén de HealthLake datos habilitado para aplicaciones médicas y tecnologías reutilizables (SMART) compatible con el FHIR permite el acceso a aplicaciones compatibles con SMART en el FHIR. HealthLake Se accede a los datos autenticando y autorizando las solicitudes mediante un servidor de autorización externo. Por lo tanto, en lugar de administrar las credenciales de los usuarios mediante AWS Identity and Access Management, lo hace utilizando un servidor de autorización compatible con SMART on FHIR.

> **Note**
>
> HealthLake es compatible con SMART en las versiones 1.0 y 2.0 del FHIR. Para obtener más información sobre estos marcos, consulte el lanzamiento de la aplicación SMART en la documentación del FHIR R4.
> HealthLake los almacenes de datos admiten los siguientes marcos de autenticación y autorización para SMART en las solicitudes del FHIR:

- OpenID (AuthN): para autenticar que la persona o la aplicación cliente es quien (o qué) dice ser.
- OAuth 2.0 (AuthZ): para autorizar qué recursos del FHIR de su almacén de HealthLake datos puede leer o escribir una solicitud autenticada. Esto se define mediante los ámbitos configurados en su servidor de autorización.

Puede crear un almacén de datos compatible con SMART on FHIR mediante la AWS CLI tecla o. AWS SDKs Para obtener más información, consulte Creación de un almacén HealthLake de datos.

Temas

- Cómo empezar a usar SMART en FHIR
- HealthLake requisitos de autenticación para SMART en FHIR
- Los osciloscopios SMART on FHIR OAuth 2.0 son compatibles con HealthLake
- Validación de token mediante AWS Lambda
- Uso de una autorización detallada con un almacén de datos habilitado para SMART on FHIR HealthLake
- Obtención del documento de descubrimiento SMART on FHIR
- Realizar una solicitud a la API REST de FHIR en un almacén de datos habilitado para SMART HealthLake

Cómo empezar a usar SMART en FHIR

En los siguientes temas se describe cómo empezar a utilizar SMART con la autorización del FHIR para. AWS HealthLake Incluyen los recursos que debe aprovisionar en su AWS cuenta, la creación de un almacén de HealthLake datos habilitado para SMART on FHIR y un ejemplo de cómo una aplicación cliente SMART on FHIR interactúa con un servidor de autorización y un almacén de datos. HealthLake

Temas

- Configuración de los recursos para SMART en el FHIR
- Flujo de trabajo de aplicaciones cliente para SMART on FHIR

Configuración de los recursos para SMART en el FHIR

Los siguientes pasos definen cómo gestionan las solicitudes SMART on FHIR HealthLake y los recursos necesarios para que tengan éxito. Los siguientes elementos se combinan en un flujo de trabajo para crear una solicitud SMART ante la FHIR:

- El usuario final: por lo general, un paciente o un médico que utiliza una aplicación SMART on FHIR de terceros para acceder a los datos de un almacén de datos. HealthLake
- La aplicación SMART on FHIR (denominada aplicación cliente): una aplicación que desea acceder a los datos que se encuentran en el almacén de datos. HealthLake
- El servidor de autorización: un servidor compatible con OpenID Connect que puede autenticar a los usuarios y emitir tokens de acceso.
- El almacén de HealthLake datos: un almacén de HealthLake datos habilitado para SMART on FHIR que utiliza una función Lambda para responder a las solicitudes REST del FHIR que proporcionan un token portador.

Para que estos elementos funcionen juntos, debe crear los siguientes recursos.

> **Note**
>
> Le recomendamos que cree su almacén de HealthLake datos compatible con SMART on FHIR una vez que haya configurado el servidor de autorización, definido los alcances necesarios y creado una AWS Lambda función para gestionar la introspección simbólica.

1. Configure un punto final del servidor de autorización

Para utilizar el marco SMART on FHIR, debe configurar un servidor de autorización externo que pueda validar las solicitudes REST del FHIR realizadas en un almacén de datos. Para obtener más información, consulte HealthLake requisitos de autenticación para SMART en FHIR.

2. Defina los ámbitos de su servidor de autorización para controlar los niveles de acceso a HealthLake los almacenes de datos

El marco SMART on FHIR utiliza los OAuth alcances para determinar a qué recursos del FHIR tiene acceso una solicitud autenticada y en qué medida. Definir los ámbitos es una forma de diseñar pensando en los privilegios mínimos. Para obtener más información, consulte Los osciloscopios SMART on FHIR OAuth 2.0 son compatibles con HealthLake.

3. Configure una AWS Lambda función capaz de realizar una introspección simbólica

Una solicitud REST del FHIR enviada por la aplicación cliente a un almacén de datos habilitado para SMART on FHIR contiene un token web JSON (JWT). Para obtener más información, consulte Decodificar un JWT.

4. Cree un almacén de datos compatible HealthLake con SMART en FHIR

Para crear un almacén de HealthLake datos SMART en el FHIR, debe proporcionar un. `IdentityProviderConfiguration` Para obtener más información, consulte Creación de un almacén HealthLake de datos.

Flujo de trabajo de aplicaciones cliente para SMART on FHIR

En la siguiente sección se explica cómo lanzar una aplicación cliente y realizar una solicitud REST del FHIR correcta en un almacén de HealthLake datos en el contexto de SMART on FHIR.

1. Realice una **`GET`** solicitud a Known Uniform Resource Identifier mediante una aplicación cliente

Una aplicación cliente compatible con SMART debe realizar una `GET` solicitud para encontrar los puntos finales de autorización de su almacén de HealthLake datos. Esto se hace mediante una solicitud de identificador uniforme de recursos (URI) bien conocido. Para obtener más información, consulte Obtención del documento de descubrimiento SMART on FHIR.

2. Solicita el acceso y los alcances

La aplicación cliente utiliza el punto final de autorización del servidor de autorización para que el usuario pueda iniciar sesión. Este proceso autentica al usuario. Los ámbitos se utilizan para definir a qué recursos del FHIR del almacén de HealthLake datos puede acceder una aplicación cliente. Para obtener más información, consulte Los osciloscopios SMART on FHIR OAuth 2.0 son compatibles con HealthLake.

3. Tokens de acceso

Ahora que el usuario se ha autenticado, una aplicación cliente recibe un token de acceso JWT del servidor de autorización. Este token se proporciona cuando la aplicación cliente envía una solicitud REST del FHIR a. HealthLake Para obtener más información, consulte Validación de tókenes.

4. Realice una solicitud a la API REST del FHIR en SMART, en un almacén de datos compatible con el FHIR HealthLake

La aplicación cliente ahora puede enviar una solicitud de API REST del FHIR a un punto final del almacén de HealthLake datos mediante el token de acceso proporcionado por el servidor de autorización. Para obtener más información, consulte Realizar una solicitud a la API REST de FHIR en un almacén de datos habilitado para SMART HealthLake.

5. Valide el token de acceso JWT

Para validar el token de acceso enviado en la solicitud REST del FHIR, utilice una función Lambda. Para obtener más información, consulte Validación de token mediante AWS Lambda.

HealthLake requisitos de autenticación para SMART en FHIR

Para acceder a los recursos del FHIR en un almacén de HealthLake datos compatible con SMART o FHIR, una aplicación cliente debe estar autorizada por un servidor de autorización OAuth compatible con la versión 2.0 y presentar un token de OAuth portador como parte de una solicitud de la API REST del FHIR. Para encontrar el punto final del servidor de autorización, utilice el documento de descubrimiento HealthLake SMART on FHIR mediante un identificador de recursos uniforme. `Well-Known` Para obtener más información sobre este proceso, consulte Obtención del documento de descubrimiento SMART on FHIR.

Al crear un banco de HealthLake datos SMART en el FHIR, debe definir el punto final del servidor de autorización y el punto final del token en el `metadata` elemento de la `CreateFHIRDatastore` solicitud. Para obtener más información sobre la definición del `metadata` elemento, consulteCreación de un almacén HealthLake de datos.

Mediante los puntos finales del servidor de autorización, la aplicación cliente autenticará a un usuario con el servicio de autorización. Una vez autorizado y autenticado, el servicio de autorización genera un token web JSON (JWT) y lo pasa a la aplicación cliente. Este token contiene los ámbitos de recursos del FHIR que la aplicación cliente puede utilizar, lo que a su vez restringe los datos a los que puede acceder el usuario. Opcionalmente, si se proporcionó el alcance del lanzamiento, la respuesta contendrá esos detalles. Para obtener más información sobre los osciloscopios SMART on FHIR compatibles HealthLake, consulte. Los osciloscopios SMART on FHIR OAuth 2.0 son compatibles con HealthLake

Con el JWT otorgado por el servidor de autorización, una aplicación cliente realiza llamadas a la API REST del FHIR a un almacén de datos SMART que esté habilitado para el FHIR. HealthLake Para validar y decodificar el JWT, debe crear una función Lambda. HealthLake invoca esta función Lambda en su nombre cuando se recibe una solicitud de la API REST del FHIR. Para ver un ejemplo de función Lambda de inicio, consulte. Validación de token mediante AWS Lambda

Elementos del servidor de autorización necesarios para crear un almacén de datos compatible HealthLake con SMART on FHIR

En la `CreateFHIRDatastore` solicitud, debes proporcionar el punto final de autorización y el punto final del token como parte del `metadata` elemento del `IdentityProviderConfiguration` objeto. Se requieren tanto el punto final de autorización como el punto final del token. Para ver un ejemplo de cómo se especifica esto en `CreateFHIRDatastore` una solicitud, consulteCreación de un almacén HealthLake de datos.

Reclamaciones obligatorias para completar una solicitud de API REST del FHIR en un almacén de datos habilitado para SMART o FHIR HealthLake

Su AWS Lambda función debe contener las siguientes afirmaciones para que sea una solicitud de API REST del FHIR válida en un almacén de datos compatible con SMART on FHIR. HealthLake

- `nbf`: Reclamación (no anterior): la afirmación «nbf» (no anterior) identifica el momento antes del cual NO SE DEBE aceptar el procesamiento del JWT. La tramitación de la reclamación «nbf» requiere que la actual date/time DEBE ser igual o igual a la cifra no date/time indicada anteriormente en la reclamación «nbf». La función Lambda de ejemplo que proporcionamos convierte la respuesta `iat` del servidor en. `nbf`
- `exp`: (fecha de caducidad): la afirmación «exp» (fecha de caducidad) identifica la fecha de caducidad a partir de la cual no se debe aceptar el JWT para su procesamiento.
- `isAuthorized`: Un booleano establecido en. `True` Indica que la solicitud se ha autorizado en el servidor de autorización.
- `aud`Afirmación: (audiencia): la afirmación «aud» (audiencia) identifica a los destinatarios a los que está destinado el JWT. Debe ser un terminal de almacenamiento de HealthLake datos compatible con SMART on FHIR.
- `scope`: Debe ser al menos un ámbito relacionado con los recursos del FHIR. Este ámbito se define en su servidor de autorización. Para obtener más información sobre los ámbitos relacionados con los recursos del FHIR aceptados por HealthLake, consulte. Los alcances de los recursos de SMART on FHIR son: HealthLake

Los osciloscopios SMART on FHIR OAuth 2.0 son compatibles con HealthLake

HealthLake usa OAuth 2.0 como protocolo de autorización. El uso de este protocolo en su servidor de autorización le permite definir HealthLake los permisos del almacén de datos (crear, leer, actualizar, eliminar y buscar) para los recursos del FHIR a los que tiene acceso una aplicación cliente.

El marco SMART on FHIR define un conjunto de ámbitos que se pueden solicitar al servidor de autorizaciones. Por ejemplo, una aplicación cliente que solo esté diseñada para permitir a los pacientes ver sus resultados de laboratorio o ver sus datos de contacto solo debería estar autorizada a solicitar `read` osciloscopios.

> **Note**
>
> HealthLake proporciona compatibilidad tanto con SMART en el FHIR V1 como en el V2, tal y como se describe a continuación. El SMART del FHIR `AuthorizationStrategy`se establece en uno de los tres valores siguientes cuando se crea el banco de datos:
>
> - `SMART_ON_FHIR_V1`— Support solo para SMART en el FHIR V1, que incluye permisos `read` (lectura/búsqueda) y `write` (create/update/delete).
> - `SMART_ON_FHIR`— Soporte para SMART en FHIR V1 y V2, que incluye `createread`, `updatedelete`, y `search` permisos.
> - `AWS_AUTH`— La estrategia de AWS HealthLake autorización predeterminada; no está asociada a SMART en el FHIR.

Alcance de lanzamiento independiente

HealthLake admite el ámbito del modo de lanzamiento independiente. `launch/patient`

En el modo de inicio independiente, una aplicación cliente solicita acceso a los datos clínicos del paciente porque la aplicación cliente no conoce al usuario ni al paciente. Por lo tanto, la solicitud de autorización de la aplicación cliente solicita explícitamente que se devuelva la sonda del paciente. Tras una autenticación correcta, el servidor de autorización emite un token de acceso que contiene el osciloscopio de pacientes de lanzamiento solicitado. El contexto del paciente necesario se proporciona junto con el token de acceso en la respuesta del servidor de autorización.

Alcances del modo de lanzamiento compatibles

Alcance	Description (Descripción)
`launch/patient`	Parámetro de una solicitud de autorización OAuth 2.0 que solicita que se devuelvan los datos del paciente en la respuesta de autorización.

Los alcances de los recursos de SMART on FHIR son: HealthLake

HealthLake define tres niveles de SMART en los ámbitos de los recursos del FHIR.

- `patient`los alcances permiten acceder a datos específicos sobre un solo paciente.
- `user`los ámbitos otorgan acceso a datos específicos a los que puede acceder un usuario.
- `system`los alcances permiten el acceso a todos los recursos del FHIR que se encuentran en el HealthLake almacén de datos.

En las siguientes secciones se muestra la sintaxis para construir los alcances de los recursos del FHIR mediante SMART en el FHIR V1 o SMART en el FHIR V2.

> **Note**
>
> La estrategia de autorización SMART on FHIR se establece cuando se crea el banco de datos. Para obtener más información, consulta `AuthorizationStrategy` en la AWS HealthLake Referencia de la API de .

Los osciloscopios SMART on FHIR V1 son compatibles con HealthLake

Cuando se utiliza SMART en el FHIR V1, la sintaxis general para construir los ámbitos de los recursos del FHIR es la siguiente. HealthLake Para ver la ruta URL completa en el siguiente ejemplo, desplácese sobre el botón Copiar.

```
('patient' | 'user' | 'system') '/' (fhir-resource | '*') '.' ('read' | 'write' | '*')
```

Los ámbitos de autorización compatibles con SMART en la versión FHIR v1

Sintaxis del ámbito	Ejemplo de ámbito	Resultado
`patient/(fhir-resource | '*').('read' | 'write' | '*')`	`patient/AllergyIntolerance.*`	La aplicación cliente para pacientes tiene acceso de lectura/escritura a nivel de instancia a todas las alergias registradas.
`user/(fhir-resource | '*').('read' | 'write' | '*')`	`user/Observation.read`	La aplicación cliente de usuario tiene acceso a nivel de instancia read/write a todas las observaciones registradas.
`system/('read' | 'write' | *)`	`system/*.*`	La aplicación cliente del sistema tiene read/write acceso a todos los datos de los recursos del FHIR.

SMART en los osciloscopios FHIR V2 compatibles con HealthLake

Cuando se utiliza SMART en el FHIR V2, la sintaxis general para construir los alcances de los recursos del FHIR es la siguiente. HealthLake Para ver la ruta URL completa en el siguiente ejemplo, desplácese sobre el botón Copiar.

```
('patient' | 'user' | 'system') '/' (fhir-resource | '*') '.' ('c' | 'r' | 'u' | 'd' | 's')
```

> **Note**
>
> Para usar SMART en el FHIR V2, debe pasar el valor `permission-v2` a la `capabilities` cadena de metadatos, que forma parte del tipo de `IdentityProviderConfiguration` datos.

HealthLake admite ámbitos granulares. Para obtener más información, consulte los ámbitos granulares compatibles en la Guía de implementación básica del FHIR para EE. UU.

Los ámbitos de autorización compatibles con SMART on FHIR V2

Sintaxis del ámbito	Ejemplo de ámbito V1	Resultado
`patient/Observation.rs`	`user/Observation.read`	Permiso para leer y buscar el `Observation` recurso del paciente actual.
`system/*.cruds`	`system/*.*`	La aplicación cliente del sistema tiene create/read/update/delete/search acceso total a todos los datos de los recursos del FHIR.

Validación de token mediante AWS Lambda

Al crear un almacén de datos habilitado para HealthLake SMART en el FHIR, debe proporcionar el ARN de AWS Lambda la función en `CreateFHIRDatastore` la solicitud. El ARN de la función Lambda se especifica en el `IdentityProviderConfiguration` objeto mediante el parámetro. `IdpLambdaArn`

Debe crear la función Lambda antes de crear su banco de datos habilitado para SMART on FHIR. Una vez creado el banco de datos, no se puede cambiar el ARN de Lambda. Para ver el ARN de Lambda que especificó al crear el banco de datos, utilice la acción de la API. `DescribeFHIRDatastore`

Para que una solicitud REST de FHIR tenga éxito en un almacén de datos habilitado para SMART on FHIR, la función Lambda debe hacer lo siguiente:

- Devuelve una respuesta en menos de 1 segundo al punto final del almacén de HealthLake datos.

- Decodifique el token de acceso proporcionado en el encabezado de autorización de la solicitud de API REST enviada por la aplicación cliente.
- Asigne una función de servicio de IAM que tenga permisos suficientes para llevar a cabo la solicitud de la API REST del FHIR.
- Para completar una solicitud de la API REST del FHIR, se requieren las siguientes afirmaciones. Para obtener más información, consulte Reclamaciones requeridas.
  - `nbf`
  - `exp`
  - `isAuthorized`
  - `aud`
  - `scope`

Al trabajar con Lambda, además de la función Lambda, debe crear un rol de ejecución y una política basada en recursos. La función de ejecución de una función de Lambda es una función de IAM que otorga a la función permiso para acceder a los servicios y recursos de AWS necesarios en tiempo de ejecución. La política basada en recursos que proporcione debe permitir HealthLake invocar su función en su nombre.

En las secciones de este tema se describe un ejemplo de solicitud de una aplicación cliente y una respuesta decodificada, los pasos necesarios para crear una función AWS Lambda y cómo crear una política basada en recursos que se pueda asumir. HealthLake

- Parte 1: Creación de una función Lambda
- Parte 2: Crear un rol HealthLake de servicio utilizado por la función AWS Lambda
- Parte 3: Actualización del rol de ejecución de la función Lambda
- Parte 4: Añadir una política de recursos a la función Lambda
- Parte 5: Aprovisionamiento simultáneo para su función Lambda

Creación de una AWS función Lambda

La función Lambda creada en este tema se activa cuando HealthLake recibe una solicitud a un almacén de datos habilitado para SMART en FHIR. La solicitud de la aplicación cliente contiene una llamada a la API REST y un encabezado de autorización que contiene un token de acceso.

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

```
Authorization: Bearer i8hweunweunweofiwweoijewiwe
```

El ejemplo de la función Lambda de este tema se utiliza AWS Secrets Manager para ocultar las credenciales relacionadas con el servidor de autorización. Recomendamos encarecidamente no proporcionar los detalles de inicio de sesión del servidor de autorización directamente en una función Lambda.

Example validar una solicitud REST del FHIR que contenga un token portador de autorización

La función Lambda de ejemplo muestra cómo validar una solicitud REST de FHIR enviada a un almacén de datos SMART en FHIR habilitado. Para ver step-by-steps instrucciones sobre cómo implementar esta función Lambda, consulte. Creación de una función Lambda mediante el Consola de administración de AWS

Si la solicitud de la API REST de FHIR no contiene un punto final de almacén de datos, un token de acceso y una operación REST válidos, la función Lambda fallará. Para obtener más información sobre los elementos del servidor de autorización necesarios, consulte. Reclamaciones requeridas

```
import base64
import boto3
import logging
import json
import os
from urllib import request, parse

logger = logging.getLogger()
logger.setLevel(logging.INFO)

## Uses Secrets manager to gain access to the access key ID and secret access key for
 the authorization server
client = boto3.client('secretsmanager', region_name="region-of-datastore")
response = client.get_secret_value(SecretId='name-specified-by-customer-in-
secretsmanager')
secret = json.loads(response['SecretString'])
client_id = secret['client_id']
client_secret = secret['client_secret']


unencoded_auth = f'{client_id}:{client_secret}'
headers = {
  'Authorization': f'Basic {base64.b64encode(unencoded_auth.encode()).decode()}',
  'Content-Type': 'application/x-www-form-urlencoded'
```

```
}

auth_endpoint = os.environ['auth-server-base-url'] # Base URL of the Authorization
 server
user_role_arn = os.environ['iam-role-arn'] # The IAM role client application will use
 to complete the HTTP request on the datastore

def lambda_handler(event, context):
    if 'datastoreEndpoint' not in event or 'operationName' not in event or
 'bearerToken' not in event:
    return {}

    datastore_endpoint = event['datastoreEndpoint']
    operation_name = event['operationName']
    bearer_token = event['bearerToken']
    logger.info('Datastore Endpoint [{}], Operation Name:
 [{}]'.format(datastore_endpoint, operation_name))

    ## To validate the token
    auth_response = auth_with_provider(bearer_token)
    logger.info('Auth response: [{}]'.format(auth_response))
    auth_payload = json.loads(auth_response)
    ## Required parameters needed to be sent to the datastore endpoint for the HTTP
 request to go through
    auth_payload["isAuthorized"] = bool(auth_payload["active"])
    auth_payload["nbf"] = auth_payload["iat"]
    return {"authPayload": auth_payload, "iamRoleARN": user_role_arn}

## access the server
def auth_with_provider(token):
    data = {'token': token, 'token_type_hint': 'access_token'}
    req = request.Request(url=auth_endpoint + '/v1/introspect',
 data=parse.urlencode(data).encode(), headers=headers)
    with request.urlopen(req) as resp:
    return resp.read().decode()
```

Creación de una función Lambda mediante el Consola de administración de AWS

El siguiente procedimiento supone que ya ha creado la función de servicio que desea asumir HealthLake al gestionar una solicitud de la API REST del FHIR en un almacén de datos habilitado para SMART on FHIR. Si no ha creado el rol de servicio, aún puede crear la función Lambda. Debe añadir el ARN del rol de servicio para que la función Lambda funcione. Para obtener más información

sobre cómo crear un rol de servicio y especificarlo en la función Lambda, consulte Crear un rol HealthLake de servicio para usarlo en la función AWS Lambda utilizada para decodificar un JWT

Para crear una función Lambda ()Consola de administración de AWS

1. Abra la página de Functions (Funciones) en la consola de Lambda.
2. Seleccione Creación de función.
3. Seleccione Crear desde cero.
4. En Información básica, introduzca un nombre de función. En Tiempo de ejecución, elija un tiempo de ejecución basado en Python.
5. En Execution role (Rol de ejecución), elija Create a new role with basic Lambda permissions (Crear un nuevo rol con permisos básicos de Lambda).

   Lambda crea un rol de ejecución que otorga a la función permiso para cargar registros en Amazon. CloudWatch La función Lambda asume la función de ejecución cuando se invoca la función y la utiliza para crear credenciales para el SDK. AWS
6. Seleccione la pestaña Código y añada la función Lambda de ejemplo.

   Si aún no ha creado el rol de servicio que va a utilizar la función Lambda, tendrá que crearlo antes de que funcione la función Lambda de ejemplo. Para obtener más información sobre la creación de un rol de servicio para la función Lambda, consulte. Crear un rol HealthLake de servicio para usarlo en la función AWS Lambda utilizada para decodificar un JWT

```
import base64
import boto3
import logging
import json
import os
from urllib import request, parse

logger = logging.getLogger()
logger.setLevel(logging.INFO)

## Uses Secrets manager to gain access to the access key ID and secret access key
 for the authorization server
client = boto3.client('secretsmanager', region_name="region-of-datastore")
response = client.get_secret_value(SecretId='name-specified-by-customer-in-
secretsmanager')
secret = json.loads(response['SecretString'])
client_id = secret['client_id']
```

```
client_secret = secret['client_secret']


unencoded_auth = f'{client_id}:{client_secret}'
headers = {
  'Authorization': f'Basic {base64.b64encode(unencoded_auth.encode()).decode()}',
  'Content-Type': 'application/x-www-form-urlencoded'
}

auth_endpoint = os.environ['auth-server-base-url'] # Base URL of the Authorization
 server
user_role_arn = os.environ['iam-role-arn'] # The IAM role client application will
 use to complete the HTTP request on the datastore

def lambda_handler(event, context):
    if 'datastoreEndpoint' not in event or 'operationName' not in event or
 'bearerToken' not in event:
    return {}

    datastore_endpoint = event['datastoreEndpoint']
    operation_name = event['operationName']
    bearer_token = event['bearerToken']
    logger.info('Datastore Endpoint [{}], Operation Name:
 [{}]'.format(datastore_endpoint, operation_name))

    ## To validate the token
    auth_response = auth_with_provider(bearer_token)
    logger.info('Auth response: [{}]'.format(auth_response))
    auth_payload = json.loads(auth_response)
    ## Required parameters needed to be sent to the datastore endpoint for the HTTP
 request to go through
    auth_payload["isAuthorized"] = bool(auth_payload["active"])
    auth_payload["nbf"] = auth_payload["iat"]
    return {"authPayload": auth_payload, "iamRoleARN": user_role_arn}

## Access the server
def auth_with_provider(token):
    data = {'token': token, 'token_type_hint': 'access_token'}
    req = request.Request(url=auth_endpoint + '/v1/introspect',
 data=parse.urlencode(data).encode(), headers=headers)
    with request.urlopen(req) as resp:
    return resp.read().decode()
```

Modificación del rol de ejecución de una función Lambda

Tras crear la función Lambda, debe actualizar el rol de ejecución para incluir los permisos necesarios para llamar a Secrets Manager. En Secrets Manager, cada secreto que crees tiene un ARN. Para aplicar el mínimo de privilegios, la función de ejecución solo debe tener acceso a los recursos necesarios para que se ejecute la función Lambda.

Puede modificar la función de ejecución de una función de Lambda buscándola en la consola de IAM o seleccionando Configuración en la consola de Lambda. Para obtener más información sobre la administración de su función de ejecución de funciones de Lambda, consulte. Rol de ejecución de Lambda

Example Función de ejecución de la función Lambda que otorga acceso a **`GetSecretValue`**

Al añadir la acción de IAM `GetSecretValue` a la función de ejecución, se otorga el permiso necesario para que funcione la función Lambda de muestra.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:secret-name-DKodTA"
        }
    ]
}
```

En este punto, ha creado una función Lambda que se puede utilizar para validar el token de acceso proporcionado como parte de la solicitud REST de FHIR enviada a su almacén de datos SMART on FHIR habilitado.

Crear un rol HealthLake de servicio para usarlo en la función AWS Lambda utilizada para decodificar un JWT

Perfil: administrador de IAM

Un usuario que puede añadir o eliminar políticas de IAM y crear nuevas identidades de IAM.

Rol de servicio

Un rol de servicio es un rol de IAM que asume un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio desde IAM. Para obtener más información, consulte Crear un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM.

Una vez decodificado el token web JSON (JWT), Lambda también necesita devolver un ARN de rol de IAM. Esta función debe tener los permisos necesarios para llevar a cabo la solicitud de la API REST o fallará por falta de permisos.

Al configurar una política personalizada mediante IAM, lo mejor es conceder los permisos mínimos necesarios. Para obtener más información, consulte Aplicar permisos con privilegios mínimos en la Guía del usuario de IAM.

La creación HealthLake de un rol de servicio para designarlo en la función Lambda de autorización requiere dos pasos.

- En primer lugar, debe crear una política de IAM. La política debe especificar el acceso a los recursos del FHIR para los que ha proporcionado ámbitos en el servidor de autorización.
- En segundo lugar, debe crear el rol de servicio. Al crear el rol, designa una relación de confianza y adjunta la política que creó en el primer paso. La relación de confianza se designa HealthLake como principal del servicio. Debe especificar un ARN de almacén de HealthLake datos y un ID de AWS cuenta en este paso.

Crear una nueva política de IAM

Los ámbitos que defina en su servidor de autorización determinan a qué recursos del FHIR tiene acceso un usuario autenticado en un almacén de datos. HealthLake

La política de IAM que cree se puede personalizar para que coincida con los ámbitos que haya definido.

Se pueden definir las siguientes acciones en el `Action` elemento de una declaración de política de IAM. Para cada uno de `Action` los elementos de la tabla, puede definir un`Resource types`. En HealthLake un banco de datos, es el único tipo de recurso admitido que se puede definir en el `Resource` elemento de una declaración de política de permisos de IAM.

Los recursos individuales del FHIR no son un recurso que se pueda definir como un elemento de una política de permisos de la IAM.

Acciones definidas por HealthLake

Acciones	Descripción	Nivel de acces	Tipo de recurso (obligatorio)
CreateResource	Otorga permiso para crear un recurso	Escrit	ARN del almacén de datos: arn:aws:healthlake: :datastore/fhir/ **your-region 111122223333 your-datastore-id**
DeleteResource	Otorga permiso para eliminar un recurso	Escrit	ARN del almacén de datos: arn:aws:healthlake: :datastore/fhir/ **your-region 111122223333 your-datastore-id**
ReadResource	Otorga permiso para leer el recurso	Lectu	ARN del almacén de datos: arn:aws:healthlake: :datastore/fhir/ **your-region 111122223333 your-datastore-id**
SearchWithGet	Otorga permiso para buscar recursos con el método GET	Lectu	ARN del almacén de datos: arn:aws:healthlake: :datastore/fhir/ **your-region 111122223333 your-datastore-id**
SearchWithPost	Otorga permiso para buscar recursos con el método POST	Lectu	ARN del almacén de datos: arn:aws:healthlake: :datastore/fhir/ **your-region 111122223333 your-datastore-id**
InicioFHIRExport	Otorga permiso para iniciar un trabajo de	Escrit	ARN del almacén de datos: arn:aws:healthlake: :datastore/fhir/ **your-region 111122223333 your-datastore-id**

Acciones	Descripción	Nivel de acces	Tipo de recurso (obligatorio)
JobWithPost	exportación del FHIR con GET		
UpdateResource	Otorga permiso para actualizar el recurso	Escrit	ARN del almacén de datos: arn:aws:healthlake: :datastore/fhir/ **`your-region 111122223333 your-datastore-id`**

Para empezar, puede utilizar. `AmazonHealthLakeFullAccess` Esta política permitiría leer, escribir, buscar y exportar todos los recursos del FHIR que se encuentren en un almacén de datos. Para conceder permisos de solo lectura en un almacén de datos, utilice. `AmazonHealthLakeReadOnlyAccess`

Para obtener más información sobre cómo crear una política personalizada mediante el Consola de administración de AWS, o el IAM AWS CLI SDKs, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Crear un rol de servicio para HealthLake (consola de IAM)

Utilice este procedimiento para crear un rol de servicio. Al crear un servicio, también tendrá que designar una política de IAM.

Para crear el rol de servicio para HealthLake (consola de IAM)

1. Inicie sesión en la consola de IAM Consola de administración de AWS y ábrala en. https://console.aws.amazon.com/iam/
2. En el panel de navegación de la consola de IAM, elija Roles.
3. A continuación, elija Create role (Crear rol).
4. En la página Seleccionar entidad de confianza, elija Política de confianza personalizada.
5. A continuación, en Política de confianza personalizada, actualice la política de ejemplo de la siguiente manera. **`your-account-id`**Sustitúyalo por su número de cuenta y añada el ARN del banco de datos que desee utilizar en sus trabajos de importación o exportación.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Principal": {
                "Service": "healthlake.amazonaws.com"
            },
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:healthlake:us-
east-1:123456789012:datastore/fhir/your-datastore-id"
                }
            }
        }
    ]
}
```

6. A continuación, elija Siguiente.
7. En la página Agregar permisos, elija la política que desee que asuma el HealthLake servicio. Para encontrar tu política, búscala en Políticas de permisos.
8. A continuación, selecciona Adjuntar política.
9. A continuación, en la página Nombre, revisión y creación, situada en Nombre del rol, introduzca un nombre.
10. (Opcional) A continuación, en Descripción, añade una breve descripción de tu función.
11. De ser posible, escriba un nombre o sufijo de nombre para el rol, que pueda ayudarle a identificar su finalidad. Los nombres de rol deben ser únicos en su Cuenta de AWS. No se distingue por caso. Por ejemplo, no puede crear funciones denominado tanto **PRODROLE** como **prodrole**. Dado que varias entidades pueden hacer referencia al rol, no puede editar el nombre del rol después de crearlo.
12. Revisa los detalles del rol y, a continuación, selecciona Crear rol.

Para obtener información sobre cómo especificar el ARN del rol en la función Lambda de ejemplo, consulte. Creación de una AWS función Lambda

Rol de ejecución de Lambda

La función de ejecución de una función Lambda es una función de IAM que concede a la función permiso para acceder a los AWS servicios y recursos. Esta página proporciona información sobre cómo crear, ver y administrar el rol de ejecución de una función de Lambda.

De forma predeterminada, Lambda crea un rol de ejecución con permisos mínimos al crear una nueva función de Lambda mediante. Consola de administración de AWS Para gestionar los permisos concedidos en la función de ejecución, consulte Creación de una función de ejecución en la consola de IAM en la Guía para desarrolladores de Lambda.

La función Lambda de ejemplo que se proporciona en este tema usa Secrets Manager para ocultar las credenciales del servidor de autorización.

Al igual que con cualquier función de IAM que cree, es importante seguir las mejores prácticas con privilegios mínimos. Durante la fase de desarrollo, a veces es posible que concedas permisos más allá de lo necesario. Antes de publicar la función en el entorno de producción, la práctica recomendada consiste en ajustar la política para incluir solo los permisos necesarios. Para obtener más información, consulte Aplicar los privilegios mínimos en la Guía del usuario de IAM.

Permita HealthLake activar su función Lambda

Para HealthLake poder invocar la función Lambda en su nombre, debe hacer lo siguiente:

- Debe establecer un `IdpLambdaArn` valor igual al ARN de la función Lambda que desee HealthLake invocar en la solicitud. `CreateFHIRDatastore`
- Necesita una política basada en recursos que le permita HealthLake invocar la función Lambda en su nombre.

Cuando HealthLake recibe una solicitud de la API REST de FHIR en un almacén de datos habilitado para SMART o FHIR, necesita permisos para invocar en su nombre la función Lambda especificada en la creación del almacén de datos. Para conceder el HealthLake acceso, utilizará una política basada en los recursos. Para obtener más información sobre cómo crear una política basada en recursos para una función de Lambda, consulte Permitir que un AWS servicio llame a una función de Lambda en la Guía para desarrolladores.AWS Lambda

Aprovisionamiento simultáneo para su función Lambda

> **Important**
>
> HealthLake requiere que el tiempo máximo de ejecución de la función Lambda sea inferior a un segundo (1000 milisegundos).
> Si la función Lambda supera el límite de tiempo de ejecución, se produce una TimeOutexcepción.

Para evitar esta excepción, se recomienda configurar la simultaneidad aprovisionada. Mediante la asignación de la simultaneidad aprovisionada antes de un aumento en las invocaciones, puede asegurarse de que todas las solicitudes se atiendan mediante instancias inicializadas con una latencia baja. Para obtener más información sobre la configuración de la simultaneidad aprovisionada, consulte Configuración de la simultaneidad aprovisionada en la Guía para desarrolladores de Lambda

Para ver el tiempo medio de ejecución de la función Lambda en la actualidad, utilice la página de supervisión de la función Lambda en la consola de Lambda. De forma predeterminada, la consola Lambda proporciona un gráfico de duración que muestra la cantidad media, mínima y máxima de tiempo que el código de función dedica a procesar un evento. Para obtener más información sobre la supervisión de las funciones de Lambda, consulte Supervisión de las funciones en la consola de Lambda en la Guía para desarrolladores de Lambda.

Si ya ha aprovisionado la simultaneidad para la función de Lambda y desea supervisarla, consulte Supervisión de la simultaneidad en la Guía para desarrolladores de Lambda.

Uso de una autorización detallada con un almacén de datos habilitado para SMART on FHIR HealthLake

Los ámbitos por sí solos no proporcionan la especificidad necesaria sobre los datos a los que el solicitante está autorizado a acceder en un almacén de datos. El uso de una autorización detallada permite un mayor nivel de especificidad a la hora de conceder acceso a un almacén de datos compatible con SMART en el FHIR. HealthLake Para utilizar una autorización detallada, establezca un `FineGrainedAuthorizationEnabled` valor igual a `True` en el parámetro de su solicitud. `IdentityProviderConfiguration` `CreateFHIRDatastore`

Si has habilitado la autorización detallada, tu servidor de autorización devuelve un `fhirUser` ámbito junto con el `id_token` token de acceso. Esto permite que la aplicación cliente recupere

la información sobre el usuario. La aplicación cliente debe tratar la `fhirUser` reclamación como el URI de un recurso del FHIR que represente al usuario actual. Este valor puede ser `Patient`, `Practitioner` o `RelatedPerson`. La respuesta del servidor de autorización también incluye un `user/` ámbito que define a qué datos puede acceder el usuario. Utiliza la sintaxis definida para los ámbitos relacionados con los ámbitos específicos de los recursos del FHIR:

```
user/(fhir-resource | '*').('read' | 'write' | '*')
```

Los siguientes son ejemplos de cómo se puede utilizar la autorización detallada para especificar con más detalle los tipos de recursos del FHIR relacionados con el acceso a los datos.

- Cuándo `fhirUser` es una autorización detalladaPractitioner, determina el conjunto de pacientes a los que puede acceder el usuario. Solo `fhirUser` se permite el acceso a aquellos pacientes en los que el paciente tenga referencias `fhirUser` como médico generalista.

  ```
  Patient.generalPractitioner : [{Reference(Practitioner)}]
  ```

- Cuando `fhirUser` es un `Patient` o `RelatedPerson` y el paciente al que se hace referencia en la solicitud es diferente del `fhirUser` indicado, la autorización detallada determina el acceso del paciente `fhirUser` solicitado. Se permite el acceso cuando existe una relación especificada en el recurso solicitado. `Patient`

  ```
  Patient.link.other : {Reference(Patient|RelatedPerson)}
  ```

Obtención del documento de descubrimiento SMART on FHIR

SMART define un documento de descubrimiento que permite a los clientes conocer el punto final de autorización URLs y cuenta con los soportes de un almacén de HealthLake datos. Esta información ayuda a los clientes a dirigir las solicitudes de autorización al punto final correcto y a crear las solicitudes de autorización compatibles con el almacén de HealthLake datos.

Para que una aplicación cliente pueda realizar correctamente una solicitud REST del FHIR HealthLake, debe reunir los requisitos de autorización definidos por el banco de HealthLake datos. No se requiere un token de portador (autorización) para que esta solicitud se realice correctamente.

