Guida per gli sviluppatori

AWS HealthLake

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

AWS HealthLake: Guida per gli sviluppatori

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

I marchi e l'immagine commerciale di Amazon non possono essere utilizzati in relazione a prodotti o servizi che non siano di Amazon, in una qualsiasi modalità che possa causare confusione tra i clienti o in una qualsiasi modalità che denigri o discrediti Amazon. Tutti gli altri marchi non di proprietà di Amazon sono di proprietà dei rispettivi proprietari, che possono o meno essere affiliati, collegati o sponsorizzati da Amazon.

Table of Contents

Che cos'è AWS HealthLake? 1
Avviso importante 2
Funzionalità 2
Servizi correlati 3
Accesso 4
HIPAA 4
Prezzi 5
Nozioni di base 6
Concetti 6
strategia di autorizzazione 7
PNL integrata 7
Analisi integrate 7
Configurazione 7
Registrati per un Account AWS 8
Configura un utente o un ruolo IAM 8
Aggiungi un utente o un ruolo Data Lake Administrator 10
Crea bucket S3 11
Crea un data store 12
Configurare le autorizzazioni di importazione 13
Configurare le autorizzazioni di esportazione 15
Installa il AWS CLI 19
Tutorial 20
Gestione degli archivi dati 21
Creazione di un archivio dati 21
Ottenere le proprietà dell'archivio dati 29
Elencare gli archivi dati 32
Etichettatura degli archivi dati 36
Tagging di un datastore 37
Elenco dei tag per un archivio dati 40
Eliminare il tag di un archivio dati 43
Eliminazione di un archivio dati 46
Gestione degli abbonamenti FHIR 51
Come funzionano gli abbonamenti FHIR 51
Componenti chiave 51

Argomenti di abbonamento 52
Sottoscrizioni 52
Canali di notifica 53
Payload di notifica 53
Best practice 53
Ciclo di vita dell'abbonamento 54
Creazione di un abbonamento 56
Esempi di payload di abbonamento 59
Esempi di payload di notifica 63
Ricerca di abbonamenti 68
Filtraggio delle notifiche 71
Importazione di dati FHIR 74
Avvio di un processo di importazione 76
Ottenere proprietà lavorative da importare 81
Elenco dei lavori di importazione 85
Gestione delle risorse FHIR 90
Creazione di una risorsa 92
Leggere una risorsa 95
Leggere la cronologia delle risorse 97
Leggere la cronologia specifica della versione 100
Aggiornamento di una risorsa 102
Aggiornamento condizionale 104
Configurazione del livello di convalida per gli aggiornamenti delle risorse 105
Modifica di una risorsa 106
Formati PATCH supportati 107
Utilizzo 107
Formato di patch JSON 108
FHIRPath Formato della patch 110
Intestazioni di richiesta 112
Risposta di esempio 113
Comportamento 113
Gestione errori 114
Riepilogo delle capacità 114
Limitazioni 115
Risorse aggiuntive 115
Raggruppamento di risorse 115

Raggruppa come entità indipendenti 120
Condizionale PUTs 124
Raggruppa come singola entità 127
Configurazione del livello di convalida per i pacchetti 130
Supporto limitato per il tipo «message» di tipo Bundle 132
Transazioni asincrone 133
Eliminazione di una risorsa 143
Eliminazione condizionale per FHIR 145
Idempotenza e concorrenza 147
Chiavi di idempotenza 147
ETag in AWS HealthLake 148
Ricerca di risorse FHIR 151
Ricerca con GET 151
GET: esempi di ricerca 154
Ricerca con POST 155
Esempi di ricerca POST 158
Livelli di coerenza della ricerca 160
Livelli di coerenza 161
Esempio di utilizzo 161
Best practice 162
Esportazione di dati FHIR 163
Avvio di un lavoro di esportazione 163
Ottenere proprietà lavorative destinate all'esportazione 168
Elenco dei lavori di esportazione 172
Esempi di codice 177
Nozioni di base 177
Azioni 178
Integrazione 225
Elaborazione linguaggio naturale 225
Librerie NLP 226
Utilizzo di FHIR APIs 228
Parametri di ricerca 228
Richieste di esempio 231
Indice e interrogazione SQL 248
Nozioni di base 248
Interrogazione con SQL 252

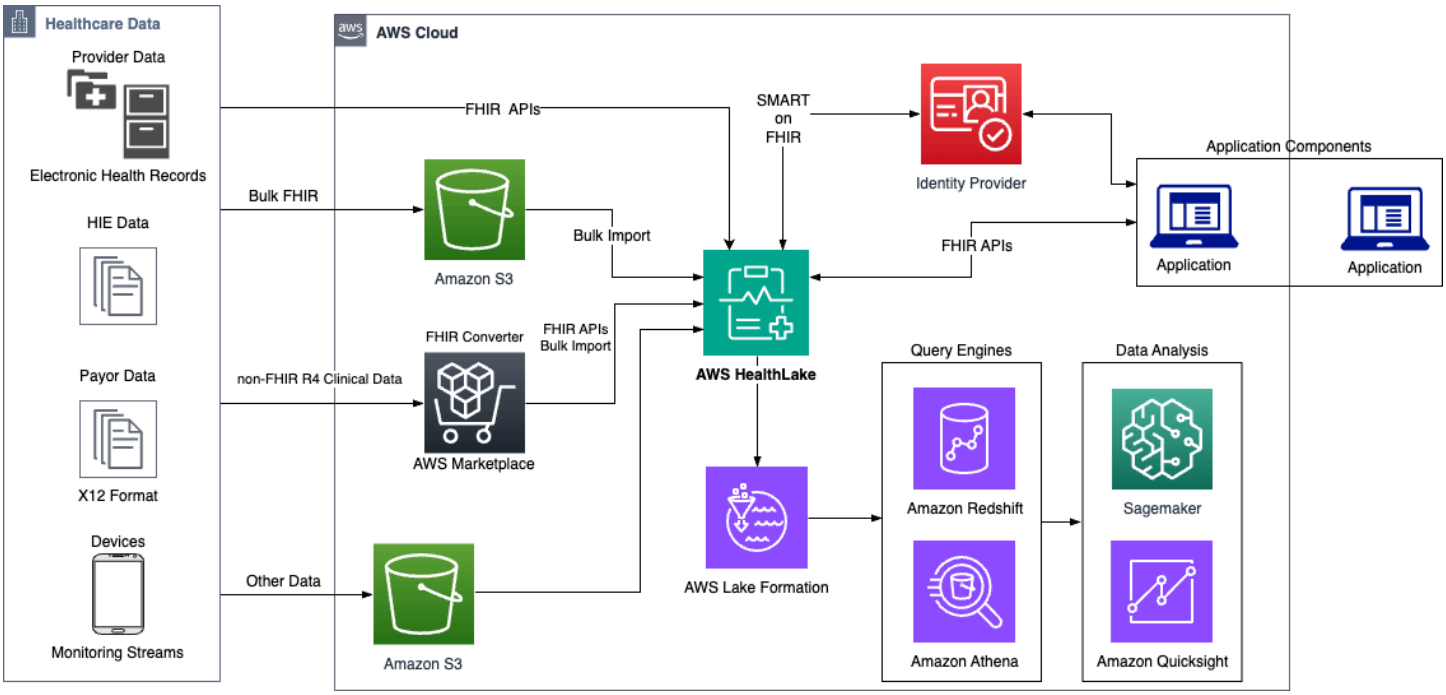
Query di esempio 259
Monitoraggio 266
CloudTrail (chiamate API) 267
AWS HealthLake Informazioni in CloudTrail 267
Comprensione delle AWS HealthLake voci dei file di registro 269
CloudWatch (Metriche) 270
Visualizzazione delle metriche HealthLake 273
Creazione di un allarme 273
EventBridge (Eventi) 274
HealthLake eventi inviati a EventBridge 274
HealthLake struttura dell'evento 276
Sicurezza 290
Protezione dei dati 291
Crittografia dei dati a riposo 292
Chiave KMS di proprietà di AWS 292
Chiavi KMS gestite dal cliente 292
Creazione di una chiave gestita dal cliente 293
Autorizzazioni IAM richieste per l'utilizzo di una chiave KMS gestita dal cliente 294
Crittografia dei dati in transito 301
Gestione dell’identità e degli accessi 301
Destinatari 302
Autenticazione con identità 302
Gestione dell’accesso tramite policy 304
Come AWS HealthLake funziona con IAM 305
Esempi di policy basate su identità 311
AWS politiche gestite 314
Risoluzione dei problemi 319
Convalida della conformità 321
Sicurezza dell’infrastruttura 321
Infrastructure as code (IaC) 322
HealthLake e CloudFormation modelli 322
Scopri di più su CloudFormation 322
Endpoint VPC 323
Considerazioni sugli endpoint HealthLake VPC 323
Creazione di un endpoint VPC di interfaccia per; HealthLake 323
Creazione di una policy per gli endpoint VPC per HealthLake 324

Best practice 324
Resilienza 325
Documentazione di riferimento 326
SMART su FHIR 326
Nozioni di base 327
Autenticazione 330
OAuth Cannocchiali 2.0 331
Convalida del token 335
Autorizzazione granulare 347
Documento Discovery 348
Esempio di richiesta 349
FHIR R4 350
Dichiarazione di capacità 351
Convalide dei profili 352
Tipi di risorse 359
Parametri di ricerca 361
$ Operazioni 374
Conformità 523
CMS 524
HealthLake 530
Punti finali e quote 530
Tipi di dati precaricati 541
Progetti di esempio 542
Risoluzione dei problemi 543
Lavorare con AWS SDKs 551
Rilasci 553
..... dlxxx

Che cos'è AWS HealthLake?

AWS HealthLake è un servizio idoneo alla normativa HIPAA per l'archiviazione, l'analisi e la condivisione di dati sanitari nel cloud utilizzando la specifica Fast Healthcare Interoperability Resources (FHIR) R4. HealthLake i casi d'uso includono:

- Dati sanitari aziendali: gestisci e condividi i dati sanitari FHIR R4 direttamente da, preservando al Cloud AWS contempo prestazioni e disponibilità elevate.
- Interoperabilità sanitaria: supporta la conformità dei clienti al 21st Century Cures Act per l'accesso dei pazienti attraverso un archivio dati FHIR completamente gestito.
- Elaborazione del linguaggio naturale (NLP): utilizza modelli NLP integrati per estrarre informazioni mediche significative da dati sanitari non strutturati.
- Analisi multimodale: combina HealthLake dati con dati e dati per fornire approfondimenti per la medicina di AWS HealthImaging precisione. AWS HealthOmics



Argomenti

- Avviso importante
- Caratteristiche di AWS HealthLake
- Servizi correlati AWS

- Accesso AWS HealthLake
- Idoneità alla normativa HIPAA e sicurezza dei dati
- Prezzi

Avviso importante

AWS HealthLake non sostituisce la consulenza, la diagnosi o il trattamento medico professionale e non è destinato a curare, trattare, mitigare, prevenire o diagnosticare alcuna malattia o condizione di salute. L'utente è responsabile dell'avvio della revisione umana nell'ambito di qualsiasi utilizzo AWS HealthLake, anche in associazione con, di qualsiasi prodotto di terze parti destinato a informare il processo decisionale clinico. AWS HealthLake deve essere usato nella cura del paziente o in scenari clinici solo dopo la revisione da parte di professionisti medici qualificati che applicano un solido giudizio medico.

Caratteristiche di AWS HealthLake

AWS HealthLake offre le seguenti funzionalità.

Importa i dati sanitari FHIR R4

Con l'azione di importazione HealthLake nativa, puoi migrare facilmente i tuoi dati FHIR da un bucket Amazon S3 a HealthLake un data store, tra cui note cliniche, report di laboratorio, richieste di risarcimento e altro ancora. HealthLake supporta la specifica FHIR R4 per lo scambio di dati sanitari. Se necessario, puoi collaborare con un AWS HealthLake partner per convertire i tuoi dati sanitari in formato FHIR R4.

Archivia i dati sanitari in modo sicuro, conforme e verificabile

Un archivio HealthLake dati aiuta a indicizzare i dati sanitari in modo che possano essere interrogati. L'archivio dati crea una visione completa della storia medica di ogni paziente in ordine cronologico e facilita lo scambio di informazioni utilizzando la specifica FHIR R4. Inoltre, è sempre attivo per mantenere l'indice aggiornato, offrendoti la possibilità di cercare le informazioni in qualsiasi momento utilizzando le interazioni FHIR R4 standard con storage primario durevole e scalabilità dell'indice.

Sfrutta il server FHIR transazionale

Sfrutta FHIR APIs per la convalida standard delle risorse, l'autorizzazione SMART on FHIR e le funzionalità di esportazione dell'API FHIR per dati in blocco per supportare l'unificazione e l'analisi

dei dati per ridurre i costi operativi e migliorare il processo decisionale. HealthLake supporta la conformità del cliente ai più recenti standard normativi ONC e CMS, tra cui: HL7 FHIR R4 APIs, FHIR Bulk Data Access, US Core IG STU, HL7 SMART App Launch Framework IG, 2.0 e OpenID Connect. OAuth

Trasforma i dati medici non strutturati utilizzando la PNL

L'elaborazione medica integrata del linguaggio naturale (NLP) trasforma tutti i dati di testo medico non elaborati in un archivio dati per comprendere ed estrarre informazioni significative da HealthLake dati sanitari non strutturati. Con la PNL medica integrata, puoi estrarre automaticamente entità, relazioni tra entità, tratti delle entità e informazioni sanitarie protette (PHI) dal tuo testo medico. Le entità estratte dall'NLP vengono archiviate come risorse FHIR R4 native all'interno di un HealthLake data store e sono accessibili tramite FHIR R4 o APIs Amazon Athena (SQL).

Servizi correlati AWS

AWS HealthLake offre una stretta integrazione con altri AWS servizi. La conoscenza dei seguenti servizi è utile per HealthLake sfruttarli appieno.

- AWS Identity and Access Management— Utilizza IAM per gestire in modo sicuro le identità e l'accesso alle risorse. HealthLake
- Amazon Simple Storage Service: usa Amazon S3 come area di staging in cui importare dati DICOM. HealthLake
- AWS CloudTrail— Utilizzabile CloudTrail per tenere traccia delle attività HealthLake degli utenti e dell'utilizzo delle API.
- Amazon CloudWatch: CloudWatch da utilizzare per osservare e monitorare HealthLake le risorse.
- AWS CloudFormation— Utilizzabile CloudFormation per implementare modelli Infrastructure as Code (IaC) in HealthLake cui creare risorse.
- AWS PrivateLink— Usa Amazon VPC per stabilire la connettività tra Amazon HealthLake e Amazon Virtual Private Cloud senza esporre i dati a Internet.
- Amazon EventBridge: EventBridge da utilizzare per creare applicazioni scalabili e basate sugli eventi creando regole che indirizzano HealthLake gli eventi verso le destinazioni.
- AWS Lake Formation— Usa Lake Formation per governare, proteggere e condividere centralmente HealthLake i dati per l'analisi e l'apprendimento automatico.

- Amazon Athena: usa Athena per interrogare HealthLake i dati con SQL per consentire un'analisi più approfondita.

Accesso AWS HealthLake

È possibile accedere AWS HealthLake utilizzando il Console di gestione AWS, AWS Command Line Interface e il AWS SDKs. Questa guida fornisce istruzioni procedurali per Console di gestione AWS e esempi di codice per AWS CLI and AWS SDKs.

AWS Command Line Interface (AWS CLI)

AWS CLI Fornisce comandi per un'ampia gamma di AWS prodotti ed è supportato su Windows, Mac e Linux. Per ulteriori informazioni, consulta la Guida per l'utente AWS Command Line Interface .

AWS SDKs

AWS SDKs fornisce librerie, esempi di codice e altre risorse per gli sviluppatori di software. Queste librerie forniscono funzioni di base che automatizzano attività come la firma crittografica delle richieste, il ritentativo delle richieste e la gestione delle risposte agli errori. Per ulteriori informazioni, consulta Tools to Building on. AWS

Console di gestione AWS

Console di gestione AWS Fornisce un'interfaccia utente basata sul Web per la gestione HealthLake e le risorse associate. Se hai registrato un AWS account, puoi accedere alla HealthLake Console.

Idoneità alla normativa HIPAA e sicurezza dei dati

Questo è un servizio idoneo ai fini HIPAA. Per ulteriori informazioni sull' AWS U.S. Health Insurance Portability and Accountability Act del 1996 (HIPAA) e sull'utilizzo AWS dei servizi per elaborare, archiviare e trasmettere informazioni sanitarie protette (PHI), vedere Panoramica HIPAA.

Le connessioni HealthLake contenenti PHI e informazioni di identificazione personale (PII) devono essere crittografate. Per impostazione predefinita, tutte le connessioni HealthLake utilizzano HTTPS su TLS. HealthLake archivia i contenuti crittografati dei clienti e opera secondo il modello di responsabilitàAWS condivisa.

Prezzi

Per informazioni HealthLake sui prezzi, consulta la pagina AWS HealthLake dei prezzi. Per stimare i costi, utilizza il calcolatore HealthLake dei prezzi.

Guida introduttiva con AWS HealthLake

Per iniziare a utilizzare AWS HealthLake, configura un AWS account e crea un AWS Identity and Access Management utente. Per utilizzare AWS CLIo AWS SDKs, è necessario installarli e configurarli.

> **Note**
>
> Il Documentazione di riferimento capitolo di questa guida fornisce contenuti di supporto per SMART su FHIR, FHIR R4 e. AWS HealthLake Ad esempio, è possibile trovare informazioni sulla configurazione SMART on FHIR, sulle convalide dei profili FHIR supportate e sugli endpoint. HealthLake

Dopo aver appreso HealthLake i concetti e la configurazione, è disponibile un breve tutorial con esempi di codice per aiutarti a iniziare.

Argomenti

- AWS HealthLake concetti
- Configurazione AWS HealthLake
- AWS HealthLake tutorial

AWS HealthLake concetti

La terminologia e i concetti seguenti sono fondamentali per la comprensione e l'uso di. AWS HealthLake

Concetti

- Strategia di autorizzazione all'archiviazione dei dati
- PNL integrata
- Analisi integrate

Strategia di autorizzazione all'archiviazione dei dati

Un HealthLake data store è un archivio di dati sanitari FHIR R4 che risiede all'interno di un unico archivio. Regione AWS HealthLake supporta le seguenti strategie di autorizzazione degli archivi dati.

- Autorizzazione SigV4: HealthLake autorizza le chiamate API FHIR utilizzando l'autorizzazione AWS Signature Version 4 (SigV4).
- Autorizzazione SMART su FHIR: HealthLake autorizza le chiamate API FHIR utilizzando Applicazioni mediche sostitutabili e tecnologie riutilizzabili (SMART) sull'autorizzazione FHIR.

Per ulteriori informazioni, consulta Creazione di un archivio HealthLake dati.

PNL integrata

AWS HealthLake si integra con le librerie di elaborazione del linguaggio naturale (NLP) idonee alla normativa HIPAA per estrarre dati sanitari significativi da testi medici non strutturati. Le librerie NLP identificano entità mediche come condizioni, farmaci, dosaggi, test, trattamenti e procedure. Riconoscono le relazioni tra le entità e le collegano a librerie di ontologia medica come ICD-10-CM e. RxNorm Per ulteriori informazioni, consulta Elaborazione integrata del linguaggio naturale (NLP) per HealthLake.

Analisi integrate

AWS HealthLake va oltre la FHIR `search` e `bundle` APIs fornisce analisi integrate per interrogare e analizzare grandi volumi di dati sanitari. Durante l'importazione, genera HealthLake automaticamente tabelle per l'indice e la query SQL. Ciò consente di ottenere informazioni utili da dati sanitari complessi senza richiedere un ampio lavoro di ingegneria dei dati. Per ulteriori informazioni, consultare Interrogazione di HealthLake dati con Amazon Athena e AWS HealthLake progetti di esempio.

Configurazione AWS HealthLake

In questo capitolo, si utilizza Console di gestione AWS per impostare le autorizzazioni necessarie per iniziare a utilizzare AWS HealthLake e creare un archivio dati. Per configurare le autorizzazioni per creare un data store, crei un utente o un ruolo IAM che sia amministratore e HealthLake amministratore del data lake. Rendi questo utente un amministratore di data lake in AWS Lake Formation. L'amministratore del data lake concede a Lake Formation l'accesso alle risorse

necessarie per utilizzare Amazon Athena per interrogare un data store. Dopo aver creato un HealthLake data store, puoi configurare le autorizzazioni per l'importazione e l'esportazione di file.

Argomenti

- Registrati per un Account AWS
- Configura un utente o un ruolo IAM da utilizzare HealthLake (amministratore IAM)
- Aggiungi un utente o un ruolo come Data Lake Administrator in Lake Formation (IAM Administrator)
- Crea bucket S3
- Crea un data store
- Impostazione delle autorizzazioni per i lavori di importazione
- Impostazione delle autorizzazioni per i lavori di esportazione
- Installa il AWS CLI

Registrati per un Account AWS

Per iniziare AWS, hai bisogno di un Account AWS. Per informazioni sulla creazione di un Account AWS, vedi Guida introduttiva a un Account AWS nella Guida Gestione dell'account AWS di riferimento.

Configura un utente o un ruolo IAM da utilizzare HealthLake (amministratore IAM)

> Persona: amministratore IAM
>
> Un utente che può creare utenti e ruoli IAM e aggiungere amministratori di data lake.

I passaggi descritti in questo argomento devono essere eseguiti da un amministratore IAM.

Per connettere il tuo HealthLake data store ad Athena, devi creare un utente o un ruolo IAM che sia un amministratore del data lake e un HealthLake amministratore. Questo nuovo utente o ruolo concede l'accesso alle risorse presenti in un data store tramite AWS Lake Formation e la policy `AmazonHealthLakeFullAccess` AWS gestita viene aggiunta al relativo utente o ruolo.

> **Important**
>
> Un utente o un ruolo IAM che è un amministratore di data lake non può creare nuovi amministratori di data lake. Per aggiungere un amministratore del data lake aggiuntivo, è necessario utilizzare un utente o un ruolo IAM a cui è stato concesso `AdministratorAccess` l'accesso.

Per creare un amministratore

1. Aggiungi la policy AWS gestita da **`AmazonHealthlakeFullAccess`** IAM a un utente o a un ruolo nella tua organizzazione.

   Se non hai dimestichezza con la creazione di un utente IAM, consulta Creazione di un utente IAM e Panoramica delle politiche AWS IAM nella Guida per l'utente IAM.
2. Concedi all'utente o al ruolo IAM l'accesso a AWS Lake Formation.

   - Aggiungi la seguente policy AWS gestita da IAM a un utente o a un ruolo nella tua organizzazione: **`AWSLakeFormationDataAdmin`**

     > **Note**
     >
     > La `AWSLakeFormationDataAdmin` politica garantisce l'accesso a tutte le risorse di AWS Lake Formation. È consigliabile utilizzare sempre le autorizzazioni minime necessarie per eseguire l'attività. Per ulteriori informazioni, consulta Best practice IAM nella Guida per l'utente di IAM.
3. Aggiungi la seguente politica in linea all'utente o al ruolo. Per ulteriori informazioni, consulta Inline policies nella IAM User Guide.

   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": "*"
        }
    ]
}
```

Per ulteriori informazioni sulla `AWSLakeFormationDataAdmin` policy, consulta Lake Formation Personas and IAM Permissions Reference nella AWS Lake Formation Developer Guide.

Aggiungi un utente o un ruolo come Data Lake Administrator in Lake Formation (IAM Administrator)

> **Note**
>
> Questo passaggio è necessario in caso di integrazione. Indice e interrogazione SQL

Successivamente, l'amministratore IAM deve aggiungere l'utente o il ruolo creato nel passaggio precedente come amministratore del data lake in Lake Formation.

Per aggiungere un utente o un ruolo IAM come amministratore del data lake

1. Apri la console AWS Lake Formation: https://console.aws.amazon.com/lakeformation/

> **Note**
>
> Se è la prima volta che visiti Lake Formation, viene visualizzata una finestra di dialogo Welcome to Lake Formation che ti chiede di definire un amministratore di Lake Formation.

2. Assegna al nuovo utente o ruolo l'amministratore del data AWS lake di Lake Formation.
   - Opzione 1: se hai ricevuto la finestra di dialogo Welcome to Lake Formation.
     1. Scegli Aggiungi altri AWS utenti o ruoli.
     2. Scegli la freccia rivolta verso il basso (▼).
     3. Scegli l' HealthLake amministratore di cui vorresti che diventasse anche amministratore di Lake Formation.
     4. Scegli Avvia.
   - Opzione 2: utilizzare il pannello di navigazione (☰).
     1. Scegli il pannello di navigazione (☰).
     2. In Autorizzazioni, scegli Ruoli e attività amministrative.
     3. Nella sezione Amministratori di Data lake, seleziona Scegli amministratori.
     4. Nella finestra di dialogo Gestisci gli amministratori del data lake, scegli la freccia rivolta verso il basso (▼).
     5. Successivamente, seleziona o cerca gli HealthLake amministratori, gli utenti o i ruoli che desideri siano anche amministratori di Lake Formation.
     6. Scegli Save (Salva).
3. Modifica le impostazioni di sicurezza predefinite che devono essere gestite da Lake Formation. Le risorse del HealthLake data store devono essere gestite da Lake Formation e non da IAM. Per aggiornare, consulta Modifica del modello di autorizzazione predefinito nella AWS Lake Formation Developer Guide.

Crea bucket S3

Per importare dati FHIR R4 in AWS HealthLake, sono consigliati due bucket Amazon S3. Il bucket di input Amazon S3 contiene i dati FHIR da importare e li HealthLake legge. Il bucket di output di

Amazon S3 memorizza i risultati di elaborazione del processo di importazione e HealthLake scrive (log) in questo bucket.

> **Note**
>
> A causa della politica AWS Identity and Access Management (IAM), i nomi dei bucket Amazon S3 devono essere univoci. Per ulteriori informazioni, consulta Regole per la denominazione dei bucket nella Guida per l'utente di Amazon Simple Storage Service.

Ai fini di questa guida, specifichiamo i seguenti bucket di input e output di Amazon S3 durante la configurazione delle autorizzazioni di importazione più avanti in questa sezione.

- Bucket di input: `arn:aws:s3:::amzn-s3-demo-source-bucket`
- Secchio di uscita: `arn:aws:s3:::amzn-s3-demo-logging-bucket`

Per ulteriori informazioni, consulta Creating a bucket nella Amazon S3 User Guide.

Crea un data store

Un HealthLake data store è un archivio di dati FHIR R4 che risiede all'interno di una singola regione. AWS Un AWS account può avere zero o molti archivi dati. HealthLake supporta due strategie di autorizzazione degli archivi dati.

> **Importante**
>
> Prima di creare un archivio HealthLake dati, esamina le politiche di controllo dei servizi (SCP) AWS dell'organizzazione che potrebbero limitare la creazione o la gestione delle HealthLake risorse. Gli SCP possono impedire la corretta creazione di archivi HealthLake dati, anche se le autorizzazioni IAM sono configurate correttamente.
> A `datastoreID` viene generato quando si crea un HealthLake data store. È necessario utilizzare il `datastoreID` quando si impostano le autorizzazioni di importazione più avanti in questa sezione.

Per creare un archivio HealthLake dati, vedereCreazione di un archivio HealthLake dati.

Impostazione delle autorizzazioni per i lavori di importazione

Prima di importare i file in un data store, devi concedere l' HealthLake autorizzazione per accedere ai tuoi bucket di input e output in Amazon S3. Per concedere HealthLake l'accesso, devi creare un ruolo di servizio IAM HealthLake, aggiungere una policy di fiducia al ruolo per concedere le autorizzazioni di HealthLake assunzione del ruolo e allegare una policy di autorizzazione al ruolo che gli consenta l'accesso ai tuoi bucket Amazon S3.

Quando crei un processo di importazione, specifichi l'Amazon Resource Name (ARN) di questo ruolo per. `DataAccessRoleArn` Per ulteriori informazioni sui ruoli IAM e sulle policy di fiducia, consulta IAM Roles.

Dopo aver impostato l'autorizzazione, sei pronto per importare i file nel tuo data store con un processo di importazione. Per ulteriori informazioni, consulta Avvio di un processo di importazione FHIR.

Per configurare le autorizzazioni di importazione

1. Se non l'hai già fatto, crea un bucket Amazon S3 di destinazione per i file di log di output. Il bucket Amazon S3 deve trovarsi nella stessa AWS regione del servizio e l'opzione Block Public Access deve essere attivata per tutte le opzioni. Per ulteriori informazioni, consulta Usare Amazon S3 per bloccare l'accesso pubblico. Per la Amazon-owned crittografia è necessario utilizzare anche una chiave KMS di proprietà del cliente. Per ulteriori informazioni sull'uso delle chiavi KMS, consulta Amazon Key Management Service.
2. Crea un ruolo di servizio di accesso ai dati HealthLake e concedi al HealthLake servizio l'autorizzazione ad assumerlo con la seguente politica di fiducia. HealthLake lo usa per scrivere il bucket Amazon S3 di output.

   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. Aggiungi una politica di autorizzazioni al ruolo di accesso ai dati che gli consenta di accedere al bucket Amazon S3. Sostituiscilo `amzn-s3-demo-bucket` con il nome del tuo 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"
    }]
}
```

Impostazione delle autorizzazioni per i lavori di esportazione

Prima di esportare file da un data store, devi concedere l' HealthLake autorizzazione per accedere al tuo bucket di output in Amazon S3. Per concedere HealthLake l'accesso, devi creare un ruolo di servizio IAMHealthLake, aggiungere una policy di fiducia al ruolo per concedere le autorizzazioni di HealthLake assunzione del ruolo e allegare una policy di autorizzazione al ruolo che gli consenta l'accesso al tuo bucket Amazon S3.

Se hai già creato un ruolo per HealthLake, puoi riutilizzarlo e concedergli le autorizzazioni aggiuntive per il tuo bucket di esportazione Amazon S3 elencate in questo argomento. Per ulteriori informazioni sui ruoli IAM e sulle policy di fiducia, consulta IAM Policies and Permissions.

> **Importante**
>
> HealthLake supporta sia le richieste di esportazione SDK native sia il funzionamento FHIR R4. `$export` È necessario fornire azioni IAM separate a seconda dell'API di esportazione che si decide di utilizzare. Ciò consente di gestire `allow` le `deny` autorizzazioni separatamente. Se desideri limitare le esportazioni di API REST HealthLake SDK e FHIR, devi applicare le autorizzazioni di negazione alle azioni IAM separate. Le modifiche alle autorizzazioni degli utenti IAM non sono necessarie se concedi agli utenti l'accesso completo a. HealthLake
>
> Utilizzo AWS CLI and AWS SDK:
>
> Le seguenti HealthLake azioni native sono disponibili per esportare dati da un data store utilizzando gli SDK AWS CLI and AWS :
>
> - `StartFHIRExportJob`

- DescribeFHIRExportJob
- ListFHIRExportJobs

Utilizzo delle API FHIR:

Le seguenti azioni IAM sono disponibili per esportare dati da un HealthLake data store e per annullare (eliminare) un processo di esportazione utilizzando l'operazione FHIR: $export POST:

- StartFHIRExportJobWithPost

GET:

- StartFHIRExportJobWithGet
- DescribeFHIRExportJobWithGet
- GetExportedFile

DELETE:

- CancelFHIRExportJobWithDelete

L'utente o il ruolo che imposta le autorizzazioni deve avere l'autorizzazione a creare ruoli, creare politiche e allegare politiche ai ruoli. La seguente policy IAM concede queste autorizzazioni.

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"
                }
            }
        }
    ]
}
```

Per impostare le autorizzazioni di esportazione

1. Se non l'hai già fatto, crea un bucket Amazon S3 di destinazione per i dati che esporterai dal tuo data store. Il bucket Amazon S3 deve trovarsi nella stessa regione AWS del servizio e Block Public Access deve essere attivato per tutte le opzioni. Per ulteriori informazioni, consulta Usare Amazon S3 per bloccare l'accesso pubblico. Per la Amazon-owned crittografia è necessario utilizzare anche una chiave KMS di proprietà del cliente. Per ulteriori informazioni sull'uso delle chiavi KMS, consulta Amazon Key Management Service.
2. Se non l'hai già fatto, crea un ruolo di servizio di accesso ai dati HealthLake e concedi al HealthLake servizio l'autorizzazione ad assumerlo con la seguente politica di fiducia. HealthLakelo usa per scrivere il bucket Amazon S3 di output. Se ne hai già creato unoImpostazione delle autorizzazioni per i lavori di importazione, puoi riutilizzarlo e concedergli le autorizzazioni per il tuo bucket Amazon S3 nel passaggio successivo.

   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. Aggiungi una politica di autorizzazioni al ruolo di accesso ai dati che gli consenta di accedere al tuo bucket Amazon S3 di output. Sostituiscilo `amzn-s3-demo-bucket` con il nome 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"
    }]
}
```

Installa il AWS CLI

AWS CLI È necessario per descrivere ed HealthLake elencare le proprietà del lavoro di importazione ed esportazione. Puoi richiedere queste informazioni anche utilizzando gli HealthLake SDK.

Per configurare il AWS CLI

1. Scarica e configura la AWS CLI. Per le istruzioni, consulta i seguenti argomenti nella Guida per l'utente di AWS Command Line Interface .

   - Installazione o aggiornamento della versione più recente di AWS CLI
   - Guida introduttiva a AWS CLI

2. Nel AWS CLI `config` file, aggiungi un profilo denominato per l'amministratore. Questo profilo viene utilizzato quando si eseguono i AWS CLI comandi. In base al principio di sicurezza del privilegio minimo, ti consigliamo di creare un ruolo IAM separato con privilegi specifici per le attività eseguite. Per ulteriori informazioni sui profili denominati, consulta Configurazione e impostazioni dei file di credenziali nella Guida per l'AWS Command Line Interface utente.

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

3. Verificate la configurazione utilizzando il seguente `help` comando.

   ```
   aws healthlake help
   ```

Se AWS CLI è configurato correttamente, viene visualizzata una breve descrizione AWS HealthLake e un elenco dei comandi disponibili.

AWS HealthLake tutorial

Obiettivo

In questo tutorial, importerai i dati FHIR R4 in un HealthLake data store utilizzando azioni native HealthLake. Successivamente, gestirete (creazione, lettura, aggiornamento, eliminazione) una risorsa FHIR utilizzando FHIR. RESTful APIs Per concludere il tutorial, esporterete i dati FHIR utilizzando azioni native. HealthLake

Prerequisiti

Tutte le procedure elencate in Configurazione sono necessarie per completare questo tutorial.

Passaggi del tutorial

1. Avvia il processo di importazione FHIR
2. Ottieni le proprietà dei lavori di importazione FHIR
3. Crea una risorsa FHIR
4. Leggi la risorsa FHIR
5. Aggiorna la risorsa FHIR
6. Elimina la risorsa FHIR
7. Esporta dati FHIR
8. Eliminare l'archivio dati

Gestione degli archivi dati con AWS HealthLake

Con AWS HealthLake, crei e gestisci archivi di dati per le risorse FHIR R4. Quando si crea un archivio HealthLake dati, un archivio di dati FHIR viene reso disponibile tramite un endpoint API. RESTful Puoi scegliere di importare (precaricare) i dati sanitari FHIR R4 open source di Synthea nel tuo archivio dati al momento della creazione. Per ulteriori informazioni, consulta Tipi di dati precaricati.

> **Importante**
>
> HealthLake supporta due tipi di strategie di autorizzazione dell'archivio dati FHIR, AWS SigV4 o SMART su FHIR. È necessario scegliere una delle strategie di autorizzazione prima di creare un HealthLake archivio dati FHIR. Per ulteriori informazioni, consulta Strategia di autorizzazione all'archiviazione dei dati.

Per trovare le funzionalità (comportamenti) relative al FHIR di un HealthLake data store attivo, recuperate la relativa dichiarazione di capacità.

I seguenti argomenti descrivono come utilizzare le azioni native del HealthLake cloud per creare, descrivere, elencare, etichettare ed eliminare gli archivi di dati FHIR utilizzando, e. AWS CLI AWS SDKs Console di gestione AWS

Argomenti

- Creazione di un archivio HealthLake dati
- Ottenere le proprietà del HealthLake data store
- Elencare archivi di HealthLake dati
- Etichettatura degli HealthLake archivi dati
- Eliminazione di un archivio HealthLake dati

Creazione di un archivio HealthLake dati

Utilizzato `CreateFHIRDatastore` per creare un AWS HealthLake data store conforme alla specifica FHIR R4. HealthLake gli archivi dati vengono utilizzati per importare, gestire, cercare ed esportare dati FHIR. Puoi scegliere di importare (precaricare) i dati sanitari FHIR R4 open source di

Synthea nel tuo archivio dati al momento della creazione. Per ulteriori informazioni, consulta Tipi di dati precaricati.

> **Importante**
>
> HealthLake supporta due tipi di strategie di autorizzazione dell'archivio dati FHIR, AWS SigV4 o SMART su FHIR. È necessario scegliere una delle strategie di autorizzazione prima di creare un HealthLake archivio dati FHIR. Per ulteriori informazioni, consulta Strategia di autorizzazione all'archiviazione dei dati.

Quando si crea un HealthLake data store, un archivio di dati FHIR viene reso disponibile tramite un RESTful endpoint API. Dopo aver creato il tuo HealthLake data store, puoi richiederne la dichiarazione di capacità per trovare tutte le funzionalità (comportamenti) associate al FHIR.

I menu seguenti forniscono esempi per AWS CLI AWS SDKs e una procedura per. Console di gestione AWS Per ulteriori informazioni, consulta `CreateFHIRDatastore` nella documentazione di riferimento dell'API AWS HealthLake .

Per creare un archivio HealthLake dati

Scegli un menu in base alle tue preferenze di accesso a AWS HealthLake.

AWS CLI e SDKs

CLI

AWS CLI

Esempio 1: creare un data store abilitato per SIGV4 HealthLake

L'`create-fhir-datastore`esempio seguente mostra come creare un nuovo archivio dati in. AWS HealthLake

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

Output:

```
{
```

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

Esempio 2: creare uno SMART su un data store abilitato per FHIR HealthLake

L'`create-fhir-datastore`esempio seguente mostra come creare un nuovo SMART su un data store abilitato per FHIR in. 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
```

Contenuto di `identity_provider_configuration.json`:

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

```
}
```

Output:

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

- Per i dettagli sull'API, consulta Create FHIRDatastore in AWS CLI Command Reference.

Python

SDK per 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
```

Il codice seguente mostra un esempio di parametri per un archivio dati SMART su 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
            )
```

- Per i dettagli sull'API, consulta Create FHIRDatastore in AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

SAP ABAP

SDK per SAP ABAP

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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.
```

- Per i dettagli sull'API, consulta Create FHIRDatastore in AWS SDK for SAP ABAP API reference.

> **Esempio di disponibilità**
>
> Non riesci a trovare quello che ti serve? Richiedi un esempio di codice utilizzando il link Fornisci feedback nella barra laterale destra di questa pagina.

AWS Console

> **Nota**
>
> La procedura seguente crea un archivio HealthLake dati con autorizzazione AWS SigV4. La HealthLake console non supporta la creazione di un archivio dati SMART su FHIR.

Per creare un archivio HealthLake dati con autorizzazione AWS SIGv4

1. Accedi alla pagina Crea archivio dati sulla HealthLake console.
2. Scegli Crea Data Store.
3. Nella sezione Impostazioni Data Store, per Nome Data Store, specifica un nome.
4. (Facoltativo) Nella sezione Impostazioni Data Store, per Precarica dati di esempio, seleziona la casella di controllo per precaricare i dati Synthea. I dati Synthea sono un set di dati di esempio open source. Per ulteriori informazioni, consulta Tipi di dati precaricati Synthea per HealthLake.
5. Nella sezione Crittografia del Data Store, scegli Usa la chiave di proprietà di AWS (impostazione predefinita) o Scegli una chiave AWS KMS diversa (avanzata).
6. Nella sezione Tag - opzionale, puoi aggiungere tag al tuo data store. Per ulteriori informazioni sull'etichettatura del data store, consultaEtichettatura degli HealthLake archivi dati.
7. Scegli Crea Data Store.

   Lo stato del tuo data store è disponibile nella pagina Data stores.

Ottenere le proprietà del HealthLake data store

`DescribeFHIRDatastore`Utilizzato per ottenere le proprietà di un AWS HealthLake data store. I seguenti menu forniscono una procedura Console di gestione AWS e alcuni esempi di codice per AWS CLI and AWS SDKs. Per ulteriori informazioni, consulta `DescribeFHIRDatastore` nella documentazione di riferimento dell'API AWS HealthLake .

Per ottenere le proprietà di un archivio HealthLake dati

Scegli un menu in base alle tue preferenze di accesso a AWS HealthLake.

AWS CLI e SDKs

CLI

AWS CLI

Come descrivere un datastore FHIR

L'`describe-fhir-datastore`esempio seguente mostra come trovare le proprietà di un archivio dati in AWS HealthLake.

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

Output:

```
{
    "DatastoreProperties": {
        "PreloadDataConfig": {
            "PreloadDataType": "SYNTHEA"
        },
        "SseConfiguration": {
            "KmsEncryptionConfig": {
                "CmkType": "CUSTOMER_MANAGED_KMS_KEY",
                "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
            }
        },
        "DatastoreName": "Demo",
        "DatastoreArn": "arn:aws:healthlake:us-east-1:<AWS Account ID>:datastore/
<Data store ID>",
```

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

- Per i dettagli sull'API, consulta Descrivi FHIRDatastore in AWS CLI Command Reference.

Python

SDK per 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
```

- Per i dettagli sull'API, consulta Descrivi FHIRDatastore in AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

SAP ABAP

SDK per SAP ABAP

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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.
```

- Per i dettagli sull'API, consulta Descrivi FHIRDatastore in AWS SDK for SAP ABAP API reference.

> **Esempio di disponibilità**
>
> Non riesci a trovare quello che ti serve? Richiedi un esempio di codice utilizzando il link Fornisci feedback nella barra laterale destra di questa pagina.

AWS Console

1. Accedi alla pagina degli archivi dati sulla HealthLake console.
2. Scegli un data store.

   Viene visualizzata la pagina dei dettagli del Data Store e tutte le proprietà del HealthLake Data Store sono disponibili.

Elencare archivi di HealthLake dati

Utilizzalo `ListFHIRDatastores` per elencare tutti gli archivi HealthLake dati presenti nell'account di un utente, indipendentemente dallo stato del data store. I menu seguenti forniscono una procedura Console di gestione AWS e alcuni esempi di codice per AWS CLI and AWS SDKs. Per ulteriori informazioni, consulta `ListFHIRDatastores` nella documentazione di riferimento dell'API AWS HealthLake .

Per elencare tutti gli archivi HealthLake dati

Scegli un menu in base alle tue preferenze di accesso a AWS HealthLake.

AWS CLI e SDKs

CLI

AWS CLI

Come elencare i datastore FHIR

L'`list-fhir-datastores`esempio seguente mostra come utilizzare il comando e come gli utenti possono filtrare i risultati in base allo stato del data store in AWS HealthLake.

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

Output:

```
{
    "DatastorePropertiesList": [
    {
        "PreloadDataConfig": {
            "PreloadDataType": "SYNTHEA"
        },
        "SseConfiguration": {
            "KmsEncryptionConfig": {
                "CmkType": "CUSTOMER_MANAGED_KMS_KEY",
                "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
            }
        },
        "DatastoreName": "Demo",
        "DatastoreArn": "arn:aws:healthlake:us-east-1:<AWS Account ID>:datastore/
<Data store ID>",
        "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/
datastore/<Data store ID>/r4/",
        "DatastoreStatus": "ACTIVE",
        "DatastoreTypeVersion": "R4",
        "CreatedAt": 1603761064.881,
        "DatastoreId": "<Data store ID>",
        "IdentityProviderConfiguration": {
```

```
            "AuthorizationStrategy": "AWS_AUTH",
            "FineGrainedAuthorizationEnabled": false
        }
    }
    ]
}
```

- Per i dettagli sull'API, consulta List FHIRDatastores in AWS CLI Command Reference.

Python

SDK per 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
```

- Per i dettagli sull'API, consulta List FHIRDatastores in AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

SAP ABAP

SDK per SAP ABAP

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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.
```

- Per i dettagli sulle API, consulta List FHIRDatastores in AWS SDK for SAP ABAP API reference.

> **Esempio di disponibilità**
>
> Non riesci a trovare quello che ti serve? Richiedi un esempio di codice utilizzando il link Fornisci feedback nella barra laterale destra di questa pagina.

AWS Console

- Accedi alla pagina degli archivi dati sulla HealthLake console.

  Tutti gli archivi HealthLake dati sono elencati nella sezione Archivi dati.

Etichettatura degli HealthLake archivi dati

Puoi assegnare metadati agli archivi HealthLake dati sotto forma di tag. Ogni tag è un'etichetta composta da una chiave e un valore definiti dall'utente. I tag consentono di gestire, identificare, organizzare, cercare e filtrare gli archivi di dati.

> **Importante**
>
> Non archiviate nei tag informazioni sanitarie protette (PHI), informazioni di identificazione personale (PII) o altre informazioni riservate o sensibili. I tag non sono destinati ad essere utilizzati per dati privati o sensibili.

I seguenti argomenti descrivono come utilizzare le operazioni di HealthLake etichettatura utilizzando, e. Console di gestione AWS AWS CLI AWS SDKs Per ulteriori informazioni, consulta Taggare le AWS risorse nella Riferimenti generali di AWS Guida.

Argomenti

- Taggare un archivio dati HealthLake
- Elencare i tag per un archivio HealthLake dati
- Rimuovere il tag di un archivio dati HealthLake

Taggare un archivio dati HealthLake

Utilizzato `TagResource` per etichettare un archivio HealthLake dati. I seguenti menu forniscono una procedura Console di gestione AWS e alcuni esempi di codice per AWS CLI and AWS SDKs. Per ulteriori informazioni, consulta `TagResource` nella documentazione di riferimento dell'API AWS HealthLake .

Per etichettare un archivio HealthLake dati

Scegli un menu in base alle tue preferenze di accesso a AWS HealthLake.

AWS CLI e SDKs

CLI

AWS CLI

Come aggiungere un tag al datastore

L'esempio `tag-resource` seguente mostra come aggiungere un tag al datastore.

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

Questo comando non produce alcun output.

.

- Per i dettagli sull'API, consulta TagResource AWS CLICommand Reference.

Python

SDK per 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
```

- Per i dettagli sull'API, consulta TagResource AWSSDK for Python (Boto3) API Reference.

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

SAP ABAP

SDK per SAP ABAP

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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.
```

- Per i dettagli sulle API, TagResourceconsulta AWS SDK for SAP ABAP API reference.

> **Esempio di disponibilità**
>
> Non riesci a trovare quello che ti serve? Richiedi un esempio di codice utilizzando il link Fornisci feedback nella barra laterale destra di questa pagina.

AWS Console

1. Accedi alla pagina degli archivi dati sulla HealthLake console.
2. Scegli un data store.

   Viene visualizzata la pagina dei dettagli del Data store.
3. Nella sezione Tag, scegli Gestisci tag.

   Si apre la pagina Gestisci tag.
4. Scegli Aggiungi nuovo tag.
5. Inserisci una chiave e un valore (opzionale).
6. Scegli Save (Salva).

Elencare i tag per un archivio HealthLake dati

`ListTagsForResource`Da utilizzare per elencare i tag per un archivio HealthLake dati. I seguenti menu forniscono una procedura Console di gestione AWS e alcuni esempi di codice per AWS CLI and AWS SDKs. Per ulteriori informazioni, consulta `ListTagsForResource` nella documentazione di riferimento dell'API AWS HealthLake .

Per elencare i tag per un archivio HealthLake dati

Scegli un menu in base alle tue preferenze di accesso a AWS HealthLake.

AWS CLI e SDKs

CLI

AWS CLI

Come elencare i tag di un datastore

L'esempio `list-tags-for-resource` seguente elenca i tag associati al datastore specificato:

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

Output:

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

- Per i dettagli sull'API, consulta ListTagsForResource AWS CLICommand Reference.

Python

SDK per 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
```

- Per i dettagli sull'API, consulta ListTagsForResource AWSSDK for Python (Boto3) API Reference.

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

SAP ABAP

SDK per SAP ABAP

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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.
```

- Per i dettagli sulle API, ListTagsForResourceconsulta AWS SDK for SAP ABAP API reference.

> **Esempio di disponibilità**
>
> Non riesci a trovare quello che ti serve? Richiedi un esempio di codice utilizzando il link Fornisci feedback nella barra laterale destra di questa pagina.

AWS Console

1. Accedi alla pagina degli archivi dati sulla HealthLake console.
2. Scegli un data store.

   Viene visualizzata la pagina dei dettagli del Data store. Nella sezione Tag, sono elencati tutti i tag del data store.

Rimuovere il tag di un archivio dati HealthLake

Utilizzato `UntagResource` per rimuovere un tag da un HealthLake data store. I seguenti menu forniscono una procedura Console di gestione AWS e alcuni esempi di codice per AWS CLI and AWS SDKs. Per ulteriori informazioni, consulta `UntagResource` nella documentazione di riferimento dell'API AWS HealthLake .

Per rimuovere i tag da un archivio dati HealthLake

Scegli un menu in base alle tue preferenze di accesso a AWS HealthLake.

AWS CLI e SDKs

CLI

AWS CLI

Come rimuovere i tag da un datastore.

L'esempio `untag-resource` seguente mostra come rimuovere i tag da un datastore.

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

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta UntagResource AWS CLICommand Reference.

Python

SDK per 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
```

- Per i dettagli sull'API, consulta UntagResource AWSSDK for Python (Boto3) API Reference.

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

SAP ABAP

SDK per SAP ABAP

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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.
```

- Per i dettagli sulle API, UntagResourceconsulta AWS SDK for SAP ABAP API reference.

> **Esempio di disponibilità**
>
> Non riesci a trovare quello che ti serve? Richiedi un esempio di codice utilizzando il link Fornisci feedback nella barra laterale destra di questa pagina.

AWS Console

1. Accedi alla pagina degli archivi dati sulla HealthLake console.
2. Scegli un data store.

   Viene visualizzata la pagina dei dettagli del Data store.
3. Nella sezione Tag, scegli Gestisci tag.

   Si apre la pagina Gestisci tag.
4. Scegli Rimuovi accanto al tag che desideri rimuovere.
5. Scegli Save (Salva).

Eliminazione di un archivio HealthLake dati

Utilizzare `DeleteFHIRDatastore` per eliminare un HealthLake data store. I seguenti menu forniscono una procedura Console di gestione AWS e alcuni esempi di codice per AWS CLI and

AWS SDKs. Per ulteriori informazioni, consulta `DeleteFHIRDatastore` nella documentazione di riferimento dell'API AWS HealthLake .

Per eliminare un archivio HealthLake dati

Scegli un menu in base alle tue preferenze di accesso a AWS HealthLake.

AWS CLI e SDKs

CLI

AWS CLI

Come eliminare un datastore FHIR

L'`delete-fhir-datastore`esempio seguente mostra come eliminare un archivio dati e tutto il suo contenuto. AWS HealthLake

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

Output:

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

- Per FHIRDatastore i dettagli sull'AWS CLI API, consulta Delete in Command Reference.

Python

SDK per 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
```

- Per i dettagli sull'API, consulta Delete FHIRDatastore in AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

SAP ABAP

SDK per SAP ABAP

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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.
```

- Per i dettagli sull'API, consulta Delete FHIRDatastore in AWS SDK for SAP ABAP API reference.

Esempio di disponibilità

Non riesci a trovare quello che ti serve? Richiedi un esempio di codice utilizzando il link Fornisci feedback nella barra laterale destra di questa pagina.

AWS Console

1. Accedi alla pagina degli archivi dati sulla HealthLake console.
2. Scegli un data store.

   Viene visualizzata la pagina dei dettagli del Data store.
3. Scegli Elimina.

   Viene visualizzata la pagina Elimina data store.
4. Per confermare l'eliminazione del data store, inserisci il nome del data store nel campo di immissione del testo.
5. Scegli Delete (Elimina).

Gestione degli abbonamenti FHIR in AWS HealthLake

AWS HealthLake supporta gli abbonamenti FHIR, che consentono di ricevere notifiche in tempo reale quando si verificano modifiche specifiche ai dati sanitari. Questa funzionalità implementa il modello di abbonamento tematico FHIR R5 Backport, offrendo una maggiore scalabilità e flessibilità rispetto al tradizionale modello di abbonamento FHIR R4.

Con gli abbonamenti FHIR, è possibile creare applicazioni sanitarie basate sugli eventi che rispondono immediatamente ai cambiamenti dei dati clinici, consentendo interventi tempestivi, flussi di lavoro automatizzati e un migliore coordinamento dell'assistenza.

Argomenti

- Come funzionano gli abbonamenti FHIR
- Componenti chiave
- Best practice
- Il ciclo di vita dell'abbonamento FHIR con AWS HealthLake
- Creazione di un abbonamento FHIR con AWS HealthLake
- Ricerca di abbonamenti FHIR con AWS HealthLake
- Filtraggio delle notifiche con AWS HealthLake

Come funzionano gli abbonamenti FHIR

Gli abbonamenti FHIR HealthLake funzionano secondo un modello tematico in cui:

1. Crea argomenti per definire gli eventi: crea argomenti di abbonamento che specificano gli eventi che possono attivare le notifiche
2. Ti iscrivi: crea sottoscrizioni a questi argomenti con criteri di filtro specifici
3. HealthLake monitora: il servizio monitora continuamente gli eventi che soddisfano i tuoi criteri
4. Notifiche inviate: si verificano eventi CWhen corrispondenti, HealthLake invia notifiche tramite il canale prescelto

Componenti chiave

Gli abbonamenti FHIR sono costituiti dai seguenti componenti.

Argomenti di abbonamento

Gli argomenti di abbonamento sono alla base del sistema di notifica e definiscono:

- Eventi di attivazione: quali modifiche attivano le notifiche (ad esempio: creazione di risorse, aggiornamenti, eliminazioni)
- Filtri disponibili: quali opzioni di filtro sono disponibili per gli abbonati
- Contenuto della notifica: quali dati sono inclusi nelle notifiche

La tabella seguente elenca i tipi di argomenti più comuni.

Tipo di evento	Description	Casi di utilizzo comune
Creazione di risorse	Attivato quando vengono create le risorse	Nuova registrazione del paziente, nuova osservazione registrata
Aggiornamenti delle risorse	Attivato quando le risorse vengono modificate	Modifiche di stato, aggiornam enti clinici
Eliminazione delle risorse	Attivato quando le risorse vengono eliminate	Controllo e monitoraggio della conformità

Sottoscrizioni

Un abbonamento è la tua richiesta di ricevere notifiche per eventi specifici definiti da un argomento di abbonamento. Ogni abbonamento include:

- Riferimento all'argomento: specifica a quale argomento di abbonamento ti stai abbonando
- Filtri: criteri per selezionare quali eventi generano notifiche
- Configurazione del canale: dove e come devono essere inviate le notifiche
- Preferenze relative al payload: quale livello di dettaglio deve essere incluso nelle notifiche

Canali di notifica

HealthLake supporta i seguenti canali di notifica:

Tipo di canale	Casi d'uso	
EventBridge	Integrazioni aziendali, flussi di lavoro serverless, orchestrazione tra servizi AWS	
REST Hook	Notifiche dirette sugli endpoint, integrazione di sistemi di terze parti	

Payload di notifica

Scegli il tipo di payload appropriato in base alle tue esigenze:

Tipo di payload	Description	Considerazioni relative alla sicurezza
Solo ID	Contiene solo identificatori di risorse	Esposizione minima al PHI
Risorsa completa	Contiene risorse complete con una dimensione massima di 256 KB. Se la dimensione è superiore a 256 KB, tornerà alla modalità Solo ID	Contiene PHI; verifica la gestione sicura

Best practice

Ottimizzazione delle prestazioni

- Utilizza filtri mirati: restringi i criteri per ricevere solo le notifiche essenziali

- Scegliete i tipi di payload appropriati: utilizzate payload solo ID quando possibile per prestazioni migliori
- Implementa ricevitori efficienti: assicurati che i destinatari delle notifiche elaborino rapidamente i messaggi

Considerazioni relative alla sicurezza

- Endpoint sicuri: implementa l'autenticazione corretta per gli endpoint REST Hook
- Protezione PHI: fai attenzione ai payload con risorse complete poiché contengono PHI
- Controllo degli accessi: limita la creazione di abbonamenti solo agli utenti autorizzati

Eccellenza operativa

- Imposta le date di fine appropriate: utilizza le date di fine per gli abbonamenti temporanei
- Monitora lo stato dell'abbonamento: controlla regolarmente lo stato dei tuoi abbonamenti
- Implementa la gestione degli errori: progetta le tue applicazioni per gestire gli errori di recapito delle notifiche

Il ciclo di vita dell'abbonamento FHIR con AWS HealthLake

Segui questi passaggi per comprendere il ciclo di vita dell'abbonamento FHIR:

1. Creazione di una `SubscriptionTopic`
   - Crea uno stato con `SubscriptionTopic` `"unknown"`
2. Creazione di una `Subscription`
   - Crea un `Subscription` con stato `"requested"`
   - HealthLake convalida la configurazione `Subscription`
   - `Subscription`deve fare riferimento a un argomento già esistente (l'argomento deve essere in status`unknown`,`draft`,`active`).
3. Attivazione
   - Se valido, HealthLake aggiorna lo stato `Subscription` di `"active"`
   - Durante la creazione di un`Subscription`, se l'argomento specificato era in stato`"unknown"`, HealthLake aggiorna lo stato a `"active"` una volta che anche l'abbonamento è attivo
   - Gli abbonamenti richiedono in genere 5-10 minuti per essere creati correttamente

- Se `Subscription` la creazione non viene completata correttamente, lo stato passerà al `error` punto in cui è necessario eseguire un'operazione DELETE, quindi riprovare a creare un abbonamento. Puoi visualizzare il `"error"` campo nella risorsa Abbonamento per vedere perché l'abbonamento non è stato creato correttamente.

4. L'ingestione durante l'abbonamento è `active`
5. Mentre è `Subscription` `active`
   - HealthLake monitora gli eventi che corrispondono ai tuoi criteri
   - Le notifiche vengono inviate all'endpoint configurato quando si verificano delle corrispondenze
6. Gestione errori
   - HealthLake tenta di riprovare per 14 giorni e poi smette di riprovare per quegli eventi
7. Disattivazione
   - A `Subscription` può essere disattivato tramite:

     Impostazione di una data di fine (disattivazione automatica)

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

Eliminazione della risorsa `Subscription`

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

Creazione di un abbonamento FHIR con AWS HealthLake

La seguente guida mostra come creare un abbonamento FHIR utilizzando. AWS HealthLake

Per creare un abbonamento FHIR

1. Creare un `SubscriptionTopic`.

   Esempio di risorsa Subscription Topic:

```
{
  "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 il tuo endpoint di notifica (canale personalizzato). I passaggi seguenti sono necessari per garantire che l'endpoint riceva le notifiche

   Quando si utilizza REST Hook

   - `events.amazonaws.com`Affidati alla tua politica delle chiavi KMS se utilizzi il datastore CM_CMK.
   - Se utilizzi un datastore CM_CMK, devi aggiungere il tag alla tua chiave KMS con il valore di `EventBridgeApiDestinations` `true`
   - HealthLake utilizza per autenticare l'endpoint REST OAuth Hook. Pertanto, quando si crea un abbonamento a un hook REST, è necessario inserire un client-id, un client-secret e un canale. oAuth-endpoint-url _type.extension [*].

   Esempio di politica chiave KMS se si utilizza il datastore CM_CMK:

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

   Quando si utilizza EventBridge

   - `events.amazonaws.com`Affidati alla politica delle chiavi KMS se utilizzi il datastore CM_CMK.
   - Verifica che la tua policy EventBridge sulle risorse si attesti come responsabile del servizio. `healthlake.amazonaws.com`
   - Se utilizzi CM_CMK ed EventBridge è l'endpoint, verifica di crittografare il EventBridge bus con la stessa chiave KMS della chiave KMS del datastore.
   - Verifica che il tuo EventBridge Bus abbia almeno una regola che corrisponda agli eventi generati da. HealthLake

   Esempio di politica delle risorse per EventBridge Channel Bus:

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

Esempio di EventBridge regola event-pattern per ricevere eventi da: HealthLake

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

> **Note**
>
> HealthLake supporta 2 fonti:
>
> - `"healthlake"`: Solo per gli abbonamenti.
> - `"aws.healthlake"`: per ricevere eventi HealthLake di servizio.
>
> Utilizza `"healthlake"` come origine per la creazione di una regola per i bus di eventi FHIR Subscriptions.

3. Crea il tuo `Subscription`

Invia una risorsa in abbonamento con:

- Stato: `"requested"`
- Riferimento all'`SubscriptionTopic`ID prescelto
- Criteri di filtro. Per ulteriori informazioni, consulta Filtraggio delle notifiche per i filtri supportati.
- Configurazione del canale

Esempi di payload di abbonamento

I seguenti esempi di codice mostrano come creare payload in abbonamento.

EventBridge

Abbonamento con `event-bridge` canale e tipo di `id-only` payload.

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

```
    }
  }
}
```

Abbonamento con `event-bridge` endpoint e tipo di `full-resource` payload.

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

Rest Hook

Abbonamento con `rest-hook` endpoint e tipo di `id-only` payload.

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

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

Abbonamento con `rest-hook` canale e tipo di `full-resource` payload.

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

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

Esempi di payload di notifica

Durante la creazione di un abbonamento, HealthLake verifica la corretta configurazione dell'abbonamento inviando un pacchetto di handshake al canale configurato. Il payload seguente è un esempio di pacchetto handshake.

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

Esempio di pacchetto di notifiche con solo 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"
              }
            ]
          }
        }
      ]
    }
  }
}
```

Esempio di pacchetto di notifiche completo di risorse.

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

Controllo delle versioni degli eventi

HealthLake supporta la cronologia FHIR per impostazione predefinita.

Per sapere quale versione della risorsa hai ricevuto nel pacchetto di notifiche:

- risorsa completa: poiché i pacchetti completi di risorse includono l'intera risorsa, la versione verrà inclusa all'interno del pacchetto `entry[*]` per ogni risorsa inclusa nel pacchetto.
- id-only: i pacchetti non includeranno alcuna informazione sulle risorse. HealthLakeinclude la versione corrispondente e inclusa nel pacchetto tramite il campo. `entry[0].notificationEvent[*].additionalContext` Questo campo è nel formato.

`<ResourceType>/<ResourceId>/_history/<Version Id>` Per ulteriori informazioni, consulta il campo AdditionalContext nell'esempio id-only payload.

Rilevamento della duplicazione degli eventi

HealthLakela funzionalità di abbonamento FHIR garantisce almeno una consegna. Ciò significa che potresti ricevere lo stesso evento più volte, nello stesso pacchetto o in un pacchetto diverso. Per identificare i duplicati, HealthLake fornisce un ID univoco per ogni evento nel pacchetto di notifiche in. `entry[0].notificationEvent[*].id`

Questo ID è univoco per la versione specifica dell'evento che è stata abbinata e consegnata. Ad esempio, se lo stesso Encounter viene aggiornato due volte ed entrambi gli aggiornamenti corrispondono ai criteri del filtro, riceverai due eventi separati con lo stesso riferimento all'incontro. Avranno lo stesso`notificationEvent[*].focus`, ma ne avranno uno unico`notificationEvent[*].id`. Inoltre, questi eventi possono essere inviati in pacchetti separati o all'interno dello stesso pacchetto di notifiche.

Ricerca di abbonamenti FHIR con AWS HealthLake

`Subscription`e `SubscriptionTopic` le risorse sono anche ricercabili. HealthLake supporta tutti i parametri di ricerca più comuni per Abbonamento e SubscriptionTopic risorse.

Inoltre, supportiamo funzionalità di ricerca aggiuntive tramite i seguenti parametri:

Subscription

Parametro di ricerca	Description	Esempio
contact	Cerca nel campo subscription.CONTACT nelle specifiche di base di R4	`Subscription?contact=phone`
criteria	Cerca nel campo subscription.criteria nella specifica di base R4	`Subscription?criteria=[baseUrl]/datastore/[datastoreId]/r4/SubscriptionTopic/[topicId]`

Parametro di ricerca	Description	Esempio
payload	Cerca nel campo subscription.channel.payload nella specifica di base R4	`Subscription?payload=application/fhir+json`
status	Cerca nel campo subscription.status nella specifica di base R4	`Subscription?status=error`
tipo	Cerca nel campo subscription.channel.type nella specifica di base R4	`Subscription?topic=event-bridge`
url	Cerca nel campo subscription.channel.Endpoint nella specifica di base R4	`Subscription?url=[Subscription.channel.endpoint]`
criteri di filtro	Cerca nel campo di estensione dei criteri con l'url "http://hl7. org/fhir/uv/subscriptions-backport/StructureDefinition/backport-filter-criteria» come stringa	`Subscription?filter-criteria=Encounter?`
canale personalizzato	Cerca nel campo di estensione del tipo di canale personalizzato con l'url "http://hl7. org/fhir/uv/subscriptions-backport/StructureDefinition/backport-channel-type» come codifica Esempio di payload dell'abbonamento	`Subscription?custom-channel=[System]|[code]`

Parametro di ricerca	Description	Esempio
tipo di payload	Cerca nel campo di estension e del tipo di payload in cui specifichi come è formattato il payload (o) `id-only` `full-resource`	`Subscription?payload-type=full-resource`
topic	Eseguite la ricerca nel campo subscription.criteria, in cui viene aggiunto l'argomento	`Subscription?topic=[topicId]`

SubscriptionTopic

Parametro di ricerca	Description	Esempio
data	Cerca nel campo della data nella SubscriptionTopic risorsa	`SubscriptionTopic?date=[SubscriptionTopic.date]`
derived-or-self	Effettua la ricerca `derivedFrom` nei campi `url` o nella SubscriptionTopic risorsa	`SubscriptionTopic?derived-or-self=[SubscriptionTopic.url | SubscriptionTopic.derivedFrom]`
identificatore	Cerca nel campo Subscript oinTopic .identifier	`SubscriptionTopic?identifier=[SubscriptionTopic.identifier]`
risorsa	Cerca nel campo Subscript ionTopic .resourceTrigger.r esource	`SubscriptionTopic?resource=Encounter`

Parametro di ricerca	Description	Esempio
status	Effettua una ricerca sullo stato del SubscriptionTopic	`SubscriptionTopic?status=unknown`
titolo	Cerca nel titolo un SubscriptionTopic	`SubscriptionTopic?title=admission`
descrizione del trigger	Cerca su .resourceTrigger.description SubscriptionTopic	`SubscriptionTopic.trigger-description=resource moving to state 'in-progress'`
url	Cerca nell'URL il SubscriptionTopic	`SubscriptionTopic?url=[SubscriptionTopic.url]`
version	Cerca nella versione il SubscriptionTopic	`SubscriptionTopic?version=1`

Filtraggio delle notifiche con AWS HealthLake

Perfeziona le tue notifiche utilizzando i parametri di ricerca FHIR standard nei tuoi `Subscription` criteri.

Filtri supportati

La tabella seguente mostra un elenco di criteri di filtro HealthLake supportati per gli abbonamenti.

Tipi di parametri di ricerca	Tipo di dati FHIR	Modifiers (Modificatori)	Prefissi supportati
Stringa	stringa HumanName Indirizzo	esatto, contiene elementi mancanti	
Token	booleano	no, testo, mancante	

Tipi di parametri di ricerca	Tipo di dati FHIR	Modifiers (Modificatori)	Prefissi supportati
	code stringa Codifica CodeableConcept Identificatore ContactPoint id		
Numero	intero decimal Int positivo Int senza segno	perso	«eq», «ne», «gt», «lt», «ge», «le», «sa», «eb», «ap»
Data	data dateTime istante Periodo Timing (Tempo)		«eq», «ne», «gt», «lt», «ge», «le», «sa», «eb», «ap»
Quantità	Quantità Soldi Intervallo SimpleQuantity		«eq», «ne», «gt», «lt», «ge», «le», «sa», «eb», «ap»

Tipi di parametri di ricerca	Tipo di dati FHIR	Modifiers (Modifica tori)	Prefissi supportati
Riferimento	Documentazione di riferimento	mancante, identific atore, tipo	
URI	uri url canonico uid oid	perso	

Esempi di filtri supportati

La tabella seguente mostra esempi di criteri di filtro HealthLake supportati per gli abbonamenti:

Scopo	Criteri di filtro	Description
Osservazioni specifiche per il paziente	`Observation?patient=Patient/[id]&status=final`	Ricevi notifiche quando le osservazioni per un paziente specifico vengono completate
Osservazioni specifiche per il paziente	`Patient?birthdate=gt2021`	Ricevi notifiche quando i pazienti nati dopo il 2021 vengono registrati o aggiornati
Osservazioni specifiche per il paziente	`Condition?code=http://snomed.info/sct|39065001`	Ricevi notifiche relative a condizioni con codici SNOMED specifici
Osservazioni specifiche per il paziente	`Observation?code=http://loinc.org|8480-6&value-quantity=gt160`	Ricevi notifiche per i valori della pressione arteriosa sistolica elevata

Importazione di dati FHIR con AWS HealthLake

Dopo aver creato un HealthLake data store, il passaggio successivo consiste nell'importare i file da un bucket Amazon Simple Storage Service (S3). Puoi avviare un processo di importazione FHIR utilizzando Console di gestione AWS, AWS CLI o. AWS SDKs AWS HealthLake Utilizzate le azioni native per avviare, descrivere ed elencare i lavori di importazione FHIR.

> **Importante**
>
> HealthLake supporta la specifica FHIR R4 per lo scambio di dati sanitari. Se necessario, puoi collaborare con un AWS HealthLake partner per convertire i tuoi dati sanitari in formato FHIR R4 prima dell'importazione.

Quando avvii un processo di importazione FHIR, specifichi una posizione di input del bucket Amazon S3, una posizione di output del bucket Amazon S3 (per i risultati di elaborazione del lavoro), un ruolo IAM che concede HealthLake l'accesso ai bucket Amazon S3 e una chiave di proprietà o di proprietà del cliente. AWS AWS Key Management Service Per ulteriori informazioni, consulta Impostazione delle autorizzazioni per i lavori di importazione.

> **Note**
>
> Puoi mettere in coda i lavori di importazione. I lavori di importazione asincroni vengono elaborati in modo FIFO (First In First Out). È possibile mettere in coda i lavori nello stesso modo in cui si avviano i lavori di importazione. Se ne è in corso uno, si metterà semplicemente in coda. È possibile creare, leggere, aggiornare o eliminare risorse FHIR mentre è in corso un processo di importazione.

HealthLake genera un `manifest.json` file per ogni processo di importazione FHIR. Il file descrive sia i successi che gli errori di un processo di importazione FHIR. HealthLake invia il `manifest.json` file nel bucket Amazon S3 specificato all'avvio di un processo di importazione FHIR. I file di registro sono organizzati in due cartelle, denominate e. `SUCCESS` `FAILURE` Utilizzate il `manifest.json` file come primo passo per la risoluzione dei problemi relativi a un processo di importazione non riuscito, in quanto fornisce dettagli su ogni file.

```
{
```

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

Configurazione del livello di convalida per le importazioni

Quando si avvia un processo di importazione FHIR, è possibile facoltativamente specificare un `ValidationLevel` da applicare a ciascuna risorsa. AWS HealthLake attualmente supporta i seguenti livelli di convalida:

- `strict`: Le risorse vengono convalidate in base all'elemento del profilo della risorsa o alla specifica R4 se non è presente alcun profilo. Questo è il livello di convalida predefinito per. AWS HealthLake
- `structure-only`: Le risorse vengono convalidate rispetto a R4, ignorando i profili di riferimento.

- `minimal`: Le risorse vengono convalidate minimamente, ignorando alcune regole R4. Le risorse che non superano i controlli di struttura richiesti search/analytics verranno aggiornate per includere un avviso per l'audit.

Quando si importa utilizzando il livello di `minimal` convalida, è possibile generare file di registro aggiuntivi in una cartella denominata. `SUCCESS_WITH_SEARCH_VALIDATION_FAILURES` Le risorse all'interno dei file di registro di questa cartella sono state inserite nel datastore nonostante i controlli di convalida relativi alla ricerca non abbiano avuto esito positivo. Ciò implica che alcuni aspetti della risorsa FHIR non erano validi secondo FHIR e che i campi con formato errato potrebbero non essere ricercabili. A queste risorse verrà `extension` aggiunta una descrizione di tale errore.

Argomenti

- Avvio di un processo di importazione FHIR
- Ottenere proprietà lavorative importate da FHIR
- Elenco dei lavori di importazione FHIR

Avvio di un processo di importazione FHIR

Utilizzato `StartFHIRImportJob` per avviare un processo di importazione FHIR in un archivio HealthLake dati. I seguenti menu forniscono una procedura per Console di gestione AWS e alcuni esempi di codice per and. AWS CLI AWS SDKs Per ulteriori informazioni, consulta `StartFHIRImportJob` nella documentazione di riferimento dell'API AWS HealthLake .

> **Importante**
>
> HealthLake supporta la specifica FHIR R4 per lo scambio di dati sanitari. Se necessario, puoi collaborare con un AWS HealthLake partner per convertire i tuoi dati sanitari in formato FHIR R4 prima dell'importazione.

Come avviare un processo di importazione FHIR

Scegli un menu in base alle tue preferenze di accesso a. AWS HealthLake

AWS CLI e SDKs

CLI

AWS CLI

Come avviare un processo di importazione FHIR

L'`start-fhir-import-job`esempio seguente mostra come avviare un processo di importazione FHIR utilizzando AWS HealthLake.

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

Output:

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

- Per i dettagli sull'API, consulta Start FHIRImport Job in AWS CLI Command Reference.

Python

SDK per 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
```

- Per i dettagli sull'API, consulta Start FHIRImport Job in AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

SAP ABAP

SDK per SAP ABAP

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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.
```

- Per i dettagli sull'API, consulta Start FHIRImport Job in AWS SDK per il riferimento all'API SAP ABAP.

> **Esempio di disponibilità**
>
> Non riesci a trovare quello che ti serve? Richiedi un esempio di codice utilizzando il link Fornisci feedback nella barra laterale destra di questa pagina.

AWS Console

1. Accedi alla pagina degli archivi dati sulla HealthLake console.
2. Scegli un data store.
3. Scegli Importa.

   Viene visualizzata la pagina Importa.
4. Nella sezione Dati di input, inserisci le seguenti informazioni:

   - Posizione dei dati di input in Amazon S3

5. Nella sezione Importa file di output, inserisci le seguenti informazioni:

   - Importa la posizione dei file di output in Amazon S3

   - Importa la crittografia dei file di output

6. Nella sezione Autorizzazioni di accesso, scegli Usa un ruolo di servizio IAM esistente e seleziona il ruolo dal menu Service role name oppure scegli Crea un ruolo IAM.
7. Scegli Import data (Importa dati).

> **Note**
>
> Durante l'importazione, scegli Copia l'ID del lavoro sul banner nella parte superiore della pagina. È possibile utilizzare il `JobID`per richiedere le proprietà del lavoro di importazione utilizzando AWS CLI. Per ulteriori informazioni, consulta Ottenere proprietà lavorative importate da FHIR.

Ottenere proprietà lavorative importate da FHIR

Utilizzatelo `DescribeFHIRImportJob` per ottenere le proprietà del lavoro di importazione FHIR. I seguenti menu forniscono una procedura per Console di gestione AWS e alcuni esempi di codice per and. AWS CLI AWS SDKs Per ulteriori informazioni, consulta `DescribeFHIRImportJob` nella documentazione di riferimento dell'API AWS HealthLake .

Per ottenere FHIR, importate le proprietà del lavoro

Scegli un menu in base alle tue preferenze di accesso a AWS HealthLake.

AWS CLI e SDKs

CLI

AWS CLI

Come descrivere un processo di importazione FHIR

L'`describe-fhir-import-job`esempio seguente mostra come apprendere le proprietà di un processo di importazione FHIR utilizzando AWS HealthLake.

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

Output:

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

- Per i dettagli sull'API, consulta Descrivi FHIRImport Job in AWS CLI Command Reference.

Python

SDK per 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
```

- Per i dettagli sull'API, consulta Descrivi FHIRImport Job in AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

SAP ABAP

SDK per SAP ABAP

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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.
```

- Per i dettagli sull'API, consulta Descrivi FHIRImport Job in AWS SDK per il riferimento all'API SAP ABAP.

> **Esempio di disponibilità**
>
> Non riesci a trovare quello che ti serve? Richiedi un esempio di codice utilizzando il link Fornisci feedback nella barra laterale destra di questa pagina.

AWS Console

> **Note**
>
> Le informazioni sul processo di importazione FHIR non sono disponibili sulla HealthLake console. Utilizza invece AWS CLI with `DescribeFHIRImportJob` per richiedere proprietà

del processo di importazione come `JobStatus`. Per ulteriori informazioni, consulta l' AWS CLI esempio in questa pagina.

Elenco dei lavori di importazione FHIR

`ListFHIRImportJobs`Da utilizzare per elencare i lavori di importazione FHIR per un archivio HealthLake dati attivo. I seguenti menu forniscono una procedura per Console di gestione AWS e alcuni esempi di codice per and. AWS CLI AWS SDKs Per ulteriori informazioni, consulta `ListFHIRImportJobs` nella documentazione di riferimento dell'API AWS HealthLake .

Per elencare i lavori di importazione FHIR

Scegliete un menu in base alle vostre preferenze di accesso a AWS HealthLake.

AWS CLI e SDKs

CLI

AWS CLI

Come elencare tutti i processi di importazione FHIR

L'esempio `list-fhir-import-jobs` seguente mostra come utilizzare il comando per visualizzare un elenco di tutti i processi di importazione associati a un account.

```
aws healthlake list-fhir-import-jobs \
    --datastore-id (Data store ID) \
    --submitted-before (DATE like 2024-10-13T19:00:00Z) \
    --submitted-after (DATE like 2020-10-13T19:00:00Z ) \
    --job-name "FHIR-IMPORT" \
    --job-status SUBMITTED  \
    -max-results (Integer between 1 and 500)
```

Output:

```
{
    "ImportJobPropertiesList": [
        {
            "JobId": "c0fddbf76f238297632d4aebdbfc9ddf",
            "JobStatus": "COMPLETED",
```

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

- Per i dettagli sull'API, consulta List FHIRImport Jobs in AWS CLI Command Reference.

Python

SDK per 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
```

- Per i dettagli sull'API, consulta List FHIRImport Jobs in AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

SAP ABAP

SDK per SAP ABAP

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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.
```

- Per i dettagli sull'API, consulta il riferimento all'API List FHIRImport Jobs in AWS SDK for SAP ABAP.

> **Esempio di disponibilità**
>
> Non riesci a trovare quello che ti serve? Richiedi un esempio di codice utilizzando il link Fornisci feedback nella barra laterale destra di questa pagina.

AWS Console

> **Note**
>
> Le informazioni sul processo di importazione FHIR non sono disponibili sulla HealthLake console. Utilizzate invece AWS CLI with `ListFHIRImportJobs` per elencare tutti i processi di importazione FHIR. Per ulteriori informazioni, fate riferimento all' AWS CLI esempio in questa pagina.

Gestione delle risorse FHIR in AWS HealthLake

Utilizza le interazioni dell' RESTful API FHIR R4 per gestire le risorse FHIR in un archivio dati. HealthLake Le sezioni seguenti descrivono tutte le interazioni RESTful API FHIR R4 HealthLake supportate e disponibili per la gestione delle risorse FHIR. Per informazioni sulle funzionalità dell'archivio HealthLake dati e sulle parti della specifica FHIR supportate, vedere. Dichiarazione di capacità FHIR R4 per AWS HealthLake

> **Note**
>
> Le interazioni FHIR elencate in questo capitolo sono realizzate in conformità allo standard HL7 FHIR R4 per lo scambio di dati sanitari. Poiché sono rappresentazioni dei servizi HL7 FHIR, non vengono offerti tramite e. AWS CLI AWS SDKs Per ulteriori informazioni, consulta RESTful API nella documentazione dell'API FHIR RESTful R4.

La tabella seguente elenca le interazioni FHIR R4 supportate da. AWS HealthLake Per informazioni sui tipi di risorse FHIR supportati da HealthLake, vedere. Tipi di risorse

Interazioni FHIR R4 supportate da AWS HealthLake

Interazione	Description
Interazioni dell'intero sistema	
`capabilities`	Ottieni una dichiarazione di capacità per il sistema. Per informazioni, consulta Dichiarazione di capacità FHIR R4 per AWS HealthLake.
`batch`	Aggiorna, crea o elimina un set di risorse in una singola interazione. Per informazioni, consulta Raggruppamento di risorse FHIR.
Interazioni a livello di tipo	
`create`	Crea una nuova risorsa con un ID assegnato dal server. Per informazioni, consulta Creazione di una risorsa FHIR.
`search`	Cerca un tipo di risorsa in base ad alcuni criteri di filtro. Per informazioni, consulta Ricerca di risorse FHIR.

Interazione	Description
`history`	Recupera la cronologia delle modifiche per un particolare tipo di risorsa. Per informazioni, consulta Leggere la cronologia delle risorse FHIR.
Interazioni a livello di istanza	
`read`	Leggi lo stato attuale di una risorsa. Per informazioni, consulta Leggere una risorsa FHIR.
`history`	Leggi la cronologia delle modifiche per una particolare risorsa. Per informazioni, consulta Leggere la cronologia delle risorse FHIR.
`vread`	Leggi lo stato di una versione specifica della risorsa. Per informazioni, consulta Leggere la cronologia delle risorse FHIR specifiche della versione.
`update`	Aggiorna una risorsa in base al suo ID (o creala se è nuova). Per informazioni, consulta Aggiornamento di una risorsa FHIR.
`delete`	Eliminare una risorsa. Per informazioni, consulta Eliminazione di una risorsa FHIR.

Argomenti

- Creazione di una risorsa FHIR
- Leggere una risorsa FHIR
- Leggere la cronologia delle risorse FHIR
- Aggiornamento di una risorsa FHIR
- Modifica delle risorse con l'operazione PATCH
- Raggruppamento di risorse FHIR
- Eliminazione di una risorsa FHIR
- Idempotenza e concorrenza

Creazione di una risorsa FHIR

L'`create`interazione FHIR crea una nuova risorsa FHIR in un HealthLake archivio dati. Per ulteriori informazioni, consulta la documentazione dell'`create`API FHIR RESTful R4.

Per creare una risorsa FHIR

1. Colleziona HealthLake `region` e `datastoreId` valorizza. Per ulteriori informazioni, consulta Ottenere le proprietà dell'archivio dati.
2. Determina il tipo di FHIR `Resource` da creare. Per ulteriori informazioni, consulta Tipi di risorse.
3. Costruisci un URL per la richiesta utilizzando i valori raccolti per HealthLake `region` e. `datastoreId` Includi anche il `Resource` tipo FHIR da creare. Per visualizzare l'intero percorso dell'URL nell'esempio seguente, scorri il pulsante Copia.

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

4. Costruisci un corpo JSON per la richiesta, specificando i dati FHIR per la nuova risorsa. Ai fini di questa procedura, stiamo usando una `Patient` risorsa FHIR, quindi salva il file con nome. `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. Inviare la richiesta . L'`create`interazione FHIR utilizza una `POST` richiesta con autorizzazione AWS Signature Version 4 o SMART on FHIR. Gli esempi seguenti creano una `Patient` risorsa FHIR HealthLake utilizzando curl o la Console. HealthLake Per visualizzare un intero esempio, scorri il pulsante Copia.

   SigV4

   Autorizzazione 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

   Esempio di autorizzazione SMART on FHIR per il tipo di dati `IdentityProviderConfiguration`.

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

Il chiamante può assegnare le autorizzazioni nella lambda di autorizzazione. Per ulteriori informazioni, consulta OAuth Cannocchiali 2.0.

AWS Console

> **Note**
>
> La HealthLake console supporta solo l'autorizzazione SigV4.AWS

1. Accedi alla pagina Esegui interrogazione sulla Console. HealthLake
2. Nella sezione Impostazioni della query, effettua le seguenti selezioni.
   - Data Store ID: scegli un ID del data store per generare una stringa di query.
   - Tipo di query: scegli`Create`.
   - Tipo di risorsa: scegli il tipo di risorsa FHIR da creare.
   - Corpo della richiesta: crea un corpo JSON per la richiesta, specificando i dati FHIR per la nuova risorsa.
3. Scegli Esegui query.

Configurazione del livello di convalida per la creazione di risorse

Quando si crea una risorsa FHIR, è possibile specificare facoltativamente un'intestazione `x-amzn-healthlake-fhir-validation-level` HTTP per configurare un livello di convalida per la risorsa. AWS HealthLake attualmente supporta i seguenti livelli di convalida:

- `strict`: Le risorse vengono convalidate in base all'elemento del profilo della risorsa o alla specifica R4 se non è presente alcun profilo. Questo è il livello di convalida predefinito per. AWS HealthLake
- `structure-only`: Le risorse vengono convalidate rispetto a R4, ignorando i profili di riferimento.
- `minimal`: Le risorse vengono convalidate minimamente, ignorando alcune regole R4. Le risorse che non superano i controlli di struttura richiesti search/analytics verranno aggiornate per includere un avviso per l'audit.

Le risorse create con il livello di convalida minimo possono essere inserite in un Datastore nonostante la mancata convalida richiesta per l'indicizzazione della ricerca. In questo caso, le risorse verranno aggiornate per includere un'estensione specifica di Healthlake per documentare tali errori:

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

Inoltre, la seguente intestazione di risposta HTTP verrà inclusa con il valore «true»:

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

> **Note**
>
> I dati inseriti che presentano un formato errato secondo la specifica R4 potrebbero non essere ricercabili come previsto se sono presenti questi errori.

Leggere una risorsa FHIR

L'`read`interazione FHIR legge lo stato corrente di una risorsa in un HealthLake archivio dati. Per ulteriori informazioni, consulta la documentazione dell'`read`API FHIR RESTful R4.

Per leggere una risorsa FHIR

1. Colleziona HealthLake `region` e `datastoreId` valorizza. Per ulteriori informazioni, consulta Ottenere le proprietà dell'archivio dati.
2. Determina il tipo di FHIR `Resource` per leggere e raccogliere il `id` valore associato. Per ulteriori informazioni, consulta Tipi di risorse.
3. Costruisci un URL per la richiesta utilizzando i valori raccolti per HealthLake `region` e. `datastoreId` Includi anche il `Resource` tipo FHIR e il relativo associato. `id` Per visualizzare l'intero percorso dell'URL nell'esempio seguente, scorri il pulsante Copia.

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

4. Inviare la richiesta . L'`read`interazione FHIR utilizza una `GET` richiesta con autorizzazione AWS Signature Version 4 o SMART on FHIR. L'`curl`esempio seguente legge lo stato corrente di una risorsa `Patient` FHIR in. HealthLake Per visualizzare l'intero esempio, scorrete il pulsante Copia.

   SigV4

   Autorizzazione 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

   Esempio di autorizzazione SMART on FHIR per il tipo di dati `IdentityProviderConfiguration`.

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

Il chiamante può assegnare le autorizzazioni nella lambda di autorizzazione. Per ulteriori informazioni, consulta OAuth Cannocchiali 2.0.

AWS Console

1. Accedi alla pagina Esegui query sulla console. HealthLake

2. Nella sezione Impostazioni della query, effettua le seguenti selezioni.

- Data Store ID: scegli un ID del data store per generare una stringa di query.
- Tipo di query: scegli`Read`.
- Tipo di risorsa: scegli il tipo di risorsa FHIR da leggere.
- ID risorsa: immettere l'ID della risorsa FHIR.

3. Scegli Esegui query.

Leggere la cronologia delle risorse FHIR

L'`history`interazione FHIR recupera la cronologia di una particolare risorsa FHIR in un archivio dati. HealthLake Utilizzando questa interazione, è possibile determinare in che modo il contenuto di una risorsa FHIR è cambiato nel tempo. È anche utile, in coordinamento con i registri di controllo, per vedere lo stato di una risorsa prima e dopo la modifica. Le interazioni `create` FHIR danno come `delete` risultato una versione storica della risorsa da salvare. `update` Per ulteriori informazioni, vedere la documentazione `history`dell'API FHIR R4 RESTful .

> **Note**
>
> È possibile disattivare tipi di risorse `history` FHIR specifici. Per annullare l'iscrizione, crea un caso utilizzando AWS Support Center Console. Per creare il tuo caso, accedi al tuo Account AWS e scegli Crea caso.

Per leggere la cronologia delle risorse FHIR

1. Colleziona HealthLake `region` e `datastoreId` valorizza. Per ulteriori informazioni, consulta Ottenere le proprietà dell'archivio dati.

2. Determina il tipo di FHIR `Resource` per leggere e raccogliere il `id` valore associato. Per ulteriori informazioni, consulta Tipi di risorse.
3. Costruisci un URL per la richiesta utilizzando i valori raccolti per HealthLake `region` e. `datastoreId` Includi anche il `Resource` tipo FHIR, i relativi parametri di ricerca associati `id` e facoltativi. Per visualizzare l'intero percorso dell'URL nell'esempio seguente, scorri il pulsante Copia.

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

HealthLake parametri di ricerca supportati per l'interazione FHIR **`history`**

Parametro	Description
`_count : integer`	Il numero massimo di risultati di ricerca in una pagina. Il server restituirà il numero richiesto o il numero massimo di risultati di ricerca consentito per impostazione predefini ta per l'archivio dati, a seconda di quale sia inferiore.
`_since : instant`	Include solo le versioni delle risorse create in un determinato istante o successivamente.
`_at : date(Time)`	Includi solo le versioni delle risorse che erano correnti in un determinato momento durante il periodo di tempo specificato nel valore di data e ora. Per ulteriori informazioni, consulta `date`la documentazione dell' RESTfulAPI HL7 FHIR.

4. Inviare la richiesta . L'`history`interazione FHIR utilizza una `GET` richiesta con autorizzazione AWS Signature Version 4 o SMART on FHIR. L'`curl`esempio seguente utilizza il parametro `_count` search per restituire 100 risultati di ricerca storici per pagina per una risorsa FHIR `Patient` in. HealthLake Per visualizzare l'intero esempio, scorri il pulsante Copia.

SigV4

Autorizzazione 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

Esempio di autorizzazione SMART on FHIR per il tipo di dati `IdentityProviderConfiguration`.

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

Il chiamante può assegnare le autorizzazioni nella lambda di autorizzazione. Per ulteriori informazioni, consulta OAuth Cannocchiali 2.0.

Il contenuto restituito da un'`history`interazione è contenuto in una risorsa FHIR`Bundle`, con tipo impostato su. `history` Contiene la cronologia delle versioni specificata, ordinata in base alle ultime versioni più vecchie e include le risorse eliminate. Per ulteriori informazioni, vedere la documentazione `Resource Bundle`di FHIR R4.

Leggere la cronologia delle risorse FHIR specifiche della versione

L'`vread`interazione FHIR esegue una lettura specifica della versione di una risorsa in un archivio dati. HealthLake Utilizzando questa interazione, è possibile visualizzare il contenuto di una risorsa FHIR com'era in un determinato momento del passato.

> **Note**
>
> Se si utilizza l'`history`interazione FHIR senza`vread`, restituisce HealthLake sempre la versione più recente dei metadati della risorsa.

HealthLake dichiara il supporto per il controllo delle versioni per ogni risorsa supportata `CapabilityStatement.rest.resource.versioning`. Tutti gli archivi HealthLake dati includono `Resource.meta.versionId` (`vid`) su tutte le risorse.

Quando `history` l'interazione FHIR è abilitata (per impostazione predefinita per gli archivi dati creati dopo il 25/10/2024 o su richiesta per gli archivi dati più vecchi), la `Bundle` risposta include l'elemento `vid` come parte dell'elemento. `location` Nell'esempio seguente, viene visualizzato come numero. `vid` 1 Per visualizzare l'esempio completo, vedi Example bundle/bundle-response (JSON).

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

Per leggere la cronologia delle risorse FHIR specifica della versione

1. Raccolta e valori HealthLake `region`. `datastoreId` Per ulteriori informazioni, consulta Ottenere le proprietà dell'archivio dati.
2. Determina il `Resource` tipo di FHIR da leggere e raccogliere i `vid` valori `id` e i valori associati. Per ulteriori informazioni, consulta Tipi di risorse.
3. Costruisci un URL per la richiesta utilizzando i valori raccolti per HealthLake e FHIR. Per visualizzare l'intero percorso dell'URL nell'esempio seguente, scorri il pulsante Copia.

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

4. Inviare la richiesta . L'`history`interazione FHIR utilizza una `GET` richiesta con autorizzazione AWS Signature Version 4 o SMART on FHIR. La seguente `vread` interazione restituisce una singola istanza con il contenuto specificato per la `Patient` risorsa FHIR per la versione dei metadati della risorsa specificata da. `vid` Per visualizzare l'intero percorso dell'URL nell'esempio seguente, scorri il pulsante Copia.

   SigV4

   Autorizzazione 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

   Esempio di autorizzazione SMART on FHIR per il tipo di dati `IdentityProviderConfiguration`.

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

```
}
```

Il chiamante può assegnare le autorizzazioni nella lambda di autorizzazione. Per ulteriori informazioni, consulta OAuth Cannocchiali 2.0.

Aggiornamento di una risorsa FHIR

L'`update`interazione FHIR crea una nuova versione corrente per una risorsa esistente o crea una versione iniziale se non esiste già alcuna risorsa per la determinata risorsa. `id` Per ulteriori informazioni, consulta la `update`documentazione dell'API FHIR R4 RESTful .

Per aggiornare una risorsa FHIR

1. Colleziona HealthLake `region` e `datastoreId` valorizza. Per ulteriori informazioni, consulta Ottenere le proprietà dell'archivio dati.
2. Determina il tipo di FHIR `Resource` da aggiornare e raccogliere il `id` valore associato. Per ulteriori informazioni, consulta Tipi di risorse.
3. Costruisci un URL per la richiesta utilizzando i valori raccolti per HealthLake `region` e. `datastoreId` Includi anche il `Resource` tipo FHIR e il relativo associato. `id` Per visualizzare l'intero percorso dell'URL nell'esempio seguente, scorri il pulsante Copia.

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

4. Costruisci un `JSON` corpo per la richiesta, specificando gli aggiornamenti dei dati FHIR da effettuare. Ai fini di questa procedura, salva il file con nome. `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. Inviare la richiesta . L'`update`interazione FHIR utilizza una `PUT` richiesta con autorizzazione AWS Signature Version 4 o SMART on FHIR. L'`curl`esempio seguente aggiorna una `Patient` risorsa in. HealthLake Per visualizzare l'intero esempio, scorri il pulsante Copia.

   SigV4

   Autorizzazione 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
   ```

   La richiesta restituirà un codice di stato `200` HTTP se una risorsa esistente viene aggiornata o un codice di stato `201` HTTP se viene creata una nuova risorsa.

   SMART on FHIR

   Esempio di autorizzazione SMART on FHIR per il tipo di `IdentityProviderConfiguration`dati.

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

Il chiamante può assegnare le autorizzazioni nella lambda di autorizzazione. Per ulteriori informazioni, consulta OAuth Cannocchiali 2.0.

AWS Console

1. Accedi alla pagina Esegui query sulla console. HealthLake
2. Nella sezione Impostazioni della query, effettua le seguenti selezioni.

- Data Store ID: scegli un ID del data store per generare una stringa di query.
- Tipo di query: scegli`Update (PUT)`.
- Tipo di risorsa: scegli il tipo di risorsa FHIR da aggiornare o creare.
- Corpo della richiesta: crea un corpo JSON per la richiesta, specificando i dati FHIR con cui aggiornare la risorsa.

3. Scegli Esegui query.

Aggiornamento delle risorse FHIR in base alle condizioni

L'aggiornamento condizionale consente di aggiornare una risorsa esistente in base ad alcuni criteri di ricerca di identificazione, anziché tramite FHIR logico. `id` Quando il server elabora l'aggiornamento, esegue una ricerca utilizzando le funzionalità di ricerca standard per il tipo di risorsa, con l'obiettivo di risolvere una singola logica `id` per la richiesta.

L'azione intrapresa dal server dipende dal numero di corrispondenze trovate:

- Nessuna corrispondenza, non **`id`** fornita nel corpo della richiesta: il server crea la risorsa FHIR.

- Nessuna corrispondenza, **`id`** fornita e la risorsa non esiste già con**`id`**: Il server considera l'interazione come un'interazione Update as Create.
- Nessuna corrispondenza, **`id`** fornita e già esistente: il server rifiuta l'aggiornamento con un `409 Conflict` errore.
- Una corrispondenza, nessuna risorsa **`id`** fornita OPPURE (risorsa **`id`** fornita e corrisponde alla risorsa trovata): il server esegue l'aggiornamento rispetto alla risorsa corrispondente come sopra dove, se la risorsa è stata aggiornata, il server DEVE restituire un`200 OK`.
- One Match, risorsa **`id`** fornita ma non corrispondente alla risorsa trovata: il server restituisce un `409 Conflict` errore che indica che la specifica dell'ID del client era un problema, preferibilmente con un `OperationOutcome`
- Corrispondenze multiple: Il server restituisce un `412 Precondition Failed` errore che indica che i criteri del client non erano sufficientemente selettivi, preferibilmente con un OperationOutcome

L'esempio seguente aggiorna una `Patient` risorsa il cui nome è peter, la data di nascita è il 1° gennaio 2000 e il numero di telefono è 1234567890.

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

Configurazione del livello di convalida per gli aggiornamenti delle risorse

Quando si aggiorna una risorsa FHIR, è possibile facoltativamente specificare un'intestazione `x-amzn-healthlake-fhir-validation-level` HTTP per configurare un livello di convalida per la risorsa. AWS HealthLake attualmente supporta i seguenti livelli di convalida:

- `strict`: Le risorse vengono convalidate in base all'elemento del profilo della risorsa o alla specifica R4 se non è presente alcun profilo. Questo è il livello di convalida predefinito per. AWS HealthLake
- `structure-only`: Le risorse vengono convalidate rispetto a R4, ignorando i profili di riferimento.
- `minimal`: Le risorse vengono convalidate minimamente, ignorando alcune regole R4. Le risorse che non superano i controlli di struttura richiesti search/analytics verranno aggiornate per includere un avviso per l'audit.

Le risorse aggiornate con il livello di convalida minimo possono essere inserite in un Datastore nonostante la convalida non riuscita richiesta per l'indicizzazione della ricerca. In questo caso, le risorse verranno aggiornate per includere un'estensione specifica di Healthlake per documentare tali errori:

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

Inoltre, la seguente intestazione di risposta HTTP verrà inclusa con il valore «true»:

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

> **Note**
>
> Tieni presente che i dati inseriti in formato non corretto secondo la specifica R4 potrebbero non essere ricercabili come previsto se sono presenti questi errori.

Modifica delle risorse con l'operazione PATCH

AWS HealthLake supporta l'operazione PATCH per le risorse FHIR, che consente di modificare le risorse prendendo di mira elementi specifici da aggiungere, sostituire o eliminare senza aggiornare l'intera risorsa. Questa operazione è particolarmente utile quando è necessario:

- Effettuare aggiornamenti mirati a risorse di grandi dimensioni
- Riduci l'utilizzo della larghezza di banda di rete
- Esegui modifiche atomiche su elementi di risorse specifici
- Riduci al minimo il rischio di sovrascrivere le modifiche simultanee
- Aggiorna le risorse come parte dei flussi di lavoro in batch e delle transazioni

Formati PATCH supportati

AWS HealthLake supporta due formati PATCH standard:

Patch JSON (RFC 6902)

Utilizza la sintassi JSON Pointer per indirizzare gli elementi in base alla loro posizione nella struttura delle risorse.

Tipo di contenuto: `application/json-patch+json`

FHIRPath Patch (specifica FHIR R4)

Utilizza FHIRPath le espressioni per indirizzare gli elementi in base al loro contenuto e alle loro relazioni, fornendo un approccio nativo FHIR all'applicazione delle patch.

Tipo di contenuto: `application/fhir+json`

Utilizzo

Operazioni PATCH dirette

L'operazione PATCH può essere richiamata direttamente sulle risorse FHIR utilizzando il metodo HTTP PATCH:

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

PATCH in pacchetti

Le operazioni PATCH possono essere incluse come voci all'interno di pacchetti FHIR di tipo `batch` oppure `transaction` consentono di combinare le operazioni di patch con altre interazioni FHIR (creazione, lettura, aggiornamento, eliminazione) in un'unica richiesta.

- Pacchetti di transazioni: tutte le voci hanno esito positivo o negativo in modo atomico
- Pacchetti Batch: ogni voce viene elaborata in modo indipendente

Formato di patch JSON

Operazioni supportate

Operation	Description
`add`	Aggiungi un nuovo valore alla risorsa
`remove`	Rimuove un valore dalla risorsa
`replace`	Sostituisci un valore esistente nella risorsa
`move`	Rimuovi un valore da una posizione e aggiungilo a un'altra
`copy`	Copia un valore da una posizione all'altra
`test`	Verifica che un valore nella posizione di destinazione sia uguale a un valore specificato

Sintassi del percorso

JSON Patch utilizza la sintassi JSON Pointer (RFC 6901):

Esempio di percorso	Description
`/name/0/family`	Elemento della famiglia del nome
`/telecom/-`	Aggiungi all'array di telecomunicazioni
`/active`	Elemento attivo a livello di root
`/address/0/line/1`	Seconda riga del primo indirizzo

Esempi

Richiesta diretta di patch JSON con più operazioni

```
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"
  }
]
```

Richiesta diretta di patch JSON con singola operazione

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

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

Patch JSON in bundle

Usa una risorsa binaria contenente il payload della patch JSON con codifica base64:

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

FHIRPath Formato della patch

Operazioni supportate

Operation	Description
`add`	Aggiungere un nuovo elemento a una risorsa
`insert`	Inserisce un elemento in una posizione specifica in un elenco

Operation	Description
`delete`	Rimuove un elemento da una risorsa
`replace`	Sostituisci il valore di un elemento esistente
`move`	Riordina gli elementi all'interno di un elenco

Sintassi del percorso

FHIRPath La patch utilizza FHIRPath espressioni che supportano:

- Accesso basato su indici: `Patient.name[0]`
- Filtraggio con: `where()` `Patient.name.where(use = 'official')`
- Logica booleana: `Patient.telecom.where(system = 'phone' and use = 'work')`
- Funzioni di sottoimpostazione:, `first()` `last()`
- Controlli di esistenza:`exists()`, `count()`
- Navigazione polimorfa: `0bservation.value`

Esempi

Richiesta diretta di patch 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 Patch inclusa nel pacchetto

Usa una risorsa Parameters come risorsa di ingresso 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"
    }
  }]
}
```

Intestazioni di richiesta

Header	Richiesto	Descrizione
`Content-Type`	Sì	`application/json-patch+json` per JSON Patch o `application/fhir+json` per FHIRPath Patch
`If-Match`	No	Aggiornamento condizionale specifico della versione utilizzando ETag

Risposta di esempio

L'operazione restituisce la risorsa aggiornata con le informazioni sulla nuova versione:

```
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"
  }
}
```

Comportamento

L'operazione PATCH:

- Convalida la sintassi della patch in base alle specifiche appropriate (RFC 6902 per JSON Patch, FHIR R4 per Patch) FHIRPath
- Applica le operazioni in modo atomico: tutte le operazioni hanno esito positivo o negativo
- Aggiorna l'ID della versione della risorsa e crea una nuova voce della cronologia
- Conserva la risorsa originale nella cronologia prima di applicare le modifiche

- Convalida i vincoli delle risorse FHIR dopo l'applicazione delle patch
- Supporta gli aggiornamenti condizionali utilizzando l'intestazione If-Match con ETag

Gestione errori

L'operazione gestisce le seguenti condizioni di errore:

- 400 Bad Request: sintassi della patch non valida (richiesta non conforme o documento di patch non valido)
- 404 Not Found: risorsa non trovata (l'ID specificato non esiste)
- 409 Conflitto: conflitto di versione (vengono forniti aggiornamenti simultanei o ID di versione non corrente)
- 422 Entità non processabile: le operazioni di patch non possono essere applicate agli elementi di risorsa specificati

Riepilogo delle capacità

Funzionalità	Patch JSON	FHIRPath Patch
Tipo di contenuto	`application/json-patch+json`	`application/fhir+json`
Formato del percorso	Puntatore JSON (RFC 6901)	FHIRPath espressioni
API Patch diretta	Supportata	Supportata
Batch del pacchetto	Supportato (tramite binario)	Supportato (tramite parametri)
Transazione in bundle	Supportato (tramite binario)	Supportato (tramite parametri)
Operazioni	aggiungere, rimuovere, sostituire, spostare, copiare, testare	aggiungere, inserire, eliminare, sostituire, spostare

Limitazioni

- Le operazioni PATCH condizionali che utilizzano condizioni di ricerca non sono supportate
- La patch JSON nei bundle deve utilizzare risorse binarie con contenuto codificato in base64
- FHIRPath Le patch nei bundle devono utilizzare risorse Parameters

Risorse aggiuntive

Per ulteriori informazioni sulle operazioni PATCH, consulta:

- Documentazione FHIR R4 PATCH
- Specifiche della patch FHIR R4 FHIRPath
- RFC 6902 - Patch JSON
- RFC 6901 - Puntatore JSON

Raggruppamento di risorse FHIR

Un FHIR `Bundle` è un contenitore per una raccolta di risorse FHIR in. AWS HealthLake AWS HealthLake supporta due tipi di pacchetti con comportamenti di elaborazione diversi.

`Batch`i pacchetti elaborano ogni risorsa in modo indipendente. Se una risorsa fallisce, le risorse rimanenti possono comunque avere successo. Ogni operazione viene elaborata singolarmente e l'elaborazione continua anche quando alcune operazioni falliscono. Utilizzate pacchetti in batch per operazioni di massa in cui è accettabile un successo parziale, come il caricamento di più cartelle cliniche di pazienti non correlate.

`Transaction`i pacchetti elaborano tutte le risorse in modo atomico come una singola unità. Tutte le operazioni relative alle risorse hanno esito positivo oppure nessuna di AWS HealthLake esse viene eseguita. Utilizza i pacchetti di transazioni quando hai bisogno di un'integrità referenziale garantita tra le risorse correlate, ad esempio per creare un paziente con osservazioni e condizioni correlate in cui tutti i dati devono essere registrati insieme.

Differenze tra pacchetti batch e pacchetti di transazioni

Funzionalità	Archiviazione	Transaction
Modello di elaborazione	Ogni operazione riesce o fallisce indipendentemente.	Tutte le operazioni hanno esito positivo o negativo come singola unità atomica.
Gestione degli errori	L'elaborazione continua anche se le singole operazioni falliscono.	L'intero pacchetto fallisce se una singola operazione fallisce.
Ordine di esecuzione	L'ordine di esecuzione non è garantito.	Le operazioni vengono elaborate nell'ordine specificato.
Integrità referenziale	Non applicato in tutte le operazioni.	Applicato per le risorse referenziate localmente all'interno del pacchetto.
Ideale per	Operazioni in blocco in cui il successo parziale è accettabile.	Risorse correlate che devono essere create o aggiornate insieme.

È possibile raggruppare risorse FHIR dello stesso tipo o di tipo diverso e queste possono includere una combinazione di operazioni FHIR, come,`create`, `readupdate`, `delete` e. `patch` Per ulteriori informazioni, vedere Resource Bundle nella documentazione di FHIR R4.

Di seguito sono riportati alcuni esempi di casi d'uso per ogni tipo di pacchetto.

Pacchetti Batch

- Carica più cartelle cliniche dei pazienti non correlate da diverse strutture durante la sincronizzazione notturna dei dati.
- Carica in blocco la cronologia dei farmaci, in cui alcuni dati potrebbero presentare problemi di convalida.
- Carica dati di riferimento, ad esempio organizzazioni e professionisti, laddove i singoli errori non influiscano sulle altre voci.

Pacchetti di transazioni

- Crea un paziente con le relative osservazioni e condizioni durante il ricovero al pronto soccorso, in cui tutti i dati devono essere registrati insieme.

- Aggiorna l'elenco dei farmaci del paziente e le relative informazioni sulle allergie, che devono rimanere coerenti.
- Registra un incontro completo con il paziente, le osservazioni, le procedure e le informazioni di fatturazione come un'unica unità atomica.

> **Important**
>
> Sia i pacchetti batch che quelli di transazione utilizzano la stessa `Bundle` struttura di risorse. L'unica differenza è il valore del `type` campo.

L'esempio seguente mostra un pacchetto di transazioni con più tipi di risorse e operazioni.

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

Raggruppamento di risorse FHIR come entità indipendenti

Raggruppare le risorse FHIR come entità indipendenti

1. Raccogli HealthLake `region` e `datastoreId` valorizza. Per ulteriori informazioni, consulta Ottenere le proprietà dell'archivio dati.
2. Costruisci un URL per la richiesta utilizzando i valori raccolti per HealthLake `region` e`datastoreId`. Non specificate un tipo di risorsa FHIR nell'URL. Per visualizzare l'intero percorso dell'URL nell'esempio seguente, scorri il pulsante Copia.