Para solicitar el documento de descubrimiento de un almacén de HealthLake datos

1. Recopile HealthLake `region` y `datastoreId` valore. Para obtener más información, consulte Obtención de propiedades de los almacenes de datos.
2. Cree una URL para la solicitud utilizando los valores recopilados para HealthLake `region` y`datastoreId`. `/.well-known/smart-configuration`Añádala al punto final de la URL. Para ver la ruta URL completa en el siguiente ejemplo, desplázate sobre el botón Copiar.

```
https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/.well-known/smart-configuration
```

3. Envíe la solicitud utilizando `GET` el protocolo de AWS firma Signature versión 4. Para ver el ejemplo completo, desplázate sobre el botón Copiar.

   curl

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

   El documento de descubrimiento del banco de HealthLake datos aparece como un blob JSON, donde puede encontrar el `authorization_endpoint` y el almacén de datos`token_endpoint`, junto con las especificaciones y las capacidades definidas para el banco de datos.

```
{
    "authorization_endpoint": "https://oidc.example.com/authorize",
    "token_endpoint": "https://oidc.example.com/oauth/token",
    "capabilities": [
        "launch-ehr",
        "client-public"
    ]
}
```

   `authorization_endpoint`Tanto el como el `token_endpoint` son necesarios para lanzar una aplicación cliente.

- Punto final de autorización: la URL necesaria para autorizar una aplicación cliente o un usuario.
- Punto final del token: punto final del servidor de autorización con el que la aplicación cliente se comunica.

Realizar una solicitud a la API REST de FHIR en un almacén de datos habilitado para SMART HealthLake

Puede realizar solicitudes a la API REST del FHIR en un almacén de datos habilitado para SMART o FHIR. HealthLake El siguiente ejemplo muestra una solicitud de una aplicación cliente que contiene un JWT en el encabezado de autorización y cómo Lambda debe decodificar la respuesta. Una vez autorizada y autenticada la solicitud de la aplicación cliente, debe recibir un token portador del servidor de autorización. Utilice el token portador del encabezado de autorización cuando envíe una solicitud de API REST del FHIR a un almacén de datos habilitado para SMART o FHIR. HealthLake

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/[ID]
Authorization: Bearer auth-server-provided-bearer-token
```

Como se encontró un token portador en el encabezado de autorización y no se detectó ninguna identidad de AWS IAM, se HealthLake invoca la función Lambda especificada cuando se creó el almacén de datos habilitado para SMART on FHIR. HealthLake Cuando la función Lambda decodifica correctamente el token, se envía el siguiente ejemplo de respuesta a. HealthLake

```
{
  "authPayload": {
    "iss": "https://authorization-server-endpoint/oauth2/token", # The issuer
 identifier of the authorization server
    "aud": "https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/", #
 Required, data store endpoint
    "iat": 1677115637,  # Identifies the time at which the token was issued
    "nbf": 1677115637,  # Required, the earliest time the JWT would be valid
    "exp": 1997877061,  # Required, the time at which the JWT is no longer valid
    "isAuthorized": "true",  # Required, boolean indicating the request has been
 authorized
    "uid": "100101",  # Unique identifier returned by the auth server
    "scope": "system/*.*" # Required, the scope of the request
  },
  "iamRoleARN": "iam-role-arn" #Required, IAM role to complete the request
```

```
}
```

Soporte FHIR R4 para AWS HealthLake

AWS HealthLake es compatible con la especificación R4 del FHIR para el intercambio de datos de salud. En las siguientes secciones se proporciona información complementaria sobre cómo se HealthLake utiliza la especificación R4 del FHIR para ayudarle a gestionar y buscar los recursos del FHIR en su HealthLake almacén de datos mediante el FHIR R4. RESTful APIs

Temas

- Declaración de capacidad del FHIR R4 para AWS HealthLake
- Validaciones de perfil FHIR para HealthLake
- Tipos de recursos compatibles con FHIR R4 para HealthLake
- Parámetros de búsqueda FHIR R4 para HealthLake
- FHIR R4 para $operations HealthLake

Declaración de capacidad del FHIR R4 para AWS HealthLake

Para encontrar las capacidades (comportamientos) relacionadas con el FHIR de un banco de HealthLake datos activo, debe recuperar su declaración de capacidad. La declaración de capacidad se utiliza como una declaración de la funcionalidad real del servidor o una declaración de la implementación del servidor requerida o deseada. La `capabilities`interacción del FHIR recupera información sobre las capacidades del almacén de HealthLake datos y las partes de la especificación del FHIR compatibles. HealthLake valida los tipos de recursos del FHIR según el recurso R4 del FHIR. `StructureDefinition`

Para obtener la declaración de capacidad de un banco de datos HealthLake

1. Recopile HealthLake `region` y `datastoreId` valore. Para obtener más información, consulte Obtención de propiedades de los almacenes de datos.
2. Cree una URL para la solicitud utilizando los valores recopilados para HealthLake `region` y`datastoreId`. Incluye también el `metadata` elemento FHIR en la URL. Para ver la ruta URL completa en el siguiente ejemplo, desplázate sobre el botón Copiar.

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

3. Envíe la solicitud . La `capabilities` interacción FHIR utiliza una `GET` solicitud con el protocolo de AWS firma Signature, versión 4. En el siguiente `curl` ejemplo, se obtiene la declaración de capacidad del almacén de HealthLake datos especificado por. `datastoreId` Para ver el ejemplo completo, desplácese sobre el botón Copiar.

   curl

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

   Recibirá un código de respuesta `200` HTTP y la declaración de capacidad de su almacén HealthLake de datos. Para obtener más información, consulte `CapabilityStatement`la documentación del FHIR R4.

Validaciones de perfil FHIR para HealthLake

AWS HealthLake es compatible con la especificación básica del FHIR R4. En la especificación básica del FHIR R4 se incluyen los perfiles FHIR. Los perfiles se utilizan en un tipo de recurso FHIR para definir una definición de tipo de recurso más específica mediante and/or extensiones de restricciones del tipo de recurso base. Por ejemplo, un perfil del FHIR puede identificar campos obligatorios, como extensiones y conjuntos de valores. Un recurso puede admitir varios perfiles. Todos los almacenes HealthLake de datos admiten el uso de perfiles FHIR.

> **Note**
>
> No es necesario añadir un perfil FHIR al añadir datos a un banco de HealthLake datos. Si no se especifica un perfil del FHIR al añadir o actualizar un recurso, el recurso se valida únicamente con el esquema R4 del FHIR base.
> Los perfiles del FHIR, a los que se ajustan los recursos del FHIR, se incluyen en los recursos antes de importarlos. HealthLake Por lo tanto, los perfiles del FHIR se validan durante la importación HealthLake .

Los perfiles del FHIR se especifican en una guía de implementación. Una guía de implementación (IG) del FHIR es un conjunto de instrucciones que describen cómo utilizar el estándar del FHIR para un propósito específico. HealthLake valida los perfiles del FHIR definidos en las siguientes guías de implementación.

Los perfiles del FHIR están respaldados por AWS HealthLake

Name	Ve	Guía de implement ación	Funcionalidad	Est de EE (Ol	Est de EE (No de Vir	Oe de EE (Or	Asi Pa fico (M	Asi Pa fico (Sí	Ca (Ce	Eu (Irl	Europa (Londres)
Núcleo estadouni dense	3.1	http://hl7.org/fhir/us/core/STU3.1.1/	Predeterminado	X	X	X	X	X	X	X	X
Núcleo estadouni dense	4.0	https://hl7.org/fhir/us/core/STU4/index.html	compatible	X	X	X	X	X	X	X	X
Núcleo estadouni dense	5.0	https://hl7.org/fhir/us/core/STU5.0.1/index.html	compatible	X	X	X	X	X	X	X	X
Núcleo estadouni dense	6.1	https://hl7.org/fhir/us/core/STU6.1/index.html	compatible	X	X	X	X	X	X	X	X
Núcleo estadouni dense	7.0	https://hl7.org/fhir/us/core/STU7/	compatible	X	X	X	X	X	X	X	X
Núcleo británico	2.0	https://simplifier.net/guide/uk-core-implementation-guide-stu2/Home/ProfilesandExtensions/Pro	compatible	X	X	X	X			X	X

Name	Ve	Guía de implementación	Funcionalidad	Est de EE (Ol	Est de EE (No de Vir	Oe de EE (Or	Asi Pa fico (Mi	Asi Pa fico (Sí	Ca (Ce	Eu (Irl	Europa (Londres)
		filesIndex?version=2.0.1									
Botón azul CARIN	1.1	http://hl7.org/fhir/us/carin-bb/STU1.1/	Predeterminado	X	X	X	X	X	X	X	X
Botón azul CARIN	1.0 2.0 2.1	https://hl7.org/fhir/us/carin-bb/history.html	compatible	X	X	X	X	X	X	X	X
Da Vinci Payer Data Exchange	1.0	https://hl7.org/fhir/us/davinci-pdex/	Predeterminado	X	X	X	X	X	X	X	X
Da Vinci Payer Data Exchange	2.0 2.1	https://hl7.org/fhir/us/davinci-pdex/history.html	compatible	X	X	X	X	X	X	X	X
Intercambio de registros de salud Da Vinci (HRex)	0.2	https://hl7.org/fhir/us/davinci-hrex/2020Sep/	Predeterminado	X	X	X	X	X	X	X	X
DaVinci Plan PDEX: Red	1.1	https://hl7.org/fhir/us/davinci-pdex-plan-net/STU1.1/	Predeterminado	X	X	X	X	X	X	X	X
DaVinci Plan PDEX Net	1.0	https://hl7.org/fhir/us/davinci-pdex-plan-net/STU1/	compatible	X	X	X	X	X	X	X	X

Name	Ve	Guía de implement ación	Funcionalidad	Est de EE (Ol	Est de EE (No de Vir	Oe de EE (Or	Asi Pa fico (Mi	Asi Pa fico (Sí	Ca (Ce	Eu (Irl	Europa (Londres)
DaVinci Formulario de medicamentos de EE. UU. Payer Data Exchange (PDeX)	1.1	https://hl7.org/fhir/us/davinci-drug-formulary/STU1.1/	Predeterminado	X	X	X	X	X	X	X	X
DaVinci Formulario de medicamentos de EE. UU. Payer Data Exchange (PDeX)	1.0 2.0 2.1	https://hl7.org/fhir/us/davinci-drug-formulary/history.html	compatible	X	X	X	X	X	X	X	X
Intercambio de Datos Clínicos Da Vinci (CDex)	2.1	https://build.fhir.org/ig/HL7/davinci-ecdx/index.html	Predeterminado	X	X	X	X	X	X	X	X

Name	Ve	Guía de implement ación	Funcionalidad	Est de EE (Ol	Est de EE (No de Vir	Oe de EE (Or	Asi Pa fico (Mi	Asi Pa fico (Sí	Ca (Ce	Eu (Irl	Europa (Londres)
Da Vinci Prior Authoriza tion Support (PAS) FHIR IG	2.1	https://hl7.org/fhir/us/davinci-pas/	Predeterminado	X	X	X	X	X	X	X	X
Guía de implement ación de HEDIS® del NCQA	0.3	https://www.ncqa.org/resources/hedis-ig-resource-page/	Predeterminado	X	X	X	X			X	X
Resumen de pacientes internaci onales (IPS)	Vo 2.0	https://hl7.org/fhir/uv/ips/2024Sep/	Predeterminado	X	X	X	X	X	X	X	X
Medida de calidad	5.0	https://registry.fhir.org/package/hl7.fhir.us.cqfmeasures%7C5.0.0	Predeterminado	X	X	X	X			X	X
Informes genómicos	3.0	https://build.fhir.org/ig/HL7/genomics-reporting/index.html	Predeterminado	X	X	X	X			X	X

Name	Ve	Guía de implement ación	Funcionalidad	Est de EE (Ol	Est de EE (No de Vir	Oe de EE (Or	Asi Pa fico (Mi	Asi Pa fico (Sí	Ca (Ce	Eu (Irl	Europa (Londres)
Misión Digital Ayushman Bharat (ABDM) de la Autoridad Nacional de Salud	2.0	https://www.nrces.in/ndhm/fhir/r4/index.html	Predeterminado	X	X	X	X			X	X
CA Core+	1.1	https://simplifier.net/ca-core	compatible						X		
CA:eReC Pan-Canadian ¿E? Referral-eConsult	1.1	https://simplifier.net/CA-eReC/~introduction	compatible						X		
Resumen para pacientes, edición canadiense - (PS-CA)	2.1	https://simplifier.net/PS-CA-R1/~introduction	compatible						X		

Name	Ve	Guía de implement ación	Funcionalidad	Est de EE (Ol	Est de EE (No de Vir	Oe de EE (Or	Asi Pa fico (M	Asi Pa fico (Sí	Ca (Ce	Eu (Irl	Europa (Londres)
Repositor io de medicamen tos de Ontario Digital Health	4.0	https://simplifier.net/ca-on-dhdr-r4/~intr oduction	compatible						X		
Núcleo AU	1.0	https://hl7.org.au/fhir/core/	compatible					X			
Gestión de la práctica de Magentus	1.2	https://fhir-vers ions.dev.geniesolu tions.io/1.2.16/downloads.html	compatible					X			

Validar los perfiles del FHIR especificados en un recurso

Para validar un perfil del FHIR, agréguelo al `profile` elemento de los recursos individuales utilizando la URL del perfil designada en la guía de implementación.

Los perfiles del FHIR se validan cuando se agrega un nuevo recurso a su banco de datos. Para agregar un nuevo recurso, puede usar la operación de la `StartFHIRImportJob` API, realizar una `POST` solicitud para agregar un nuevo recurso o `PUT` actualizar un recurso existente.

Example— Para ver a qué perfil del FHIR se hace referencia en un recurso

La URL del perfil se añade al `profile` elemento del par `"meta" : "profile"` clave-valor. Este recurso se ha truncado para mayor claridad.

```
{
```

```
    "resourceType": "Patient",
    "id": "abcd1234efgh5678hijk9012",
    "meta": {
        "lastUpdated": "2023-05-30T00:48:07.8443764-07:00",
        "profile": [
            "http://hl7.org/fhir/us/core/StructureDefinition/us-core-patient"
        ]
    }
}
```

Example— Cómo hacer referencia a un perfil FHIR compatible y no predeterminado

Para realizar la validación con un perfil no predeterminado compatible, añada la URL del perfil versionado al elemento. `meta.profile` La URL versionada incluye la URL del perfil base seguida de un carácter vertical (`|`) y el número de versión. Este recurso de ejemplo se ha truncado para mayor claridad.

```
{
    "resourceType": "ExplanationOfBenefit",
    "id": "sample-EOB",
    "meta": {
        "lastUpdated": "2024-02-02T05:56:09.4+00:00",
        "profile": [
            "http://hl7.org/fhir/us/carin-bb/StructureDefinition/C4BB-
ExplanationOfBenefit-Pharmacy|1.0.0"
        ]
    }
}
```

También puede incluir la URL versionada y la URL del perfil base. Ambos formatos son válidos.

```
{
    "resourceType": "ExplanationOfBenefit",
    "id": "sample-EOB",
    "meta": {
        "lastUpdated": "2024-02-02T05:56:09.4+00:00",
        "profile": [
            "http://hl7.org/fhir/us/carin-bb/StructureDefinition/C4BB-
ExplanationOfBenefit-Pharmacy|1.0.0",
            "http://hl7.org/fhir/us/carin-bb/StructureDefinition/C4BB-
ExplanationOfBenefit-Pharmacy"
        ]
```

```
    }
}
```

Tipos de recursos compatibles con FHIR R4 para HealthLake

En la siguiente tabla se enumeran los tipos de recursos del FHIR R4 compatibles con. AWS HealthLake Para obtener más información, consulte el índice de recursos en la documentación del FHIR R4.

Los tipos de recursos del FHIR R4 son compatibles con HealthLake

Cuenta	DetectedIssue	Factura	Profesional
ActivityDefinition	Dispositivo	Library	PractitionerRole
AdverseEvent	DeviceDefinition	Vinculación	Procedimiento
AllergyIntolerance	DeviceMetric	Enumeración	Procedencia
Cita	DeviceUseStatement	Ubicación	Cuestionario
AppointmentResponse	DeviceRequest	Medida	QuestionnaireResponse
AuditEvent - Ver nota	DiagnosticReport	MeasureReport	RelatedPerson
Binario	DocumentManifest	Multimedia	RequestGroup
BodyStructure	DocumentReference	¿Medicamento	ResearchStudy
Paquete: consulte la nota	EffectEvidenceSynthesis	MedicationAdministration	ResearchSubject
CapabilityStatement	¿Encuentro	MedicationDispense	RiskAssessment
CarePlan	Punto de conexión	MedicationKnowledge	RiskEvidenceSynthesis
CareTeam	EpisodeOfCare	MedicationRequest	Schedule
ChargeItem	EnrollmentRequest	MedicationStatement	ServiceRequest

ChargeItemDefinition	EnrollmentResponse	MessageHeader	Ranura
Reclamación	ExplanationOfBenefit	MolecularSequence	Ejemplar
ClaimResponse	FamilyMemberHistory	NutritionOrder	StructureDefinition
Comunicación	Indicador	Observación	StructureMap
CommunicationRequest	Objetivo	OperationOutcome - Ver nota	Substance
Composición	Group	Organización	SupplyDelivery
ConceptMap	GuidanceResponse	OrganizationAffiliation	SupplyRequest
Condición	HealthcareService	Parámetros: consulte la nota	Tarea
Consentimiento	ImagingStudy	Paciente	ValueSet
Contrato	Inmunización	PaymentNotice	VisionPrescription
Cobertura	ImmunizationEvaluation	PaymentReconciliation	VerificationResult - Ver nota
CoverageEligibilityRequest	ImmunizationRecommendation	Persona	
CoverageEligibilityResponse	InsurancePlan	PlanDefinition	

⚠ Especificaciones del FHIR y HealthLake

- No puede realizar `GET` ni `POST` solicitar el FHIR ni los tipos `OperationOutcome` de `Parameters` recursos.
- AuditEvent— Se puede crear o leer un AuditEvent recurso, pero no se puede actualizar ni eliminar.

- Paquete: hay varias formas de HealthLake gestionar las solicitudes de paquetes. Para obtener más información, consulte Agrupación de recursos del FHIR.
- VerificationResult— Este tipo de recurso solo es compatible con los almacenes de datos creados después del 9 de diciembre de 2023.

Parámetros de búsqueda FHIR R4 para HealthLake

Utilice la `search`interacción del FHIR para buscar un conjunto de recursos del FHIR en un banco de HealthLake datos en función de algunos criterios de filtrado. La `search` interacción se puede realizar mediante una solicitud `GET` o`POST`. Para las búsquedas que incluyan información de identificación personal (PII) o información de salud protegida (PHI), se recomienda utilizar `POST` solicitudes, ya que la PII y la PHI se agregan como parte del cuerpo de la solicitud y se cifran durante el tránsito.

> **Note**
>
> La `search` interacción entre el FHIR que se describe en este capítulo se basa en el estándar R4 del HL7 FHIR para el intercambio de datos de atención médica. Como es una representación de un servicio de la HL7 FHIR, no se ofrece a través de y. AWS CLI AWS SDKs Para obtener más información, consulte la documentación de `search`la API R4 RESTful del FHIR.
> También puede consultar almacenes HealthLake de datos con SQL mediante Amazon Athena. Para obtener más información, consulte Integración.

HealthLake admite el siguiente subconjunto de parámetros de búsqueda del FHIR R4. Para obtener más información, consulte Parámetros de búsqueda FHIR R4 para HealthLake.

Tipos de parámetros de búsqueda compatibles

En la siguiente tabla se muestran los tipos de parámetros de búsqueda admitidos en HealthLake.

Tipos de parámetros de búsqueda compatibles

Parámetro de búsqueda	Description (Descripción)
_id	ID del recurso (no es una URL completa)

Parámetro de búsqueda	Description (Descripción)
_Última actualización	Fecha de la última actualización. El servidor tiene discreción en cuanto a la precisión de los límites.
_tag	Busca por etiqueta de recurso.
_perfil	Busca todos los recursos etiquetados con un perfil.
_seguridad	Busque en las etiquetas de seguridad aplicadas a este recurso.
_fuente	Busque de dónde proviene el recurso.
_texto	Busque en la narrativa del recurso.
createdAt	Busque en la extensión personalizada CreatedAt.

> **Note**
>
> Los siguientes parámetros de búsqueda solo se admiten en los almacenes de datos creados después del 9 de diciembre de 2023: _security, _source, _text, CreatedAt.

En la siguiente tabla se muestran ejemplos de cómo modificar las cadenas de consulta en función de los tipos de datos especificados para un tipo de recurso determinado. Para mayor claridad, los caracteres especiales de la columna de ejemplos no se han codificado. Para que la consulta se realice correctamente, asegúrese de que la cadena de consulta se haya codificado correctamente.

Ejemplos de parámetros de búsqueda

Tipos de parámetros de búsqueda	Details	Ejemplos
Número	Busca un valor numérico en un recurso específico. Se	`[parameter]=100`

Tipos de parámetros de búsqueda	Details	Ejemplos

	observan cifras significativas. El número de dígitos significativos es específico por valor de parámetro de búsqueda, excluyendo los ceros iniciales. Se permiten prefijos de comparación.	`[parameter]=1e2`
		
		`[parameter]=lt100`

Fecha/ DateTime	Busca una fecha u hora específica. El formato esperado es `yyyy-mm-ddThh:mm:ss[Z|(+|-)hh:mm]` , pero puede variar.
	
	Acepta los siguientes tipos de datos: `datedateTime`,`instant`,`PeriyTiming`. Para obtener más información sobre el uso de estos tipos de datos en las búsquedas, consulta la fecha en la documentación de la RESTful API R4 del FHIR.
	
	Se permiten los prefijos de comparación.
		`[parameter]=eq2013-01-14`
		
		`[parameter]=gt2013-01-14T10:00`
		
		`[parameter]=ne2013-01-14`

Tipos de parámetros de búsqueda	Details	Ejemplos

Cadena	Busca una secuencia de caracteres distinguiendo mayúsculas de minúsculas.
	
	Admite ambos `HumanName` `Address` tipos. Para obtener más información, consulte la entrada del tipo de HumanName datos y las entradas del tipo de `Address` datos en la documentación del FHIR R4.
	
	Se admite la búsqueda avanzada mediante `:text` modificadores.
		`[base]/Patient?given=eve`
		
		`[base]/Patient?given:contains=eve`

Tipos de parámetros de búsqueda	Details	Ejemplos
Token	Busca una close-to-exact coincidencia con una cadena de caracteres, a menudo comparada con un par de valores de códigos médicos. La distinción entre mayúsculas y minúsculas está vinculada al sistema de código utilizado al crear una consulta. Las consultas basadas en subsuposiciones pueden ayudar a reducir los problemas relacionados con la distinción entre mayúsculas y minúsculas. Para mayor claridad, no se ha codificado. `|`	`[parameter]=[system]|[code]` : Aquí `[system]` se refiere a un sistema de codificación y `[code]` se refiere al valor de código que se encuentra dentro de ese sistema específico. `[parameter]=[code]` : En este caso, la entrada coincidirá con un código o un sistema. `[parameter]=|[code]` : En este caso, la entrada coincidirá con un código y la propiedad del sistema no tiene ningún identificador.

Compuesto	Busca varios parámetros dentro de un único tipo de recurso mediante los modificadores `$` y la `,` operación.
	
	Se permiten los prefijos de comparación.
		`/Patient?language=FR,NL&language=EN`
		
		`Observation?component-code-value-quantity=http://loinc.org|8480-6$lt60`
		
		`[base]/Group?characteristic-value=gender$mixed`

Tipos de parámetros de búsqueda	Details	Ejemplos

Cantidad	Busca un número, un sistema y un código como valores. Se requiere un número, pero el sistema y el código son opcionales. Basado en el tipo de datos de cantidad. Para obtener más información, consulta la sección Cantidad en la documentación del FHIR R4.
	
	Utiliza la siguiente sintaxis asumida `[parameter]=[prefix][number]|[system]|[code]`
		`[base]/Observation?value-quantity=5.4|http://unitsofmeasure.org|mg`
		
		`[base]/Observation?value-quantity=5.4|http://unitsofmeasure.org|mg`
		
		`[base]/Observation?value-quantity=5.4|http://unitsofmeasure.org|mg`
		
		`[base]/Observation?value-quantity=le5.4|http://unitsofmeasure.org|mg`

Referencia	Busca referencias a otros recursos.	`[base]/Observation?subject=Patient/23`
		
		`test`

URI	Busca una cadena de caracteres que identifique de forma inequívoca un recurso concreto.	`[base]/ValueSet?url=http://acme.org/fhir/ValueSet/123`
Especial	Búsquedas basadas en extensiones de PNL médicas integradas.	

Parámetros de búsqueda avanzada compatibles con HealthLake

HealthLake admite los siguientes parámetros de búsqueda avanzada.

Name	Descripción	Ejemplo	Funcionalidad
`_includ`	Se utiliza para solicitar que se devuelvan recursos adicionales en una solicitud de búsqueda. Devuelve los recursos a los que hace referencia la instancia de recurso de destino.	`Encounter?_include=Encounter:subject`	
`_revinc de`	Se utiliza para solicitar que se devuelvan recursos adicionales en una solicitud de búsqueda. Devuelve los recursos que hacen referencia a la instancia de recurso principal.	`Patient?_id=` **`patient-identifier`** `&_revinclude=Encounter:patient`	
`_summar`	El resumen se puede utilizar para solicitar un subconjunto del recurso.	`Patient?_summary=text`	Se admiten los siguientes parámetros de resumen:`_summary=true` ,, `_summary=false` `_summary=text` ,`_summary=data` .
`_elemen`	Solicite que se devuelva un conjunto específico de elementos como parte de un recurso en los resultados de la búsqueda.	`Patient?_elements=identifier,active,link`	
`_total`	Devuelve el número de recursos que coinciden con los parámetros de búsqueda.	`Patient?_total=accurate`	Support`_total=accurate` ,`_total=none` .

Name	Descripción	Ejemplo	Funcionalidad
`_sort`	Indique el orden de clasificación de los resultados de búsqueda devueltos mediante una lista separada por comas. El `-` prefijo se puede usar para cualquier regla de clasificación de la lista separada por comas para indicar un orden descendente.	`Observation?_sort=status,-date`	Support ordenar por campos con tipos`Number, String, Quantity, Token, URI, Reference` . La clasificación por solo `Date` se admite en los almacenes de datos creados después del 9 de diciembre de 2023. Support hasta 5 reglas de clasificación.
`_count`	Controle cuántos recursos se devuelven por página del paquete de búsqueda.	`Patient?_count=100`	El tamaño máximo de página es 100.
`chainin`	Elementos de búsqueda de los recursos referenciados. `.`Dirige la búsqueda encadenada al elemento dentro del recurso referenciado.	`DiagnosticReport?subject:Patient.name=peter`	
`reverse chainin (_has)`	Busque un recurso en función de los elementos de los recursos que hacen referencia a él.	`Patient?_has:Observation:patient:code=1234-5`	

`_include`

Si se utiliza `_include` en una consulta de búsqueda, también se pueden obtener recursos adicionales específicos del FHIR. Se usa `_include` para incluir recursos que están enlazados hacia adelante.

Example— Para localizar _include a los pacientes o al grupo de pacientes a los que se les ha diagnosticado tos

Debería buscar en el tipo de `Condition` recurso especificando el código de diagnóstico de la tos y, a continuación, `_include` especificando que también desea que se devuelva ese diagnóstico. `subject` En el tipo `Condition` de recurso, `subject` se refiere al tipo de recurso del paciente o al tipo de recurso del grupo.

Para mayor claridad, los caracteres especiales del ejemplo no se han codificado. Para que la consulta se realice correctamente, asegúrese de que la cadena de consulta se haya codificado correctamente.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
Condition?code=49727002&_include=Condition:subject
```

`_include:iterate`Modificador

El `_include:iterate` modificador permite la inclusión recursiva de los recursos referenciados en dos niveles. Por ejemplo:

```
GET /ServiceRequest?
identifier=025C0931195&_include=ServiceRequest:requester&_include:iterate=PractitionerRole:prac
```

devolverá el ServiceRequest, está asociado PractitionerRole (mediante la referencia del solicitante) y, a continuación, incluirá de forma recursiva al profesional al que hace referencia. PractitionerRole Este modificador está disponible para todos los tipos de recursos en. HealthLake

`_include=*`Modificador

El `_include=*` modificador es un comodín que incluye automáticamente todos los recursos a los que hacen referencia directamente los resultados de la búsqueda. Por ejemplo:

```
GET /ServiceRequest?specimen.accession=12345&_include=*
```

devolverá la coincidencia ServiceRequest junto con todos los recursos a los que haga referencia (como pacientes, profesionales, muestras, etc.) sin necesidad de especificar cada ruta de referencia de forma individual. Este modificador está disponible para todos los tipos de recursos en HealthLake.

`_revinclude`

Si se utiliza `_revinclude` en una consulta de búsqueda, también se pueden devolver recursos adicionales del FHIR especificados. `_revincludeUtilícelo` para incluir recursos que estén enlazados hacia atrás.

Example— _revinclude Para incluir tipos de recursos relacionados con el encuentro y la observación vinculados a un paciente específico

Para realizar esta búsqueda, primero debe definir a la persona `Patient` especificando su identificador en el parámetro `_id` de búsqueda. Luego, especificaría recursos adicionales del FHIR utilizando la estructura `Encounter:patient` y`Observation:patient`.

Para mayor claridad, los caracteres especiales del ejemplo no se han codificado. Para que la consulta se realice correctamente, asegúrese de que la cadena de consulta se haya codificado correctamente.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
Patient?_id=patient-
identifier&_revinclude=Encounter:patient&_revinclude=Observation:patient
```

`_summary`

El uso `_summary` en una consulta de búsqueda permite al usuario solicitar un subconjunto del recurso FHIR. Puede contener uno de los siguientes valores:. `true, text, data, false` Cualquier otro valor se considerará inválido. Los recursos devueltos se marcarán con una `'SUBSETTED'` meta.tag para indicar que los recursos están incompletos.

- `true`: Devuelve todos los elementos admitidos que estén marcados como «resumen» en la definición básica de los recursos.
- `text`: Devuelve solo los elementos «texto», «identificador» y «meta» y solo los elementos obligatorios de nivel superior.
- `data`: Devuelve todas las partes excepto el elemento «texto».
- `false`: Devuelve todas las partes de los recursos

En una sola solicitud de búsqueda, `_summary=text` no se puede combinar con `_include` los parámetros `_revinclude` de búsqueda.

Example— Obtenga el elemento «texto» de los recursos para pacientes en un almacén de datos.

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

`_elements`

El uso `_elements` en una consulta de búsqueda permite solicitar elementos de recursos específicos del FHIR. Los recursos devueltos se marcarán con una `'SUBSETTED'` etiqueta meta para indicar que los recursos están incompletos.

El `_elements` parámetro consiste en una lista de nombres de elementos base separados por comas, como los elementos definidos en el nivel raíz del recurso. Solo se devolverán los elementos que estén en la lista. Si los valores de los `_elements` parámetros contienen elementos no válidos, el servidor los ignorará y devolverá los elementos obligatorios y los elementos válidos.

`_elements`no se aplicará a los recursos incluidos (recursos devueltos cuyo modo de búsqueda es`include`).

En una sola solicitud de búsqueda, `_elements` no se puede combinar con los parámetros `_summary` de búsqueda.

Example— Obtenga los elementos «identificadores», «activos» y «enlaces» de los recursos para pacientes en su almacén de HealthLake datos.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?
_elements=identifier,active,link
```

`_total`

Si se utiliza `_total` en una consulta de búsqueda, se devolverá el número de recursos que coinciden con los parámetros de búsqueda solicitados. HealthLake devolverá el número total de recursos coincidentes (recursos devueltos cuyo modo de búsqueda es`match`) en la respuesta `Bundle.total` de búsqueda.

`_total`admite los valores `accurate` de los `none` parámetros. `_total=estimate`no es compatible. Cualquier otro valor se considerará inválido. `_total`no se aplica a los recursos incluidos (recursos devueltos cuyo modo de búsqueda es`include`).

Example— Obtenga el número total de recursos para pacientes en un almacén de datos:

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

_sort

Al `_sort` utilizarlos en la consulta de búsqueda, se ordenan los resultados en un orden específico. Los resultados se ordenan según la lista de reglas de clasificación separadas por comas en orden de prioridad. Las reglas de clasificación deben ser parámetros de búsqueda válidos. Los demás valores se considerarán inválidos.

En una sola solicitud de búsqueda, puede utilizar hasta 5 parámetros de búsqueda de clasificación. Si lo desea, puede utilizar un `-` prefijo para indicar el orden descendente. El servidor ordenará en orden ascendente de forma predeterminada.

Los tipos de parámetros de búsqueda de clasificación admitidos son:`Number, String, Date, Quantity, Token, URI, Reference`. Si un parámetro de búsqueda hace referencia a un elemento que está anidado, este parámetro de búsqueda no se admite para la ordenación. Por ejemplo, busque por «nombre» del tipo de recurso Paciente hace referencia a Patient.El elemento NAME con el tipo de HumanName datos se considera anidado. Por lo tanto, no se permite ordenar los recursos de los pacientes por «nombre».

Example— Coloque los recursos para pacientes en un almacén de datos y ordénelos por fecha de nacimiento en orden ascendente:

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

_count

El parámetro `_count` se define como una instrucción al servidor sobre cuántos recursos deben devolverse en una sola página.

El tamaño máximo de página es 100. Los valores superiores a 100 no son válidos. `_count=0`no se admite.

Example— Busque el recurso para pacientes y establezca el tamaño de la página de búsqueda en 25:

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?_count=25
```

Chaining and Reverse Chaining(_has)

El encadenamiento y el encadenamiento inverso en el FHIR proporcionan una forma más eficiente y compacta de obtener datos interconectados, lo que reduce la necesidad de realizar múltiples consultas independientes y hace que la recuperación de datos sea más cómoda para los desarrolladores y los usuarios.

Si algún nivel de recursión arroja más de 100 resultados, HealthLake devolverá 4xx para evitar que el almacén de datos se sobrecargue y provoque múltiples paginaciones.

Example— Encadenar: obtiene todo DiagnosticReport lo que hace referencia a un paciente cuyo nombre es peter.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/DiagnosticReport?
subject:Patient.name=peter
```

Example— Encadenamiento inverso: obtenga los recursos para pacientes, donde al menos una observación hace referencia al recurso para pacientes, donde la observación tiene el código 1234 y donde la observación hace referencia al recurso para pacientes en el parámetro de búsqueda de pacientes.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?
_has:Observation:patient:code=1234
```

Modificadores de búsqueda compatibles

Los modificadores de búsqueda se utilizan con campos basados en cadenas. Todos los modificadores de búsqueda HealthLake utilizan una lógica booleana. Por ejemplo, puede especificar `:contains` que un campo de cadena más grande incluya una cadena pequeña para que se incluya en los resultados de la búsqueda.

Modificadores de búsqueda compatibles

Modificador de búsqueda	Tipo
:falta	Todos los parámetros excepto `Composite`
:exacto	Cadena
:contiene	Cadena
:no	Token
:texto	Token
:identificador	Referencia
:abajo	URI

Comparadores de búsqueda compatibles

Puede utilizar los comparadores de búsqueda para controlar la naturaleza de la coincidencia en una búsqueda. Puede utilizar los comparadores al buscar en los campos de número, fecha y cantidad. En la siguiente tabla se enumeran los comparadores de búsqueda y sus definiciones compatibles con. HealthLake

Comparadores de búsqueda compatibles

Comparador de búsquedas	Description (Descripción)
eq	El valor del parámetro en el recurso es igual al valor proporcionado.
ne	El valor del parámetro en el recurso no es igual al valor proporcionado.
gt	El valor del parámetro en el recurso es mayor que el valor proporcionado.
lt	El valor del parámetro en el recurso es inferior al valor proporcionado.

Comparador de búsquedas	Description (Descripción)
Edad	El valor del parámetro del recurso es mayor o igual al valor proporcionado.
le	El valor del parámetro en el recurso es menor o igual al valor proporcionado.
sa	El valor del parámetro del recurso comienza después del valor proporcionado.
eb	El valor del parámetro en el recurso termina antes del valor proporcionado.

Los parámetros de búsqueda del FHIR no son compatibles con HealthLake

HealthLake admite todos los parámetros de búsqueda del FHIR, a excepción de los que se muestran en la siguiente tabla. Para obtener una lista completa de los parámetros de búsqueda del FHIR, consulte el registro de parámetros de búsqueda del FHIR.

Parámetros de búsqueda no compatibles

Composición del paquete	Ubicación: cerca
Identificador de paquete	C onsent-source-reference
Paquete: mensaje	Paciente contratado
Tipo de paquete	Contenido del recurso
Marca de tiempo del paquete	Consulta de recursos

FHIR R4 para **`$operations`** HealthLake

Las operaciones FHIR $ (también denominadas «operaciones en dólares») son funciones especiales del servidor que van más allá de las operaciones CRUD (`Create`,, `ReadUpdate`,`Delete`) estándar de la especificación FHIR. Estas operaciones se identifican con el prefijo «$» y permiten un procesamiento complejo, la transformación de datos y operaciones masivas que serían difíciles

o ineficientes de realizar con llamadas a la API REST estándar. $ Las operaciones se pueden invocar a nivel del sistema, a nivel de tipo de recurso o en instancias de recursos específicas, lo que proporciona formas flexibles de interactuar con los datos del FHIR. AWS HealthLake admite varios FHIR`$operations`. Consulte cada una de las páginas individuales que aparecen a continuación para obtener más información.

Temas

- Funcionamiento FHIR R4 $attribution-status para HealthLake
- Eliminar tipos de recursos con $bulk-delete
- operación $bulk-member-match para HealthLake
- Funcionamiento FHIR R4 $confirm-attribution-list para HealthLake
- Funcionamiento FHIR R4 $davinci-data-export para HealthLake
- Generación de documentos clínicos con $document
- Eliminar recursos de forma permanente con $erase
- Obtener datos de pacientes con Patient/$everything
- Recuperación de ValueSet códigos con $expand
- Exportación HealthLake de datos con FHIR $export
- $inquireOperación FHIR para HealthLake
- Recuperación de detalles conceptuales con $lookup
- $member-addoperación para HealthLake
- $member-matchoperación para HealthLake
- $member-removeoperación para HealthLake
- Eliminar los recursos del compartimento de pacientes con $purge
- $questionnaire-packageOperación FHIR para HealthLake
- $submitOperación FHIR para HealthLake
- Validación de los recursos del FHIR con $validate

Funcionamiento FHIR R4 **`$attribution-status`** para HealthLake

Recupera el estado de atribución de un miembro específico y devuelve un paquete que contiene todos los recursos de atribución relacionados con el paciente. Esta operación forma parte de la implementación de la Lista IG 2.1.0 de la Lista de Atribución de Miembros (ATR) del FHIR.

Punto de conexión

```
POST [base]/Group/[id]/$attribution-status
```

Parámetros de la solicitud

La operación acepta los siguientes parámetros opcionales:

Parámetro	Tipo	Description (Descripción)
memberId	Identificador	El MemberId del miembro para el que se solicita el estado de atribución
Referencia del paciente	Referencia	Referencia al recurso para pacientes en el sistema del productor

> **Note**
>
> `patientReference`Se puede proporcionar uno `memberId` o ambos, o ambos con fines de validación.

Solicitud de muestra

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "memberId",
      "valueIdentifier": {
        "system": "http://example.org",
        "value": "MBR123456789"
      }
    },
    {
      "name": "patientReference",
      "valueReference": {
        "reference": "Patient/patient-123",
        "display": "John Doe"
```

```
      }
    }
  ]
}
```

Respuesta

Devuelve un paquete que contiene recursos de atribución relacionados con el paciente:

Recurso	Cardinalidad	Ubicación
Paciente	1..1	Grupo, miembro, entidad
Cobertura	0.1..	Group.Member.Extension: referencia de cobertura
Organization/Practitioner/PractitionerRole	0.1..	group.member.extension: proveedor atribuido
¿Algún recurso	0.1..	Group.Member.Extension: datos asociados

Respuesta de ejemplo

```
{
  "resourceType": "Bundle",
  "id": "bundle-response",
  "meta": {
    "lastUpdated": "2014-08-18T01:43:33Z"
  },
  "type": "collection",
  "entry": [
    {
      "fullUrl": "http://example.org/fhir/Patient/12423",
      "resource": {
        "resourceType": "Patient",
        "id": "12423",
        "meta": {
          "versionId": "1",
          "lastUpdated": "2014-08-18T01:43:31Z"
        },
```

```
        "active": true,
        "name": [
          {
            "use": "official",
            "family": "Chalmers",
            "given": ["Peter", "James"]
          }
        ],
        "gender": "male",
        "birthDate": "1974-12-25"
      }
    },
    {
      "fullUrl": "http://example.org/fhir/Coverage/123456",
      "resource": {
        "resourceType": "Coverage",
        "id": "1"
        // ... additional Coverage resource details
      }
    },
    {
      "fullUrl": "http://example.org/fhir/Organization/666666",
      "resource": {
        "resourceType": "Organization",
        "id": "2"
        // ... additional Organization resource details
      }
    }
  ]
}
```

Gestión de errores

La operación gestiona las siguientes condiciones de error:

Error	Estado HTTP	Description (Descripción)
Solicitud de operación no válida	400	Estructura o parámetros de solicitud no conformes
No se encontró el recurso de grupo	404	El ID de grupo especificado no existe

Error	Estado HTTP	Description (Descripción)
No se encontró el recurso para pacientes	404	La referencia de paciente especificada no existe

Autorización y seguridad

Requisitos de SMART Scope

Los clientes deben tener los privilegios adecuados para leer los recursos del grupo y los recursos de atribución relacionados

Los mecanismos de autorización estándar del FHIR se aplican a todas las operaciones

Eliminar tipos de recursos con **`$bulk-delete`**

AWS HealthLake admite la `$bulk-delete` operación, lo que permite eliminar todos los recursos de un tipo específico de un almacén de datos. Esta operación resulta especialmente útil cuando se necesita:

- Realice auditorías y limpiezas estacionales
- Gestione el ciclo de vida de los datos a escala
- Elimine tipos de recursos específicos
- Cumpla con las políticas de retención de datos

De uso

La `$bulk-delete` operación se puede invocar mediante los métodos POST:

```
POST [base]/[ResourceType]/$bulk-delete?isHardDelete=false&deleteAuditEvent=true
```

Parameters

Parámetro	Tipo	Obligatorio/a	Predeterminado	Description (Descripción)
`isHardDelete`	booleano	No	false	Si es verdadero, elimina permanentemente los recursos del almacenamiento
`deleteAuditEvent`	booleano	No	true	Si es verdadero, elimina los eventos de auditoría asociados
`_since`	cadena	No	Hora de creación del almacén de datos	Cuando se introduce, selecciona la hora límite de inicio para buscar los recursos en función de su hora de última modificación. No se puede usar ni al principio ni al final
`start`	cadena	No	Hora de creación del almacén de datos	Cuando se introduce, selecciona la hora límite para buscar los recursos en función de su hora de última modificación. Se puede usar con el final
`end`	cadena	No	Tiempo de presentación de trabajos	Cuando se introduce, selecciona la hora límite de finalización para buscar los recursos en función de su hora de última modificación

Ejemplos

Solicitud de ejemplo

```
POST [base]/Observation/$bulk-delete?isHardDelete=false
```

Respuesta de ejemplo

```
{
      "jobId": "jobId",
      "jobStatus": "SUBMITTED"
    }
```

Estado del trabajo

Para comprobar el estado de un trabajo de eliminación masiva:

```
GET [base]/$bulk-delete/[jobId]
```

La operación devuelve información sobre el estado del trabajo:

```
{
      "datastoreId": "datastoreId",
      "jobId": "jobId",
      "status": "COMPLETED",
      "submittedTime": "2025-10-09T15:09:51.336Z"
    }
```

Comportamiento

La `$bulk-delete` operación:

1. Procesa de forma asíncrona para gestionar grandes volúmenes de recursos
2. Mantiene las transacciones ACID para garantizar la integridad de los datos
3. Proporciona un seguimiento del estado de los trabajos con recuentos de eliminaciones de recursos
4. Soporta los modos de borrado suave y duro
5. Incluye un registro de auditoría completo de las actividades de eliminación
6. Permite la eliminación selectiva de versiones históricas y eventos de auditoría

Registro de auditoría

La `$bulk-delete` operación se registra como Inicio FHIRBulk DeleteJob y Descripción FHIRBulk DeleteJob con información detallada sobre la operación.