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

3. Costruisci un corpo JSON per la richiesta, specificando ogni verbo HTTP come parte degli elementi. `method` L'esempio seguente utilizza un'interazione `batch` di tipo con la `Bundle` risorsa per creare nuove risorse. `Patient` `Medication` Tutte le sezioni obbligatorie vengono commentate di conseguenza. Ai fini di questa procedura, salva il file con nome. `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. Inviare la richiesta . Il tipo di `Bundle` batch FHIR utilizza una `POST` richiesta con autorizzazione AWS Signature Version 4 o SMART on FHIR. Il seguente esempio di codice utilizza lo strumento da riga di `curl` comando a scopo dimostrativo.

   SigV4

   Autorizzazione 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

   Esempio di autorizzazione SMART on FHIR per il tipo di dati `IdentityProviderConfiguration`.

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

```
}
```

Il chiamante può assegnare le autorizzazioni nella lambda di autorizzazione. Per ulteriori informazioni, consulta OAuth Cannocchiali 2.0.

Il server restituisce una risposta che mostra le `Medication` risorse `Patient` e le risorse create a seguito della `Bundle` richiesta di tipo batch.

Condizionale PUTs nei pacchetti

AWS HealthLake supporta gli aggiornamenti condizionali all'interno dei pacchetti utilizzando i seguenti parametri di query:

> **Note**
>
> I condizionali PUTs sono supportati solo nei pacchetti. `batch` `Transaction`i pacchetti non supportano il condizionale. PUTs

- `_id`(autonomo)
- `_id`in combinazione con uno dei seguenti:
  - `_tag`
  - `_createdAt`
  - `_lastUpdated`

Quando si utilizza il condizionale PUTs nei pacchetti, AWS HealthLake valuta i parametri della query rispetto alle risorse esistenti e interviene in base ai risultati della corrispondenza.

Comportamento di aggiornamento condizionale

Scenario	Stato HTTP	Azione intrapresa
Risorsa senza ID fornito	201 Creato	Crea sempre una nuova risorsa.
Risorsa con nuovo ID (nessuna corrispondenza)	201 Creato	Crea una nuova risorsa con l'ID specificato.

Scenario	Stato HTTP	Azione intrapresa
Risorsa con ID esistente (corrispondenza singola)	200 OK	Aggiorna la risorsa corrispondente.
Risorsa con ID esistente (è stato rilevato un conflitto)	409 Conflitto	Restituisce un errore. Non viene apportata alcuna modifica.
Risorsa con ID esistente (ID non corrispondente)	400 Richiesta non valida	Restituisce un errore. Non viene apportata alcuna modifica.
Risorse multiple soddisfano le condizioni	412 Precondizione non riuscita	Restituisce un errore. Non viene apportata alcuna modifica.

Nel seguente esempio, abbinato a un aggiornamento condizionale, la `Patient` risorsa con ID FHIR si `476` aggiorna solo se la condizione `_lastUpdated=lt2025-04-20` è soddisfatta.

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

Raggruppamento delle risorse FHIR come un'unica entità

Per raggruppare le risorse FHIR come un'unica entità

1. Raccolta HealthLake `region` e valori. `datastoreId` Per ulteriori informazioni, consulta Ottenere le proprietà dell'archivio dati.
2. Costruisci un URL per la richiesta utilizzando i valori raccolti per HealthLake `region` e`datastoreId`. Includi il tipo di risorsa FHIR `Bundle` come parte dell'URL. Per visualizzare l'intero percorso dell'URL nell'esempio seguente, scorri il pulsante Copia.

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

3. Costruisci un corpo JSON per la richiesta, specificando le risorse FHIR da raggruppare. L'esempio seguente raggruppa due risorse in. `Patient` HealthLake Ai fini di questa procedura, salva il file con nome`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. Inviare la richiesta . Il tipo di `Bundle` documento FHIR utilizza una `POST` richiesta con protocollo di AWS firma Signature Version 4. Il seguente esempio di codice utilizza lo strumento da riga di `curl` comando a scopo dimostrativo.

   SigV4

   Autorizzazione 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

   Esempio di autorizzazione SMART on FHIR per il tipo di dati `IdentityProviderConfiguration`.

   ```
   {
   ```

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

Il chiamante può assegnare le autorizzazioni nella lambda di autorizzazione. Per ulteriori informazioni, consulta OAuth Cannocchiali 2.0.

Il server restituisce una risposta che mostra due `Patient` risorse create a seguito della richiesta del `Bundle` tipo di documento.

Configurazione del livello di convalida per i pacchetti

Quando raggruppate le risorse FHIR, potete facoltativamente specificare un'intestazione `x-amzn-healthlake-fhir-validation-level` HTTP per configurare un livello di convalida per la risorsa. Questo livello di convalida verrà impostato per tutte le richieste di creazione e aggiornamento all'interno del pacchetto. AWS HealthLake attualmente supporta i seguenti livelli di convalida:

- `strict`: Le risorse vengono convalidate in base all'elemento del profilo della risorsa o alla specifica R4 se non è presente alcun profilo. Questo è il livello di convalida predefinito per. AWS HealthLake
- `structure-only`: Le risorse vengono convalidate rispetto a R4, ignorando i profili di riferimento.
- `minimal`: Le risorse vengono convalidate minimamente, ignorando alcune regole R4. Le risorse che non superano i controlli di struttura richiesti search/analytics verranno aggiornate per includere un avviso per l'audit.

Le risorse associate al livello di convalida minimo possono essere inserite in un Datastore nonostante la convalida non riuscita richiesta per l'indicizzazione della ricerca. In questo caso, le risorse verranno aggiornate per includere un'estensione specifica di Healthlake per documentare tali errori e le voci contenute nella risposta del Bundle includeranno le seguenti risorse: 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."
                        }
                    ]
                }
            }
        }
    ]
}
```

Inoltre, la seguente intestazione di risposta HTTP verrà inclusa con il valore «true»:

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

> **Note**
>
> Tieni presente che i dati inseriti in formato non corretto secondo la specifica R4 potrebbero non essere ricercabili come previsto se questi errori sono presenti.

Supporto limitato per il tipo «message» di tipo Bundle

HealthLake fornisce un supporto limitato per il tipo FHIR Bundle `message` attraverso un processo di conversione interno. Questo supporto è progettato per scenari in cui i pacchetti di messaggi non possono essere riformattati all'origine, come l'importazione di feed ADT (Admission, Discharge, Transfer) da sistemi ospedalieri preesistenti.

> **Warning**
>
> Questa funzionalità richiede un elenco esplicito degli AWS account consentiti e non applica la semantica dei messaggi FHIR R4 o l'integrità referenziale. Contatta l' AWS assistenza per richiedere l'attivazione del tuo account prima di utilizzare i pacchetti di messaggi.

Principali differenze rispetto all'elaborazione standard dei messaggi

- Pacchetti di messaggi (specifica FHIR): la prima immissione deve essere una `MessageHeader` che faccia riferimento ad altre risorse. Alle risorse mancano i singoli `request` oggetti e l' MessageHeader evento determina le azioni di elaborazione.
- HealthLake Elaborazione: converte i pacchetti di messaggi in pacchetti batch assegnando automaticamente le operazioni PUT a ciascuna risorsa immessa. Le risorse vengono elaborate in modo indipendente senza applicare la semantica dei messaggi o l'integrità referenziale.

Limitazioni importanti

- Le regole di elaborazione specifiche dei messaggi FHIR R4 non vengono applicate
- Nessuna integrità transazionale tra le risorse
- I riferimenti tra le risorse non vengono convalidati
- Richiede un elenco esplicito degli account consentiti

Esempio di struttura del pacchetto di messaggi

```
{
              "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**
>
> Ogni risorsa viene archiviata in modo indipendente come se fosse inviata tramite singole operazioni PUT. Se è richiesta la semantica completa della messaggistica FHIR o la convalida dell'integrità referenziale, preelabora i pacchetti di messaggi o implementa la convalida a livello di applicazione prima dell'invio.

Transazioni di bundle asincrone

AWS HealthLake supporta il `Bundle` tipo asincrono `transaction` che consente di inviare transazioni con un massimo di 500 risorse. Quando invii una transazione asincrona, la mette in HealthLake coda per l'elaborazione e restituisce immediatamente un URL di polling. Puoi utilizzare

questo URL per controllare lo stato e recuperare la risposta. Questo segue lo schema del pacchetto asincrono FHIR.

Quando utilizzare le transazioni asincrone

- È necessario inviare più di 100 risorse (limite sincrono) in una singola transazione.
- Desiderate evitare di bloccare l'applicazione in attesa del completamento dell'elaborazione della transazione.
- È necessario elaborare volumi elevati di risorse correlate con un throughput migliore.

Important

I risultati dei sondaggi sono disponibili per 90 giorni dopo il completamento della transazione. Dopo questo periodo di 90 giorni, l'URL del sondaggio non restituisce più risultati. Progetta la tua integrazione per recuperare e archiviare i risultati all'interno di questa finestra.

Note

Il `Bundle` tipo sincrono `transaction` continua a supportare fino a 100 risorse ed è la modalità di elaborazione predefinita. Se invii un `Bundle` tipo `transaction` con più di 100 risorse senza l'`Prefer: respond-async`intestazione, HealthLake restituisce un `422 Unprocessable Entity` errore. I pacchetti con tipo non `batch` sono supportati per l'elaborazione asincrona: solo i tipi `transaction` possono essere inviati in modo asincrono (con un massimo `Bundle` di 500 operazioni).

Note

PATCHle operazioni e le condizioni non sono supportate nelle transazioni di bundle asincrone. PUTs

Invio di una transazione asincrona

Per inviare una transazione asincrona, invia una POST richiesta all'endpoint del data store con l'intestazione. `Prefer: respond-async` Il pacchetto deve avere un tipo. `transaction` I pacchetti con tipo non `batch` sono supportati per l'elaborazione asincrona dei pacchetti.

HealthLake esegue le convalide iniziali del pacchetto al momento dell'invio. Se la convalida ha esito positivo, HealthLake restituisce HTTP 202 Accepted con un'intestazione di `content-location` risposta che contiene l'URL del polling.

Per inviare un tipo asincrono **`Bundle` `transaction`**

1. Invia una POST richiesta all'endpoint del HealthLake data store.

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

2. Costruisci un corpo JSON per la richiesta con il tipo di bundle. `transaction` Ai fini di questa procedura, salva il file con nome. `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. Invia la richiesta con l'`Prefer: respond-async`intestazione. Il tipo di `Bundle` transazione FHIR utilizza una `POST` richiesta con autorizzazione AWS Signature Version 4 o SMART on FHIR. Il seguente esempio di codice utilizza lo strumento da riga di `curl` comando a scopo dimostrativo.

   SigV4

   Autorizzazione 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

Esempio di autorizzazione SMART on FHIR per il tipo di dati `IdentityProviderConfiguration`.

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

Il chiamante può assegnare le autorizzazioni nella lambda di autorizzazione. Per ulteriori informazioni, consulta OAuth Cannocchiali 2.0.

4. In caso di invio riuscito, il server restituisce HTTP 202 Accepted. L'intestazione della `content-location` risposta contiene l'URL del sondaggio. Il corpo della risposta è una `OperationOutcome` risorsa.

```
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"
            ]
        }
    ]
}
```

Sondaggio sullo stato delle transazioni

Dopo aver inviato una transazione asincrona, utilizza l'URL di polling dall'intestazione della `content-location` risposta per verificare lo stato della transazione. Invia una `GET` richiesta all'URL del sondaggio.

> **Note**
>
> Per gli archivi dati compatibili con SMART on FHIR, il token di autorizzazione deve includere `read` le autorizzazioni sul tipo di `Transaction` risorsa per verificare lo stato della transazione. Per ulteriori informazioni sugli ambiti SMART on FHIR, vedere. SMART su ambiti FHIR OAuth 2.0 supportati da HealthLake

Invia una `GET` richiesta all'URL del sondaggio. L'esempio seguente utilizza lo strumento da `curl` riga di comando.

SigV4

Autorizzazione 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

Autorizzazione SMART su FHIR. Il token di autorizzazione deve includere `read` le autorizzazioni sul tipo di `Transaction` risorsa.

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

La tabella seguente descrive le possibili risposte.

Codici di risposta ai sondaggi

Stato HTTP	Significato	Corpo di risposta
202 Accettato	La transazione è in coda	`OperationOutcome` con diagnostica «INVIATA»
202 Accettato	La transazione è in fase di elaborazione	`OperationOutcome` con diagnostica «IN_PROGRESS»
200 OK	Transazione completata con successo	`Bundle`con tipo `transaction-response`
4xx/5xx	Transazione fallita	`OperationOutcome` con dettagli sull'errore

Gli esempi seguenti mostrano ogni tipo di risposta.

Transazione in coda (202)

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

```
            "diagnostics": "SUBMITTED"
        }
    ]
}
```

Elaborazione delle transazioni (202)

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

Transazione completata (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"
            }
        }
```

```
    ]
}
```

Transazione non riuscita (4xx/5xx)

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

Ordine di elaborazione

I pacchetti di tipo asincrono vengono messi in coda ma non `transaction` vengono elaborati in un rigoroso ordine di invio. HealthLake ottimizza l'elaborazione in base alla capacità disponibile e al carico di sistema.

> **Important**
>
> Non dipendono dal fatto che le transazioni vengano elaborate nell'ordine in cui sono state inviate. Ad esempio, se invii la Transazione A alle 10:00 e la Transazione B alle 10:01, la Transazione B potrebbe essere completata prima della Transazione A. Progetta la tua applicazione per:
>
> - Gestire out-of-order il completamento.
> - Utilizza l'URL del sondaggio per tracciare ogni transazione in modo indipendente.
> - Implementa il sequenziamento a livello di applicazione se l'ordine è importante per il tuo caso d'uso.

Quote e limitazioni

Le quote e i limiti di velocità seguenti si applicano alle transazioni asincrone.

Quote di transazione asincrone

Quota	Valore	Regolabile
Numero massimo di operazioni per transazione asincrona	500	No
Numero massimo di transazioni in sospeso per archivio dati	500	Sì

- Le transazioni asincrone condividono gli stessi limiti di velocità API definiti in. Service Quotas
- Il polling sullo stato delle transazioni condivide gli stessi limiti di velocità API delle operazioni read (`GET`) sulle risorse FHIR.
- Se viene raggiunto il limite di transazioni in sospeso, gli invii successivi restituiscono un errore fino al completamento delle transazioni esistenti.

Gestione degli errori

Per un pacchetto di «transazioni», tutte le risorse FHIR contenute nel pacchetto vengono elaborate come un'operazione atomica. Tutte le risorse dell'operazione devono avere esito positivo, altrimenti non viene elaborata alcuna operazione nel pacchetto.

Gli errori si dividono in due categorie: gli errori di invio che vengono HealthLake restituiti in modo sincrono e gli errori di elaborazione recuperati tramite il polling.

Errori di invio

HealthLake convalida il pacchetto al momento dell'invio e restituisce gli errori in modo sincrono prima che la transazione venga messa in coda. Gli errori di invio includono errori di convalida delle risorse FHIR non validi, tipi di risorse non supportati, superamento del limite di 500 operazioni e utilizzo dell'intestazione con pacchetti batch. `Prefer: respond-async` Se è stato raggiunto il limite di transazioni in sospeso per l'archivio dati, restituisce un. HealthLake `ThrottlingException` Quando si verifica un errore di invio, la transazione non verrà messa in coda.

Errori di elaborazione

Gli errori di elaborazione si verificano dopo che la transazione è stata messa in coda e vengono restituiti tramite l'URL di polling. Questi includono i conflitti di transazione, in cui un'altra operazione

ha modificato una risorsa che fa parte della transazione, e gli errori del server durante l'elaborazione. Quando si verifica un errore di elaborazione, non viene effettuata alcuna modifica delle risorse della transazione. L'URL di polling restituirà un `OperationOutcome` con i dettagli dell'errore.

Eliminazione di una risorsa FHIR

L'`delete`interazione FHIR rimuove una risorsa FHIR esistente da un HealthLake archivio dati. Per ulteriori informazioni, consulta la documentazione dell'`delete`API FHIR RESTful R4.

Per eliminare una risorsa FHIR

1. Raccogli HealthLake `region` e `datastoreId` valorizza. Per ulteriori informazioni, consulta Ottenere le proprietà dell'archivio dati.
2. Determina il tipo di FHIR `Resource` da eliminare e raccogli il `id` valore associato. Per ulteriori informazioni, consulta Tipi di risorse.
3. Costruisci un URL per la richiesta utilizzando i valori raccolti per HealthLake `region` e. `datastoreId` Includi anche il `Resource` tipo FHIR e il relativo associato. `id` Per visualizzare l'intero percorso dell'URL nell'esempio seguente, scorri il pulsante Copia.

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

4. Inviare la richiesta . L'`delete`interazione FHIR utilizza una `DELETE` richiesta con autorizzazione AWS Signature Version 4 o SMART on FHIR. L'`curl`esempio seguente rimuove una `Patient` risorsa FHIR esistente da un HealthLake archivio dati. Per visualizzare l'intero esempio, scorrete il pulsante Copia.

   SigV4

   Autorizzazione 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'
   ```

Il server restituisce un codice di stato 204 HTTP che conferma che la risorsa è stata rimossa dall' HealthLake archivio dati. Se una richiesta di eliminazione ha esito negativo, riceverai una 400 serie di codici di stato HTTP che indica il motivo per cui la richiesta non è riuscita.

SMART on FHIR

Esempio di autorizzazione SMART on FHIR per il tipo di `IdentityProviderConfiguration`dati.

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

Il chiamante può assegnare le autorizzazioni nella lambda di autorizzazione. Per ulteriori informazioni, consulta OAuth Cannocchiali 2.0.

AWS Console

1. Accedi alla pagina Esegui query sulla console. HealthLake

2. Nella sezione Impostazioni della query, effettua le seguenti selezioni.

- Data Store ID: scegli un ID del data store per generare una stringa di query.
- Tipo di query: scegli`Delete`.
- Tipo di risorsa: scegli il tipo di risorsa FHIR da eliminare.

- ID risorsa: immettere l'ID della risorsa FHIR.

3. Scegli Esegui query.

Eliminazione delle risorse FHIR in base a condizioni

L'eliminazione condizionale è particolarmente utile quando non si conosce l'ID specifico della risorsa FHIR ma si hanno altre informazioni di identificazione sulla risorsa che si desidera eliminare.

L'eliminazione condizionale consente di eliminare una risorsa esistente in base a criteri di ricerca anziché in base all'ID FHIR logico. Quando il server elabora la richiesta di eliminazione, esegue una ricerca utilizzando funzionalità di ricerca standard per il tipo di risorsa per risolvere un singolo ID logico per la richiesta.

Come funziona l'eliminazione condizionale

L'azione del server dipende dal numero di corrispondenze che trova:

1. Nessuna corrispondenza: il server tenta un'eliminazione ordinaria e risponde in modo appropriato (404 Not Found per una risorsa inesistente, 204 Nessun contenuto per una risorsa già eliminata)
2. Una risposta: il server esegue un'eliminazione ordinaria sulla risorsa corrispondente
3. Corrispondenze multiple: restituisce un errore 412 Precondition Failed che indica che i criteri del client non erano sufficientemente selettivi

Scenari di risposta

AWS HealthLake gestisce le operazioni di eliminazione condizionale con i seguenti modelli di risposta:

Operazioni riuscite

- Quando i criteri di ricerca identificano correttamente una singola risorsa attiva, il sistema restituisce 204 No Content dopo aver completato l'eliminazione, proprio come le operazioni di eliminazione standard.

Eliminazione condizionale basata su ID

Quando si esegue l'eliminazione condizionale in base `id` a parametri aggiuntivi (`createdAt`,`tag`, o): `_lastUpdated`

- 204 Nessun contenuto: la risorsa è già stata eliminata
- 404 Not Found: La risorsa non esiste
- 409 Conflitto: l'ID corrisponde ma gli altri parametri non corrispondono

Non-ID-Based Eliminazione condizionale

Quando non `id` viene fornito o quando si utilizzano parametri diversi da `createdAttag`, oppure`_lastUpdated`:

- 404 non trovato: Nessuna corrispondenza trovata

Situazioni di conflitto

Diversi scenari generano 412 risposte non riuscite alla condizione preliminare:

- Diverse risorse corrispondono ai criteri di ricerca (criteri non sufficientemente specifici)
- La versione è in conflitto quando si utilizzano ETag intestazioni con `If-Match`
- Aggiornamenti delle risorse che si verificano tra le operazioni di ricerca ed eliminazione

Esempio di eliminazione condizionale riuscita

L'esempio seguente elimina una risorsa Patient in base a criteri specifici:

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

Questa richiesta elimina una risorsa Patient in cui:

- Il nome è «peter»
- La data di nascita è il 1 gennaio 2000
- Il numero di telefono è 1234567890

Best practice

1. Utilizza criteri di ricerca specifici per evitare corrispondenze multiple ed evitare 412 errori.
2. Prendi in considerazione le ETag intestazioni per il controllo della versione quando necessario per gestire le modifiche simultanee.
3. Gestisci le risposte agli errori in modo appropriato:
   - Per 404: perfeziona i criteri di ricerca
   - Per 412: rendi i criteri più specifici o risolvi i conflitti di versione
4. Preparatevi ai conflitti temporali in ambienti ad alta concorrenza in cui le risorse possono essere modificate tra le operazioni di ricerca ed eliminazione.

Idempotenza e concorrenza

Chiavi di idempotenza

AWS HealthLake supporta le chiavi di idempotenza per le POST operazioni FHIR, fornendo un meccanismo robusto per garantire l'integrità dei dati durante la creazione delle risorse. Includendo un UUID univoco come chiave di idempotenza nell'intestazione della richiesta, le applicazioni sanitarie possono garantire che ogni risorsa FHIR venga creata esattamente una volta, anche in scenari che prevedono instabilità di rete o tentativi automatici.

Questa funzionalità è particolarmente importante per i sistemi sanitari in cui la duplicazione delle cartelle cliniche potrebbe avere gravi conseguenze. Quando una richiesta viene ricevuta con la stessa chiave di idempotenza di una richiesta precedente, HealthLake restituirà la risorsa originale invece di crearne una duplicata. Ad esempio, ciò potrebbe verificarsi durante un ciclo di tentativi o a causa di pipeline di richieste ridondanti. L'utilizzo della chiave di idempotenza consente di mantenere la coerenza dei dati fornendo HealthLake al contempo un'esperienza senza interruzioni per le applicazioni client che gestiscono problemi di connettività intermittenti.

Implementazione

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

Scenari di risposta

Prima richiesta (201 create)

- Nuova risorsa creata con successo
- La risposta include l'ID della risorsa

Richiesta duplicata (409 Conflict)

- È stata rilevata la stessa chiave di idempotenza
- Risorsa originale restituita
- Nessuna nuova risorsa creata

Richiesta non valida (400 Richiesta errata)

- UUID malformato
- Campi obbligatori mancanti

Best practice

- Genera un UUID univoco per ogni nuova creazione di risorse
- Memorizza le chiavi di idempotenza per la logica dei tentativi
- Usa un formato di chiave coerente: UUID v4 consigliato
- Implementa nelle applicazioni client per la gestione della creazione di risorse

> **Note**
>
> Questa funzionalità è particolarmente utile per i sistemi sanitari che richiedono una rigorosa precisione dei dati e impediscono la duplicazione delle cartelle cliniche.

ETag in AWS HealthLake

AWS HealthLake utilizza ETags un controllo ottimistico della concorrenza nelle risorse FHIR, fornendo un meccanismo affidabile per gestire le modifiche simultanee e mantenere la coerenza dei dati. An ETag è un identificatore univoco che rappresenta una versione specifica di una risorsa, che funziona come un sistema di controllo della versione tramite intestazioni HTTP. Durante la lettura o la modifica delle risorse, le applicazioni possono utilizzarle ETags per prevenire sovrascritture

involontarie e garantire l'integrità dei dati, in particolare in scenari con potenziali aggiornamenti simultanei.

Esempio di implementazione

```
// 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
```

Scenari di risposta

Operazione riuscita (200 OK o 204 Nessun contenuto)

- ETag corrisponde alla versione corrente
- L'operazione procede come previsto

Conflitto di versione (condizione preliminare 412 non riuscita)

- ETag non corrisponde alla versione corrente
- Aggiornamento rifiutato per prevenire la perdita di dati

Best practice

- ETags Includi in tutte le operazioni di aggiornamento ed eliminazione
- Implementa la logica dei tentativi per la gestione dei conflitti di versione
- Usa If-None-Match: * per gli scenari create-if-not-exists
- Verifica sempre la ETag freschezza prima delle modifiche

Questo sistema di controllo simultaneo è essenziale per mantenere l'integrità dei dati sanitari, specialmente in ambienti con più utenti o sistemi che accedono e modificano le stesse risorse.

Ricerca di risorse FHIR in AWS HealthLake

Utilizza l'`search`interazione FHIR per cercare un set di risorse FHIR in un archivio HealthLake dati in base ad alcuni criteri di filtro. L'`search`interazione può essere eseguita utilizzando una richiesta `GET` or`POST`. Per le ricerche che coinvolgono informazioni di identificazione personale (PII) o informazioni sanitarie protette (PHI), si consiglia di utilizzare `POST` le richieste, poiché PII e PHI vengono aggiunti come parte del corpo della richiesta e vengono crittografati durante il transito.

> **Note**
>
> L'`search`interazione FHIR descritta in questo capitolo è costruita in conformità allo standard FHIR R4 per lo HL7 scambio di dati sanitari. Poiché si tratta di una rappresentazione di un servizio HL7 FHIR, non viene offerto tramite e. AWS CLI AWS SDKs Per ulteriori informazioni, consulta la `search`documentazione dell'API FHIR R4 RESTful .

HealthLake supporta un sottoinsieme di parametri di ricerca FHIR R4. Per ulteriori informazioni, consulta Parametri di ricerca FHIR R4 per HealthLake.

Argomenti

- Ricerca di risorse FHIR con GET
- Ricerca di risorse FHIR con POST
- Livelli di coerenza della ricerca FHIR

Ricerca di risorse FHIR con GET

È possibile utilizzare `GET` le richieste per effettuare ricerche in un archivio HealthLake dati. Durante l'utilizzo`GET`, HealthLake supporta la fornitura di parametri di ricerca come parte dell'URL, ma non come parte del corpo della richiesta. Per ulteriori informazioni, consulta Parametri di ricerca FHIR R4 per HealthLake.

> **Importante**
>
> Per le ricerche che coinvolgono informazioni di identificazione personale (PII) o informazioni sanitarie protette (PHI), le migliori pratiche di sicurezza richiedono l'utilizzo delle `POST`

richieste, poiché PII e PHI vengono aggiunti come parte del corpo della richiesta e vengono crittografati durante il transito. Per ulteriori informazioni, consulta Ricerca di risorse FHIR con POST.

La procedura seguente è seguita da esempi da utilizzare GET per effettuare ricerche in un archivio dati. HealthLake

Per effettuare ricerche in un HealthLake data store con **GET**

1. Raccogli HealthLake `region` e `datastoreId` valorizza. Per ulteriori informazioni, consulta Ottenere le proprietà dell'archivio dati.
2. Determina il tipo di risorsa FHIR da cercare e raccogliere il `id` valore associato. Per ulteriori informazioni, consulta Tipi di risorse.
3. Costruisci un URL per la richiesta utilizzando i valori raccolti per HealthLake `region` e. `datastoreId` Includi anche il `Resource` tipo FHIR e i parametri di ricerca supportati. Per visualizzare l'intero percorso dell'URL nell'esempio seguente, scorri il pulsante Copia.

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

4. Invia la GET richiesta con l'autorizzazione AWS Signature Version 4 o SMART on FHIR. L'`curl`esempio seguente restituisce il numero totale di `Patient` risorse in un archivio HealthLake dati. Per visualizzare l'intero esempio, scorri il pulsante Copia.

   SigV4

   Autorizzazione 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

Esempio di autorizzazione SMART on FHIR per il tipo di dati `IdentityProviderConfiguration`.

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

Il chiamante può assegnare le autorizzazioni nella lambda di autorizzazione. Per ulteriori informazioni, consulta OAuth Cannocchiali 2.0.

AWS Console

> **Note**
>
> La HealthLake console supporta solo l'autorizzazione SigV4. L'autorizzazione SMART on FHIR è supportata tramite e. AWS CLI AWS SDKs

1. Accedi alla pagina Esegui interrogazione sulla HealthLake console.

2. Nella sezione Impostazioni della query, effettua le seguenti selezioni.

- Data Store ID: scegli un ID del data store per generare una stringa di query.

- Tipo di query: scegli`Search with GET`.
- Tipo di risorsa: scegli il tipo di risorsa FHIR su cui eseguire la ricerca.
- Parametri di ricerca: seleziona un parametro di ricerca o una combinazione di parametri di ricerca per concentrare la ricerca su record specifici.

3. Scegli Esegui query.

Esempi: ricerca con GET

Le schede seguenti forniscono esempi per la ricerca su tipi di risorse FHIR specifici con. `GET` Gli esempi mostrano come specificare i parametri di ricerca nella richiesta. URLs

> **Note**
>
> La HealthLake console supporta solo l'autorizzazione SigV4. L'autorizzazione SMART on FHIR è supportata tramite e. AWS CLI AWS SDKs
> HealthLake supporta un sottoinsieme di parametri di ricerca FHIR R4. Per ulteriori informazioni, consulta Parametri di ricerca.

Patient (age)

Sebbene l'età non sia un tipo di risorsa definito in FHIR, viene acquisita come elemento del tipo di risorsa. `Patient` Utilizzate l'esempio seguente per effettuare una richiesta di ricerca `GET` basata sui tipi di `Patient`risorse utilizzando l'elemento BirthDate e il `eq` comparatore di ricerca per cercare individui nati nel 1997.

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

Condition

Utilizzate l'esempio seguente per effettuare una `GET` richiesta sul tipo di `Condition`risorsa. La ricerca trova le condizioni nell'archivio HealthLake dati che contengono il codice medico SNOMED`72892002`, che si traduce in. `Normal pregnancy`

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

DocumentationReference

L'esempio seguente mostra come creare una GET richiesta sul tipo di DocumentReferencerisorsa per Patient una o più persone con diagnosi di streptococco e a cui è stata prescritta anche amoxicillina.

```
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

Utilizzate l'esempio seguente per effettuare una richiesta sul tipo di risorsa. GET Location La ricerca seguente trova le posizioni nel tuo archivio HealthLake dati che contengono il nome della città Boston come parte dell'indirizzo.

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

Observation

Utilizza l'esempio seguente per effettuare una richiesta di ricerca GET basata sul tipo di Observationrisorsa. Questa ricerca utilizza il parametro value-concept di ricerca per cercare il codice medico266919005, che si traduce inNever smoker.

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

Ricerca di risorse FHIR con POST

Puoi utilizzare l'searchinterazione FHIR con POST le richieste di ricerca in un archivio HealthLake dati. Quando si utilizzaPOST, HealthLake supporta i parametri di ricerca nell'URL o nel corpo di una richiesta, ma non è possibile utilizzarli entrambi in una singola richiesta.

> **Importante**
>
> Per le ricerche che coinvolgono informazioni di identificazione personale (PII) o informazioni sanitarie protette (PHI), le migliori pratiche di sicurezza richiedono l'utilizzo delle `POST` richieste, poiché PII e PHI vengono aggiunti come parte del corpo della richiesta e vengono crittografati durante il transito.

La procedura seguente è seguita da esempi che utilizzano l'interazione FHIR R4 `search` con per effettuare ricerche in un archivio dati. `POST` HealthLake Gli esempi mostrano come specificare i parametri di ricerca nel corpo della richiesta JSON.

Per effettuare ricerche in un archivio HealthLake dati con **`POST`**

1. Raccogli HealthLake `region` e `datastoreId` valorizza. Per ulteriori informazioni, consulta Ottenere le proprietà dell'archivio dati.
2. Determina il tipo di risorsa FHIR da cercare e raccogliere il `id` valore associato. Per ulteriori informazioni, consulta Tipi di risorse.
3. Costruisci un URL per la richiesta utilizzando i valori raccolti per HealthLake `region` e. `datastoreId` Includi anche il `Resource` tipo e `_search` l'interazione FHIR. Per visualizzare l'intero percorso dell'URL nell'esempio seguente, scorri il pulsante Copia.

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

4. Costruisci un corpo JSON per la richiesta, specificando i dati FHIR da cercare. Ai fini di questa procedura, cercherai `Observation` risorse per scoprire pazienti che non hanno mai fumato. Per specificare lo stato del codice medico`Never smoker`, impostalo `value-concept=266919005` nel corpo della richiesta JSON. Salva il file con nome `search-observation.json`.

   ```
   value-concept=266919005
   ```

5. Inviare la richiesta . L'`search`interazione FHIR utilizza la `GET` richiesta con l'autorizzazione AWS Signature Version 4 o SMART on FHIR.

> **Note**
>
> Quando si effettua una POST richiesta con parametri di ricerca nel corpo della richiesta, usala `Content-Type: application/x-www-form-urlencoded` come parte dell'intestazione.

L'`curl`esempio seguente effettua una richiesta di ricerca basata su POST sul tipo di `Observation` risorsa. La richiesta utilizza il parametro `value-concept`di ricerca per cercare il codice medico `266919005` che indichi il valore`Never smoker`. Per visualizzare l'intero esempio, scorri il pulsante Copia.

SigV4

Autorizzazione 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

Esempio di autorizzazione SMART on FHIR per il tipo di dati `IdentityProviderConfiguration`.

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

Il chiamante può assegnare le autorizzazioni nella lambda di autorizzazione. Per ulteriori informazioni, consulta OAuth Cannocchiali 2.0.

Esempi: ricerca con POST

Le schede seguenti forniscono esempi per la ricerca su tipi di risorse FHIR specifici con. `POST` Gli esempi mostrano come specificare una richiesta in. URLs

> **Note**
>
> La HealthLake console supporta solo l'autorizzazione SIGv4. L'autorizzazione SMART on FHIR è supportata tramite e. AWS CLI AWS SDKs
> HealthLake supporta un sottoinsieme di parametri di ricerca FHIR R4. Per ulteriori informazioni, consulta Parametri di ricerca.

Patient (age)

Sebbene l'età non sia un tipo di risorsa definito in FHIR, viene acquisita come elemento del tipo di risorsa. `Patient` Utilizzate l'esempio seguente per effettuare una richiesta di ricerca `POST` basata sul tipo di `Patient` risorsa. L'esempio di ricerca seguente utilizza il `eq` comparatore di ricerca per cercare persone nate nel 1997.

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

Per specificare l'anno 1997 nella ricerca, aggiungete il seguente elemento al corpo della richiesta.

```
birthdate=eq1997
```

Condition

Utilizzare quanto segue per effettuare una POST richiesta sul tipo di `Condition` risorsa. Questa ricerca consente di trovare le posizioni nell'archivio HealthLake dati che contengono il codice medico72892002.

È necessario specificare l'URL della richiesta e il corpo della richiesta. Ecco un esempio di URL di richiesta.

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

Per specificare il codice medico da cercare, aggiungi il seguente elemento JSON al corpo della richiesta.

```
code=72892002
```

DocumentReference

Per visualizzare i risultati dell'elaborazione integrata HealthLake del linguaggio naturale (NLP) quando si effettua una POST richiesta sul tipo di `DocumentReference` risorsa, formatta una richiesta come segue.

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

Per specificare i parametri di `DocumentReference` ricerca a cui fare riferimento, vedere. Tipi di parametri di ricerca La seguente stringa di query utilizza più parametri di ricerca per cercare nelle operazioni dell'API Amazon Comprehend Medical utilizzate per generare i risultati NLP integrati.

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

Location

Usa l'esempio seguente per effettuare una POST richiesta sul tipo di risorsa. `Location` La ricerca trova le posizioni nell'archivio HealthLake dati che contengono il nome della città Boston come parte dell'indirizzo.

È necessario specificare un URL e un corpo della richiesta. Ecco un esempio di URL di richiesta.

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

Per specificare `Boston` nella ricerca, aggiungi il seguente elemento al corpo della richiesta:

```
address=Boston
```

Observation

Utilizzate l'esempio seguente per effettuare una richiesta di ricerca `POST` basata sul tipo di `Observation` risorsa. La ricerca utilizza il parametro `value-concept` di ricerca per cercare il codice medico, `266919005` che indica`Never smoker`.

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

Per specificare lo stato`Never smoker`, imposta `value-concept=266919005` nel corpo del file JSON.

```
value-concept=266919005
```

Livelli di coerenza della ricerca FHIR

L'indice HealthLake di ricerca di AWS opera su un modello Eventual Consistency per `GET` e `POST` con le operazioni SEARCH. Con una certa coerenza, se c'è un aggiornamento dell'indice di ricerca in sospeso per una risorsa, i risultati della ricerca escludono la versione N-1 della risorsa fino al completamento dell'aggiornamento dell'indice.

AWS HealthLake ora include la possibilità di selezionare il comportamento del modello di coerenza per le risorse aggiornate. Gli sviluppatori possono includere «Eventual Consistency», il comportamento predefinito descritto sopra o «Strong Consistency». Strong Consistency consentirà di includere nei risultati di ricerca la versione N-1 della risorsa per le risorse con aggiornamenti dell'indice di ricerca in sospeso. Questo può essere utilizzato in scenari d'uso in cui tutte le risorse sono necessarie nel risultato anche quando l'aggiornamento dell'indice di ricerca non è ancora stato completato. I clienti possono controllare questo comportamento utilizzando l'intestazione della `x-amz-fhir-history-consistency-level` richiesta.

Livelli di coerenza

Forte coerenza

Impostato `x-amz-fhir-history-consistency-level: strong` per restituire tutti i record corrispondenti, inclusi quelli con aggiornamenti dell'indice di ricerca in sospeso. Utilizzate questa opzione quando avete bisogno di cercare risorse subito dopo gli aggiornamenti.

Consistenza finale

Impostata `x-amz-fhir-history-consistency-level: eventual` per restituire solo i record che hanno completato gli aggiornamenti dell'indice di ricerca. Questo è il comportamento predefinito se non viene specificato alcun livello di coerenza.

Esempio di utilizzo

1. Quando si aggiorna una risorsa:

```
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. Ricerca successiva:

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

Best practice

- Usa una forte coerenza quando devi cercare immediatamente risorse aggiornate di recente
- Utilizza la coerenza finale per le domande generiche in cui la visibilità immediata non è fondamentale
- Considerate il compromesso tra visibilità immediata e potenziale impatto sulle prestazioni

> **Note**
>
> L'impostazione del livello di coerenza influisce sulla velocità di visualizzazione delle risorse aggiornate nei risultati di ricerca, ma non sull'effettiva archiviazione delle risorse.
> L'impostazione dell'`x-amz-fhir-history-consistency-level`intestazione opzionale su «strong» raddoppia il consumo di capacità di scrittura per risorsa.
> Questa funzionalità è applicabile solo ai data store per cui è abilitata la cronologia delle versioni (tutti i datastore creati dopo il 25 ottobre 2024 l'hanno abilitata per impostazione predefinita).

Esportazione di dati FHIR con AWS HealthLake

Utilizza AWS HealthLake le azioni native per avviare, descrivere ed elencare i lavori di esportazione FHIR. È possibile mettere in coda i lavori di esportazione. I lavori di esportazione asincroni vengono elaborati in modo FIFO (First In First Out). È possibile mettere in coda i lavori nello stesso modo in cui si iniziano i lavori di esportazione. Se ne è in corso uno, si metterà semplicemente in coda. È possibile creare, leggere, aggiornare o eliminare risorse FHIR mentre è in corso un processo di esportazione.

> **Note**
>
> È inoltre possibile esportare i dati FHIR da un archivio HealthLake dati utilizzando l'operazione FHIR R4. `$export` Per ulteriori informazioni, consulta Esportazione di HealthLake dati con FHIR `$export`.

Argomenti

- Avvio di un lavoro di esportazione FHIR
- Far esportare proprietà lavorative dalla FHIR
- Elenco dei lavori di esportazione FHIR

Avvio di un lavoro di esportazione FHIR

Utilizzato `StartFHIRExportJob` per avviare un processo di esportazione FHIR da un archivio HealthLake dati. I seguenti menu forniscono una procedura per Console di gestione AWS e alcuni esempi di codice per and. AWS CLI AWS SDKs Per ulteriori informazioni, consulta `StartFHIRExportJob` nella documentazione di riferimento dell'API AWS HealthLake .

> **Nota**
>
> HealthLake supporta la specifica FHIR R4 per lo scambio di dati sanitari. Pertanto, tutti i dati sanitari vengono esportati in formato FHIR R4.

Come avviare un processo di esportazione FHIR

Scegli un menu in base alle tue preferenze di accesso a. AWS HealthLake

AWS CLI e SDKs

CLI

AWS CLI

Come avviare un processo di esportazione FHIR

L'`start-fhir-export-job`esempio seguente mostra come avviare un processo di esportazione FHIR utilizzando AWS HealthLake.

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

Output:

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

- Per i dettagli sull'API, consulta Start FHIRExport Job in AWS CLI Command Reference.

Python

SDK per 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
```

- Per i dettagli sull'API, consulta Start FHIRExport Job in AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

SAP ABAP

SDK per SAP ABAP

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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.
```

- Per i dettagli sull'API, consulta Start FHIRExport Job in AWS SDK for SAP ABAP API reference.

> **Esempio di disponibilità**
>
> Non riesci a trovare quello che ti serve? Richiedi un esempio di codice utilizzando il link Fornisci feedback nella barra laterale destra di questa pagina.

AWS Console

1. Accedi alla pagina degli archivi dati sulla HealthLake console.
2. Scegli un data store.
3. Scegli Export (Esporta).

   Viene visualizzata la pagina Esporta.
4. Nella sezione Dati di output, inserisci le seguenti informazioni:

   - Posizione dei dati di output in Amazon S3
   - Crittografia dell'output
5. Nella sezione Autorizzazioni di accesso, scegli Usa un ruolo di servizio IAM esistente e seleziona il ruolo dal menu Nome ruolo o scegli Crea un ruolo IAM.

6. Seleziona Esporta dati.

> **Note**
>
> Durante l'esportazione, scegli Copia l'ID del lavoro sul banner nella parte superiore della pagina. Puoi utilizzare il `JobID`per richiedere le proprietà del lavoro di esportazione utilizzando AWS CLI. Per ulteriori informazioni, consulta Far esportare proprietà lavorative dalla FHIR.

Far esportare proprietà lavorative dalla FHIR

`DescribeFHIRExportJob`Utilizzatelo per esportare le proprietà del lavoro da un archivio HealthLake dati. I seguenti menu forniscono una procedura per Console di gestione AWS e alcuni esempi di codice per AWS CLI and AWS SDKs. Per ulteriori informazioni, consulta `DescribeFHIRExportJob` nella documentazione di riferimento dell'API AWS HealthLake .

> **Nota**
>
> HealthLake supporta la specifica FHIR R4 per lo scambio di dati sanitari. Pertanto, tutti i dati sanitari vengono esportati in formato FHIR R4.

Come descrivere un processo di esportazione FHIR

Scegli un menu in base alle tue preferenze di accesso a. AWS HealthLake

AWS CLI e SDKs

CLI

AWS CLI

Come descrivere un processo di esportazione FHIR

L'`describe-fhir-export-job`esempio seguente mostra come trovare le proprietà di un processo di esportazione FHIR in AWS HealthLake.

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

```
    --job-id 9b9a51943afaedd0a8c0c26c49135a31
```

Output:

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

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

- Per i dettagli sull'API, consulta Descrivi FHIRExport Job in AWS CLI Command Reference.

Python

SDK per 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
```

- Per i dettagli sull'API, consulta Descrivi FHIRExport Job in AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

SAP ABAP

SDK per SAP ABAP

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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.
```

- Per i dettagli sull'API, consulta Descrivi FHIRExport Job in AWS SDK per il riferimento all'API SAP ABAP.

> **Esempio di disponibilità**
>
> Non riesci a trovare quello che ti serve? Richiedi un esempio di codice utilizzando il link Fornisci feedback nella barra laterale destra di questa pagina.

AWS Console

> **Note**
>
> Le informazioni sul processo di esportazione FHIR non sono disponibili sulla HealthLake console. Utilizza invece AWS CLI with `DescribeFHIRExportJob` per richiedere proprietà

del lavoro di esportazione come `JobStatus`. Per ulteriori informazioni, consulta l' AWS CLI esempio in questa pagina.

Elenco dei lavori di esportazione FHIR

`ListFHIRExportJobs`Da utilizzare per elencare i lavori di esportazione FHIR per un archivio HealthLake dati. I seguenti menu forniscono una procedura Console di gestione AWS e alcuni esempi di codice per and. AWS CLI AWS SDKs Per ulteriori informazioni, consulta `ListFHIRExportJobs` nella documentazione di riferimento dell'API AWS HealthLake .

> **Nota**
>
> HealthLake supporta la specifica FHIR R4 per lo scambio di dati sanitari. Pertanto, tutti i dati sanitari vengono esportati in formato FHIR R4.

Per elencare i lavori di esportazione FHIR

Scegliete un menu in base alle vostre preferenze di accesso a AWS HealthLake.

AWS CLI e SDKs

CLI

AWS CLI

Come elencare tutti i processi di esportazione FHIR

L'esempio `list-fhir-export-jobs` seguente mostra come utilizzare il comando per visualizzare un elenco di processi di esportazione associati a un account.

```
aws healthlake list-fhir-export-jobs \
    --datastore-id (Data store ID) \
    --submitted-before (DATE like 2024-10-13T19:00:00Z)\
    --submitted-after (DATE like 2020-10-13T19:00:00Z )\
    --job-name "FHIR-EXPORT" \
    --job-status SUBMITTED  \
    --max-results (Integer between 1 and 500)
```

Output:

```
{
    "ExportJobPropertiesList": [
        {
            "ExportJobProperties": {
                "OutputDataConfig": {
                    "S3Uri": "s3://(Bucket Name)/(Prefix Name)/",
                    "S3Configuration": {
                        "S3Uri": "s3://(Bucket Name)/(Prefix Name)/",
                        "KmsKeyId": "(KmsKey Id)"
                    }
                },
                "DataAccessRoleArn": "arn:aws:iam::(AWS Account ID):role/(Role
 Name)",
                "JobStatus": "COMPLETED",
                "JobId": "c145fbb27b192af392f8ce6e7838e34f",
                "JobName": "FHIR-EXPORT",
                "SubmitTime": "2024-11-20T11:31:46.672000-05:00",
                "EndTime": "2024-11-20T11:34:01.636000-05:00",
                "DatastoreId": "(Data store ID)"
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta List FHIRExport Jobs in AWS CLI Command Reference.

Python

SDK per 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
```

- Per i dettagli sull'API, consulta List FHIRExport Jobs in AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

SAP ABAP

SDK per SAP ABAP

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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.
```

- Per i dettagli sull'API, consulta il riferimento all'API List FHIRExport Jobs in AWS SDK for SAP ABAP.

> **Esempio di disponibilità**
>
> Non riesci a trovare quello che ti serve? Richiedi un esempio di codice utilizzando il link Fornisci feedback nella barra laterale destra di questa pagina.

AWS Console

> **Note**
>
> Le informazioni sul processo di esportazione FHIR non sono disponibili sulla HealthLake console. Utilizza invece AWS CLI with `ListFHIRExportJobs` per elencare tutti i lavori di esportazione FHIR. Per ulteriori informazioni, fate riferimento all' AWS CLI esempio in questa pagina.

Esempi di codice per HealthLake l'utilizzo AWS SDKs

I seguenti esempi di codice mostrano come utilizzare un kit HealthLake di sviluppo AWS software (SDK).

Le azioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le azioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo HealthLake con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Esempi di codice

- Esempi di base per l' HealthLake utilizzo AWS SDKs
  - Azioni per l'utilizzo HealthLake AWS SDKs
    - Utilizzo CreateFHIRDatastore con un AWS SDK o una CLI
    - Utilizzo DeleteFHIRDatastore con un AWS SDK o una CLI
    - Utilizzo DescribeFHIRDatastore con un AWS SDK o una CLI
    - Utilizzo DescribeFHIRExportJob con un AWS SDK o una CLI
    - Utilizzo DescribeFHIRImportJob con un AWS SDK o una CLI
    - Utilizzo ListFHIRDatastores con un AWS SDK o una CLI
    - Utilizzo ListFHIRExportJobs con un AWS SDK o una CLI
    - Utilizzo ListFHIRImportJobs con un AWS SDK o una CLI
    - Utilizzo ListTagsForResource con un AWS SDK o una CLI
    - Utilizzo StartFHIRExportJob con un AWS SDK o una CLI
    - Utilizzo StartFHIRImportJob con un AWS SDK o una CLI
    - Utilizzo TagResource con un AWS SDK o una CLI
    - Utilizzo UntagResource con un AWS SDK o una CLI

Esempi di base per l' HealthLake utilizzo AWS SDKs

I seguenti esempi di codice mostrano come utilizzare le nozioni di base di AWS HealthLake with. AWS SDKs

Esempi

- Azioni per l'utilizzo HealthLake AWS SDKs
  - Utilizzo CreateFHIRDatastore con un AWS SDK o una CLI
  - Utilizzo DeleteFHIRDatastore con un AWS SDK o una CLI
  - Utilizzo DescribeFHIRDatastore con un AWS SDK o una CLI
  - Utilizzo DescribeFHIRExportJob con un AWS SDK o una CLI
  - Utilizzo DescribeFHIRImportJob con un AWS SDK o una CLI
  - Utilizzo ListFHIRDatastores con un AWS SDK o una CLI
  - Utilizzo ListFHIRExportJobs con un AWS SDK o una CLI
  - Utilizzo ListFHIRImportJobs con un AWS SDK o una CLI
  - Utilizzo ListTagsForResource con un AWS SDK o una CLI
  - Utilizzo StartFHIRExportJob con un AWS SDK o una CLI
  - Utilizzo StartFHIRImportJob con un AWS SDK o una CLI
  - Utilizzo TagResource con un AWS SDK o una CLI
  - Utilizzo UntagResource con un AWS SDK o una CLI

Azioni per l'utilizzo HealthLake AWS SDKs

I seguenti esempi di codice mostrano come eseguire singole HealthLake azioni con AWS SDKs. Ogni esempio include un collegamento a GitHub, dove sono disponibili le istruzioni per la configurazione e l'esecuzione del codice.

Gli esempi seguenti includono solo le azioni più comunemente utilizzate. Per un elenco completo, consulta la documentazione di riferimento dell'API AWS HealthLake.

Esempi

- Utilizzo CreateFHIRDatastore con un AWS SDK o una CLI
- Utilizzo DeleteFHIRDatastore con un AWS SDK o una CLI
- Utilizzo DescribeFHIRDatastore con un AWS SDK o una CLI
- Utilizzo DescribeFHIRExportJob con un AWS SDK o una CLI
- Utilizzo DescribeFHIRImportJob con un AWS SDK o una CLI
- Utilizzo ListFHIRDatastores con un AWS SDK o una CLI
- Utilizzo ListFHIRExportJobs con un AWS SDK o una CLI

- Utilizzo ListFHIRImportJobs con un AWS SDK o una CLI
- Utilizzo ListTagsForResource con un AWS SDK o una CLI
- Utilizzo StartFHIRExportJob con un AWS SDK o una CLI
- Utilizzo StartFHIRImportJob con un AWS SDK o una CLI
- Utilizzo TagResource con un AWS SDK o una CLI
- Utilizzo UntagResource con un AWS SDK o una CLI

Utilizzo **`CreateFHIRDatastore`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `CreateFHIRDatastore`.

CLI

AWS CLI

Esempio 1: creare un archivio dati compatibile con SigV4 HealthLake

L'`create-fhir-datastore`esempio seguente mostra come creare un nuovo archivio dati in. AWS HealthLake

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

Output:

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

Esempio 2: creare uno SMART su un data store abilitato per FHIR HealthLake

L'`create-fhir-datastore`esempio seguente mostra come creare un nuovo SMART su un data store abilitato per FHIR in. 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
```

Contenuto di `identity_provider_configuration.json`:

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

Output:

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

- Per i dettagli sull'API, consulta Create FHIRDatastore in AWS CLI Command Reference.

Python

SDK per 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
```

Il codice seguente mostra un esempio di parametri per un archivio dati SMART su 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
            )
```

- Per i dettagli sull'API, consulta Create FHIRDatastore in AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

SAP ABAP

SDK per SAP ABAP

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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.
```

- Per i dettagli sull'API, consulta Create FHIRDatastore in AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo HealthLake con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DeleteFHIRDatastore`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `DeleteFHIRDatastore`.

CLI

AWS CLI

Come eliminare un datastore FHIR

L'`delete-fhir-datastore`esempio seguente mostra come eliminare un archivio dati e tutto il suo contenuto. AWS HealthLake

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

Output:

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

- Per FHIRDatastore i dettagli sull'AWS CLI API, consulta Delete in Command Reference.

Python

SDK per 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
```

- Per i dettagli sull'API, consulta Delete FHIRDatastore in AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

SAP ABAP

SDK per SAP ABAP

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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.
```

- Per i dettagli sull'API, consulta Delete FHIRDatastore in AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo HealthLake con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DescribeFHIRDatastore`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `DescribeFHIRDatastore`.

CLI

AWS CLI

Come descrivere un datastore FHIR

L'`describe-fhir-datastore`esempio seguente mostra come trovare le proprietà di un archivio dati in. AWS HealthLake

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

Output:

```
{
    "DatastoreProperties": {
        "PreloadDataConfig": {
            "PreloadDataType": "SYNTHEA"
        },
        "SseConfiguration": {
            "KmsEncryptionConfig": {
                "CmkType": "CUSTOMER_MANAGED_KMS_KEY",
                "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
            }
        },
        "DatastoreName": "Demo",
        "DatastoreArn": "arn:aws:healthlake:us-east-1:<AWS Account ID>:datastore/
<Data store ID>",
        "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/
datastore/<Data store ID>/r4/",
        "DatastoreStatus": "ACTIVE",
        "DatastoreTypeVersion": "R4",
        "CreatedAt": 1603761064.881,
        "DatastoreId": "<Data store ID>",
        "IdentityProviderConfiguration": {
            "AuthorizationStrategy": "AWS_AUTH",
            "FineGrainedAuthorizationEnabled": false
        }
    }
}
```

- Per i dettagli sull'API, consulta Descrivi FHIRDatastore in AWS CLI Command Reference.

Python

SDK per 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
```

- Per i dettagli sull'API, consulta Descrivi FHIRDatastore in AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

SAP ABAP

SDK per SAP ABAP

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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.
```

- Per i dettagli sull'API, consulta Descrivi FHIRDatastore in AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo HealthLake con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DescribeFHIRExportJob`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `DescribeFHIRExportJob`.

CLI

AWS CLI

Come descrivere un processo di esportazione FHIR

L'`describe-fhir-export-job`esempio seguente mostra come trovare le proprietà di un processo di esportazione FHIR in. AWS HealthLake

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

Output:

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

        },
```

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

- Per i dettagli sull'API, consulta Descrivi FHIRExport Job in AWS CLI Command Reference.