Limitaciones

- Si `isHardDelete` se establece en True, los recursos eliminados de forma permanente no aparecerán en los resultados de búsqueda ni en `_history` las consultas.
- Es posible que los recursos que se eliminen mediante esta operación no estén accesibles temporalmente durante el procesamiento
- La medición del almacenamiento solo se ajusta en las versiones históricas; deleteVersionHistory =false no ajustará el almacenamiento en el almacén de datos

`operación $bulk-member-match para` HealthLake

AWS HealthLake admite la `$bulk-member-match` operación de procesar las solicitudes de emparejamiento de varios miembros de forma asíncrona. Esta operación permite a las organizaciones de atención médica comparar de manera eficiente los identificadores únicos de cientos de miembros de diferentes sistemas de salud utilizando información demográfica y de cobertura en una sola solicitud masiva. Esta capacidad es esencial para el intercambio de datos entre pagadores a gran escala, las transiciones de los miembros y los requisitos de conformidad con los CMS, y sigue la especificación de la FHIR.

> **Note**
>
> La `$bulk-member-match` operación se basa en una especificación subyacente del FHIR que actualmente se encuentra en fase experimental y está sujeta a cambios. A medida que la especificación evolucione, el comportamiento y la interfaz de esta API se actualizarán en consecuencia. Se recomienda a los desarrolladores que supervisen las notas de la HealthLake versión de AWS y las actualizaciones de las especificaciones del FHIR pertinentes para mantenerse informados de cualquier cambio que pueda afectar a sus integraciones.

Esta operación resulta especialmente útil cuando necesita:

- Procese la comparación de miembros a gran escala durante los períodos de inscripción abierta
- Facilite las transiciones masivas de miembros entre pagadores
- Support con los requisitos de intercambio de datos de conformidad con los CMS a gran escala
- Haga coincidir de manera eficiente las cohortes de miembros en todas las redes de atención médica

- Minimice las llamadas a la API y mejore la eficiencia operativa en escenarios de coincidencia de gran volumen

De uso

La `$bulk-member-match` operación es una operación asíncrona que se invoca en los recursos del grupo mediante el método POST:

```
POST [base]/Group/$bulk-member-match
```

Tras enviar una solicitud de coincidencia masiva, puedes sondear el estado del trabajo mediante:

```
GET [base]/$bulk-member-match-status/{jobId}
```

Parámetros admitidos

HealthLake admite los siguientes `$bulk-member-match` parámetros del FHIR:

Parámetro	Tipo	Obligatorio	Description (Descripción)
`MemberPatient`	Paciente	Sí	Recurso para pacientes que contiene información demográfica sobre el miembro al que se va a vincular.
`CoverageToMatch`	Cobertura	Sí	Recurso de cobertura que se utilizará para compararlo con los registros existentes.
`CoverageToLink`	Cobertura	No	El recurso de cobertura se vinculará durante el proceso de emparejamiento.
`Consent`	Consentimiento	Sí	También se almacena el recurso de consentimiento para fines de autorización. Esto es diferente de la `$member-match` operación individual en la que no se requiere el consentimiento.

Solicitud POST para enviar un trabajo de emparejamiento masivo de miembros

En el siguiente ejemplo, se muestra una solicitud POST para enviar un trabajo de afiliación masiva de miembros. Cada miembro está incluido en un `MemberBundle` parámetro que contiene los `Consent` recursos necesarios `MemberPatient` y uno opcional`CoverageToLink`. `CoverageToMatch`

```
POST [base]/Group/$bulk-member-match
Content-Type: application/fhir+json
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "MemberBundle",
      "part": [
        {
          "name": "MemberPatient",
          "resource": {
            "resourceType": "Patient",
            "identifier": [
              {
                "system": "http://example.org/patient-id",
                "value": "patient-0"
              }
            ],
            "name": [
              {
                "family": "Smith",
                "given": ["James"]
              }
            ],
            "gender": "male",
            "birthDate": "1950-01-01"
          }
        },
        {
          "name": "CoverageToMatch",
          "resource": {
            "resourceType": "Coverage",
            "status": "active",
            "identifier": [
              {
                "system": "http://example.org/coverage-id",
```

```
                "value": "cov-0"
              }
            ],
            "subscriberId": "sub-0",
            "beneficiary": {
              "reference": "Patient/patient123"
            },
            "relationship": {
              "coding": [
                {
                  "system": "http://terminology.hl7.org/CodeSystem/subscriber-
relationship",
                  "code": "self"
                }
              ]
            },
            "payor": [
              {
                "reference": "Organization/org123"
              }
            ]
          }
        },
        {
          "name": "Consent",
          "resource": {
            "resourceType": "Consent",
            "status": "active",
            "scope": {
              "coding": [
                {
                  "system": "http://terminology.hl7.org/CodeSystem/consentscope",
                  "code": "patient-privacy"
                }
              ]
            },
            "category": [
              {
                "coding": [
                  {
                    "system": "http://terminology.hl7.org/CodeSystem/v3-ActCode",
                    "code": "IDSCL"
                  }
                ]
```

```
            }
          ],
          "patient": {
            "reference": "Patient/patient123"
          },
          "performer": [
            {
              "reference": "Patient/patient123"
            }
          ],
          "sourceReference": {
            "reference": "http://example.org/DocumentReference/consent-source"
          },
          "policy": [
            {
              "uri": "http://hl7.org/fhir/us/davinci-hrex/StructureDefinition-hrex-
consent.html#regular"
            }
          ],
          "provision": {
            "type": "permit",
            "period": {
              "start": "2024-01-01",
              "end": "2025-12-31"
            },
            "actor": [
              {
                "role": {
                  "coding": [
                    {
                      "system": "http://terminology.hl7.org/CodeSystem/provenance-
participant-type",
                      "code": "performer"
                    }
                  ]
                },
                "reference": {
                  "identifier": {
                    "system": "http://hl7.org/fhir/sid/us-npi",
                    "value": "9876543210"
                  },
                  "display": "Old Health Plan"
                }
              },
```

```
                {
                  "role": {
                    "coding": [
                      {
                        "system": "http://terminology.hl7.org/CodeSystem/v3-
ParticipationType",
                        "code": "IRCP"
                      }
                    ]
                  },
                  "reference": {
                    "identifier": {
                      "system": "http://hl7.org/fhir/sid/us-npi",
                      "value": "0123456789"
                    },
                    "display": "New Health Plan"
                  }
                }
              ],
              "action": [
                {
                  "coding": [
                    {
                      "system": "http://terminology.hl7.org/CodeSystem/consentaction",
                      "code": "disclose"
                    }
                  ]
                }
              ]
            }
          }
        },
        {
          "name": "CoverageToLink",
          "resource": {
            "resourceType": "Coverage",
            "status": "active",
            "identifier": [
              {
                "system": "http://example.org/coverage-link-id",
                "value": "cov-link-0"
              }
            ],
            "subscriberId": "new-sub-0",
```

```
            "beneficiary": {
              "reference": "Patient/patient123"
            },
            "relationship": {
              "coding": [
                {
                  "system": "http://terminology.hl7.org/CodeSystem/subscriber-
relationship",
                  "code": "self"
                }
              ]
            },
            "payor": [
              {
                "identifier": {
                  "system": "http://hl7.org/fhir/sid/us-npi",
                  "value": "0123456789"
                },
                "display": "New Health Plan"
              }
            ]
          }
        }
      ]
    }
  ]
}
```

Respuesta al trabajo completada con salida

Cuando se completa el trabajo, la respuesta incluye los metadatos del trabajo y un recurso de parámetros del FHIR que contiene tres recursos del grupo que clasifican los resultados de la coincidencia.

```
{
  "datastoreId": "datastoreId",
  "jobId": "jobId",
  "status": "COMPLETED",
  "submittedTime": "2026-03-20T18:45:26.321Z",
  "numberOfMembers": 3,
  "numberOfMembersProcessedSuccessfully": 3,
  "numberOfMembersWithCustomerError": 0,
  "numberOfMembersWithServerError": 0,
```

```
  "output": {
    "resourceType": "Parameters",
    "meta": {
      "profile": [
        "http://hl7.org/fhir/us/davinci-pdex/StructureDefinition/pdex-parameters-multi-
member-match-bundle-out"
      ]
    },
    "parameter": [
      {
        "name": "MatchedMembers",
        "resource": {
          "resourceType": "Group",
          "id": "group1",
          "text": {
            "status": "generated",
            "div": "<div xmlns=\"http://www.w3.org/1999/xhtml\">Matched members group</
div>"
          },
          "contained": [
            {
              "resourceType": "Patient",
              "id": "1",
              "identifier": [
                {
                  "system": "http://example.org/patient-id",
                  "value": "patient-0"
                }
              ],
              "name": [
                {
                  "family": "Smith",
                  "given": ["James"]
                }
              ],
              "gender": "male",
              "birthDate": "1950-01-01"
            }
          ],
          "type": "person",
          "actual": true,
          "code": {
            "coding": [
              {
```

```
                "system": "http://hl7.org/fhir/us/davinci-pdex/CodeSystem/
PdexMultiMemberMatchResultCS",
                "code": "match",
                "display": "Matched"
              }
            ]
          },
          "quantity": 1,
          "member": [
            {
              "entity": {
                "extension": [
                  {
                    "url": "http://hl7.org/fhir/us/davinci-pdex/StructureDefinition/
base-ext-match-parameters",
                    "valueReference": {
                      "reference": "#1"
                    }
                  }
                ],
                "reference": "Patient/patient123"
              }
            }
          ]
        }
      },
      {
        "name": "NonMatchedMembers",
        "resource": {
          "resourceType": "Group",
          "id": "Group2",
          "text": {
            "status": "generated",
            "div": "<div xmlns=\"http://www.w3.org/1999/xhtml\">Non-matched members
 group</div>"
          },
          "contained": [
            {
              "resourceType": "Patient",
              "id": "1",
              "identifier": [
                {
                  "system": "http://example.org/patient-id",
                  "value": "patient-501"
```

```
                }
              ],
              "name": [
                {
                  "family": "Carter",
                  "given": ["Emily"]
                }
              ],
              "gender": "female",
              "birthDate": "1985-06-15"
            }
          ],
          "type": "person",
          "actual": true,
          "code": {
            "coding": [
              {
                "system": "http://hl7.org/fhir/us/davinci-pdex/CodeSystem/
PdexMultiMemberMatchResultCS",
                "code": "nomatch",
                "display": "Not Matched"
              }
            ]
          },
          "quantity": 1,
          "member": [
            {
              "entity": {
                "extension": [
                  {
                    "url": "http://hl7.org/fhir/us/davinci-pdex/StructureDefinition/
base-ext-match-parameters",
                    "valueReference": {
                      "reference": "#1"
                    }
                  }
                ],
                "reference": "Patient/patient123"
              }
            }
          ]
        }
      },
      {
```

```
        "name": "ConsentConstrainedMembers",
        "resource": {
          "resourceType": "Group",
          "id": "group3",
          "text": {
            "status": "generated",
            "div": "<div xmlns=\"http://www.w3.org/1999/xhtml\">Consent constrained
 members group</div>"
          },
          "contained": [
            {
              "resourceType": "Patient",
              "id": "1",
              "identifier": [
                {
                  "system": "http://example.org/patient-id",
                  "value": "patient-502"
                }
              ],
              "name": [
                {
                  "family": "Nguyen",
                  "given": ["David"]
                }
              ],
              "gender": "male",
              "birthDate": "1972-11-22"
            }
          ],
          "type": "person",
          "actual": true,
          "code": {
            "coding": [
              {
                "system": "http://hl7.org/fhir/us/davinci-pdex/CodeSystem/
PdexMultiMemberMatchResultCS",
                "code": "consentconstraint",
                "display": "Consent Constraint"
              }
            ]
          },
          "quantity": 1,
          "member": [
            {
```

```
              "entity": {
                "extension": [
                  {
                    "url": "http://hl7.org/fhir/us/davinci-pdex/StructureDefinition/
base-ext-match-parameters",
                    "valueReference": {
                      "reference": "#1"
                    }
                  }
                ],
                "reference": "Patient/123"
              }
            }
          ]
        }
      }
    ]
  }
}
```

¿Cómo HealthLake clasifica los miembros en grupos de salida

Cada miembro presentado en una `$bulk-member-match` solicitud se evalúa mediante un proceso secuencial. El resultado de cada paso determina en qué grupo de salida se encuentra el miembro.

1. Validación estructural: ¿se MemberBundle ajusta a los perfiles requeridos? En caso de fallo: error (no en ningún grupo).
2. Coincidencia de pacientes: ¿se pueden HealthLake encontrar pacientes que coincidan con los datos demográficos enviados? Sobre el fracaso: NonMatchedMembers.
3. Confirmación de cobertura: ¿se puede HealthLake limitar a exactamente un paciente con validez CoverageToMatch? En caso de fallo: NonMatchedMembers.
4. Evaluación del consentimiento: ¿el consentimiento presentado es honorable en este momento? (estado = activo, el período abarca la fecha actual, el ejecutante puede ser validado). En caso de fallo: ConsentConstrainedMembers.
5. Éxito: se aprueban todos los controles. El consentimiento se almacena en el almacén de datos. Miembro ingresado. MatchedMembers

Principio clave: un miembro solo puede aparecer en un destino. El primer paso que falle determina la ubicación. Los miembros que no aprueben los pasos 2 o 3 nunca serán incluidos en ese grupo

ConsentConstrainedMembers ; ese grupo es exclusivamente para los miembros que hayan conseguido emparejarse con éxito, pero cuyo consentimiento no podrá respetarse.

Detalles de la evaluación del consentimiento (paso 4):

- Comprobación 1: Estado del consentimiento: ¿es `Consent.status` igual a «activo»? Si no → ConsentConstrainedMembers.
- Verificación 2: Período de provisión: ¿`provision.period`cubre la fecha actual? Si la fecha actual es anterior `period.start` o posterior a `period.end` → ConsentConstrainedMembers.
- Comprobación 3: Validación del ejecutante: ¿Se puede validar la `Consent.performer` referencia? Si el recurso al que se hace referencia no se encuentra en el almacén de datos o no está asociado al paciente coincidente →. ConsentConstrainedMembers

Se deben pasar todos los controles para poder ingresar al miembro MatchedMembers y para almacenar el consentimiento.

Comportamiento de coincidencia de cobertura

Durante la comparación de miembros, solo `CoverageToMatch` se valida con el almacén de datos del pagador que responde. `CoverageToLink`pertenece al new/requesting pagador y no se valida con el almacén de datos del pagador anterior. Incluirlo `CoverageToLink` en la solicitud no afectará a los resultados coincidentes.

Cada combinación de paciente y cobertura de la solicitud se procesa de forma independiente. El mismo paciente puede presentarse varias veces con diferentes planes de cobertura, y cada solicitud recibe su propio resultado en función de su cobertura específica.

Consentimiento, ejecutante, manejo de referencias

El nuevo pagador puede enviar una referencia temporal o local de un paciente `Consent.performer` (por ejemplo, la misma referencia utilizada en`Consent.patient`). HealthLake resuelve estas referencias automáticamente:

- Si `Consent.performer` contiene la misma referencia local que`Consent.patient`, la HealthLake reemplaza por la referencia real del paciente coincidente una vez que la comparación se realice correctamente.
- HealthLake admite referencias a artistas intérpretes o ejecutantes de los tipos Paciente RelatedPerson PractitionerRole, Médico y Organización (tanto referencias directas como referencias de identificador lógico).

- Si la validación del ejecutante falla (no se encuentra el recurso o no está asociado al paciente compatible), se coloca al miembro en ConsentConstrainedMembers lugar de mostrar un error.

Recursos del grupo de salida

El trabajo completado devuelve un recurso de parámetros que contiene tres recursos de grupo:

MatchedMembers Group (Grupo)

Contiene las referencias de los pacientes de todos los miembros seleccionados correctamente cuyo consentimiento está activo y válido en el momento de la solicitud. El recurso Consent se crea y almacena en el almacén de datos de cada miembro coincidente. Este grupo se crea en el almacén de datos y se puede usar directamente con. `$davinci-data-export`

NonMatchedMembers Group (Grupo)

Contiene referencias a miembros en los que no se encontró ninguna coincidencia única. Un miembro aparece aquí cuando ningún paciente del almacén de datos coincide con los datos demográficos proporcionados, no existe una cobertura válida para ningún paciente candidato compatible o varios pacientes coinciden con los datos demográficos y varios tienen una cobertura válida (ambiguo).

ConsentConstrainedMembers Group (Grupo)

Contiene las referencias de los pacientes a los que se pudo vincular correctamente (se confirmaron los datos demográficos y la cobertura), pero cuyo consentimiento no se pudo cumplir en el momento de la solicitud. El recurso Consent no se almacena para los miembros con restricciones de consentimiento. La identidad (MemberIdentifier y MemberId) del miembro coincidente se sigue incluyendo para que el pagador solicitante sepa quién tuvo la restricción.

El `Group.quantity` campo contiene el recuento total de miembros de cada uno de sus grupos respectivos.

Referencias de miembros del grupo:

- `Group.member.entity.reference`— Para MatchedMembers y ConsentConstrainedMembers, contiene la identificación del paciente del miembro coincidente en el sistema del pagador que responde. Para NonMatchedMembers, hace referencia a la entrada contenida, Paciente.
- `Group.member.entity.extension (base-ext-match-parameters)`— Contiene la identificación del paciente de la solicitud de entrada original (la identificación presentada por

el pagador solicitante, derivada de `Patient.idCoverage.beneficiary.reference`, o`Consent.patient.reference`).

Consent-Patient vinculación

> **Important**
>
> El recurso de consentimiento almacenado conserva la referencia del paciente exactamente como la envió el pagador solicitante. HealthLake no actualiza automáticamente el campo de paciente del consentimiento para que apunte al paciente coincidente en el almacén de datos receptor.

Para vincular un consentimiento almacenado con el paciente compatible, utilice el resultado del trabajo: cada miembro del MatchedMembers grupo tiene un botón que `member.entity.reference` apunta al paciente coincidente y un `member.entity.extension` (`base-ext-match-parameters`) que apunta al paciente de entrada contenido. Cross-reference todo ello con el campo de paciente de Consent para crear el mapeo en la capa de aplicación.

¿Qué se almacena frente a qué es transitorio

La siguiente tabla documenta lo que HealthLake permanece en el almacén de datos durante el `$bulk-member-match` procesamiento y lo que solo existe en la respuesta al trabajo:

Recurso	¿Almacenado?	¿Se puede consultar mediante REST?	Notas
MemberPatient (entrada)	No	No	Se usa solo para hacer coincidir; no persiste
CoverageToMatch (entrada)	No	No	Se usa solo para confirmar la cobertura
CoverageToLink (entrada)	No	No	No se valida con el almacén de datos; pertenece al nuevo pagador

Recurso	¿Almacenado?	¿Se puede consultar mediante REST?	Notas
Consentimiento (miembros coincidentes)	Sí	Sí, GET [base] / Consent/ {id}	Se guarda tal como se recibió del pagador solicitante
Consentimiento (miembros restringidos)	No	No	No almacenado. La identidad del miembro sigue incluida en la respuesta.
MatchedMembers Grupo (salida)	Sí	Sí, GET [base] / Group/ {id}	Instanciado; se puede usar con $davinci-data-export
NonMatchedMembers Grupo	No	No	Job response únicamente
ConsentConstrained Members Grupo	No	No	Job response únicamente

Integración con $davinci-data-export

El recurso del MatchedMembers grupo devuelto por se `$bulk-member-match` puede utilizar directamente con la `$davinci-data-export` operación para recuperar datos masivos de los miembros:

```
POST [base]/Group/{matched-group-id}/$davinci-data-export
GET [base]/Group/{matched-group-id}
```

Esta integración permite flujos de trabajo eficientes en los que primero se identifican los miembros coincidentes de forma masiva y, a continuación, se exportan sus historiales médicos completos utilizando el recurso del grupo resultante.

Uso de $member-remove antes de exportar

Si necesitas excluir a miembros específicos de la exportación después de la coincidencia (por ejemplo, si un miembro revoca el consentimiento entre la comparación y la exportación), usa `$member-remove` On the Group. MatchedMembers

> **Important**
>
> Al eliminar a un miembro mediante, se `$member-remove` marca como inactivo en el Grupo, pero `$davinci-data-export` solo se excluyen los miembros inactivos una vez que el Grupo pasa al estado «definitivo». Si llamas a `$davinci-data-export` un grupo que todavía tiene el estado predeterminado, es posible que los miembros eliminados sigan apareciendo en los resultados de exportación.

Flujo de trabajo:

1. `POST [base]/Group/{id}/$member-remove`— marcar los miembros como inactivos
2. `PUT [base]/Group/{id}`— actualizar el estado del grupo a «definitivo»
3. `POST [base]/Group/{id}/$davinci-data-export`— la exportación ahora excluye a los miembros eliminados

Características de rendimiento

La `$bulk-member-match` operación está diseñada para el procesamiento de grandes volúmenes y se ejecuta de forma asíncrona:

- Simultaneidad: máximo de 5 operaciones simultáneas por almacén de datos.
- Escalabilidad: gestiona hasta 500 miembros por solicitud (límite de carga útil de 5 MB).
- Operaciones paralelas: compatible con operaciones simultáneas de importación, exportación o eliminación masiva.

Autorización

La API utiliza el protocolo de autorización SMART on FHIR con los siguientes ámbitos obligatorios:

- `system/Patient.read`— Necesaria para buscar y comparar los recursos de los pacientes.

- `system/Coverage.read`— Necesario para validar la información de cobertura.
- `system/Group.write`— Necesario para crear los recursos del grupo de resultados.
- `system/Organization.read`— Condicional, obligatorio si la cobertura hace referencia a organizaciones.
- `system/Practitioner.read`— Condicional, obligatorio si la cobertura hace referencia a profesionales.
- `system/PractitionerRole.read`— Condicional, obligatorio si la cobertura hace referencia a las funciones de los profesionales.
- `system/Consent.write`— Condicional, obligatorio si se proporcionan recursos de consentimiento.

La operación también admite la autorización de la versión 4 (SiGv4) de AWS IAM Signature para el acceso programático.

Reglas de validación

En el paso 1, se aplican las siguientes reglas de validación MemberBundle a cada una de ellas. Los miembros que no pasan la validación se registran como errores y no aparecen en ningún grupo de salida.

MemberPatient

Campo	¿Cómo lo HealthLake usa	Fallo de validación si...
`name.family`	Búsqueda demográfica	Missing (Ausente)
`name.given`	Búsqueda demográfica	Falta (se requiere al menos uno)
`birthDate`	Búsqueda demográfica	Missing (Ausente)
`gender`	Búsqueda demográfica; si no está presente, se utiliza la extensión de nacimiento y sexo	No hay presencia de sexo ni sexo de nacimiento (hrex-pat-1)
`identifier`	Se incluye en la búsqueda cuando está presente; mejora la confianza	Nunca provoca fallos (opcional)

CoverageToMatch / CoverageToLink

Campo	¿Cómo lo HealthLake usa	Fallo de validación si...
`status`	Confirma que la cobertura es procesable	Missing (Ausente)
`beneficiary`	Vincula la cobertura a un paciente candidato	Missing (Ausente)
`payor`	Desambiguación cuando existen varios candidatos	Falta o hay más de un pagador
`relationship`	Confirma la relación entre el suscriptor y el beneficiario	Missing (Ausente)
`identifier (MB) or subscriberId`	Clave de desambiguación principal	Ninguno de los presentes

Consentimiento

Campo	¿Cómo lo HealthLake usa	Fallo de validación si...
`scope`	Confirma que el alcance del consentimiento es la privacidad del paciente	Falta el código de privacidad del paciente o no existe
`category`	Confirma la clasificación de la divulgación	Missing (Ausente)
`patient`	Identifica al sujeto del consentim iento	Missing (Ausente)
`performer`	Identifica quién está de acuerdo	Missing (Ausente)
`sourceReference`	Documenta la fuente del consentimiento	Missing (Ausente)

Campo	¿Cómo lo HealthLake usa	Fallo de validación si...
`policy.uri`	Determina el alcance del intercambio de datos	Falta o el URI no termina en #regular o #sensitive
`provision.type`	Debe ser «permiso» según el perfil de HRex Consent	Falta o no el «permiso» (incluido el «denegar»)
`provision.period`	Se evaluó en el paso 4 para comprobar si el consentimiento estaba restringido	Falta o no start/end
`status`	Se evaluó en el paso 4 (NO en el paso 1)	Nunca provoca un error en el paso 1: HealthLake acepta cualquier estado válido y lo evalúa en el paso 4

> **Note**
>
> El perfil HRex Consent define el estado con un valor fijo de «activo». HealthLake flexibiliza intencionalmente esta restricción para que un consentimiento inactivo reciba una clasificación significativa (ConsentConstrainedMembers) en lugar de un rechazo general por validación.

Comportamiento coincidente

- Búsqueda de pacientes (paso 2): HealthLake busca utilizando el signo `name.family` + `name.given` (exacto, no distingue mayúsculas de minúsculas), `birthDate` (exacto), `gender` (exacto; se utiliza el sexo de nacimiento si no existe el género) y `identifier` (si está presente, opcional).
- Desambiguación de la cobertura (paso 3): cuando se encuentran varios pacientes candidatos, `CoverageToMatch` se utiliza para seleccionar uno solo. Una cobertura es «válida» cuando existe un recurso de cobertura activo en el almacén de datos que coincide con `subscriberId` o `identifier` (tipo MB) y. `payor`
- Evaluación del consentimiento (paso 4): solo se realiza después de una coincidencia única exitosa. Consulte la sección anterior sobre los detalles de la evaluación del consentimiento.

Gestión de errores

La operación gestiona las siguientes condiciones de error:

- 400 Solicitud errónea: el formato de solicitud no es válido, faltan los parámetros necesarios o la carga útil supera los límites de tamaño (500 miembros o 5 MB).
- 422 Entidad inprocesable: errores de procesamiento durante la ejecución del trabajo.
- Errores de miembros individuales: cuando un miembro específico no se procesa, la operación continúa con los miembros restantes e incluye los detalles del error en el NonMatchedMembers grupo con los códigos de motivo correspondientes. Por ejemplo, si `MemberBundle` a un paciente le falta el `birthDate` parámetro, se devolverá el siguiente error:

```
"errors": [
  {
    "memberIndex": 1,
    "jsonBlob": {
      "resourceType": "OperationOutcome",
      "issue": [
        {
          "severity": "error",
          "code": "invalid",
          "diagnostics": "MemberPatient.birthDate is required"
        }
      ],
      "statusCode": 400
    }
  }
]
```

Los detalles del error están disponibles en el punto final del sondeo de estado e incluyen:

- `numberOfMembersWithCustomerError`: Recuento de miembros con errores de validación o introducción de datos.
- `numberOfMembersWithServerError`: Recuento de miembros con errores de procesamiento en el servidor.

Operaciones de relacionadas

- the section called “$member-match”— Operación de emparejamiento de miembros individuales.

- the section called “$davinci-data-export”— Exportación masiva de datos utilizando recursos del grupo.
- the section called “$ Operaciones”— Lista completa de operaciones compatibles.

Funcionamiento FHIR R4 **$confirm-attribution-list** para HealthLake

Indica a un productor que el consumidor no tiene más cambios que hacer en la lista de atribuciones. Esta operación finaliza la lista de atribuciones eliminando los miembros inactivos de la lista, cambiando el estado a «final» y confirmando la operación. Esta operación forma parte de la implementación de la IG 2.1.0 de la Lista de Atribución de Miembros (ATR) del FHIR.

Punto de conexión

```
POST [base]/Group/[id]/$confirm-attribution-list
```

Solicitud

No se requieren parámetros.

```
POST [base]/Group/[id]/$confirm-attribution-list
```

Respuesta

Devuelve HTTP 201 con un mensaje de confirmación.

Respuesta de ejemplo

```
HTTP Status Code: 201

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "message",
      "valueString": "Accepted."
    }
  ]
}
```

Estado del grupo tras la confirmación

Tras la confirmación correcta, el estado de la lista de atribuciones del recurso del grupo pasará a ser «final»:

```
{
  "resourceType": "Group",
  "id": "fullexample",
  "extension": [
    {
      "url": "http://hl7.org/fhir/us/davinci-atr/StructureDefinition/ext-
attributionListStatus",
      "valueCode": "final"
    }
  ]
  // ... other Group properties
}
```

Gestión de errores

La operación gestiona las siguientes condiciones de error:

Error	Estado HTTP	Description (Descripción)
Solicitud de operación no válida	400	Estructura o parámetros de solicitud no conformes
No se encontró el recurso de grupo	404	El ID de grupo especificado no existe

Autorización y seguridad

Requisitos de SMART Scope

- Los clientes deben tener los privilegios adecuados para leer los recursos del grupo y los recursos de atribución relacionados
- Pues`$confirm-attribution-list`, los clientes también deben tener privilegios de escritura para modificar los recursos del grupo
- Los mecanismos de autorización estándar del FHIR se aplican a todas las operaciones

Funcionamiento FHIR R4 **$davinci-data-export** para HealthLake

La `$davinci-data-export` operación es una operación FHIR asíncrona desde la que puede exportar datos de atención médica. AWS HealthLake Esta operación admite varios tipos de exportación, como la atribución de miembros (ATR), el acceso a PDex proveedores y el acceso a miembros. Payer-to-Payer APIs Es una versión especializada de la `$export` operación estándar del FHIR, diseñada para cumplir con los requisitos de las guías de DaVinci implementación.

Características principales de

- Procesamiento asíncrono: sigue el patrón estándar de solicitudes asíncronas del FHIR
- Exportación a nivel de grupo: exporta los datos de los miembros de un recurso grupal específico
- Múltiples tipos de exportación: admite ATR (atribución de miembros), acceso a PDex proveedores y Payer-to-Payer acceso a miembros APIs
- Soporte integral de perfiles: incluye US Core, CARIN Blue Button y PDex perfiles
- Filtrado flexible: admite el filtrado por pacientes, tipos de recursos y rangos de tiempo
- Salida NDJSON: proporciona datos en formato JSON delimitado por líneas nuevas

Punto final de la operación

```
GET [base]/Group/[id]/$davinci-data-export
POST [base]/Group/[id]/$davinci-data-export
```

Parámetros de la solicitud

Parámetro	Cardinalidad	Description (Descripción)
`patient`	0.. *	Miembros específicos cuyos datos se van a exportar. Si se omite, se exportan todos los miembros del grupo.
`_type`	0.1.	Lista delimitada por comas de los tipos de recursos del FHIR que se van a exportar. Si se omite, se incluyen todos los tipos de recursos compatibles con el tipo de exportación especific ado. Para las exportaciones ATR, el valor predeterminado es de 8 tipos de recursos de

Parámetro	Cardinalidad	Description (Descripción)
		atribución. En el PDex caso de las exportaciones, se incluyen todos los tipos de recursos de atribución, además de los tipos de recursos clínicos y de reclamaciones de US Core, CARIN Blue Button y los perfiles. PDex
`_since`	0.1	Incluya únicamente los recursos actualizados después de esta fecha y hora.
`_until`	0.1..	Incluya únicamente los recursos actualizados antes de esta fecha y hora.
`exportType`	0.1..	Tipo de exportación que se va a realizar. Valores válidos: `hl7.fhir.us.davinci-atr` , `hl7.fhir.us.davinci-pdex` , `hl7.fhir.us.davinci-pdex.p2p` , `hl7.fhir.us.davinci-pdex.member` . Predeterminado: `hl7.fhir.us.davinci-atr`.
`_includeEOB2xWoFinancial`	0.1..	Especifica si se deben incluir los ExplanationOfBenefit recursos de CARIN BB 2.x sin datos financieros. Predeterminado: `false`.
`_security`	0.. *	Filtra los recursos exportados por valores `meta.security` de codificación. Utilice el `system|code` formato (el carácter vertical debe estar codificado como `%7C` URL). Cuando se proporcionan varios valores, los recursos deben coincidir con todos ellos (y con la semántica). Úselo `system|` (barra vertical final, sin código) para hacer coincidir cualquier código de un sistema determinado.

Parámetro	Cardinalidad	Description (Descripción)
`_tag`	0.. *	Filtra los recursos exportados por valores `meta.tag` de codificación. Utiliza el mismo `system|code` formato y la misma semántica AND que`_security` . Cuando se `_tag` especifican ambos `_security` y, los recursos deben coincidir con ambos filtros.

Tipos de recursos admitidos

Los tipos de recursos admitidos dependen del tipo de exportación que especifique. Para las exportaciones ATR, se admiten los siguientes tipos de recursos:

- `Group`
- `Patient`
- `Coverage`
- `RelatedPerson`
- `Practitioner`
- `PractitionerRole`
- `Organization`
- `Location`

Para PDex las exportaciones (Provider Access y Member Access), se admiten todos los tipos de recursos clínicos y de reclamaciones, además de los tipos anteriores. Payer-to-Payer Para obtener una lista completa de los tipos de recursos compatibles, consulte la Guía de implementación básica de EE. UU. (STU 6.1), la Guía de implementación de CARIN Blue Button y la Guía de implementación del soporte de autorización previa de Da Vinci.

Tipos de exportación

La `$davinci-data-export` operación admite los siguientes tipos de exportación. El tipo de exportación se especifica mediante el `exportType` parámetro.

Tipo de exportación	Finalidad	Alcance de los datos	Límite temporal
`hl7.fhir.us.davinci-atr`	Lista de atribuciones de miembros	Recursos relacionados con la atribución	Ninguno
`hl7.fhir.us.davinci-pdex`	API de acceso para proveedores	Datos clínicos y de reclamaciones de los pacientes atribuidos	5 años
`hl7.fhir.us.davinci-pdex.p2p`	Payer-to-Payer Intercambio	Datos históricos de los miembros para las transiciones de seguros	5 años
`hl7.fhir.us.davinci-pdex.member`	API de acceso para miembros	Datos de salud propios del miembro	5 años

> **Note**
>
> En el caso de PDex las exportaciones, el límite temporal de 5 años no se aplica a los tipos de recursos ATR (`GroupPatientCoverage`,`RelatedPerson`,`Practitioner`,`PractitionerRole`,`Organization`, Estos recursos siempre se incluyen independientemente de la edad.