Python

SDK per 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
```

- Per i dettagli sull'API, consulta Descrivi FHIRExport Job in AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

SAP ABAP

SDK per SAP ABAP

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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.
```

- Per i dettagli sull'API, consulta Descrivi FHIRExport Job in AWS SDK per il riferimento all'API SAP ABAP.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo HealthLake con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DescribeFHIRImportJob`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `DescribeFHIRImportJob`.

CLI

AWS CLI

Come descrivere un processo di importazione FHIR

L'`describe-fhir-import-job`esempio seguente mostra come apprendere le proprietà di un processo di importazione FHIR utilizzando. AWS HealthLake

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

Output:

```
{
    "ImportJobProperties": {
    "InputDataConfig": {
        "S3Uri": "s3://(Bucket Name)/(Prefix Name)/"
        { "arrayitem2": 2 }
    },
    "DataAccessRoleArn": "arn:aws:iam::(AWS Account ID):role/(Role Name)",
```

```
    "JobStatus": "COMPLETED",
    "JobId": "c145fbb27b192af392f8ce6e7838e34f",
    "SubmitTime": 1606272542.161,
    "EndTime": 1606272609.497,
    "DatastoreId": "(Data store ID)"
    }
}
```

- Per i dettagli sull'API, consulta Descrivi FHIRImport Job in AWS CLI Command Reference.

Python

SDK per 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
```

- Per i dettagli sull'API, consulta Descrivi FHIRImport Job in AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

SAP ABAP

SDK per SAP ABAP

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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.
```

- Per i dettagli sull'API, consulta Descrivi FHIRImport Job in AWS SDK per il riferimento all'API SAP ABAP.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo HealthLake con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`ListFHIRDatastores`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `ListFHIRDatastores`.

CLI

AWS CLI

Come elencare i datastore FHIR

L'`list-fhir-datastores`esempio seguente mostra come utilizzare il comando e come gli utenti possono filtrare i risultati in base allo stato del data store in AWS HealthLake.

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

Output:

```
{
    "DatastorePropertiesList": [
    {
        "PreloadDataConfig": {
```

```
            "PreloadDataType": "SYNTHEA"
        },
        "SseConfiguration": {
            "KmsEncryptionConfig": {
                "CmkType": "CUSTOMER_MANAGED_KMS_KEY",
                "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
            }
        },
        "DatastoreName": "Demo",
        "DatastoreArn": "arn:aws:healthlake:us-east-1:<AWS Account ID>:datastore/
<Data store ID>",
        "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/
datastore/<Data store ID>/r4/",
        "DatastoreStatus": "ACTIVE",
        "DatastoreTypeVersion": "R4",
        "CreatedAt": 1603761064.881,
        "DatastoreId": "<Data store ID>",
        "IdentityProviderConfiguration": {
            "AuthorizationStrategy": "AWS_AUTH",
            "FineGrainedAuthorizationEnabled": false
        }
    }
    ]
}
```

- Per i dettagli sull'API, consulta List FHIRDatastores in AWS CLI Command Reference.

Python

SDK per 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
```

- Per i dettagli sull'API, consulta List FHIRDatastores in AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

SAP ABAP

SDK per SAP ABAP

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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.
```

- Per i dettagli sulle API, consulta List FHIRDatastores in AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo HealthLake con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`ListFHIRExportJobs`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `ListFHIRExportJobs`.

CLI

AWS CLI

Come elencare tutti i processi di esportazione FHIR

L'esempio `list-fhir-export-jobs` seguente mostra come utilizzare il comando per visualizzare un elenco di processi di esportazione associati a un account.

```
aws healthlake list-fhir-export-jobs \
    --datastore-id (Data store ID) \
    --submitted-before (DATE like 2024-10-13T19:00:00Z)\
    --submitted-after (DATE like 2020-10-13T19:00:00Z )\
    --job-name "FHIR-EXPORT" \
    --job-status SUBMITTED  \
    --max-results (Integer between 1 and 500)
```

Output:

```
{
    "ExportJobPropertiesList": [
        {
            "ExportJobProperties": {
                "OutputDataConfig": {
                    "S3Uri": "s3://(Bucket Name)/(Prefix Name)/",
                    "S3Configuration": {
                        "S3Uri": "s3://(Bucket Name)/(Prefix Name)/",
                        "KmsKeyId": "(KmsKey Id)"
                    }
                },
                "DataAccessRoleArn": "arn:aws:iam::(AWS Account ID):role/(Role
 Name)",
                "JobStatus": "COMPLETED",
                "JobId": "c145fbb27b192af392f8ce6e7838e34f",
                "JobName": "FHIR-EXPORT",
                "SubmitTime": "2024-11-20T11:31:46.672000-05:00",
                "EndTime": "2024-11-20T11:34:01.636000-05:00",
                "DatastoreId": "(Data store ID)"
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta List FHIRExport Jobs in AWS CLI Command Reference.

Python

SDK per 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
```

- Per i dettagli sull'API, consulta List FHIRExport Jobs in AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

SAP ABAP

SDK per SAP ABAP

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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.
```

- Per i dettagli sull'API, consulta il riferimento all'API List FHIRExport Jobs in AWS SDK for SAP ABAP.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo HealthLake con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`ListFHIRImportJobs`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `ListFHIRImportJobs`.

CLI

AWS CLI

Come elencare tutti i processi di importazione FHIR

L'esempio `list-fhir-import-jobs` seguente mostra come utilizzare il comando per visualizzare un elenco di tutti i processi di importazione associati a un account.

```
aws healthlake list-fhir-import-jobs \
    --datastore-id (Data store ID) \
    --submitted-before (DATE like 2024-10-13T19:00:00Z) \
    --submitted-after (DATE like 2020-10-13T19:00:00Z ) \
    --job-name "FHIR-IMPORT" \
    --job-status SUBMITTED  \
    -max-results (Integer between 1 and 500)
```

Output:

```
{
    "ImportJobPropertiesList": [
        {
            "JobId": "c0fddbf76f238297632d4aebdbfc9ddf",
            "JobStatus": "COMPLETED",
            "SubmitTime": "2024-11-20T10:08:46.813000-05:00",
            "EndTime": "2024-11-20T10:10:09.093000-05:00",
            "DatastoreId": "(Data store ID)",
            "InputDataConfig": {
                "S3Uri": "s3://(Bucket Name)/(Prefix Name)/"
            },
            "JobOutputDataConfig": {
                "S3Configuration": {
                    "S3Uri": "s3://(Bucket Name)/
import/6407b9ae4c2def3cb6f1a46a0c599ec0-FHIR_IMPORT-
c0fddbf76f238297632d4aebdbfc9ddf/",
```

```
                    "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/b7f645cb-
e564-4981-8672-9e012d1ff1a0"
                }
            },
            "JobProgressReport": {
                "TotalNumberOfScannedFiles": 1,
                "TotalSizeOfScannedFilesInMB": 0.001798,
                "TotalNumberOfImportedFiles": 1,
                "TotalNumberOfResourcesScanned": 1,
                "TotalNumberOfResourcesImported": 1,
                "TotalNumberOfResourcesWithCustomerError": 0,
                "TotalNumberOfFilesReadWithCustomerError": 0,
                "Throughput": 0.0
            },
            "DataAccessRoleArn": "arn:aws:iam::(AWS Account ID):role/(Role Name)"
        }
    ]
}
```

- Per i dettagli sull'API, consulta List FHIRImport Jobs in AWS CLI Command Reference.

Python

SDK per 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
```

- Per i dettagli sull'API, consulta List FHIRImport Jobs in AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

SAP ABAP

SDK per SAP ABAP

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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.
```

- Per i dettagli sull'API, consulta il riferimento all'API List FHIRImport Jobs in AWS SDK for SAP ABAP.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo HealthLake con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`ListTagsForResource`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `ListTagsForResource`.

CLI

AWS CLI

Come elencare i tag di un datastore

L'esempio `list-tags-for-resource` seguente elenca i tag associati al datastore specificato:

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

Output:

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

```
    }
}
```

- Per i dettagli sull'API, consulta ListTagsForResource AWS CLICommand Reference.

Python

SDK per 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
```

- Per i dettagli sull'API, consulta ListTagsForResource AWSSDK for Python (Boto3) API Reference.

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

SAP ABAP

SDK per SAP ABAP

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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.
```

- Per i dettagli sulle API, ListTagsForResourceconsulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo HealthLake con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`StartFHIRExportJob`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `StartFHIRExportJob`.

CLI

AWS CLI

Come avviare un processo di esportazione FHIR

L'`start-fhir-export-job`esempio seguente mostra come avviare un processo di esportazione FHIR utilizzando. AWS HealthLake

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

Output:

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

- Per i dettagli sull'API, consulta Start FHIRExport Job in AWS CLI Command Reference.

Python

SDK per 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
```

- Per i dettagli sull'API, consulta Start FHIRExport Job in AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

SAP ABAP

SDK per SAP ABAP

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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.
```

- Per i dettagli sull'API, consulta Start FHIRExport Job in AWS SDK per il riferimento all'API SAP ABAP.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo HealthLake con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`StartFHIRImportJob`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `StartFHIRImportJob`.

CLI

AWS CLI

Come avviare un processo di importazione FHIR

L'`start-fhir-import-job`esempio seguente mostra come avviare un processo di importazione FHIR utilizzando. AWS HealthLake

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

Output:

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

- Per i dettagli sull'API, consulta Start FHIRImport Job in AWS CLI Command Reference.

Python

SDK per 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
```

- Per i dettagli sull'API, consulta Start FHIRImport Job in AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

SAP ABAP

SDK per SAP ABAP

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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.
```

- Per i dettagli sull'API, consulta Start FHIRImport Job in AWS SDK per il riferimento all'API SAP ABAP.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo HealthLake con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`TagResource`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `TagResource`.

CLI

AWS CLI

Come aggiungere un tag al datastore

L'esempio `tag-resource` seguente mostra come aggiungere un tag al datastore.

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

Questo comando non produce alcun output.

.

- Per i dettagli sull'API, consulta TagResource AWS CLICommand Reference.

Python

SDK per 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
```

- Per i dettagli sull'API, consulta TagResource AWSSDK for Python (Boto3) API Reference.

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

SAP ABAP

SDK per SAP ABAP

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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.
```

- Per i dettagli sulle API, TagResourceconsulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo HealthLake con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`UntagResource`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `UntagResource`.

CLI

AWS CLI

Come rimuovere i tag da un datastore.

L'esempio `untag-resource` seguente mostra come rimuovere i tag da un datastore.

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

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta UntagResource AWS CLICommand Reference.

Python

SDK per 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
```

- Per i dettagli sull'API, consulta UntagResource AWSSDK for Python (Boto3) API Reference.

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

SAP ABAP

SDK per SAP ABAP

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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.
```

- Per i dettagli sulle API, UntagResourceconsulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo HealthLake con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Integrazione AWS HealthLake

I seguenti AWS servizi si integrano direttamente con AWS HealthLake per consentire l'elaborazione integrata del linguaggio naturale, le query SQL e il data warehousing.

- Amazon Comprehend Medical è un servizio di elaborazione del linguaggio naturale (NLP) idoneo alla normativa HIPAA che utilizza librerie di apprendimento automatico per estrarre dati sanitari significativi da testi medici non strutturati. HealthLake Per ulteriori informazioni, consulta la Amazon Comprehend Medical Developer Guide.
- Amazon Athena è un servizio di query interattivo che consente di analizzare HealthLake i dati direttamente nei bucket Amazon Simple Storage Service (Amazon S3) utilizzando SQL standard. Per ulteriori informazioni, consulta la Amazon Athena Developer Guide.

Argomenti

- Elaborazione integrata del linguaggio naturale (NLP) per HealthLake
- Interrogazione di HealthLake dati con Amazon Athena

Elaborazione integrata del linguaggio naturale (NLP) per HealthLake

AWS HealthLake fornisce librerie integrate di elaborazione del linguaggio naturale (NLP) per l'analisi, l'identificazione e la mappatura di dati non strutturati archiviati nei tipi di risorse FHIR. `DocumentReference`

> **Importante**
>
> La PNL integrata per è disattivata per impostazione predefinita. HealthLake Per attivarlo, invia una richiesta di assistenza utilizzando. AWS Support Center Console Per creare il tuo caso, accedi al tuo Account AWS e scegli Crea caso.

HealthLake la PNL integrata funziona richiamando le azioni Amazon Comprehend `DetectEntities-V2` Medical e `InferICD10-CM` API. `InferRxNorm` Le azioni aggiungono i risultati come estensione alla risorsa. `DocumentReference` Quando le azioni dell'API Amazon Comprehend Medical rilevano tratti che `SIGN` sono `SYMPTOM` `DIAGNOSIS` e/o generano una

`Linkage`risorsa FHIR. Le nuove `Condition` `Observation` risorse vengono create da entità identificate con le caratteristiche di `SIGN` o `DIAGNOSIS` e sono collegate al documento di origine utilizzando la risorsa. `SYMPTOM` `Linkage`

> **Note**
>
> Sebbene `GET` le richieste siano supportate per le risorse FHIR generate dalla PNL HealthLake integrata, la funzionalità dell'API FHIR non lo è. `search` Per ulteriori informazioni sulla ricerca di estensioni NLP utilizzando l' HealthLakeintegrazione con Athena, consulta. Indice e interrogazione SQL

Indice

- HealthLake Librerie NLP integrate
- Utilizzo delle interazioni dell'API REST FHIR
- Parametri di ricerca per la PNL integrata HealthLake
- HealthLake richieste di esempio di PNL integrate

HealthLake Librerie NLP integrate

HealthLake deduce i dati trovati nel tipo di `DocumentReference` risorsa utilizzando le librerie Amazon Comprehend Medical. L'API Amazon Comprehend Medical `DetectEntities-V2` opera `InferICD10-CM` `InferRxNorm` e rileva le condizioni mediche come caratteristiche. Ogni operazione fornisce informazioni diverse.

> **Supporto linguistico**
>
> Le operazioni dell'API Amazon Comprehend Medical rilevano solo entità mediche nei testi in lingua inglese.

- DetectEntities-V2: ispeziona il testo clinico per una varietà di entità mediche e restituisce informazioni specifiche su di esse, come la categoria dell'entità, l'ubicazione e il punteggio di fiducia.
- Inferimento ICD1 0-CM: rileva le condizioni mediche nella cartella clinica di un paziente come entità e collega tali entità agli identificatori concettuali normalizzati nella base di conoscenza ICD-10-

CM del National Center for Health Statistics del CDC, con l'autorizzazione dell'Organizzazione Mondiale della Sanità (OMS).

- InferRxNorm: rileva i farmaci come entità elencate nella cartella di un paziente e li collega agli identificatori concettuali normalizzati presenti nel database della National Library of Medicine. RxNorm

Le caratteristiche supportate per ogni operazione API sono`SIGN`, e. `SYMPTOM` `DIAGNOSIS` Se vengono rilevate caratteristiche, vengono aggiunte come estensioni conformi a FHIR in diverse posizioni del data store. HealthLake

Posizioni in cui vengono aggiunte le estensioni.

- `DocumentReference`: i risultati delle operazioni dell'API Amazon Comprehend Medical vengono aggiunti come `extension` un unico documento trovato all'interno `DocumentReference` del tipo di risorsa. I risultati dell'estensione sono divisi in due gruppi. Puoi trovarli nei risultati in base ai loro`URL`.
  - `http://healthlake.amazonaws.com/system-generated-resources/`
    - Si tratta di tipi di risorse che sono stati creati o aggiunti da HealthLake.
  - `http://healthlake.amazonaws.com/aws-cm/`
    - Dove l'output non elaborato delle operazioni dell'API Amazon Comprehend Medical viene aggiunto al HealthLake tuo data store.
- `Linkage`: questo tipo di risorsa viene aggiunto o creato come risultato dell'NLP integrato. Una `GET` richiesta su uno specifico `Linkage` restituisce un elenco di risorse collegate. Per identificare se `Linkage` è stato aggiunto un da HealthLake, cerca la coppia `"tag": [{"display": "SYSTEM_GENERATED"}]` chiave-valore aggiunta. Per ulteriori informazioni sulle specifiche FHIR per Linkage, consulta `Linkage`la documentazione FHIR R4.
- Tipi di risorse FHIR generati come risultato delle operazioni di Amazon Comprehend Medical.
  - `Observation`: include i risultati delle `DetectEntities-V2` azioni dell'API Amazon Comprehend Medical `InferICD10-CM` e quando le caratteristiche `SIGN` sono o. `SYMPTOM`
  - `Condition`: include i risultati delle `DetectEntities-V2` azioni dell'API Amazon Comprehend Medical `InferICD10-CM` e quando la caratteristica `DIAGNOSIS` è.
  - `MedicationStatement`: include i risultati delle azioni dell'API `InferRxNorm` Amazon Comprehend Medical.

Utilizzo delle interazioni dell'API REST FHIR

Per impostazione predefinita, le caratteristiche rilevate dalle operazioni dell'API Amazon Comprehend Medical non vengono restituite quando si effettua una richiesta. `GET` Per visualizzare i risultati delle operazioni NLP integrate, è necessario specificare una risorsa FHIR nota `ID` per i seguenti tipi di risorse.

- `Linkage`
- `Observation`
- `Condition`
- `MedicationStatement`

I risultati delle azioni NLP HealthLake integrate al di fuori del tipo di `DocumentReference` risorsa sono disponibili utilizzando una `GET` richiesta in cui `ID` è noto che lo specificato contiene risultati delle operazioni dell'API Amazon Comprehend Medical.

Parametri di ricerca per la PNL integrata HealthLake

La tabella seguente elenca gli attributi ricercabili per la HealthLake PNL integrata.

Parametri di ricerca per la PNL HealthLake

Parametri di ricerca	Trova corrispondenze per
Rileva E ntities-entity-category	Categoria di entità all'interno della DetectEntities sottoestensione all'interno dell'estensione CM AWS
Rileva E ntities-entity-text	Testo dell'entità all'interno della DetectEntities sottoestensione all'interno dell'estensione CM AWS
Rileva E ntities-entity-type	Tipo di entità all'interno della DetectEntities sottoeste nsione all'interno dell'estensione CM AWS
Rilevato E ntities-entity-score	Entity Score all'interno della DetectEntities sottoeste nsione all'interno dell'estensione CM AWS
infer-icd10 cm-entity-text	Testo dell'entità all'interno della sottoestensione Infer ICD1 0CM all'interno dell'estensione CM AWS

Parametri di ricerca	Trova corrispondenze per
infer-icd10 cm-entity-score	Entity Score all'interno della sottoestensione Infer ICD1 0CM all'interno dell'estensione CM AWS
infer-icd10 cm-entity-concept-code	Entity Concept Code all'interno della sottoestensione Infer ICD1 0CM all'interno dell'estensione CM AWS
infer-icd10 cm-entity-concept-description	Descrizione del concetto di entità all'interno della sottoestensione Infer ICD1 0CM all'interno dell'este nsione CM AWS
infer-icd10 cm-entity-concept-score	Entity Concept Score all'interno della sottoestensione Infer ICD1 0CM all'interno dell'estensione CM AWS
infer-rxnorm-entity-score	Entity Score all'interno della InferRxNorm sottoeste nsione all'interno dell'estensione CM AWS
infer-rxnorm-entity-text	Testo dell'entità all'interno della InferRxNorm sottoestensione all'interno dell' AWS estensione CM
infer-rxnorm-entity-concept-codice	Entity Concept Code all'interno della InferRxNorm sottoestensione all'interno dell' AWS estensione CM
infer-rxnorm-entity-concept-descrizione	Descrizione del concetto di entità all'interno della InferRxNorm sottoestensione all'interno dell' AWS estensione CM
infer-rxnorm-entity-concept-punteggio	Entity Concept Score all'interno della InferRxNorm sottoestensione all'interno dell' AWS estensione CM

HealthLake fornisce una ricerca speciale per soddisfare i criteri in cui `EntityText` e `EntityCategory` fanno parte della stessa entità. La tabella seguente descrive i parametri di ricerca speciali supportati da HealthLake.

Parametri di ricerca

Parametri di ricerca	Partite restituite
Rilevato E ntities-entity-text-category	Se è presente almeno un'entità nella DetectEntities sottoestensione che corrisponde sia a EntityText che a EntityCategory.
Rilevato ntities-entity-type-score	Se è presente almeno un'entità nella DetectEntities sottoestensione che corrisponde sia a EntityType che a EntityScore.
Rilevato ntities-entity-text-score	Se è presente almeno un'entità nella DetectEntities sottoestensione che corrisponde sia a EntityText che a EntityScore.
Rilevato ntities-entity-text-type	Se è presente almeno un'entità nella DetectEntities sottoestensione che corrisponde sia a EntityText che a EntityType.
Rilevato ntities-entity-category-score	Se esiste almeno un'entità che corrisponde sia a EntityCategory che a EntityScore.
codice infer-icd10 cm-entity-text-concept	Se c'è almeno un'entità nella sottoestensione Infer ICD1 0CM che corrisponde a EntityText e c'è almeno un ConceptCode per quell'entità che corrisponde al codice.
infer-icd10 -score cm-entity-text-concept	Se c'è almeno un'entità nella sottoestensione Infer ICD1 0CM che corrisponde a EntityText e c'è almeno un ConceptScore per quell'entità che corrisponde al punteggio.
infer-icd10 -concept-score cm-entity-concept-description	Se c'è almeno un concetto all'interno dell'entità nella sottoestensione Infer ICD1 0CM che corrisponde alla descrizione del concetto e al ConceptScore.
infer-rxnorm-entity-text-codice concettua le	Se c'è almeno un'entità nella InferRxNorm sottoeste nsione che corrisponde a EntityText e c'è almeno

Parametri di ricerca	Partite restituite
	un ConceptCode per quell'entità che corrisponde al codice.
infer-rxnorm-entity-text-concept-score	Se c'è almeno un'entità nella InferRxNorm sottoestensione che corrisponde a EntityText e c'è almeno un ConceptScore per quell'entità che corrisponde al punteggio.
infer-rxnorm-entity-concept-description-concept-score	Se c'è almeno un concetto all'interno dell'entità nella InferRxNorm sottoestensione che corrisponde alla descrizione del concetto e al ConceptScore.

HealthLake richieste di esempio di PNL integrate

Esempio 1: **`Patient`** record inserito in un HealthLake archivio dati

Di seguito è riportato un esempio di nota clinica basata su un `Patient` incontro con un operatore sanitario.

> **Dati sintetici**
>
> Il testo dell'esempio seguente è un contenuto sintetico e non contiene informazioni sanitarie protette (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)
```

Come promemoria, queste informazioni sono codificate in formato base64 nella risorsa. `DocumentReference` Quando questo documento viene inserito HealthLake e le operazioni dell'API Amazon Comprehend Medical sono complete, per vedere i risultati, puoi iniziare con `GET` la richiesta sul `DocumentReference` tipo di risorsa.

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

Quando le operazioni dell'API Amazon Comprehend Medical hanno esito positivo, cerca queste coppie chiave-valore all'interno `extension` del link seguente `"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."
 }
```

Le seguenti schede mostrano come la cartella clinica ingerita viene riportata nel tuo archivio HealthLake dati in base al tipo di risorsa.

DocumentReference

Per visualizzare i risultati per un singolo tipo di `DocumentReference` risorsa, effettua una `GET` richiesta in cui viene fornita `id` una risorsa specifica.

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

In caso di successo, si ottiene un codice di risposta `200` HTTP e la seguente risposta JSON (che è stata troncata per maggiore chiarezza).

Ecco la parte. `http://healthlake.amazonaws.com/system-generated-resources/` Puoi vedere che ne `Linkage/e366d29f-2c22-4c19-866e-09603937935a` è stata aggiunta una nuova. Puoi anche vedere dove sono HealthLake stati aggiunti risultati basati sull'inferenza a tipi di `Condition` risorse specifici`Observation`.

Per vedere come sono stati modificati questi tipi di risorse, scegli le schede correlate.

```
{
  "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

Per visualizzare i risultati per un singolo tipo di `Linkage` risorsa, effettua una `GET` richiesta in cui `ID` viene fornita una risorsa specifica.

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

In caso di successo, si ottiene un codice di risposta `200` HTTP e la seguente risposta JSON troncata.

La risposta contiene l'elemento. `item` In essa, la coppia chiave-valore `"type": "source"` indica la `DocumentReference` voce specifica utilizzata per modificare la coppia chiave-valore `Condition` ed è `Observations` elencata sotto la coppia `"type": "alternate"` chiave-valore.

Sono inoltre visibili l'*`meta`*elemento e la corrispondente coppia chiave-valore, che indica che queste risorse sono state *`"tag": [{"display": "SYSTEM_GENERATED"}]`* create da. 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

Per visualizzare i risultati per un singolo tipo di `Observation` risorsa, effettua una `GET` richiesta in cui viene `ID` fornita una risorsa specifica.

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

I risultati delle operazioni dell'API Amazon Comprehend Medical vengono modificati in base ai seguenti elementi`code`:`meta`,, `modifierExtension` e.

`code`

Un elemento di tipo`CodeableConcept`. Per ulteriori informazioni, consultate la [`CodeableConcept`](#)documentazione di FHIR R4.

HealthLake aggiunge le seguenti tre coppie chiave-valore.

- `"system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/"`: dove l'URL si riferisce a una specifica operazione dell'API Amazon Comprehend Medical. In questo caso, Infer 0CM. ICD1
- `"code": "A52.06"`: `A52.06` Dov'è il codice ICD-10-CM che identifica il concetto trovato nella knowledge base dei Centers for Disease Control.
- `"display": "Other syphilitic heart involvement"`: `"Other syphilitic heart involvement"` Dov'è la descrizione estesa del codice ICD-10-CM nell'ontologia.

La seguente risposta JSON troncata contiene solo l'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"
 }
```

Per capire se il modello ritiene che il codice ICD-10-CM assegnato sia corretto, utilizzate l'elemento. `modifierExtension`

`meta`

L'`meta`elemento contiene metadati che indicano se l'`code`elemento contiene dettagli che sono stati aggiunti dalle operazioni dell'API Amazon Comprehend Medical.

La seguente risposta JSON troncata contiene solo l'elemento. `meta`

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

```
 }
```

modifierExtension

L'`modifierExtension`elemento contiene ulteriori dettagli sul livello di confidenza dei codici assegnati presenti nell'elemento. `code` Dispone inoltre di coppie chiave-valore che forniscono un collegamento all'originale DocumentReference utilizzato per generare i risultati e al tipo di risorsa Linkage correlato.

Per ogni `coding` elemento aggiunto, vedrai un `entity-score` e un `entity-Concept-Score` aggiunto a ModifierExtension. Per ogni valore nella coppia chiave-valore, viene visualizzato un punteggio. Infatti`entity-score`, questo punteggio è il livello di fiducia che Amazon Comprehend Medical ha nell'accuratezza del rilevamento. Infatti`entity-Concept-Score`, questo punteggio è il livello di fiducia di Amazon Comprehend Medical nel fatto che l'entità sia accuratamente collegata a un concetto ICD-10-CM.

La seguente risposta JSON troncata contiene solo l'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"
   }
  }
 ]
```

Risposta 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

Per visualizzare i risultati per un singolo tipo di `Condition` risorsa, effettua una `GET` richiesta in cui viene fornito il contenuto `ID` di una risorsa specifica.

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

I risultati delle operazioni dell'API Amazon Comprehend Medical vengono modificati in base ai seguenti elementicode:`meta`,, `modifierExtension` e.

`code`

Un elemento di tipo`CodeableConcept`. Per ulteriori informazioni, consultate la `CodeableConcept`documentazione di FHIR R4.

HealthLake aggiunge le seguenti tre coppie chiave-valore.

- `"system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/"`: dove l'URL si riferisce a una specifica operazione dell'API Amazon Comprehend Medical. In questo caso, Infer 0CM. ICD1
- `"code": "I70.0"`: `A52.06` Dov'è il codice ICD-10-CM che identifica il concetto trovato nella knowledge base dei Centers for Disease Control.
- `"display": "Atherosclerosis of aorta"`: `"0ther syphilitic heart involvement"` Dov'è la descrizione estesa del codice ICD-10-CM nell'ontologia.

La seguente risposta JSON troncata contiene solo l'elemento. `code`

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

```
    }
  ],
  "text": "Atherosclerosis of aorta"
 }
```

Per capire se il modello ritiene che il codice ICD-10-CM assegnato sia corretto, utilizzate l'elemento. `modifierExtension`

`meta`

L'`meta`elemento contiene metadati che indicano se l'`code`elemento contiene dettagli che sono stati aggiunti dalle operazioni dell'API Amazon Comprehend Medical.

La seguente risposta JSON troncata contiene solo l'elemento. `meta`

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

`modifierExtension`

L'`modifierExtension`elemento contiene ulteriori dettagli sul livello di confidenza dei codici assegnati presenti nell'elemento. `code` Dispone inoltre di coppie chiave-valore che forniscono un collegamento all'originale DocumentReference utilizzato per generare i risultati e al tipo di risorsa Linkage correlato.

Per ogni `coding` elemento aggiunto, vedrai un `entity-score` e un `entity-Concept-Score` aggiunto a ModifierExtension. Per ogni valore nella coppia chiave-valore, viene visualizzato un punteggio. Infatti`entity-score`, questo punteggio è il livello di fiducia che Amazon Comprehend Medical ha nell'accuratezza del rilevamento. Infatti`entity-Concept-Score`, questo punteggio è il livello di fiducia di Amazon Comprehend Medical nel fatto che l'entità sia accuratamente collegata a un concetto ICD-10-CM.

La seguente risposta JSON troncata contiene solo l'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"
   }
  }
 ]
```

Risposta 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"
 }
```

Esempio 2: A **`DocumentReference`** che contiene un tipo di **`MedicationStatement`** risorsa

Ecco un esempio di nota clinica basata sull'incontro di un paziente con un medico.

> ⚠ Dati sintetici
>
> Il testo in questo esempio è un contenuto sintetico e non contiene informazioni sanitarie protette (PHI).

```
Tom is not prescribed Advil
```

Le schede seguenti mostrano come la cartella clinica ingerita viene riportata nell'archivio HealthLake dati in base al tipo di risorsa.

DocumentReference

Per visualizzare i risultati per un singolo tipo di `DocumentReference` risorsa, effettua una `GET` richiesta in cui viene fornita `ID` una risorsa specifica.

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

In caso di successo, si ottiene un codice di risposta `200` HTTP e la seguente risposta JSON troncata.

La coppia chiave-valore indica che `"url": "http://healthlake.amazonaws.com/system-generated-resources/"` i tipi di risorse al suo interno `extension` sono stati aggiunti dalle operazioni dell'API Amazon Comprehend Medical. Puoi vedere il nuovo tipo di `Linkage` risorsa e più risorse. `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

Per visualizzare i risultati per un singolo tipo di `Linkage` risorsa, effettua una `GET` richiesta in cui viene fornita una risorsa specifica. `ID`

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

In caso di successo, si ottiene un codice di risposta `200` HTTP e la seguente risposta JSON.

La risposta contiene l'`item`elemento. In essa, la coppia chiave-valore `"type": "source"` indica la `DocumentReference` voce specifica utilizzata per modificare i tipi di `MedicationStatement` risorse.

È inoltre possibile visualizzare l'`meta`elemento e la corrispondente coppia chiave-valore`"tag": [{"display": "SYSTEM_GENERATED"}]`, a indicare che queste risorse sono state create da. 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

Per visualizzare i risultati per un singolo tipo di `MedicationStatement` risorsa, effettua una `GET` richiesta in cui viene `ID` fornita una risorsa specifica.

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

Il tipo di `MedicationStatement` risorsa è dove si trovano i risultati dell'operazione dell'API Amazon Comprehend InferRxNorm Medical. I risultati vengono modificati in base ai seguenti elementi: `medicationCodeableConceptmeta`, e`modifierExtension`.

medicationCodeableConcept

Un elemento di tipo`CodeableConcept`. Per ulteriori informazioni, consultate la `CodeableConcept`documentazione di FHIR R4.

HealthLake aggiunge le seguenti tre coppie chiave-valore.

- `"system": ""http://healthlake.amazonaws.com/aws-cm/infer-rxnorm/`: dove l'URL si riferisce a una specifica operazione dell'API Amazon Comprehend Medical. In questo caso, InferRxNorm.
- `"code": "731533"`: `731533` Dov'è un ID RxNorm concettuale, noto anche come RxCUI.
- `"display": "ibuprofen 200 MG Oral Capsule [Advil]"`: `ibuprofen 200 MG Oral Capsule [Advil]` Dov'è la descrizione del RxNorm concetto.

La seguente risposta JSON troncata contiene solo l'elemento. `MedicationStatement`

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

```
  ]
 }
```

meta

L'`meta`elemento contiene metadati che indicano se l'`code`elemento contiene dettagli che sono stati aggiunti dalle operazioni dell'API Amazon Comprehend Medical.

La seguente risposta JSON troncata contiene solo l'elemento. `meta`

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

modifierExtension

L'`modifierExtension`elemento contiene coppie chiave-valore che forniscono un collegamento all'originale `DocumentReference` utilizzato per generare i risultati e al tipo di risorsa Linkage correlato.

```
"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"
    }
   }
 ]
```

Interrogazione di HealthLake dati con Amazon Athena

Durante un processo di HealthLake importazione, i dati JSON FHIR annidati vengono sottoposti a un processo ETL e vengono archiviati nel formato di tabella aperta Apache Iceberg, in cui ogni tipo di risorsa FHIR è rappresentato come una tabella individuale in Athena. Ciò consente agli utenti di interrogare i dati FHIR utilizzando SQL, ma senza doverli prima esportare. Questo è prezioso, in quanto consente a medici e scienziati di interrogare i dati FHIR per convalidare le loro decisioni o far progredire la loro ricerca. Per ulteriori informazioni sul funzionamento delle tabelle Apache Iceberg in Athena, consulta Interrogare le tabelle Apache Iceberg nella Guida per l'utente di Athena.

> **Note**
>
> HealthLake supporta l'`read`interazione FHIR R4 sui HealthLake dati in Athena. Per ulteriori informazioni, consulta Leggere una risorsa FHIR.

Gli argomenti di questa sezione descrivono come connettere il tuo HealthLake data store ad Athena, come interrogarlo utilizzando SQL e come connettere i risultati con altri AWS servizi per ulteriori analisi.

Argomenti

- Guida introduttiva ad Amazon Athena
- Interrogazione HealthLake dei dati con SQL
- Esempi di query SQL con filtri complessi

Guida introduttiva ad Amazon Athena

Per l'integrazione HealthLake con Amazon Athena, devi configurare le autorizzazioni. A tale scopo, creerai un utente, un gruppo o un ruolo Athena e concederai loro l'accesso alle risorse FHIR situate all'interno di un HealthLake archivio dati.

- Concedere a un utente, un gruppo o un ruolo l'accesso a un HealthLake data store (AWS Lake Formation Console)
- Configurazione di un account Athena

Concedere a un utente, un gruppo o un ruolo l'accesso a un HealthLake data store (AWS Lake Formation Console)

> Persona: amministratore HealthLake
>
> L' HealthLake amministratore è un amministratore del data lake di AWS Lake Formation. Consentono l'accesso agli archivi di HealthLake dati in Lake Formation.

Per ogni data store creato, ci sono due voci visibili nella console di AWS Lake Formation. Una voce è un collegamento a una risorsa. I nomi dei link alle risorse sono sempre visualizzati in corsivo. Ogni collegamento alla risorsa viene visualizzato con il nome e il proprietario della relativa risorsa condivisa collegata. Per tutti gli archivi HealthLake dati, il proprietario della risorsa condivisa è l'account HealthLake di servizio. L'altra voce è l'archivio HealthLake dati nell'account del HealthLake servizio. I passaggi di questa procedura utilizzano l'archivio dati che è il collegamento alla risorsa.

Per ulteriori informazioni sui collegamenti alle risorse, consulta Come funzionano i collegamenti alle risorse in Lake Formation nella AWS Lake Formation Developer Guide.

Affinché un utente, un gruppo o un ruolo possa interrogare i dati in Athena, è necessario concedere l'autorizzazione Descrivi sul database delle risorse. Quindi, è necessario concedere Select e Descrivi sulle tabelle.

FASE 1: Concedere le autorizzazioni DESCRIBE su un database di link alle risorse del HealthLake Data Store

1. Apri la console AWS Lake Formation: https://console.aws.amazon.com/lakeformation/
2. Nella barra di navigazione principale, scegli Database.
3. Nella pagina Database, scegli il pulsante di opzione accanto al nome del data store in corsivo.
4. Scegli Azioni (▾).
5. Scegliere Concedi.
6. Nella pagina Concedi le autorizzazioni ai dati, in Principali, scegli utenti o ruoli IAM.
7. In Utenti o ruoli IAM, utilizza la freccia rivolta verso il basso (▾) o cerca l'utente, il ruolo o il gruppo IAM su cui desideri poter effettuare interrogazioni in Athena.
8. In LF-Tags o nella scheda delle risorse del catalogo, scegli l'opzione Named data catalog resources.

9. In Database, utilizzate la freccia rivolta verso il basso (▼) per scegliere il database del HealthLake data store a cui desiderate condividere l'accesso.
10. Nella scheda Autorizzazioni Resource link, in Autorizzazioni Resource link, scegli Descrivi.

Quando la concessione ha esito positivo, viene visualizzato il banner Concedi autorizzazione con successo. Per visualizzare l'autorizzazione appena concessa, scegli Autorizzazioni Data lake. Trova l'utente, il gruppo e il ruolo nella tabella. Nella colonna Autorizzazioni, vedrai l'elenco Descrivi.

Ora devi usare Grant on target per concedere Select e Descrivi su tutte le tabelle del database.

FASE 2: Concedere l'accesso a tutte le tabelle in un collegamento alle risorse del HealthLake data store

1. Apri la console AWS Lake Formation: https://console.aws.amazon.com/lakeformation/
2. Nella barra di navigazione principale, scegli Database.
3. Nella pagina Database, scegli il pulsante di opzione accanto al nome del data store in corsivo.
4. Scegli Azioni (▼).
5. Scegli Grant on target.
6. Nella pagina Concedi le autorizzazioni ai dati, in Principali, scegli utenti o ruoli IAM.
7. In Utenti o ruoli IAM, utilizza la freccia rivolta verso il basso (▼) o cerca l'utente, il gruppo o il ruolo IAM su cui desideri poter effettuare interrogazioni in Athena.
8. In LF-Tags o nella scheda delle risorse del catalogo, scegli l'opzione Named data catalog resources.
9. In Database, utilizzate la freccia rivolta verso il basso (▼) per scegliere il database del HealthLake data store a cui desiderate concedere l'accesso.
10. In Tabelle, scegli Tutte le tabelle per condividere tutte le tabelle con un HealthLake utente.
11. Nella scheda Autorizzazioni della tabella, in Autorizzazioni della tabella, scegli Descrivi e seleziona.
12. Scegliere Concedi.

Dopo aver scelto Concedi, viene visualizzato il banner Concedi autorizzazioni di successo. L'utente specificato può ora effettuare interrogazioni su un archivio HealthLake dati in Athena.

Guida introduttiva ad Athena

> **HealthLake utente**
>
> L' HealthLake utente utilizzerà la console Athena o AWS SDKs interrogherà un HealthLake data store condiviso con lui dall' HealthLake amministratore. AWS CLI

Per interrogare un data store utilizzando Athena, è necessario eseguire le tre operazioni seguenti.

- Concedi all'utente o al ruolo IAM l'accesso al HealthLake data store tramite Lake Formation. Per ulteriori informazioni, consulta Concedere a un utente, un gruppo o un ruolo l'accesso a un HealthLake data store (AWS Lake Formation Console).
- Crea un gruppo di lavoro per il tuo HealthLake data store.
- Imposta un bucket Amazon S3 per archiviare i risultati delle query.

Per iniziare a usare Athena, aggiungi le AmazonAthenaFullAccesspolicy FullAccess AWS gestite da AmazonS3 al tuo utente, gruppo o ruolo. L'utilizzo di una politica AWS gestita è un ottimo modo per iniziare a utilizzare un nuovo servizio. Tieni presente che le policy gestite da AWS potrebbero non concedere autorizzazioni con privilegi minimi per i tuoi casi d'uso specifici perché sono disponibili per l'uso da parte di tutti i clienti AWS. Quando imposti le autorizzazioni con le policy IAM, concedi solo le autorizzazioni richieste per eseguire un'attività. Per ulteriori informazioni su IAM e sull'applicazione dei privilegi minimi, consulta Applica le autorizzazioni con privilegi minimi nella IAM User Guide.

> **Important**
>
> Per interrogare un archivio HealthLake dati in Athena, è necessario utilizzare la versione 3 del motore Athena.

I gruppi di lavoro sono risorse e pertanto è possibile utilizzare policy basate su IAM per controllare l'accesso a gruppi di lavoro specifici. Per ulteriori informazioni, consulta Utilizzo dei gruppi di lavoro per controllare l'accesso alle query e i costi nella Guida per l'utente di Athena.

Per ulteriori informazioni sulla configurazione dei gruppi di lavoro, consulta https://docs.aws.amazon.com/athena/latest/ug/workgroups-procedure.html la Guida per l'utente di Athena.

> **Note**
>
> La regione in cui si trova il bucket Amazon S3 e la console Athena devono corrispondere.

Prima di poter eseguire una query, è necessario specificare una posizione del bucket dei risultati delle query in Amazon S3, o utilizzare un gruppo di lavoro che ha specificato un bucket e la cui configurazione sostituisce le impostazioni del client. I file di output vengono salvati automaticamente per ogni query eseguita.

Per ulteriori dettagli sulla specificazione delle posizioni dei risultati delle query nella console Athena, consulta Specificare una posizione dei risultati delle query utilizzando la console Athena nella Amazon Athena User Guide.

Per vedere esempi di come interrogare il tuo archivio HealthLake dati in Athena, consulta. Interrogazione HealthLake dei dati con SQL

Interrogazione HealthLake dei dati con SQL

Quando importi i dati FHIR nell'archivio dati, HealthLake i dati JSON FHIR annidati vengono contemporaneamente sottoposti a un processo ETL e vengono archiviati in formato tabella aperta Apache Iceberg in Amazon S3. Ogni tipo di risorsa FHIR del tuo HealthLake data store viene convertito in una tabella, dove può essere interrogato utilizzando Amazon Athena. Le tabelle possono essere interrogate singolarmente o in gruppo utilizzando query basate su SQL. A causa della struttura degli archivi dati, i dati vengono importati in Athena come diversi tipi di dati. Per ulteriori informazioni sulla creazione di query SQL in grado di accedere a questi tipi di dati, consulta Matrici di query con tipi complessi e strutture annidate nella Amazon Athena User Guide.

> **Note**
>
> Tutti gli esempi in questo argomento utilizzano dati fittizi creati con Synthea. Per ulteriori informazioni sulla creazione di un data store precaricato con dati Synthea, consulta. Creazione di un archivio HealthLake dati

Per ogni elemento di un tipo di risorsa, la specifica FHIR definisce una cardinalità. La cardinalità di un elemento definisce i limiti inferiore e superiore di quante volte questo elemento può apparire. Quando si costruisce una query SQL, è necessario tenerne conto. Ad esempio, diamo un'occhiata ad alcuni elementi in Tipo di risorsa: Paziente.

- Elemento: Nome La specifica FHIR imposta la cardinalità come. `0..*`

  L'elemento viene acquisito come matrice.

```
[{
 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
}]
```

  In Athena, per vedere come è stato importato un tipo di risorsa, cercalo in Tabelle e viste. Per accedere agli elementi di questo array, puoi usare la notazione a punti. Ecco un semplice esempio che consentirebbe di accedere ai valori di `given` e`family`.

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

- Elemento: MaritalStatus La specifica FHIR imposta la cardinalità come. `0..1`

  Questo elemento viene acquisito come 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
}
```

In Athena, per vedere come è stato importato un tipo di risorsa, cercalo in Tabelle e viste. Per accedere alle coppie chiave-valore in JSON, puoi usare la notazione a punti. Poiché non è un array, non è richiesto alcun indice di matrice. Ecco un semplice esempio che consentirebbe di accedere al valore di`text`.

```
SELECT
    maritalstatus.text as MaritalStatus
FROM Patient
```

Per ulteriori informazioni sull'accesso e sulla ricerca in JSON, consulta Querying JSON nella Athena User Guide.

Le istruzioni di interrogazione DML (Athena Data Manipulation Language) sono basate su Trino. Athena non supporta tutte le funzionalità di Trino e presenta differenze significative. Per ulteriori informazioni, consulta le interrogazioni, le funzioni e gli operatori DML nella Guida per l'utente di Amazon Athena.

Inoltre, Athena supporta diversi tipi di dati che potresti incontrare durante la creazione di query sul tuo HealthLake archivio dati. Per ulteriori informazioni sui tipi di dati in Athena, consulta Tipi di dati in Amazon Athena nella Amazon Athena User Guide.

Per ulteriori informazioni su come funzionano le query SQL in Athena, consulta il riferimento SQL per Amazon Athena nella Amazon Athena User Guide.

Ogni scheda mostra esempi di come effettuare ricerche sui tipi di risorse specificati e sugli elementi associati utilizzando Athena.

Element: Extension

L'elemento `extension` viene utilizzato per creare campi personalizzati in un archivio dati.

Questo esempio mostra come accedere alle funzionalità dell'`extension`elemento presente nel tipo di `Patient` risorsa.

Quando l'archivio HealthLake dati viene importato in Athena, gli elementi di un tipo di risorsa vengono analizzati in modo diverso. Poiché la struttura della variabile `element` is, non può essere specificata completamente nello schema. Per gestire tale variabilità, gli elementi all'interno dell'array vengono passati come stringhe.

Nella descrizione della tabella di`Patient`, è possibile vedere l'elemento `extension` descritto come`array<string>`, il che significa che è possibile accedere agli elementi dell'array utilizzando un valore di indice. Per accedere agli elementi della stringa, tuttavia, è necessario utilizzare`json_extract`.

Ecco una singola voce dell'`extension`elemento che si trova nella tabella dei pazienti.

```
[{
  "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"
 }
```

```
]
```

Anche se si tratta di un JSON valido, Athena lo considera come una stringa.

Questo esempio di query SQL dimostra come creare una tabella che contenga gli elementi and. `patient-mothersMaidenName` `patient-birthPlace` Per accedere a questi elementi, è necessario utilizzare diversi indici di matrice e `json_extract.`

```
SELECT
    extension[1],
    json_extract(extension[1], '$.valueString') AS MothersMaidenName,
    extension[2],
    json_extract(extension[2], '$.valueAddress.city') AS birthPlace
FROM patient
```

Per ulteriori informazioni sulle query che coinvolgono JSON, consulta Estrazione di dati da JSON nella Amazon Athena User Guide.

Element: birthDate (Age)

L'età non è un elemento del tipo di risorsa per i pazienti in FHIR. Ecco due esempi di ricerche che filtrano in base all'età.

Poiché l'età non è un elemento, utilizziamo il `birthDate` per le query SQL. Per vedere come un elemento è stato inserito in FHIR, cercate il nome della tabella in Tabelle e viste. Puoi vedere che è di tipo stringa.

Esempio 1: calcolo di un valore per l'età

In questa query SQL di esempio, utilizziamo uno strumento SQL integrato `year` per estrarre tali componenti. `current_date` Quindi, li sottraiamo per restituire l'età effettiva del paziente, sotto forma di colonna denominata`age`.

```
SELECT
 (year(current_date) - year(date(birthdate))) as age
FROM patient
```

Esempio 2: Filtraggio per pazienti nati prima `2019-01-01` e che lo sono. `male`

La query SQL mostra come utilizzare la `CAST` funzione per convertire l'`birthDate`elemento come tipo `DATE` e come filtrare in base a due criteri della `WHERE` clausola. Poiché l'elemento viene inserito come stringa di tipo per impostazione predefinita, è necessario che `CAST` sia di

tipo. `DATE` Quindi puoi usare l'<operatore per confrontarlo con una data diversa,. `2019-01-01` UtilizzandoAND, è possibile aggiungere un secondo criterio alla `WHERE` clausola.

```
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

Questo esempio mostra le ricerche di località all'interno del tipo di risorsa Location in cui il nome della città è 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 condizione relativa al tipo di risorsa memorizza i dati di diagnosi relativi a problemi che hanno raggiunto un livello di preoccupazione. HealthLakel'elaborazione medica integrata del linguaggio naturale (NLP) genera nuove `Condition` risorse sulla base dei dettagli presenti nel DocumentReference tipo di risorsa. Quando viene generata una nuova risorsa, HealthLake aggiunge il tag `SYSTEM_GENERATED` all'elemento. `meta` Questa query SQL di esempio dimostra come è possibile cercare nella tabella delle condizioni e restituire risultati laddove i `SYSTEM_GENERATED` risultati sono stati rimossi.

Per ulteriori informazioni sull'elaborazione integrata HealthLake del linguaggio naturale (NLP), vedere. Elaborazione integrata del linguaggio naturale (NLP) per HealthLake

```
SELECT *
FROM condition
WHERE meta.tag[1] is NULL
```

Puoi anche cercare all'interno di un elemento di stringa specificato per filtrare ulteriormente la tua query. L'`modifierextension`elemento contiene dettagli su quale `DocumentReference` risorsa è stata utilizzata per generare un insieme di condizioni. Ancora una volta, è necessario utilizzare `json_extract` per accedere agli elementi JSON annidati che vengono importati in Athena come stringa.

Questa query SQL di esempio dimostra come è possibile cercare tutto ciò `Condition` che è stato generato in base a uno specifico. `DocumentReference` `CAST`Da utilizzare per impostare l'elemento JSON come stringa in modo da poterlo utilizzare `LIKE` per il confronto.

```
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

Il tipo di risorsa Observation memorizza le misurazioni e le semplici asserzioni fatte su un paziente, un dispositivo o un altro argomento. HealthLake(Natural Language Processing, NLP) integrata genera nuove `Observation` risorse sulla base dei dettagli presenti in una risorsa. `DocumentReference` Questa query SQL di esempio include `WHERE meta.tag[1] is NULL` commenti, il che significa che i `SYSTEM_GENERATED` risultati sono inclusi.

```
SELECT valueCodeableConcept.coding[1].code
FROM Observation
WHERE  valueCodeableConcept.coding[1].code = '266919005'
-- WHERE meta.tag[1] is NULL
```

Questa colonna è stata importata come `struct`. Pertanto, è possibile accedere agli elementi al suo interno utilizzando la notazione a punti.

Resource type: MedicationStatement

MedicationStatement è un tipo di risorsa FHIR che è possibile utilizzare per memorizzare informazioni sui farmaci che un paziente ha assunto, sta assumendo o assumerà in futuro. HealthLakel'elaborazione medica integrata del linguaggio naturale (NLP) genera nuove

MedicationStatement risorse sulla base dei documenti presenti nel DocumentReference tipo di risorsa. Quando vengono generate nuove risorse, HealthLake aggiunge il tag `SYSTEM_GENERATED` all'elemento. `meta` Questo esempio di query SQL dimostra come creare una query che filtra in base a un singolo paziente utilizzando il relativo identificatore e trova le risorse che sono state aggiunte dalla HealthLake PNL integrata.

```
SELECT *
FROM medicationstatement
WHERE meta.tag[1].display = 'SYSTEM_GENERATED' AND subject.reference =
 'Patient/0679b7b7-937d-488a-b48d-6315b8e7003b';
```

Per ulteriori informazioni sull'elaborazione integrata HealthLake del linguaggio naturale (NLP), vedere. Elaborazione integrata del linguaggio naturale (NLP) per HealthLake

Esempi di query SQL con filtri complessi

Gli esempi seguenti mostrano come utilizzare le query SQL di Amazon Athena con filtri complessi per localizzare i dati FHIR da un data store. HealthLake

Example Crea criteri di filtraggio basati su dati demografici

L'identificazione dei dati demografici corretti dei pazienti è importante quando si crea una coorte di pazienti. Questa query di esempio dimostra come utilizzare la notazione a punti Trino e `json_extract` filtrare i dati nell'archivio dati. HealthLake

```
SELECT
    id
    , CONCAT(name[1].family, ' ', name[1].given[1]) as name
    , (year(current_date) - year(date(birthdate))) as age
    , gender as gender
    , json_extract(extension[1], '$.valueString') as MothersMaidenName
    , json_extract(extension[2], '$.valueAddress.city') as birthPlace
    , maritalstatus.coding[1].display as maritalstatus
    , address[1].line[1] as addressline
    , address[1].city as city
    , address[1].district as district
    , address[1].state as state
    , address[1].postalcode as postalcode
    , address[1].country as country
    , json_extract(address[1].extension[1], '$.extension[0].valueDecimal') as latitude
```

```
    , json_extract(address[1].extension[1], '$.extension[1].valueDecimal') as longitude
    , telecom[1].value as telNumber
    , deceasedboolean as deceasedIndicator
    , deceaseddatetime
FROM database.patient;
```

Utilizzando la console Athena, puoi ordinare e scaricare ulteriormente i risultati.

Example Crea filtri per un paziente e le sue condizioni correlate

La seguente query di esempio mostra come è possibile trovare e ordinare tutte le condizioni correlate per i pazienti presenti in un archivio HealthLake dati.

```
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;
```

Puoi utilizzare la console Athena per ordinare ulteriormente i risultati o scaricarli per ulteriori analisi.

Example Crea filtri per i pazienti e le relative osservazioni

La seguente query di esempio mostra come trovare e ordinare tutte le osservazioni correlate per i pazienti trovate in un archivio HealthLake dati.

```
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 Crea condizioni di filtraggio per un paziente e le relative procedure

Collegare le procedure ai pazienti è un aspetto importante dell'assistenza sanitaria. La seguente query di esempio SQL dimostra come utilizzare FHIR `Patient` e i tipi di `Procedure` risorse per eseguire questa operazione. La seguente query SQL restituirà tutti i pazienti e le relative procedure presenti nell'archivio dati HealthLake .

```
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;
```

Puoi utilizzare la console Athena per scaricare i risultati per ulteriori analisi o ordinarli per comprenderli meglio.

Example Crea condizioni di filtraggio per un paziente e le relative prescrizioni

È importante consultare un elenco aggiornato dei farmaci che i pazienti assumono. Usando Athena, puoi scrivere una query SQL che utilizza sia i `Patient` tipi di `MedicationRequest` risorse che si trovano nel tuo archivio HealthLake dati.

La seguente query SQL unisce le `MedicationRequest` tabelle `Patient` e importate in Athena. Inoltre, organizza le prescrizioni nelle rispettive voci utilizzando la notazione a punti.

```
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
```

Puoi utilizzare la console Athena per ordinare i risultati o scaricarli per ulteriori analisi.

Example Vedi i farmaci presenti nel tipo di **`MedicationStatement`** risorsa

La seguente query di esempio mostra come organizzare il JSON annidato importato in Athena utilizzando SQL. La query utilizza l'`meta`elemento FHIR per indicare quando è stato aggiunto un farmaco mediante l'elaborazione integrata HealthLake del linguaggio naturale (NLP). Viene inoltre utilizzato `json_extract` per cercare dati all'interno dell'array di stringhe JSON. Per ulteriori informazioni, consulta Elaborazione linguaggio naturale.

```
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;
```

Puoi usare la console Athena per scaricare questi risultati o ordinarli.

Example Filtra per un tipo di malattia specifico

L'esempio mostra come trovare un gruppo di pazienti, di età compresa tra 18 e 75 anni, a cui è stato diagnosticato il diabete.

```
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%');
```

Ora puoi usare la console Athena per ordinare i risultati o scaricarli per ulteriori analisi.

Monitoraggio AWS HealthLake

Il monitoraggio e la registrazione sono parti importanti per mantenere la sicurezza, l'affidabilità, la disponibilità e le prestazioni di AWS HealthLake. AWS fornisce i seguenti servizi per monitorare HealthLake, segnalare quando qualcosa non va e intraprendere azioni automatiche quando necessario.

- AWS CloudTrailacquisisce le chiamate API e gli eventi correlati effettuati da o per conto del tuo AWS account e invia i file di log a un bucket Amazon S3 da te specificato. Puoi identificare quali utenti e account hanno chiamato AWS, l'indirizzo IP di origine da cui sono state effettuate le chiamate e quando sono avvenute le chiamate. Per ulteriori informazioni, consulta la Guida per l'utente AWS CloudTrail.
- Amazon CloudWatch monitora AWS le tue risorse e le applicazioni su cui esegui AWS in tempo reale. È possibile raccogliere e tenere traccia dei parametri, creare pannelli di controllo personalizzati e impostare allarmi per inviare una notifica o intraprendere azioni quando un parametro specificato raggiunge una determinata soglia. Ad esempio, puoi tenere CloudWatch traccia dell'utilizzo della CPU o di altri parametri delle tue EC2 istanze Amazon e avviare automaticamente nuove istanze quando necessario. Per ulteriori informazioni, consulta la Amazon CloudWatch User Guide.
- Amazon EventBridge è un servizio di bus eventi senza server che semplifica la connessione delle applicazioni con dati provenienti da una varietà di fonti. EventBridge fornisce un flusso di dati in tempo reale dalle tue applicazioni, applicazioni Software-as-a-Service (SaaS) e AWS servizi e indirizza tali dati verso destinazioni come Lambda. In questo modo puoi monitorare gli eventi che si verificano nei servizi e creare architetture basate su eventi. Per ulteriori informazioni, consulta la Amazon EventBridge User Guide.

Argomenti

- Registrazione delle chiamate HealthLake API utilizzando AWS CloudTrail
- Monitoraggio delle HealthLake metriche tramite Amazon CloudWatch
- Monitoraggio HealthLake degli eventi tramite Amazon EventBridge

Registrazione delle chiamate HealthLake API utilizzando AWS CloudTrail

AWS HealthLake è integrato con AWS CloudTrail, un servizio che fornisce una registrazione delle azioni intraprese da un utente, ruolo o AWS servizio in HealthLake. CloudTrail acquisisce tutte le chiamate API HealthLake come eventi. Le chiamate acquisite includono chiamate dalla HealthLake console e chiamate di codice alle operazioni HealthLake API. Se crei un trail, puoi abilitare la distribuzione continua di CloudTrail eventi a un bucket Amazon S3, inclusi gli eventi per. HealthLake Se non configuri un percorso, puoi comunque visualizzare gli eventi più recenti nella CloudTrail console nella cronologia degli eventi. Utilizzando le informazioni raccolte da CloudTrail, puoi determinare a quale richiesta è stata inviata HealthLake, l'indirizzo IP da cui è stata effettuata la richiesta, chi ha effettuato la richiesta, quando è stata effettuata e dettagli aggiuntivi.

Per ulteriori informazioni CloudTrail, consulta la Guida AWS CloudTrail per l'utente.

AWS HealthLake Informazioni in CloudTrail

CloudTrail è abilitato sul tuo AWS account al momento della creazione dell'account. Quando si verifica un'attività in HealthLake, tale attività viene registrata in un CloudTrail evento insieme ad altri eventi AWS di servizio nella cronologia degli eventi. È possibile visualizzare, cercare e scaricare gli eventi recenti nell'account AWS . Per ulteriori informazioni, consulta Visualizzazione degli eventi con la cronologia degli CloudTrail eventi.

Per una registrazione continua degli eventi nel tuo AWS account, inclusi gli eventi di HealthLake, crea un percorso. Un trail consente di CloudTrail inviare file di log a un bucket Amazon S3. Per impostazione predefinita, quando si crea un trail nella console, il trail sarà valido in tutte le regioni AWS. Il trail registra gli eventi di tutte le regioni della AWS partizione e consegna i file di log al bucket Amazon S3 specificato. Inoltre, puoi configurare altri AWS servizi per analizzare ulteriormente e agire in base ai dati sugli eventi raccolti nei log. CloudTrail Per ulteriori informazioni, consulta gli argomenti seguenti:

- Panoramica della creazione di un trail
- CloudTrail Servizi e integrazioni supportati
- Configurazione delle notifiche Amazon SNS per CloudTrail
- Ricezione di file di CloudTrail registro da più regioni e ricezione di file di CloudTrail registro da più account

Tutte HealthLake le azioni vengono registrate CloudTrail e sono documentate nell'HealthLake API Reference e in questa Guida per gli sviluppatori per le azioni eseguite utilizzando l'API REST FHIR. Ad esempio, le chiamate alle seguenti azioni generano voci nei file di registro: CloudTrail

- `DescribeFHIRImportJob`
- `DescribeFHIRExportJob`
- `StartFHIRImportJob`
- `ListFHIRImportJobs`
- `StartFHIRExportJob`
- `ListFHIRExportJobs`
- `CreateFHIRDatastore`
- `ListFHIRDatastores`
- `DeleteFHIRDatastore`
- `DescribeFHIRDatastore`
- `UpdateResource`
- `CreateResource`
- `DeleteResource`
- `ReadResource`
- `GetCapabilities`
- `SearchWithGet`
- `SearchWithPost`

Ogni evento o voce di log contiene informazioni sull'utente che ha generato la richiesta. Le informazioni di identità consentono di determinare quanto segue:

- Se la richiesta è stata effettuata con credenziali utente root o AWS Identity and Access Management (IAM).
- Se la richiesta è stata effettuata con le credenziali di sicurezza temporanee per un ruolo o un utente federato.
- Se la richiesta è stata effettuata da un altro AWS servizio.

Per ulteriori informazioni, consulta Elemento CloudTrail userIdentity.

Comprensione delle AWS HealthLake voci dei file di registro

Un trail è una configurazione che consente la distribuzione di eventi come file di log in un bucket Amazon S3 specificato dall'utente. CloudTrail i file di registro contengono una o più voci di registro. Un evento rappresenta una singola richiesta proveniente da qualsiasi fonte e include informazioni sull'azione richiesta, la data e l'ora dell'azione, i parametri della richiesta e così via. CloudTrail i file di registro non sono una traccia ordinata dello stack delle chiamate API pubbliche, quindi non vengono visualizzati in un ordine specifico.

L'esempio seguente mostra una voce di CloudTrail registro che illustra l'`CreateFHIRDatastore`azione.


```
{
    "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"
}
```


Monitoraggio delle HealthLake metriche tramite Amazon CloudWatch

Puoi monitorare HealthLake utilizzando Amazon CloudWatch, che raccoglie dati grezzi e li elabora in metriche leggibili quasi in tempo reale. Queste statistiche vengono conservate per 15 mesi, quindi puoi utilizzare tali informazioni storiche e avere una prospettiva migliore sulle prestazioni della tua applicazione o del tuo servizio web. È anche possibile impostare allarmi che controllano determinate soglie e inviare notifiche o intraprendere azioni quando queste soglie vengono raggiunte. Per ulteriori informazioni, consulta la Amazon CloudWatch User Guide.

> **Note**
>
> Le metriche vengono riportate per tutte le HealthLakeazioni native.

Le tabelle seguenti elencano le HealthLake metriche e le dimensioni a cui sono state riferite. CloudWatch Ciascuna viene presentata come conteggio delle frequenze per un intervallo di dati specificato dall'utente.

Le seguenti HealthLake metriche sono riportate a. CloudWatch

HealthLake metriche riportate a CloudWatch

Metrica	Description
Numero di chiamate	Il numero di chiamate verso APIs. Questo può essere segnalato per l'account o per un archivio dati specificato. Unità: numero Statistiche valide: somma, conteggio Dimensioni: operazione, ID del datastore, tipo di archivio dati
Richieste riuscite	Il numero di richieste API riuscite. Unità: numero Statistiche valide: Sum, Average Dimensioni: funzionamento, ID dell'archivio dati, tipo di archivio dati
Errori dell'utente	Il numero di richieste non riuscite a causa di un errore dell'utente. Unità: numero Statistiche valide: Sum, Average Dimensioni: funzionamento, ID dell'archivio dati, tipo di archivio dati
Errori del server	Il numero di richieste non riuscite a causa di un errore del server.

Metrica	Description
	Unità: numero Statistiche valide: Sum, Average Dimensioni: funzionamento, ID dell'archivio dati, tipo di archivio dati
Richieste con throttling	Il numero di richieste che sono state limitate. Questa metrica non è inclusa nel conteggio degli errori degli utenti o del server. Unità: numero Statistiche valide: Sum, Average Dimensioni: funzionamento, ID dell'archivio dati, tipo di archivio dati
Latenza	Il tempo impiegato in millisecondi per elaborare la richiesta dell'utente. Unità: millisecondi Statistiche valide: Minima, Massima, Media Dimensioni: funzionamento, ID dell'archivio dati, tipo di archivio dati

Le seguenti HealthLake dimensioni sono segnalate a CloudWatch.

HealthLake Dimensioni riportate a CloudWatch

Dimensione	Description
Operation	L'operazione API utilizzata nella richiesta
DataStoreID	L'ID dell'archivio dati utilizzato nella richiesta
DataStoreType	Il tipo di archivio dati utilizzato nella richiesta

Puoi ottenere metriche per la HealthLake Console di gestione AWS AWS CLI, o l' CloudWatchAPI. Puoi utilizzare l' CloudWatch API tramite uno degli Amazon AWS Software Development Kit (SDKs) o gli strumenti CloudWatch API. La HealthLake console mostra grafici basati sui dati grezzi dell' CloudWatch API.

È necessario disporre delle CloudWatch autorizzazioni appropriate con cui eseguire il monitoraggio HealthLake . CloudWatch Per ulteriori informazioni, consulta la sezione Gestione delle identità e degli accessi per Amazon CloudWatch nella Amazon CloudWatch User Guide.

Visualizzazione delle HealthLake metriche

Per visualizzare le metriche (console) CloudWatch

1. Accedi a Console di gestione AWS e apri la CloudWatchconsole.
2. Scegli Metriche, scegli Tutte le metriche, quindi scegli AWS/. HealthLake
3. Scegliere la dimensione, selezionare un nome parametro e scegliere Add to graph (Aggiungi a grafico).
4. Seleziona un valore per l'intervallo di date. Il numero di parametri per l'intervallo di date selezionato è visualizzato nel grafico.

Creazione di un allarme utilizzando CloudWatch

Un CloudWatch allarme controlla una singola metrica in un periodo di tempo specificato ed esegue una o più azioni: inviare una notifica a un argomento di Amazon Simple Notification Service (SNS) o a una politica di Auto Scaling. L'azione o le azioni si basano sul valore della metrica relativo a una determinata soglia per un certo numero di periodi di tempo specificati. CloudWatch può anche inviarti un messaggio SNS quando l'allarme cambia stato.

> **Note**
>
> CloudWatch gli allarmi richiamano azioni solo quando lo stato cambia e persiste per il periodo specificato.

Per visualizzare le metriche (console) CloudWatch

1. Accedi alla console CloudWatch.

2. Seleziona Alarms (Allarmi), quindi Create Alarm (Crea allarme).
3. Scegli AWS/ HealthLake, quindi scegli una metrica.
4. In Time Range (Intervallo di tempo), scegli un intervallo di tempo durante il quale eseguire il monitoraggio, quindi scegli Next (Successivo).
5. Immetti il Name (Nome) e la Description (Descrizione).
6. Per Whenever, scegli >= e digita un valore massimo.
7. Se desideri CloudWatch inviare un'e-mail quando viene raggiunto lo stato di allarme, nella sezione Azioni, per Ogni volta che si verifica un allarme, scegli Lo stato è ALLARME. Per Invia notifica a, scegli una mailing list o scegli Nuova lista e crea una nuova mailing list.
8. Visualizza un'anteprima dell'allarme nella sezione Alarm Preview (Anteprima allarme). Se sei soddisfatto dell'allarme, scegli Create Alarm (Crea allarme).

Monitoraggio HealthLake degli eventi tramite Amazon EventBridge

Amazon EventBridge è un servizio serverless che utilizza gli eventi per connettere tra loro i componenti delle applicazioni, semplificando la creazione di applicazioni scalabili basate sugli eventi. La base di EventBridge è creare regole che indirizzino gli eventi verso gli obiettivi. AWS HealthLake fornisce una fornitura duratura delle modifiche di stato a EventBridge. Per ulteriori informazioni, consulta What is Amazon EventBridge? nella Amazon EventBridge User Guide.

> **Note**
>
> Per informazioni su come inviare HealthLake eventi ad Amazon EventBridge, consulta Amazon EventBridge integration for AWS HealthLake nel blog AWS for Industries.

Argomenti

- HealthLake eventi inviati a EventBridge
- HealthLake struttura dell'evento

HealthLake eventi inviati a EventBridge

La tabella seguente elenca tutti HealthLake gli eventi inviati EventBridge ad essere elaborati.

HealthLake tipo di evento	Stato
eventi dell'archivio dati	
Creazione di archivi dati	`CREATING`
Data Store attivo	`ACTIVE`
Eliminazione dell'archivio dati	`DELETING`
Archivio dati eliminato	`DELETED`
Creazione dell'archivio dati non riuscita	`CREATE_FAILED`
Per ulteriori informazioni, consulta `datastoreStatus` nella documentazione di riferimento dell'API AWS HealthLake .	
Importazione degli eventi di lavoro	
Importa Job inviato	`SUBMITTED`
Importazione Job in corso	`IN_PROGRESS`
Job di importazione completato con errori	`COMPLETED_WITH_ERRORS`
Importazione Job completata	`COMPLETED`
Importazione Job non riuscita	`FAILED`
Per ulteriori informazioni, consulta `jobStatus` nella documentazione di riferimento dell'API AWS HealthLake .	
Esportazione degli eventi di lavoro	
Export Job inviato	`SUBMITTED`
Esporta Job in corso	`IN_PROGRESS`
Job di esportazione completato con errori	`COMPLETED_WITH_ERRORS`
Job di esportazione completato	`COMPLETED`

HealthLake tipo di evento	Stato
Esportazione Job non riuscita	`FAILED`

Per ulteriori informazioni, consulta `jobStatus` nella documentazione di riferimento dell'API AWS HealthLake .

HealthLake struttura dell'evento

HealthLake gli eventi sono oggetti con struttura JSON che contengono anche dettagli sui metadati. È possibile utilizzare i metadati come input per ricreare un evento o ottenere ulteriori informazioni. Tutti i campi di metadati associati sono elencati in una tabella sotto gli esempi di codice nei seguenti menu. Per ulteriori informazioni, consulta i metadati degli eventi di AWS servizio nella Amazon EventBridge User Guide.

> **Note**
>
> Per informazioni su come inviare HealthLake eventi ad Amazon EventBridge, consulta Amazon EventBridge integration for AWS HealthLake nel blog AWS for Industries.

Eventi del data store

Data Store Creating

Stato - **`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

Stato - **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

Stato - **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

Stato - **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/"
    }
}
```

Eventi dell'archivio dati: descrizioni dei metadati

Name	Tipo	Description
`version`	stringa	La versione dello schema degli EventBridge eventi.
`id`	stringa	L'UUID della versione 4 generato per ogni evento.
`detail-type`	stringa	Il tipo di evento che viene inviato.
`source`	stringa	Identifica il servizio che ha generato l’evento.
`account`	stringa	L'ID dell'account AWS a 12 cifre del proprietario del data store.
`time`	stringa	L'ora in cui si è verificato l'evento.
`region`	stringa	Identifica la AWS regione dell'archivio dati.
`resources`	array (stringa)	Un array JSON che contiene l'ARN dell'archivio dati.
`detail`	oggetto	Un oggetto JSON contenente informazioni sull’evento.

Name	Tipo	Description
`detail.datastoreId`	stringa	L'ID del data store associato all'evento di modifica dello stato.
`detail.datastoreName`	stringa	Il nome del data store.
`detail.datastoreTypeVersion`	stringa	La versione FHIR del data store.
`detail.datastoreEndpoint`	stringa	L'endpoint del data store.

Importa eventi di lavoro

Import Job Submitted

Stato - **`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

Stato - **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

Stato - **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

Stato - **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

Stato - **`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/"
        }
    }
}
```

Importa eventi di lavoro - descrizioni dei metadati

Name	Tipo	Description
`version`	stringa	La versione dello schema degli EventBridge eventi.

Name	Tipo	Description
`id`	stringa	L'UUID della versione 4 generato per ogni evento.
`detail-type`	stringa	Il tipo di evento che viene inviato.
`source`	stringa	Identifica il servizio che ha generato l’evento.
`account`	stringa	L'ID dell'account AWS a 12 cifre del proprietario del data store.
`time`	stringa	L'ora in cui si è verificato l'evento.
`region`	stringa	Identifica la AWS regione dell'archivio dati.
`resources`	array (stringa)	Un array JSON che contiene l'ARN dell'archivio dati.
`detail`	oggetto	Un oggetto JSON contenente informazioni sull’evento.
`detail.jobId`	stringa	L'ID del processo di importazione associato all'evento di modifica dello stato.
`detail.submitTime`	stringa	L'ora in cui è stato inviato il processo di importazione.
`detail.datastoreId`	stringa	L'archivio dati che ha generato l'evento di modifica dello stato.

Name	Tipo	Description
`detail.inputDataConfig`	stringa	Il percorso del prefisso di input per il bucket Amazon S3 che contiene i file FHIR da importare.

Esporta eventi di lavoro

Export Job Submitted

Stato - **`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

Stato - **`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

Stato - **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

Stato - **`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

Stato - **`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/"
        }
    }
}
```

Esporta eventi di lavoro - descrizioni dei metadati

Name	Tipo	Description
`version`	stringa	La versione dello schema degli EventBridge eventi.
`id`	stringa	L'UUID della versione 4 generato per ogni evento.
`detail-type`	stringa	Il tipo di evento che viene inviato.
`source`	stringa	Identifica il servizio che ha generato l'evento.

Name	Tipo	Description
`account`	stringa	L'ID dell'account AWS a 12 cifre del proprietario del data store.
`time`	stringa	L'ora in cui si è verificato l'evento.
`region`	stringa	Identifica la AWS regione dell'archivio dati.
`resources`	array (stringa)	Un array JSON che contiene l'ARN dell'archivio dati.
`detail`	oggetto	Un oggetto JSON contenente informazioni sull’evento.
`detail.jobId`	stringa	L'ID del lavoro di esportazi one associato all'evento di modifica dello stato.
`detail.submitTime`	stringa	L'ora in cui è stato inviato il processo di esportazione.
`detail.datastoreId`	stringa	L'archivio dati che ha generato l'evento di modifica dello stato.
`detail.outputDataC onfig`	stringa	Il percorso del prefisso di output per il bucket Amazon S3 che contiene i file FHIR da esportare.

Sicurezza in AWS HealthLake

La sicurezza del cloud AWS è la massima priorità. In qualità di AWS cliente, puoi beneficiare di un data center e di un'architettura di rete progettati per soddisfare i requisiti delle organizzazioni più sensibili alla sicurezza.

La sicurezza è una responsabilità condivisa tra AWS te e te. Il modello di responsabilità condivisa descrive questo aspetto come sicurezza del cloud e sicurezza nel cloud:

- La sicurezza del cloud AWS è responsabile della protezione dell'infrastruttura che gestisce AWS i servizi nel AWS cloud. AWS ti fornisce anche servizi che puoi utilizzare in modo sicuro. Third-party i revisori testano e verificano regolarmente l'efficacia della nostra sicurezza nell'ambito dei AWS Programmi di AWS conformità dei Programmi di conformità dei di . Per ulteriori informazioni sui programmi di conformità applicabili HealthLake, consulta AWS Services in Scope by Compliance Program .
- Sicurezza nel cloud: la tua responsabilità è determinata dal AWS servizio che utilizzi. L'utente è anche responsabile di altri fattori, tra cui la riservatezza dei dati, i requisiti della propria azienda e le leggi e normative vigenti.

Questa documentazione ti aiuta a capire come applicare il modello di responsabilità condivisa durante l'utilizzo HealthLake. Negli argomenti seguenti viene illustrato come eseguire la configurazione HealthLake per soddisfare gli obiettivi di sicurezza e conformità. Imparerai anche a usare altri servizi AWS che ti aiutano a monitorare e proteggere HealthLake le tue risorse.

Argomenti

- Protezione dei dati in AWS HealthLake
- Crittografia presso REST per AWS HealthLake
- Crittografia in transito per AWS HealthLake
- Gestione delle identità e degli accessi per AWS HealthLake
- Convalida della conformità per AWS HealthLake
- Sicurezza dell'infrastruttura in AWS HealthLake
- Creazione di AWS HealthLake risorse con AWS CloudFormation
- AWS HealthLake e endpoint VPC di interfaccia ()AWS PrivateLink
- Le migliori pratiche di sicurezza in AWS HealthLake

- Resilienza in AWS HealthLake

Protezione dei dati in AWS HealthLake

Il modello di responsabilità AWS condivisa Il modello di si applica alla protezione dei dati in AWS HealthLake. Come descritto in questo modello, AWS è responsabile della protezione dell'infrastruttura globale che gestisce tutti i Cloud AWS. L'utente è responsabile del controllo dei contenuti ospitati su questa infrastruttura. L'utente è inoltre responsabile della configurazione della protezione e delle attività di gestione per i Servizi AWS utilizzati. Per ulteriori informazioni sulla privacy dei dati, consulta Domande frequenti sulla privacy dei dati . Per informazioni sulla protezione dei dati in Europa, consulta il General Data Protection Regulation (GDPR) Center.

Ai fini della protezione dei dati, ti consigliamo di proteggere Account AWS le credenziali e di configurare i singoli utenti con AWS IAM Identity Center or AWS Identity and Access Management (IAM). In tal modo, a ogni utente verranno assegnate solo le autorizzazioni necessarie per svolgere i suoi compiti. Suggeriamo, inoltre, di proteggere i dati nei seguenti modi:

- Utilizza l'autenticazione a più fattori (MFA) con ogni account.
- SSL/TLS Da utilizzare per comunicare con AWS le risorse. È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Configura l'API e la registrazione delle attività degli utenti con AWS CloudTrail. Per informazioni sull'utilizzo dei CloudTrail percorsi per acquisire AWS le attività, consulta Lavorare con i CloudTrail percorsi nella Guida per l'AWS CloudTrail utente.
- Utilizza soluzioni di AWS crittografia, insieme a tutti i controlli di sicurezza predefiniti all'interno Servizi AWS.
- Utilizza i servizi di sicurezza gestiti avanzati, come Amazon Macie, che aiutano a individuare e proteggere i dati sensibili archiviati in Amazon S3.
- Se hai bisogno di moduli crittografici convalidati FIPS 140-3 per accedere AWS tramite un'interfaccia a riga di comando o un'API, usa un endpoint FIPS. Per ulteriori informazioni sugli endpoint FIPS disponibili, consulta il Federal Information Processing Standard (FIPS) 140-3.

Ti consigliamo di non inserire mai informazioni riservate o sensibili, ad esempio gli indirizzi e-mail dei clienti, nei tag o nei campi di testo in formato libero, ad esempio nel campo Nome. Ciò include quando lavori o Servizi AWS utilizzi la console, l'API HealthLake o gli SDK. AWS CLI AWS I dati inseriti nei tag o nei campi di testo in formato libero utilizzati per i nomi possono essere utilizzati per i la fatturazione o i log di diagnostica. Quando si fornisce un URL a un server esterno, suggeriamo

vivamente di non includere informazioni sulle credenziali nell'URL per convalidare la richiesta al server.

Crittografia presso REST per AWS HealthLake

HealthLake fornisce la crittografia di default per proteggere i dati sensibili dei clienti archiviati utilizzando una chiave AWS Key Management Service (AWS KMS) di proprietà del servizio. Customer-managed Sono supportate anche le chiavi KMS, necessarie sia per importare che per esportare file da un data store. Per ulteriori informazioni su Customer-managed KMS Key, consulta Amazon Key Management Service. I clienti possono scegliere una chiave KMS di proprietà di AWS o una chiave Customer-managed KMS quando creano un data store. La configurazione di crittografia non può essere modificata dopo la creazione di un data store. Se un data store utilizza una chiave KMS di proprietà di AWS, verrà indicata come AWS_OWNED_KMS_KEY e non verrà visualizzata la chiave specifica utilizzata per la crittografia.

Chiave KMS di proprietà di AWS

HealthLake utilizza queste chiavi per impostazione predefinita per crittografare automaticamente le informazioni potenzialmente sensibili come i dati personali identificabili o i dati PHI (Private Health Information) inattivi. Le chiavi KMS di proprietà di AWS non sono archiviate nel tuo account. Fanno parte di una raccolta di chiavi KMS di proprietà e gestione di AWS da utilizzare in più account AWS. I servizi AWS possono utilizzare chiavi KMS di proprietà di AWS per proteggere i tuoi dati. Non puoi visualizzare, gestire, utilizzare le chiavi KMS di proprietà di AWS o verificarne l'utilizzo. Tuttavia, non è necessario eseguire alcuna operazione o modificare alcun programma per proteggere le chiavi che crittografano i dati.

Non ti viene addebitato un canone mensile o un canone di utilizzo se utilizzi chiavi KMS di proprietà di AWS e non vengono conteggiate nelle quote AWS KMS per il tuo account. Per ulteriori informazioni, consulta le chiavi di proprietà di AWS.

Chiavi KMS gestite dal cliente

HealthLake supporta l'uso di una chiave KMS simmetrica gestita dal cliente che puoi creare, possedere e gestire per aggiungere un secondo livello di crittografia rispetto alla crittografia esistente di proprietà di AWS. Avendo il pieno controllo di questo livello di crittografia, è possibile eseguire operazioni quali:

- Stabilire e mantenere politiche chiave, politiche IAM e sovvenzioni

- Ruotare i materiali crittografici delle chiavi
- Abilitare e disabilitare le policy delle chiavi
- Aggiungere tag
- Creare alias delle chiavi
- Pianificare l'eliminazione delle chiavi

Puoi anche utilizzarlo CloudTrail per tenere traccia delle richieste HealthLake inviate a per tuo AWS KMS conto. AWS KMS Si applicano costi aggiuntivi. Per ulteriori informazioni, consulta la sezione Customer Owned Keys.

Creazione di una chiave gestita dal cliente

Puoi creare una chiave simmetrica gestita dal cliente utilizzando la Console di gestione AWS o le AWS KMS API.

Segui i passaggi per la creazione di una chiave gestita dal cliente simmetrica nella AWS Key Management Service Developer Guide.

Le policy della chiave controllano l'accesso alla chiave gestita dal cliente. Ogni chiave gestita dal cliente deve avere esattamente una policy della chiave, che contiene istruzioni che determinano chi può usare la chiave e come la possono usare. Quando crei la chiave gestita dal cliente, è possibile specificare una policy della chiave. Per ulteriori informazioni, consulta Managing access to customer managed keys nella AWS Key Management Service Developer Guide.

Per utilizzare la chiave gestita dal cliente con HealthLake le tue risorse, kms: CreateGrant le operazioni devono essere consentite nella policy chiave. Ciò aggiunge una concessione a una chiave gestita dal cliente che controlla l'accesso a una chiave KMS specificata, che fornisce a un utente l'accesso alle operazioni kms:grant richieste. HealthLake Per ulteriori informazioni, vedi Utilizzo delle sovvenzioni.

Per utilizzare la chiave KMS gestita dai clienti con HealthLake le tue risorse, nella policy chiave devono essere consentite le seguenti operazioni API:

- kms: CreateGrant aggiunge le concessioni a una chiave KMS gestita dal cliente specifica che consente l'accesso alle operazioni di concessione.
- kms: DescribeKey fornisce i dettagli chiave gestiti dal cliente necessari per convalidare la chiave. Questo è necessario per tutte le operazioni.

- kms: GenerateDataKey fornisce l'accesso alle risorse di crittografia a riposo per tutte le operazioni di scrittura.
- KMS:Decrypt fornisce l'accesso alle operazioni di lettura o ricerca per risorse crittografate.

Di seguito è riportato un esempio di dichiarazione politica che consente a un utente di creare e interagire con un archivio dati in AWS HealthLake cui è crittografato da tale chiave:

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

Autorizzazioni IAM richieste per l'utilizzo di una chiave KMS gestita dal cliente

Quando si crea un data store con AWS KMS crittografia abilitata utilizzando una chiave KMS gestita dal cliente, sono necessarie le autorizzazioni sia per la policy chiave che per la policy IAM per l'utente o il ruolo che crea il data store. HealthLake

Puoi utilizzare la chiave kms: ViaService condition per limitare l'uso della chiave KMS solo alle richieste che provengono da. HealthLake

Per ulteriori informazioni sulle policy chiave, consulta Enabling IAM policies nella AWS Key Management Service Developer Guide.

L'utente IAM, il ruolo IAM o l'account AWS che crea i tuoi repository deve disporre delle autorizzazioni kms:CreateGrant, kms: GenerateDataKey e kms: più DescribeKey le autorizzazioni necessarie. HealthLake

Come HealthLake utilizza le sovvenzioni in AWS KMS

HealthLake richiede una concessione per utilizzare la chiave KMS gestita dal cliente. Quando crei un Data Store crittografato con una chiave KMS gestita dal cliente, HealthLake crea una concessione per tuo conto inviando una CreateGrantrichiesta ad AWS KMS. Le sovvenzioni in AWS KMS vengono utilizzate per consentire l' HealthLake accesso a una chiave KMS in un account cliente.

Le sovvenzioni HealthLake create per tuo conto non devono essere revocate o ritirate. Se revochi o ritiri la concessione che HealthLake autorizza l'uso delle chiavi AWS KMS nel tuo account, HealthLake non puoi accedere a questi dati, crittografare le nuove risorse FHIR inviate al data store o decrittografarle quando vengono estratte. Quando revochi o ritiri una sovvenzione per, la modifica avviene immediatamente. HealthLake Per revocare i diritti di accesso, è necessario eliminare l'archivio dati anziché revocare la concessione. Quando un data store viene eliminato, annulla le HealthLake concessioni per tuo conto.

Monitoraggio delle chiavi di crittografia per HealthLake

Puoi utilizzarlo CloudTrail per tenere traccia delle richieste HealthLake inviate a per tuo AWS KMS conto quando utilizzi una chiave KMS gestita dal cliente. Le voci di registro nel CloudTrail registro mostrano healthlake.amazonaws.com nel campo UserAgent per distinguere chiaramente le richieste effettuate da. HealthLake

Gli esempi seguenti sono CloudTrail eventi per CreateGrant GenerateDataKey, Decrypt e per monitorare le AWS KMS operazioni richiamate per accedere DescribeKey ai dati crittografati dalla chiave gestita dal cliente HealthLake .

Di seguito viene illustrato come utilizzare CreateGrant per consentire l'accesso HealthLake a una chiave KMS fornita dal cliente, abilitando l'utilizzo di tale chiave KMS HealthLake per crittografare tutti i dati inattivi del cliente.

Gli utenti non sono tenuti a creare le proprie sovvenzioni. HealthLake crea una sovvenzione per tuo conto inviando una CreateGrant richiesta ad AWS KMS. Le sovvenzioni AWS KMS vengono utilizzate per HealthLake consentire l'accesso a una AWS KMS chiave in un account 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"
}
```

Gli esempi seguenti mostrano come GenerateDataKey garantire che l'utente disponga delle autorizzazioni necessarie per crittografare i dati prima di archiviarli.

```
            {
    "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"
}
```


L'esempio seguente mostra come HealthLake richiama l'operazione Decrypt per utilizzare la chiave di dati crittografati archiviata per accedere ai dati crittografati.

```
         {
    "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"
}
```

L'esempio seguente mostra come HealthLake utilizza l' DescribeKey operazione per verificare se la AWS KMS chiave di proprietà AWS KMS del cliente è in uno stato utilizzabile e per aiutare l'utente a risolvere i problemi se non funziona.

```
         {
    "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"
}
```

Ulteriori informazioni

Le seguenti risorse forniscono ulteriori informazioni sulla crittografia dei dati a riposo.

Per ulteriori informazioni sui concetti di base di AWS Key Management Service, consulta la AWS KMS documentazione.

Per ulteriori informazioni sulle best practice di sicurezza, consulta la AWS KMS documentazione.

Crittografia in transito per AWS HealthLake

AWS HealthLake utilizza TLS 1.2 per crittografare i dati in transito attraverso l'endpoint pubblico e tramite i servizi di backend.

Gestione delle identità e degli accessi per AWS HealthLake

AWS Identity and Access Management (IAM) è un software Servizio AWS che aiuta un amministratore a controllare in modo sicuro l'accesso alle AWS risorse. Gli amministratori IAM controllano chi può essere autenticato (effettuato l'accesso) e autorizzato (disporre delle

autorizzazioni) a utilizzare le risorse. HealthLake IAM è uno Servizio AWS strumento che puoi utilizzare senza costi aggiuntivi.

Argomenti

- Destinatari
- Autenticazione con identità
- Gestione dell'accesso tramite policy
- Come AWS HealthLake funziona con IAM
- Esempi di policy basate sull'identità per AWS HealthLake
- AWS politiche gestite per AWS HealthLake
- Risoluzione dei problemi AWS HealthLake di identità e accesso

Destinatari

Il modo in cui utilizzi AWS Identity and Access Management (IAM) varia in base al tuo ruolo:

- Utente del servizio: richiedi le autorizzazioni all'amministratore se non riesci ad accedere alle funzionalità (consulta Risoluzione dei problemi AWS HealthLake di identità e accesso)
- Amministratore del servizio: determina l'accesso degli utenti e invia le richieste di autorizzazione (consulta Come AWS HealthLake funziona con IAM)
- Amministratore IAM: scrivi policy per gestire l'accesso (consulta Esempi di policy basate sull'identità per AWS HealthLake)

Autenticazione con identità

L'autenticazione è il modo in cui accedi AWS utilizzando le tue credenziali di identità. Devi autenticarti come utente IAM o assumendo un ruolo IAM. Utente root dell'account AWS

Puoi accedere come identità federata utilizzando credenziali provenienti da una fonte di identità come AWS IAM Identity Center (IAM Identity Center), autenticazione Single Sign-On o credenziali. Google/ Facebook Per ulteriori informazioni sull'accesso, consulta Come accedere all' Account AWS nella Guida per l'utente di Accedi ad AWS .

Per l'accesso programmatico, AWS fornisce un SDK e una CLI per firmare crittograficamente le richieste. Per ulteriori informazioni, consulta AWS Signature Version 4 per le richieste API nella Guida per l'utente di IAM.

Account AWS utente root

Quando si crea un Account AWS, si inizia con un'identità di accesso denominata utente Account AWS root che ha accesso completo a tutte Servizi AWS le risorse. Consigliamo vivamente di non utilizzare l'utente root per le attività quotidiane. Per le attività che richiedono le credenziali dell'utente root, consulta Attività che richiedono le credenziali dell'utente root nella Guida per l'utente IAM.

Identità federata

Come procedura ottimale, richiedi agli utenti umani di utilizzare la federazione con un provider di identità per accedere Servizi AWS utilizzando credenziali temporanee.

Un'identità federata è un utente della directory aziendale, del provider di identità Web o Directory Service che accede Servizi AWS utilizzando le credenziali di una fonte di identità. Le identità federate assumono ruoli che forniscono credenziali temporanee.

Per la gestione centralizzata degli accessi, si consiglia di utilizzare AWS IAM Identity Center. Per ulteriori informazioni, consulta Che cos'è il Centro identità IAM? nella Guida per l'utente di AWS IAM Identity Center .

Utenti e gruppi IAM

Un utente IAM è una identità che dispone di autorizzazioni specifiche per una singola persona o applicazione. Ti consigliamo di utilizzare credenziali temporanee invece di utenti IAM con credenziali a lungo termine. Per ulteriori informazioni, consulta Richiedere agli utenti umani di utilizzare la federazione con un provider di identità per accedere AWS utilizzando credenziali temporanee nella Guida per l'utente IAM.

Un gruppo IAM specifica una raccolta di utenti IAM e semplifica la gestione delle autorizzazioni per gestire gruppi di utenti di grandi dimensioni. Per ulteriori informazioni, consulta Casi d'uso per utenti IAM nella Guida per l'utente di IAM.

Ruoli IAM

Un ruolo IAM è un'identità con autorizzazioni specifiche che fornisce credenziali temporanee. Puoi assumere un ruolo passando da un ruolo utente a un ruolo IAM (console) o chiamando un'operazione AWS CLI o AWS API. Per ulteriori informazioni, consulta Metodi per assumere un ruolo nella Guida per l'utente di IAM.

I ruoli IAM sono utili per l'accesso degli utenti federati, le autorizzazioni utente IAM temporanee, l'accesso multi-account, l'accesso multi-servizio e le applicazioni in esecuzione su Amazon EC2. Per

maggiori informazioni, consultare Accesso a risorse multi-account in IAM nella Guida per l'utente IAM.

Gestione dell'accesso tramite policy

Puoi controllare l'accesso AWS creando policy e associandole a AWS identità o risorse. Una policy definisce le autorizzazioni quando è associata a un'identità o a una risorsa. AWS valuta queste politiche quando un preside effettua una richiesta. La maggior parte delle politiche viene archiviata AWS come documenti JSON. Per maggiori informazioni sui documenti delle policy JSON, consulta Panoramica delle policy JSON nella Guida per l'utente IAM.

Utilizzando le policy, gli amministratori specificano chi ha accesso a cosa definendo quale principale può eseguire azioni su quali risorse e in quali condizioni.

Per impostazione predefinita, utenti e ruoli non dispongono di autorizzazioni. Un amministratore IAM crea le policy IAM e le aggiunge ai ruoli, che gli utenti possono quindi assumere. Le policy IAM definiscono le autorizzazioni indipendentemente dal metodo utilizzato per eseguirle.

Policy basate sull'identità

Le policy basate su identità sono documenti di policy di autorizzazione JSON che è possibile collegare a un'identità (utente, gruppo o ruolo). Tali policy controllano le operazioni autorizzate per l'identità, nonché le risorse e le condizioni in cui possono essere eseguite. Per informazioni su come creare una policy basata su identità, consultare Definizione di autorizzazioni personalizzate IAM con policy gestite dal cliente nella Guida per l'utente IAM.

Le policy basate su identità possono essere policy in linea (con embedding direttamente in una singola identità) o policy gestite (policy autonome collegate a più identità). Per informazioni su come scegliere tra una policy gestita o una policy inline, consulta Scegliere tra policy gestite e policy in linea nella Guida per l'utente di IAM.

Policy basate sulle risorse

Le policy basate su risorse sono documenti di policy JSON che è possibile collegare a una risorsa. Gli esempi includono le policy di trust dei ruoli IAM e le policy dei bucket di Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarli per controllare l'accesso a una risorsa specifica. In una policy basata sulle risorse è obbligatorio specificare un'entità principale.

Le policy basate sulle risorse sono policy inline che si trovano in tale servizio. Non è possibile utilizzare le policy AWS gestite di IAM in una policy basata sulle risorse.

Altri tipi di policy

AWS supporta tipi di policy aggiuntivi che possono impostare le autorizzazioni massime concesse dai tipi di policy più comuni:

- Limiti delle autorizzazioni: imposta il numero massimo di autorizzazioni che una policy basata su identità ha la possibilità di concedere a un'entità IAM. Per ulteriori informazioni, consulta Limiti delle autorizzazioni per le entità IAM nella Guida per l'utente di IAM.
- Politiche di controllo del servizio (SCPs): specificano le autorizzazioni massime per un'organizzazione o un'unità organizzativa in. AWS Organizations Per ulteriori informazioni, consultare Policy di controllo dei servizi nella Guida per l'utente di AWS Organizations .
- Politiche di controllo delle risorse (RCPs): imposta le autorizzazioni massime disponibili per le risorse nei tuoi account. Per ulteriori informazioni, consulta Politiche di controllo delle risorse (RCPs) nella Guida per l'AWS Organizations utente.
- Policy di sessione: policy avanzate passate come parametro quando si crea una sessione temporanea per un ruolo o un utente federato. Per maggiori informazioni, consultare Policy di sessione nella Guida per l'utente IAM.

Più tipi di policy

Quando a una richiesta si applicano più tipi di policy, le autorizzazioni risultanti sono più complicate da comprendere. Per scoprire come si AWS determina se consentire o meno una richiesta quando sono coinvolti più tipi di policy, consulta Logica di valutazione delle policy nella IAM User Guide.

Come AWS HealthLake funziona con IAM

Prima di utilizzare IAM per gestire l'accesso a HealthLake, scopri con quali funzionalità IAM è disponibile l'uso HealthLake.

Funzionalità IAM che puoi utilizzare con AWS HealthLake

Funzionalità IAM	HealthLake supporto
Policy basate sull'identità	Sì

Funzionalità IAM	HealthLake supporto
Policy basate su risorse	No
Operazioni di policy	Sì
Risorse relative alle policy	Sì
Chiavi di condizione delle policy	Sì
ACLs	No
ABAC (tag nelle policy)	Sì
Credenziali temporanee	Sì
Autorizzazioni del principale	Sì
Ruoli di servizio	Sì
Ruoli collegati al servizio	No

Per avere una panoramica di alto livello su come HealthLake e altri AWS servizi funzionano con la maggior parte delle funzionalità IAM, consulta AWS i servizi che funzionano con IAM nella IAM User Guide.

Politiche basate sull'identità per AWS HealthLake

Supporta le policy basate sull'identità: sì

Le policy basate sull'identità sono documenti di policy di autorizzazione JSON che è possibile allegare a un'identità (utente, gruppo di utenti o ruolo IAM). Tali policy definiscono le operazioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta Definizione di autorizzazioni personalizzate IAM con policy gestite dal cliente nella Guida per l'utente di IAM.

Con le policy basate sull'identità di IAM, è possibile specificare quali operazioni e risorse sono consentite o respinte, nonché le condizioni in base alle quali le operazioni sono consentite o respinte. Per informazioni su tutti gli elementi utilizzabili in una policy JSON, consulta Guida di riferimento agli elementi delle policy JSON IAM nella Guida per l'utente IAM.

Esempi di politiche basate sull'identità per AWS HealthLake

Per visualizzare esempi di politiche basate sull' HealthLake identità, vedere. Esempi di policy basate sull'identità per AWS HealthLake

Politiche basate sulle risorse all'interno AWS HealthLake

Supporta le policy basate su risorse: no

Le policy basate su risorse sono documenti di policy JSON che è possibile collegare a una risorsa. Esempi di policy basate sulle risorse sono le policy di attendibilità dei ruoli IAM e le policy di bucket Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarli per controllare l'accesso a una risorsa specifica. Quando è collegata a una risorsa, una policy definisce le operazioni che un principale può eseguire su tale risorsa e a quali condizioni. In una policy basata sulle risorse è obbligatorio specificare un'entità principale. I principali possono includere account, utenti, ruoli, utenti federati o. Servizi AWS

Per consentire l'accesso multi-account, è possibile specificare un intero account o entità IAM in un altro account come entità principale in una policy basata sulle risorse. Per ulteriori informazioni, consulta Accesso a risorse multi-account in IAM nella Guida per l'utente IAM.

Azioni politiche per AWS HealthLake

Supporta le operazioni di policy: si

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento `Action` di una policy JSON descrive le operazioni che è possibile utilizzare per consentire o negare l'accesso in una policy. Includere le operazioni in una policy per concedere le autorizzazioni a eseguire l'operazione associata.

Per visualizzare un elenco di HealthLake azioni, vedere Azioni definite da AWS HealthLake nel Service Authorization Reference.

Le azioni politiche in HealthLake uso utilizzano il seguente prefisso prima dell'azione:

```
healthlake
```

Per specificare più azioni in una singola istruzione, separa ogni azione con una virgola.

```
"Action": [
      "healthlake:action1",
      "healthlake:action2"
         ]
```

Per visualizzare esempi di politiche HealthLake basate sull'identità, vedere. Esempi di policy basate sull'identità per AWS HealthLake

Risorse politiche per AWS HealthLake

Supporta le risorse relative alle policy: sì

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento JSON `Resource` della policy specifica l'oggetto o gli oggetti ai quali si applica l'operazione. Come best practice, specifica una risorsa utilizzando il suo nome della risorsa Amazon (ARN). Per le azioni che non supportano le autorizzazioni a livello di risorsa, si utilizza un carattere jolly (*) per indicare che l'istruzione si applica a tutte le risorse.

```
"Resource": "*"
```

Per visualizzare un elenco dei tipi di HealthLake risorse e relativi ARNs, vedere Resources defined by AWS HealthLake nel Service Authorization Reference. Per informazioni sulle azioni con cui è possibile specificare l'ARN di ogni risorsa, vedere Azioni definite da. AWS HealthLake

Per visualizzare esempi di politiche HealthLake basate sull'identità, vedere. Esempi di policy basate sull'identità per AWS HealthLake

Chiavi relative alle condizioni delle politiche per AWS HealthLake

Supporta le chiavi di condizione delle policy specifiche del servizio: sì

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento `Condition` specifica quando le istruzioni vengono eseguite in base a criteri definiti. È possibile compilare espressioni condizionali che utilizzano operatori di condizione, ad esempio

uguale a o minore di, per soddisfare la condizione nella policy con i valori nella richiesta. Per visualizzare tutte le chiavi di condizione AWS globali, consulta le chiavi di contesto delle condizioni AWS globali nella Guida per l'utente IAM.

Per visualizzare un elenco di chiavi di HealthLake condizione, consulta Condition keys for AWS HealthLake nel Service Authorization Reference. Per informazioni sulle azioni e le risorse con cui è possibile utilizzare una chiave di condizione, consulta Azioni definite da AWS HealthLake.

Per visualizzare esempi di politiche HealthLake basate sull'identità, vedere. Esempi di policy basate sull'identità per AWS HealthLake

Liste di controllo degli accessi () in ACLs AWS HealthLake

Supporti ACLs: no

Le liste di controllo degli accessi (ACLs) controllano quali principali (membri dell'account, utenti o ruoli) dispongono delle autorizzazioni per accedere a una risorsa. ACLs sono simili alle politiche basate sulle risorse, sebbene non utilizzino il formato del documento di policy JSON.

Controllo degli accessi basato sugli attributi (ABAC) con AWS HealthLake

Supporta ABAC (tag nelle policy): sì

Il controllo degli accessi basato su attributi (ABAC) è una strategia di autorizzazione che definisce le autorizzazioni in base ad attributi chiamati tag. Puoi allegare tag a entità e AWS risorse IAM, quindi progettare politiche ABAC per consentire le operazioni quando il tag del principale corrisponde al tag sulla risorsa.

Per controllare l'accesso basato su tag, fornire informazioni sui tag nell'elemento condizione di una policy utilizzando le chiavi di condizione `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` o `aws:TagKeys`.

Se un servizio supporta tutte e tre le chiavi di condizione per ogni tipo di risorsa, il valore per il servizio è Sì. Se un servizio supporta tutte e tre le chiavi di condizione solo per alcuni tipi di risorsa, allora il valore sarà Parziale.

Per maggiori informazioni su ABAC, consulta Definizione delle autorizzazioni con autorizzazione ABAC nella Guida per l'utente di IAM. Per visualizzare un tutorial con i passaggi per l'impostazione di ABAC, consulta Utilizzo del controllo degli accessi basato su attributi (ABAC) nella Guida per l'utente di IAM.

Utilizzo di credenziali temporanee con AWS HealthLake

Supporta le credenziali temporanee: sì

Le credenziali temporanee forniscono l'accesso a breve termine alle AWS risorse e vengono create automaticamente quando si utilizza la federazione o si cambia ruolo. AWS consiglia di generare dinamicamente credenziali temporanee anziché utilizzare chiavi di accesso a lungo termine. Per ulteriori informazioni, consulta Credenziali di sicurezza temporanee in IAM e Servizi AWS compatibili con IAM nella Guida per l'utente IAM.

Autorizzazioni principali multiservizio per AWS HealthLake

Supporta l'inoltro delle sessioni di accesso (FAS): sì

Le sessioni di accesso inoltrato (FAS) utilizzano le autorizzazioni del principale che chiama an Servizio AWS, combinate con la richiesta di effettuare richieste Servizio AWS ai servizi downstream. Per i dettagli delle policy relative alle richieste FAS, consulta Forward access sessions.

Ruoli di servizio per AWS HealthLake

Supporta i ruoli di servizio: sì

Un ruolo di servizio è un ruolo IAM che un servizio assume per eseguire operazioni per tuo conto. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta Create a role to delegate permissions to an Servizio AWS nella Guida per l'utente IAM.

Per informazioni sui ruoli di servizio e sulla politica in linea richiesta per l'accesso completo a AWS HealthLake, vedereConfigurazione AWS HealthLake.

> **Warning**
>
> La modifica delle autorizzazioni per un ruolo di servizio potrebbe comprometterne la funzionalità. HealthLake Modifica i ruoli di servizio solo quando viene HealthLake fornita una guida in tal senso.

Ruoli collegati ai servizi per AWS HealthLake

Supporta i ruoli collegati ai servizi: no

Un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un. Servizio AWS Il servizio può assumere il ruolo per eseguire un'operazione per tuo conto. I ruoli collegati al servizio vengono visualizzati nel tuo account Account AWS e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati al servizio, ma non modificarle.

Per ulteriori informazioni su come creare e gestire i ruoli collegati ai servizi, consulta Servizi AWS supportati da IAM. Trova un servizio nella tabella che include un `Yes` nella colonna Service-linked role (Ruolo collegato ai servizi). Scegli il collegamento Sì per visualizzare la documentazione relativa al ruolo collegato ai servizi per tale servizio.

Esempi di policy basate sull'identità per AWS HealthLake

Per impostazione predefinita, gli utenti e i ruoli non dispongono dell'autorizzazione per creare o modificare risorse HealthLake. Per concedere agli utenti l'autorizzazione a eseguire azioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM.

Per informazioni su come creare una policy basata su identità IAM utilizzando questi documenti di policy JSON di esempio, consulta Creazione di policy IAM (console) nella Guida per l'utente di IAM.

Per informazioni dettagliate sulle azioni e sui tipi di risorse definiti da HealthLake, incluso il formato di ARNs per ogni tipo di risorsa, vedere Azioni, risorse e chiavi di condizione AWS HealthLake nel Service Authorization Reference.

Argomenti

- Best practice per le policy
- Utilizzo della console AWS HealthLake
- Accesso a un AWS HealthLake data store in Amazon Athena
- Consentire agli utenti di visualizzare le loro autorizzazioni

Best practice per le policy

Le politiche basate sull'identità determinano se qualcuno può creare, accedere o eliminare HealthLake risorse nel tuo account. Queste azioni possono comportare costi aggiuntivi per l' Account AWS. Quando si creano o modificano policy basate sull'identità, seguire queste linee guida e raccomandazioni:

- Inizia con le policy AWS gestite e passa alle autorizzazioni con privilegi minimi: per iniziare a concedere autorizzazioni a utenti e carichi di lavoro, utilizza le politiche gestite che concedono

le autorizzazioni per molti casi d'uso comuni.AWS Sono disponibili nel tuo. Account AWS Ti consigliamo di ridurre ulteriormente le autorizzazioni definendo politiche gestite dai AWS clienti specifiche per i tuoi casi d'uso. Per maggiori informazioni, consulta Policy gestite da AWS o Policy gestite da AWS per le funzioni dei processi nella Guida per l'utente di IAM.

- Applicazione delle autorizzazioni con privilegio minimo - Quando si impostano le autorizzazioni con le policy IAM, concedere solo le autorizzazioni richieste per eseguire un'attività. È possibile farlo definendo le azioni che possono essere intraprese su risorse specifiche in condizioni specifiche, note anche come autorizzazioni con privilegio minimo. Per maggiori informazioni sull'utilizzo di IAM per applicare le autorizzazioni, consulta Policy e autorizzazioni in IAM nella Guida per l'utente di IAM.
- Condizioni d'uso nelle policy IAM per limitare ulteriormente l'accesso - Per limitare l'accesso ad azioni e risorse è possibile aggiungere una condizione alle policy. Ad esempio, è possibile scrivere una condizione di policy per specificare che tutte le richieste devono essere inviate utilizzando SSL. Puoi anche utilizzare le condizioni per concedere l'accesso alle azioni del servizio se vengono utilizzate tramite uno specifico Servizio AWS, ad esempio CloudFormation. Per maggiori informazioni, consultare la sezione Elementi delle policy JSON di IAM: condizione nella Guida per l'utente di IAM.
- Utilizzo dello strumento di analisi degli accessi IAM per convalidare le policy IAM e garantire autorizzazioni sicure e funzionali - Lo strumento di analisi degli accessi IAM convalida le policy nuove ed esistenti in modo che aderiscano al linguaggio (JSON) della policy IAM e alle best practice di IAM. Lo strumento di analisi degli accessi IAM offre oltre 100 controlli delle policy e consigli utili per creare policy sicure e funzionali. Per maggiori informazioni, consultare Convalida delle policy per il Sistema di analisi degli accessi IAM nella Guida per l'utente di IAM.
- Richiedi l'autenticazione a più fattori (MFA): se hai uno scenario che richiede utenti IAM o un utente root nel Account AWS tuo, attiva l'MFA per una maggiore sicurezza. Per richiedere la MFA quando vengono chiamate le operazioni API, aggiungere le condizioni MFA alle policy. Per maggiori informazioni, consultare Protezione dell'accesso API con MFA nella Guida per l'utente di IAM.

Per maggiori informazioni sulle best practice in IAM, consulta Best practice di sicurezza in IAM nella Guida per l'utente di IAM.

Utilizzo della console AWS HealthLake

Per accedere alla AWS HealthLake console, è necessario disporre di un set minimo di autorizzazioni. Queste autorizzazioni devono consentirti di elencare e visualizzare i dettagli sulle HealthLake risorse del tuo. Account AWS Se crei una policy basata sull'identità più restrittiva rispetto alle autorizzazioni

minime richieste, la console non funzionerà nel modo previsto per le entità (utenti o ruoli) associate a tale policy.

Non è necessario consentire autorizzazioni minime per la console agli utenti che effettuano chiamate solo verso AWS CLI o l' AWS API. Al contrario, è opportuno concedere l'accesso solo alle azioni che corrispondono all'operazione API che stanno cercando di eseguire.