ATR (hl7.fhir.us.davinci-atr)

Con el tipo de exportación ATR, puedes exportar los datos de la lista de atribución de miembros. Usa este tipo de exportación para recuperar recursos relacionados con la atribución para los miembros de un grupo. Para obtener más información, consulte la operación de exportación de ATR de Da Vinci.

Tipos de recursos admitidos

`Group`, `Patient`, `Coverage`, `RelatedPerson`, `Practitioner`, `PractitionerRole`, `Organization`, `Location`

Filtrado temporal

No se aplica ningún filtrado temporal. Todos los recursos coincidentes se exportan independientemente de la fecha.

PDex Tipos de exportación

Todos los tipos de PDex exportación comparten los mismos perfiles compatibles y la misma lógica de filtrado. Para obtener más información, consulte la API de Da Vinci PDex Provider Access. Se admiten los siguientes perfiles:

- US Core 3.1.1, 6.1.0 y 7.0.0
- PDex Autorización previa (no se admite para el acceso de los miembros)
- Perfiles básicos de CARIN BB 2.x: pacientes hospitalizados, ambulatorios, institucionales, profesionales, orales, farmacéuticos NonClinician

En el PDex caso de las exportaciones, se descubren automáticamente los recursos clínicos y de reclamaciones de cada paciente del Grupo. No es necesario que haga referencia explícita a estos recursos en el recurso del Grupo. La operación busca todos los recursos del compartimento de pacientes (como`Observation`,, `Condition` `Coverage` `RelatedPerson`, `MedicationRequest`, y`ExplanationOfBenefit`) que pertenecen a los pacientes atribuidos. Solo `Patient` los tipos non-patient-compartment ATR (`Practitioner`,, `PractitionerRole` `Organization`,`Location`) requieren referencias explícitas en el Grupo. `Group`

Acceso al proveedor (`hl7.fhir.us.davinci-pdex`)

Permite a los proveedores de la red recuperar los datos de los pacientes atribuidos.

Payer-to-Payer (`hl7.fhir.us.davinci-pdex.p2p`)

Permite el intercambio de datos entre los pagadores cuando un paciente cambia de seguro.

Acceso de miembros () `hl7.fhir.us.davinci-pdex.member`

Permite a los miembros acceder a sus propios datos de salud. Este tipo de exportación puede incluir datos financieros en los recursos de reclamaciones.

Lógica de inclusión y soporte de perfiles

En el caso de PDex las exportaciones, la `$davinci-data-export` operación utiliza declaraciones de perfil en el `meta.profile` elemento para determinar qué recursos incluir en la exportación.

ExplanationOfBenefit Manejo de recursos

`ExplanationOfBenefit`Los recursos (EOB) se incluyen o excluyen de PDex las exportaciones en función de sus `meta.profile` declaraciones:

- ExplanationOfBenefit los recursos con un perfil CARIN BB 1.x se excluyen de la exportación.
- ExplanationOfBenefit los recursos sin ningún `meta.profile` conjunto se excluyen de la exportación.
- ExplanationOfBenefit siempre se incluyen los recursos con un perfil CARIN BB 2.x Basis.
- ExplanationOfBenefit los recursos con un perfil CARIN BB 2.x que contiene datos financieros se excluyen de forma predeterminada. Cuando `_includeEOB2xWoFinancial=true` están configurados, se incluyen sin datos financieros y el recurso se transforma en el perfil base correspondiente.
- ExplanationOfBenefit los recursos con un perfil de autorización PDex previa siempre se incluyen.

Transformación de datos financieros

Cuando se configura`_includeEOB2xWoFinancial=true`, la operación transforma los ExplanationOfBenefit recursos de CARIN BB 2.x en sus perfiles básicos correspondientes mediante la eliminación de los datos financieros. Por ejemplo, un `C4BB ExplanationOfBenefit Oral` recurso se transforma en`C4BB ExplanationOfBenefit Oral Basis`, lo que elimina los datos financieros del registro según la especificación de la FHIR.

Los siguientes elementos de datos financieros se eliminan durante la transformación:

- Todos los elementos se dividen `total`
- Todos los `adjudication` elementos con rebanada `amounttype`
- Todos los `item.adjudication` elementos con información sobre el importe

La operación también actualiza los metadatos del perfil durante la transformación:

- `meta.profile`se actualiza a la URL canónica del perfil básico
- La versión se ha actualizado a la versión básica CARIN BB 2.x

- Los recursos existentes en el almacén de datos no se modifican
- Los recursos exportados no se devuelven al almacén de datos

Reglas de detección de perfiles

La operación utiliza las siguientes reglas para detectar y validar los perfiles:

- La detección de versiones se basa en la versión `meta.profile` canónica URLs
- Se incluye un recurso si ALGUNO de sus perfiles declarados coincide con los criterios de exportación
- La validación del perfil se produce durante el procesamiento de la exportación

Filtrado temporal de cinco años para PDex las exportaciones

Para todos los tipos de PDex exportación, HealthLake aplica un filtro temporal de 5 años en función de cuándo se actualizó el recurso por última vez. El filtro temporal se aplica a todos los recursos, excepto a los siguientes tipos de recursos de atribución principales, que siempre se exportan independientemente de su antigüedad:

- `Patient`
- `Coverage`
- `Organization`
- `Practitioner`
- `PractitionerRole`
- `RelatedPerson`
- `Location`
- `Group`

Estos recursos administrativos y demográficos están exentos porque proporcionan un contexto esencial para los datos exportados. Las exportaciones de ATR no están sujetas a ningún filtrado temporal.

Solicitudes de ejemplo

Los siguientes ejemplos muestran cómo iniciar trabajos de exportación para diferentes tipos de exportación.

Exportación ATR

```
GET https://healthlake.{region}.amazonaws.com/datastore/
{datastoreId}/r4/Group/example-group/$davinci-data-export?
_type=Group,Patient,Coverage,Practitioner,Organization&exportType=hl7.fhir.us.davinci-
atr

POST https://healthlake.{region}.amazonaws.com/datastore/
{datastoreId}/r4/Group/example-group/$davinci-data-export?
_type=Group,Patient,Coverage,Practitioner,Organization&exportType=hl7.fhir.us.davinci-
atr
Content-Type: application/json

{
  "DataAccessRoleArn": "arn:aws:iam::444455556666:role/your-healthlake-service-role",
  "JobName": "attribution-export-job",
  "OutputDataConfig": {
    "S3Configuration": {
      "S3Uri": "s3://your-export-bucket/EXPORT-JOB",
      "KmsKeyId":
 "arn:aws:kms:region:444455556666:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    }
  }
}
```

Exportación de Provider Access con eliminación de datos ExplanationOfBenefit financieros

```
GET https://healthlake.{region}.amazonaws.com/datastore/
{datastoreId}/r4/Group/example-group/$davinci-data-export?
_type=Patient,Observation,Condition,MedicationRequest,ExplanationOfBenefit&exportType=hl7.fhir.
pdex&_includeEOB2xWoFinancial=true
```

Payer-to-Payer exportar

```
GET https://healthlake.{region}.amazonaws.com/datastore/
{datastoreId}/r4/Group/example-group/$davinci-data-export?
_type=Patient,Coverage,ExplanationOfBenefit,Condition,Procedure&exportType=hl7.fhir.us.davinci-
pdex.p2p&_includeEOB2xWoFinancial=true
```

Exportación de Member Access para un paciente específico

```
GET https://healthlake.{region}.amazonaws.com/datastore/
{datastoreId}/r4/Group/example-group/$davinci-data-export?
_type=Patient,Observation,Condition,ExplanationOfBenefit,MedicationRequest&exportType=hl7.fhir.
pdex.member&patient=Patient/example-patient-id
```

Respuesta de ejemplo

```
{
  "datastoreId": "eaee622d8406b41eb86c0f4741201ff9",
  "jobStatus": "SUBMITTED",
  "jobId": "48d7b91dae4a64d00d54b70862f33f61"
}
```

Relaciones de recursos

La operación exporta los recursos en función de sus relaciones en la lista de atribuciones de miembros:

```
Group (Attribution List)
### Patient (Members)
### Coverage # RelatedPerson (Subscribers)
### Practitioner (Attributed Providers)
### PractitionerRole # Location
### Organization (Attributed Providers)
```

> **Note**
>
> El diagrama de relaciones de recursos anterior se aplica a las exportaciones de ATR. En el PDex caso de las exportaciones, los recursos clínicos y de reclamaciones se descubren automáticamente mediante la búsqueda de pacientes y no es necesario incluir referencias explícitas en el recurso del Grupo.

Fuentes de recursos

Recurso	Ubicación de la fuente	Description (Descripción)
`Patient`	`Group.member.entity`	Los pacientes que son miembros de la lista de atribuciones

Recurso	Ubicación de la fuente	Description (Descripción)
`Coverage`	`Group.member.extension:coverageReference`	Cobertura que dio lugar a la afiliación de los pacientes
`Organization`	`Group.member.extension:attributedProvider`	Organizaciones a las que se atribuyen los pacientes
`Practitioner`	`Group.member.extension:attributedProvider`	Profesionales individuales a los que se atribuyen los pacientes
`PractitionerRole`	`Group.member.extension:attributedProvider`	Funciones del médico a las que se atribuyen los pacientes
`RelatedPerson`	`Coverage.subscriber`	Suscriptores de la cobertura
`Location`	`PractitionerRole.location`	Ubicaciones asociadas a las funciones de los profesionales
`Group`	Punto final de entrada	La propia lista de atribuciones

Gestión de trabajos

Comprobar el estado del trabajo

```
GET [base]/export/[job-id]
```

Cancelar trabajo

```
DELETE [base]/export/[job-id]
```

Ciclo de vida del trabajo

- `SUBMITTED`- Se ha recibido el trabajo y se ha puesto en cola
- `IN_PROGRESS`- Job se está procesando activamente
- `COMPLETED`- Job finalizado correctamente, archivos disponibles para descargar
- `FAILED`- El trabajo encontró un error

Output Format (Formato de salida)

- Formato de archivo: NDJSON (JSON delimitado por nueva línea)
- Organización de archivos: archivos separados para cada tipo de recurso
- Extensión de archivo: .ndjson
- Ubicación: depósito y ruta de S3 especificados

Gestión de errores

La operación devuelve una solicitud incorrecta de HTTP 400 con un valor OperationOutcome para las siguientes condiciones:

Errores de autorización

La función de IAM especificada en `DataAccessRoleArn` no tiene permisos suficientes para realizar la operación de exportación. Para ver la lista completa de los permisos de S3 y KMS necesarios, consulte Configurar los permisos para los trabajos de exportación.

Errores de validación de parámetros

- El `patient` parámetro no tiene el formato `Patient/id,Patient/id,...`
- Una o más referencias de pacientes no son válidas o no pertenecen al grupo especificado
- El valor del `exportType` parámetro no es un tipo de exportación compatible
- El `_type` parámetro contiene tipos de recursos que no son compatibles con el tipo de exportación especificado
- En el `_type` parámetro faltan los tipos de recursos necesarios (`Group`,`Patient`,`Coverage`) para el tipo de `hl7.fhir.us.davinci-atr` exportación
- El valor del `_includeEOB2xWoFinancial` parámetro no es un booleano válido

Errores de validación de recursos

- El recurso de grupo especificado no existe en el banco de datos
- El recurso de grupo especificado no tiene miembros
- Uno o más miembros del grupo hacen referencia a recursos para pacientes que no existen en el almacén de datos

Seguridad y autorización

- Se aplican los mecanismos de autorización estándar del FHIR

- La función de acceso a los datos debe tener los permisos de IAM necesarios para las operaciones de S3 y KMS. Para ver la lista completa de los permisos necesarios, consulte Configurar los permisos para los trabajos de exportación.

Prácticas recomendadas

- Selección del tipo de recurso: solicite únicamente los tipos de recursos que necesite para minimizar el tamaño de la exportación y el tiempo de procesamiento
- Filtrado basado en el tiempo: utilice el `_since` parámetro para exportaciones incrementales
- Filtrado de pacientes: utilice el `patient` parámetro cuando solo necesite datos de miembros específicos
- Supervisión del trabajo: compruebe periódicamente el estado del trabajo para grandes exportaciones
- Gestión de errores: Implemente la lógica de reintento adecuada para los trabajos fallidos
- Conocimiento del filtro temporal: para PDex las exportaciones, tenga en cuenta el filtro temporal de 5 años al seleccionar los tipos de recursos
- Eliminación de datos financieros: `_includeEOB2xWoFinancial=true` utilícelo cuando necesite datos de reclamaciones sin información financiera
- Gestión de perfiles: asegúrese de que los recursos cuenten con las declaraciones de perfil adecuadas, compárelos con los perfiles de destino antes de su ingestión y utilice el control de versiones de los perfiles para controlar el comportamiento de exportación

Limitaciones

- Se puede especificar un máximo de 500 pacientes en el parámetro `patient`
- La exportación se limita únicamente a las operaciones a nivel de grupo
- Solo admite el conjunto predefinido de tipos de recursos para cada tipo de exportación
- La salida siempre está en formato NDJSON
- PDex las exportaciones están limitadas a 5 años de datos clínicos y de reclamaciones
- La transformación de datos financieros solo se aplica a los perfiles CARIN BB 2.x ExplanationOfBenefit

Recursos adicionales

- Lista de atribuciones de miembros de Da Vinci (IG)
- Intercambio de datos de jugadores Da Vinci IG
- Sistema de intercambio de datos de pagadores dirigido al consumidor de CARIN
- Guía de implementación básica de EE. UU.
- Especificación de acceso masivo a datos del FHIR

Generación de documentos clínicos con **$document**

AWS HealthLake ahora es compatible con el `$document` funcionamiento de los recursos de Composition, lo que le permite generar un documento clínico completo agrupando la composición con todos sus recursos de referencia en un único paquete cohesivo. Esta operación es esencial para las aplicaciones de atención médica que necesitan:

- Cree documentos clínicos estandarizados
- Intercambie los registros completos de los pacientes
- Almacene documentación clínica completa
- Genere informes que incluyan todo el contexto relevante

De uso

La `$document` operación se puede invocar en los recursos de Composition mediante los métodos GET y POST:

Operaciones admitidas

```
GET/POST [base]/Composition/[id]/$document
```

Parámetros admitidos

HealthLake admite el siguiente `$document` parámetro FHIR:

Parámetro	Tipo	Obligatorio/a	Predeterminado	Description (Descripción)
`persist`	booleano	No	false	Booleano que indica si el servidor debe almacenar el paquete de documentos generado

Ejemplos

Solicitud GET

```
GET [base]/Composition/180f219f-97a8-486d-99d9-ed631fe4fc57/$document?persist=true
```

Solicitud POST con parámetros

```
POST [base]/Composition/180f219f-97a8-486d-99d9-ed631fe4fc57/$document
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "persist",
      "valueBoolean": true
    }
  ]
}
```

Respuesta de ejemplo

La operación devuelve un recurso de paquete del tipo «documento» que contiene la composición y todos los recursos a los que se hace referencia:

```
{
  "resourceType": "Bundle",
  "id": "180f219f-97a8-486d-99d9-ed631fe4fc57",
  "type": "document",
  "identifier": {
    "system": "urn:ietf:rfc:3986",
```

```
    "value": "urn:uuid:0c3151bd-1cbf-4d64-b04d-cd9187a4c6e0"
  },
  "timestamp": "2024-06-21T15:30:00Z",
  "entry": [
    {
      "fullUrl": "http://example.org/fhir/Composition/180f219f-97a8-486d-99d9-
ed631fe4fc57",
      "resource": {
        "resourceType": "Composition",
        "id": "180f219f-97a8-486d-99d9-ed631fe4fc57",
        "status": "final",
        "type": {
          "coding": [
            {
              "system": "http://loinc.org",
              "code": "34133-9",
              "display": "Summary of Episode Note"
            }
          ]
        },
        "subject": {
          "reference": "Patient/example"
        },
        "section": [
          {
            "title": "Allergies",
            "entry": [
              {
                "reference": "AllergyIntolerance/123"
              }
            ]
          }
        ]
      }
    },
    {
      "fullUrl": "http://example.org/fhir/Patient/example",
      "resource": {
        "resourceType": "Patient",
        "id": "example",
        "name": [
          {
            "family": "Smith",
            "given": ["John"]
```

```
          }
        ]
      }
    },
    {
      "fullUrl": "http://example.org/fhir/AllergyIntolerance/123",
      "resource": {
        "resourceType": "AllergyIntolerance",
        "id": "123",
        "patient": {
          "reference": "Patient/example"
        },
        "code": {
          "coding": [
            {
              "system": "http://snomed.info/sct",
              "code": "418689008",
              "display": "Allergy to penicillin"
            }
          ]
        }
      }
    }
  ]
}
```

Comportamiento

La `$document` operación:

1. Toma el recurso de composición especificado como base del documento
2. Identifica y recupera todos los recursos a los que hace referencia directamente la composición
3. Empaqueta la composición y todos los recursos a los que se hace referencia en un paquete del tipo «documento»
4. Almacena el paquete de documentos generado en el almacén de datos cuando el parámetro de persistencia se establece en true
5. Identifica y recupera los recursos a los que hace referencia indirectamente la composición para una generación completa de documentos

Actualmente, la `$document` operación admite la recuperación de referencias a recursos en el siguiente formato:

1.
```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Resource/id
```
2. Recurso/ID

Las referencias a recursos no admitidas en el recurso de composición se eliminarán del documento generado.

Gestión de errores

La operación gestiona las siguientes condiciones de error:

- 400 Solicitud errónea: `$document` operación no válida (solicitud no conforme) o si el documento resultante no pasa la validación del FHIR debido a que se filtraron las referencias cuando persisten está establecido en true
- 404 No encontrado: no se ha encontrado el recurso de composición

Para obtener más información sobre la especificación de la `$document` operación, consulte la documentación de composición `$document` del FHIR R4.

Eliminar recursos de forma permanente con **`$erase`**

AWS HealthLake admite la `$erase` operación, lo que permite la eliminación permanente de un recurso específico y sus versiones históricas. Esta operación resulta especialmente útil cuando se necesita:

- Eliminar permanentemente los recursos individuales
- Eliminar historiales de versiones específicos
- Gestione los ciclos de vida de los recursos individuales
- Cumpla con los requisitos específicos de eliminación de datos

De uso

La `$erase` operación se puede invocar en dos niveles:

Nivel de instancia de recurso

```
POST [base]/[ResourceType]/[ID]/$erase?deleteAuditEvent=true
```

Nivel específico de la versión

```
POST [base]/[ResourceType]/[ID]/_history/[VersionID]/$erase
```

Parameters

Parámetro	Tipo	Obligatorio/a	Predeterminado	Description (Descripción)
`deleteAuditEvent`	booleano	No	false	Si es verdadero, elimina los eventos de auditoría asociados

Ejemplos

Solicitud de ejemplo

```
POST [base]/Patient/example-patient/$erase
```

Respuesta de ejemplo

```
{
      "jobId": "5df47e2f51ff3c731847678cb8cad48e",
      "jobStatus": "SUBMITTED"
    }
```

Estado del trabajo

Para comprobar el estado de un trabajo de borrado:

```
GET [base]/$erase/[jobId]
```

La operación devuelve información sobre el estado del trabajo:

```
{
```

```
    "datastoreId": "36622996b1fcecb7e12ee2ee085308d3",
    "jobId": "5df47e2f51ff3c731847678cb8cad48e",
    "status": "COMPLETED",
    "submittedTime": "2025-10-30T16:39:24.160Z"
}
```

Comportamiento

La `$erase` operación:

1. Procesa de forma asíncrona para garantizar la integridad de los datos
2. Mantiene las transacciones ACID
3. Proporciona un seguimiento del estado del trabajo
4. Elimina permanentemente el recurso especificado y sus versiones
5. Incluye un registro de auditoría completo de las actividades de eliminación
6. Soporta la eliminación selectiva de eventos de auditoría

Registro de auditoría

La `$erase` operación se registra DeleteResource con el ID de usuario, la marca de tiempo y los detalles del recurso.

Limitaciones

- `$erase`del recurso no aparecerá en los resultados de búsqueda ni `_history` en las consultas.
- Es posible que los recursos que se estén borrando estén inaccesibles temporalmente durante el procesamiento
- La medición del almacenamiento se ajusta inmediatamente a medida que los recursos se eliminan permanentemente

Obtener datos de pacientes con **`Patient/$everything`**

La `Patient/$everything` operación se utiliza para consultar un `Patient` recurso del FHIR, junto con cualquier otro recurso relacionado con él. `Patient` La operación se puede utilizar para proporcionar a un paciente acceso a su historial completo o para que un proveedor realice una descarga masiva de datos relacionados con un paciente. HealthLakeapoyos `Patient/$everything` para un paciente específico`id`.

`Patient/$everything`es una operación de la API REST del FHIR que se puede invocar como se muestra en los ejemplos siguientes.

GET request

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything
```

> **Note**
>
> Los recursos en respuesta se ordenan por tipo de recurso y recursoid.
> La respuesta siempre se rellena con`Bundle.total`.

Parámetros **`Patient/$everything`**

HealthLake admite los siguientes parámetros de consulta

Parámetro	Details
iniciar	Obtenga todos los `Patient` datos después de una fecha de inicio especificada.
finales	Obtenga todos los `Patient` datos antes de una fecha de finalización especificada.
since	Actualice todos `Patient` los datos después de una fecha específica.
_tipo	Obtenga `Patient` datos para tipos de recursos específicos.
_count	Obtenga `Patient` datos y especifique el tamaño de la página.

Example- Obtenga todos los datos del paciente después de una fecha de inicio específica

`Patient/$everything`puede usar el `start` filtro para consultar solo los datos después de una fecha específica.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?start=2024-03-15T00:00:00.000Z
```

Example- Obtenga todos los Patient datos antes de una fecha de finalización especificada

El paciente $everything puede usar el `end` filtro solo para consultar datos anteriores a una fecha específica.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?end=2024-03-15T00:00:00.000Z
```

Example- Actualice todos los Patient datos después de una fecha específica

`Patient/$everything`puede usar el `since` filtro para consultar solo los datos actualizados después de una fecha específica.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?since=2024-03-15T00:00:00.000Z
```

Example- Obtenga Patient datos para tipos de recursos específicos

El paciente $everything puede usar el `_type` filtro para especificar tipos de recursos específicos que se incluirán en la respuesta. Se pueden especificar varios tipos de recursos en una lista separada por comas.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?_type=Observation,Condition
```

Example- Obtenga Patient datos y especifique el tamaño de la página

Patient $everything puede usar el `_count` para configurar el tamaño de la página.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?_count=15
```

`Patient/$everything` **`start`** y atributos **`end`**

HealthLake admite los siguientes atributos de recurso para los parámetros de `end` consulta `Patient/ $everything` `start` y.

Recurso	Elemento de recurso
Cuenta	Account.ServicePeriod.start

Recurso	Elemento de recurso
AdverseEvent	AdverseEvent.fecha
AllergyIntolerance	AllergyIntolerance. Fecha de registro
Cita	Cita. Inicio
AppointmentResponse	AppointmentResponse.iniciar
AuditEvent	AuditEvent.periodo.inicio
Basic	Básico. Creado
BodyStructure	NO_DATE
CarePlan	CarePlan.periodo.inicio
CareTeam	CareTeam.periodo.inicio
ChargeItem	ChargeItem. occurrenceDateTime, ChargeItem .OccurrencePeriod.Start, .OccurrenceTiming.Event ChargeItem
Reclamación	Reclamación. Periodo facturable. Start
ClaimResponse	ClaimResponse.creado
ClinicalImpression	ClinicalImpression.fecha
Comunicación	Comunicación. Enviada

Recurso	Elemento de recurso
CommunicationRequest	CommunicationRequest. occurrenceDateTime,. Periodo de aparición CommunicationRequest. Inicio
Composición	Composición.Fecha
Condición	Condición. Fecha de registro
Consentimiento	Consentimiento. Fecha y hora
Cobertura	Cobertura. Periodo. Inicio
CoverageEligibilityRequest	CoverageEligibilityRequest.creado
CoverageEligibilityResponse	CoverageEligibilityResponse.creado
DetectedIssue	DetectedIssue.identificado
DeviceRequest	DeviceRequest.autor en
DeviceUseStatement	DeviceUseStatement. Grabado en
DiagnosticReport	DiagnosticReport... efectivo
DocumentManifest	DocumentManifest.creado

Recurso	Elemento de recurso
DocumentReference	DocumentReference.context.period.start
Encuentro	Encuentro, punto, inicio
EnrollmentRequest	EnrollmentRequest.creado
EpisodeOfCare	EpisodeOfCare.periodo.inicio
ExplanationOfBenefit	ExplanationOfBenefit.Periodo facturable. Inicio
FamilyMemberHistory	SIN FECHA
Indicador	Bander.Period.Start
Objetivo	Objetivo. Fecha de estado
Group	SIN FECHA
ImagingStudy	ImagingStudy.iniciado
Inmunización	Inmunización. Registrada
ImmunizationEvaluation	ImmunizationEvaluation.fecha
ImmunizationRecommendation	ImmunizationRecommendation.fecha

Recurso	Elemento de recurso
Factura	Factura.fecha
Enumeración	Fecha de lista
MeasureReport	MeasureReport.periodo.inicio
Multimedia	Medios de comunicación. Emitido
MedicationAdministration	MedicationAdministration... efectivo
MedicationDispense	MedicationDispense. Cuando esté preparado
MedicationRequest	MedicationRequest. escrito en
MedicationStatement	MedicationStatement.fecha afirmada
Molecular Sequence	SIN FECHA
Nutrition Order	NutritionOrder.Fecha y hora
Observación	Observación. Efectiva
Paciente	NO_DATE
Persona	SIN FECHA
Procedimiento	Procedimiento. Realizado

Recurso	Elemento de recurso
Procedencia	Procedencia. Periodo ocurrido. Inicio, procedencia. occurredDateTime
QuestionnaireResponse	QuestionnaireResponse.creado
RelatedPerson	NO_DATE
RequestGroup	RequestGroup.autor el
ResearchSubject	ResearchSubject.. punto
RiskAssessment	RiskAssessment. occurrenceDateTime,. Periodo de aparición RiskAssessment. Inicio
Schedule	Schedule.Planning Horizon
ServiceRequest	ServiceRequest. Autor en
Ejemplar	Espécimen. Hora de recepción
SupplyDelivery	SupplyDelivery. occurrenceDateTime, SupplyDelivery .OccurrencePeriod. Start, .OccurrenceTiming.Event SupplyDelivery
SupplyRequest	SupplyRequest. Escrito el
VisionPrescription	VisionPrescription.Fecha de escritura

Recuperación de ValueSet códigos con **$expand**

AWS HealthLake ahora es compatible con la `$expand` operación ValueSets que usted haya introducido como cliente, lo que le permite recuperar la lista completa de códigos contenidos en esos ValueSet recursos. Esta operación resulta especialmente útil cuando se necesita:

- Recupere todos los códigos posibles con fines de validación
- Muestra las opciones disponibles en las interfaces de usuario
- Realice búsquedas de código exhaustivas dentro de un contexto terminológico específico

De uso

La `$expand` operación se puede invocar en ValueSet los recursos mediante los métodos GET y POST:

Operaciones admitidas

```
GET/POST [base]/ValueSet/[id]/$expand
GET [base]/ValueSet/$expand?url=http://example.com
POST [base]/ValueSet/$expand
```

Parámetros admitidos

HealthLake admite un subconjunto de parámetros del FHIR `$expand` R4:

Parámetro	Tipo	Obligatorio	Description (Descripción)
`url`	uri	No	URL canónica del que se va a expandir ValueSet
`id`	id	No	ValueSet identificador del recurso que se va a expandir (para operaciones GET o POST)
`filter`	cadena	No	Filtra el resultado de la expansión del código
`count`	entero	No	Número de códigos a devolver
`offset`	entero	No	Número de códigos coincidentes que se deben omitir antes de la devolución. Se aplica

Parámetro	Tipo	Obligatorio	Description (Descripción)
			después del filtrado y solo a los códigos coincidentes, no a todo el contenido del original sin filtrar ValueSet

Ejemplos

Solicitud GET por ID

```
GET [base]/ValueSet/example-valueset/$expand
```

OBTENGA la solicitud por URL con filtro

```
GET [base]/ValueSet/$expand?url=http://example.com/ValueSet/my-
valueset&filter=male&count=5
```

Solicitud POST con parámetros (por ID)

```
POST [base]/ValueSet/example-valueset/$expand
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "count",
      "valueInteger": 10
    },
    {
      "name": "filter",
      "valueString": "admin"
    }
  ]
}
```

Solicitud POST con parámetros (por URL)

```
POST [base]/ValueSet/$expand
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "url",
      "valueUri": "http://hl7.org/fhir/ValueSet/administrative-gender"
    },
    {
      "name": "count",
      "valueInteger": 10
    }
  ]
}
```

Respuesta de ejemplo

La operación devuelve un ValueSet recurso con un `expansion` elemento que contiene los códigos expandidos:

```
{
  "resourceType": "ValueSet",
  "id": "administrative-gender",
  "status": "active",
  "expansion": {
    "identifier": "urn:uuid:12345678-1234-1234-1234-123456789abc",
    "timestamp": "2024-01-15T10:30:00Z",
    "total": 4,
    "parameter": [
      {
        "name": "count",
        "valueInteger": 10
      }
    ],
    "contains": [
      {
        "system": "http://hl7.org/fhir/administrative-gender",
        "code": "male",
        "display": "Male"
      },
```

```
      {
        "system": "http://hl7.org/fhir/administrative-gender",
        "code": "female",
        "display": "Female"
      },
      {
        "system": "http://hl7.org/fhir/administrative-gender",
        "code": "other",
        "display": "Other"
      },
      {
        "system": "http://hl7.org/fhir/administrative-gender",
        "code": "unknown",
        "display": "Unknown"
      }
    ]
  }
}
```

La respuesta incluye:

- expansion.total: número total de códigos en el expandido ValueSet
- expansion.contains: matriz de códigos expandidos con su sistema, código y valores de visualización
- expansion.parameter: parámetros utilizados en la solicitud de expansión

Para obtener más información sobre la especificación de `$expand` operación, consulte la documentación del FHIR R4. ValueSet `$expand`

Exportación HealthLake de datos con FHIR **`$export`**

Puede exportar datos de forma masiva desde su almacén de HealthLake datos mediante la operación $export de FHIR. HealthLake admite el `$export` uso `POST` y las solicitudes del FHIR. `GET` Para realizar una solicitud de exportación`POST`, debe tener un usuario, grupo o rol de IAM con los permisos necesarios, especificarlo `$export` como parte de la solicitud e incluir los parámetros deseados en el cuerpo de la solicitud.

> **Note**
>
> Todas las solicitudes de HealthLake exportación realizadas mediante FHIR `$export` se devuelven en `ndjson` formato y se exportan a un bucket de Amazon S3, donde cada objeto de Amazon S3 contiene solo un tipo de recurso FHIR.
> Puede poner en cola las solicitudes de exportación según las cuotas de servicio de la AWS cuenta. Para obtener más información, consulte Cuotas de servicio.

HealthLake admite los siguientes tres tipos de solicitudes de punto final de exportación masiva.

HealthLake **`$export`**tipos masivos

Tipo de exportación	Description (Descripción)	Sintaxis
Sistema	Exporte todos los datos del servidor HealthLake FHIR.	`POST https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/$export`

Todos los pacientes	Exporte todos los datos relacionados con todos los pacientes, incluidos los tipos de recursos asociados al tipo de recurso del paciente.	`POST https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/Patient/$export`
		
		`GET https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/Patient/$export`

Grupo de pacientes	Exporte todos los datos relacionados con un grupo de pacientes especificado con un identificador de grupo.	`POST https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/Group/ id/$export`
		
		`GET https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/Group/ id/$export`

Antes de empezar

Cumpla los siguientes requisitos para realizar una solicitud de exportación mediante la API REST del FHIR para HealthLake.

- Debe haber configurado un usuario, grupo o rol que tenga los permisos necesarios para realizar la solicitud de exportación. Para obtener más información, consulte Autorizar una solicitud `$export`.
- Debe haber creado un rol de servicio que conceda HealthLake acceso al bucket de Amazon S3 al que quiere que se exporten sus datos. El rol de servicio también debe especificarse HealthLake como principal de servicio. Para obtener más información sobre la configuración de permisos, consulteConfigurar los permisos para los trabajos de exportación.

Autorizar una solicitud **`$export`**

Para realizar correctamente una solicitud de exportación mediante la API REST de FHIR, autorice a su usuario, grupo o rol mediante IAM o .0. OAuth2 También debe tener un rol de servicio.

Autorizar una solicitud mediante IAM

Al realizar una `$export` solicitud, el usuario, el grupo o el rol deben incluir las acciones de IAM en la política. Para obtener más información, consulte Configurar los permisos para los trabajos de exportación.

Autorizar una solicitud mediante SMART en el FHIR (2.0) OAuth

Al realizar una `$export` solicitud en un almacén de HealthLake datos compatible con SMART on FHIR, debe tener asignados los ámbitos adecuados. Para obtener más información, consulte Los alcances de los recursos de SMART on FHIR son: HealthLake.

> **Note**
>
> El FHIR `$export` y `GET` las solicitudes requieren el mismo método de autenticación o el mismo token de soporte (en el caso de SMART en el FHIR) para solicitar la exportación y la recuperación de archivos. Los archivos exportados mediante el FHIR `$export` con `GET` están disponibles para su descarga durante 48 horas.

¿Hacer una solicitud **$export**

En esta sección se describen los pasos necesarios que debe seguir al realizar una solicitud de exportación mediante la API REST del FHIR.

Para evitar cargos accidentales en tu AWS cuenta, te recomendamos probar tus solicitudes realizando una `POST` solicitud sin proporcionar la `$export` sintaxis.

Para realizar la solicitud, debes hacer lo siguiente:

1. Especifique `$export` en la URL de la `POST` solicitud un punto final compatible.
2. Especifique los parámetros de cabecera necesarios.
3. Especifique el cuerpo de la solicitud que defina los parámetros necesarios.

Paso 1: especifique **$export** en la URL de la **POST** solicitud un punto final compatible.

HealthLake admite tres tipos de solicitudes de punto final de exportación masiva. Para realizar una solicitud de exportación masiva, debe realizar una solicitud `POST` basada en uno de los tres puntos de enlace compatibles. Los siguientes ejemplos muestran dónde especificar `$export` en la URL de la solicitud.

- `POST https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/$export`
- `POST https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/$export`
- `POST https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Group/id/$export`

Puede utilizar los siguientes parámetros de búsqueda admitidos en la cadena de `POST` solicitud.

Parámetros de búsqueda compatibles

HealthLake admite los siguientes modificadores de búsqueda en las solicitudes de exportación masiva.

Los siguientes ejemplos incluyen caracteres especiales que deben codificarse antes de enviar la solicitud.

Name	¿Obligatorio?	Description (Descripción)	Ejemplo
`_outputFormat`	No	El formato para generar los archivos de datos masivos solicitados. Los valores aceptados son `application/fhir+ndjson` , `application/ndjson` , `ndjson`.	
`_type`	No	Cadena de tipos de recursos FHIR delimitados por comas que desea incluir en su trabajo de exportación. Recomendamos incluirlo `_type` porque esto puede tener repercusiones en los costes cuando se exportan todos los recursos.	`&_type=MedicationStatement,Observation`
`_since`	No	Tipos de recursos modificados en o después de la marca de fecha y hora. Si un tipo de recurso no tiene una hora de última actualiza	`&_since=2024-05-09T00%3A00%3A00Z`

Name	¿Obligatorio?	Description (Descripción)	Ejemplo
		ción, se incluirá en la respuesta.	
`_until`	No	Tipos de recursos modificados en o antes de la fecha y hora. Se utiliza en combinación con `_since` para definir un intervalo de tiempo específico para la exportación. Si un tipo de recurso no tiene una hora de última actualización, se excluirá de tu respuesta.	`&_until=2024-12-31T23%3A59%3A59Z`

Name	¿Obligatorio?	Description (Descripción)	Ejemplo
`_security`	No	Filtra los recursos exportados por valores `meta.security` de codificación. Usa el `system|code` formato. Cuando se proporcionan varios valores, los recursos deben coincidir con todos ellos (y con la semántica). Úselo `system|` (barra vertical final, sin código) para hacer coincidir cualquier código de un sistema determinado.	`&_security=https://myorg.com/tenant%7Cclinic-A`
`_tag`	No	Filtre los recursos exportados por valores `meta.tag` de codificación. Utiliza el mismo `system|code` formato y la misma semántica AND que `_security`. Cuando se `_tag` especifican ambos `_security` y, los recursos deben coincidir con ambos filtros.	`&_tag=https://myorg.com/dept%7Ccardiology`

Paso 2: especifique los parámetros de cabecera necesarios

Para realizar una solicitud de exportación mediante la API REST del FHIR, debe especificar los siguientes parámetros de encabezado.

- Content-Type: `application/fhir+json`
- Prefiero: `respond-async`

A continuación, debe especificar los elementos necesarios en el cuerpo de la solicitud.

Paso 3: especifique un cuerpo de solicitud que defina los parámetros necesarios.

La solicitud de exportación también requiere un cuerpo en `JSON` formato. El cuerpo puede incluir los siguientes parámetros.

Key	¿Obligatorio?	Description (Descripción)	Valor
DataAccessRoleArn	Sí	Un ARN de un rol de HealthLake servicio. El rol de servicio utilizado debe especificarse HealthLake como principal de servicio.	`arn:aws:iam:: 444455556666 :role/your-healthlake-service-role`
JobName	No	El nombre de la solicitud de exportación.	**`your-export-job-name`**
S3Uri	Sí	Parte de una OutputDataConfig clave. El URI de S3 del depósito de destino donde se descargarán los datos exportados.	`s3://amzn-s3-demo-bucket/ EXPORT-JOB /`

Key	¿Obligatorio?	Description (Descripción)	Valor
KmsKeyId	Sí	Parte de una OutputDataConfig clave. El ARN de la AWS KMS clave utilizada para proteger el bucket de Amazon S3.	`arn:aws:kms: region-of-bucket:123456789012 :key/1234abcd-12ab-34cd-56ef-1234567890ab`

Example Cuerpo de una solicitud de exportación realizada mediante la API REST del FHIR

Para realizar una solicitud de exportación mediante la API REST del FHIR, debe especificar un cuerpo, como se muestra a continuación.

```
{
  "DataAccessRoleArn": "arn:aws:iam::444455556666:role/your-healthlake-service-role",
  "JobName": "your-export-job",
  "OutputDataConfig": {
    "S3Configuration": {
      "S3Uri": "s3://amzn-s3-demo-bucket/EXPORT-JOB",
      "KmsKeyId": "arn:aws:kms:region-of-
bucket:444455556666:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    }
  }
}
```

Cuando la solicitud se realice correctamente, recibirá la siguiente respuesta.