Per un accesso completo a HealthLake, collega le seguenti policy a un utente o ruolo IAM: `AmazonHealthLakeFullAccess` and`AWSLakeFormationDataAdmin`. È inoltre necessario allegare la politica HealthLake in linea che è un ruolo di servizio. Un ruolo di servizio è un ruolo IAM che un servizio assume per eseguire operazioni per tuo conto. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta Create a role to delegate permissions to an Servizio AWS nella Guida per l'utente IAM. Per informazioni sulla politica in linea che crea il ruolo di servizio richiesto, vedere. Configurazione AWS HealthLake È inoltre necessario utilizzare la AWS Lake Formation console o la CLI per assegnare all'amministratore il ruolo di HealthLake amministratore di AWS Lake Formation Data Lake. Per ulteriori informazioni, consulta Configurazione AWS HealthLake.

Accesso a un AWS HealthLake data store in Amazon Athena

Se desideri fornire a utenti e ruoli l'accesso agli archivi HealthLake dati di Amazon Athena, collega le seguenti politiche IAM al ruolo o all'utente: `AmazonAthenaFullAccess` e`AmazonS3FullAccess`. `Select`e `Describe` le autorizzazioni sono richieste anche per le tabelle gestite da AWS Lake Formation. AWS Lake Formation le autorizzazioni per le tabelle vengono concesse da un AWS Lake Formation amministratore nella AWS Lake Formation console o tramite la CLI. Per ulteriori informazioni, consulta Configurazione AWS HealthLake

Consentire agli utenti di visualizzare le loro autorizzazioni

Questo esempio mostra in che modo è possibile creare una policy che consente agli utenti IAM di visualizzare le policy inline e gestite che sono collegate alla relativa identità utente. Questa politica include le autorizzazioni per completare questa azione sulla console o utilizzando l'API o a livello di codice. AWS CLI AWS

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

```
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS politiche gestite per AWS HealthLake

Una politica AWS gestita è una politica autonoma creata e amministrata da AWS. AWS le politiche gestite sono progettate per fornire autorizzazioni per molti casi d'uso comuni, in modo da poter iniziare ad assegnare autorizzazioni a utenti, gruppi e ruoli.

Tieni presente che le policy AWS gestite potrebbero non concedere le autorizzazioni con il privilegio minimo per i tuoi casi d'uso specifici, poiché sono disponibili per tutti i clienti. AWS Si consiglia pertanto di ridurre ulteriormente le autorizzazioni definendo policy gestite dal cliente specifiche per i propri casi d'uso.

Non è possibile modificare le autorizzazioni definite nelle politiche gestite. AWS Se AWS aggiorna le autorizzazioni definite in una politica AWS gestita, l'aggiornamento ha effetto su tutte le identità

principali (utenti, gruppi e ruoli) a cui è associata la politica. AWS è più probabile che aggiorni una policy AWS gestita quando ne Servizio AWS viene lanciata una nuova o quando diventano disponibili nuove operazioni API per i servizi esistenti.

Per ulteriori informazioni, consultare Policy gestite da AWS nella Guida per l'utente di IAM.

AWS politica gestita: AmazonHealthLakeFullAccess

La `AmazonHealthLakeFullAccess` politica fornisce l'accesso completo a HealthLake. Con questa policy associata al proprio utente o ruolo, gli utenti possono utilizzarla HealthLake per accedere, interrogare, importare ed esportare dati in HealthLake. Per eseguire molte azioni comuni in HealthLake, è necessario aggiungere politiche aggiuntive all'utente o al ruolo. Per ulteriori informazioni, consulta Configurazione AWS HealthLake e HealthLake operazioni e autorizzazioni.

È possibile allegare la policy `AmazonHealthLakeFullAccess` alle identità IAM.

Questa politica concede autorizzazioni amministrative e di collaborazione che consentono a utenti e ruoli di eseguire query, cercare, importare ed esportare e consente inoltre di eseguire azioni HealthLake per conto degli utenti e dei ruoli che dispongono di tali autorizzazioni. HealthLake

Dettagli delle autorizzazioni

Questa politica include la seguente dichiarazione.

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 politica gestita: AmazonHealthLakeReadOnlyAccess

`AmazonHealthLakeReadOnlyAccess`la politica concede l'accesso e le autorizzazioni in sola lettura HealthLake e alle risorse correlate in altri servizi. AWS Applica questo criterio agli utenti a cui desideri concedere la possibilità di interrogare e visualizzare gli HealthLake archivi dati, ma non la possibilità di crearli o modificarli.

È possibile allegare la policy `AmazonHealthLakeReadOnlyAccess` alle identità IAM.

Questa politica concede *`read-only`* autorizzazioni che consentono a utenti e ruoli di eseguire query. HealthLake

Dettagli delle autorizzazioni

Questa politica include la seguente dichiarazione.

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 operazioni e autorizzazioni

Nella tabella seguente sono elencate le operazioni tipiche di HealthLake e le autorizzazioni necessarie per eseguirle.

HealthLake operazioni	Autorizzazioni richieste
Creare un archivio dati in HealthLake	`AmazonHealthLakeFullAccess` `AmazonLakeFormationDataAdmin` , policy in linea e autorizzazioni di AWS Lake Formation amministratore gestite da AWS Lake Formation
Eliminare un archivio dati in HealthLake	`AmazonHealthLakeFullAccess` `AmazonLakeFormatio`

HealthLake operazioni	Autorizzazioni richieste
	`nDataAdmin` , policy in linea e autorizzazioni di AWS Lake Formation amministratore gestite da AWS Lake Formation
Elenca, cerca o interroga un archivio dati in HealthLake	`AmazonHealthLakeReadOnlyAccess`
Effettua una query su un data store utilizzando Amazon Athena	`AmazonAthenaFullAccess` AWS Lake Formation `Selecte Describe` autorizza zioni sulle tabelle gestite da `AmazonS3F ullAccess` AWS Lake Formation
Importa dati da HealthLake	Per informazioni, consulta Impostazione delle autorizzazioni per i lavori di importazione.
Esporta dati da HealthLake	Per informazioni, consulta Impostazione delle autorizzazioni per i lavori di esportazione.

HealthLake aggiornamenti alle politiche AWS gestite

Visualizza i dettagli sugli aggiornamenti delle politiche AWS gestite HealthLake dal momento in cui questo servizio ha iniziato a tenere traccia di queste modifiche. Per ricevere avvisi automatici sulle modifiche a questa pagina, iscriviti al feed RSS nella pagina della cronologia dei HealthLake documenti.

Modifica	Descrizione	Data
AmazonHealthLakeFullAccess	`AmazonHealthLakeFu llAccess` criterio richiesto per consentire l'accesso completo a. HealthLake	14 novembre 2022
AmazonHealthLakeRe adOnlyAccess	`AmazonHealthLakeRe adOnlyAccess` criterio richiesto per l'accesso in sola lettura a. HealthLake	14 novembre 2022

Modifica	Descrizione	Data
HealthLake ha iniziato a tenere traccia delle modifiche	HealthLake ha iniziato a tenere traccia delle modifiche per le sue politiche AWS gestite.	14 novembre 2022

Risoluzione dei problemi AWS HealthLake di identità e accesso

Utilizza le seguenti informazioni per aiutarti a diagnosticare e risolvere i problemi più comuni che potresti riscontrare quando lavori con un HealthLake IAM.

Argomenti

- Non sono autorizzato a eseguire alcuna azione in AWS HealthLake
- Non sono autorizzato a eseguire iam: PassRole
- Voglio consentire a persone esterne al mio AWS account di accedere alle mie AWS HealthLake risorse

Non sono autorizzato a eseguire alcuna azione in AWS HealthLake

Se ti Console di gestione AWS dice che non sei autorizzato a eseguire un'azione, devi contattare l'amministratore per ricevere assistenza. L'amministratore è la persona da cui si sono ricevuti il nome utente e la password.

L'errore di esempio seguente si verifica quando l'utente `mateojackson` IAM prova a utilizzare la console per visualizzare i dettagli relativi a una risorsa *`my-example-widget`* fittizia ma non dispone di autorizzazioni `healthlake:`*`GetWidget`* fittizie.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 healthlake:GetWidget on resource: my-example-widget
```

In questo caso, Mateo chiede al suo amministratore di aggiornare le policy per poter accedere alla risorsa *`my-example-widget`* mediante l'operazione `healthlake:`*`GetWidget`*.

Non sono autorizzato a eseguire iam: PassRole

Se ricevi un errore che indica che non sei autorizzato a eseguire l'operazione `iam:PassRole`, le tue policy devono essere aggiornate per poter passare un ruolo a HealthLake.

Alcuni Servizi AWS consentono di passare un ruolo esistente a quel servizio invece di creare un nuovo ruolo di servizio o un ruolo collegato al servizio. Per eseguire questa operazione, è necessario disporre delle autorizzazioni per trasmettere il ruolo al servizio.

L'errore di esempio seguente si verifica quando un utente IAM denominato `marymajor` cerca di utilizzare la console per eseguire un'operazione in HealthLake. Tuttavia, l'operazione richiede che il servizio disponga delle autorizzazioni concesse da un ruolo di servizio. Mary non dispone delle autorizzazioni per trasmettere il ruolo al servizio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In questo caso, le policy di Mary devono essere aggiornate per poter eseguire l'operazione `iam:PassRole`.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Voglio consentire a persone esterne al mio AWS account di accedere alle mie AWS HealthLake risorse

È possibile creare un ruolo con il quale utenti in altri account o persone esterne all'organizzazione possono accedere alle tue risorse. È possibile specificare chi è attendibile per l'assunzione del ruolo. Per i servizi che supportano politiche basate sulle risorse o liste di controllo degli accessi (ACLs), puoi utilizzare tali politiche per concedere alle persone l'accesso alle tue risorse.

Per maggiori informazioni, consulta gli argomenti seguenti:

- Per sapere se HealthLake supporta queste funzionalità, consulta. Come AWS HealthLake funziona con IAM
- Per scoprire come fornire l'accesso alle tue risorse attraverso Account AWS le risorse di tua proprietà, consulta Fornire l'accesso a un utente IAM in un altro Account AWS di tua proprietà nella IAM User Guide.
- Per scoprire come fornire l'accesso alle tue risorse a terze parti Account AWS, consulta Fornire l'accesso a soggetti Account AWS di proprietà di terze parti nella Guida per l'utente IAM.

- Per informazioni su come fornire l'accesso tramite la federazione delle identità, consulta Fornire l'accesso a utenti autenticati esternamente (federazione delle identità) nella Guida per l'utente IAM.
- Per informazioni sulle differenze di utilizzo tra ruoli e policy basate su risorse per l'accesso multi-account, consulta Accesso a risorse multi-account in IAM nella Guida per l'utente IAM.

Convalida della conformità per AWS HealthLake

Third-party i revisori valutano la sicurezza e la conformità nell' AWS HealthLake ambito di più programmi di AWS conformità. Per HealthLake questo include l'HIPAA.

Per sapere se un Servizio AWS programma rientra nell'ambito di specifici programmi di conformità, consulta Servizi AWS la sezione Scope by Compliance Program Servizi AWS e scegli il programma di conformità che ti interessa. Per informazioni generali, consulta Programmi di AWS conformità Programmi di di .

È possibile scaricare report di audit di terze parti utilizzando AWS Artifact. Per ulteriori informazioni, consulta Scaricamento dei report in AWS Artifact .

La vostra responsabilità di conformità durante l'utilizzo Servizi AWS è determinata dalla sensibilità dei dati, dagli obiettivi di conformità dell'azienda e dalle leggi e dai regolamenti applicabili. Per ulteriori informazioni sulla responsabilità di conformità durante l'utilizzo Servizi AWS, consulta la Documentazione AWS sulla sicurezza.

Sicurezza dell'infrastruttura in AWS HealthLake

In quanto servizio gestito, AWS HealthLake è protetto dalle procedure di sicurezza della rete AWS globale descritte nel white paper Amazon Web Services: Overview of Security Processes.

Utilizzi chiamate API AWS pubblicate per accedere HealthLake attraverso la rete. I client devono supportare Transport Layer Security (TLS) 1.0 o versioni successive. È consigliabile TLS 1.2 o versioni successive. I client devono inoltre supportare suite di crittografia con Perfect Forward Secrecy (PFS) come Ephemeral (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Diffie-Hellman La maggior parte dei sistemi moderni, come Java 7 e versioni successive, supporta tali modalità.

Inoltre, le richieste devono essere firmate utilizzando un ID chiave di accesso e una chiave di accesso segreta associata a un principale IAM. In alternativa è possibile utilizzare AWS Security

Token Service (AWS STS) per generare credenziali di sicurezza temporanee per sottoscrivere le richieste.

Creazione di AWS HealthLake risorse con AWS CloudFormation

AWS HealthLake è integrato con AWS CloudFormation, un servizio che consente di modellare e configurare le AWS risorse in modo da dedicare meno tempo alla creazione e alla gestione delle risorse e dell'infrastruttura. Crei un modello che descrive tutte le AWS risorse che desideri e fornisce CloudFormation e configura tali risorse per te.

Quando lo utilizzi CloudFormation, puoi riutilizzare il modello per configurare le HealthLake risorse in modo coerente e ripetuto. Descrivi le tue risorse una sola volta, quindi fornisci le stesse risorse più e più volte in più Account AWS aree geografiche.

HealthLake e CloudFormation modelli

Per fornire e configurare le risorse HealthLake e i servizi correlati, è necessario conoscere CloudFormation i modelli. I modelli sono file di testo formattati in JSON o YAML. Questi modelli descrivono le risorse che desideri fornire negli CloudFormation stack. Se non conosci JSON o YAML, puoi usare CloudFormation Designer per iniziare a usare i modelli. CloudFormation Per ulteriori informazioni, consulta Che cos'è CloudFormation Designer? nella Guida per l'utente di AWS CloudFormation .

> **Note**
>
> AWS HealthLake supporta la creazione di archivi dati con. CloudFormationPer ulteriori informazioni, inclusi esempi di modelli JSON e YAML per il provisioning degli archivi HealthLake dati, consulta il riferimento ai tipi di AWS HealthLake risorse nella Guida per l'utente.AWS CloudFormation

Scopri di più su CloudFormation

Per ulteriori informazioni CloudFormation, consulta le seguenti risorse:

- AWS CloudFormation
- AWS CloudFormation Guida per l'utente
- Documentazione di riferimento dell'API CloudFormation

- AWS CloudFormation Guida per l'utente dell'interfaccia a riga di comando

AWS HealthLake e endpoint VPC di interfaccia ()AWS PrivateLink

Puoi stabilire una connessione privata tra il tuo VPC e creare un AWS HealthLake endpoint VPC di interfaccia. Gli endpoint VPC di interfaccia sono alimentati da AWS PrivateLinkuna tecnologia che puoi utilizzare per l'accesso privato HealthLake; APIs senza un gateway Internet, un dispositivo NAT, una connessione VPN o una connessione. Direct Connect Le istanze nel tuo VPC non necessitano di indirizzi IP pubblici con cui comunicare HealthLake;. APIs Il traffico tra il tuo VPC e HealthLake; non esce dalla rete Amazon.

Ogni endpoint dell'interfaccia è rappresentato da una o più interfacce di rete elastiche nelle sottoreti.

Per ulteriori informazioni, consulta Interface VPC endpoints (AWS PrivateLink) nella Amazon VPC User Guide.

Considerazioni sugli endpoint HealthLake VPC

Prima di configurare un endpoint VPC di interfaccia per HealthLake, assicurati di consultare le proprietà e le limitazioni degli endpoint dell'interfaccia nella Amazon VPC User Guide.

HealthLake supporta l'esecuzione di chiamate a tutte le sue azioni API dal tuo VPC.

Creazione di un endpoint VPC di interfaccia per; HealthLake

Puoi creare un endpoint VPC per il servizio HealthLake; utilizzando la console Amazon VPC o il (). AWS Command Line Interface AWS CLI Per ulteriori informazioni, consultare Creazione di un endpoint di interfaccia nella Guida per l'utente di Amazon VPC.

Crea un endpoint VPC per HealthLake; utilizzando il seguente nome di servizio:

- com.amazonaws. *region*.salutelake

Se attivi il DNS privato per l'endpoint, puoi effettuare richieste API HealthLake utilizzando il nome DNS predefinito per la regione. Ad esempio, *healthlake.us-east-1.amazonaws.com*.

Per ulteriori informazioni, consultare Accesso a un servizio tramite un endpoint di interfaccia in Guida per l'utente di Amazon VPC.

Creazione di una policy per gli endpoint VPC per HealthLake

È possibile allegare un criterio all'endpoint VPC che controlla l'accesso all' HealthLake. La policy specifica le informazioni riportate di seguito:

- Il principale che può eseguire operazioni.
- Le azioni che possono essere eseguite.
- Le risorse sui cui si possono eseguire operazioni.

Per ulteriori informazioni, consulta Controllo degli accessi ai servizi con endpoint VPC in Guida per l'utente di Amazon VPC.

Esempio: policy degli endpoint VPC per le azioni HealthLake

Di seguito è riportato un esempio di policy sugli endpoint per. HealthLake Se collegata a un endpoint, questa policy garantisce l'accesso all' HealthLake`CreateFHIRDatastore`azione a tutti i principali su tutte le risorse.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "healthlake:create-fhir-datastore"
         ],
         "Resource":"*"
      }
   ]
}
```

Le migliori pratiche di sicurezza in AWS HealthLake

AWS HealthLake fornisce una serie di funzionalità di sicurezza da considerare durante lo sviluppo e l'implementazione delle proprie politiche di sicurezza. Le seguenti best practice sono linee guida generali e non rappresentano una soluzione di sicurezza completa. Poiché queste best practice potrebbero non essere appropriate o sufficienti per l’ambiente, sono da considerare come considerazioni utili anziché prescrizioni.

- Implementa l'accesso con privilegi minimi.
- Quando possibile, utilizza Customer-Managed-Keys (CMK) per crittografare i dati. Per ulteriori informazioni sulle CMK, consulta Amazon Key Management Service.
- Usa la ricerca con POST, non la ricerca con GET quando richiedi PHI o PII nel tuo archivio dati.
- Limita l'accesso a funzioni di controllo sensibili e importanti.
- Quando create risorse tramite le API di aggiornamento o importazione in blocco, non utilizzate PHI o PII, inclusi i nomi degli archivi dati e dei lavori, in alcun campo visibile o nell'ID FHIR logico (LID).
- Quando inviate richieste di creazione, lettura, aggiornamento, eliminazione o ricerca, non utilizzate PHI nell'intestazione HTTP.
- Abilita AWS CloudTrail per controllare AWS HealthLake l'utilizzo e per garantire che non vi siano attività impreviste.
- Consulta le best practice per utilizzare i bucket Amazon S3 in modo sicuro. Per ulteriori informazioni, consulta le best practice di sicurezza nella guida per l'utente di Amazon S3.

Resilienza in AWS HealthLake

L'infrastruttura AWS globale è costruita attorno a AWS regioni e zone di disponibilità. AWS Le regioni forniscono più zone di disponibilità fisicamente separate e isolate, collegate con reti a bassa latenza, ad alto throughput e altamente ridondanti. Con le zone di disponibilità, puoi progettare e gestire applicazioni e database che eseguono automaticamente il failover tra zone di disponibilità senza interruzioni. Le zone di disponibilità sono più disponibili, tolleranti ai guasti e scalabili rispetto alle infrastrutture a data center singolo o multiplo tradizionali.

Se occorre replicare i dati o le applicazioni su distanze geografiche più ampie, utilizza le regioni locali AWS . Una regione AWS locale è un singolo data center progettato per integrare una regione esistente. AWS Come tutte le AWS regioni, le regioni AWS locali sono completamente isolate dalle altre AWS regioni.

Per ulteriori informazioni su AWS regioni e zone di disponibilità, consulta AWS Global Infrastructure.

AWS HealthLake riferimento

Il seguente materiale di riferimento di supporto è disponibile per SMART su FHIR, FHIR e. AWS HealthLake

> **Note**
>
> Tutte le HealthLake azioni e i tipi di dati nativi sono descritti in un riferimento separato. Per ulteriori informazioni, consulta la documentazione di riferimento dell'API di AWS HealthLake .

Argomenti

- Supporto SMART su FHIR per AWS HealthLake
- Supporto FHIR R4 per AWS HealthLake
- Riferimento di conformità per AWS HealthLake
- Supporto di riferimento per AWS HealthLake

Supporto SMART su FHIR per AWS HealthLake

Un archivio HealthLake dati compatibile con SMART (Substitutable Medical Applications and Reusable Technologies) su FHIR consente l'accesso a SMART su applicazioni conformi a FHIR. HealthLake l'accesso ai dati avviene autenticando e autorizzando le richieste utilizzando un server di autorizzazione di terze parti. Quindi, invece di gestire le credenziali degli utenti tramite AWS Identity and Access Management, lo fai utilizzando un server di autorizzazione conforme a SMART on FHIR.

> **Note**
>
> HealthLake supporta SMART nelle versioni 1.0 e 2.0 di FHIR. Per ulteriori informazioni su questi framework, consulta SMART App Launch nella documentazione di FHIR R4. HealthLake gli archivi di dati supportano i seguenti framework di autenticazione e autorizzazione per SMART su richieste FHIR:
>
> - OpenID (AuthN): per autenticare la persona o l'applicazione client indica chi (o cosa) dichiara di essere.

- OAuth 2.0 (AuthZ): per autorizzare le risorse FHIR presenti nell'archivio HealthLake dati su cui una richiesta autenticata può leggere o scrivere. Questo è definito dagli ambiti configurati nel server di autorizzazione.

È possibile creare un archivio dati SMART su FHIR abilitato utilizzando o. AWS CLI AWS SDKs Per ulteriori informazioni, consulta Creazione di un archivio HealthLake dati.

Argomenti

- Guida introduttiva a SMART su FHIR
- HealthLake requisiti di autenticazione per SMART on FHIR
- SMART su ambiti FHIR OAuth 2.0 supportati da HealthLake
- Validazione dei token utilizzando AWS Lambda
- Utilizzo di un'autorizzazione granulare con un data store abilitato SMART on FHIR HealthLake
- Recupero del documento SMART su FHIR Discovery
- Esecuzione di una richiesta API REST FHIR su un data store abilitato per Smart HealthLake

Guida introduttiva a SMART su FHIR

I seguenti argomenti descrivono come iniziare a utilizzare l'autorizzazione SMART on FHIR per. AWS HealthLake Includono le risorse da fornire nel proprio AWS account, la creazione di un HealthLake data store abilitato per SMART on FHIR e un esempio di come un'applicazione client SMART on FHIR interagisce con un server di autorizzazione e un archivio dati. HealthLake

Argomenti

- Configurazione delle risorse per SMART su FHIR
- Flusso di lavoro delle applicazioni client per SMART su FHIR

Configurazione delle risorse per SMART su FHIR

I passaggi seguenti definiscono in che modo vengono gestite le richieste SMART on FHIR HealthLake e le risorse necessarie per il loro successo. I seguenti elementi interagiscono in un flusso di lavoro per creare una richiesta SMART on FHIR:

- L'utente finale: in genere, un paziente o un medico che utilizza un'applicazione SMART on FHIR di terze parti per accedere ai dati in un archivio dati. HealthLake
- L'applicazione SMART on FHIR (denominata applicazione client): un'applicazione che desidera accedere ai dati presenti nell'archivio dati. HealthLake
- Il server di autorizzazione: un server conforme a OpenID Connect in grado di autenticare gli utenti ed emettere token di accesso.
- L'archivio HealthLake dati: un data store abilitato HealthLake per SMART on FHIR che utilizza una funzione Lambda per rispondere alle richieste FHIR REST che forniscono un token bearer.

Affinché questi elementi funzionino insieme, è necessario creare le seguenti risorse.

> **Note**
>
> Ti consigliamo di creare il tuo HealthLake data store abilitato per SMART on FHIR dopo aver configurato il server di autorizzazione, definito gli ambiti necessari su di esso e creato una AWS Lambda funzione per gestire l'introspezione dei token.

1. Configura un endpoint del server di autorizzazione

Per utilizzare il framework SMART on FHIR è necessario configurare un server di autorizzazione di terze parti in grado di convalidare le richieste FHIR REST effettuate su un archivio dati. Per ulteriori informazioni, consulta HealthLake requisiti di autenticazione per SMART on FHIR.

2. Definisci gli ambiti sul tuo server di autorizzazione per controllare i livelli di accesso all'archivio dati HealthLake

Il framework SMART on FHIR utilizza OAuth gli ambiti per determinare a quali risorse FHIR ha accesso una richiesta autenticata e in che misura. La definizione degli ambiti è un modo per progettare con privilegi minimi. Per ulteriori informazioni, consulta SMART su ambiti FHIR OAuth 2.0 supportati da HealthLake.

3. Imposta una funzione in grado di eseguire l' AWS Lambda introspezione dei token

Una richiesta FHIR REST inviata dall'applicazione client su un data store abilitato per SMART on FHIR contiene un JSON Web Token (JWT). Per ulteriori informazioni, vedere Decodifica di un JWT.

4. Crea un archivio dati compatibile con SMART su FHIR HealthLake

Per creare un HealthLake data store SMART su FHIR è necessario fornire un. `IdentityProviderConfiguration` Per ulteriori informazioni, consulta Creazione di un archivio HealthLake dati.

Flusso di lavoro delle applicazioni client per SMART su FHIR

La sezione seguente spiega come avviare un'applicazione client ed effettuare con successo una richiesta FHIR REST su un HealthLake data store nel contesto di SMART su FHIR.

1. Effettua una **`GET`** richiesta a Wonderful Uniform Resource Identifier utilizzando l'applicazione client

Un'applicazione client abilitata a SMART deve effettuare una `GET` richiesta per trovare gli endpoint di autorizzazione del HealthLake data store. Questa operazione viene eseguita tramite una richiesta URI (Known Uniform Resource Identifier). Per ulteriori informazioni, consulta Recupero del documento SMART su FHIR Discovery.

2. Richiesta di accesso e ambiti

L'applicazione client utilizza l'endpoint di autorizzazione del server di autorizzazione, in modo che l'utente possa accedere. Questo processo autentica l'utente. Gli ambiti vengono utilizzati per definire a quali risorse FHIR nell'archivio HealthLake dati può accedere un'applicazione client. Per ulteriori informazioni, consulta SMART su ambiti FHIR OAuth 2.0 supportati da HealthLake.

3. Token di accesso

Ora che l'utente è stato autenticato, un'applicazione client riceve un token di accesso JWT dal server di autorizzazione. Questo token viene fornito quando l'applicazione client invia una richiesta FHIR REST a. HealthLake Per ulteriori informazioni, consulta Convalida del token.

4. Effettua una richiesta API REST FHIR su SMART su un data store abilitato a FHIR HealthLake

L'applicazione client può ora inviare una richiesta API REST FHIR a un endpoint del HealthLake data store utilizzando il token di accesso fornito dal server di autorizzazione. Per ulteriori informazioni, consulta Esecuzione di una richiesta API REST FHIR su un data store abilitato per Smart HealthLake.

5. Convalida il token di accesso JWT

Per convalidare il token di accesso inviato nella richiesta FHIR REST, usa una funzione Lambda. Per ulteriori informazioni, consulta Validazione dei token utilizzando AWS Lambda.

HealthLake requisiti di autenticazione per SMART on FHIR

Per accedere alle risorse FHIR in un HealthLake data store abilitato per SMART on FHIR, un'applicazione client deve essere autorizzata da un server di autorizzazione OAuth conforme alla versione 2.0 e presentare un token OAuth Bearer come parte di una richiesta API REST FHIR. Per trovare l'endpoint del server di autorizzazione, utilizzate lo SMART on FHIR Discovery Document tramite un Uniform Resource Identifier HealthLake . `Well-Known` Per ulteriori informazioni su questo processo, consultare Recupero del documento SMART su FHIR Discovery.

Quando si crea un HealthLake data store SMART on FHIR, è necessario definire l'endpoint del server di autorizzazione e l'endpoint del token nell'elemento della `metadata` richiesta. `CreateFHIRDatastore` Per ulteriori informazioni sulla definizione dell'`metadata`elemento, vedere. Creazione di un archivio HealthLake dati

Utilizzando gli endpoint del server di autorizzazione, l'applicazione client autenticherà un utente con il servizio di autorizzazione. Una volta autorizzato e autenticato, un JSON Web Token (JWT) viene generato dal servizio di autorizzazione e passato all'applicazione client. Questo token contiene gli ambiti di risorse FHIR che l'applicazione client può utilizzare, il che a sua volta limita i dati a cui l'utente può accedere. Facoltativamente, se è stato fornito l'ambito di avvio, la risposta conterrà tali dettagli. Per ulteriori informazioni sugli oscilloscopi SMART on FHIR supportati da HealthLake, vedere. SMART su ambiti FHIR OAuth 2.0 supportati da HealthLake

Utilizzando il JWT concesso dal server di autorizzazione, un'applicazione client effettua chiamate API REST FHIR a un archivio dati abilitato per SMART on FHIR. HealthLake Per convalidare e decodificare il JWT, è necessario creare una funzione Lambda. HealthLake richiama questa funzione Lambda per tuo conto quando viene ricevuta una richiesta API REST FHIR. Per vedere un esempio di funzione Lambda di avvio, vedere. Validazione dei token utilizzando AWS Lambda

Elementi del server di autorizzazione necessari per creare un archivio dati compatibile con SMART on FHIR HealthLake

Nella `CreateFHIRDatastore` richiesta, è necessario fornire l'endpoint di autorizzazione e l'endpoint del token come parte dell'`metadata`elemento nell'`IdentityProviderConfiguration`oggetto. Sono necessari sia l'endpoint di autorizzazione che l'endpoint del token. Per vedere un esempio di come questo viene specificato nella `CreateFHIRDatastore` richiesta, vedere. Creazione di un archivio HealthLake dati

Dichiarazioni obbligatorie per completare una richiesta API REST FHIR su un data store abilitato SMART on FHIR HealthLake

Affinché si tratti di una richiesta API REST FHIR valida su un data store abilitato SMART on FHIR, la AWS Lambda funzione deve contenere le seguenti affermazioni. HealthLake

- `nbf`: Reclamo (non prima) - L'attestazione «nbf» (non prima) identifica il momento prima del quale il JWT NON DEVE essere accettato per l'elaborazione. L'elaborazione del reclamo «nbf» richiede che il reclamo corrente date/time DEVE essere successivo o uguale al non precedente date/time elencato nel reclamo «nbf». La funzione Lambda di esempio che forniamo converte `iat` dalla risposta del server in. `nbf`
- `exp`: Richiesta (ora di scadenza) - L'attestazione «exp» (ora di scadenza) identifica l'ora di scadenza in cui o dopo la quale il JWT non deve essere accettato per l'elaborazione.
- `isAuthorized`: Un valore booleano impostato su. `True` Indica che la richiesta è stata autorizzata sul server di autorizzazione.
- `aud`: Reclamo (Audience) - L'attestazione «aud» (audience) identifica i destinatari a cui è destinato il JWT. Deve essere un endpoint di archiviazione dati compatibile HealthLake con SMART on FHIR.
- `scope`: Questo deve essere almeno un ambito relativo alle risorse FHIR. Questo ambito è definito sul server di autorizzazione. Per ulteriori informazioni sugli ambiti relativi alle risorse FHIR accettati da HealthLake, vedere. SMART sugli ambiti di risorse FHIR per HealthLake

SMART su ambiti FHIR OAuth 2.0 supportati da HealthLake

HealthLake utilizza OAuth 2.0 come protocollo di autorizzazione. L'utilizzo di questo protocollo sul server di autorizzazione consente di definire le autorizzazioni dell'archivio HealthLake dati (creazione, lettura, aggiornamento, eliminazione e ricerca) per le risorse FHIR a cui ha accesso un'applicazione client.

Il framework SMART on FHIR definisce un set di ambiti che possono essere richiesti al server di autorizzazione. Ad esempio, un'applicazione client progettata esclusivamente per consentire ai pazienti di visualizzare i risultati di laboratorio o visualizzare i propri dati di contatto dovrebbe essere autorizzata solo a richiedere `read` gli oscilloscopi.

> **Note**
>
> HealthLake fornisce supporto sia per SMART su FHIR V1 che V2, come descritto di seguito. SMART su FHIR `AuthorizationStrategy`è impostato su uno dei tre valori seguenti quando viene creato il data store:
>
> - `SMART_ON_FHIR_V1`— Supporto solo per SMART su FHIR V1, che include le autorizzazioni `read` (lettura/ricerca) e `write` (). create/update/delete
> - `SMART_ON_FHIR`— Supporto per SMART su FIR V1 e V2, che include,`create`, `read` `updatedelete`, e autorizzazioni. `search`
> - `AWS_AUTH`— La strategia di AWS HealthLake autorizzazione predefinita; non affiliata a SMART su FHIR.

Ambito di lancio autonomo

HealthLake supporta l'ambito della modalità di avvio autonoma. `launch/patient`

In modalità di avvio autonoma, un'applicazione client richiede l'accesso ai dati clinici del paziente perché l'utente e il paziente non sono noti all'applicazione client. Pertanto, la richiesta di autorizzazione dell'applicazione client richiede esplicitamente la restituzione dell'ambito del paziente. Una volta completata con successo l'autenticazione, il server di autorizzazione emette un token di accesso contenente l'ambito del paziente di avvio richiesto. Il contesto paziente necessario viene fornito insieme al token di accesso nella risposta del server di autorizzazione.

Ambiti della modalità di avvio supportati

Scope	Description
`launch/patient`	Un parametro in una richiesta di autorizzazione OAuth 2.0 che richiede la restituzione dei dati del paziente nella risposta di autorizzazione.

SMART sugli ambiti di risorse FHIR per HealthLake

HealthLake definisce tre livelli di SMART sugli ambiti di risorse FHIR.

- `patient`gli ambiti garantiscono l'accesso a dati specifici su un singolo paziente.
- `user`gli ambiti garantiscono l'accesso a dati specifici a cui un utente può accedere.

- `system`gli ambiti concedono l'accesso a tutte le risorse FHIR presenti nell' HealthLake archivio dati.

Le sezioni seguenti elencano la sintassi per la costruzione di ambiti di risorse FHIR utilizzando SMART su FHIR V1 o SMART su FHIR V2.

> **Note**
>
> La strategia di autorizzazione SMART on FHIR viene impostata al momento della creazione dell'archivio dati. Per ulteriori informazioni, consulta `AuthorizationStrategy` nella documentazione di riferimento dell'API AWS HealthLake .

Ambiti SMART su FHIR V1 supportati da HealthLake

Quando si utilizza SMART su FHIR V1, segue la sintassi generale per la costruzione degli ambiti di risorse FHIR. HealthLake Per visualizzare l'intero percorso dell'URL nell'esempio seguente, scorri il pulsante Copia.

```
('patient' | 'user' | 'system') '/' (fhir-resource | '*') '.' ('read' | 'write' | '*')
```

SMART on FHIR v1 supportava gli ambiti di autorizzazione

Sintassi dell'ambito	Ambito di esempio	Risultato
`patient/(fhir-resource | '*').('read' | 'write' | '*')`	`patient/AllergyIntolerance.*`	L'applicazione client per i pazienti dispone di accesso in lettura/scrittura a livello di istanza a tutte le allergie registrate.
`user/(fhir-resource | '*').('read' | 'write' | '*')`	`user/Observation.read`	L'applicazione client utente ha accesso a livello di istanza read/write a tutte le osservazioni registrate.

Sintassi dell'ambito	Ambito di esempio	Risultato
`system/('read' | 'write' | *)`	`system/*.*`	L'applicazione client di sistema ha read/write accesso a tutti i dati delle risorse FHIR.

Ambiti SMART su FHIR V2 supportati da HealthLake

Quando si utilizza SMART su FHIR V2, segue la sintassi generale per la costruzione degli ambiti di risorse FHIR. HealthLake Per visualizzare l'intero percorso dell'URL nell'esempio seguente, scorri il pulsante Copia.

```
('patient' | 'user' | 'system') '/' (fhir-resource | '*') '.' ('c' | 'r' | 'u' | 'd' |
 's')
```

> **Note**
>
> Per utilizzare SMART su FHIR V2, è necessario inserire il valore `permission-v2`nella `capabilities` stringa di metadati, che è un membro del `IdentityProviderConfiguration`tipo di dati.
> HealthLake supporta cannocchiali granulari. Per ulteriori informazioni, consulta gli ambiti granulari supportati nella FHIR US Core Implementation Guide.

Ambiti di autorizzazione supportati da SMART on FHIR V2

Sintassi dell'ambito	Esempio di ambito V1	Risultato
`patient/Observation.rs`	`user/Observation.read`	Autorizzazione a leggere e cercare `Observation` risorse per il paziente attuale.
`system/*.cruds`	`system/*.*`	L'applicazione client di sistema ha pieno

Sintassi dell'ambito	Esempio di ambito V1	Risultato
		create/read/update/delete/search accesso a tutti i dati delle risorse FHIR.

Validazione dei token utilizzando AWS Lambda

Quando si crea un archivio dati HealthLake SMART su FHIR abilitato, è necessario fornire l'ARN della funzione AWS Lambda nella `CreateFHIRDatastore` richiesta. L'ARN della funzione Lambda è specificato nell'`IdentityProviderConfiguration`oggetto utilizzando il parametro. `IdpLambdaArn`

È necessario creare la funzione Lambda prima di creare l'archivio dati abilitato per SMART on FHIR. Una volta creato il data store, l'ARN Lambda non può essere modificato. Per visualizzare l'ARN Lambda specificato al momento della creazione del data store, utilizza l'azione API. `DescribeFHIRDatastore`

Affinché una richiesta REST FHIR abbia esito positivo su un data store abilitato SMART on FHIR, la funzione Lambda deve eseguire le seguenti operazioni:

- Restituisce una risposta in meno di 1 secondo all'endpoint del HealthLake data store.
- Decodifica il token di accesso fornito nell'intestazione di autorizzazione della richiesta API REST inviata dall'applicazione client.
- Assegna un ruolo di servizio IAM con autorizzazioni sufficienti per eseguire la richiesta API REST FHIR.
- Le seguenti affermazioni sono necessarie per completare una richiesta API REST FHIR. Per ulteriori informazioni, consulta Richieste necessarie.
  - `nbf`
  - `exp`
  - `isAuthorized`
  - `aud`
  - `scope`

Quando lavori con Lambda, devi creare un ruolo di esecuzione e una policy basata sulle risorse oltre alla funzione Lambda. Il ruolo di esecuzione di una funzione Lambda è un ruolo IAM che concede alla funzione l'autorizzazione ad accedere ai servizi e alle risorse AWS necessari in fase di esecuzione. La policy basata sulle risorse che fornisci deve consentire di richiamare la tua funzione HealthLake per tuo conto.

Le sezioni di questo argomento descrivono una richiesta di esempio da un'applicazione client e una risposta decodificata, i passaggi necessari per creare una funzione AWS Lambda e come creare una politica basata sulle risorse che possa presupporre. HealthLake

- Parte 1: creazione di una funzione Lambda
- Parte 2: creazione di un ruolo HealthLake di servizio utilizzato dalla funzione AWS Lambda
- Parte 3: Aggiornamento del ruolo di esecuzione della funzione Lambda
- Parte 4: aggiunta di una politica delle risorse alla funzione Lambda
- Parte 5: Fornire la concorrenza per la funzione Lambda

Creazione di una funzione AWS Lambda

La funzione Lambda creata in questo argomento viene attivata quando HealthLake riceve una richiesta a un data store abilitato SMART on FHIR. La richiesta proveniente dall'applicazione client contiene una chiamata API REST e un'intestazione di autorizzazione contenente un token di accesso.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
Authorization: Bearer i8hweunweunweofiwweoijewiwe
```

L'esempio della funzione Lambda in questo argomento utilizza Gestione dei segreti AWS per oscurare le credenziali relative al server di autorizzazione. Consigliamo vivamente di non fornire i dettagli di accesso al server di autorizzazione direttamente in una funzione Lambda.

Example convalida di una richiesta FHIR REST contenente un token del portatore di autorizzazione

La funzione Lambda di esempio mostra come convalidare una richiesta FHIR REST inviata a un archivio dati SMART su FHIR abilitato. Per visualizzare step-by-steps le istruzioni su come implementare questa funzione Lambda, vedere. Creazione di una funzione Lambda utilizzando Console di gestione AWS

Se la richiesta API REST FHIR non contiene un endpoint di data store, un token di accesso e un'operazione REST validi, la funzione Lambda avrà esito negativo. Per ulteriori informazioni sugli elementi richiesti del server di autorizzazione, consulta. Richieste necessarie

```
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()
```

Creazione di una funzione Lambda utilizzando Console di gestione AWS

La procedura seguente presuppone che tu abbia già creato il ruolo di servizio che desideri assumere quando gestisci una richiesta API REST FHIR su un data store abilitato HealthLake per SMART on FHIR. Se non hai creato il ruolo di servizio, puoi comunque creare la funzione Lambda. È necessario aggiungere l'ARN del ruolo di servizio prima che la funzione Lambda funzioni. Per ulteriori informazioni sulla creazione di un ruolo di servizio e sulla sua specificazione nella funzione Lambda, consulta Creazione di un ruolo HealthLake di servizio da utilizzare nella funzione AWS Lambda utilizzata per decodificare un JWT

Per creare una funzione Lambda ()Console di gestione AWS

1. Aprire la pagina Funzioni della console Lambda.
2. Scegli Crea funzione.
3. Scegli Crea da zero.
4. In Informazioni di base, inserisci il nome di una funzione. In Runtime scegli un runtime basato su Python.
5. In Execution role (Ruolo di esecuzione), scegli Create a new role with basic Lambda permissions (Crea un nuovo ruolo con le autorizzazioni Lambda di base).

Lambda crea un ruolo di esecuzione che concede alla funzione l'autorizzazione a caricare i log su Amazon. CloudWatch La funzione Lambda assume il ruolo di esecuzione quando si richiama la funzione e utilizza il ruolo di esecuzione per creare credenziali per l'SDK. AWS

6. Scegliete la scheda Codice e aggiungete la funzione Lambda di esempio.

   Se non hai ancora creato il ruolo di servizio per la funzione Lambda da utilizzare, dovrai crearlo prima che la funzione Lambda di esempio funzioni. Per ulteriori informazioni sulla creazione di un ruolo di servizio per la funzione Lambda, vedere. Creazione di un ruolo HealthLake di servizio da utilizzare nella funzione AWS Lambda utilizzata per decodificare 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()
```

Modifica del ruolo di esecuzione di una funzione Lambda

Dopo aver creato la funzione Lambda, è necessario aggiornare il ruolo di esecuzione per includere le autorizzazioni necessarie per chiamare Secrets Manager. In Secrets Manager, ogni segreto creato ha un ARN. Per applicare il privilegio minimo, il ruolo di esecuzione deve avere accesso solo alle risorse necessarie per l'esecuzione della funzione Lambda.

Puoi modificare il ruolo di esecuzione di una funzione Lambda cercandola nella console IAM o scegliendo Configurazione nella console Lambda. Per ulteriori informazioni sulla gestione del ruolo di esecuzione delle funzioni Lambda, consulta. Ruolo di esecuzione Lambda

Example Ruolo di esecuzione della funzione Lambda che concede l'accesso a **`GetSecretValue`**

L'aggiunta dell'azione IAM `GetSecretValue` al ruolo di esecuzione concede l'autorizzazione necessaria per il funzionamento della funzione Lambda di esempio.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:secret-name-DKodTA"
        }
    ]
}
```

A questo punto hai creato una funzione Lambda che può essere utilizzata per convalidare il token di accesso fornito come parte della richiesta FHIR REST inviata al tuo data store abilitato SMART on FHIR.

Creazione di un ruolo HealthLake di servizio da utilizzare nella funzione AWS Lambda utilizzata per decodificare un JWT

Persona: amministratore IAM

Un utente che può aggiungere o rimuovere policy IAM e creare nuove identità IAM.

Ruolo del servizio

Un ruolo di servizio è un ruolo IAM che un servizio assume per eseguire operazioni per tuo conto. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta Create a role to delegate permissions to an Servizio AWS nella Guida per l'utente IAM.

Dopo la decodifica del JSON Web Token (JWT), l'autorizzazione necessaria a Lambda deve restituire anche un ruolo IAM ARN. Questo ruolo deve disporre delle autorizzazioni necessarie per eseguire la richiesta API REST o fallirà a causa di autorizzazioni insufficienti.

Quando si configura una policy personalizzata utilizzando IAM, è meglio concedere le autorizzazioni minime richieste. Per ulteriori informazioni, consulta Applica le autorizzazioni con privilegi minimi nella Guida per l'utente IAM.

La creazione di un ruolo di HealthLake servizio da designare nella funzione di autorizzazione Lambda richiede due passaggi.

- Innanzitutto, è necessario creare una policy IAM. La policy deve specificare l'accesso alle risorse FHIR per le quali sono stati forniti gli ambiti nel server di autorizzazione.
- In secondo luogo, è necessario creare il ruolo di servizio. Quando si crea il ruolo, si designa una relazione di fiducia e si allega la politica creata nella prima fase. La relazione di fiducia viene designata HealthLake come responsabile del servizio. In questo passaggio è necessario specificare un ARN del HealthLake data store e un ID AWS account.

Creazione di una nuova policy IAM

Gli ambiti definiti nel server di autorizzazione determinano a quali risorse FHIR un utente autenticato ha accesso in un HealthLake archivio dati.

La policy IAM che crei può essere personalizzata in base agli ambiti che hai definito.

È possibile definire le seguenti azioni nell'`Action`elemento di una dichiarazione di policy IAM. Per ognuno `Action` nella tabella puoi definire un`Resource types`. In HealthLake un data store è l'unico tipo di risorsa supportato che può essere definito nell'`Resource`elemento di una dichiarazione di policy di autorizzazione IAM.

Le singole risorse FHIR non sono una risorsa che è possibile definire come elemento in una politica di autorizzazione IAM.

Azioni definite da HealthLake

Azioni	Descrizione	Livello di acces	Tipo di risorsa (obbligatorio)
CreateResource	Concede l'autorizzazione a creare una risorsa	Scrittu	ARN dell'archivio dati: arn:aws:healthlake: ::datastore/fhir/ **your-region 111122223333 your-datastore-id**
DeleteResource	Concede l'autorizzazione per eliminare la risorsa	Scrittu	ARN dell'archivio dati: arn:aws:healthlake: ::datastore/fhir/ **your-region 111122223333 your-datastore-id**
ReadResource	Concede l'autorizzazione per leggere la risorsa	Lettur	ARN dell'archivio dati: arn:aws:healthlake: ::datastore/fhir/ **your-region 111122223333 your-datastore-id**
SearchWithGet	Concede l'autorizzazione per cercare risorse con il metodo GET	Lettur	ARN dell'archivio dati: arn:aws:healthlake: ::datastore/fhir/ **your-region 111122223333 your-datastore-id**
SearchWithPost	Concede l'autorizzazione per cercare risorse con il metodo POST	Lettur	ARN dell'archivio dati: arn:aws:healthlake: ::datastore/fhir/ **your-region 111122223333 your-datastore-id**
AvviareFHIRExportJobWithPost	Concede il permesso di iniziare un lavoro in FHIR Export con GET	Scrittu	ARN dell'archivio dati: arn:aws:healthlake: ::datastore/fhir/ **your-region 111122223333 your-datastore-id**
UpdateResource	Concede l'autorizzazione per aggiornare la risorsa	Scrittu	ARN dell'archivio dati: arn:aws:healthlake: ::datastore/fhir/ **your-region 111122223333 your-datastore-id**

Per iniziare, puoi usare. `AmazonHealthLakeFullAccess` Questa politica garantirebbe la lettura, la scrittura, la ricerca e l'esportazione su tutte le risorse FHIR presenti in un archivio dati. Per concedere autorizzazioni di sola lettura su un data store, utilizzare. `AmazonHealthLakeReadOnlyAccess`

Per ulteriori informazioni sulla creazione di una policy personalizzata utilizzando o IAM Console di gestione AWS AWS CLI SDKs, consulta Creating IAM policies nella IAM User Guide.

Creazione di un ruolo di servizio per HealthLake (console IAM)

Utilizzare questa procedura per creare un ruolo di servizio. Quando crei un servizio, dovrai anche designare una policy IAM.

Per creare il ruolo di servizio per HealthLake (console IAM)

1. Accedi Console di gestione AWS e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione della console IAM seleziona Ruoli.
3. Quindi seleziona Create role (Crea ruolo).
4. Nella pagina Seleziona un'entità fiduciaria, scegli Politica di fiducia personalizzata.
5. Successivamente, in Politica di fiducia personalizzata, aggiorna la politica di esempio come segue. Sostituisci **`your-account-id`** con il tuo numero di account e aggiungi l'ARN del data store che desideri utilizzare nei processi di importazione o esportazione.

   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. Quindi, seleziona Successivo.
7. Nella pagina Aggiungi autorizzazioni, scegli la politica che desideri che il HealthLake servizio assuma. Per trovare la tua politica, cercala in Politiche di autorizzazione.
8. Quindi, scegli Allega politica.
9. Quindi nella pagina Nome, rivedi e crea in Nome del ruolo, inserisci un nome.
10. (Facoltativo) Quindi, in Descrizione, aggiungi una breve descrizione del tuo ruolo.
11. Se possibile, specifica un nome del ruolo o un suffisso del nome del ruolo per facilitare l'identificazione dello scopo del ruolo. I nomi dei ruoli devono essere univoci all'interno dell' Account AWS. Non si distinguono per caso. Ad esempio, non è possibile creare ruoli denominati sia **PRODROLE** che **prodrole**. Poiché varie entità possono fare riferimento al ruolo, non è possibile modificare il nome del ruolo dopo averlo creato.
12. Esamina i dettagli del ruolo, quindi scegli Crea ruolo.

Per informazioni su come specificare il ruolo ARN nella funzione Lambda di esempio, vedere. Creazione di una funzione AWS Lambda

Ruolo di esecuzione Lambda

Il ruolo di esecuzione di una funzione Lambda è un ruolo IAM che concede alla funzione il permesso di accedere a AWS servizi e risorse. Questa pagina fornisce informazioni su come creare, visualizzare e gestire il ruolo di esecuzione di una funzione Lambda.

Per impostazione predefinita, Lambda crea un ruolo di esecuzione con autorizzazioni minime quando si crea una nuova funzione Lambda utilizzando. Console di gestione AWS Per gestire le autorizzazioni concesse nel ruolo di esecuzione, consulta Creazione di un ruolo di esecuzione nella console IAM nella Lambda Developer Guide.

La funzione Lambda di esempio fornita in questo argomento utilizza Secrets Manager per oscurare le credenziali del server di autorizzazione.

Come per qualsiasi ruolo IAM creato, è importante seguire le best practice con privilegi minimi. Durante la fase di sviluppo, a volte potresti concedere autorizzazioni oltre a quelle richieste. Prima di pubblicare la funzione nell'ambiente di produzione, una best practice consiste nel modificare la policy in modo da includere solo le autorizzazioni richieste. Per ulteriori informazioni, consulta Apply least-privilege nella IAM User Guide.

Consenti HealthLake di attivare la funzione Lambda

Quindi HealthLake puoi invocare la funzione Lambda per tuo conto, devi fare quanto segue:

- È necessario impostare `IdpLambdaArn` uguale all'ARN della funzione Lambda che si desidera HealthLake richiamare nella richiesta. `CreateFHIRDatastore`
- È necessaria una politica basata sulle risorse che consenta di HealthLake richiamare la funzione Lambda per conto dell'utente.

Quando HealthLake riceve una richiesta API REST FHIR su un data store abilitato SMART on FHIR, ha bisogno delle autorizzazioni per richiamare la funzione Lambda specificata al momento della creazione del data store per conto dell'utente. Per concedere HealthLake l'accesso, utilizzerai una politica basata sulle risorse. Per ulteriori informazioni sulla creazione di una politica basata sulle risorse per una funzione Lambda, consulta Consentire AWS a un servizio di chiamare una funzione Lambda nella Developer Guide.AWS Lambda

Fornire la concorrenza per la funzione Lambda

> **Important**
>
> HealthLake richiede che il tempo di esecuzione massimo per la funzione Lambda sia inferiore a un secondo (1000 millisecondi).
> Se la funzione Lambda supera il limite di tempo di esecuzione, si ottiene un'eccezione. TimeOut

Per evitare che si verifichi questa eccezione, consigliamo di configurare la concorrenza fornita. Allocando la simultaneità fornita prima di un aumento delle chiamate, è possibile assicurarsi che tutte le richieste siano servite da istanze inizializzate con latenza bassa. Per ulteriori informazioni sulla configurazione della concorrenza predisposta, consulta Configuring provisioned concurrency nella Lambda Developer Guide

Per vedere il tempo di esecuzione medio della tua funzione Lambda, utilizza attualmente la pagina Monitoraggio della funzione Lambda sulla console Lambda. Per impostazione predefinita, la console Lambda fornisce un grafico della durata che mostra il tempo medio, minimo e massimo impiegato dal codice della funzione per l'elaborazione di un evento. Per ulteriori informazioni sul monitoraggio delle funzioni Lambda, consulta Monitoring functions in Lambda console Lambda nella Lambda Developer Guide.

Se hai già predisposto la concorrenza per la tua funzione Lambda e desideri monitorarla, consulta Monitoring concurrency nella Lambda Developer Guide.

Utilizzo di un'autorizzazione granulare con un data store abilitato SMART on FHIR HealthLake

Gli ambiti da soli non forniscono la specificità necessaria sui dati a cui un richiedente è autorizzato ad accedere in un data store. L'utilizzo di un'autorizzazione granulare consente un livello di specificità più elevato quando si concede l'accesso a un data store abilitato per SMART on FHIR. HealthLake Per utilizzare un'autorizzazione granulare, imposta `FineGrainedAuthorizationEnabled` uguale a nel parametro della richiesta. `True` `IdentityProviderConfiguration` `CreateFHIRDatastore`

Se hai abilitato l'autorizzazione granulare, il tuo server di autorizzazione restituisce un `fhirUser` ambito insieme al `id_token` token di accesso. Ciò consente di recuperare le informazioni sull'utente dall'applicazione client. L'applicazione client deve trattare il `fhirUser` claim come l'URI di una risorsa FHIR che rappresenta l'utente corrente. Questo valore può essere `Patient`, `Practitioner` o `RelatedPerson`. La risposta del server di autorizzazione include anche un `user/` ambito che definisce a quali dati l'utente può accedere. Questo utilizza la sintassi definita per gli ambiti relativi agli ambiti specifici delle risorse FHIR:

```
user/(fhir-resource | '*').('read' | 'write' | '*')
```

Di seguito sono riportati alcuni esempi di come l'autorizzazione granulare può essere utilizzata per specificare ulteriormente i tipi di risorse FHIR relativi all'accesso ai dati.

- Quando `fhirUser` è un'autorizzazione `Practitioner` dettagliata determina l'insieme di pazienti a cui l'utente può accedere. L'accesso `fhirUser` è consentito solo ai pazienti per i quali il Paziente si rivolge a un medico `fhirUser` generico.

  ```
  Patient.generalPractitioner : [{Reference(Practitioner)}]
  ```

- Quando `fhirUser` è un `Patient` operatore `RelatedPerson` e il paziente a cui si fa riferimento nella richiesta è diverso dall'autorizzazione `fhirUser` dettagliata a cui il paziente richiesto può accedere. `fhirUser` L'accesso è consentito quando esiste una relazione specificata nella risorsa richiesta. `Patient`

```
Patient.link.other : {Reference(Patient|RelatedPerson)}
```

Recupero del documento SMART su FHIR Discovery

SMART definisce un Discovery Document che consente ai clienti di conoscere l'endpoint di autorizzazione URLs e le funzionalità supportate da un HealthLake data store. Queste informazioni aiutano i client a indirizzare le richieste di autorizzazione all'endpoint corretto e a creare richieste di autorizzazione supportate dal HealthLake data store.

Affinché un'applicazione client possa inoltrare correttamente una richiesta FHIR REST HealthLake, deve raccogliere i requisiti di autorizzazione definiti dall' HealthLake archivio dati. Non è necessario un token al portatore (autorizzazione) per il successo di questa richiesta.

Per richiedere il Discovery Document per un HealthLake data store

1. Raccolta HealthLake `region` e `datastoreId` valori. Per ulteriori informazioni, consulta Ottenere le proprietà dell'archivio dati.
2. Costruisci un URL per la richiesta utilizzando i valori raccolti per HealthLake `region` e`datastoreId`. Aggiungi `/.well-known/smart-configuration` all'endpoint dell'URL. Per visualizzare l'intero percorso dell'URL nell'esempio seguente, scorri il pulsante Copia.

   ```
   https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/.well-known/smart-configuration
   ```
3. Invia la richiesta utilizzando il protocollo `GET` di AWS firma Signature Version 4. Per visualizzare l'intero esempio, scorri il pulsante Copia.

   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'
```

Il Discovery Document per il HealthLake data store viene restituito come un blob JSON, in cui puoi trovare l'`authorization_endpointetoken_endpoint`, insieme alle specifiche e alle funzionalità definite per il data store.

```
{
    "authorization_endpoint": "https://oidc.example.com/authorize",
    "token_endpoint": "https://oidc.example.com/oauth/token",
    "capabilities": [
        "launch-ehr",
        "client-public"
    ]
}
```

Entrambi `token_endpoint` sono necessari per avviare un'applicazione client. `authorization_endpoint`

- Endpoint di autorizzazione: l'URL necessario per autorizzare un'applicazione o un utente client.
- Endpoint token: l'endpoint del server di autorizzazione utilizzato dall'applicazione client per comunicare.

Esecuzione di una richiesta API REST FHIR su un data store abilitato per Smart HealthLake

È possibile effettuare richieste API REST FHIR su un data store abilitato per SMART on HealthLake FHIR. L'esempio seguente mostra una richiesta da un'applicazione client contenente un JWT nell'intestazione di autorizzazione e come Lambda deve decodificare la risposta. Dopo che la richiesta dell'applicazione client è stata autorizzata e autenticata, deve ricevere un token portatore dal server di autorizzazione. Utilizza il token bearer nell'intestazione di autorizzazione quando invii una richiesta API REST FHIR su un data store abilitato per SMART on FHIR. HealthLake

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/[ID]
Authorization: Bearer auth-server-provided-bearer-token
```

Poiché è stato trovato un token bearer nell'intestazione di autorizzazione e non è stata rilevata alcuna identità AWS IAM, HealthLake richiama la funzione Lambda specificata quando è stato creato il data store abilitato SMART on HealthLake FHIR. Quando il token viene decodificato correttamente dalla funzione Lambda, viene inviata la seguente risposta di esempio 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
}
```

Supporto FHIR R4 per AWS HealthLake

AWS HealthLake supporta la specifica FHIR R4 per lo scambio di dati sanitari. Le sezioni seguenti forniscono informazioni di supporto su come HealthLake utilizza la specifica FHIR R4 per aiutarti a gestire e cercare le risorse FHIR nel tuo HealthLake archivio dati utilizzando FHIR R4. RESTful APIs

Argomenti

- Dichiarazione di capacità FHIR R4 per AWS HealthLake
- Validazioni del profilo FHIR per HealthLake
- Tipi di risorse supportati da FHIR R4 per HealthLake
- Parametri di ricerca FHIR R4 per HealthLake
- FHIR R4 per $operations HealthLake

Dichiarazione di capacità FHIR R4 per AWS HealthLake

Per trovare le funzionalità (comportamenti) relative al FHIR di un archivio HealthLake dati attivo, è necessario recuperarne la dichiarazione di capacità. La Capability Statement viene utilizzata come dichiarazione della funzionalità effettiva del server o come dichiarazione dell'implementazione del server richiesta o desiderata. L'`capabilities`interazione FHIR recupera informazioni sulle funzionalità dell'archivio HealthLake dati e sulle parti della specifica FHIR supportate. HealthLake convalida i tipi di risorse FHIR in base alla risorsa FHIR R4. `StructureDefinition`

Per ottenere la dichiarazione di capacità per un archivio dati HealthLake

1. Raccolta HealthLake `region` e `datastoreId` valori. Per ulteriori informazioni, consulta Ottenere le proprietà dell'archivio dati.
2. Costruisci un URL per la richiesta utilizzando i valori raccolti per HealthLake `region` e`datastoreId`. Includi anche l'`metadata`elemento FHIR nell'URL. Per visualizzare l'intero percorso dell'URL nell'esempio seguente, scorri il pulsante Copia.

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

3. Inviare la richiesta . L'`capabilities`interazione FHIR utilizza una `GET` richiesta con il protocollo di AWS firma Signature Version 4. L'`curl`esempio seguente ottiene la dichiarazione di capacità per l'archivio HealthLake dati specificato da. `datastoreId` Per visualizzare l'intero esempio, scorri il pulsante Copia.

   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'
   ```

   Riceverai un codice di risposta `200` HTTP e la dichiarazione di capacità per il tuo HealthLake data store. Per ulteriori informazioni, vedere la `CapabilityStatement`documentazione di FHIR R4.

Validazioni del profilo FHIR per HealthLake

AWS HealthLake supporta la specifica FHIR R4 di base. Nella specifica FHIR R4 di base sono inclusi i profili FHIR. I profili vengono utilizzati su un tipo di risorsa FHIR per definire una definizione di tipo di risorsa più specifica utilizzando le and/or estensioni dei vincoli sul tipo di risorsa di base. Ad esempio, un profilo FHIR può identificare campi obbligatori come estensioni e set di valori. Una risorsa può supportare più profili. Tutti gli archivi HealthLake dati supportano l'utilizzo dei profili FHIR.

> **Note**
>
> L'aggiunta di un profilo FHIR non è richiesta quando si aggiungono dati a un archivio HealthLake dati. Se non viene specificato un profilo FHIR quando una risorsa viene aggiunta o aggiornata, la risorsa viene convalidata solo rispetto allo schema FHIR R4 di base. I profili FHIR, a cui si conformano le risorse FHIR, vengono inclusi nelle risorse prima di essere importati. HealthLake Pertanto, i profili FHIR vengono convalidati durante l'importazione. HealthLake

I profili FHIR sono specificati in una guida all'implementazione. Una guida all'implementazione FHIR (IG) è un insieme di istruzioni che descrivono come utilizzare lo standard FHIR per uno scopo specifico. HealthLake convalida i profili FHIR definiti nelle seguenti guide di implementazione.

profili FHIR supportati da AWS HealthLake

Nome	Ve	Guida all'imple mentazione	Funzionalità	Sta Un ori (Ol	Sta Un ori (Vi set ona	Sta Un occ li (Or	Asi Pa (Mu	Asi Pa (Sy	Ca (Ce	Eu (Irl	Europa (Londra)
U.S. Core	3.1	http://hl7.org/fhir/us/core/STU3.1.1/	Predefinita	X	X	X	X	X	X	X	X
U.S. Core	4.0	https://hl7.org/fhir/us/core/STU4/index.html	Supportata	X	X	X	X	X	X	X	X

Nome	Ve	Guida all'imple mentazione	Funzionalità	Sta Un ori (Ol	Sta Un ori (Vi set ona	Sta Un occ li (Or	Asi Pa (Mu	Asi Pa (Sy	Ca (Ce	Eu (Irl	Europa (Londra)
U.S. Core	5.0	https://hl7.org/fhir/us/core/STU5.0.1/index.html	Supportata	X	X	X	X	X	X	X	X
U.S. Core	6.1	https://hl7.org/fhir/us/core/STU6.1/index.html	Supportata	X	X	X	X	X	X	X	X
U.S. Core	7.0	https://hl7.org/fhir/us/core/STU7/	Supportata	X	X	X	X	X	X	X	X
UK Core	2.0	https://simplifier.net/guide/uk-core-implementation-guide-stu2/Home/ProfilesandExtensions/ProfilesIndex?version=2.0.1	Supportata	X	X	X	X			X	X
Bottone blu CARIN	1.1	http://hl7.org/fhir/us/carin-bb/STU1.1/	Predefinita	X	X	X	X	X	X	X	X
Bottone blu CARIN	1.0 2.0 2.1	https://hl7.org/fhir/us/carin-bb/history.html	Supportata	X	X	X	X	X	X	X	X
Da Vinci Payer Data Exchange	1.0	https://hl7.org/fhir/us/davinci-pdex/	Predefinita	X	X	X	X	X	X	X	X

Nome	Ve	Guida all'implementazione	Funzionalità	Sta Uni orie (Ol	Sta Uni orie (Vi set ona	Sta Uni occ li (Or	Asi Pa (Mu	Asi Pa (Sy	Ca (Ce	Eu (Irl	Europa (Londra)
Da Vinci Payer Data Exchange	2.0 2.1	https://hl7.org/fhir/us/davinci-pdex/history.html	Supportata	X	X	X	X	X	X	X	X
Da Vinci Health Record Exchange (HREx)	0.2	https://hl7.org/fhir/us/davinci-hrex/2020Sep/	Predefinita	X	X	X	X	X	X	X	X
DaVinci Piano PDEX Net	1.1	https://hl7.org/fhir/us/davinci-pdex-plan-net/STU1.1/	Predefinita	X	X	X	X	X	X	X	X
DaVinci Piano PDEX Net	1.0	https://hl7.org/fhir/us/davinci-pdex-plan-net/STU1/	Supportata	X	X	X	X	X	X	X	X
DaVinci Formulario farmaceutico statunitense Payer Data Exchange (PDex)	1.1	https://hl7.org/fhir/us/davinci-drug-formulary/STU1.1/	Predefinita	X	X	X	X	X	X	X	X

Nome	Ve	Guida all'imple mentazione	Funzionalità	Sta Un ori (Ol	Sta Un ori (Vi set ona	Sta Un occ li (Or	Asi Pa (Mu	Asi Pa (Sy	Ca (Ce	Eu (Irl	Europa (Londra)
DaVinci Formulario farmaceutico statunitense Payer Data Exchange (PDex)	1.0 2.0 2.1	https://hl7.org/fhir/us/davinci-drug-formulary/history.html	Supportata	X	X	X	X	X	X	X	X
Da Vinci Clinical Data Exchange (CDex)	2.1	https://build.fhir.org/ig/HL7/davinci-ecdx/index.html	Predefinita	X	X	X	X	X	X	X	X
Da Vinci Prior Authorization Support (PAS) FHIR IG	2.1	https://hl7.org/fhir/us/davinci-pas/	Predefinita	X	X	X	X	X	X	X	X
Guida all'implementazione di NCA HEDIS®	0.3	https://www.ncqa.org/resources/hedis-ig-resource-page/	Predefinita	X	X	X	X			X	X

Nome	Ve	Guida all'imple mentazione	Funzionalità	Sta Un ori (Ol	Sta Un ori (Vi set ona	Sta Un occ li (Or	Asi Pa (Mu	Asi Pa (Sy	Ca (Ce	Eu (Irl	Europa (Londra)
Riepilogo internazionale dei pazienti (IPS)	2.0 bal lott	https://hl7.org/fhir/uv/ips/2024Sep/	Predefinita	X	X	X	X	X	X	X	X
Misura di qualità	5.0	https://registry.fhir.org/package/hl7.fhir.us.cqfmeasures%7C5.0.0	Predefinita	X	X	X	X			X	X
Rapporti genomici	3.0	https://build.fhir.org/ig/HL7/genomics-reporting/index.html	Predefinita	X	X	X	X			X	X
Ayushman Bharat Digital Mission (ABDM) della National Health Authority	2.0	https://www.nrces.in/ndhm/fhir/r4/index.html	Predefinita	X	X	X	X			X	X
CA Core+	1.1	https://simplifier.net/ca-core	Supportata						X		

Nome	Ve	Guida all'imple mentazione	Funzionalità	Sta Un ori (Ol	Sta Un ori (Vi set ona	Sta Un occ li (Or	Asi Pa (Mu	Asi Pa (Sy	Ca (Ce	Eu (Irl	Europa (Londra)
CA:eReC Pan-Canad ian e Referral-eConsult	1.1	https://simplifier.net/CA-eReC/~introduction	Supportata						X		
Riepilogo del paziente Edizione canadese - (PS-CA)	2.1	https://simplifier.net/PS-CA-R1/~introduction	Supportata						X		
Ontario Digital Health Drug Repository	4.0	https://simplifier.net/ca-on-dhdr-r4/~introduction	Supportata						X		
Codice AU	1.0	https://hl7.org.au/fhir/core/	Supportata					X			
Gestione pratica Magentus	1.2	https://fhir-versions.dev.geniesolutions.io/1.2.16/downloads.html	Supportata					X			

Convalida dei profili FHIR specificati in una risorsa

Per convalidare un profilo FHIR, aggiungilo all'`profile`elemento delle singole risorse utilizzando l'URL del profilo indicato nella guida all'implementazione.

I profili FHIR vengono convalidati quando aggiungi una nuova risorsa al tuo archivio dati. Per aggiungere una nuova risorsa, è possibile utilizzare l'operazione `StartFHIRImportJob` API, effettuare una `POST` richiesta per aggiungere una nuova risorsa o `PUT` aggiornare una risorsa esistente.

Example— Per vedere a quale profilo FHIR viene fatto riferimento in una risorsa

L'URL del profilo viene aggiunto all'`profile`elemento nella coppia `"meta" : "profile"` chiave-valore. Questa risorsa è stata troncata per motivi di chiarezza.

```
{
    "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— Come fare riferimento a un profilo FHIR supportato non predefinito

Per eseguire la convalida rispetto a un profilo non predefinito supportato, aggiungete l'URL del profilo con versione all'elemento. `meta.profile` L'URL con versione include l'URL del profilo di base seguito da un carattere pipe (`|`) e dal numero di versione. Questa risorsa di esempio è stata troncata per motivi di chiarezza.

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

Puoi anche includere sia l'URL con versione che l'URL del profilo di base. Entrambi i formati sono validi.

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

Tipi di risorse supportati da FHIR R4 per HealthLake

La tabella seguente elenca i tipi di risorse FHIR R4 supportati da. AWS HealthLake Per ulteriori informazioni, vedere Resource Index nella documentazione FHIR R4.

Tipi di risorse FHIR R4 supportati da HealthLake

Account	DetectedIssue	Fattura	Professionista
ActivityDefinition	Dispositivo	Libreria	PractitionerRole
AdverseEvent	DeviceDefinition	Collegamento	Procedura
AllergyIntolerance	DeviceMetric	List	Provenienza
Appuntamento	DeviceUseStatement	Location (Ubicazione)	Questionario
AppointmentResponse	DeviceRequest	Misura	QuestionnaireResponse
AuditEvent - Vedi nota	DiagnosticReport	MeasureReport	RelatedPerson
Binario	DocumentManifest	Media	RequestGroup
BodyStructure	DocumentReference	Farmaco	ResearchStudy
Pacchetto - Vedi nota	EffectEvidenceSynthesis	MedicationAdministration	ResearchSubject

CapabilityStatement	Incontro	MedicationDispense	RiskAssessment
CarePlan	Endpoint	MedicationKnowledge	RiskEvidenceSynthesis
CareTeam	EpisodeOfCare	MedicationRequest	Schedule
ChargeItem	EnrollmentRequest	MedicationStatement	ServiceRequest
ChargeItemDefinition	EnrollmentResponse	MessageHeader	Slot
Richiedi	ExplanationOfBenefit	MolecularSequence	Esemplare
ClaimResponse	FamilyMemberHistory	NutritionOrder	StructureDefinition
Communication	Flag	Osservazione	StructureMap
CommunicationRequest	Obiettivo	OperationOutcome - Vedi nota	Sostanza
Composizione	Gruppo	Organizzazione	SupplyDelivery
ConceptMap	GuidanceResponse	OrganizationAffiliation	SupplyRequest
Condizione	HealthcareService	Parametri - Vedi nota	Operazione
Consenso	ImagingStudy	Paziente	ValueSet
Contratto	Immunizzazione	PaymentNotice	VisionPrescription
Copertura	ImmunizationEvaluation	PaymentReconciliation	VerificationResult - Vedi nota
CoverageEligibilityRequest	ImmunizationRecommendation	Person	
CoverageEligibilityResponse	InsurancePlan	PlanDefinition	

⚠ Specifiche FHIR e HealthLake

- Non è possibile effettuare `GET` o `POST` richiedere con FHIR `OperationOutcome` e tipi di `Parameters` risorse.
- AuditEvent— Una AuditEvent risorsa può essere creata o letta, ma non può essere aggiornata o eliminata.
- Bundle: esistono diversi modi per HealthLake gestire le richieste Bundle. Per ulteriori dettagli, consultare Raggruppamento di risorse FHIR.
- VerificationResult— Questo tipo di risorsa è supportato solo per gli archivi dati creati dopo il 09 dicembre 2023.

Parametri di ricerca FHIR R4 per HealthLake

Usa l'`search`interazione FHIR per cercare un set di risorse FHIR in un archivio HealthLake dati in base ad alcuni criteri di filtro. L'`search`interazione può essere eseguita utilizzando una richiesta `GET` or`POST`. Per le ricerche che coinvolgono informazioni di identificazione personale (PII) o informazioni sanitarie protette (PHI), si consiglia di utilizzare `POST` le richieste, poiché PII e PHI vengono aggiunti come parte del corpo della richiesta e vengono crittografati durante il transito.

Note

L'`search`interazione FHIR descritta in questo capitolo è costruita in conformità allo standard FHIR R4 per lo HL7 scambio di dati sanitari. Poiché è una rappresentazione di un servizio HL7 FHIR, non viene offerto tramite e. AWS CLI AWS SDKs Per ulteriori informazioni, consulta la `search`documentazione dell'API FHIR R4 RESTful .
Puoi anche interrogare gli archivi HealthLake dati con SQL utilizzando Amazon Athena. Per ulteriori informazioni, consulta Integrazione.

HealthLake supporta il seguente sottoinsieme di parametri di ricerca FHIR R4. Per ulteriori informazioni, consulta Parametri di ricerca FHIR R4 per HealthLake.

Tipi di parametri di ricerca supportati

La tabella seguente mostra i tipi di parametri di ricerca supportati in HealthLake.

Tipi di parametri di ricerca supportati

Parametro di ricerca	Description
_id	ID della risorsa (non un URL completo)
_Ultimo aggiornamento	Data ultimo aggiornamento. Il server ha discrezione sulla precisione dei limiti.
_etichetta	Cerca in base a un tag di risorsa.
_profilo	Cerca tutte le risorse contrassegnate con un profilo.
_sicurezza	Cerca sulle etichette di sicurezza applicate a questa risorsa.
_fonte	Cerca da dove proviene la risorsa.
_testo	Cerca nella narrazione della risorsa.
createdAt	Cerca sull'estensione personalizzata CreateDat .

> **Note**
>
> I seguenti parametri di ricerca sono supportati solo per i datastore creati dopo il 09 dicembre 2023: _security, _source, _text, CreateDat.

La tabella seguente mostra esempi di come modificare le stringhe di query in base a tipi di dati specificati per un determinato tipo di risorsa. Per maggiore chiarezza, i caratteri speciali nella colonna degli esempi non sono stati codificati. Per eseguire correttamente una query, assicuratevi che la stringa di query sia stata codificata correttamente.

Esempi di parametri di ricerca

Tipi di parametri di ricerca	Informazioni	Esempi

Numero	Cerca un valore numerico in una risorsa specificata. Si osservano cifre significative. Il numero di cifre significative è specifico in base al valore del parametro di ricerca, esclusi gli zeri iniziali. I prefissi di confronto sono consentiti.	`[parameter]=100`
		
		`[parameter]=1e2`
		
		`[parameter]=lt100`

Data/ DateTime	Cerca una data o un'ora specifica. Il formato previsto è `yyyy-mm-ddThh:mm:ss[Z|(+|-)hh:mm]` ma può variare.
	
	Accetta i seguenti tipi di dati: `datedateTime`,`instant`,`PeriodeTiming`. Per maggiori dettagli sull'utilizzo di questi tipi di dati nelle ricerche, consulta la data nella documentazione dell'API FHIR R4 RESTful .
	
	I prefissi di confronto sono consentiti.
		`[parameter]=eq2013-01-14`
		
		`[parameter]=gt2013-01-14T10:00`
		
		`[parameter]=ne2013-01-14`

Stringa	Cerca una sequenza di caratteri con distinzione tra maiuscole e minuscole.
	
	Supporta entrambi i tipiHumanName . `Address` Per ulteriori dettagli, vedere la voce relativa al tipo di
		`[base]/Patient?given=eve`
		
		`[base]/Patient?given:contains=eve`

Tipi di parametri di ricerca	Informazioni	Esempi
	HumanName dati e le voci relative al tipo di `Address` dati nella documentazione FHIR R4. La ricerca avanzata è supportata tramite `:text` modificatori.	
Token	Cerca una close-to-exact corrispondenza in base a una stringa di caratteri, spesso confrontata con un paio di valori di codice medico. La distinzione tra maiuscole e minuscole è collegata al sistema di codice utilizzato durante la creazione di una query.Le query basate su Subsumption possono aiutare a ridurre i problemi legati alla distinzione tra maiuscole e minuscole. Per chiarezza non è stato codificato. |	`[parameter]=[system]|[code]` : Qui `[system]` si riferisce a un sistema di codifica e si `[code]` riferisce al valore di codice trovato all'interno di quel sistema specifico. `[parameter]=[code]` : Qui il tuo input corrisponderà a un codice o a un sistema. `[parameter]=|[code]` : Qui il tuo input corrisponderà a un codice e la proprietà del sistema non ha un identificatore.

Tipi di parametri di ricerca	Informazioni	Esempi

Composita	Cerca più parametri all'interno di un singolo tipo di risorsa, utilizzando i modificatori e l'operazione$. ,
	
	I prefissi di confronto sono consentiti.
		`/Patient?language=FR,NL&language=EN`
		
		`Observation?component-code-value-quantity=http://loinc.org|8480-6$lt60`
		
		`[base]/Group?characteristic-value=gender$mixed`

Quantità	Cerca un numero, un sistema e un codice come valori. È richiesto un numero, ma il sistema e il codice sono facoltativi. In base al tipo di dati sulla quantità. Per ulteriori dettagli, vedere Quantity nella documentazione FHIR R4.
	
	Utilizza la seguente sintassi presunta `[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`

Documentazione di riferimento	Cerca riferimenti ad altre risorse.	`[base]/Observation?subject=Patient/23`
		
		`test`

Tipi di parametri di ricerca	Informazioni	Esempi
URI	Cerca una stringa di caratteri che identifichi in modo inequivocabile una particolare risorsa.	`[base]/ValueSet?url=http://acme.org/fhir/ValueSet/123`
Speciale	Ricerche basate su estensioni NLP mediche integrate.	

Parametri di ricerca avanzati supportati da HealthLake

HealthLake supporta i seguenti parametri di ricerca avanzata.

Nome	Descrizione	Esempio	Funzionalità
`_include`	Utilizzato per richiedere la restituzione di risorse aggiuntive in una richiesta di ricerca. Restituisce risorse a cui fa riferimento l'istanza della risorsa di destinazione.	`Encounter?_include=Encounter:subject`	
`_revinclude`	Utilizzato per richiedere la restituzione di risorse aggiuntive in una richiesta di ricerca. Restituisce risorse che fanno riferimento all'istanza della risorsa principale.	`Patient?_id=` **`patient-identifier`** `&_revinclude=Encounter:patient`	
`_summary`	Il riepilogo può essere utilizzato per richiedere un sottoinsieme della risorsa.	`Patient?_summary=text`	Sono supportati i seguenti parametri di riepilogo:`_summary=true` ,`_summary=false` ,`_summary=text` ,`_summary=data` .

Nome	Descrizione	Esempio	Funzionalità
`_elemen`	Richiedi la restituzione di un set specifico di elementi come parte di una risorsa nei risultati della ricerca.	`Patient?_elements=identifier,active,link`	
`_total`	Restituisce il numero di risorse che corrispondono ai parametri di ricerca.	`Patient?_total=accurate`	Support`_total=accurate` ,`_total=none` .
`_sort`	Indica l'ordinamento dei risultati di ricerca restituiti utilizzando un elenco separato da virgole. Il `-` prefisso può essere utilizzato per qualsiasi regola di ordinamento nell'elenco separato da virgole per indicare l'ordine decrescente.	`Observation?_sort=status,-date`	Supporta l'ordinamento per campi con tipi`Number, String, Quantity, Token, URI, Reference` . L'ordinamento per `Date` è supportato solo per gli archivi dati creati dopo il 09 dicembre 2023. Supporta fino a 5 regole di ordinamento.
`_count`	Controlla quante risorse vengono restituite per pagina del pacchetto di ricerca.	`Patient?_count=100`	La dimensione massima della pagina è 100.
`chainin`	Elementi di ricerca delle risorse referenziate. `.`Indirizza la ricerca concatenata all'elemento all'interno della risorsa referenziata.	`DiagnosticReport?subject:Patient.name=peter`	
`reverse chainin (_has)`	Cerca una risorsa in base agli elementi delle risorse che le fanno riferimento.	`Patient?_has:Observation:patient:code=1234-5`	

`_include`

L'utilizzo `_include` in una query di ricerca consente di restituire anche risorse FHIR aggiuntive specificate. `_include`Da utilizzare per includere risorse collegate in avanti.

Example— Da utilizzare _include per trovare i pazienti o il gruppo di pazienti a cui è stata diagnosticata la tosse

È possibile eseguire la ricerca in base al tipo di `Condition` risorsa specificando il codice diagnostico per la tosse e quindi utilizzando `_include` specificare che si desidera che venga restituita anche la `subject` diagnosi. Nel tipo di `Condition` risorsa `subject` si riferisce al tipo di risorsa per il paziente o al tipo di risorsa del gruppo.

Per maggiore chiarezza, i caratteri speciali dell'esempio non sono stati codificati. Per eseguire correttamente una query, assicuratevi che la stringa di query sia stata codificata correttamente.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
Condition?code=49727002&_include=Condition:subject
```

`_include:iterate`Modificatore

Il `_include:iterate` modificatore consente l'inclusione ricorsiva delle risorse referenziate su due livelli. Ad esempio,

```
GET /ServiceRequest?
identifier=025C0931195&_include=ServiceRequest:requester&_include:iterate=PractitionerRole:prac
```

restituirà il ServiceRequest, è associato PractitionerRole (tramite il riferimento del richiedente) e quindi includerà in modo ricorsivo il Practitioner a cui fa riferimento. PractitionerRole Questo modificatore è disponibile per tutti i tipi di risorse in. HealthLake

`_include=*`Modificatore

Il `_include=*` modificatore è un jolly che include automaticamente tutte le risorse a cui fanno riferimento direttamente i risultati della ricerca. Ad esempio,

```
GET /ServiceRequest?specimen.accession=12345&_include=*
```

restituirà la corrispondenza ServiceRequest insieme a tutte le risorse a cui fa riferimento (come Patient, Practitioner, Specimen, ecc.) senza dover specificare ogni percorso di riferimento singolarmente. Questo modificatore è disponibile per tutti i tipi di risorse in. HealthLake

`_revinclude`

L'utilizzo `_revinclude` in una query di ricerca consente di restituire anche risorse FHIR aggiuntive specificate. `_revinclude`Da utilizzare per includere risorse collegate all'indietro.

Example— Da utilizzare _revinclude per includere tipi di risorse correlate all'incontro e all'osservazione collegate a un paziente specifico

Per effettuare questa ricerca, è necessario innanzitutto definire la persona `Patient` specificando il suo identificatore nel parametro di `_id` ricerca. Quindi è necessario specificare risorse FHIR aggiuntive utilizzando la struttura e. `Encounter:patient` `Observation:patient`

Per maggiore chiarezza, i caratteri speciali dell'esempio non sono stati codificati. Per eseguire correttamente una query, assicuratevi che la stringa di query sia stata codificata correttamente.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
Patient?_id=patient-
identifier&_revinclude=Encounter:patient&_revinclude=Observation:patient
```

`_summary`

L'utilizzo `_summary` in una query di ricerca consente all'utente di richiedere un sottoinsieme della risorsa FHIR. Può contenere uno dei seguenti valori:. `true, text, data, false` Qualsiasi altro valore verrà considerato non valido. Le risorse restituite verranno contrassegnate con `'SUBSETTED'` meta.tag, per indicare che le risorse sono incomplete.

- `true`: Restituisce tutti gli elementi supportati contrassegnati come «riepilogo» nella definizione di base delle risorse.
- `text`: restituisce solo gli elementi 'text', 'id', 'meta' e solo gli elementi obbligatori di primo livello.
- `data`: restituisce tutte le parti tranne l'elemento 'text'.
- `false`: restituisce tutte le parti delle risorse

In una singola richiesta di ricerca, `_summary=text` non può essere combinato con i `_include` nostri parametri `_revinclude` di ricerca.

Example— Ottieni l'elemento «testuale» delle risorse per i pazienti in un archivio dati.

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

`_elements`

L'utilizzo `_elements` in una query di ricerca consente di richiedere elementi di risorse FHIR specifici. Le risorse restituite verranno contrassegnate con `'SUBSETTED'` meta.tag, per indicare che le risorse sono incomplete.

Il `_elements` parametro è costituito da un elenco separato da virgole di nomi di elementi di base, come gli elementi definiti al livello principale della risorsa. Devono essere restituiti solo gli elementi elencati. Se i valori dei `_elements` parametri contengono elementi non validi, il server li ignorerà e restituirà elementi obbligatori e elementi validi.

`_elements`non sarà applicabile alle risorse incluse (risorse restituite la cui modalità di ricerca è`include`).

In una singola richiesta di ricerca, `_elements` non può essere combinato con i parametri `_summary` di ricerca.

Example— Ottieni elementi «identificativi», «attivi» e «link» delle risorse per i pazienti nel tuo archivio HealthLake dati.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?
_elements=identifier,active,link
```

`_total`

L'utilizzo `_total` in una query di ricerca restituirà il numero di risorse che corrispondono ai parametri di ricerca richiesti. HealthLake restituirà il numero totale di risorse corrispondenti (risorse restituite la cui modalità di ricerca è`match`) nella risposta `Bundle.total` di ricerca.

`_total`supporta i valori dei none parametri`accurate`,. `_total=estimate`non è supportato. Tutti gli altri valori verranno considerati non validi. `_total`non è applicabile alle risorse incluse (risorse restituite la cui modalità di ricerca è`include`).

Example— Ottieni il numero totale di risorse per i pazienti in un archivio dati:

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

`_sort`

L'utilizzo `_sort` nella query di ricerca ordina i risultati in un ordine specifico. I risultati vengono ordinati in base all'elenco di regole di ordinamento separate da virgole in ordine di priorità. Le regole di ordinamento devono essere parametri di ricerca validi. Tutti gli altri valori verranno considerati non validi.

In una singola richiesta di ricerca, puoi utilizzare fino a 5 parametri di ricerca di ordinamento. Facoltativamente, puoi utilizzare un `-` prefisso per indicare l'ordine decrescente. Per impostazione predefinita, il server ordinerà in ordine crescente.

I tipi di parametri di ricerca di ordinamento supportati sono:`Number, String, Date, Quantity, Token, URI, Reference`. Se un parametro di ricerca si riferisce a un elemento annidato, questo parametro di ricerca non è supportato per l'ordinamento. Ad esempio, la ricerca in base al «nome» del tipo di risorsa Patient si riferisce a Patient.name. L'elemento con tipo di HumanName dati è considerato annidato. Pertanto, l'ordinamento delle risorse del paziente per «nome» non è supportato.

Example— Ottieni le risorse per i pazienti in un archivio dati e ordinale per data di nascita in ordine crescente:

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

`_count`

Il parametro `_count` è definito come un'istruzione al server relativa a quante risorse devono essere restituite in una singola pagina.

La dimensione massima della pagina è 100. Qualsiasi valore superiore a 100 non è valido. `_count=0`non è supportato.

Example— Cerca la risorsa Patient e imposta la dimensione della pagina di ricerca su 25:

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?_count=25
```

Chaining and Reverse Chaining(_has)

Il concatenamento e il concatenamento inverso in FHIR offrono un modo più efficiente e compatto per ottenere dati interconnessi, riducendo la necessità di più query separate e rendendo il recupero dei dati più comodo per sviluppatori e utenti.

Se un livello di ricorsione restituisce più di 100 risultati, HealthLake restituirà 4xx per proteggere l'archivio dati dal sovraccarico e dalla causa di più impaginazioni.

Example— Concatenamento: ottiene tutto ciò DiagnosticReport che si riferisce a un paziente il cui nome del paziente è peter.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/DiagnosticReport?
subject:Patient.name=peter
```

Example— Reverse Chaining - Get Patient Resources, dove la risorsa relativa al paziente viene indicata da almeno un'osservazione, dove l'osservazione ha il codice 1234 e dove l'osservazione si riferisce alla risorsa del paziente nel parametro di ricerca del paziente.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?
_has:Observation:patient:code=1234
```

Modificatori di ricerca supportati

I modificatori di ricerca vengono utilizzati con i campi basati su stringhe. Tutti i modificatori di ricerca utilizzano una logica basata su booleani. HealthLake Ad esempio, è possibile specificare di `:contains` specificare che il campo stringa più grande deve includere una stringa piccola per poterla includere nei risultati della ricerca.

Modificatori di ricerca supportati

Modificatore di ricerca	Tipo
: mancante	Tutti i parametri tranne `Composite`
: esatto	Stringa
:contiene	Stringa
:non	Token
:testo	Token
:identificatore	Documentazione di riferimento
: sotto	URI

Comparatori di ricerca supportati

È possibile utilizzare i comparatori di ricerca per controllare la natura della corrispondenza in una ricerca. È possibile utilizzare i comparatori durante la ricerca nei campi relativi a numeri, date e quantità. La tabella seguente elenca i comparatori di ricerca e le relative definizioni supportati da. HealthLake

Comparatori di ricerca supportati

Comparatore di ricerca	Description
eq	Il valore del parametro nella risorsa è uguale al valore fornito.
uno	Il valore del parametro nella risorsa non è uguale al valore fornito.
gt	Il valore del parametro nella risorsa è maggiore del valore fornito.
lt	Il valore del parametro nella risorsa è inferiore al valore fornito.
età	Il valore del parametro nella risorsa è maggiore o uguale al valore fornito.
le	Il valore del parametro nella risorsa è inferiore o uguale al valore fornito.
come	Il valore del parametro nella risorsa inizia dopo il valore fornito.
eb	Il valore del parametro nella risorsa termina prima del valore fornito.

Parametri di ricerca FHIR non supportati da HealthLake

HealthLake supporta tutti i parametri di ricerca FHIR ad eccezione di quelli elencati nella tabella seguente. Per un elenco completo dei parametri di ricerca FHIR, consultate il registro dei parametri di ricerca FHIR.

Parametri di ricerca non supportati

Composizione del pacchetto	Ubicazione: vicino
Identificatore del pacchetto	C onsent-source-reference
Messaggio del pacchetto	Paziente a contratto
Tipo di pacchetto	Contenuto delle risorse
Timestamp del pacchetto	Interrogazione delle risorse

FHIR R4 per **`$operations`** HealthLake

Le operazioni FHIR $ (chiamate anche «operazioni in dollari») sono funzioni speciali lato server che vanno oltre le operazioni CRUD standard (`Create`,, `ReadUpdate`,`Delete`) previste dalle specifiche FHIR. Queste operazioni sono identificate dal prefisso «$» e consentono l'elaborazione complessa, la trasformazione dei dati e operazioni di massa che sarebbero difficili o inefficienti da eseguire utilizzando chiamate API REST standard. $ Le operazioni possono essere richiamate a livello di sistema, a livello di tipo di risorsa o su istanze di risorse specifiche, fornendo modi flessibili per interagire con i dati FHIR. AWS HealthLake supporta più FHIR. `$operations` Si prega di fare riferimento a ogni singola pagina di seguito per ulteriori dettagli.

Argomenti

- Funzionamento FHIR R4 $attribution-status per HealthLake
- Eliminazione dei tipi di risorse con $bulk-delete
- Operazione $bulk-member-match per HealthLake
- Funzionamento FHIR R4 $confirm-attribution-list per HealthLake
- Funzionamento FHIR R4 $davinci-data-export per HealthLake
- Generazione di documenti clinici con $document
- Rimozione permanente di risorse con $erase
- Acquisizione dei dati dei pazienti con Patient/$everything
- Recupero dei codici con ValueSet $expand
- Esportazione di HealthLake dati con FHIR $export
- $inquireOperazione FHIR per HealthLake

- Recupero dei dettagli concettuali con $lookup
- $member-addoperazione per HealthLake
- $member-matchoperazione per HealthLake
- $member-removeoperazione per HealthLake
- Eliminazione delle risorse del comparto paziente con $purge
- $questionnaire-packageOperazione FHIR per HealthLake
- $submitOperazione FHIR per HealthLake
- Convalida delle risorse FHIR con $validate

Funzionamento FHIR R4 **`$attribution-status`** per HealthLake

Recupera lo stato di attribuzione di un membro specifico, restituendo un pacchetto contenente tutte le risorse di attribuzione relative al paziente. Questa operazione fa parte dell'implementazione FHIR Member Attribution (ATR) List IG 2.1.0.

Endpoint

```
POST [base]/Group/[id]/$attribution-status
```

Parametri della richiesta

L'operazione accetta i seguenti parametri opzionali:

Parametro	Tipo	Description
memberId	Identificatore	Il MemberId membro per il quale è richiesto lo stato di attribuzione
Riferimento per il paziente	Documentazione di riferimento	Riferimento alla risorsa per i pazienti nel sistema del produttore

> **Note**
>
> `memberId`O `patientReference` può essere fornito, o entrambi a scopo di convalida.

Richiesta di esempio

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "memberId",
      "valueIdentifier": {
        "system": "http://example.org",
        "value": "MBR123456789"
      }
    },
    {
      "name": "patientReference",
      "valueReference": {
        "reference": "Patient/patient-123",
        "display": "John Doe"
      }
    }
  ]
}
```

Risposta

Restituisce un pacchetto contenente risorse di attribuzione relative al paziente:

Risorsa	Cardinalità	Location (Ubicazione)
Paziente	1..1	Group.Member.Entity
Copertura	0.1..	group.member.extension: CoverageReference
Organization/Practitioner/PractitionerRole	0.1..	group.member.extension: AttributedProvider
Qualsiasi risorsa	0.1..	group.member.extension: dati associati

Risposta di esempio

```
{
```

```
  "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
      }
    }
```

```
  ]
}
```

Gestione errori

L'operazione gestisce le seguenti condizioni di errore:

Errore	Stato HTTP	Description
Richiesta di operazione non valida	400	Parametri o struttura della richiesta non conformi
Risorsa di gruppo non trovata	404	L'ID di gruppo specificato non esiste
Risorsa per il paziente non trovata	404	Il riferimento del paziente specificato non esiste

Autorizzazione e sicurezza

Requisiti di SMART

I clienti devono disporre dei privilegi appropriati per leggere le risorse del Gruppo e le relative risorse di attribuzione

I meccanismi di autorizzazione FHIR standard si applicano a tutte le operazioni

Eliminazione dei tipi di risorse con **`$bulk-delete`**

AWS HealthLake supporta l'`$bulk-delete`operazione, abilitando l'eliminazione di tutte le risorse di un tipo specifico all'interno di un datastore. Questa operazione è particolarmente utile quando è necessario:

- Effettuare verifiche e pulizie stagionali
- Gestisci il ciclo di vita dei dati su larga scala
- Rimuovi tipi di risorse specifici
- Rispetta le politiche di conservazione dei dati

Utilizzo

L'`$bulk-delete`operazione può essere richiamata utilizzando i metodi POST:

```
POST [base]/[ResourceType]/$bulk-delete?isHardDelete=false&deleteAuditEvent=true
```

Parameters

Parametro	Tipo	Obbligatorio	Predefinita	Description
`isHardDelete`	booleano	No	false	Se impostato su true, rimuove definitivamente le risorse dallo storage
`deleteAuditEvent`	booleano	No	true	Se impostato su true, elimina gli eventi di controllo associati
`_since`	stringa	No	Ora di creazione del datastore	Una volta inserito, seleziona l'orario limite iniziale per trovare le risorse in base all'ora dell'ultima modifica. Non può essere utilizzato con start o end
`start`	stringa	No	Ora di creazione del datastore	Una volta inserito, seleziona l'orario limite per la ricerca delle risorse in base all'ora dell'ultima modifica. Può essere usato con fine
`end`	stringa	No	Ora di invio del lavoro	Una volta inserito, seleziona l'orario limite finale per trovare le risorse in base all'ora dell'ultima modifica

Esempi

Richiesta di esempio

```
POST [base]/Observation/$bulk-delete?isHardDelete=false
```

Risposta di esempio

```
{
      "jobId": "jobId",
      "jobStatus": "SUBMITTED"
    }
```

Stato di un processo

Per verificare lo stato di un processo di eliminazione in blocco:

```
GET [base]/$bulk-delete/[jobId]
```

L'operazione restituisce informazioni sullo stato del lavoro:

```
{
      "datastoreId": "datastoreId",
      "jobId": "jobId",
      "status": "COMPLETED",
      "submittedTime": "2025-10-09T15:09:51.336Z"
    }
```

Comportamento

L'`$bulk-delete`operazione:

1. Processi in modo asincrono per gestire grandi volumi di risorse
2. Mantiene le transazioni ACID per l'integrità dei dati
3. Fornisce il monitoraggio dello stato del lavoro con il conteggio delle eliminazioni delle risorse
4. Supporta sia la modalità di cancellazione temporanea che quella definitiva
5. Include una registrazione di controllo completa delle attività di eliminazione
6. Consente l'eliminazione selettiva delle versioni storiche e degli eventi di controllo

Registrazione di audit

Le `$bulk-delete` operazioni vengono registrate come Start FHIRBulk DeleteJob e Descrivi FHIRBulk DeleteJob con informazioni dettagliate sull'operazione.

Limitazioni

- Se `isHardDelete` è impostata su true, le risorse eliminate definitivamente non verranno visualizzate nei risultati di ricerca o nelle query. `_history`
- Le risorse eliminate tramite questa operazione potrebbero essere temporaneamente inaccessibili durante l'elaborazione
- La misurazione dello storage viene regolata solo in base alle versioni storiche: deleteVersionHistory =false non aggiusterà lo storage del datastore

Operazione **`$bulk-member-match per`** HealthLake

AWS HealthLake supporta l'`$bulk-member-match`operazione per l'elaborazione asincrona delle richieste di abbinamento di più membri. Questa operazione consente alle organizzazioni sanitarie di abbinare in modo efficiente gli identificatori univoci di centinaia di membri in diversi sistemi sanitari utilizzando informazioni demografiche e di copertura in un'unica richiesta di massa. Questa funzionalità è essenziale per lo scambio di dati payer-to-payer su larga scala, le transizioni tra i membri e i requisiti di conformità CMS e segue le specifiche FHIR.

> **Note**
>
> L'`$bulk-member-match`operazione si basa su una specifica FHIR sottostante che è attualmente sperimentale e soggetta a modifiche. Man mano che le specifiche si evolvono, il comportamento e l'interfaccia di questa API verranno aggiornati di conseguenza. Si consiglia agli sviluppatori di monitorare le note di HealthLake rilascio di AWS e gli aggiornamenti delle specifiche FHIR pertinenti per rimanere informati su eventuali modifiche che potrebbero influire sulle loro integrazioni.

Questa operazione è particolarmente utile quando è necessario:

- Elaborare l'abbinamento dei membri su larga scala durante i periodi di iscrizione aperti
- Facilita le transizioni di massa dei membri tra i paganti
- Supporta i requisiti di scambio di dati di conformità CMS su larga scala
- Abbina in modo efficiente le coorti di membri attraverso le reti sanitarie
- Riduci al minimo le chiamate API e migliora l'efficienza operativa per scenari di abbinamento ad alto volume

Utilizzo

L'`$bulk-member-match`operazione è un'operazione asincrona richiamata sulle risorse del gruppo utilizzando il metodo POST:

```
POST [base]/Group/$bulk-member-match
```

Dopo aver inviato una richiesta di corrispondenza in blocco, puoi verificare lo stato del lavoro utilizzando:

```
GET [base]/$bulk-member-match-status/{jobId}
```

Parametri supportati

HealthLake supporta i seguenti parametri FHIR: `$bulk-member-match`

Parametro	Tipo	Campo obbligatorio	Description
`MemberPatient`	Paziente	Sì	Risorsa per il paziente contenente informazioni demografiche relative al membro da abbinare.
`CoverageToMatch`	Copertura	Sì	Risorsa di copertura che verrà utilizzata per il confronto con i record esistenti.
`CoverageToLink`	Copertura	No	Risorsa di copertura da collegare durante il processo di abbinamento.
`Consent`	Consenso	Sì	Viene inoltre memorizzata la risorsa di consenso a fini di autorizzazione. Questa operazione è diversa dalla singola `$member-match` operazione in cui non è richiesto il consenso.

Richiesta POST per inviare in blocco un'offerta di lavoro abbinata a un membro

L'esempio seguente mostra una richiesta POST per inviare un lavoro collettivo riservato ai membri. Ogni membro è racchiuso in un `MemberBundle` parametro contenente le risorse obbligatorie

e `Consent` le risorse `MemberPatientCoverageToMatch`, oltre a un parametro opzionale. `CoverageToLink`

```
POST [base]/Group/$bulk-member-match
Content-Type: application/fhir+json
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "MemberBundle",
      "part": [
        {
          "name": "MemberPatient",
          "resource": {
            "resourceType": "Patient",
            "identifier": [
              {
                "system": "http://example.org/patient-id",
                "value": "patient-0"
              }
            ],
            "name": [
              {
                "family": "Smith",
                "given": ["James"]
              }
            ],
            "gender": "male",
            "birthDate": "1950-01-01"
          }
        },
        {
          "name": "CoverageToMatch",
          "resource": {
            "resourceType": "Coverage",
            "status": "active",
            "identifier": [
              {
                "system": "http://example.org/coverage-id",
                "value": "cov-0"
              }
            ],
            "subscriberId": "sub-0",
```

```
            "beneficiary": {
              "reference": "Patient/patient123"
            },
            "relationship": {
              "coding": [
                {
                  "system": "http://terminology.hl7.org/CodeSystem/subscriber-
relationship",
                  "code": "self"
                }
              ]
            },
            "payor": [
              {
                "reference": "Organization/org123"
              }
            ]
          }
        },
        {
          "name": "Consent",
          "resource": {
            "resourceType": "Consent",
            "status": "active",
            "scope": {
              "coding": [
                {
                  "system": "http://terminology.hl7.org/CodeSystem/consentscope",
                  "code": "patient-privacy"
                }
              ]
            },
            "category": [
              {
                "coding": [
                  {
                    "system": "http://terminology.hl7.org/CodeSystem/v3-ActCode",
                    "code": "IDSCL"
                  }
                ]
              }
            ],
            "patient": {
              "reference": "Patient/patient123"
```

```
            },
            "performer": [
              {
                "reference": "Patient/patient123"
              }
            ],
            "sourceReference": {
              "reference": "http://example.org/DocumentReference/consent-source"
            },
            "policy": [
              {
                "uri": "http://hl7.org/fhir/us/davinci-hrex/StructureDefinition-hrex-
consent.html#regular"
              }
            ],
            "provision": {
              "type": "permit",
              "period": {
                "start": "2024-01-01",
                "end": "2025-12-31"
              },
              "actor": [
                {
                  "role": {
                    "coding": [
                      {
                        "system": "http://terminology.hl7.org/CodeSystem/provenance-
participant-type",
                        "code": "performer"
                      }
                    ]
                  },
                  "reference": {
                    "identifier": {
                      "system": "http://hl7.org/fhir/sid/us-npi",
                      "value": "9876543210"
                    },
                    "display": "Old Health Plan"
                  }
                },
                {
                  "role": {
                    "coding": [
                      {
```

```
                        "system": "http://terminology.hl7.org/CodeSystem/v3-
ParticipationType",
                        "code": "IRCP"
                      }
                    ]
                  },
                  "reference": {
                    "identifier": {
                      "system": "http://hl7.org/fhir/sid/us-npi",
                      "value": "0123456789"
                    },
                    "display": "New Health Plan"
                  }
                }
              ],
              "action": [
                {
                  "coding": [
                    {
                      "system": "http://terminology.hl7.org/CodeSystem/consentaction",
                      "code": "disclose"
                    }
                  ]
                }
              ]
            }
          }
        },
        {
          "name": "CoverageToLink",
          "resource": {
            "resourceType": "Coverage",
            "status": "active",
            "identifier": [
              {
                "system": "http://example.org/coverage-link-id",
                "value": "cov-link-0"
              }
            ],
            "subscriberId": "new-sub-0",
            "beneficiary": {
              "reference": "Patient/patient123"
            },
            "relationship": {
```

```
              "coding": [
                {
                  "system": "http://terminology.hl7.org/CodeSystem/subscriber-
relationship",
                  "code": "self"
                }
              ]
            },
            "payor": [
              {
                "identifier": {
                  "system": "http://hl7.org/fhir/sid/us-npi",
                  "value": "0123456789"
                },
                "display": "New Health Plan"
              }
            ]
          }
        }
      ]
    }
  ]
}
```

Risposta al lavoro completata con output

Una volta completato il lavoro, la risposta include i metadati del lavoro e una risorsa FHIR Parameters contenente tre risorse di gruppo che classificano i risultati delle partite.

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

Come HealthLake classifica i membri in gruppi di output

Ogni membro inviato in una `$bulk-member-match` richiesta viene valutato tramite una pipeline sequenziale. Il risultato di ogni passaggio determina in quale gruppo di output viene inserito il membro.

1. Validazione strutturale: è MemberBundle conforme ai profili richiesti? In caso di errore: errore (non in nessun gruppo).
2. Abbinamento tra pazienti: è possibile HealthLake trovare pazienti corrispondenti ai dati demografici inviati? In caso di fallimento:. NonMatchedMembers
3. Conferma della copertura: è possibile HealthLake restringere la copertura a un solo paziente con valido CoverageToMatch? In caso di fallimento: NonMatchedMembers.
4. Valutazione del consenso: il consenso presentato è onorevole in questo momento? (status = attivo, il periodo copre la data corrente, l'esecutore può essere convalidato). In caso di fallimento:. ConsentConstrainedMembers
5. Operazione riuscita: tutti i controlli vengono superati. Consenso memorizzato nel datastore. Membro inserito. MatchedMembers

Principio chiave: un membro può apparire solo in una destinazione. Il primo passaggio non riuscito determina il posizionamento. I membri che non superano le fasi 2 o 3 non vengono mai inseriti ConsentConstrainedMembers : tale gruppo è riservato esclusivamente ai membri che hanno raggiunto l'abbinamento con successo ma il cui consenso non può essere rispettato.

Dettagli sulla valutazione del consenso (Fase 4):

- Verifica 1 — Stato del consenso: è `Consent.status` uguale a «attivo»? In caso contrario → ConsentConstrainedMembers.
- Verifica 2 - Periodo di fornitura: `provision.period` copre la data corrente? Se la data corrente è precedente `period.start` o successiva a `period.end` → ConsentConstrainedMembers.
- Controllo 3 — Convalida dell'esecutore: Il `Consent.performer` riferimento può essere convalidato? Se la risorsa referenziata non viene trovata nel datastore o non è associata al paziente corrispondente →. ConsentConstrainedMembers

Tutti i controlli devono essere superati affinché il membro venga inserito MatchedMembers e il consenso venga archiviato.

Comportamento di corrispondenza della copertura

Durante l'abbinamento dei membri, solo `CoverageToMatch` viene convalidato rispetto al datastore del pagante che risponde. `CoverageToLink`appartiene al new/requesting pagante e non è convalidato rispetto al datastore del vecchio pagante. L'inclusione `CoverageToLink` nella richiesta non influirà sui risultati corrispondenti.

Ogni combinazione Patiente+Copertura nella richiesta viene elaborata in modo indipendente. Lo stesso paziente può essere inviato più volte con piani di copertura diversi e ogni candidatura riceve il proprio risultato in base alla copertura specifica.

Gestione delle referenze da parte dell'esecutore del consenso

Il nuovo pagatore può inviare una referenza temporanea o locale per il paziente `Consent.performer` (ad esempio, la stessa referenza utilizzata in`Consent.patient`). HealthLake risolve automaticamente questi riferimenti:

- Se `Consent.performer` contiene lo stesso riferimento locale di`Consent.patient`, lo HealthLake sostituisce con il riferimento del paziente effettivamente corrispondente dopo che l'abbinamento è riuscito.
- HealthLake supporta riferimenti agli esecutori di tipo Paziente RelatedPerson, Professionista e Organizzazione (sia riferimenti diretti che riferimenti identificativi logici). PractitionerRole
- Se la convalida dell'esecutore fallisce (risorsa non trovata o non associata al paziente corrispondente), il membro viene inserito anziché restituire un errore. ConsentConstrainedMembers

Risorse di Output Group

Il lavoro completato restituisce una risorsa Parameters contenente tre risorse di gruppo:

MatchedMembers Group (Gruppo)

Contiene i riferimenti dei pazienti per tutti i membri abbinati correttamente il cui consenso è attivo e valido al momento della richiesta. La risorsa Consent viene creata e archiviata nel datastore per ogni membro corrispondente. Questo gruppo è istanziato nel datastore e può essere utilizzato direttamente con. `$davinci-data-export`

NonMatchedMembers Group (Gruppo)

Contiene riferimenti ai membri in cui non è stata trovata alcuna corrispondenza univoca. Un membro viene inserito qui quando nessun paziente nel datastore corrisponde ai dati demografici forniti, non esiste una copertura valida per nessun paziente candidato corrispondente o più pazienti corrispondono ai dati demografici e più pazienti hanno una copertura valida (ambiguo).