Encabezado de respuesta

```
content-location: https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
export/your-export-request-job-id
```

Cuerpo de respuesta

```
{
  "datastoreId": "your-data-store-id",
```

```
  "jobStatus": "SUBMITTED",
  "jobId": "your-export-request-job-id"
}
```

Gestionar tu solicitud de exportación

Tras realizar correctamente una solicitud de exportación, puede gestionarla `$export` para describir el estado de una solicitud de exportación actual y `$export` cancelar una solicitud de exportación actual.

Cuando cancelas una solicitud de exportación mediante la API REST, solo se te facturará la parte de los datos que se exportaron hasta el momento en que enviaste la solicitud de cancelación.

En los siguientes temas se describe cómo puede obtener el estado de una solicitud de exportación actual o cancelarla.

Cancelar una solicitud de exportación

Para cancelar una solicitud de exportación, realice una `DELETE` solicitud e introduzca el identificador de trabajo en la URL de la solicitud.

```
DELETE https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/export/your-export-request-job-id
```

Cuando la solicitud se realice correctamente, recibirá lo siguiente.

```
{
  "exportJobProperties": {
    "jobId": "your-original-export-request-job-id",
    "jobStatus": "CANCEL_SUBMITTED",
    "datastoreId": "your-data-store-id"
  }
}
```

Si su solicitud no se acepta, recibirá lo siguiente.

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
```

```
      "code": "not-supported",
      "diagnostics": "Interaction not supported."
    }
  ]
}
```

Describir una solicitud de exportación

Para conocer el estado de una solicitud de exportación, haga una GET solicitud utilizando `export` y su**`export-request-job-id`**.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/export/your-export-request-id
```

La respuesta de JSON contendrá un `ExportJobProperties` objeto. Puede contener los siguientes pares clave-valor.

Name	¿Obligatorio?	Description (Descripción)	Valor
DataAccessRoleArn	No	Un ARN de un rol de HealthLake servicio. El rol de servicio utilizado debe especificarse HealthLake como principal de servicio.	`arn:aws:iam:: 444455556666 :role/your-healthlake-service-role`
SubmitTime	No	Fecha y hora en que se envió un trabajo de exportación.	`Apr 21, 2023 5:58:02`
EndTime	No	La hora en que se completó un trabajo de exportación.	`Apr 21, 2023 6:00:08 PM`
JobName	No	El nombre de la solicitud de exportación.	**`your-export-job-name`**

Name	¿Obligatorio?	Description (Descripción)	Valor
JobStatus	No		Los valores válidos son los siguientes: `SUBMITTED | IN_PROGRESS | COMPLETED_WITH_ERRORS | COMPLETED | FAILED`
S3Uri	Sí	Parte de un OutputDataConfigobjeto. El URI de Amazon S3 del depósito de destino donde se descargarán los datos exportados.	`s3://amzn-s3-demo-bucket/ EXPORT-JOB /`
KmsKeyId	Sí	Parte de un OutputDataConfigobjeto. El ARN de la AWS KMS clave utilizada para proteger el bucket de Amazon S3.	`arn:aws:kms: region-of-bucket:123456789012 :key/1234abcd-12ab-34cd-56ef-1234567890ab`

Example: Cuerpo de una solicitud de exportación descrita realizada mediante la API REST del FHIR

Si se ejecuta correctamente, obtendrá la siguiente respuesta en JSON.

```
{
  "exportJobProperties": {
    "jobId": "your-export-request-id",
    "JobName": "your-export-job",
    "jobStatus": "SUBMITTED",
```

```
    "submitTime": "Apr 21, 2023 5:58:02 PM",
    "endTime": "Apr 21, 2023 6:00:08 PM",
    "datastoreId": "your-data-store-id",
    "outputDataConfig": {
      "s3Configuration": {
        "S3Uri": "s3://amzn-s3-demo-bucket/EXPORT-JOB",
        "KmsKeyId": "arn:aws:kms:region-of-
bucket:444455556666:key/1234abcd-12ab-34cd-56ef-1234567890ab""
      }
    },
    "DataAccessRoleArn": "arn:aws:iam::444455556666:role/your-healthlake-service-role",
  }
}
```

$inquire Operación FHIR para HealthLake

La `$inquire` operación le permite comprobar el estado de una solicitud de autorización previa presentada anteriormente. Esta operación implementa la Guía de implementación de Da Vinci Prior Authorization Support (PAS), que proporciona un flujo de trabajo estandarizado basado en el FHIR para recuperar la decisión de autorización actual.

Funcionamiento

- Enviar consulta: envías un paquete FHIR que contiene la reclamación que deseas comprobar y la información justificativa
- Búsqueda: HealthLake busca la correspondiente ClaimResponse en su almacén de datos
- Recuperar: se recupera el estado de autorización más reciente
- Responder: recibirá una respuesta inmediata con el estado actual de la autorización (en cola, aprobada, denegada, etc.)

> **Note**
>
> `$inquire`es una operación de solo lectura que recupera el estado de autorización existente. No modifica ni actualiza ningún recurso del almacén de datos.

Punto de conexión de la API

```
POST /datastore/{datastoreId}/r4/Claim/$inquire
```

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

Estructura de la solicitud

Requisitos del paquete

Su solicitud debe ser un recurso del paquete FHIR con:

- Tipo de paquete: debe ser `"collection"`
- Bundle.entry: debe contener exactamente un recurso de reclamación con:
  - `use = "preauthorization"`
  - `status = "active"`
- Recursos de referencia: todos los recursos a los que se hace referencia en la reclamación deben incluirse en el paquete

> **Consulta por ejemplo**
>
> Los recursos de tu paquete de entrada sirven como plantilla de búsqueda. HealthLake utiliza la información proporcionada para localizar el correspondiente ClaimResponse.

Recursos necesarios de

Recurso	Cardinalidad	Perfil	Description (Descripción)
Reclamación	1	Consulta de reclamación PAS	La autorización previa sobre la que está solicitando información
¿Paciente	1	Paciente beneficiario del PAS	Información demográfica del paciente
Organización (aseguradora)	1	Organización aseguradora PAS	Compañía de seguros
Organización (proveedor)	1	Organización solicitante del PAS	Proveedor de atención médica que presentó la solicitud

Criterios de búsqueda importantes

HealthLake busca el ClaimResponse uso de:

- Referencia del paciente recogida en la reclamación
- Referencia de la aseguradora en la reclamación
- Referencia del proveedor en la reclamación
- Fecha de creación a partir de la reclamación (como filtro de tiempo)

> **Solo consultas específicas para pacientes**
>
> Todas las consultas deben estar vinculadas a un paciente específico. No se permiten consultas en todo el sistema sin la identificación del paciente.

Ejemplo de solicitud

```
POST /datastore/example-datastore/r4/Claim/$inquire
Content-Type: application/fhir+json
Authorization: Bearer <your-token>

{
  "resourceType": "Bundle",
  "id": "PASClaimInquiryBundleExample",
  "meta": {
    "profile": ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-pas-
inquiry-request-bundle"]
  },
  "identifier": {
    "system": "http://example.org/SUBMITTER_TRANSACTION_IDENTIFIER",
    "value": "5269368"
  },
  "type": "collection",
  "timestamp": "2005-05-02T14:30:00+05:00",
  "entry": [
    {
      "fullUrl": "http://example.org/fhir/Claim/MedicalServicesAuthorizationExample",
      "resource": {
        "resourceType": "Claim",
        "id": "MedicalServicesAuthorizationExample",
        "meta": {
```

```
          "profile": ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
claim-inquiry"]
        },
        "status": "active",
        "type": {
          "coding": [{
            "system": "http://terminology.hl7.org/CodeSystem/claim-type",
            "code": "professional"
          }]
        },
        "use": "preauthorization",
        "patient": {
          "reference": "Patient/SubscriberExample"
        },
        "created": "2005-05-02T11:01:00+05:00",
        "insurer": {
          "reference": "Organization/InsurerExample"
        },
        "provider": {
          "reference": "Organization/UMOExample"
        }
      }
    },
    {
      "fullUrl": "http://example.org/fhir/Patient/SubscriberExample",
      "resource": {
        "resourceType": "Patient",
        "id": "SubscriberExample",
        "meta": {
          "profile": ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
beneficiary"]
        },
        "name": [{
          "family": "SMITH",
          "given": ["JOE"]
        }],
        "gender": "male"
      }
    },
    {
      "fullUrl": "http://example.org/fhir/Organization/UMOExample",
      "resource": {
        "resourceType": "Organization",
        "id": "UMOExample",
```

```
        "meta": {
          "profile": ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
requestor"]
        },
        "name": "Provider Organization"
      }
    },
    {
      "fullUrl": "http://example.org/fhir/Organization/InsurerExample",
      "resource": {
        "resourceType": "Organization",
        "id": "InsurerExample",
        "meta": {
          "profile": ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
insurer"]
        },
        "name": "Insurance Company"
      }
    }
  ]
}
```

Formato de las respuestas

Respuesta correcta (200 OK)

Recibirá un paquete de respuestas a una consulta del PAS que contiene:

- ClaimResponsecon el estado de autorización actual; múltiple ClaimResponsesi coincide con los criterios de búsqueda
- Todos los recursos originales de tu solicitud (reproducidos)
- Marca de tiempo del momento en que se recopiló la respuesta

Posibles resultados ClaimResponse

Resultado	Description (Descripción)
`queued`	La solicitud de autorización aún está pendiente de revisión

Resultado	Description (Descripción)
`complete`	Se ha tomado la decisión de autorizar (compruebe si `disposition` está aprobada o denegada)
`error`	Se ha producido un error durante el procesamiento
`partial`	Autorización parcial concedida

```
{
  "resourceType": "Bundle",
  "identifier": {
        "system": "http://example.org/SUBMITTER_TRANSACTION_IDENTIFIER",
        "value": "5269367"
  },
  "type": "collection",
  "timestamp": "2005-05-02T14:30:15+05:00",
  "entry": [
    {
      "fullUrl": "http://example.org/fhir/ClaimResponse/InquiryResponseExample",
      "resource": {
        "resourceType": "ClaimResponse",
        "id": "InquiryResponseExample",
        "meta": {
          "profile": ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
claimresponse-inquiry"]
        },
        "status": "active",
        "type": {
          "coding": [{
            "system": "http://terminology.hl7.org/CodeSystem/claim-type",
            "code": "professional"
          }]
        },
        "use": "preauthorization",
        "patient": {
          "reference": "Patient/SubscriberExample"
        },
        "created": "2005-05-02T11:05:00+05:00",
        "insurer": {
          "reference": "Organization/InsurerExample"
```

```
        },
        "request": {
          "reference": "Claim/MedicalServicesAuthorizationExample"
        },
        "outcome": "complete",
        "disposition": "Approved",
        "preAuthRef": "AUTH12345"
      }
    },
    {
      "fullUrl": "http://example.org/fhir/Claim/MedicalServicesAuthorizationExample",
      "resource": {
        "resourceType": "Claim",
        "id": "MedicalServicesAuthorizationExample",
        "meta": {
          "profile": ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
claim-inquiry"]
        },
        "status": "active",
        "type": {
          "coding": [{
            "system": "http://terminology.hl7.org/CodeSystem/claim-type",
            "code": "professional"
          }]
        },
        "use": "preauthorization",
        "patient": {
          "reference": "Patient/SubscriberExample"
        },
        "created": "2005-05-02T11:01:00+05:00",
        "insurer": {
          "reference": "Organization/InsurerExample"
        },
        "provider": {
          "reference": "Organization/UMOExample"
        }
      }
    },
    {
      "fullUrl": "http://example.org/fhir/Patient/SubscriberExample",
      "resource": {
        "resourceType": "Patient",
        "id": "SubscriberExample",
        "meta": {
```

```
          "profile": ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
beneficiary"]
        },
        "name": [{
          "family": "SMITH",
          "given": ["JOE"]
        }],
        "gender": "male"
      }
    },
    {
      "fullUrl": "http://example.org/fhir/Organization/UMOExample",
      "resource": {
        "resourceType": "Organization",
        "id": "UMOExample",
        "meta": {
          "profile": ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
requestor"]
        },
        "name": "Provider Organization"
      }
    },
    {
      "fullUrl": "http://example.org/fhir/Organization/InsurerExample",
      "resource": {
        "resourceType": "Organization",
        "id": "InsurerExample",
        "meta": {
          "profile": ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
insurer"]
        },
        "name": "Insurance Company"
      }
    }
  ]
}
```

Respuestas de error

400: solicitud maligna

Se devuelve cuando el formato de la solicitud no es válido o se produce un error en la validación.

```
{
```

```
    "resourceType": "OperationOutcome",
    "issue": [
        {
            "severity": "error",
            "code": "required",
            "diagnostics": "Reference 'Patient/SubscriberExample' at path 'patient' for
 'CLAIM' resource not found(at Bundle.entry[0].resource)"
        }
    ]
}
```

401 sin autorización

Se devuelve cuando faltan credenciales de autenticación o no son válidas.

```
{
    "resourceType": "OperationOutcome",
    "issue": [
        {
            "severity": "error",
            "code": "forbidden",
            "diagnostics": "Invalid authorization header"
        }
    ]
}
```

403: prohibido

Se devuelve cuando el usuario autenticado no tiene permiso para acceder al recurso solicitado.

```
{
    "resourceType": "OperationOutcome",
    "issue": [
        {
            "severity": "error",
            "code": "exception",
            "diagnostics": "Insufficient SMART scope permissions."
        }
    ]
}
```

400 Cuando no se encuentra ninguno

Se devuelve cuando no ClaimResponse se encuentra ninguna coincidencia con la consulta.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "not-found",
    "diagnostics": "Resource not found. No ClaimResponse found from the input Claim
 that matches the specified Claim properties patient, insurer, provider, and created(at
 Bundle.entry[0].resource)"
  }]
}
```

4.15 Tipo de medio no compatible

Se devuelve cuando el encabezado Content-Type no es application/fhir+json.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "value",
    "diagnostics": "Incorrect MIME-type. Update request Content-Type header."
  }]
}
```

429 Demasiadas solicitudes

Se devuelve cuando se superan los límites de velocidad.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "throttled",
    "diagnostics": "Rate limit exceeded. Please retry after some time."
  }]
}
```

Reglas de validación

HealthLake realiza una validación exhaustiva de su consulta:

Validación de paquetes

- Debe ajustarse al perfil del paquete de solicitud de consulta del PAS
- `Bundle.type`debe ser `"collection"`
- Debe contener exactamente un recurso de reclamación
- Todos los recursos a los que se hace referencia deben incluirse en el paquete

Validación de la reclamación

- Debe ajustarse al perfil de consulta de reclamaciones de PAS
- `Claim.use`debe ser `"preauthorization"`
- `Claim.status`debe ser `"active"`
- Campos obligatorios:`patient`,`insurer`,`provider`, `created`

Validación de recursos

- Todos los recursos deben ajustarse a sus respectivos perfiles de consulta del PAS
- Deben estar presentes los recursos de apoyo necesarios (paciente, organización aseguradora, organización proveedora)
- Las referencias cruzadas deben ser válidas y poder resolverse dentro del paquete

Especificaciones de rendimiento

Métrica	Especificación
Límite de recuento de recursos	500 recursos por paquete
Límite de tamaño del paquete	5 MB como máximo

Permisos necesarios

Para utilizar la `$inquire` operación, asegúrese de que su función de IAM tenga:

- `healthlake:InquirePreAuthClaim`- Para convocar a la operación

SMART en FHIR Scopes

Alcances mínimos requeridos:

- SMART v1: `user/ClaimResponse.read`
- SMART v2: `user/ClaimResponse.s`

Notas importantes sobre la implementación

Comportamiento de búsqueda

Al enviar una consulta, HealthLake busca el ClaimResponse uso de:

- Referencia del paciente a partir de la reclamación ingresada
- Referencia de la aseguradora a partir de la reclamación ingresada
- Referencia del proveedor a partir de la reclamación ingresada
- Fecha de creación a partir de la reclamación ingresada (como filtro de tiempo)

Coincidencias múltiples: si varias ClaimResponses coinciden con sus criterios de búsqueda, HealthLake devuelve todos los resultados coincidentes. Debe usar la `ClaimResponse.created` marca de tiempo más reciente para identificar el estado más reciente.

Reclamaciones actualizadas

Si has enviado varias actualizaciones a la misma autorización previa (p. ej., Claim v1.1, v1.2 o v1.3), la `$inquire` operación recuperará la versión ClaimResponse asociada a la versión más reciente en función de los criterios de búsqueda proporcionados.

Operación de solo lectura

La `$inquire` operación:

- Recupera el estado de autorización existente
- Devuelve el más reciente ClaimResponse
- No modifica ni actualiza ningún recurso
- No crea nuevos recursos

- No desencadena un nuevo procesamiento de autorizaciones

Ejemplo de flujo de trabajo

Flujo de trabajo típico de consultas de autorización previa

```
1. Provider submits PA request
   POST /Claim/$submit
   # Returns ClaimResponse with outcome="queued"

2. Payer reviews request (asynchronous)
   # Updates ClaimResponse status internally

3. Provider checks status
   POST /Claim/$inquire
   # Returns ClaimResponse with outcome="queued" (still pending)

4. Provider checks status again later
   POST /Claim/$inquire
   # Returns ClaimResponse with outcome="complete", disposition="Approved"
```

Operaciones de relacionadas

- `Claim/$submit`- Presente una nueva solicitud de autorización previa o actualice una existente
- `Patient/$everything`- Recupere datos completos de los pacientes para utilizarlos en el contexto de la autorización previa

Recuperación de detalles conceptuales con **`$lookup`**

AWS HealthLake ahora admite la `$lookup` operación de CodeSystem recursos, lo que le permite recuperar detalles sobre un concepto específico de un sistema de códigos proporcionando información de identificación, como su código. Esta operación resulta especialmente útil cuando se necesita:

- Recuperar información detallada sobre códigos médicos específicos
- Valide el significado y las propiedades del código
- Acceda a las definiciones y relaciones de los conceptos
- Support la toma de decisiones clínicas con datos terminológicos precisos

De uso

La `$lookup` operación se puede invocar en CodeSystem los recursos mediante los métodos GET y POST:

Operaciones admitidas

```
GET [base]/CodeSystem/$lookup?system=http://snomed.info/
sct&code=73211009&version=20230901
POST [base]/CodeSystem/$lookup
```

Parámetros admitidos

HealthLake admite un subconjunto de parámetros del FHIR `$lookup` R4:

Parámetro	Tipo	Obligatorio	Description (Descripción)
`code`	code	Sí	El código conceptual que está buscando (por ejemplo, «71620000" en SNOMED CT)
`system`	uri	Sí	La URL canónica del sistema de códigos (por ejemplo, "http://snomed.info/sct «)
`version`	cadena	No	Versión específica del sistema de códigos

Ejemplos

Solicitud GET

```
GET [base]/CodeSystem/$lookup?system=http://snomed.info/
sct&code=71620000&version=2023-09
```

Solicitud POST

```
POST [base]/CodeSystem/$lookup
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
```

```
  "parameter": [
    {
      "name": "system",
      "valueUri": "http://snomed.info/sct"
    },
    {
      "name": "code",
      "valueCode": "71620000"
    },
    {
      "name": "version",
      "valueString": "2023-09"
    }
  ]
}
```

Respuesta de ejemplo

La operación devuelve un recurso de parámetros que contiene los detalles del concepto:

```
{
    "resourceType": "Parameters",
    "parameter": [{
            "name": "name",
            "valueString": "SNOMED CT Fractures"
        },
        {
            "name": "version",
            "valueString": "2023-09"
        },
        {
            "name": "display",
            "valueString": "Fracture of femur"
        },
        {
            "name": "property",
            "part": [{
                    "name": "code",
                    "valueCode": "child"
                },
                {
                    "name": "value",
                    "valueCode": "263225007"
                },
```

```
                {
                    "name": "description",
                    "valueString": "Fracture of neck of femur"
                }
            ]
        },
        {
            "name": "property",
            "part": [{
                    "name": "code",
                    "valueCode": "child"
                },
                {
                    "name": "value",
                    "valueCode": "263227004"
                },
                {
                    "name": "description",
                    "valueString": "Fracture of shaft of femur"
                }
            ]
        }
    ]
}
```

Parámetros de respuesta

La respuesta incluye los siguientes parámetros cuando están disponibles:

Parámetro	Tipo	Description (Descripción)
`name`	cadena	Nombre del sistema de códigos
`version`	cadena	Versión del sistema de códigos
`display`	cadena	Mostrar el nombre del concepto
`designation`	BackboneElement	Representaciones adicionales para este concepto.
`property`	BackboneElement	Propiedades adicionales del concepto (definición, relaciones, etc.)

Comportamiento

La `$lookup` operación:

1. Valida los parámetros necesarios (`codeysystem`)
2. Busca el concepto en el sistema de códigos especificado almacenado en el almacén de datos
3. Devuelve información conceptual detallada, incluidos el nombre para mostrar, las designaciones y las propiedades.
4. Admite búsquedas específicas de la versión cuando se proporciona el parámetro `version`
5. Funciona solo en sistemas de código almacenados explícitamente en el almacén de datos HealthLake

Gestión de errores

La operación gestiona las siguientes condiciones de error:

- 400 Solicitud errónea: `$lookup` operación no válida (solicitud no conforme o faltan parámetros obligatorios)
- 404 No encontrado: no se encontró el sistema de códigos o el código no se encontró en el sistema de códigos especificado

Advertencias

En esta versión, no se admite lo siguiente:

- `$lookup`operación mediante la llamada a servidores terminológicos externos
- `$lookup`operación CodeSystems gestionada por el almacén de datos HealthLake , pero no almacenada explícitamente en él

Para obtener más información sobre la especificación de la `$lookup` operación, consulte la documentación del FHIR R4. CodeSystem `$lookup`

`$member-add`operación para HealthLake

La `$member-add` operación del FHIR añade a un miembro (paciente) a un recurso del grupo, específicamente a una lista de atribuciones de miembros. Esta operación forma parte de la Guía de

implementación de las atribuciones de DaVinci los miembros y apoya el proceso de conciliación para gestionar las atribuciones de los miembros.

Operation Endpoint

```
POST [base]/datastore/{datastoreId}/r4/Group/{groupId}/$member-add
Content-Type: application/json
```

Parameters

La operación acepta un recurso de parámetros del FHIR con las siguientes combinaciones de parámetros:

Opciones de parámetros

Puede utilizar una de las siguientes combinaciones de parámetros:

Opción 1: ID de miembro + NPI del proveedor

- `memberId` + `providerNpi`
- `memberId` + `providerNpi` + `attributionPeriod`

Opción 2: referencia del paciente + referencia del proveedor

- `patientReference` + `providerReference`
- `patientReference` + `providerReference` + `attributionPeriod`

Detalles de los parámetros

ID de miembro (opcional)

El identificador del miembro que se va a añadir al grupo.

Tipo: identificador

Sistema: sistema de identificación de pacientes

```
{
  "name": "memberId",
  "valueIdentifier": {
    "system": "http://example.org/patient-id",
    "value": "patient-new"
```

```
  }
}
```

Proveedor NPI (opcional)

El identificador nacional de proveedores (NPI) del proveedor atribuido.

Tipo: identificador

Sistema: http://terminology.hl7. org/CodeSystem/NPI

```
{
  "name": "providerNpi",
  "valueIdentifier": {
    "system": "http://terminology.hl7.org/CodeSystem/NPI",
    "value": "1234567890"
  }
}
```

Referencia del paciente (opcional)

Referencia directa al recurso para pacientes que se va a añadir.

Tipo: Referencia

```
{
  "name": "patientReference",
  "valueReference": {
    "reference": "Patient/patient-123"
  }
}
```

ProviderReference (opcional)

Referencia directa al recurso del proveedor.

Tipo: Referencia

```
{
  "name": "providerReference",
  "valueReference": {
    "reference": "Practitioner/provider-456"
  }
```

```
}
```

Periodo de atribución (opcional)

El período durante el cual se atribuye el paciente al proveedor.

Tipo: Periodo

```
{
  "name": "attributionPeriod",
  "valuePeriod": {
    "start": "2024-07-15",
    "end": "2025-07-14"
  }
}
```

Ejemplos de solicitudes

Uso de la identificación de miembro y el NPI del proveedor

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "memberId",
      "valueIdentifier": {
        "system": "http://example.org/patient-id",
        "value": "patient-new"
      }
    },
    {
      "name": "providerNpi",
      "valueIdentifier": {
        "system": "http://terminology.hl7.org/CodeSystem/NPI",
        "value": "1234567890"
      }
    },
    {
      "name": "attributionPeriod",
      "valuePeriod": {
        "start": "2024-07-15",
        "end": "2025-07-14"
      }
```

```
    }
  ]
}
```

Uso de referencias de pacientes y proveedores

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "patientReference",
      "valueReference": {
        "reference": "Patient/patient-123"
      }
    },
    {
      "name": "providerReference",
      "valueReference": {
        "reference": "Practitioner/provider-456"
      }
    },
    {
      "name": "attributionPeriod",
      "valuePeriod": {
        "start": "2024-07-15",
        "end": "2025-07-14"
      }
    }
  ]
}
```

Formato de respuesta

Respuesta de adición exitosa

```
HTTP Status: 200 OK
Content-Type: application/fhir+json

{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "success",
```

```
      "code": "informational",
      "details": {
        "text": "Member Patient/patient-new successfully added to the Member
 Attribution List."
      }
    }
  ]
}
```

Respuestas de error

Sintaxis de solicitud no válida

Estado HTTP: 400 solicitud incorrecta

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "invalid",
      "details": {
        "text": "Invalid parameter combination provided"
      }
    }
  ]
}
```

Recurso no encontrado

Estado HTTP: 404 no encontrado

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "not-found",
      "details": {
        "text": "Resource not found."
      }
    }
  ]
```

```
}
```

Conflicto de versiones

Estado HTTP: 409 Conflicto

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "conflict",
      "details": {
        "text": "Resource version conflict detected"
      }
    }
  ]
}
```

Estado de atribución no válido

Estado HTTP: 422 Entidad inprocesable

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "business-rule",
      "details": {
        "text": "Cannot add member to Attribution List with status 'final'. Status
 must be 'draft' or 'open'."
      }
    }
  ]
}
```

Reglas de negocio

Validación del estado de la atribución

La operación solo se puede realizar cuando el estado de atribución del grupo es:

- `draft`
- `open`

No se permiten operaciones cuando el estado es`final`.

Prevención de miembros duplicados

El sistema evita la adición de miembros duplicados en función de la combinación única de:

- Identificador de miembro
- Identificador del pagador
- Identificador de cobertura

Validación del período de cobertura

Cuando `attributionPeriod` se proporciona uno, debe estar dentro de los límites del período de cobertura del miembro. El sistema hará lo siguiente:

- Busque el recurso de cobertura del miembro
- Usa la cobertura más reciente (VersionID más alta) si existen varias
- Valide que el período de atribución no exceda el período de cobertura

Validación de referencia

Cuando se proporcionan tanto la identificación como la referencia para el mismo recurso (paciente o proveedor), el sistema valida que corresponden al mismo recurso.

Cuando se proporcionan los campos ID y reference.identifier para el mismo recurso (paciente o proveedor), se produce un error.

Autenticación y autorización

La operación requiere la autorización de SMART on FHIR para:

- Permisos de lectura: para validar los recursos de pacientes, proveedores y grupos
- Permisos de búsqueda: para buscar recursos por identificador
- Permisos de actualización: para modificar el recurso del grupo

Comportamiento operativo

Actualizaciones de recursos

- Actualiza el ID de versión del recurso del grupo

- Crea una entrada en el historial con el estado original del recurso antes de la operación
- Añade la información de los miembros a la matriz group.Member con:
  - Referencia del paciente en entity.reference
  - Período de atribución en período
  - Información sobre la cobertura y el proveedor en los campos de extensión

Pasos de validación

- Validación de parámetros: garantiza la validez de las combinaciones de parámetros
- Existencia de recursos: valida la existencia de recursos para pacientes, proveedores y grupos
- Estado de atribución: confirma que el estado del grupo permite modificaciones
- Verificación duplicada: impide añadir miembros existentes
- Validación de la cobertura: garantiza que el período de atribución esté dentro de los límites de cobertura

Limitaciones

- Todos los recursos a los que se hace referencia deben estar en el mismo almacén de datos
- La operación solo funciona con los recursos del grupo de listas de atribución de miembros
- Los períodos de atribución deben estar dentro de los límites de cobertura
- No se pueden modificar los grupos con el estado «final»

`$member-match`operación para HealthLake

AWS HealthLake ahora apoya la `$member-match` operación de recursos para pacientes, lo que permite a las organizaciones de atención médica encontrar el identificador único de un miembro en diferentes sistemas de salud utilizando información demográfica y de cobertura. Esta operación es esencial para lograr el cumplimiento de los CMS y facilitar el intercambio seguro de payer-to-payer datos, al tiempo que se mantiene la privacidad de los pacientes.

Esta operación resulta especialmente útil cuando se necesita:

- Permitir el intercambio seguro de datos de salud entre organizaciones
- Mantenga la continuidad de la atención de los pacientes en los diferentes sistemas
- Support CMS cumple con los requisitos
- Facilite la identificación precisa de los miembros en todas las redes de atención médica

De uso

La `$member-match` operación se puede invocar en los recursos del paciente mediante el método POST:

```
POST [base]/Patient/$member-match
```

Parámetros admitidos

HealthLake admite los siguientes `$member-match` parámetros del FHIR:

Parámetro	Tipo	Obligatorio/a	Predeterminado	Description (Descripción)
MemberPatient	Paciente	Sí	—	Recurso para pacientes que contiene información demográfica sobre el miembro al que se va a vincular
CoverageToMatch	Cobertura	Sí	—	Recurso de cobertura que se utilizará para compararlo con los registros existentes
CoverageToLink	Cobertura	No	—	Recurso de cobertura que se vinculará durante el proceso de emparejamiento
Consentimiento	Consentimiento	No	—	Recurso de consentimiento para fines de autorización

Ejemplos

Solicitud POST con parámetros

```
POST [base]/Patient/$member-match
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
  "parameter": [
```

```
    {
      "name": "MemberPatient",
      "resource": {
        "resourceType": "Patient",
        "name": [
          {
            "family": "Jones",
            "given": ["Sarah"]
          }
        ],
        "gender": "female",
        "birthDate": "1985-05-15"
      }
    },
    {
      "name": "CoverageToMatch",
      "resource": {
        "resourceType": "Coverage",
        "status": "active",
        "beneficiary": {
          "reference": "Patient/1"
        },
        "relationship": {
          "coding": [
            {
              "system": "http://terminology.hl7.org/CodeSystem/subscriber-
relationship",
              "code": "self",
              "display": "Self"
            }
          ]
        },
        "payor": [
          {
            "reference": "Organization/payer456"
          }
        ]
      }
    },
    {
      "name": "Consent",
      "resource": {
        "resourceType": "Consent",
        "status": "active",
```

```
        "scope": {
          "coding": [
            {
              "system": "http://terminology.hl7.org/CodeSystem/consentscope",
              "code": "patient-privacy"
            }
          ]
        },
        "category": [
          {
            "coding": [
              {
                "system": "http://terminology.hl7.org/CodeSystem/v3-ActCode",
                "code": "IDSCL"
              }
            ]
          }
        ],
        "patient": {
          "reference": "Patient/1"
        },
        "performer": [
          {
            "reference": "Patient/patient123"
          }
        ],
        "sourceReference": {
          "reference": "Document/someconsent"
        },
        "policy": [
          {
            "uri": "http://hl7.org/fhir/us/davinci-hrex/StructureDefinition-hrex-
consent.html#regular"
          }
        ]
      }
    }
  ]
}
```

Respuesta de ejemplo

La operación devuelve un recurso de parámetros que contiene los resultados coincidentes:

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "MemberIdentifier",
      "valueIdentifier": {
        "system": "http://hospital.org/medical-record-number",
        "value": "MRN-123456"
      }
    },
    {
      "name": "MemberId",
      "valueReference": {
        "reference": "Patient/patient123"
      }
    },
    {
      "name": "matchAlgorithm",
      "valueString": "DEMOGRAPHIC_MATCH"
    },
    {
      "name": "matchDetails",
      "valueString": "Demographic match: DOB + Name"
    },
    {
      "name": "matchedFields",
      "valueString": "given,birthdate,gender,family"
    }
  ]
}
```

Parámetros de respuesta

La respuesta incluye los siguientes parámetros cuando se encuentra una coincidencia:

Parámetro	Tipo	Description (Descripción)
MemberIdentifier	Identificador	El identificador único del miembro coincidente
MemberId	Referencia	Referencia al recurso para pacientes

Parámetro	Tipo	Description (Descripción)
Algoritmo de coincidencia	Cadena	Tipo de algoritmo de coincidencia utilizado (EXACT_MATCH, STRONG_MATCH o DEMOGRAPHIC_MATCH)
Detalles del partido	Cadena	Información detallada sobre el proceso de emparejamiento
Campos coincidentes	Cadena	Lista de campos específicos que se emparejar on correctamente

algoritmos coincidentes

La `$member-match` API emplea un enfoque de coincidencia de varios niveles para garantizar una identificación precisa de los miembros:

EXACT_MATCH

Utiliza el identificador del paciente combinado con la cobertura SubscriberId

Proporciona el nivel de confianza más alto para la identificación de miembros

STRONG_MATCH

Utiliza el identificador del paciente con información de cobertura mínima

Ofrece una gran confianza cuando no se cumplen los criterios de coincidencia exactos

DEMOGRAPHIC_MATCH

Se basa en información demográfica básica

Se utiliza cuando no es posible la coincidencia basada en identificadores

Comportamiento

La operación: `$member-match`

- Acepta como datos los datos demográficos del paciente, los detalles de la cobertura y la información de consentimiento opcional

- Devuelve un identificador de miembro único que se puede utilizar en interacciones posteriores
- Implementa una combinación de varios niveles (exacta, sólida y demográfica) para garantizar una identificación precisa de los miembros en los diferentes sistemas de salud
- Guarda cualquier información de consentimiento proporcionada para futuros fines de autorización
- Soporta el intercambio seguro payer-to-payer de datos al tiempo que mantiene la privacidad del paciente
- Cumple con los requisitos de los CMS para el intercambio de datos sanitarios

Autorización

La API utiliza el protocolo de autorización SMART on FHIR con los siguientes alcances obligatorios:

- `system/Patient.read`
- `system/Coverage.read`
- `system/Organization.read`(condicional)
- `system/Practitioner.read`(condicional)
- `system/PractitionerRole.read`(condicional)
- `system/Consent.write`(condicional)

Gestión de errores

La operación gestiona las siguientes condiciones de error:

- `400 Bad Request`: `$member-match` Operación no válida (solicitud no conforme o faltan parámetros obligatorios)
- `422 Unprocessable Entity`: No se encontraron coincidencias o se encontraron varias coincidencias

`$member-remove`operación para HealthLake

La `$member-remove` operación le permite eliminar miembros de una lista de atribución de miembros de la FHIR (recurso grupal) en. AWS HealthLake Esta operación forma parte de la Guía de implementación de la atribución de DaVinci miembros y apoya el proceso de conciliación para gestionar las atribuciones de los miembros.

Requisitos previos

- AWS HealthLake Almacén de datos FHIR
- Permisos de IAM adecuados para las operaciones HealthLake
- Lista de atribuciones de miembros (recurso grupal) en estado borrador o abierta

Detalles de la operación

Punto de conexión

```
POST /Group/{id}/$member-remove
```

Tipo de contenido

```
application/fhir+json
```

Parameters

La operación acepta un recurso de parámetros del FHIR con los siguientes parámetros opcionales:

Parámetro	Cardinalidad	Tipo	Description (Descripción)
memberId	0.1	Identificador	Identificador empresarial del miembro que se va a eliminar
NPI del proveedor	0.1..	Identificador	NPI del proveedor atribuido
Referencia del paciente	0.1..	Referencia	Referencia directa al recurso para pacientes
Referencia del proveedor	0.1..	Referencia	Referencia directa al recurso del proveedor (profesional u organización) PractitionerRole
Referencia de cobertura	0.1..	Referencia	Referencia al recurso Cobertura

Combinaciones de parámetros compatibles

Se admiten las siguientes combinaciones de parámetros:

- `memberId`solo: elimina todas las atribuciones del miembro especificado
- `memberId`+ `providerNpi` - Elimina las atribuciones de la combinación específica de miembro y proveedor
- `patientReference`solo: elimina todas las atribuciones del paciente especificado
- `patientReference`+ `providerReference` - Elimina las atribuciones de la combinación específica de paciente y proveedor
- `patientReference`+ `providerReference` + `coverageReference` - Elimina la atribución específica en función del paciente, el proveedor y la cobertura

Ejemplo de solicitud

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "patientReference",
      "valueReference": {
        "reference": "Patient/12345"
      }
    },
    {
      "name": "providerReference",
      "valueReference": {
        "reference": "Practitioner/67890"
      }
    }
  ]
}
```

Respuesta

Respuesta correcta

```
{
  "resourceType": "Parameters",
  "parameter": [
```

```
    {
      "name": "result",
      "valueBoolean": true
    },
    {
      "name": "effectiveDate",
      "valueDate": "2024-06-30"
    },
    {
      "name": "status",
      "valueCode": "inactive"
    },
    {
      "name": "message",
      "valueString": "Member successfully removed from attribution list"
    }
  ]
}
```

Comportamiento

Requisitos de estado

- La operación solo funciona en listas de atribución con estado `draft` o `open`
- Las listas con `final` estado rechazarán la operación con un error 422

Proceso de eliminación de miembros

- Borradores de listas de estado: los miembros se marcan como inactivos (`inactive: true`) y su `changeType` extensión se actualiza a `changed`
- Listas de estado abiertas: comportamiento similar al estado preliminar
- Listas de estado finales: se rechaza la operación

Validación

- Las referencias se validan para garantizar que existan en el HealthLake almacén de datos
- Si se proporcionan tanto el identificador como la referencia para el mismo tipo de recurso, deben corresponder al mismo recurso
- Las combinaciones de parámetros se validan de acuerdo con los patrones admitidos

Gestión de errores

Respuestas de error comunes

Recurso no encontrado (404)

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "not-found",
      "details": {
        "text": "Patient with identifier 'http://example.org/fhir/identifiers|99999'
 not found in system"
      },
      "diagnostics": "Cannot remove member from attribution list. Verify patient
 identifier and try again.",
      "expression": ["memberId"]
    }
  ]
}
```

Estado final de la lista de atribuciones (422)

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "business-rule",
      "details": {
        "coding": [
          {
            "system": "http://hl7.org/fhir/us/davinci-atr/CodeSystem/atr-error-
codes",
            "code": "list-final",
            "display": "Attribution list is final and cannot be modified"
          }
        ]
      },
      "diagnostics": "Cannot modify attribution list with status 'final'. List
 modifications are not permitted after finalization.",
```

```
      "expression": ["Group.status"]
    }
  ]
}
```

Operación no válida (400)

Se devuelve cuando las combinaciones de parámetros no son válidas o están mal formadas.

Se han encontrado varias coincidencias (412)

Se devuelve cuando los parámetros proporcionados coinciden con varios miembros de la lista de atribuciones.

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "processing",
      "diagnostics": "Multiple members found matching the criteria"
    }
  ]
}
```

Prácticas recomendadas

- Utilice parámetros específicos: cuando sea posible, utilice la combinación de parámetros más específica para evitar eliminaciones involuntarias
- Estado de la lista de atribuciones: comprueba el estado de la lista de atribuciones antes de intentar eliminarla
- Maneje los errores con elegancia: implemente un manejo de errores adecuado para todas las condiciones de error posibles
- Valide las referencias: asegúrese de que existan todos los recursos a los que se hace referencia antes de realizar la solicitud

Eliminar los recursos del compartimento de pacientes con **$purge**

AWS HealthLake apoya la `$purge` operación, lo que permite la eliminación permanente de todos los recursos del compartimento del paciente. Esta operación resulta especialmente útil cuando se necesita:

- Eliminar todos los datos asociados a un paciente
- Cumpla con las solicitudes de eliminación de datos de los pacientes
- Gestione el ciclo de vida de los datos
- Realice una limpieza exhaustiva de los registros de los pacientes

De uso

La `$purge` operación se puede invocar en los recursos del paciente:

```
POST [base]/Patient/[ID]/$purge?deleteAuditEvent=true
```

Parameters

Parámetro	Tipo	Obligatorio/a	Predeterminado	Description (Descripción)
`deleteAuditEvent`	booleano	No	false	Si es verdadero, elimina los eventos de auditoría asociados
`_since`	cadena	No	Hora de creación del almacén de datos	Cuando se introduce, selecciona la hora límite de inicio para buscar los recursos en función de su hora de última modificación. No se puede usar ni al principio ni al final
`start`	cadena	No	Hora de creación del almacén de datos	Cuando se introduce, selecciona la hora límite para buscar los recursos en función de su hora de última modificación. Se puede usar con el final

Parámetro	Tipo	Obligatorio/a	Predeterminado	Description (Descripción)
end	cadena	No	Tiempo de presentación de trabajos	Cuando se introduce, selecciona la hora límite de finalización para buscar los recursos en función de su hora de última modificación

Ejemplos

Solicitud de ejemplo

```
POST [base]/Patient/example-patient/$purge?deleteAuditEvent=true
```

Respuesta de ejemplo

```
{
  "resourceType": "OperationOutcome",
  "id": "purge-job",
  "issue": [
    {
      "severity": "information",
      "code": "informational",
      "diagnostics": "Purge job started successfully. Job ID:
 12345678-1234-1234-1234-123456789012"
    }
  ]
}
```

Estado del trabajo

Para comprobar el estado de un trabajo de purga:

```
GET [base]/$purge/[jobId]
```

La operación devuelve información sobre el estado del trabajo:

```
{
```

```
    "datastoreId": "36622996b1fcecb7e12ee2ee085308d3",
    "jobId": "3dd1c7a5b6c0ef8c110f566eb87e2ef9",
    "status": "COMPLETED",
    "submittedTime": "2025-10-31T18:43:21.822Z"
}
```

Comportamiento

La `$purge` operación:

1. Procesa de forma asíncrona para gestionar varios recursos
2. Mantiene las transacciones ACID para garantizar la integridad de los datos
3. Proporciona un seguimiento del estado de los trabajos con recuentos de eliminaciones de recursos
4. Elimina permanentemente todos los recursos del compartimento del paciente
5. Incluye un registro de auditoría completo de las actividades de eliminación
6. Soporta la eliminación selectiva de eventos de auditoría

Registro de auditoría

La `$purge` operación se registra como Inicio FHIRBulk DeleteJob y Descripción FHIRBulk DeleteJob con información detallada sobre la operación.

Limitaciones

- Los recursos purgados no aparecerán en las respuestas de búsqueda
- Es posible que los recursos que se están purgando estén inaccesibles temporalmente durante el procesamiento
- Todos los recursos del compartimento del paciente se retiran permanentemente

`$questionnaire-package`Operación FHIR para HealthLake

La `$questionnaire-package` operación recupera un paquete completo que contiene un cuestionario del FHIR y todas sus dependencias necesarias para procesar y procesar el cuestionario. Esta operación implementa la Guía de implementación de las plantillas y reglas de documentación (DTR) de Da Vinci, que permite la representación dinámica de los formularios para los requisitos de documentación en los flujos de trabajo de atención médica.

Funcionamiento

- Solicitud: Usted envía los parámetros que identifican los cuestionarios necesarios, junto con la cobertura y el contexto del pedido
- Recuperar: HealthLake recopila el cuestionario y todas las dependencias (ValueSetsbibliotecas de CQL, etc.)
- Paquete: Todos los recursos se agrupan en un formato estandarizado
- Responda: recibirá un paquete completo listo para su procesamiento y recopilación de datos

Casos de uso

- Documentación de autorización previa: recopile la información clínica requerida para las solicitudes de autorización previa
- Requisitos de cobertura: reúna la documentación necesaria para cumplir con los requisitos de cobertura del pagador
- Clinical Data Exchange: estructura los datos clínicos para enviarlos a los pagadores
- Formularios dinámicos: renderice cuestionarios con datos de pacientes previamente rellenados y lógica condicional

Punto de conexión de la API

```
POST /datastore/{datastoreId}/r4/Questionnaire/$questionnaire-package
Content-Type: application/fhir+json
```

Parámetros de solicitud

Parámetros de entrada

El cuerpo de la solicitud debe contener un recurso de parámetros del FHIR con los siguientes parámetros:

Parámetro	Tipo	Cardinalidad	Description (Descripción)
`coverage`	Cobertura	1.. * (Obligatorio)	Recurso (s) de cobertura para determinar el miembro y cobertura de la documentación

Parámetro	Tipo	Cardinalidad	Description (Descripción)
`questionnaire`	canonical	0.. *	URL canónicas de cuestionarios específicos a devolver (pueden incluir una versión)
`order`	Recurso	0.. *	Solicite recursos (DeviceRequest,, ServiceRequest MedicationRequest, Encuentro, Cita) para establecer el contexto
`changedSince`	dateTime	0.1	Si está presente, devuelva solo los recursos modificados después de esta marca de tiempo

Reglas de validación de parámetros

Debe proporcionarse al menos UNO de los siguientes datos (además de los obligatorios`coverage`):

- Uno o más `questionnaire` canónicos URLs
- Uno o más recursos `order`

Combinaciones de solicitudes válidas:

- `coverage` + `questionnaire`
- `coverage` + `order`
- `coverage` + `questionnaire` + `order`

Ejemplo de solicitud

```
POST /datastore/example-datastore/r4/Questionnaire/$questionnaire-package
Content-Type: application/fhir+json
Authorization: Bearer <your-token>

{
  "resourceType": "Parameters",
  "parameter": [
```

```
    {
      "name": "coverage",
      "resource": {
        "resourceType": "Coverage",
        "id": "example-coverage",
        "status": "active",
        "beneficiary": {
          "reference": "Patient/example-patient"
        },
        "payor": [{
          "reference": "Organization/example-payer"
        }],
        "class": [{
          "type": {
            "coding": [{
              "system": "http://terminology.hl7.org/CodeSystem/coverage-class",
              "code": "group"
            }]
          },
          "value": "12345"
        }]
      }
    },
    {
      "name": "questionnaire",
      "valueCanonical": "http://example.org/fhir/Questionnaire/home-oxygen-therapy|2.0"

    },
    {
      "name": "order",
      "resource": {
        "resourceType": "ServiceRequest",
        "id": "example-service-request",
        "status": "active",
        "intent": "order",
        "code": {
          "coding": [{
            "system": "http://www.ama-assn.org/go/cpt",
            "code": "94660",
            "display": "Continuous positive airway pressure ventilation (CPAP)"
          }]
        },
        "subject": {
          "reference": "Patient/example-patient"
```

```
        }
      }
    },
    {
      "name": "changedSince",
      "valueDateTime": "2024-01-01T00:00:00Z"
    }
  ]
}
```

Formato de las respuestas

Respuesta correcta (200 OK)

La operación devuelve un recurso de parámetros FHIR que contiene uno o más Package Bundles. Cada Package Bundle incluye:

Tipo de entrada	Cardinalidad	Description (Descripción)
Cuestionario	1	El cuestionario que se va a entregar
QuestionnaireResponse	0.1..	Respuesta rellenada previamente o completada parcialmente (si corresponde)
Library	0.. *	Bibliotecas CQL que contienen lógica condicional y previa a la población
ValueSet	0.. *	Ampliado ValueSets (para opciones de respuesta con menos de 40 expansiones)

Example Ejemplo de respuesta

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "PackageBundle",
      "resource": {
        "resourceType": "Bundle",
        "id": "questionnaire-package-example",
        "meta": {
```

```
          "profile": ["http://hl7.org/fhir/us/davinci-dtr/StructureDefinition/DTR-
QPackageBundle"]
        },
        "type": "collection",
        "timestamp": "2024-03-15T10:30:00Z",
        "entry": [
          {
            "fullUrl": "http://example.org/fhir/Questionnaire/home-oxygen-therapy",
            "resource": {
              "resourceType": "Questionnaire",
              "id": "home-oxygen-therapy",
              "url": "http://example.org/fhir/Questionnaire/home-oxygen-therapy",
              "version": "2.0",
              "status": "active",
              "title": "Home Oxygen Therapy Documentation",
              "item": [
                {
                  "linkId": "1",
                  "text": "Patient diagnosis",
                  "type": "choice",
                  "answerValueSet": "http://example.org/fhir/ValueSet/oxygen-diagnoses"

                }
              ]
            }
          },
          {
            "fullUrl": "http://example.org/fhir/Library/oxygen-prepopulation",
            "resource": {
              "resourceType": "Library",
              "id": "oxygen-prepopulation",
              "url": "http://example.org/fhir/Library/oxygen-prepopulation",
              "version": "1.0",
              "type": {
                "coding": [{
                  "system": "http://terminology.hl7.org/CodeSystem/library-type",
                  "code": "logic-library"
                }]
              },
              "content": [{
                "contentType": "text/cql",
                "data": "bGlicmFyeSBPeHlnZW5QcmVwb3B1bGF0aW9u..."
              }]
            }
```

```
          },
          {
            "fullUrl": "http://example.org/fhir/ValueSet/oxygen-diagnoses",
            "resource": {
              "resourceType": "ValueSet",
              "id": "oxygen-diagnoses",
              "url": "http://example.org/fhir/ValueSet/oxygen-diagnoses",
              "status": "active",
              "expansion": {
                "timestamp": "2024-03-15T10:30:00Z",
                "contains": [
                  {
                    "system": "http://hl7.org/fhir/sid/icd-10",
                    "code": "J44.0",
                    "display": "COPD with acute lower respiratory infection"
                  },
                  {
                    "system": "http://hl7.org/fhir/sid/icd-10",
                    "code": "J96.01",
                    "display": "Acute respiratory failure with hypoxia"
                  }
                ]
              }
            }
          },
          {
            "fullUrl": "http://example.org/fhir/QuestionnaireResponse/example-
prepopulated",
            "resource": {
              "resourceType": "QuestionnaireResponse",
              "id": "example-prepopulated",
              "questionnaire": "http://example.org/fhir/Questionnaire/home-oxygen-
therapy|2.0",
              "status": "in-progress",
              "subject": {
                "reference": "Patient/example-patient"
              },
              "basedOn": [{
                "reference": "ServiceRequest/example-service-request"
              }],
              "item": [
                {
                  "linkId": "1",
                  "text": "Patient diagnosis",
```

```
                  "answer": [{
                    "valueCoding": {
                      "system": "http://hl7.org/fhir/sid/icd-10",
                      "code": "J44.0",
                      "display": "COPD with acute lower respiratory infection"
                    }
                  }]
                }
              ]
            }
          }
        ]
      }
    },
    {
      "name": "Outcome",
      "resource": {
        "resourceType": "OperationOutcome",
        "issue": [{
          "severity": "information",
          "code": "informational",
          "details": {
            "text": "Successfully retrieved questionnaire package"
          }
        }]
      }
    }
  ]
}
```

Flujo de trabajo operativo

¿Cómo HealthLake procesa su solicitud

Cuando llames`$questionnaire-package`, HealthLake realiza los siguientes pasos:

1. Identifique al paciente y al pagador: extraiga el paciente y la organización de seguros a partir de su `coverage` parámetro.
2. Encuentre el cuestionario correcto:
   - Con **`questionnaire`** el parámetro: usa la URL canónica que proporcionaste
   - Con el `order` parámetro: Coincide con el código del pedido (CPT/HCPCS/LOINC) y el pagador para encontrar el cuestionario correspondiente

3. Reunir dependencias: recupera automáticamente todo lo necesario para procesar el cuestionario:
   - Bibliotecas CQL: lógica para preguntas condicionales y previas al rellenado
   - ValueSets- Opciones de respuesta (se amplían automáticamente si hay menos de 40 opciones)
   - QuestionnaireResponse- Cualquier respuesta existente, en curso o completada
4. Package todo junto:
   - Agrupa todos los recursos (cada recurso se incluye solo una vez)
   - Filtra por `changedSince` marca de tiempo si se proporciona
   - Añade advertencias en `Outcome` caso de que falte algún recurso

Resultado: un paquete completo e independiente listo para renderizarse.

Respuestas de error

400: solicitud maligna

Se devuelve cuando la validación de la solicitud falla.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "required",
    "details": {
      "text": "At least one of 'questionnaire' or 'order' must be provided along with
 'coverage'"
    }
  }]
}
```

4.2.4 Dependencia fallida

Se devuelve cuando no se puede recuperar un recurso dependiente.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "warning",
    "code": "not-found",
    "details": {
```

```
      "text": "Referenced Library 'http://example.org/fhir/Library/missing-library'
 could not be retrieved"
    }
  }]
}
```

401 sin autorización

Se devuelve cuando faltan credenciales de autenticación o no son válidas.

403: prohibido

Se devuelve cuando el usuario autenticado no tiene permiso para acceder a los recursos solicitados.

406 No es aceptable

Se devuelve cuando no se puede proporcionar el tipo de contenido solicitado.

Conflicto, 409

Se devuelve cuando hay un conflicto de versiones o de simultaneidad.

410 Se ha ido

Se devuelve cuando el recurso solicitado se ha eliminado permanentemente.

429 Demasiadas solicitudes

Se devuelve cuando se superan los límites de velocidad.

500 Error de servidor interno

Se devuelve cuando se produce un error inesperado en el servidor.

501 No implementado

Se devuelve cuando la operación solicitada aún no está implementada.

Reglas de validación

Validación de entradas

- `coverage`el parámetro es obligatorio (1.. * cardinalidad)
- `order`Debe proporcionarse al menos uno de `questionnaire` ellos

- Todos los recursos de cobertura deben ser recursos válidos del FHIR
- Todos los recursos del pedido deben ser recursos válidos del FHIR
- Canonical URLs debe tener el formato correcto
- `changedSince`debe ser un DateTime ISO 8601 válido

QuestionnaireResponse validación

- `status`debe ser apropiado (`in-progress`,`completed`,`amended`)
- La estructura debe coincidir con el cuestionario al que se hace referencia
- `basedOn`debe hacer referencia a los recursos del pedido válidos
- `subject`debe hacer referencia a recursos válidos para pacientes

Deduplicación de recursos

- Cada recurso aparece solo una vez en el paquete
- Excepción: es posible que se incluyan diferentes versiones del mismo recurso
- Los recursos se identifican por su URL y versión canónicas

Especificaciones de rendimiento

Métrica	Especificación
Límite de recuento de recursos	500 recursos por paquete
Límite de tamaño del paquete	5 MB como máximo

Permisos necesarios

Para utilizar la `$questionnaire-package` operación, asegúrese de que su función de IAM tenga:

- `healthlake:QuestionnairePackage`- Para convocar a la operación
- `healthlake:ReadResource`- Para recuperar el cuestionario y los recursos dependientes
- `healthlake:SearchWithPost`- Para buscar recursos QuestionnaireResponse y recursos relacionados

SMART en FHIR Scopes

Alcances mínimos requeridos:

- SMART v1: `user/Questionnaire.read user/Library.read user/ValueSet.read user/QuestionnaireResponse.read`
- SMART v2: `user/Questionnaire.rs user/Library.rs user/ValueSet.rs user/QuestionnaireResponse.rs`

Notas importantes sobre la implementación

Estrategia de recuperación de recursos

Prioridad de identificación del cuestionario:

- URL canónica (si se proporciona el `questionnaire` parámetro): prioridad máxima
- Análisis de pedidos (si se proporciona `order` el parámetro):
  - Haga coincidir los códigos de pedido (CPT, HCPCS, LOINC) con las pólizas médicas del pagador
  - Use el pagador de cobertura para filtrar los cuestionarios específicos del pagador
  - Tenga en cuenta los códigos de motivo para obtener un contexto adicional

Resolución de dependencias

Bibliotecas CQL:

- Recuperado a través de la `cqf-library` extensión sobre los recursos del cuestionario
- Busca de forma recursiva las bibliotecas dependientes mediante `Library.relatedArtifact` el tipo `depends-on`
- Todas las dependencias de la biblioteca están incluidas en el paquete

ValueSets:

- Se expanden automáticamente si contienen menos de 40 conceptos
- ValueSets Los más grandes se incluyen sin expansión

- ValueSets Se incluyen los recursos referenciados tanto en el cuestionario como en la biblioteca

QuestionnaireResponse prepoblación

La operación puede devolver un QuestionnaireResponse con datos rellenados previamente cuando:

- Se encuentra una respuesta ya finalizada o en curso
- La lógica CQL de las bibliotecas asociadas puede extraer datos de los registros de los pacientes
- La respuesta está vinculada al pedido y la cobertura pertinentes

Criterios de búsqueda para QuestionnaireResponse:

Parámetro de búsqueda	Ruta FHIR	Description (Descripción)
`based-on`	`QuestionnaireResponse.basedOn`	Enlaces a ServiceRequest o CarePlan
`patient`	`QuestionnaireResponse.subject`	El paciente que es el sujeto
`questionnaire`	`QuestionnaireResponse.questionnaire`	El cuestionario que se está respondiendo

Se modificó el filtrado de recursos

Cuando se proporciona el `changedSince` parámetro:

- Solo se incluyen los recursos modificados después de la marca de tiempo especificada
- Si ningún recurso ha cambiado, regresa `200 OK` con un paquete vacío
- Útil para actualizaciones incrementales y estrategias de almacenamiento en caché
- La comparación de marcas de tiempo utiliza el campo de recursos `meta.lastUpdated`

Múltiples paquetes

La operación puede devolver varios Package Bundles cuando:

- Se solicitan varios cuestionarios a través de Canonical URLs
- Los pedidos múltiples requieren cuestionarios diferentes
- Se aplican diferentes versiones del mismo cuestionario

Cada Package Bundle es autónomo y cuenta con todas las dependencias necesarias.

Casos de uso comunes

Caso de uso 1: documentación de autorización previa

Escenario: un proveedor debe recopilar la documentación necesaria para obtener una autorización previa para la oxigenoterapia domiciliaria.

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "coverage",
      "resource": { /* Patient's insurance coverage */ }
    },
    {
      "name": "order",
      "resource": {
        "resourceType": "ServiceRequest",
        "code": {
          "coding": [{
            "system": "http://www.ama-assn.org/go/cpt",
            "code": "94660"
          }]
        }
      }
    }
  ]
}
```

Resultado: devuelve un paquete con el cuestionario de oxigenoterapia, rellenado previamente con los datos vitales del paciente y los códigos de diagnóstico del EHR.

Caso de uso 2: recupere una versión específica del cuestionario

Escenario: un proveedor necesita una versión específica de un cuestionario para cumplir con los requisitos.

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "coverage",
      "resource": { /* Coverage resource */ }
    },
    {
      "name": "questionnaire",
      "valueCanonical": "http://example.org/fhir/Questionnaire/dme-request|3.1.0"
    }
  ]
}
```

Resultado: devuelve exactamente la versión 3.1.0 del cuestionario de solicitud del DME con todas las dependencias.

Caso de uso 3: comprobar si hay actualizaciones

Escenario: un proveedor quiere comprobar si algún recurso del cuestionario se ha actualizado desde la última consulta.

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "coverage",
      "resource": { /* Coverage resource */ }
    },
    {
      "name": "questionnaire",
      "valueCanonical": "http://example.org/fhir/Questionnaire/medication-request"
    },
    {
      "name": "changedSince",
      "valueDateTime": "2024-03-01T00:00:00Z"
    }
  ]
}
```

Resultado: devuelve solo los recursos que se hayan modificado después del 1 de marzo de 2024 o un paquete vacío si no ha cambiado nada.

Caso de uso 4: pedidos múltiples

Escenario: un proveedor envía varias solicitudes de servicio que pueden requerir cuestionarios diferentes.

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "coverage",
      "resource": { /* Coverage resource */ }
    },
    {
      "name": "order",
      "resource": { /* ServiceRequest for imaging */ }
    },
    {
      "name": "order",
      "resource": { /* ServiceRequest for DME */ }
    }
  ]
}
```

Resultado: devuelve varios Package Bundles, uno para cada cuestionario aplicable.

Integración con otros Da Vinci IGs

Descubrimiento de requisitos de cobertura (CRD)

Integración del flujo de trabajo:

- El proveedor solicita un servicio en su registro electrónico electrónico
- El CRD Hook se dispara, comprobando los requisitos de cobertura
- El pagador responde indicando que necesita documentación
- El proveedor llama `$questionnaire-package` para recuperar el formulario de documentación
- El proveedor completa el cuestionario
- La documentación se envía a través del PAS o CDex

Soporte de autorización previa (PAS)

Integración del flujo de trabajo:

- Se utiliza `$questionnaire-package` para recuperar los requisitos de documentación
- Complete el formulario QuestionnaireResponse con los datos clínicos requeridos
- Envíe la autorización previa `Claim/$submit` junto con la cumplimentada QuestionnaireResponse
- Compruebe el estado utilizando `Claim/$inquire`

Intercambio de datos clínicos (CDex)

Integración del flujo de trabajo:

- El pagador solicita documentación adicional para una reclamación
- El proveedor utiliza `$questionnaire-package` para recuperar el formulario de recopilación de datos estructurados
- El proveedor completa el QuestionnaireResponse
- La documentación se envía al pagador a través del flujo de trabajo CDex adjunto

Guía para solucionar problemas

Problema: No se devolvió el cuestionario

Posibles causas:

- La URL canónica no coincide con ningún cuestionario del almacén de datos
- El código del pedido no se corresponde con ningún cuestionario de la política médica del pagador
- El pagador de la cobertura no tiene cuestionarios asociados

Soluciones:

- Verifica que la URL canónica sea correcta y que el cuestionario exista
- Comprueba que los códigos de pedido (CPT/HCPCS) estén correctamente especificados
- Confirme que la organización pagadora tenga configurados los cuestionarios

Problema: Faltan dependencias en el paquete

Posibles causas:

- Biblioteca a la que se hace referencia o ValueSet no existe en el almacén de datos
- Las referencias a la biblioteca están rotas o son incorrectas
- ValueSet la expansión ha fallado

Soluciones:

- Compruebe el `Outcome` parámetro para ver si hay advertencias sobre la falta de recursos
- Compruebe que todos los recursos a los que se hace referencia existan en el banco de datos
- Asegúrese de que ValueSet URLs son correctos y se pueden resolver

Problema: Empty Package con ChangedSince

Posibles causas:

- Este es el comportamiento esperado: no se ha modificado ningún recurso desde la marca de tiempo especificada

Soluciones:

- Utilice la versión en caché del paquete
- Elimine `changedSince` el parámetro para recuperar el paquete completo

Problema: QuestionnaireResponse no se ha rellenado previamente

Posibles causas:

- No se QuestionnaireResponse encontró ningún elemento existente
- La lógica de la biblioteca CQL no pudo extraer los datos necesarios
- Faltan datos del paciente o están incompletos

Soluciones:

- Esto es de esperar: no todos los cuestionarios tienen una lógica previa a la población
- Compruebe que los datos del paciente existan en el almacén de datos
- Revise la lógica de la biblioteca CQL para conocer los requisitos de extracción de datos

Prácticas recomendadas

1. Utilice Canonical URLs con las versiones

Especifique siempre los números de versión cuando solicite cuestionarios específicos:

```
{
  "name": "questionnaire",
  "valueCanonical": "http://example.org/fhir/Questionnaire/dme|2.1.0"
}
```

Por qué: Garantiza la coherencia y evita cambios inesperados cuando se actualizan los cuestionarios.

2. Aproveche los cambios desde entonces para mejorar el rendimiento

Para los cuestionarios a los que se accede con frecuencia, utilícelos `changedSince` para minimizar la transferencia de datos:

```
{
  "name": "changedSince",
  "valueDateTime": "2024-03-10T15:30:00Z"
}
```

Por qué: reduce la latencia y el uso de ancho de banda al recuperar solo los recursos actualizados.

3. Incluya información completa sobre la cobertura

Proporcione detalles de cobertura completos para garantizar la correcta selección del cuestionario:

```
{
  "name": "coverage",
  "resource": {
    "resourceType": "Coverage",
    "beneficiary": { "reference": "Patient/123" },
```

```
    "payor": [{ "reference": "Organization/payer-abc" }],
    "class": [{ /* Group/plan information */ }]
  }
}
```

Por qué: Ayuda a HealthLake identificar los cuestionarios y requisitos específicos del pagador.

Operaciones de relacionadas

- `Claim/$submit`- Presente las solicitudes de autorización previa con la documentación completa
- `Claim/$inquire`- Verificar el estado de las autorizaciones previas presentadas
- `ValueSet/$expand`- Ampliar ValueSets para ver las opciones de respuesta

`$submit`Operación FHIR para HealthLake

La `$submit` operación le permite enviar electrónicamente solicitudes de autorización previa a los pagadores para su aprobación. Esta operación implementa la Guía de implementación de Da Vinci Prior Authorization Support (PAS), que proporciona un flujo de trabajo estandarizado basado en el FHIR para la presentación de autorizaciones previas.

Funcionamiento

- Enviar: envía un paquete FHIR que contiene su solicitud de autorización previa y los datos clínicos de respaldo
- Validar: HealthLake valida la presentación según los requisitos del PAS
- Persistir: todos los recursos se almacenan en su almacén HealthLake de datos
- Responder: recibirá una respuesta inmediata con el estado de «en cola»
- Proceso: el pagador procesa la decisión de autorización de forma asíncrona

Punto de conexión de la API

```
POST /datastore/{datastoreId}/r4/Claim/$submit
Content-Type: application/fhir+json
```

Estructura de la solicitud

Requisitos del paquete

Su solicitud debe ser un recurso del paquete FHIR con:

- Tipo de paquete: debe ser `"collection"`
- Bundle.entry: debe contener exactamente un recurso de reclamación con `use = "preauthorization"`
- Recursos de referencia: todos los recursos a los que se hace referencia en la reclamación deben incluirse en el paquete

Recursos necesarios de

Recurso	Cardinalidad	Perfil	Description (Descripción)
Reclamación	1	Reclamación PAS	La solicitud de autorización previa
Paciente	1	Paciente PAS	Información demográfica del paciente
Organización (aseguradora)	1	Aseguradora PAS	Compañía de seguros
Organización (proveedor)	1	Solicitante del PAS	Proveedor de atención médica que presenta la solicitud
Cobertura	1 o más	Cobertura PAS	Detalles de la cobertura del seguro

Recursos opcionales

Recurso	Cardinalidad	Perfil	Description (Descripción)
Práctico	0 o más	Profesional de PAS	Profesionales de la salud
Practitio nerRole	0 o más	PAS PractitionerRole	Funciones de los profesionales
ServiceRe quest	0 o más	PAS ServiceRequest	Servicios médicos solicitados
DeviceRequest	0 o más	PAS DeviceRequest	Dispositivos médicos solicitados

Recurso	Cardinalidad	Perfil	Description (Descripción)
MedicationRequest	0 o más	PAS MedicationRequest	Medicamentos solicitados
DocumentReference	0 o más	PAS DocumentReference	Documentación clínica de apoyo

Ejemplo de solicitud

```
POST /datastore/example-datastore/r4/Claim/$submit
Content-Type: application/fhir+json
Authorization: Bearer <your-token>

{
  "resourceType" : "Bundle",
  "id" : "MedicalServicesAuthorizationBundleExample",
  "meta" : {
    "profile" : ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-pas-
request-bundle"]
  },
  "identifier" : {
    "system" : "http://example.org/SUBMITTER_TRANSACTION_IDENTIFIER",
    "value" : "5269367"
  },
  "type" : "collection",
  "timestamp" : "2005-05-02T11:01:00+05:00",
  "entry" : [{
    "fullUrl" : "http://example.org/fhir/Claim/MedicalServicesAuthorizationExample",
    "resource" : {
      "resourceType" : "Claim",
      "id" : "MedicalServicesAuthorizationExample",
      "meta" : {
        "profile" : ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
claim"]
      },
      "identifier" : [{
        "system" : "http://example.org/PATIENT_EVENT_TRACE_NUMBER",
        "value" : "111099"
      }],
      "status" : "active",
```

```
      "type" : {
        "coding" : [{
          "system" : "http://terminology.hl7.org/CodeSystem/claim-type",
          "code" : "professional"
        }]
      },
      "use" : "preauthorization",
      "patient" : {
        "reference" : "Patient/SubscriberExample"
      },
      "created" : "2005-05-02T11:01:00+05:00",
      "insurer" : {
        "reference" : "Organization/InsurerExample"
      },
      "provider" : {
        "reference" : "Organization/UMOExample"
      },
      "priority" : {
        "coding" : [{
          "system" : "http://terminology.hl7.org/CodeSystem/processpriority",
          "code" : "normal"
        }]
      },
      "insurance" : [{
        "sequence" : 1,
        "focal" : true,
        "coverage" : {
          "reference" : "Coverage/InsuranceExample"
        }
      }],
      "item" : [{
        "extension" : [{
          "url" : "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/extension-
serviceItemRequestType",
          "valueCodeableConcept" : {
            "coding" : [{
              "system" : "https://codesystem.x12.org/005010/1525",
              "code" : "IN",
              "display" : "Initial Medical Services Reservation"
            }]
          }
        },
        {
```

```
          "url" : "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/extension-
certificationType",
          "valueCodeableConcept" : {
            "coding" : [{
              "system" : "https://codesystem.x12.org/005010/1322",
              "code" : "I",
              "display" : "Initial"
            }]
          }
        },
        {
          "url" : "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/extension-
authorizationNumber",
          "valueString" : "1122344"
        },
        {
          "url" : "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/extension-
administrationReferenceNumber",
          "valueString" : "33441122"
        }],
        "sequence" : 1,
        "category" : {
          "coding" : [{
            "system" : "https://codesystem.x12.org/005010/1365",
            "code" : "1",
            "display" : "Medical Care"
          }]
        },
        "productOrService" : {
          "coding" : [{
            "system" : "http://www.cms.gov/Medicare/Coding/HCPCSReleaseCodeSets",
            "code" : "99212",
            "display" : "Established Office Visit"
          }]
        },
        "servicedDate" : "2005-05-10",
        "locationCodeableConcept" : {
          "coding" : [{
            "system" : "https://www.cms.gov/Medicare/Coding/place-of-service-codes/
Place_of_Service_Code_Set",
            "code" : "11"
          }]
        }
      }]
```

```
    }
  },
  {
    "fullUrl" : "http://example.org/fhir/Organization/UMOExample",
    "resource" : {
      "resourceType" : "Organization",
      "id" : "UMOExample",
      "meta" : {
        "profile" : ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
requestor"]
      },
      "identifier" : [{
        "system" : "http://hl7.org/fhir/sid/us-npi",
        "value" : "8189991234"
      }],
      "active" : true,
      "type" : [{
        "coding" : [{
          "system" : "https://codesystem.x12.org/005010/98",
          "code" : "X3"
        }]
      }],
      "name" : "DR. JOE SMITH CORPORATION",
      "address" : [{
        "line" : ["111 1ST STREET"],
        "city" : "SAN DIEGO",
        "state" : "CA",
        "postalCode" : "92101",
        "country" : "US"
      }]
    }
  },
  {
    "fullUrl" : "http://example.org/fhir/Organization/InsurerExample",
    "resource" : {
      "resourceType" : "Organization",
      "id" : "InsurerExample",
      "meta" : {
        "profile" : ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
insurer"]
      },
      "identifier" : [{
        "system" : "http://hl7.org/fhir/sid/us-npi",
        "value" : "1234567893"
```

```
      }],
      "active" : true,
      "type" : [{
        "coding" : [{
          "system" : "https://codesystem.x12.org/005010/98",
          "code" : "PR"
        }]
      }],
      "name" : "MARYLAND CAPITAL INSURANCE COMPANY"
    }
  },
  {
    "fullUrl" : "http://example.org/fhir/Coverage/InsuranceExample",
    "resource" : {
      "resourceType" : "Coverage",
      "id" : "InsuranceExample",
      "meta" : {
        "profile" : ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
coverage"]
      },
      "status" : "active",
      "subscriberId" : "1122334455",
      "beneficiary" : {
        "reference" : "Patient/SubscriberExample"
      },
      "relationship" : {
        "coding" : [{
          "system" : "http://terminology.hl7.org/CodeSystem/subscriber-relationship",
          "code" : "self"
        },
        {
          "system" : "https://codesystem.x12.org/005010/1069",
          "code" : "18"
        }]
      },
      "payor" : [{
        "reference" : "Organization/InsurerExample"
      }]
    }
  },
  {
    "fullUrl" : "http://example.org/fhir/Patient/SubscriberExample",
    "resource" : {
      "resourceType" : "Patient",
```

```
      "id" : "SubscriberExample",
      "meta" : {
        "profile" : ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
subscriber"]
      },
      "extension" : [{
        "url" : "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/extension-
militaryStatus",
        "valueCodeableConcept" : {
          "coding" : [{
            "system" : "https://codesystem.x12.org/005010/584",
            "code" : "RU"
          }]
        }
      }],
      "identifier" : [{
        "type" : {
          "coding" : [{
            "system" : "http://terminology.hl7.org/CodeSystem/v2-0203",
            "code" : "MB"
          }]
        },
        "system" : "http://example.org/MIN",
        "value" : "12345678901"
      }],
      "name" : [{
        "family" : "SMITH",
        "given" : ["JOE"]
      }],
      "gender" : "male"
    }
  }]
}
```

Formato de las respuestas

Respuesta satisfactoria (200 OK)

Recibirá un paquete de respuesta del PAS que contiene:

- ClaimResponsecon `outcome: "queued"` y `status: "active"`
- Todos los recursos originales de su solicitud
- Marca de tiempo que confirma la recepción

```
{
  "resourceType" : "Bundle",
  "identifier": {
        "system": "http://example.org/SUBMITTER_TRANSACTION_IDENTIFIER",
        "value": "5269367"
  },
  "type" : "collection",
  "timestamp" : "2005-05-02T11:02:00+05:00",
  "entry" : [{
    "fullUrl" : "http://example.org/fhir/ClaimResponse/
PractitionerRequestorPendingResponseExample",
    "resource" : {
      "resourceType" : "ClaimResponse",
      "id" : "PractitionerRequestorPendingResponseExample",
      "meta" : {
        "profile" : ["http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
claimresponse"]
      },
      "identifier" : [{
        "system" : "http://example.org/PATIENT_EVENT_TRACE_NUMBER",
        "value" : "111099"
      }],
      "status" : "active",
      "type" : {
        "coding" : [{
          "system" : "http://terminology.hl7.org/CodeSystem/claim-type",
          "code" : "professional"
        }]
      },
      "use" : "preauthorization",
      "patient" : {
        "reference" : "Patient/SubscriberExample"
      },
      "created" : "2005-05-02T11:02:00+05:00",
      "insurer" : {
        "reference" : "Organization/InsurerExample"
      },
      "requestor" : {
        "reference" : "PractitionerRole/ReferralPractitionerRoleExample"
      },
      "request" : {
        "reference" : "Claim/MedicalServicesAuthorizationExample"
      },
```

```
      "outcome" : "queued"
    }
  },
  {
    "fullUrl" : "http://example.org/fhir/Claim/MedicalServicesAuthorizationExample",
    "resource" : {
      "resourceType" : "Claim",
      "id" : "MedicalServicesAuthorizationExample",
      "meta" : {
        "profile": [
            "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-claim",
            "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-claim|
2.1.0"
        ]
      },
      "identifier" : [{
        "system" : "http://example.org/PATIENT_EVENT_TRACE_NUMBER",
        "value" : "111099"
        }
      }],
      "status" : "active",
      "type" : {
        "coding" : [{
          "system" : "http://terminology.hl7.org/CodeSystem/claim-type",
          "code" : "professional"
        }]
      },
      "use" : "preauthorization",
      "patient" : {
        "reference" : "Patient/SubscriberExample"
      },
      "created" : "2005-05-02T11:01:00+05:00",
      "insurer" : {
        "reference" : "Organization/InsurerExample"
      },
      "provider" : {
        "reference" : "Organization/UMOExample"
      },
      "priority" : {
        "coding" : [{
          "system" : "http://terminology.hl7.org/CodeSystem/processpriority",
          "code" : "normal"
        }]
      },
```

```
      "insurance" : [{
        "sequence" : 1,
        "focal" : true,
        "coverage" : {
          "reference" : "Coverage/InsuranceExample"
        }
      }],
      "item" : [{
        "extension" : [{
          "url" : "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/extension-
serviceItemRequestType",
          "valueCodeableConcept" : {
            "coding" : [{
              "system" : "https://codesystem.x12.org/005010/1525",
              "code" : "IN",
              "display" : "Initial Medical Services Reservation"
            }]
          }
        },
        {
          "url" : "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/extension-
certificationType",
          "valueCodeableConcept" : {
            "coding" : [{
              "system" : "https://codesystem.x12.org/005010/1322",
              "code" : "I",
              "display" : "Initial"
            }]
          }
        },
        {
          "url" : "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/extension-
authorizationNumber",
          "valueString" : "1122344"
        },
        {
          "url" : "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/extension-
administrationReferenceNumber",
          "valueString" : "33441122"
        }],
        "sequence" : 1,
        "category" : {
          "coding" : [{
            "system" : "https://codesystem.x12.org/005010/1365",
```

```
            "code" : "1",
            "display" : "Medical Care"
          }]
        },
        "productOrService" : {
          "coding" : [{
            "system" : "http://www.cms.gov/Medicare/Coding/HCPCSReleaseCodeSets",
            "code" : "99212",
            "display" : "Established Office Visit"
          }]
        },
        "servicedDate" : "2005-05-10",
        "locationCodeableConcept" : {
          "coding" : [{
            "system" : "https://www.cms.gov/Medicare/Coding/place-of-service-codes/
Place_of_Service_Code_Set",
            "code" : "11"
          }]
        }
      }]
    }
  },
  {
    "fullUrl" : "http://example.org/fhir/Organization/UMOExample",
    "resource" : {
      "resourceType" : "Organization",
      "id" : "UMOExample",
      "meta" : {
        "profile": [
            "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-requestor",

            "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-requestor|
2.1.0"
        ]
      },
      "identifier" : [{
        "system" : "http://hl7.org/fhir/sid/us-npi",
        "value" : "8189991234"
      }],
      "active" : true,
      "type" : [{
        "coding" : [{
          "system" : "https://codesystem.x12.org/005010/98",
          "code" : "X3"
```

```
        }]
      }],
      "name" : "DR. JOE SMITH CORPORATION",
      "address" : [{
        "line" : ["111 1ST STREET"],
        "city" : "SAN DIEGO",
        "state" : "CA",
        "postalCode" : "92101",
        "country" : "US"
      }]
    }
  },
  {
    "fullUrl" : "http://example.org/fhir/Organization/InsurerExample",
    "resource" : {
      "resourceType" : "Organization",
      "id" : "InsurerExample",
      "meta" : {
           "profile": [
                "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
insurer",
                "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
insurer|2.1.0"
            ]
      },
      "identifier" : [{
        "system" : "http://hl7.org/fhir/sid/us-npi",
        "value" : "1234567893"
      }],
      "active" : true,
      "type" : [{
        "coding" : [{
          "system" : "https://codesystem.x12.org/005010/98",
          "code" : "PR"
        }]
      }],
      "name" : "MARYLAND CAPITAL INSURANCE COMPANY"
    }
  },
  {
    "fullUrl" : "http://example.org/fhir/Coverage/InsuranceExample",
    "resource" : {
      "resourceType" : "Coverage",
      "id" : "InsuranceExample",
```

```
      "meta": {
            "profile": [
                "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
coverage",
                "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
coverage|2.1.0"
            ]
        },
      "status" : "active",
      "subscriberId" : "1122334455",
      "beneficiary" : {
        "reference" : "Patient/SubscriberExample"
      },
      "relationship" : {
        "coding" : [{
          "system" : "http://terminology.hl7.org/CodeSystem/subscriber-relationship",
          "code" : "self"
        },
        {
          "system" : "https://codesystem.x12.org/005010/1069",
          "code" : "18"
        }]
      },
      "payor" : [{
        "reference" : "Organization/InsurerExample"
      }]
    }
  },
  {
    "fullUrl" : "http://example.org/fhir/Patient/SubscriberExample",
    "resource" : {
      "resourceType" : "Patient",
      "id" : "SubscriberExample",
      "meta": {
            "profile": [
                "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
subscriber",
                "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
beneficiary",
                "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/profile-
beneficiary|2.1.0"
            ]
        },
      "extension" : [{
```

```
        "url" : "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/extension-
militaryStatus",
        "valueCodeableConcept" : {
          "coding" : [{
            "system" : "https://codesystem.x12.org/005010/584",
            "code" : "RU"
          }]
        }
      }],
      "identifier" : [{
        "type" : {
          "coding" : [{
            "system" : "http://terminology.hl7.org/CodeSystem/v2-0203",
            "code" : "MB"
          }]
        },
        "system" : "http://example.org/MIN",
        "value" : "12345678901"
      }],
      "name" : [{
        "family" : "SMITH",
        "given" : ["JOE"]
      }],
      "gender" : "male"
    }
  }]
}
```

Respuestas de error

400: solicitud maligna

Se devuelve cuando el formato de la solicitud no es válido o está mal formado.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "invalid",
    "diagnostics": "The provided payload was invalid and could not be parsed
 correctly."
  }]
}
```

Condición previa con error, 412

Se devuelve cuando ya se ha presentado la misma solicitud de autorización previa (se ha detectado un envío duplicado).

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "processing",
    "diagnostics": "PreAuth Claim already exists"
  }]
}
```

Idempotencia

La `$submit` operación es idempotente. Si se envía la misma solicitud varias veces, no se duplicarán las solicitudes de autorización previa. En su lugar, recibirás un mensaje de error 412 que te indicará que lo utilices `$inquire` para comprobar el estado del envío original.

4.2.2 Entidad inprocesable

Se devuelve cuando la validación del FHIR falla.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "required",
    "diagnostics": "Bundle contains more than one preauthorization claim"
  }]
}
```

Reglas de validación

HealthLake realiza una validación exhaustiva de su envío:

Validación del paquete

- Debe ajustarse al perfil del paquete de solicitud de PAS

- `Bundle.type`debe ser `"collection"`
- Puede contener varios recursos de reclamación
- Sin embargo, debe contener exactamente un recurso de reclamación con uso previo a la autorización
  - Y este recurso de reclamación debe ser la primera entrada del paquete
- Todos los recursos a los que se hace referencia deben incluirse en el paquete

Validación de la reclamación

- Debe ajustarse al perfil de reclamación del PAS
- `Claim.use`debe ser `"preauthorization"`
- Campos obligatorios:`patient`,`insurer`,`provider`,`created`, `priority`
- Los identificadores comerciales deben estar presentes y ser válidos

Validación de recursos

- Todos los recursos deben ajustarse a sus respectivos perfiles de PAS
- Deben estar presentes los recursos de apoyo necesarios (paciente, cobertura, organización)
- Las referencias cruzadas deben ser válidas y poder resolverse dentro del paquete

Especificaciones de rendimiento

Métrica	Especificación
Límite de tamaño del paquete	5 MB como máximo
Límite de recuento de recursos	500 recursos por paquete

Permisos necesarios

Para usar la `$submit` operación, se puede usar AWS Sigv4 o SMART en FHIR:

- Asegúrese de que su función de IAM tenga: `healthlake:SubmitPreAuthClaim` - Llamar a la operación

SMART en los osciloscopios FHIR

Alcances mínimos requeridos:

- SMART v1: `user/Claim.write & <all_resourceTypes_in_Bundle>.write`
- SMART v2: `user/Claim.c & <all_resourceTypes_in_Bundle>.c or system/*.*`

Notas importantes sobre la implementación

Persistencia de los recursos

- Todas las entradas del paquete se conservan como recursos FHIR individuales en su almacén de datos
- Las suministradas por el cliente se conservan cuando se proporcionan IDs
- El historial de versiones se mantiene con fines de auditoría
- La detección de duplicados evita conflictos de recursos

Comportamiento de procesamiento

- Cada envío válido da como resultado exactamente uno ClaimResponse con `"queued"` resultado
- Los envíos no válidos devuelven 400 o 422 códigos de estado con información de error detallada
- Los errores del sistema devuelven los códigos de estado 5xx correspondientes
- Todos los envíos aprobados devuelven el estado 200 con un estado pendiente ClaimResponse

Requisitos del paquete

- `Bundle.entry.fullUrl`los valores deben estar en formato REST URLs o en `"urn:uuid:[guid]"` formato
- Todos los envíos GUIDs deben ser únicos (excepto las mismas instancias de recursos)
- Los recursos a los que se hace referencia deben existir en el paquete o ser resolubles

Operaciones de relacionadas

- `Claim/$inquire`- Consulta el estado de una solicitud de autorización previa presentada
- `Patient/$everything`- Recupere datos completos de los pacientes para utilizarlos en el contexto de la autorización previa

Validación de los recursos del FHIR con **`$validate`**

AWS HealthLake ahora admite la `$validate` operación con los recursos del FHIR, lo que le permite validar un recurso según la especificación del FHIR y comprobar su conformidad con un perfil específico o una definición de recurso base sin realizar ninguna operación de almacenamiento. Esta operación resulta especialmente útil cuando se necesita:

- Validar los requisitos de conformidad con el CMS del FHIR
- Pruebe los recursos antes de usarlos en producción
- Proporcione comentarios de validación en tiempo real a medida que los usuarios editan los datos clínicos
- Reduzca el envío de datos no válidos para crear y actualizar APIs

De uso

La `$validate` operación se puede invocar en los recursos del FHIR mediante los métodos POST:

Operaciones admitidas

```
POST [base]/[type]/[id]/$validate
POST [base]/[type]/$validate
```

Cargas útiles compatibles

Recurso de parámetros

HealthLake admite los siguientes `$validate` parámetros del FHIR:

Parámetro	Tipo	Obligatorio	Description (Descripción)
`resource`	Recurso	Sí	El recurso que se va a validar
`profile`	canonical	No	URL canónica del perfil con el que se va a realizar la validación
`mode`	code	No	Modo de validación:`create`, o `update`

Recurso directo con parámetros de consulta

Parámetro	Tipo	Obligatorio	Description (Descripción)
`profile`	canonical	No	URL canónica del perfil con el que realizar la validación
`mode`	code	No	Modo de validación:`create`, o `update`

Ejemplos

Solicitud POST de recurso con carga útil de ID y parámetros

```
POST [base]/Patient/example-patient/$validate
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "resource",
      "resource": {
        "resourceType": "Patient",
        "id": "example-patient",
        "name": [
          {
            "family": "Smith",
            "given": ["John"]
          }
        ],
        "gender": "male",
        "birthDate": "1990-01-01"
      }
    },
    {
      "name": "profile",
      "valueCanonical": "http://hl7.org/fhir/us/core/StructureDefinition/us-core-
patient"
    },
    {
```

```
      "name": "mode",
      "valueString": "create"
    }
  ]
}
```

Solicitud POST de carga útil sobre el tipo de recurso y los parámetros

```
POST [base]/Patient/$validate
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "resource",
      "resource": {
        "resourceType": "Patient",
        "name": [
          {
            "family": "Doe",
            "given": ["Jane"]
          }
        ],
        "gender": "female",
        "birthDate": "1985-05-15"
      }
    },
    {
      "name": "profile",
      "valueCanonical": "http://hl7.org/fhir/us/core/StructureDefinition/us-core-
patient"
    },
    {
      "name": "mode",
      "valueString": "update"
    }
  ]
}
```

Solicitud POST de recurso con ID y carga útil de recurso directa

```
POST [base]/Patient/example-patient/$validate?profile=http://hl7.org/fhir/us/core/
StructureDefinition/us-core-patient&mode=create
Content-Type: application/fhir+json

{
    "resourceType": "Patient",
    "id": "example-patient",
    "name": [
        {
        "family": "Smith",
        "given": ["John"]
        }
    ],
    "gender": "male",
    "birthDate": "1990-01-01"
}
```

Solicitud POST de tipo de recurso y carga útil directa del recurso

```
POST [base]/Patient/$validate?profile=http://hl7.org/fhir/us/core/StructureDefinition/
us-core-patient&mode=create
Content-Type: application/fhir+json

{
    "resourceType": "Patient",
    "id": "example-patient",
    "name": [
        {
        "family": "Smith",
        "given": ["John"]
        }
    ],
    "gender": "male",
    "birthDate": "1990-01-01"
}
```

Respuesta de ejemplo

La operación devuelve un OperationOutcome recurso con los resultados de la validación:

```
{
  "resourceType": "OperationOutcome",
```

```
  "issue": [
    {
      "severity": "information",
      "code": "informational",
      "diagnostics": "Validation successful"
    }
  ]
}
```

Ejemplo de respuesta con errores de validación

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "required",
      "details": {
        "text": "Missing required element"
      },
      "diagnostics": "Patient.identifier is required by the US Core Patient profile",
      "location": [
        "Patient.identifier"
      ]
    },
    {
      "severity": "warning",
      "code": "code-invalid",
      "details": {
        "text": "Invalid code value"
      },
      "diagnostics": "The provided gender code is not from the required value set",
      "location": [
        "Patient.gender"
      ]
    }
  ]
}
```

Comportamiento

La `$validate` operación:

1. Valida el recurso según la especificación del FHIR y la definición del recurso base
2. Comprueba la conformidad con los perfiles especificados cuando se proporciona el parámetro `profile`
3. Valida en función del modo (`createo`) especificado `update`
4. Devuelve los resultados de la validación detallados, incluidos los errores, las advertencias y los mensajes informativos
5. No realiza ninguna operación de almacenamiento, solo de validación
6. Devuelve HTTP 200 OK cuando se puede realizar la validación, independientemente de si se han detectado problemas de validación

Modos de validación

- crear: valida el recurso como si se estuviera creando (recurso nuevo)
- actualizar: valida el recurso como si se estuviera actualizando (recurso existente)

Gestión de errores

La operación devuelve:

- 200 OK: La validación se realizó correctamente (independientemente del resultado de la validación)
- 400 Solicitud errónea: el formato o los parámetros de la solicitud no son válidos
- 404 No se encontró: no se encontró el tipo de recurso o perfil

Para obtener más información sobre la especificación de la `$validate` operación, consulte la documentación de recursos `$validate` del FHIR R4.

Referencia de cumplimiento para AWS HealthLake

AWS HealthLake proporciona funciones diseñadas para ayudarlo a rastrear y reportar el uso de las API de acuerdo con los requisitos de interoperabilidad de los CMS (Centros de Servicios de Medicare y Medicaid). Estas funciones le permiten clasificar las transacciones de la API según las categorías exigidas por el CMS y recopilar automáticamente las métricas de uso para elaborar informes de conformidad.

⚠ Comprenda sus responsabilidades de cumplimiento

El uso AWS HealthLake y sus puntos finales de interoperabilidad del CMS no son suficientes por sí solos para lograr la conformidad con los CMS. Su responsabilidad incluye:

- Asigne correctamente los flujos de trabajo de sus API a los puntos finales de la categoría de CMS correspondiente en función de sus casos de uso específicos y sus obligaciones reglamentarias
- Implementar los controles de autenticación y autorización adecuados que cumplan con los requisitos de los CMS
- Garantizar que sus recursos e intercambios de datos del FHIR cumplan con las normas y guías de implementación aplicables de la CMS
- Configurar y monitorear las CloudWatch métricas para respaldar sus necesidades de informes de cumplimiento
- Comprenda qué reglas de CMS se aplican a su organización e implemente los controles técnicos y operativos adecuados

AWS HealthLake proporciona la infraestructura y las herramientas necesarias para respaldar sus esfuerzos de cumplimiento, pero debe utilizar estas funciones de forma adecuada en función de sus requisitos reglamentarios específicos. El simple hecho de dirigir las llamadas a la API a través de estos puntos finales no hace que su aplicación cumpla automáticamente con las normas de los CMS.

Temas

- Características de conformidad con los CMS

Características de conformidad con los CMS

AWS HealthLake ofrece funciones que le ayudan a cumplir los requisitos de interoperabilidad y conformidad de los CMS (Centros de Servicios de Medicare y Medicaid). Estas funciones le permiten realizar un seguimiento del uso de las API por categoría de CMS y, posteriormente, elaborar informes sobre las métricas de uso con fines de cumplimiento.

Temas

- Puntos finales de interoperabilidad de los CMS
- CloudWatch Métricas mejoradas para el cumplimiento de los CMS

Puntos finales de interoperabilidad de los CMS

Descripción general de

HealthLake proporciona cuatro puntos finales de interoperabilidad del CMS que corresponden a las categorías de API exigidas por el CMS. La URL base subyacente de su almacén de HealthLake datos no cambia. Estos puntos de enlace simplemente proporcionan una forma de categorizar y realizar un seguimiento de las llamadas a la API con el fin de generar informes sobre los CMS.

Finalidad

El objetivo principal de estos puntos finales de interoperabilidad es permitir a los clientes:

- Realice un seguimiento sencillo de las transacciones de API por categoría de CMS
- Informe automáticamente las métricas de uso para garantizar el cumplimiento de los CMS
- Mantenga los flujos de trabajo del FHIR existentes con cambios mínimos

Todas las llamadas a la API funcionan de forma idéntica tanto si se utiliza el punto final de interoperabilidad como el punto final estándar del FHIR; la única diferencia es la forma en que se clasifican las transacciones en las métricas. CloudWatch

Puntos finales de interoperabilidad de CMS compatibles

Categoría CMS	Punto final de interoper abilidad	Ejemplo de uso
Acceso de pacientes	`/patientaccess/v2/r4`	`baseURL/patientaccess/v2/r4/Patient/123`
Acceso a proveedores	`/provideraccess/v2/r4`	`baseURL/provideraccess/v2/r4/Observation?patient=123`
De pagador a pagador	`/payertopayerdx/v2/r4`	`baseURL/payertopayerdx/v2/r4/Practitioner/456`
Servicio de autentica ción previa	`/priorauthservice/v2/r4`	`baseURL/priorauthservice/v2/r4/ExplanationOfBenefit?patient=789`

Cómo funcionan los puntos finales de interoperabilidad

Llamada a la HealthLake API estándar:

```
baseURL/resourceType/[id]
baseURL/resourceType?[parameters]
```

Con CMS Interoperability Endpoint:

```
baseURL/interoperability-endpoint/resourceType/[id]
baseURL/interoperability-endpoint/resourceType?[parameters]
```

La ruta del punto final de interoperabilidad simplemente se inserta entre la URL base y el tipo de recurso. Todo lo que sigue a la ruta del punto final de interoperabilidad permanece exactamente igual que las llamadas a la API actuales.

Ejemplos de uso

Ejemplo 1: API de acceso para pacientes

Llamada a la API actual (aún funciona):

```
curl -X GET \
  https://healthlake.us-east-1.amazonaws.com/datastore/abc123/r4/Patient/123 \
  -H "Authorization: Bearer <token>" \
  -H "Content-Type: application/fhir+json"
```

Con el punto final de interoperabilidad de Patient Access (para el seguimiento de los CMS):

```
curl -X GET \
  https://healthlake.us-east-1.amazonaws.com/datastore/abc123/patientaccess/v2/r4/
Patient/123 \
  -H "Authorization: Bearer <token>" \
  -H "Content-Type: application/fhir+json"
```

Puntos clave:

- La URL base sigue siendo: `https://healthlake.us-east-1.amazonaws.com/datastore/abc123`
- Se ha insertado el punto final de interoperabilidad: `/patientaccess/v2/r4`

- La ruta de recursos no ha cambiado: `/Patient/123`
- Ambas llamadas devuelven respuestas idénticas
- La llamada al punto final de interoperabilidad se rastrea automáticamente en `URIType=patient-access` CloudWatch
- Las operaciones POST, PUT, PATCH y DELETE funcionan de forma idéntica.
- El cuerpo de la solicitud permanece inalterado.

Referencia de traducción del punto final

Punto final de interoperabilidad	Se traduce como	Categoría CMS
`baseURL/patientaccess/v2/r4/Patient`	`baseURL/r4/Patient`	Acceso para pacientes
`baseURL/provideraccess/v2/r4/Observation`	`baseURL/r4/Observation`	Acceso a proveedores
`baseURL/payertopayerdx/v2/r4/Practitioner/456`	`baseURL/r4/Practitioner/456`	Data Exchange de pagador a pagador
`baseURL/priorauthservice/v2/r4/ExplanationOfBenefit?patient=789`	`baseURL/r4/ExplanationOfBenefit?patient=789`	Autorización previa

Notas importantes

- Sin diferencia funcional: los puntos finales de interoperabilidad y los puntos finales estándar del FHIR arrojan respuestas idénticas y admiten operaciones idénticas
- URL base sin cambios: el punto final de su almacén de HealthLake datos sigue siendo el mismo
- Integración sencilla: inserte la ruta del punto final de interoperabilidad entre la URL base y el tipo de recurso
- Seguimiento automático: CloudWatch las métricas clasifican automáticamente las llamadas según el punto final de interoperabilidad utilizado

- Compatible con versiones anteriores: las llamadas a la API existentes sin puntos de enlace de interoperabilidad siguen funcionando con normalidad

CloudWatch Métricas mejoradas para el cumplimiento de los CMS

Descripción general de

Cuando utiliza los puntos finales de interoperabilidad del CMS, emite HealthLake automáticamente CloudWatch métricas mejoradas con dimensiones adicionales para cumplir con los requisitos de presentación de informes de los CMS. Estas métricas rastrean el uso de la API por identidad de la persona que llama, aplicación y tipos de URI específicos del CMS sin necesidad de realizar ninguna configuración adicional.

Funcionamiento

Cuando realizas una llamada a la API mediante un punto final de interoperabilidad:

```
# This call...
curl https://healthlake.us-east-1.amazonaws.com/datastore/abc123/patientaccess/v2/r4/
Patient/123

# Automatically generates metrics with:
# - URIType: "patient-access"
# - Sub: extracted from your bearer token (SMART on FHIR datastores only)
# - ClientId: extracted from your bearer token (SMART on FHIR datastores only)
# - Plus all standard dimensions (DatastoreId, Operation, etc.)
```

No se necesita ningún código o configuración adicional. Simplemente utilice los puntos finales de interoperabilidad y las métricas mejoradas se capturarán automáticamente.

> **Note**
>
> En el caso de los almacenes de datos del FHIR que no son inteligentes, la `URIType` dimensión se sigue capturando, lo que permite realizar un seguimiento del uso de las API por categoría de CMS. `ClientId`Las dimensiones `Sub` y solo están disponibles cuando se utiliza la autenticación SMART on FHIR con fichas portadoras que contengan estas afirmaciones.

Nuevas dimensiones métricas

Además de las dimensiones existentes (`DatastoreId`,,`Operation`)`DatastoreType`, las siguientes dimensiones se añaden automáticamente cuando se utilizan puntos finales de interoperabilidad:

Dimensión	Description (Descripción)	Valores de ejemplo	origen
URIType	Categoría de conformidad con los CMS	`patient-access` , `provider-access` , `payer-to-payer` , `prior-authorization`	Se determina automáticamente a partir de la ruta del punto final de interoperabilidad
Sub	Identidad de la persona que llama	Identificador de usuario/entidad	Extraído de la afirmación del token del portador `sub`
ClientId	Identificador de la aplicación	`portal_app` , `ehr_system`	Extraído de la afirmación del portador del token `client_id`

Métricas disponibles

Todas las HealthLake métricas existentes ahora incluyen las dimensiones adicionales cuando se utilizan los puntos finales de interoperabilidad:

- CallCount- Número total de llamadas a la API
- Latencia: tiempo de respuesta de la API en milisegundos
- UserErrors- Recuento de 4 xx errores de cliente
- SystemErrors- Recuento de 5xx errores en el servidor
- Throttles: número de solicitudes restringidas
- SuccessfulRequests- Número de llamadas a la API que se han realizado correctamente

Consultando métricas en CloudWatch

CloudWatch Ejemplo de consulta de Insights

Consulta todas las llamadas a la API Patient Access por aplicación:

```
SELECT SUM(CallCount)
FROM "AWS/HealthLake"
WHERE DatastoreId = '75c1cf9b0d71cd38fec8f7fb317c4c1a'
    AND URIType = 'patient-access'
GROUP BY ClientId
```

Referencia de soporte para AWS HealthLake

El siguiente material de referencia de apoyo está disponible para AWS HealthLake.

> **Note**
>
> Todas HealthLake las acciones y tipos de datos nativos se describen en una referencia independiente. Para obtener más información, consulte la Referencia de la API de AWS HealthLake .

Temas

- AWS HealthLake cuotas y puntos de enlace
- Tipos de datos precargados de Synthea para HealthLake
- AWS HealthLake proyectos de muestra
- Solución de problemas AWS HealthLake
- HealthLake Utilización con un AWS SDK

AWS HealthLake cuotas y puntos de enlace

Los siguientes temas contienen información sobre los puntos finales y las cuotas del AWS HealthLake servicio.

Temas

- Puntos de conexión de servicio

- Cuotas de servicio

Puntos de conexión de servicio

Los puntos de conexión de servicio son URL que identifican un host y un puerto como puntos de entrada de un servicio web. Cada solicitud de servicio web contiene un punto de enlace. La mayoría de AWS los servicios proporcionan puntos de conexión para regiones específicas a fin de permitir una conectividad más rápida. En la siguiente tabla se enumeran los puntos finales del servicio para. AWS HealthLake

Nombre de la región	Región	Punto de conexión	Protocolo
Este de EE. UU. (Ohio)	us-east-2	healthlake.us-east-2.amazonaws.com healthlake-fips.us-east-2.amazonaws.com	HTTPS HTTPS
Este de EE. UU. (Norte de Virginia)	us-east-1	healthlake.us-east-1.amazonaws.com healthlake-fips.us-east-1.amazonaws.com	HTTPS HTTPS
Oeste de EE. UU. (Oregón)	us-west-2	healthlake.us-west-2.amazonaws.com healthlake-fips.us-west-2.amazonaws.com	HTTPS HTTPS
Asia-Pacífico (Mumbai)	ap-south-1	healthlake.ap-south-1.amazonaws.com	HTTPS
Asia-Pacífico (Sídney)	ap-southeast-2	healthlake.ap-southeast-2.amazonaws.com	HTTPS
Canadá (centro)	ca-central-1	healthlake.ca-central-1.amazonaws.com	HTTPS

Nombre de la región	Región	Punto de conexión	Protocolo
Europa (Irlanda)	eu-west-1	healthlake.eu-west-1.amazonaws.com	HTTPS
Europa (Londres)	eu-west-2	healthlake.eu-west-2.amazonaws.com	HTTPS

Cuotas de servicio

Las cuotas de servicio se definen como el valor máximo de los recursos, acciones y elementos de su AWS cuenta.

> **Note**
>
> Puede solicitar aumentos de cuota para cuotas ajustables mediante Service Quotas. Para obtener más información, consulte Solicitud de aumento de cuota en la Guía del usuario de Service Quotas.
> La velocidad de la API de sincronización y escritura aumenta proporcionalmente con el tamaño de la carga útil, y cada incremento de 1 KB consume capacidad adicional (por ejemplo, una carga útil de 4 KB utiliza 4 veces la capacidad de escritura). Si se configura el `x-amz-fhir-history-consistency-level` encabezado opcional para `strong` duplicar el consumo de capacidad de escritura por recurso.
> Los recursos de los paquetes siguen los read/write límites estándar basados en un tamaño de carga útil de 1 KB. Las transacciones de tipo lote consumen el doble de capacidad de escritura que las de tipo lote, lo que significa que los paquetes de lotes pueden procesar el doble de recursos por segundo que los paquetes de transacciones.

En la siguiente tabla se muestran las cuotas predeterminadas para. AWS HealthLake

Name	Predeterminado	Ajusta	Description (Descripción)
Número de suscripciones activas por cuenta	Cada región admitida: 100	Sí	El número máximo de recursos de suscripción activos por cuenta.
Número de recursos de suscripción activos por almacén de datos	Cada región admitida: 50	Sí	El número máximo de recursos de suscripción activos por almacén de datos.
Número de activos por cuenta SubscriptionTopic	Cada región admitida: 100	Sí	El número máximo de SubscriptionTopic recursos activos por cuenta.
Número de SubscriptionTopic recursos activos por almacén de datos	Cada región admitida: 50	Sí	El número máximo de SubscriptionTopic recursos activos por almacén de datos.
Número de caracteres de una nota médica	Cada región admitida: 10 000	No	El número máximo de caracteres de una nota médica individual dentro del tipo de DocumentReference recurso (POST/PUT solicitudes).
Número de trabajos simultáneos StartFHIRExportJob	Cada región admitida: 1	No	El número máximo de trabajos simultáneos StartFHIRExportJob .
Número de trabajos simultáneos StartFHIRImportJob	Cada región admitida: 1	No	El número máximo de trabajos simultáneos StartFHIRImportJob .
Número de almacenes de datos por cuenta	Cada región admitida: 10	Sí	El número máximo predeterminado de

Name	Predeterminado	Ajusta	Description (Descripción)
			almacenes de datos activos por cuenta.
Número de archivos en un StartFHIR ImportJob	Cada región admitida: 1 000 000	Sí	El número máximo de archivos en un StartFHIR ImportJob.
Número de recursos por paquete	Cada región admitida: 500	No	El número máximo de recursos permitido en una solicitud de paquete.
Porcentaje de CancelFHIRExportJob solicitudes que utilizan DELETE por cuenta	Cada región admitida: 1	No	El número máximo de CancelFHIRExportJob solicitudes mediante DELETE que puedes realizar por segundo por cuenta.
Tasa de solicitudes de CreateFHIRDatastore por cuenta	Cada región admitida: 1	No	El número máximo de solicitudes de CreateFHIRDatastore que puede realizar por minuto por cuenta.
Tasa de solicitudes DeleteFHIRDatastore por cuenta	Cada región admitida: 1	No	Número máximo de solicitudes de la API DeleteFHIRDatastore que se pueden realizar por minuto por cuenta.
Tasa de DescribeFHIRBulkDeleteJob solicitudes por cuenta	Cada región admitida: 10	Sí	El número máximo de DescribeFHIRBulkDeleteJob solicitudes que puedes realizar por segundo por cuenta.

Name	Predeterminado	Ajusta	Description (Descripción)
Tasa de DescribeFHIRBulkDeleteJob solicitudes por almacén de datos	Cada región admitida: 10	Sí	El número máximo de DescribeFHIRBulkDeleteJob solicitudes que puede realizar por segundo por almacén de datos.
Tasa de solicitudes de DescribeFhirDataStore por cuenta	Cada región admitida: 10	No	El número máximo de solicitudes de DescribeFHIRDatastore que puede realizar por segundo por cuenta.
Tasa de DescribeFHIRExportJob solicitudes por cuenta	Cada región admitida: 10	No	El número máximo de DescribeFHIRExportJob solicitudes que puedes realizar por segundo por cuenta.
Porcentaje de DescribeFHIRExportJob solicitudes por cuenta que utilizan GET	Cada región admitida: 10	No	El número máximo de DescribeFHIRExportJob solicitudes que puedes realizar mediante GET por segundo por cuenta.
Tasa de DescribeFHIRImportJob solicitudes por cuenta	Cada región admitida: 10	No	El número máximo de DescribeFHIRImportJob solicitudes que puedes realizar por segundo por cuenta.
Tasa de solicitudes de descubrimiento por cuenta	Cada región admitida: 10	No	El número máximo de solicitudes de descubrimiento que puede realizar por minuto por cuenta.

Name	Predeterminado	Ajusta	Description (Descripción)
Tasa de solicitudes GET por cuenta	Cada región admitida: 6000	Sí	Número máximo de solicitudes de GET que puede realizar por segundo y por cuenta.
Tasa de solicitudes GET por almacén de datos	Cada región admitida: 3000	Sí	Número máximo de solicitudes de GET que puede realizar por segundo y por almacén de datos. Los almacenes de datos creados antes de 8/21 /2023 se limitarán a 100 solicitudes por segundo.
Tasa de GetCapabilities solicitudes por cuenta	Cada región admitida: 10	No	El número máximo de GetCapabilities solicitud es que puedes realizar por segundo por cuenta.
Tasa de GetExportedFile solicitudes por almacén de datos	Cada región admitida: 10	No	El número máximo de GetExportedFile solicitud es que puede realizar por segundo por almacén de datos.
Tasa de solicitudes de ListFHIRD atastores por cuenta	Cada región admitida: 10	No	Número máximo de solicitudes ListFHIRD atastores que puede realizar por segundo.
Tasa de ListFHIRExportJobs solicitudes por cuenta	Cada región admitida: 10	No	El número máximo de ListFHIRExportJobs solicitudes que puedes realizar por segundo por cuenta.

Name	Predeterminado	Ajusta	Description (Descripción)
Tasa de ListFHIRImportJobs solicitudes por cuenta	Cada región admitida: 10	No	El número máximo de ListFHIRImportJobs solicitudes que puedes realizar por segundo por cuenta.
Tasa de ListTagsforResource solicitudes por cuenta	Cada región admitida: 10	No	El número máximo de ListTagsforResource solicitudes que puedes realizar por segundo por cuenta.
Tasa de SearchEverything solicitudes por cuenta	Cada región admitida: 10	Sí	El número máximo de SearchEverything solicitudes que puedes realizar por segundo por cuenta.
Tasa de SearchEverything solicitudes por almacén de datos	Cada región admitida: 10	Sí	El número máximo de SearchEverything solicitudes que puede realizar por segundo por almacén de datos.
Tasa de StartFHIRBulkDeleteJob solicitudes por cuenta	Cada región admitida: 10	Sí	El número máximo de StartFHIRBulkDeleteJob solicitudes que puedes realizar por segundo por cuenta.
Tasa de StartFHIRBulkDeleteJob solicitudes por almacén de datos	Cada región admitida: 10	Sí	El número máximo de StartFHIRBulkDeleteJob solicitudes que puede realizar por segundo por almacén de datos.

Name	Predeterminado	Ajusta	Description (Descripción)
Tasa de StartFHIRBulkMembe rMatchJob solicitudes por cuenta	Cada región admitida: 10	Sí	El número máximo de StartFHIRBulkMembe rMatchJob solicitudes que puedes realizar por segundo por cuenta.
Tasa de StartFHIRBulkMembe rMatchJob solicitudes por almacén de datos	Cada región admitida: 10	Sí	El número máximo de StartFHIRBulkMembe rMatchJob solicitudes que puede realizar por segundo por almacén de datos.
Tasa de StartFHIRExportJob solicitudes por cuenta	Cada región admitida: 1	No	El número máximo de StartFHIRExportJob solicitudes que puedes realizar por segundo por cuenta.
Porcentaje de StartFHIRExportJob solicitudes por cuenta que utilizan GET	Cada región admitida: 1	No	El número máximo de StartFHIRExportJob solicitudes que puedes realizar mediante GET por segundo por cuenta.
Tasa de StartFHIRExportJob solicitudes mediante POST por cuenta	Cada región admitida: 1	No	El número máximo de StartFHIRExportJob solicitudes mediante POST que puedes realizar por segundo por cuenta.

Name	Predeterminado	Ajusta	Description (Descripción)
Tasa de StartFHIRImportJob solicitudes por cuenta	Cada región admitida: 25	No	El número máximo de StartFHIRImportJob solicitudes que puedes realizar por segundo por cuenta.
Tasa de TagResource solicitudes por cuenta	Cada región admitida: 10	No	El número máximo de TagResource solicitudes que puedes realizar por segundo por cuenta.
Tasa de UntagResource solicitudes por cuenta	Cada región admitida: 10	No	El número máximo de UntagResource solicitudes que puedes realizar por segundo por cuenta.
Tasa de ValidateResource solicitudes por cuenta	Cada región admitida: 2000	Sí	El número máximo de ValidateResource solicitudes que puedes realizar por segundo por cuenta. Los almacenes de datos creados antes de 8/21 /2023 se limitarán a 300 solicitudes por segundo.

Name	Predeterminado	Ajusta	Description (Descripción)
Tasa de ValidateResource solicitudes por almacén de datos	Cada región admitida: 1000	Sí	El número máximo de ValidateResource solicitudes que puede realizar por segundo por almacén de datos. Los almacenes de datos creados antes de 8/21 /2023 se limitarán a 300 solicitudes por segundo.
Tasa de solicitudes de WRITE por cuenta	Cada región admitida: 6000	Sí	El número máximo de solicitudes de CREATE|UPDATE|DELETE que se pueden realizar por segundo por cuenta.
Tasa de solicitudes de WRITE por almacén de datos	Cada región admitida: 3000	Sí	El número máximo de solicitudes de CREATE|UPDATE|DELETE que se pueden realizar por segundo por almacén de datos. Los almacenes de datos creados antes de 8/21 /2023 se limitarán a 300 solicitudes por segundo.
Tasa de solicitudes de búsqueda utilizando GET por cuenta	Cada región admitida: 200	Sí	El número máximo de solicitudes GET que se pueden realizar por segundo en la región actual.

Name	Predeterminado	Ajusta	Description (Descripción)
Tasa de solicitudes de búsqueda utilizando GET por almacén de datos	Cada región admitida: 100	Sí	El número máximo de solicitudes de búsqueda utilizando GET que puede realizar por segundo por almacén de datos.
Tasa de solicitudes de búsqueda mediante POST por cuenta	Cada región admitida: 200	Sí	El número máximo de solicitudes POST que se pueden realizar por segundo en la región actual.
Tasa de solicitudes de búsqueda utilizando POST por almacén de datos	Cada región admitida: 100	Sí	El número máximo de solicitudes de búsqueda utilizando POST que puede realizar por segundo por almacén de datos.
Tamaño del archivo importado individual	Cada región admitida: 50 gigabytes	No	El tamaño máximo (en GB) de un archivo individual incluido en un StartFHIRImportJob.
Total de transacciones de paquetes asíncronos en cola por almacén de datos	Cada región admitida: 500	Sí	El número máximo de transacciones de paquetes asíncronos en cola por almacén de datos en un momento dado.
Total de trabajos de exportación masiva en cola por almacén de datos	Cada región admitida: 25	Sí	El número máximo de trabajos de exportaci ón masiva en cola por almacén de datos en un momento dado.

Name	Predeterminado	Ajusta	Description (Descripción)
Total de trabajos de importación masiva en cola por almacén de datos	Cada región admitida: 25	Sí	El número máximo de trabajos de importaci ón masiva en cola por almacén de datos en un momento dado.
Tamaño total del trabajo de importación	Cada región compatible: 5000 gigabytes	Sí	El tamaño máximo (en GB) de todos los archivos incluidos en el trabajo de importación.

Tipos de datos precargados de Synthea para HealthLake

HealthLake solo admite SYNTHEA como tipo de datos precargado. Synthea es un generador sintético de pacientes que `Patient` modela la historia clínica. Está alojado en un repositorio Git de código abierto que permite generar un recurso `Bundle` compatible con FHIR R4 HealthLake para que los usuarios puedan probar modelos sin utilizar datos reales de pacientes.

Los siguientes tipos de recursos están disponibles en los almacenes de datos precargados. HealthLake Para obtener más información sobre la precarga de HealthLake los almacenes de datos con datos de Synthea, consulte. Creación de un almacén HealthLake de datos

> **Note**
>
> Para ver una lista completa de los recursos del FHIR R4 HealthLake compatibles, consulte. Tipos de recursos compatibles con FHIR R4 para HealthLake

Tipos de recursos del FHIR de síntesis compatibles con HealthLake

AllergyIntolerance	Ubicación
CarePlan	MedicationAdministration
CareTeam	MedicationRequest

Reclamación	Observación
Condición	Organización
Dispositivo	Paciente
DiagnosticReport	Práctico
Encuentro	PractitionerRole
ExplanationofBenefit	Procedimiento
ImagingStudy	Procedencia
Inmunización	

AWS HealthLake proyectos de muestra

Para profundizar en el análisis, puede utilizarlos HealthLake con otros AWS servicios, como se muestra en los siguientes ejemplos de entradas de blog.

HealthLake análisis integrados

- Aplicaciones de salud de la población con AWS HealthLake — Parte 1: Análisis y monitoreo con Amazon Quick.
- Creación de modelos predictivos de enfermedades con Amazon SageMaker AI con datos AWS HealthLake normalizados.
- Cree una búsqueda cognitiva y un gráfico de conocimientos sobre salud mediante los servicios de AWS IA.

HealthLake monitoreo de eventos

- EventBridge Integración de Amazon con AWS HealthLake.

Solución de problemas AWS HealthLake

En los temas siguientes se proporcionan consejos para solucionar los errores y problemas que puedan surgir al utilizar la HealthLake consola AWS CLI AWS SDKs, o. Si encuentra un problema

que no aparece en esta sección, utilice el botón Enviar comentarios en la barra lateral derecha de esta página para informarlo.

Temas

- Acciones de almacenamiento de datos
- Acciones de importación
- FHIR APIs
- Integraciones de PNL
- Integraciones de SQL

Acciones de almacenamiento de datos

Problema: cuando intento crear un banco de HealthLake datos, aparece el siguiente error:

```
AccessDeniedException: Insufficient Lake Formation permission(s): Required Database on
 Catalog
```

El 14 de noviembre de 2022, HealthLake actualicé los permisos de IAM necesarios para crear un nuevo banco de datos. Para obtener más información, consulte Configure el usuario o rol de IAM que desee utilizar HealthLake (administrador de IAM).

Problema: Al crear un banco de HealthLake datos mediante el AWS SDKs, el estado de creación del banco de datos devuelve un estado de excepción o desconocido.

Actualice su AWS SDK a la última versión si sus llamadas `DescribeFHIRDatastore` o las de la `ListFHIRDatastores` API devuelven una excepción o un estado desconocido del almacén de datos.

Acciones de importación

Problema: ¿Puedo seguir utilizándolos HealthLake si mis datos no están en formato FHIR R4?

Solo los datos con formato FHIR R4 se pueden importar a un almacén de datos. HealthLake Para ver una lista de socios que pueden ayudar a transformar los datos de salud existentes al formato R4 del FHIR, consulte Socios.AWS HealthLake

Problema: ¿Por qué falló mi trabajo de importación del FHIR?

Si la importación se realiza correctamente, se generará una carpeta con los resultados (registro de salida) en `.ndjson` formato; sin embargo, es posible que no se puedan importar registros individuales. Cuando esto suceda, se generará una segunda `FAILURE` carpeta con un manifiesto de los registros que no se pudieron importar. Para obtener más información, consulte Importación de datos del FHIR con AWS HealthLake.

Para analizar por qué ha fallado un trabajo de importación, utilice la `DescribeFHIRImportJob` API para analizar el JobProperties. Se recomienda lo siguiente:

- Si el estado es `FAILED` y aparece un mensaje, los errores están relacionados con parámetros del trabajo, como el tamaño de los datos de entrada o el número de archivos de entrada, que superan HealthLake las cuotas.
- Si el estado del trabajo de importación es`COMPLETED_WITH_ERRORS`, consulte el archivo de manifiesto para obtener información sobre los archivos que no se importaron correctamente. `manifest.json`
- Si el estado del trabajo de importación es `FAILED` y no hay ningún mensaje, vaya a la ubicación de salida del trabajo para acceder al archivo de manifiesto`manifest.json`.

Para cada archivo de entrada, hay un archivo de salida con un nombre de archivo de entrada para cualquier recurso que no se pueda importar. Las respuestas contienen el número de línea (LineID) correspondiente a la ubicación de los datos de entrada, el objeto de respuesta del FHIR (UpdateResourceResponse) y el código de estado (StatusCode) de la respuesta.

Un ejemplo de archivo de salida podría ser similar al siguiente:

```
{"lineId":3, UpdateResourceResponse:{"jsonBlob":
{"resourceType":"OperationOutcome","issue":
[{"severity":"error","code":"processing","diagnostics":"1 validation error detected:
 Value 'Patient123' at 'resourceType' failed to satisfy constraint: Member must satisfy
 regular expression pattern: [A-Za-z]{1,256}"}]}, "statusCode":400}
{"lineId":5, UpdateResourceResponse:{"jsonBlob":
{"resourceType":"OperationOutcome","issue":
[{"severity":"error","code":"processing","diagnostics":"This property must be an
 simple value, not a com.google.gson.JsonArray","location":["/EffectEvidenceSynthesis/
name"]},{"severity":"error","code":"processing","diagnostics":"Unrecognised
 property '@telecom'","location":["/EffectEvidenceSynthesis"]},
{"severity":"error","code":"processing","diagnostics":"Unrecognised
 property '@gender'","location":["/EffectEvidenceSynthesis"]},
{"severity":"error","code":"processing","diagnostics":"Unrecognised
 property '@birthDate'","location":["/EffectEvidenceSynthesis"]},
```

```
{"severity":"error","code":"processing","diagnostics":"Unrecognised
 property '@address'","location":["/EffectEvidenceSynthesis"]},
{"severity":"error","code":"processing","diagnostics":"Unrecognised
 property '@maritalStatus'","location":["/EffectEvidenceSynthesis"]},
{"severity":"error","code":"processing","diagnostics":"Unrecognised
 property '@multipleBirthBoolean'","location":["/EffectEvidenceSynthesis"]},
{"severity":"error","code":"processing","diagnostics":"Unrecognised
 property '@communication'","location":["/EffectEvidenceSynthesis"]},
{"severity":"warning","code":"processing","diagnostics":"Name should be usable as an
 identifier for the module by machine processing applications such as code generation
 [name.matches('[A-Z]([A-Za-z0-9_]){0,254}')]","location":["EffectEvidenceSynthesis"]},
{"severity":"error","code":"processing","diagnostics":"Profile http://hl7.org/fhir/
StructureDefinition/EffectEvidenceSynthesis, Element 'EffectEvidenceSynthesis.status':
 minimum required = 1, but only found 0","location":["EffectEvidenceSynthesis"]},
{"severity":"error","code":"processing","diagnostics":"Profile
 http://hl7.org/fhir/StructureDefinition/EffectEvidenceSynthesis,
 Element 'EffectEvidenceSynthesis.population': minimum required
 = 1, but only found 0","location":["EffectEvidenceSynthesis"]},
{"severity":"error","code":"processing","diagnostics":"Profile
 http://hl7.org/fhir/StructureDefinition/EffectEvidenceSynthesis,
 Element 'EffectEvidenceSynthesis.exposure': minimum required =
 1, but only found 0","location":["EffectEvidenceSynthesis"]},
{"severity":"error","code":"processing","diagnostics":"Profile http://
hl7.org/fhir/StructureDefinition/EffectEvidenceSynthesis, Element
 'EffectEvidenceSynthesis.exposureAlternative': minimum required
 = 1, but only found 0","location":["EffectEvidenceSynthesis"]},
{"severity":"error","code":"processing","diagnostics":"Profile http://hl7.org/fhir/
StructureDefinition/EffectEvidenceSynthesis, Element 'EffectEvidenceSynthesis.outcome':
 minimum required = 1, but only found 0","location":["EffectEvidenceSynthesis"]},
{"severity":"information","code":"processing","diagnostics":"Unknown
 extension http://synthetichealth.github.io/synthea/disability-adjusted-
life-years","location":["EffectEvidenceSynthesis.extension[3]"]},
{"severity":"information","code":"processing","diagnostics":"Unknown extension
 http://synthetichealth.github.io/synthea/quality-adjusted-life-years","location":
["EffectEvidenceSynthesis.extension[4]"]}]}, "statusCode":400}
{"lineId":7, UpdateResourceResponse:{"jsonBlob":
{"resourceType":"OperationOutcome","issue":
[{"severity":"error","code":"processing","diagnostics":"2 validation errors detected:
 Value at 'resourceId' failed to satisfy constraint: Member must satisfy regular
 expression pattern: [A-Za-z0-9-.]{1,64}; Value at 'resourceId' failed to satisfy
 constraint: Member must have length greater than or equal to 1"}]}, "statusCode":400}
{"lineId":9, UpdateResourceResponse:{"jsonBlob":
{"resourceType":"OperationOutcome","issue":
```

```
[{"severity":"error","code":"processing","diagnostics":"Missing required id field in
 resource json"}]}, "statusCode":400}
{"lineId":15, UpdateResourceResponse:{"jsonBlob":
{"resourceType":"OperationOutcome","issue":
[{"severity":"error","code":"processing","diagnostics":"Invalid JSON found in input
 file"}]}, "statusCode":400}
```

El ejemplo anterior muestra que hubo errores en las líneas 3, 4, 7, 9 y 15 de las líneas de entrada correspondientes del archivo de entrada. Para cada una de esas líneas, las explicaciones son las siguientes:

- En la línea 3, la respuesta explica lo que `resourceType` se proporciona en la línea 3 del archivo de entrada no es válido.
- En la línea 5, la respuesta explica que hay un error de validación del FHIR en la línea 5 del archivo de entrada.
- En la línea 7, la respuesta explica que hay un problema de validación y `resourceId` se proporciona como entrada.
- En la línea 9, la respuesta explica que el archivo de entrada debe contener un identificador de recurso válido.
- En la línea 15, la respuesta del archivo de entrada es que el archivo no está en un formato JSON válido.

FHIR APIs

Asunto: ¿Cómo puedo implementar la autorización para el RESTful APIs FHIR?

Determine el que se Estrategia de autorización del almacén de datos va a utilizar.

Para crear una autorización SigV4 mediante el AWS SDK para Python (Boto3), cree un script similar al siguiente ejemplo.

```
import boto3
import requests
import json
from requests_auth_aws_sigv4 import AWSSigV4

# Set the input arguments
data_store_endpoint = 'https://healthlake.us-east-1.amazonaws.com/datastore/<datastore
 id>/r4//'
```

```
resource_path = "Patient"
requestBody = {"resourceType": "Patient", "active": True, "name": [{"use":
 "official","family": "Dow","given": ["Jen"]},{"use": "usual","given":
 ["Jen"]}],"gender": "female","birthDate": "1966-09-01"}
region = 'us-east-1'

#Frame the resource endpoint
resource_endpoint = data_store_endpoint+resource_path
session = boto3.session.Session(region_name=region)
client = session.client("healthlake")

# Frame authorization
auth = AWSSigV4("healthlake", session=session)

# Call data store FHIR endpoint using SigV4 auth

r = requests.post(resource_endpoint, json=requestBody, auth=auth, )
print(r.json())
```

Problema: ¿Por qué recibo *`AccessDenied`* errores al utilizar el FHIR RESTful APIs para un almacén de datos cifrado con una clave KMS gestionada por el cliente?

Para que un usuario o rol pueda acceder a un almacén de datos, se requieren permisos tanto para las claves administradas por el cliente como para las políticas de IAM. El usuario debe tener los permisos de IAM necesarios para utilizar una clave gestionada por el cliente. Si un usuario revoca o retira una concesión que daba HealthLake permiso para usar la clave de KMS administrada por el cliente, HealthLake devolverá un `AccessDenied` error.

HealthLake debe tener el permiso para acceder a los datos de los clientes, cifrar los nuevos recursos del FHIR importados a un almacén de datos y descifrar los recursos del FHIR cuando se soliciten. Para obtener más información, consulte Solución de problemas de permisos. AWS KMS

Problema: una operación de la *`POST`* API del FHIR que HealthLake utiliza un documento de 10 MB devuelve el *`413 Request Entity Too Large`* error.

AWS HealthLake tiene un límite sincrónico de 5 MB para crear y actualizar la API para evitar el aumento de las latencias y los tiempos de espera. Puede ingerir documentos de gran tamaño, de hasta 164 MB, utilizando el tipo de `Binary` recurso que utilice la API de importación masiva.

Integraciones de PNL

Problema: ¿Cómo activo la función HealthLake de procesamiento de lenguaje natural integrada?

A partir del 14 de noviembre de 2022, el comportamiento predeterminado de HealthLake los almacenes de datos cambió.

Almacenes de datos actuales: todos los almacenes de HealthLake datos actuales dejarán de utilizar el procesamiento del lenguaje natural (NLP) en los recursos codificados en base64`DocumentReference`. Esto significa que `DocumentReference` los nuevos recursos no se analizarán mediante la PNL y no se generarán nuevos recursos a partir del texto del tipo de recurso. `DocumentReference` En el caso de `DocumentReference` los recursos existentes, los datos y los recursos generados mediante la PNL permanecen, pero no se actualizarán después del 20 de febrero de 2023.

Nuevos almacenes de datos: HealthLake los almacenes de datos creados después del 20 de febrero de 2023 no procesarán el lenguaje natural (NLP) en recursos codificados en `DocumentReference` base64.

Para activar la integración de la HealthLake PNL, cree un caso de soporte utilizando. AWS Support Center Console Para crear tu caso, inicia sesión en tu y Cuenta de AWS, a continuación, selecciona Crear caso. Para obtener más información sobre la creación de un caso y la gestión de casos, consulte Creación de casos de soporte y gestión de casos en la Guía del Soporte usuario.

Problema: **>** ¿Cómo puedo encontrar *`DocumentReference`* recursos que no puedan procesarse con la PNL integrada?

Si un `DocumentReference` recurso no es válido, HealthLake proporciona una extensión que indica un error de validación en lugar de incluirlo en el resultado de la PNL médica integrada. Para encontrar **`DocumentReference`** los recursos que provocaron un error de validación durante el procesamiento de la PNL, puede utilizar la **`search`** función FHIR con la clave HealthLake de búsqueda cm-decoration-statusy el valor de búsqueda VALIDATION_ERROR. Esta búsqueda mostrará una lista de todos los `DocumentReference` recursos que provocaron errores de validación, junto con un mensaje de error que describe la naturaleza del error. La estructura del campo de extensión en los `DocumentReference` recursos con errores de validación se parecerá a la del siguiente ejemplo.

```
"extension": [
          {
              "extension": [
                  {
                      "url": "http://healthlake.amazonaws.com/aws-cm/status/",
                      "valueString": "VALIDATION_ERROR"
```

```
                },
                {
                    "url": "http://healthlake.amazonaws.com/aws-cm/message/",
                    "valueString": "Resource led to too many nested objects after NLP
 operation processed the document. 10937 nested objects exceeds the limit of 10000."
                }
            ],
            "url": "http://healthlake.amazonaws.com/aws-cm/"
        }
    ]
```

> **Note**
>
> También `VALIDATION_ERROR` puede ocurrir si la decoración de la PNL crea más de 10 000 objetos anidados. Cuando esto sucede, el documento debe dividirse en documentos más pequeños antes de procesarlo.

Integraciones de SQL

Problema: ¿Por qué obtengo un Lake Formation *`permissions error: lakeformation:PutDataLakeSettings`* al añadir un nuevo administrador de lagos de datos?

Si su usuario o rol de IAM contiene la política `AWSLakeFormationDataAdmin` AWS administrada, no podrá añadir nuevos administradores de lagos de datos. Aparecerá un error que contiene lo siguiente:

```
User arn:aws:sts::111122223333:assumed-role/lakeformation-admin-user is not authorized
 to perform: lakeformation:PutDataLakeSettings on resource: arn:aws:lakeformation:us-
east-2:111122223333:catalog:111122223333 with an explicit deny in an identity-based
 policy
```

La política AWS gestionada `AdministratorAccess` es necesaria para añadir un usuario o rol de IAM como administrador AWS del lago de datos de Lake Formation. Si su usuario o rol de IAM también lo contiene, `AWSLakeFormationDataAdmin` la acción fallará. La política `AWSLakeFormationDataAdmin` AWS gestionada contiene una denegación explícita de la operación de la API AWS Lake Formation,`PutDataLakeSetting`. Incluso los administradores con acceso total al AWS uso de la política `AdministratorAccess` administrada pueden estar limitados por la `AWSLakeFormationDataAdmin` política.

Problema: ¿Cómo puedo migrar un almacén de HealthLake datos existente para usar la integración de Amazon Athena SQL?

HealthLake los almacenes de datos creados antes del 14 de noviembre de 2022 son funcionales, pero no se pueden consultar en Athena mediante SQL. Para consultar un banco de datos preexistente con Athena, primero debe migrarlo a un banco de datos nuevo.

Para migrar sus HealthLake datos a un nuevo banco de datos

1. Cree un nuevo banco de datos.
2. Exporte los datos del depósito preexistente a un bucket de Amazon S3.
3. Importe los datos al nuevo almacén de datos desde el bucket de Amazon S3.

> **Note**
>
> La exportación de datos a un bucket de Amazon S3 conlleva un coste adicional. El cargo adicional depende del tamaño de los datos que exporte.

Problema: Al crear un nuevo HealthLake banco de datos para la integración con SQL, el estado del banco de datos no cambia*`Creating`*.

Si intenta crear un nuevo banco de HealthLake datos y su estado no cambia desde Crear, debe actualizar Athena para usar el. AWS Glue Data Catalog Para obtener más información, consulte Actualización al catálogo de datos de AWS Glue step-by-step en la Guía del usuario de Amazon Athena.

Después de actualizar correctamente el AWS Glue Data Catalog, puede crear un almacén de HealthLake datos.

Para eliminar un almacén de HealthLake datos antiguo, cree un caso de soporte utilizando AWS Support Center Console. Para crear su caso, inicie sesión en su y Cuenta de AWS, a continuación, seleccione Crear caso. Para obtener más información, consulte Creación de casos de soporte y administración de casos en la Guía del Soporte usuario.

Problema: la consola Athena no funciona después de importar datos a un nuevo HealthLake almacén de datos

Después de importar los datos a un nuevo banco de HealthLake datos, es posible que los datos no estén disponibles para su uso inmediato. Esto es para dar tiempo a que los datos se ingieran en las tablas de Apache Iceberg. Inténtelo de nuevo más tarde.

Problema: ¿Cómo puedo conectar los resultados de búsqueda de Athena con otros AWS servicios?

Al compartir los resultados de búsqueda de Athena con otros AWS servicios, pueden producirse problemas al utilizarlos `json_extract[1]` como parte de una consulta de búsqueda de SQL. Para solucionar este problema, debe actualizar a`CATVAR`.

Este problema puede producirse al intentar crear resultados guardados, una tabla (estática) o una vista (dinámica).

HealthLake Utilización con un AWS SDK

AWS Los kits de desarrollo de software (SDKs) están disponibles para muchos lenguajes de programación populares. Cada SDK proporciona una API, ejemplos de código y documentación que facilitan a los desarrolladores la creación de aplicaciones en su lenguaje preferido.

Documentación de SDK	Ejemplos de código
AWS SDK para C++	AWS SDK para C++ ejemplos de código
AWS CLI	AWS CLI ejemplos de código
AWS SDK para Go	AWS SDK para Go ejemplos de código
AWS SDK para Java	AWS SDK para Java ejemplos de código
AWS SDK para JavaScript	AWS SDK para JavaScript ejemplos de código
AWS SDK para Kotlin	AWS SDK para Kotlin ejemplos de código
AWS SDK para .NET	AWS SDK para .NET ejemplos de código
AWS SDK para PHP	AWS SDK para PHP ejemplos de código
Herramientas de AWS para PowerShell	Herramientas de AWS para PowerShell ejemplos de código

Documentación de SDK	Ejemplos de código
AWS SDK para Python (Boto3)	AWS SDK para Python (Boto3) ejemplos de código
AWS SDK para Ruby	AWS SDK para Ruby ejemplos de código
AWS SDK para Rust	AWS SDK para Rust ejemplos de código
AWS SDK para SAP ABAP	AWS SDK para SAP ABAP ejemplos de código
AWS SDK para Swift	AWS SDK para Swift ejemplos de código

Ejemplo de disponibilidad

¿No encuentra lo que necesita? Solicite un ejemplo de código a través del enlace de Enviar comentarios que se encuentra al final de esta página.

AWS HealthLake lanzamientos

En la siguiente tabla se muestra cuándo se publicaron las funciones y actualizaciones de AWS HealthLake. Para obtener más información sobre una versión, consulte el tema vinculado.

Cambio	Descripción	Fecha
Seguridad y filtrado de etiquetas para $export y $davinci-data-export	Las `$davinci-data-export` operaciones `$export` y ahora admiten `_security` y `_tag` consultan parámetros para filtrar los recursos exportados por valores de `meta.tag` codificación `meta.security` y filtrarlos. Esto permite la exportación multiusuario y el control de acceso detallado mediante el formato FHIR `system|code` estándar.	30 de abril de 2026
bulk-member-match Operación $	AWS HealthLake ahora admite la `$bulk-member-match` operación para procesar las solicitudes de coincidencia de varios miembros de forma asíncrona. Esta operación permite a las organizaciones de atención médica comparar de manera eficiente los identificadores únicos de cientos de miembros de diferentes sistemas de salud utilizando información demográfica y de cobertura en una sola solicitud masiva.	1 de abril de 2026

	• Gestiona hasta 500 miembros por solicitud con hasta 5 operaciones simultáneas por almacén de datos • Los resultados se clasifica n en MatchedMembers NonMatchedMembers, y grupos ConsentConstrained Members • Se integra con flujos `$davinci-data-expo rt` de trabajo de datos end-to-end masivos Para obtener más informaci ón, consulte the section called “$bulk-member-match”.	
Transacciones de paquetes asíncronas	AWS HealthLake ahora admite el `Bundle` tipo asíncrono `transaction` , lo que le permite enviar transacciones con hasta 500 recursos. HealthLake pone en cola la transacción para su procesami ento e inmediatamente devuelve una URL de sondeo para comprobar su estado y recuperar los resultados. Para obtener más informaci ón, consulte Transacciones de paquetes asíncronas.	24 de marzo de 2026

Soporte de parches en operaciones de paquetes	HealthLake ahora es compatible con Bundle Patch, lo que habilita FHIRPath tanto un parche JSON para los paquetes de transacciones como para los lotes, así como FHIRPath para la API de parches. Esta capacidad permite a los clientes parchear los recursos directamente dentro de las operaciones de Bundle sin necesidad de reemplazar todos los recursos, lo que reduce el tamaño de la carga útil, simplifica los flujos de trabajo de integración y permite una ingesta de datos más rápida.	20 de marzo de 2026
DaVinci PDex Tipos de exportación por $ davinci-data-export	La `$davinci-data-export` operación ahora admite los tipos de PDex exportación para Provider Access y Member Access APIs. Payer-to-Payer • Lógica de inclusión de recursos basada en perfiles ExplanationOfBenefit • Transformación de datos financieros con el parámetro `_includeEOB2xWoFinancial` • Filtro temporal de 5 años para los datos clínicos y de siniestros	20 de marzo de 2026

Parámetro _until en $export y $ davinci-data-export	Parámetro de filtrado temporal para operaciones de exportación	26 de febrero de 2026
_include parámetro de búsqueda	HealthLake ahora admite include: * e include:iterate	26 de febrero de 2026
Puntos finales de interoperabilidad de los CMS	Esta función le permite realizar un seguimiento del uso de las API por categoría de CMS y, posteriormente, informar sobre las métricas de uso con fines de cumplimiento.	26 de febrero de 2026
Bundle Message Type Support	Soporte limitado para los recursos de FHIR Bundle con tipo de mensaje	26 de febrero de 2026
Se agregó soporte para nuevos IGs	AWS HealthLake ha ampliado el soporte de sus guías de implementación del FHIR (IGs) para el CMS 0057F: • Support para CARIN Blue Button 2.0.0 y 2.1.0 • Support para Da Vinci Payer Data Exchange 2.0.0 y 2.1.0 • Support for DaVinci Payer Data Exchange (PDex) US Drug Formulary 2.0.1 y 2.1.0 • Support for Da Vinci Clinical Data Exchange (CDex) 2.1.0 • Soporte para Da Vinci Prior Authorization Support (PAS) FHIR IG 2.1.0	26 de febrero de 2026

Operación $submit	La `$submit` operación le permite enviar electróni camente solicitudes de autorización previa a los pagadores para su aprobació n.	26 de febrero de 2026
Funcionamiento del paquete $questionaire-package	La `$questionnaire-pac kage` operación recupera un paquete completo que contiene un cuestionario del FHIR y todas sus dependenc ias necesarias para procesar y procesar el cuestionario.	26 de febrero de 2026
Operación $inquire	La `$inquire` operación le permite comprobar el estado de una solicitud de autorización previa presentada anteriormente.	26 de febrero de 2026
Operación $member-remove	La operación $member-remove permite eliminar miembros de una lista de atribución de miembros del FHIR (recurso grupal) en. AWS HealthLake	12 de noviembre de 2025

Operación $member-match	AWS HealthLake ahora apoya la operación $member-match para recursos para pacientes , que permite a las organizac iones de atención médica encontrar el identificador único de un miembro en diferentes sistemas de salud utilizando información demográfica y de cobertura.	12 de noviembre de 2025
Operación $member-add	La operación $member-add del FHIR añade un miembro (paciente) a un recurso del grupo, específicamente a una lista de atribuciones de miembros.	12 de noviembre de 2025
davinci-data-export Operación $	La davinci-data-export operación $ es una operación FHIR asíncrona que permite exportar datos de la lista de atribución de miembros desde. AWS HealthLake	12 de noviembre de 2025
confirm-attribution-list Operación $	Indica al productor que el consumidor no tiene más cambios que realizar en la lista de atribuciones. Al finalizar la lista de atribucio nes, se eliminan los miembros inactivos y se cambia el estado a «definitivo».	12 de noviembre de 2025

Operación $attribution-status	Recupera el estado de atribución de un miembro específico y devuelve un paquete que contiene todos los recursos de atribución relacionados con el paciente.	12 de noviembre de 2025
Suscripciones al FHIR	HealthLake es compatible con las suscripciones del FHIR, lo que le permite recibir notificaciones en tiempo real cuando se produzcan cambios específicos en los datos de atención médica y crear flujos de trabajo basados en eventos.	30 de octubre de 2025
Expansión de la región a Montreal, Canadá	HealthLake está disponible en la región Canadá (Montreal). Para obtener más informaci ón, consulte Puntos finales del servicio.	17 de octubre de 2025

Se agregó soporte para nuevos IGs	AWS HealthLake ha ampliado el soporte de sus guías de implementación del FHIR (IGs) para los siguientes mercados canadienses: • Perfil FHIR de referencia de CA Core+Canadian que define los elementos de datos básicos y las limitaciones de la interoperabilidad entre los sistemas de salud canadienses. • CA:EREC Pan-Canadian eReferral: especificación eConsultStandardized FHIR para derivaciones y consultas electrónicas entre proveedores de atención médica de todo Canadá. • Resumen del paciente (edición canadiense) (PS-CA) Adaptación canadiense del Resumen internacional de pacientes (IPS) para compartir información esencial sobre la salud de los pacientes en todos los entornos de atención. • Perfil FHIR Repositor yOntario específico de Ontario Digital Health Drug para el intercambio de datos estandarizados sobre medicamentos y recetas	17 de octubre de 2025

	dentro del sistema de salud provincial.	
Soporte de IG específico para cada región	HealthLake ahora es compatible con regiones específicas IGs. Para obtener más información, consulte las validaciones de perfil.	8 de octubre de 2025
Funcionamiento del parche	HealthLake permite modificar elementos específicos de los recursos del FHIR mediante operaciones de parche JSON sin actualizar todo el recurso.	18 de agosto de 2025
Operación $validate	HealthLake permite validar los recursos del FHIR según las especificaciones y los perfiles sin realizar operacion es de almacenamiento, lo que arroja resultados de validación detallados.	18 de agosto de 2025
$purge (operación)	HealthLake incluye la funcionalidad para eliminar permanentemente del almacén de datos todos los recursos del compartimento del paciente.	18 de agosto de 2025
Operación $lookup	HealthLake proporciona la posibilidad de recuperar información detallada sobre un concepto específico en un, CodeSystem proporcionando el código y el identificador del sistema.	18 de agosto de 2025

Operación $expand	HealthLake ahora permite ampliar ValueSet los recursos para recuperar la lista completa de códigos contenidos en los archivos ingresados por el cliente ValueSets.	18 de agosto de 2025
Operación $erase	HealthLake ahora ofrece la eliminación permanente de un recurso específico y todas sus versiones históricas del almacén de datos.	18 de agosto de 2025
Operación $document	HealthLake permite generar documentos clínicos completos al agrupar un recurso de composición con todos sus recursos de referencia en un solo paquete de documentos.	18 de agosto de 2025
permite generar documentos clínicos completos al agrupar un recurso de composición con todos sus recursos de referencia en un solo paquete de documentos. —sep—:below Search Modifier	HealthLake introduce la búsqueda de valores de URI que estén jerárquicamente por debajo del URI especificado en un sistema terminológico.	8 de agosto de 2025

Eliminación condicional	HealthLake ahora es compatible con la eliminación condicional del FHIR, lo que permite a las organizaciones sanitarias eliminar un recurso existente en función de los criterios de búsqueda y no mediante un identificador FHIR lógico. Para obtener más información, consulte Eliminar los recursos del FHIR en función de las condiciones.	7 de julio de 2025
Se agregó soporte para nuevos IGs	AWS HealthLake ha ampliado el soporte de sus guías de implementación del FHIR (IGs) para lo siguiente: • US Core 7.0.0, que especifica cómo usar el FHIR para implementar el estándar USCDI 4.0 • Guía de implementación de UK Core 2.0.1 para proporcionar orientación de implementación del FHIR en todo el Reino Unido	7 de julio de 2025
Expansión de la región a Dublín, Irlanda	HealthLake está disponible en la región de la UE (Dublín). Para obtener más información, consulte Puntos finales del servicio.	9 de junio de 2025

Transacciones de tipo paquete	HealthLake ahora es compatible con el tipo de paquete «Transacción» del FHIR, lo que permite a las organizaciones sanitarias enviar varios recursos en una sola operación atómica. Esto permite flujos de trabajo de intercambio e integración de datos más eficientes. Por ejemplo, un proveedor de atención médica ahora puede actualizar el registro de un paciente, la lista de medicamentos y la cita en una sola transacción, lo que reduce la complejidad y los posibles errores. Consulte Cómo agrupar los recursos del FHIR para obtener más información.	28 de abril de 2025

Se agregó soporte para nuevos IGs	AWS HealthLake AWS HealthLake ha ampliado su compatibilidad con las guías de implementación del FHIR (IGs) para lo siguiente: • Guía de implementación del HEDIS® del NCQA (0.3.1): permite medir la calidad e informar sobre la calidad del conjunto de datos e información sobre la eficacia de la atención médica (HEDIS). • Resumen internacional para pacientes (IPS) (2.0.0): permite el intercambio de información de salud esencial para respaldar la continuidad de la atención de los pacientes. • Medida de calidad (5.0.0): apoya la representación y el intercambio de definiciones y datos de las medidas de calidad. • Informes genómicos (3.0.0): facilitan el intercambio de datos e informes genómicos .	28 de abril de 2025

Claves de idempotencia	HealthLake ahora admite claves de idempotencia para las operaciones POST del FHIR, lo que proporciona un mecanismo sólido para garantizar la integridad de los datos durante la creación de los recursos. Consulte Idempotencia y simultaneidad para obtener más información.	18 de abril de 2025
Coherencia del historial del FHIR	HealthLake ahora es compatible con una sólida coherencia en los almacenes de datos basados en el historial mediante el nuevo `x-amz-fhir-history-consistency-level` encabezado. Si se establece como «seguro», los resultados de búsqueda del FHIR incluyen todos los registros indexados, independientemente del estado de actualización. Consulte los niveles de consistencia de las búsquedas del FHIR para obtener más información.	18 de abril de 2025

Etiqueta y «si coincide»	HealthLake ahora es compatible con ETag, lo que permite a los clientes utilizar el encabezado «If-Match » para garantizar actualiza ciones idempotentes. Esto ayuda a mantener la integrida d de los datos al evitar que se sobrescriban accidenta lmente durante las actualiza ciones simultáneas. Esto es particularmente valioso en entornos de atención médica de gran volumen, donde varios sistemas pueden intentar actualizar el mismo registro simultáneamente. Consulte ETag AWS HealthLake para obtener más información.	18 de abril de 2025

Condicional PUTs en paquetes	HealthLake ahora admite actualizaciones condicionales para los paquetes FHIR, lo que brinda a las organizaciones de atención médica más flexibilidad para administrar y actualizar sus datos. Los clientes ahora pueden especificar criterios para crear, actualizar o eliminar recursos de forma condicional como parte de una transacción de paquete. Esto simplifica los procesos de sincronización de datos entre sistemas y reduce la necesidad de una lógica compleja del lado del cliente. Consulte Condicional PUTs en paquetes para obtener más información.	18 de abril de 2025

SMART en los osciloscopios FHIR V2	HealthLake es compatible con los osciloscopios SMART on FHIR V2 para crear, leer, actualizar, eliminar y buscar recursos del FHIR. Para obtener más información, consulte SMART sobre los ámbitos de los recursos del FHIR para. HealthLake • Los osciloscopios SMART on FHIR V2 están disponibl es para todos los almacenes de HealthLake datos creados después del 22/01/2025. Si su almacén de datos se creó antes de esta fecha, puede enviar una solicitud de asistenci a para activar SMART en los osciloscopios FHIR V2. Cree un caso utilizand o. AWS Support Center Console Para crear tu caso, inicia sesión en tu cuenta Cuenta de AWS y seleccion a Crear caso.	22 de enero de 2025
Perfil básico de FHIR US, versión 6.1.0	HealthLake es compatible con la versión 6.1.0 del FHIR US Core Profile. Para obtener más información, consulte las validaciones de los perfiles del FHIR para. HealthLake	22 de enero de 2025

FHIR $EXPORT CON GET	HealthLake apoya `$export` FHIR con. `GET` Para obtener más información, consulte Exportación de HealthLake datos con el `$export` FHIR.	22 de enero de 2025
Guía para desarrolladores refactorizada con ejemplos de código probados	HealthLake presenta una guía para desarrolladores rediseñada con ejemplos de código probados para acciones nativas y del SDK. AWS CLI AWS Además, ahora hay procedimientos disponibles para todas las interacciones de la API del FHIR compatibles. Para obtener más información, consulte Ejemplos de código y Administración de los recursos del FHIR.	18 de diciembre de 2024

El FHIR y `history` las interacciones `vread`	HealthLake admite la `history` interacción del FHIR para recuperar el historial de un recurso en particular y la `vread` interacción para realizar una lectura específica de una versión de un recurso. Para obtener más informaci ón, consulte Leer el historial de recursos del FHIR. • El recurso FHIR `history` está habilitado de forma predeterminada en todos los almacenes de HealthLak e datos creados después del 25 de octubre de 2024. Si su banco de datos se creó antes de esta fecha, puede enviar una solicitud de soporte para activar la interacción con el `history` FHIR. Cree un caso utilizando AWS Support Center Console. Para crear tu caso, inicia sesión en tu cuenta Cuenta de AWS y selecciona Crear caso.	25 de octubre de 2024

Operación FHIR `Patient/$everything`	HealthLake apoya la `Patient/$everything` operación del FHIR en la búsqueda de un `Patient` recurso y todos sus recursos relacionados. Con esta operación, puede acceder a la historia clínica completa de un paciente o descargar `Patient` datos de forma masiva. Para obtener más información, consulte Obtener datos de pacientes con `Patient/$everything`. • El FHIR `Patient/$everything` está activado de forma predeterminada en todos los almacenes de HealthLake datos creados después del 27/02/2024. Si su almacén de datos se creó antes de esta fecha, puede enviar un ticket de soporte para activar la `Patient/$everything` operación. Cree un caso utilizando AWS Support Center Console. Para crear tu caso, inicia sesión en tu cuenta Cuenta de AWS y seleccione Crear caso.	27 de febrero de 2024

Recurso FHIR `VerificationResult`	HealthLake admite el tipo de `VerificationResult` recurso FHIR para describir los requisitos de validación, las fuentes, el estado y las fechas de uno o más elementos. Para obtener más información, consulte los tipos de recursos R4 del FHIR para. HealthLake	9 de diciembre de 2023

Funcionamiento del FHIR `$export`	HealthLake apoya la `$export` operación del FHIR de exportación masiva de datos de salud desde un almacén de HealthLake datos. Para obtener más información, consulte Exportación de HealthLake datos con el `$export` FHIR. • El FHIR `$export` está activado de forma predeterminada en todos los almacenes de HealthLake datos creados después del 1 de junio de 2023. Si su almacén de datos se creó antes de esta fecha, puede enviar un ticket de soporte para activar la `$export` operación. Cree un caso utilizando AWS Support Center Console. Para crear tu caso, inicia sesión en tu cuenta Cuenta de AWS y selecciona Crear caso. • HealthLake Los almacenes de datos creados antes del 23 de enero de 2006 solo admiten solicitudes de `$export` trabajo para exportaciones a todo el sistema. • HealthLake Los almacenes de datos creados antes del 23 de enero de 2006 no	1 de junio de 2023

	permiten obtener el estado de un FHIR `$export` mediante una `GET` solicitud en el punto final del almacén de datos.	
SMART cuenta con el apoyo del FHIR	HealthLake añade compatibilidad con SMART con la autorización del FHIR. Para obtener más información, consulte SMART sobre el soporte del FHIR para. AWS HealthLake	31 de mayo de 2023
Validaciones de perfiles del FHIR	HealthLake admite las validaciones de perfiles del FHIR para definir definiciones de tipos de recursos específicos mediante extensiones de restricciones and/or en los tipos de recursos básicos. Para obtener más información, consulte Validaciones de perfiles.	31 de mayo de 2023
Región Asia-Pacífico (Bombay)	HealthLake está disponible en la región Asia Pacífico (Bombay). Para obtener más información, consulte Puntos finales del servicio.	4 de abril de 2023

El procesamiento del lenguaje natural está desactivado de forma predeterminada	HealthLake desactivó el procesamiento del lenguaje natural (NLP) integrado en todos los almacenes de datos a partir del 20 de febrero de 2023. Puede enviar un ticket de soporte para activar la funcionalidad de PNL integrada. Crea un caso usando AWS Support Center Console. Para crear tu caso, inicia sesión en tu cuenta Cuenta de AWS y seleccion a Crear caso. Para obtener más información sobre la PNL integrada, consulte Integrar la PNL con. HealthLake •	20 de febrero de 2023

Consulta e índice de SQL con Amazon Athena	HealthLake admite la consulta de datos de FHIR con SQL mediante Amazon Athena. Para obtener más informaci ón, consulte Consultas de HealthLake datos con Amazon Athena. • La funcionalidad de consulta SQL está habilitada de forma predeterminada en todos los almacenes de HealthLake datos creados después del 14 de noviembre de 2022. Si su banco de datos se creó antes de esta fecha, puede enviar una solicitud de soporte para activar la funcionalidad de consultas SQL. Cree un caso utilizand o AWS Support Center Console. Para crear tu caso, inicia sesión en tu cuenta Cuenta de AWS y seleccion a Crear caso. • Con la funcionalidad de consulta SQL, se HealthLak e debe actualizar la configuración de IAM para acceder. Tanto para crear almacenes de HealthLake datos como para conceder acceso a ellos en Athena, debe añadir la política `AWSLakeFormationDa`	14 de noviembre de 2022

	`taAdmin` gestionada a su usuario, grupo o función de IAM. Puede usar la `AWSLakeFormationDa taAdmin` política para crear administradores de lagos de datos y conceder acceso a los almacenes de datos en Athena. Para obtener más información, consulte Configurar un usuario o rol de IAM.	
El tamaño total del trabajo de importación aumentó	HealthLake actualiza el `Total import job size` para una `StartFHIRImportJob` solicitud a 500 GB. Para obtener más información, consulte Service Quotas.	3 de octubre de 2022
Recurso FHIR `Bundle`	HealthLake admite el tipo de `Bundle` recurso FHIR para procesar varios recursos FHIR simultáneamente. Para obtener más informaci ón, consulte Agrupación de recursos del FHIR.	5 de agosto de 2022
Actualizaciones de cuotas para las interacciones del FHIR	HealthLake actualiza las cuotas para las interacciones de gestión de recursos del FHIR. Para obtener más información, consulte Service Quotas.	16 de julio de 2022

Parámetro de `_include` búsqueda FHIR	HealthLake añade compatibilidad con el parámetro de `_include` búsqueda FHIR para devolver recursos adicionales en una `search` solicitud. Para obtener más información, consulte Parámetros de búsqueda avanzada.	16 de julio de 2022
AWS HealthLake está disponible de forma general	HealthLake está disponible de forma general en todas las regiones compatibles. Para obtener más información, consulte Puntos finales de servicio.	15 de julio de 2021

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.