ConsentConstrainedMembers Group (Gruppo)

Contiene i riferimenti dei pazienti che sono stati abbinati con successo (dati demografici e copertura confermati) ma il cui consenso non può essere rispettato al momento della richiesta. La risorsa Consent non viene archiviata per i membri con vincoli di consenso. L'identità del membro corrispondente (MemberIdentifier e MemberId) è ancora inclusa in modo che il pagante richiedente sappia chi era vincolato.

Il `Group.quantity` campo contiene il numero totale di membri in ciascuno dei rispettivi gruppi.

Riferimenti dei membri del gruppo:

- `Group.member.entity.reference`— Per MatchedMembers e ConsentConstrainedMembers, contiene l'ID del paziente del membro corrispondente nel sistema del pagatore che risponde. Per NonMatchedMembers, fa riferimento all'input Paziente contenuto.
- `Group.member.entity.extension (base-ext-match-parameters)`— Contiene l'ID del paziente riportato nella richiesta di immissione originale (l'ID inviato dal pagatore richiedente, derivato da `Patient.idCoverage.beneficiary.reference`, o`Consent.patient.reference`).

Consent-Patient collegamento

> **Important**
>
> La risorsa Consent memorizzata conserva il riferimento del paziente esattamente come inviato dal pagatore richiedente. HealthLake non aggiorna automaticamente il campo del paziente del consenso in modo che punti al Paziente corrispondente nel datastore ricevente.

Per collegare un Consenso memorizzato al Paziente corrispondente, utilizza il job output: ogni membro del MatchedMembers Gruppo rimanda al `member.entity.reference` Paziente corrispondente e un `member.entity.extension` (`base-ext-match-parameters`) rimanda al Paziente di input contenuto. Cross-reference aggiungili al campo Consent's patient per creare la mappatura nel livello di candidatura.

Cosa viene archiviato rispetto a cosa è transitorio

La tabella seguente documenta ciò che HealthLake persiste nel datastore durante l'`$bulk-member-match`elaborazione e ciò che esiste solo nella risposta al lavoro:

Risorsa	Memorizzato?	Interrogabile tramite REST?	Note
MemberPatient (ingresso)	No	No	Utilizzato solo per la corrispondenza; non persistente
CoverageToMatch (input)	No	No	Utilizzato solo per la conferma della copertura
CoverageToLink (input)	No	No	Non convalidato rispetto al datastore; appartiene al nuovo pagatore
Consenso (membri abbinati)	Sì	Sì, GET [base] / Consent/ {id}	Memorizzato così come ricevuto dal pagatore richiedente

Risorsa	Memorizzato?	Interrogabile tramite REST?	Note
Consenso (membri vincolati)	No	No	Non memorizzato. L'identità del membro è ancora inclusa nella risposta.
MatchedMembers Gruppo (output)	Sì	Sì, GET [base] / Group/ {id}	Istanziato; utilizzabile con $davinci-data-export
NonMatchedMembers Gruppo	No	No	Solo Job response
ConsentConstrained Members Gruppo	No	No	Solo Job response

Integrazione con $davinci-data-export

La risorsa MatchedMembers di gruppo restituita da `$bulk-member-match` può essere utilizzata direttamente con l'`$davinci-data-export`operazione per recuperare i dati di massa dei membri:

```
POST [base]/Group/{matched-group-id}/$davinci-data-export
GET [base]/Group/{matched-group-id}
```

Questa integrazione consente flussi di lavoro efficienti in cui si identificano innanzitutto i membri corrispondenti in blocco, quindi si esportano le relative cartelle cliniche complete utilizzando la risorsa di Gruppo risultante.

Usare $member-remove prima dell'esportazione

Se devi escludere membri specifici dall'esportazione dopo l'abbinamento (ad esempio, un membro revoca il consenso tra l'abbinamento e l'esportazione), usalo nel gruppo. `$member-remove` MatchedMembers

> **Important**
>
> La rimozione di un membro tramite `$member-remove` contrassegna il membro come inattivo nel Gruppo, ma esclude i membri inattivi `$davinci-data-export` solo dopo che il Gruppo

è stato aggiornato allo stato «finale». Se `$davinci-data-export` chiami un Gruppo che ha ancora lo status predefinito, i membri rimossi potrebbero comunque apparire nei risultati di esportazione.

Flusso di lavoro:

1. `POST [base]/Group/{id}/$member-remove`— contrassegna i membri inattivi
2. `PUT [base]/Group/{id}`— aggiorna lo stato del gruppo a «finale»
3. `POST [base]/Group/{id}/$davinci-data-export`— l'esportazione ora esclude i membri rimossi

Caratteristiche prestazionali

L'`$bulk-member-match`operazione è progettata per l'elaborazione di grandi volumi e viene eseguita in modo asincrono:

- Concorrenza: massimo 5 operazioni simultanee per archivio dati.
- Scalabilità: gestisce fino a 500 membri per richiesta (limite di payload di 5 MB).
- Operazioni parallele: compatibile con operazioni simultanee di importazione, esportazione o eliminazione in blocco.

Autorizzazione

L'API utilizza il protocollo di autorizzazione SMART on FHIR con i seguenti ambiti richiesti:

- `system/Patient.read`— Necessario per cercare e abbinare le risorse dei pazienti.
- `system/Coverage.read`— Necessario per convalidare le informazioni sulla copertura.
- `system/Group.write`— Necessario per creare risorse del Gruppo di risultati.
- `system/Organization.read`— Condizionale, obbligatorio se la copertura si riferisce alle organizzazioni.
- `system/Practitioner.read`— Condizionale, obbligatorio se la copertura si riferisce ai professionisti.
- `system/PractitionerRole.read`— Condizionale, obbligatorio se la copertura si riferisce ai ruoli dei professionisti.
- `system/Consent.write`— Condizionale, obbligatorio se vengono fornite risorse per il consenso.

L'operazione supporta anche l'autorizzazione AWS IAM Signature Version 4 (SigV4) per l'accesso programmatico.

Regole di convalida

Le seguenti regole di convalida vengono applicate a ciascuna di esse MemberBundle nella Fase 1. I membri che non superano la convalida vengono segnalati come errori e non compaiono in nessun gruppo di output.

MemberPatient

Campo	Come HealthLake lo usa	Errore di convalida se...
`name.family`	Ricerca demografica	Mancante
`name.given`	Ricerca demografica	Mancante (almeno una richiesta)
`birthDate`	Ricerca demografica	Mancante
`gender`	Ricerca demografica; se assente, viene utilizzata l'estensione Birth Sex	Non sono presenti né sesso né sesso di nascita (hrex-pat-1)
`identifier`	Incluso nella ricerca quando presente; migliora la fiducia	Non causa mai guasti (opzionale)

CoverageToMatch / CoverageToLink

Campo	Come lo HealthLake usa	Errore di convalida se...
`status`	Conferma che la copertura è utilizzabile	Mancante
`beneficiary`	Collega la copertura a un candidato paziente	Mancante
`payor`	Disambiguazione quando esistono più candidati	Mancante o più di un pagatore

Campo	Come lo HealthLake usa	Errore di convalida se...
`relationship`	Conferma la relazione abbonato-beneficiario	Mancante
`identifier (MB) or subscriberId`	Chiave di disambiguazione principale	Nessuno dei due presenti

Consenso

Campo	Come lo HealthLake utilizza	Errore di convalida se...
`scope`	Conferma che l'ambito del consenso è la privacy del paziente	Codice sulla privacy del paziente mancante o assente
`category`	Conferma la classificazione delle divulgazioni	Mancante
`patient`	Identifica il soggetto del consenso	Mancante
`performer`	Identifica chi è d'accordo	Mancante
`sourceReference`	Documenta la fonte del consenso	Mancante
`policy.uri`	Determina l'ambito di condivisione dei dati	Mancante o URI che non termina con #regular o #sensitive
`provision.type`	Deve essere «permesso» per il profilo HREx Consent	«Permesso» mancante o assente (incluso «rifiuto»)
`provision .period`	Valutato nella Fase 4 per un controllo limitato al consenso	Mancante o assente start/end
`status`	Valutato nella fase 4 (NON nella fase 1)	Non causa mai un errore nella Fase 1: HealthLake accetta qualsiasi stato valido e valuta nella Fase 4

> **Note**
>
> Il profilo HREx Consent definisce lo stato con un valore fisso di «attivo». HealthLake allenta intenzionalmente questo vincolo in modo che un consenso non attivo riceva una classificazione significativa () ConsentConstrainedMembers anziché un rifiuto generalizzato di convalida.

Comportamento di corrispondenza

- Ricerca paziente (fase 2): HealthLake ricerche utilizzando `name.family` + `name.given` (esatto, senza distinzione tra maiuscole e minuscole), `birthDate` (esatto), `gender` (esatto; sesso di nascita utilizzato se il sesso non è presente) e `identifier` (se presente, facoltativo).
- Disambiguazione della copertura (Fase 3): quando vengono trovati più pazienti candidati, `CoverageToMatch` viene utilizzata per restringere il campo a uno solo. Una copertura è «valida» quando nel datastore è presente una risorsa Coverage attiva corrispondente a `subscriberId` o `identifier` (tipo MB) AND. `payor`
- Valutazione del consenso (Fase 4): eseguita solo dopo una corrispondenza univoca riuscita. Vedi la sezione precedente relativa ai dettagli sulla valutazione del consenso.

Gestione degli errori

L'operazione gestisce le seguenti condizioni di errore:

- 400 Richiesta errata: formato di richiesta non valido, parametri obbligatori mancanti o il payload supera i limiti di dimensione (500 membri o 5 MB).
- 422 Entità non processabile: errori di elaborazione durante l'esecuzione del lavoro.
- Errori individuali dei membri: quando un membro specifico non viene elaborato, l'operazione continua con i membri rimanenti e include i dettagli dell'errore nel NonMatchedMembers Gruppo con i codici motivo appropriati. Ad esempio, se `MemberBundle` a un paziente manca il `birthDate` parametro, verrà restituito il seguente errore:

```
"errors": [
  {
    "memberIndex": 1,
    "jsonBlob": {
      "resourceType": "OperationOutcome",
      "issue": [
```

```
        {
          "severity": "error",
          "code": "invalid",
          "diagnostics": "MemberPatient.birthDate is required"
        }
      ],
      "statusCode": 400
    }
  }
]
```

I dettagli sull'errore sono disponibili tramite l'endpoint di polling dello stato e includono:

- `numberOfMembersWithCustomerError`: Numero di membri con errori di convalida o di input.
- `numberOfMembersWithServerError`: Numero di membri con errori di elaborazione sul lato server.

Operazioni correlate

- the section called “$member-match”— Operazione di abbinamento dei singoli membri.
- the section called “$davinci-data-export”— Esportazione di dati in blocco utilizzando le risorse del Gruppo.
- the section called “$ Operazioni”— Elenco completo delle operazioni supportate.

Funzionamento FHIR R4 **`$confirm-attribution-list`** per HealthLake

Indica a un produttore che il consumatore non ha più modifiche da apportare all'elenco di attribuzione. Questa operazione finalizza l'elenco di attribuzione rimuovendo i membri inattivi dall'elenco, modificando lo stato in «finale» e confermando l'operazione. Questa operazione fa parte dell'implementazione FHIR Member Attribution (ATR) List IG 2.1.0.

Endpoint

```
POST [base]/Group/[id]/$confirm-attribution-list
```

Richiesta

Nessun parametro richiesto.

```
POST [base]/Group/[id]/$confirm-attribution-list
```

Risposta

Restituisce HTTP 201 con un messaggio di conferma.

Risposta di esempio

```
HTTP Status Code: 201

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "message",
      "valueString": "Accepted."
    }
  ]
}
```

Stato del gruppo dopo la conferma

Dopo la conferma avvenuta con successo, lo stato della lista di attribuzione della risorsa del Gruppo sarà impostato su «finale»:

```
{
  "resourceType": "Group",
  "id": "fullexample",
  "extension": [
    {
      "url": "http://hl7.org/fhir/us/davinci-atr/StructureDefinition/ext-
attributionListStatus",
      "valueCode": "final"
    }
  ]
  // ... other Group properties
}
```

Gestione errori

L'operazione gestisce le seguenti condizioni di errore:

Errore	Stato HTTP	Description
Richiesta di operazione non valida	400	Parametri o struttura della richiesta non conformi
Risorsa di gruppo non trovata	404	L'ID di gruppo specificato non esiste

Autorizzazione e sicurezza

Requisiti di SMART

I clienti devono disporre dei privilegi appropriati per leggere le risorse del Gruppo e le relative risorse di attribuzione

Infatti`$confirm-attribution-list`, i client devono disporre anche dei privilegi di scrittura per modificare le risorse del Gruppo

I meccanismi di autorizzazione FHIR standard si applicano a tutte le operazioni

Funzionamento FHIR R4 **`$davinci-data-export`** per HealthLake

L'`$davinci-data-export`operazione è un'operazione FHIR asincrona da cui è possibile esportare dati sanitari. AWS HealthLake Questa operazione supporta diversi tipi di esportazione, tra cui Member Attribution (ATR), PDex Provider Access e Member Access. Payer-to-Payer APIs È una versione specializzata del `$export` funzionamento standard FHIR, progettata per soddisfare i requisiti delle guide all'implementazione. DaVinci

Caratteristiche chiave

- Elaborazione asincrona: segue lo schema di richiesta asincrona FHIR standard
- Esportazione a livello di gruppo: esporta i dati per i membri all'interno di una specifica risorsa del gruppo
- Diversi tipi di esportazione: supporta ATR (Member Attribution), PDex Provider Access e Member Access Payer-to-Payer APIs
- Profile Support completo: include US Core, CARIN Blue Button e PDex profili
- Filtraggio flessibile: supporta il filtraggio per pazienti, tipi di risorse e intervalli di tempo

- Output NDJSON: fornisce dati in formato JSON delimitato da nuove righe

Operazione Endpoint

```
GET [base]/Group/[id]/$davinci-data-export
POST [base]/Group/[id]/$davinci-data-export
```

Parametri della richiesta

Parametro	Cardinalità	Description
`patient`	0.. *	Membri specifici di cui esportare i dati. Se omesso, tutti i membri del Gruppo vengono esportati.
`_type`	0.1.	Elenco delimitato da virgole di tipi di risorse FHIR da esportare. Se omesso, vengono inclusi tutti i tipi di risorse supportati per il tipo di esportazione specificato. Per le esportazioni ATR, il valore predefinito è costituito dagli 8 tipi di risorse di attribuzione. Per PDex le esportazioni, sono inclusi tutti i tipi di risorse di attribuzione, oltre ai tipi di risorse cliniche e per i reclami di US Core, CARIN Blue Button e dei profili. PDex
`_since`	0.1.	Includi solo le risorse aggiornate dopo questa data e ora.
`_until`	0.1..	Includi solo le risorse aggiornate prima di questa data e ora.
`exportType`	0.1..	Tipo di esportazione da eseguire. Valori validi: `hl7.fhir.us.davinci-atr` , `hl7.fhir.us.davinci-pdex` , `hl7.fhir.us.davinci-pdex.p2p` , `hl7.fhir.us.davinci-pdex.member` . Default: `hl7.fhir.us.davinci-atr` .

Parametro	Cardinalità	Description
`_includeEOB2xWoFinancial`	0.1.	Specifica se includere le ExplanationOfBenefit risorse CARIN BB 2.x con i dati finanziari rimossi. Default: `false`.
`_security`	0.. *	Filtra le risorse esportate in base ai valori di `meta.security` codifica. Usa il `system|code` formato (il carattere pipe deve avere la codifica URL come). `%7C` Quando vengono forniti più valori, le risorse devono corrispondere a tutti (semantica AND). Usa `system|` (trailing pipe, no code) per abbinare qualsiasi codice di un determinato sistema.
`_tag`	0.. *	Filtra le risorse esportate in base ai valori di `meta.tag` codifica. Utilizza lo stesso `system|code` formato e la stessa semantica AND di. `_security` Quando `_tag` vengono specificati entrambi `_security` i filtri, le risorse devono corrispondere a entrambi i filtri.

Tipi di risorsa supportati

I tipi di risorse supportati dipendono dal tipo di esportazione specificato. Per le esportazioni ATR, sono supportati i seguenti tipi di risorse:

- `Group`
- `Patient`
- `Coverage`
- `RelatedPerson`
- `Practitioner`
- `PractitionerRole`
- `Organization`
- `Location`

Per PDex le esportazioni (Provider Access e Member Access), oltre ai tipi precedenti, sono supportati tutti i tipi di risorse cliniche e relative ai reclami. Payer-to-Payer Per un elenco completo dei tipi di risorse supportati, consulta la US Core Implementation Guide (STU 6.1), la CARIN Blue Button Implementation Guide e la Da Vinci Prior Authorization Support Implementation Guide.

Tipi di esportazione

L'`$davinci-data-export`operazione supporta i seguenti tipi di esportazione. È possibile specificare il tipo di esportazione utilizzando il `exportType` parametro.

Tipo di esportazione	Scopo	Ambito dei dati	Limite temporale
`hl7.fhir.us.davinci-atr`	Elenco di attribuzione dei membri	Risorse relative all'attribuzione	Nessuno
`hl7.fhir.us.davinci-pdex`	API Provider Access	Dati clinici e relativi ai reclami relativi ai pazienti attribuiti	5 anni
`hl7.fhir.us.davinci-pdex.p2p`	Payer-to-Payer Scambio	Dati storici dei membri per le transizioni assicurative	5 anni
`hl7.fhir.us.davinci-pdex.member`	API di accesso ai membri	Dati sanitari propri del socio	5 anni

> **Note**
>
> Per PDex le esportazioni, il limite temporale di 5 anni non si applica ai tipi di risorse ATR (`Group`,,`Patient`,`Coverage`,`RelatedPerson`, `PractitionerPractitionerRole`,`Organization`). `Location` Queste risorse sono sempre incluse indipendentemente dall'età.

ATR (hl7.fhir.us.davinci-atr)

Con il tipo di esportazione ATR, puoi esportare i dati della Member Attribution List. Utilizzate questo tipo di esportazione per recuperare risorse relative all'attribuzione per i membri di un gruppo. Per ulteriori informazioni, consulta l'operazione di esportazione ATR Da Vinci.

Tipi di risorsa supportati

`Group`, `Patient`, `Coverage`, `RelatedPerson`, `Practitioner`, `PractitionerRole`, `Organization`, `Location`

Filtraggio temporale

Non viene applicato alcun filtro temporale. Tutte le risorse corrispondenti vengono esportate indipendentemente dalla data.

PDex Tipi di esportazione

Tutti i tipi di PDex esportazione condividono gli stessi profili e la stessa logica di filtraggio supportati. Per ulteriori informazioni, consulta l'API Da Vinci PDex Provider Access. Sono supportati i seguenti profili:

- US Core 3.1.1, 6.1.0 e 7.0.0
- PDex Autorizzazione preventiva (non supportata per l'accesso ai membri)
- CARIN BB 2.x Profili di base: ospedaliero istituzionale, ambulatoriale istituzionale, professionale NonClinician, orale, farmaceutico

Per PDex le esportazioni, le risorse cliniche e per i reclami vengono scoperte automaticamente per ogni paziente del Gruppo. Non è necessario fare riferimento esplicito a queste risorse nella risorsa del Gruppo. L'operazione cerca tutte le risorse relative al comparto pazienti (ad esempio`Observation`,, `Condition` `Coverage` `RelatedPerson`, `MedicationRequest`, e`ExplanationOfBenefit`) che appartengono ai pazienti attribuiti. Solo `Patient` i tipi e non-patient-compartment ATR (`Practitioner`,, `PractitionerRole`,`Organization`,`Location`) richiedono riferimenti espliciti nel Gruppo. `Group`

Accesso al provider () `hl7.fhir.us.davinci-pdex`

Consente ai provider in rete di recuperare i dati dei pazienti per i pazienti attribuiti.

Payer-to-Payer (`hl7.fhir.us.davinci-pdex.p2p`)

Consente lo scambio di dati tra i pagatori quando un paziente cambia assicurazione.

Accesso per i membri () `hl7.fhir.us.davinci-pdex.member`

Consente ai membri di accedere ai propri dati sanitari. Questo tipo di esportazione può includere dati finanziari nelle risorse relative ai reclami.

Logica di supporto e inclusione del profilo

Per PDex le esportazioni, l'`$davinci-data-export`operazione utilizza le dichiarazioni di profilo nell'`meta.profile`elemento per determinare quali risorse includere nell'esportazione.

ExplanationOfBenefit Gestione delle risorse

`ExplanationOfBenefit`Le risorse (EOB) sono incluse o escluse dalle PDex esportazioni in base alle loro `meta.profile` dichiarazioni:

- ExplanationOfBenefit le risorse con un profilo CARIN BB 1.x sono escluse dall'esportazione.
- ExplanationOfBenefit le risorse senza `meta.profile` set sono escluse dall'esportazione.
- ExplanationOfBenefit le risorse con un profilo CARIN BB 2.x Basis sono sempre incluse.
- ExplanationOfBenefit le risorse con un profilo CARIN BB 2.x che contiene dati finanziari sono escluse per impostazione predefinita. Quando `_includeEOB2xWoFinancial=true` è impostato, vengono incluse con i dati finanziari rimossi e la risorsa viene trasformata nel profilo Basis corrispondente.
- ExplanationOfBenefit le risorse con un profilo di autorizzazione PDex preventiva sono sempre incluse.

Trasformazione dei dati finanziari

Una volta impostata`_includeEOB2xWoFinancial=true`, l'operazione trasforma le ExplanationOfBenefit risorse CARIN BB 2.x nei profili Basis corrispondenti rimuovendo i dati finanziari. Ad esempio, una `C4BB ExplanationOfBenefit Oral` risorsa viene trasformata in`C4BB ExplanationOfBenefit Oral Basis`, il che rimuove i dati finanziari dal record in base alla specifica FHIR.

I seguenti elementi di dati finanziari vengono rimossi durante la trasformazione:

- Tutte le suddivisioni sugli elementi `total`
- Tutti gli `adjudication` elementi con slice `amounttype`
- Tutti gli `item.adjudication` elementi con informazioni sulla quantità

L'operazione aggiorna anche i metadati del profilo durante la trasformazione:

- `meta.profile`viene aggiornato all'URL canonico del profilo Basis
- La versione è aggiornata alla versione CARIN BB 2.x Basis
- Le risorse esistenti nell'archivio dati non vengono modificate
- Le risorse esportate non vengono salvate in modo persistente nell'archivio dati

Regole di rilevamento del profilo

L'operazione utilizza le seguenti regole per rilevare e convalidare i profili:

- Il rilevamento della versione si basa sul codice canonico `meta.profile` URLs
- Una risorsa viene inclusa se UNO QUALSIASI dei suoi profili dichiarati soddisfa i criteri di esportazione
- La convalida del profilo avviene durante l'elaborazione dell'esportazione

Filtraggio temporale quinquennale per le esportazioni PDex

Per tutti i tipi di PDex esportazione, HealthLake applica un filtro temporale di 5 anni basato sull'ultimo aggiornamento della risorsa. Il filtro temporale si applica a tutte le risorse ad eccezione dei seguenti tipi di risorse di attribuzione principali, che vengono sempre esportati indipendentemente dall'età:

- `Patient`
- `Coverage`
- `Organization`
- `Practitioner`
- `PractitionerRole`
- `RelatedPerson`
- `Location`

- `Group`

Queste risorse amministrative e demografiche sono esenti perché forniscono un contesto essenziale per i dati esportati. Le esportazioni ATR non sono soggette ad alcun filtro temporale.

Richieste di esempio

Gli esempi seguenti mostrano come avviare processi di esportazione per diversi tipi di esportazione.

Esportazione 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"
    }
  }
}
```

Esportazione di Provider Access con ExplanationOfBenefit rimozione dei dati finanziari

```
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 esportazione

```
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
```

Esportazione di Member Access per un paziente specifico

```
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
```

Risposta di esempio

```
{
  "datastoreId": "eaee622d8406b41eb86c0f4741201ff9",
  "jobStatus": "SUBMITTED",
  "jobId": "48d7b91dae4a64d00d54b70862f33f61"
}
```

Relazioni con le risorse

L'operazione esporta le risorse in base alle loro relazioni all'interno dell'Elenco di attribuzione dei membri:

```
Group (Attribution List)
### Patient (Members)
### Coverage # RelatedPerson (Subscribers)
### Practitioner (Attributed Providers)
### PractitionerRole # Location
### Organization (Attributed Providers)
```

> **Note**
>
> Il precedente diagramma delle relazioni tra le risorse si applica alle esportazioni ATR. Per quanto riguarda PDex le esportazioni, le risorse cliniche e relative ai reclami vengono individuate automaticamente tramite la ricerca dei pazienti e non richiedono riferimenti espliciti nella risorsa del Gruppo.

Fonti di risorse

Risorsa	Ubicazione della fonte	Description
`Patient`	`Group.member.entity`	I pazienti che sono membri della lista di attribuzione
`Coverage`	`Group.member.extension:coverageReference`	Copertura che ha portato all'iscrizione dei pazienti
`Organization`	`Group.member.extension:attributedProvider`	Organizzazioni a cui vengono attribuiti i pazienti
`Practitioner`	`Group.member.extension:attributedProvider`	Professionisti individuali a cui vengono attribuiti i pazienti
`PractitionerRole`	`Group.member.extension:attributedProvider`	Ruoli professionali a cui vengono attribuiti i pazienti
`RelatedPerson`	`Coverage.subscriber`	Abbonati alla copertura
`Location`	`PractitionerRole.location`	Sedi associate ai ruoli professionali
`Group`	Endpoint di input	La lista di attribuzione stessa

Gestione del Job

Verifica lo stato del lavoro

```
GET [base]/export/[job-id]
```

Annullamento di un processo

```
DELETE [base]/export/[job-id]
```

Ciclo di vita del processo

- `SUBMITTED`- Il lavoro è stato ricevuto e messo in coda
- `IN_PROGRESS`- Job è in fase di elaborazione attiva

- `COMPLETED`- Job terminato con successo, file disponibili per il download
- `FAILED`- Job ha riscontrato un errore

Formato di output

- Formato di file: NDJSON (Newline Delimited JSON)
- Organizzazione dei file: file separati per ogni tipo di risorsa
- Estensione del file: .ndjson
- Posizione: bucket e percorso S3 specificati

Gestione errori

L'operazione restituisce HTTP 400 Bad Request con un OperationOutcome per le seguenti condizioni:

Errori di autorizzazione

Il ruolo IAM specificato in `DataAccessRoleArn` non dispone di autorizzazioni sufficienti per eseguire l'operazione di esportazione. Per l'elenco completo delle autorizzazioni S3 e KMS richieste, consulta Configurazione delle autorizzazioni per i lavori di esportazione.

Errori di convalida dei parametri

- Il `patient` parametro non è formattato come `Patient/id,Patient/id,...`
- Uno o più riferimenti ai pazienti non sono validi o non appartengono al gruppo specificato
- Il valore del `exportType` parametro non è un tipo di esportazione supportato
- Il `_type` parametro contiene tipi di risorse che non sono supportati per il tipo di esportazione specificato
- Nel `_type` parametro mancano i tipi di risorse richiesti (`Group`,`Patient`,`Coverage`) per il tipo di `hl7.fhir.us.davinci-atr` esportazione
- Il valore del `_includeEOB2xWoFinancial` parametro non è un valore booleano valido

Errori di convalida delle risorse

- La risorsa di gruppo specificata non esiste nell'archivio dati
- La risorsa di gruppo specificata non ha membri
- Uno o più membri del gruppo fanno riferimento a risorse per i pazienti che non esistono nell'archivio dati

Sicurezza e autorizzazione

- Si applicano i meccanismi di autorizzazione FHIR standard
- Il ruolo di accesso ai dati deve disporre delle autorizzazioni IAM richieste per le operazioni S3 e KMS. Per l'elenco completo delle autorizzazioni richieste, consulta Configurazione delle autorizzazioni per i lavori di esportazione.

Best practice

- Selezione del tipo di risorsa: richiedete solo i tipi di risorse necessari per ridurre al minimo le dimensioni di esportazione e i tempi di elaborazione
- Filtraggio basato sul tempo: utilizza il `_since` parametro per le esportazioni incrementali
- Filtraggio dei pazienti: utilizza il `patient` parametro quando sono necessari solo i dati per membri specifici
- Job Monitoring: controlla regolarmente lo stato del lavoro per esportazioni di grandi dimensioni
- Gestione degli errori: Implementa una logica di ripetizione corretta per i lavori non riusciti
- Consapevolezza del filtro temporale: per PDex le esportazioni, considera il filtro temporale quinquennale quando selezioni i tipi di risorse
- Rimozione dei dati finanziari: da utilizzare `_includeEOB2xWoFinancial=true` quando sono necessari i dati relativi ai reclami senza informazioni finanziarie
- Gestione dei profili: assicurati che le risorse dispongano di dichiarazioni di profilo appropriate, esegui la convalida rispetto ai profili di destinazione prima dell'ingestione e utilizza il controllo delle versioni dei profili per controllare il comportamento di esportazione

Limitazioni

- Nel parametro è possibile specificare un massimo di 500 pazienti `patient`
- L'esportazione è limitata alle sole operazioni a livello di gruppo
- Supporta solo il set predefinito di tipi di risorse per ogni tipo di esportazione
- L'output è sempre in formato NDJSON
- PDex le esportazioni sono limitate a 5 anni di dati clinici e relativi ai reclami
- La trasformazione dei dati finanziari si applica solo ai profili CARIN BB 2.x ExplanationOfBenefit

Risorse aggiuntive

- Elenco di attribuzione dei soci Da Vinci IG
- Da Vinci Payer Data Exchange IG
- IG CARIN Consumer Directed Payer Data Exchange
- Guida all'implementazione di base negli Stati Uniti
- Specifiche FHIR Bulk Data Access

Generazione di documenti clinici con **`$document`**

AWS HealthLake ora supporta il `$document` funzionamento delle risorse di Composizione, consentendovi di generare un documento clinico completo raggruppando la Composizione con tutte le sue risorse di riferimento in un unico pacchetto coeso. Questa operazione è essenziale per le applicazioni sanitarie che devono:

- Creare documenti clinici standardizzati
- Scambia le cartelle cliniche complete dei pazienti
- Archivia una documentazione clinica completa
- Genera report che includano tutto il contesto pertinente

Utilizzo

L'`$document`operazione può essere richiamata sulle risorse di composizione utilizzando i metodi GET e POST:

Operazioni supportate

```
GET/POST [base]/Composition/[id]/$document
```

Parametri supportati

HealthLake supporta il seguente parametro FHIR`$document`:

Parametro	Tipo	Obbligatorio	Predefinita	Description
`persist`	booleano	No	false	Booleano che indica se il server deve archiviare il pacchetto di documenti generato

Esempi

Richiesta GET

```
GET [base]/Composition/180f219f-97a8-486d-99d9-ed631fe4fc57/$document?persist=true
```

Richiesta POST con parametri

```
POST [base]/Composition/180f219f-97a8-486d-99d9-ed631fe4fc57/$document
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "persist",
      "valueBoolean": true
    }
  ]
}
```

Risposta di esempio

L'operazione restituisce una risorsa Bundle di tipo «documento» contenente la composizione e tutte le risorse di riferimento:

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

Comportamento

L'`$document`operazione:

1. Prende la risorsa Composition specificata come base per il documento
2. Identifica e recupera tutte le risorse a cui fa riferimento direttamente la composizione
3. Raggruppa la composizione e tutte le risorse referenziate in un pacchetto di tipo «documento»
4. Memorizza il pacchetto di documenti generato nel datastore quando il parametro persist è impostato su true
5. Identifica e recupera le risorse a cui fa riferimento indirettamente la Composizione per una generazione completa di documenti

L'`$document`operazione attualmente supporta il recupero dei riferimenti alle risorse nel seguente formato:

1. `GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Resource/id`
2. Risorsa/ID

I riferimenti alle risorse non supportati all'interno della risorsa Composition verranno filtrati dal documento generato.

Gestione errori

L'operazione gestisce le seguenti condizioni di errore:

- 400 Richiesta non valida: `$document` operazione non valida (richiesta non conforme) o se il documento risultante non supera la convalida FHIR a causa di riferimenti filtrati quando persist è impostato su true
- 404 Not Found: Risorsa di composizione non trovata

Per ulteriori informazioni sulle specifiche `$document` operative, consultate la documentazione sulla composizione `$document` FHIR R4.

Rimozione permanente di risorse con **`$erase`**

AWS HealthLake supporta l'`$erase`operazione, abilitando l'eliminazione permanente di una risorsa specifica e delle relative versioni storiche. Questa operazione è particolarmente utile quando è necessario:

- Rimuovere definitivamente le singole risorse
- Eliminare cronologie di versioni specifiche
- Gestisci i cicli di vita delle singole risorse
- Rispetta i requisiti specifici di rimozione dei dati

Utilizzo

L'`$erase`operazione può essere richiamata a due livelli:

Livello di istanza di risorsa

```
POST [base]/[ResourceType]/[ID]/$erase?deleteAuditEvent=true
```

Livello specifico della versione

```
POST [base]/[ResourceType]/[ID]/_history/[VersionID]/$erase
```

Parameters

Parametro	Tipo	Obbligatorio	Predefinita	Description
`deleteAuditEvent`	booleano	No	false	Se impostato su true, elimina gli eventi di controllo associati

Esempi

Richiesta di esempio

```
POST [base]/Patient/example-patient/$erase
```

Risposta di esempio

```
{
      "jobId": "5df47e2f51ff3c731847678cb8cad48e",
      "jobStatus": "SUBMITTED"
    }
```

Stato di un processo

Per verificare lo stato di un processo di cancellazione:

```
GET [base]/$erase/[jobId]
```

L'operazione restituisce informazioni sullo stato del lavoro:

```
{
      "datastoreId": "36622996b1fcecb7e12ee2ee085308d3",
      "jobId": "5df47e2f51ff3c731847678cb8cad48e",
      "status": "COMPLETED",
```

```
    "submittedTime": "2025-10-30T16:39:24.160Z"
}
```

Comportamento

L'`$erase`operazione:

1. Processi in modo asincrono per garantire l'integrità dei dati
2. Mantiene le transazioni ACID
3. Fornisce il monitoraggio dello stato del lavoro
4. Rimuove definitivamente la risorsa specificata e le relative versioni
5. Include una registrazione di controllo completa delle attività di eliminazione
6. Supporta l'eliminazione selettiva degli eventi di controllo

Registrazione degli audit

L'`$erase`operazione viene registrata in base DeleteResource all'ID utente, al timestamp e ai dettagli delle risorse.

Limitazioni

- `$erased`la risorsa non verrà visualizzata nei risultati di ricerca o nelle query. `_history`
- Le risorse in fase di cancellazione potrebbero essere temporaneamente inaccessibili durante l'elaborazione
- La misurazione dello storage viene regolata immediatamente man mano che le risorse vengono rimosse definitivamente

Acquisizione dei dati dei pazienti con **`Patient/$everything`**

L'`Patient/$everything`operazione viene utilizzata per interrogare una `Patient` risorsa FHIR, insieme a qualsiasi altra risorsa ad essa correlata. `Patient` L'operazione può essere utilizzata per fornire a un paziente l'accesso all'intera cartella clinica o consentire a un fornitore di eseguire un download di massa di dati relativi a un paziente. HealthLakesupporti `Patient/$everything` per un paziente `id` specifico.

`Patient/$everything`è un'operazione API REST FHIR che può essere richiamata come mostrato negli esempi seguenti.

GET request

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything
```

> **Note**
>
> Le risorse in risposta sono ordinate per tipo di risorsa e risorsa. `id`
> La risposta è sempre compilata con. `Bundle.total`

Parametri di **`Patient/$everything`**

HealthLake supporta i seguenti parametri di interrogazione

Parametro	Informazioni
rapida	Recupera tutti `Patient` i dati dopo una data di inizio specificata.
end	Ottieni tutti i `Patient` dati prima di una data di fine specificata.
since	Aggiorna tutti `Patient` i dati dopo una data specificata.
_tipo	Ottieni `Patient` dati per tipi di risorse specifici.
_conta	Recupera `Patient` dati e specifica la dimensione della pagina.

Example- Ottieni tutti i dati del paziente dopo una data di inizio specificata

`Patient/$everything`può utilizzare il `start` filtro per interrogare solo i dati dopo una data specifica.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?start=2024-03-15T00:00:00.000Z
```

Example- Ottieni tutti i Patient dati prima di una data di fine specificata

Patient $everything può utilizzare il end filtro solo per interrogare i dati prima di una data specifica.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?end=2024-03-15T00:00:00.000Z
```

Example- Aggiorna tutti Patient i dati dopo una data specificata

`Patient/$everything`può utilizzare il `since` filtro per interrogare solo i dati aggiornati dopo una data specifica.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?since=2024-03-15T00:00:00.000Z
```

Example- Ottieni Patient dati per tipi di risorse specifici

Patient $everything può utilizzare il `_type` filtro per specificare tipi di risorse specifici da includere nella risposta. È possibile specificare più tipi di risorse in un elenco separato da virgole.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?_type=Observation,Condition
```

Example- Ottieni Patient dati e specifica la dimensione della pagina

Patient $everything può utilizzare il `_count` per impostare la dimensione della pagina.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?_count=15
```

`Patient/$everything``start`e attributi `end`

HealthLake supporta i seguenti attributi di risorsa per i parametri `Patient/ $everything` `start` e di `end` interrogazione.

Risorsa	Elemento risorsa
Account	Account.ServicePeriod.start
AdverseEvent	AdverseEvent.data
AllergyIntolerance	AllergyIntolerance.data registrata

Risorsa	Elemento risorsa
Appuntamento	Appuntamento. Inizio
AppointmentResponse	AppointmentResponse.inizio
AuditEvent	AuditEvent.periodo.inizio
Di base	Di base. Creato
BodyStructure	NESSUNA_DATA
CarePlan	CarePlan.periodo. inizio
CareTeam	CareTeam.periodo.inizio
ChargeItem	ChargeItem. occurrenceDateTime, ChargeItem .OccurrencePeriod. START, .OccurrenceTiming.Event ChargeItem
Richiedi	claim. billablePeriod.START
ClaimResponse	ClaimResponse.creato
ClinicalImpression	ClinicalImpression.data
Communication	Comunicazione. Inviata
CommunicationRequest	CommunicationRequest. occurrenceDateTime, CommunicationRequest .Periodo di occorrenza. Inizio
Composizione	Composizione. Data

Risorsa	Elemento risorsa
Condizione	Condizione. Data registrata
Consenso	Consenso. Data/ora
Copertura	Copertura.Period.Inizio
CoverageEligibilityRequest	CoverageEligibilityRequest.creato
CoverageEligibilityResponse	CoverageEligibilityResponse.creato
DetectedIssue	DetectedIssue.identificato
DeviceRequest	DeviceRequest. Scritto il
DeviceUseStatement	DeviceUseStatement. Registrato su
DiagnosticReport	DiagnosticReport.efficace
DocumentManifest	DocumentManifest.creato
DocumentReference	DocumentReference.contesto.periodo.inizio
Incontro	Incontro.period.start
EnrollmentRequest	EnrollmentRequest.creato

Risorsa	Elemento risorsa
EpisodeOfCare	EpisodeOfCare.periodo.inizio
ExplanationOfBenefit	ExplanationOfBenefit. Periodo fatturabile. Inizio
FamilyMemberHistory	NESSUNA_DATA
Flag	flag.period.start
Obiettivo	Obiettivo. StatusDate
Gruppo	NESSUNA_DATA
ImagingStudy	ImagingStudy.iniziato
Immunizzazione	Immunizzazione. Registrata
ImmunizationEvaluation	ImmunizationEvaluation.data
ImmunizationRecommendation	ImmunizationRecommendation.data
Fattura	Data della fattura
List	Elenco. Data
MeasureReport	MeasureReport.periodo.inizio
Media	Media. Emesso

Risorsa	Elemento risorsa
MedicationAdministration	MedicationAdministration.efficace
MedicationDispense	MedicationDispense. Quando preparato
MedicationRequest	MedicationRequest. Scritto su
MedicationStatement	MedicationStatement.Data inserita
MolecularSequence	NESSUNA_DATA
NutritionOrder	NutritionOrder.Data/ora
Osservazione	Osservazione. Efficace
Paziente	NESSUNA_DATA
Person	NESSUNA_DATA
Procedura	Procedura. Eseguita
Provenienza	Provenienza. Periodo occorso. Inizio, Provenienza. occurredDateTime
QuestionnaireResponse	QuestionnaireResponse.scritto
RelatedPerson	NESSUNA_DATA

Risorsa	Elemento risorsa
RequestGroup	RequestGroup. Scritto il
ResearchSubject	ResearchSubject.periodo
RiskAssessment	RiskAssessment. occurrenceDateTime, RiskAssessment .Periodo di occorrenza. Inizio
Schedule	Schedule.PlanningHorizon
ServiceRequest	ServiceRequest. Scritto il
Esemplare	Esemplare. Ora di ricezione
SupplyDelivery	SupplyDelivery. occurrenceDateTime, SupplyDelivery .OccurrencePeriod. START, .OccurrenceTiming.Event SupplyDelivery
SupplyRequest	SupplyRequest. Scritto il
VisionPrescription	VisionPrescription.data scritta

Recupero dei codici con ValueSet **$expand**

AWS HealthLake ora supporta l'`$expand`operazione relativa ValueSets ai codici acquisiti da te come cliente, consentendoti di recuperare l'elenco completo dei codici contenuti in quelle ValueSet risorse. Questa operazione è particolarmente utile quando è necessario:

- Recuperare tutti i codici possibili a scopo di convalida
- Visualizza le opzioni disponibili nelle interfacce utente
- Esegui ricerche complete di codice all'interno di un contesto terminologico specifico

Utilizzo

L'`$expand`operazione può essere richiamata sulle ValueSet risorse utilizzando i metodi GET e POST:

Operazioni supportate

```
GET/POST [base]/ValueSet/[id]/$expand
GET [base]/ValueSet/$expand?url=http://example.com
POST [base]/ValueSet/$expand
```

Parametri supportati

HealthLake supporta un sottoinsieme di parametri FHIR R4: `$expand`

Parametro	Tipo	Campo obbligatorio	Description
`url`	uri	No	URL canonico del file da espandere ValueSet
`id`	id	No	ValueSet id della risorsa da espandere (per le operazioni GET o POST)
`filter`	stringa	No	Filtra il risultato dell'espansione del codice
`count`	numero intero	No	Numero di codici da restituire
`offset`	numero intero	No	Numero di codici corrispondenti da saltare prima della restituzione. Si applica dopo il filtraggio e solo ai codici corrispondenti, non al contenuto completo e non filtrato dell'originale ValueSet

Esempi

GET Richiesta per ID

```
GET [base]/ValueSet/example-valueset/$expand
```

Richiesta GET per URL con filtro

```
GET [base]/ValueSet/$expand?url=http://example.com/ValueSet/my-
valueset&filter=male&count=5
```

Richiesta POST con parametri (per ID)

```
POST [base]/ValueSet/example-valueset/$expand
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "count",
      "valueInteger": 10
    },
    {
      "name": "filter",
      "valueString": "admin"
    }
  ]
}
```

Richiesta POST con parametri (tramite 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
    }
  ]
```

```
}
```

Risposta di esempio

L'operazione restituisce una ValueSet risorsa con un `expansion` elemento contenente i codici espansi:

```
{
  "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 risposta include:

- expansion.total: numero totale di codici nell'espanso ValueSet
- expansion.contains: matrice di codici espansi con i relativi valori di sistema, codice e visualizzazione
- expansion.parameter: parametri utilizzati nella richiesta di espansione

Per ulteriori informazioni sulle specifiche `$expand` operative, consultate la documentazione FHIR R4. ValueSet `$expand`

Esportazione di HealthLake dati con FHIR **`$export`**

Puoi esportare i dati in blocco dal tuo archivio HealthLake dati utilizzando l'operazione FHIR $export. HealthLake supporta l'utilizzo e le richieste di FHIR`$export`. `POST` `GET` Per effettuare una richiesta di esportazione`POST`, devi disporre di un utente, gruppo o ruolo IAM con le autorizzazioni richieste, specificarlo `$export` come parte della richiesta e includere i parametri desiderati nel corpo della richiesta.

> **Note**
>
> Tutte le richieste di HealthLake esportazione effettuate utilizzando FHIR `$export` vengono restituite in `ndjson` formato ed esportate in un bucket Amazon S3, dove ogni oggetto Amazon S3 contiene un solo tipo di risorsa FHIR.
> Puoi mettere in coda le richieste di esportazione in base alle quote di servizio dell'account. AWS Per ulteriori informazioni, consulta Service Quotas.

HealthLake supporta i seguenti tre tipi di richieste endpoint di esportazione in blocco.

HealthLake tipi in blocco **`$export`**

Tipo di esportazione	Description	Sintassi
Sistema	Esporta tutti i dati dal server HealthLake FHIR.	`POST https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/$export`

Tutti i pazienti	Esporta tutti i dati relativi a tutti i pazienti, compresi i tipi di risorse associati al tipo di risorsa Paziente.	`POST https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/Patient/$export`
		
		`GET https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/Patient/$export`

Gruppo di pazienti	Esporta tutti i dati relativi a un gruppo di pazienti specificato con un ID di gruppo.	`POST https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/Group/ id/$export`
		
		`GET https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/Group/ id/$export`

Prima di iniziare

Soddisfa i seguenti requisiti per effettuare una richiesta di esportazione utilizzando l'API REST FHIR per HealthLake.

- È necessario aver impostato un utente, un gruppo o un ruolo con le autorizzazioni necessarie per effettuare la richiesta di esportazione. Per ulteriori informazioni, consulta Autorizzazione di una richiesta `$export`.
- Devi aver creato un ruolo di servizio che garantisca HealthLake l'accesso al bucket Amazon S3 in cui desideri esportare i tuoi dati. Il ruolo di servizio deve inoltre essere specificato HealthLake

come principale del servizio. Per ulteriori informazioni sulla configurazione delle autorizzazioni, vedereImpostazione delle autorizzazioni per i lavori di esportazione.

Autorizzazione di una richiesta **`$export`**

Per effettuare correttamente una richiesta di esportazione utilizzando l'API REST FHIR, autorizza il tuo utente, gruppo o ruolo utilizzando IAM o .0. OAuth2 È inoltre necessario avere un ruolo di servizio.

Autorizzazione di una richiesta tramite IAM

Quando effettui una `$export` richiesta, l'utente, il gruppo o il ruolo devono includere azioni IAM nella policy. Per ulteriori informazioni, consulta Impostazione delle autorizzazioni per i lavori di esportazione.

Autorizzazione di una richiesta utilizzando SMART on FHIR (2.0) OAuth

Quando si effettua una `$export` richiesta su un HealthLake data store compatibile con SMART on FHIR, è necessario assegnare gli ambiti appropriati. Per ulteriori informazioni, consulta SMART sugli ambiti di risorse FHIR per HealthLake.

> **Note**
>
> Le FHIR `$export` con `GET` richieste richiedono lo stesso metodo di autenticazione o lo stesso token portatore (nel caso di SMART su FHIR) per richiedere l'esportazione e il recupero dei file. I file esportati utilizzando FHIR `$export` con `GET` sono disponibili per il download per 48 ore.

Effettuare una richiesta **`$export`**

Questa sezione descrive i passaggi necessari da eseguire quando si effettua una richiesta di esportazione utilizzando l'API REST FHIR.

Per evitare addebiti accidentali sul tuo AWS account, ti consigliamo di testare le tue richieste effettuando una `POST` richiesta senza fornire la `$export` sintassi.

Per effettuare la richiesta, devi fare quanto segue:

1. Specificare `$export` nell'URL della `POST` richiesta per un endpoint supportato.
2. Specificate i parametri di intestazione richiesti.

3. Specificate un corpo della richiesta che definisca i parametri richiesti.

Passo 1: Specificare **`$export`** nell'URL della **`POST`** richiesta un endpoint supportato.

HealthLake supporta tre tipi di richieste endpoint di esportazione in blocco. Per effettuare una richiesta di esportazione in blocco, è necessario effettuare una richiesta `POST` basata su uno dei tre endpoint supportati. Gli esempi seguenti mostrano dove specificare `$export` nell'URL della richiesta.

- `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`

È possibile utilizzare i seguenti parametri di ricerca supportati nella stringa di `POST` richiesta.

Parametri di ricerca supportati

HealthLake supporta i seguenti modificatori di ricerca nelle richieste di esportazione in blocco.

I seguenti esempi includono caratteri speciali che devono essere codificati prima di inviare la richiesta.

Nome	Obbligatorio?	Description	Esempio
`_outputFormat`	No	Il formato per i file Bulk Data richiesti da generare. I valori accettati sonoapplication/fhir+ndjson ,application/ndjson ,ndjson.	
`_type`	No	Una stringa di tipi di risorse FHIR	`&_type=MedicationS`

Nome	Obbligatorio?	Description	Esempio
		delimitati da virgole da includere nel processo di esportazione. Ti consigliamo di includerla `_type` perché ciò può avere un impatto sui costi quando tutte le risorse vengono esportate.	`tatement, Observation`
`_since`	No	Tipi di risorse modificati in o dopo la data e l'ora. Se un tipo di risorsa non ha l'ora dell'ultimo aggiornamento, verranno inclusi nella risposta.	`&_since=2024-05-09T00%3A00%3A00Z`
`_until`	No	Tipi di risorse modificati in data e ora o prima. Utilizzato in combinazione con `_since` per definire un intervallo di tempo specifico per l'esportazione. Se un tipo di risorsa non ha un'ora dell'ultimo aggiornamento, verrà escluso dalla risposta.	`&_until=2024-12-31T23%3A59%3A59Z`

Nome	Obbligatorio?	Description	Esempio
`_security`	No	Filtra le risorse esportate in base ai valori di `meta.security` codifica. Usa il formato. `system|code` Quando vengono forniti più valori, le risorse devono corrispondere a tutti (e alla semantica). Usa `system|` (trailing pipe, no code) per abbinare qualsiasi codice di un determinato sistema.	`&_security=https://myorg.com/tenant%7Cclinic-A`
`_tag`	No	Filtra le risorse esportate in base ai valori di `meta.tag` codifica. Utilizza lo stesso `system|code` formato e la stessa semantica AND di. `_security` Quando `_tag` vengono specificati entrambi `_security` i filtri, le risorse devono corrispondere a entrambi i filtri.	`&_tag=https://myorg.com/dept%7Ccardiology`

Fase 2: Specificare i parametri di intestazione richiesti

Per effettuare una richiesta di esportazione utilizzando l'API REST FHIR, è necessario specificare i seguenti parametri di intestazione.

- Tipo di contenuto: `application/fhir+json`
- Preferisco: `respond-async`

Successivamente, è necessario specificare gli elementi richiesti nel corpo della richiesta.

Fase 3: Specificare un corpo della richiesta e definire i parametri richiesti.

La richiesta di esportazione richiede anche un corpo in `JSON` formato. Il corpo può includere i seguenti parametri.

Chiave	Obbligatorio?	Description	Valore
DataAccessRoleArn	Sì	Un ARN di un ruolo di HealthLake servizio. Il ruolo di servizio utilizzato deve essere specificato HealthLake come principale del servizio.	`arn:aws:iam:: 444455556666 :role/your-healthlake-service-role`
JobName	No	Il nome della richiesta di esportazione.	**`your-export-job-name`**
S3Uri	Sì	Parte di una OutputDataConfig chiave. L'URI S3 del bucket di destinazione in cui verranno scaricati i dati esportati.	`s3://amzn-s3-demo-bucket/ EXPORT-JOB /`
KmsKeyId	Sì	Parte di una chiave. OutputDataConfig L'ARN della AWS	`arn:aws:kms: region-of-`

Chiave	Obbligatorio?	Description	Valore
		KMS chiave utilizzata per proteggere il bucket Amazon S3.	**bucket:123456789012** :key/**1234abcd-12ab-34cd-56ef-1234567890ab**

Example Corpo di una richiesta di esportazione effettuata utilizzando l'API REST FHIR

Per effettuare una richiesta di esportazione utilizzando l'API REST FHIR, è necessario specificare un corpo, come illustrato di seguito.

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

Quando la richiesta avrà esito positivo, riceverai la seguente risposta.

Intestazione della risposta

```
content-location: https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
export/your-export-request-job-id
```

Corpo di risposta

```
{
  "datastoreId": "your-data-store-id",
  "jobStatus": "SUBMITTED",
  "jobId": "your-export-request-job-id"
}
```

Gestione della richiesta di esportazione

Dopo aver effettuato una richiesta di esportazione corretta, puoi gestirla descrivendo `$export` lo stato di una richiesta di esportazione corrente e `$export` annullando una richiesta di esportazione corrente.

Quando annulli una richiesta di esportazione utilizzando l'API REST, ti viene fatturata solo la parte dei dati che sono stati esportati fino al momento in cui hai inviato la richiesta di annullamento.

I seguenti argomenti descrivono come visualizzare lo stato o annullare una richiesta di esportazione corrente.

Annullamento di una richiesta di esportazione

Per annullare una richiesta di esportazione, effettua una `DELETE` richiesta e fornisci l'ID del lavoro nell'URL della richiesta.

```
DELETE https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/export/your-export-request-job-id
```

Quando la richiesta ha esito positivo, riceverai quanto segue.

```
{
  "exportJobProperties": {
    "jobId": "your-original-export-request-job-id",
    "jobStatus": "CANCEL_SUBMITTED",
    "datastoreId": "your-data-store-id"
  }
}
```

Quando la tua richiesta non va a buon fine, ricevi quanto segue.

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "not-supported",
      "diagnostics": "Interaction not supported."
    }
  ]
}
```

Descrizione di una richiesta di esportazione

Per conoscere lo stato di una richiesta di esportazione, effettua una `GET` richiesta utilizzando `export` and your**`export-request-job-id`**.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/export/your-export-request-id
```

La risposta JSON conterrà un `ExportJobProperties` oggetto. Può contenere le seguenti coppie chiave:valore.

Nome	Obbligatorio?	Description	Valore
DataAccessRoleArn	No	Un ARN di un ruolo di HealthLake servizio. Il ruolo di servizio utilizzato deve essere specificato HealthLake come principale del servizio.	`arn:aws:iam:: 444455556666 :role/your-healthlake-service-role`
SubmitTime	No	La data e l'ora in cui è stato inviato un processo di esportazione.	`Apr 21, 2023 5:58:02`
EndTime	No	L'ora in cui è stato completato un processo di esportazione.	`Apr 21, 2023 6:00:08 PM`
JobName	No	Il nome della richiesta di esportazione.	**`your-export-job-name`**
JobStatus	No		I valori validi sono: `SUBMITTED | IN_PROGRESS | COMPLETED`

Nome	Obbligatorio?	Description	Valore
			_WITH_ERRORS | COMPLETED | FAILED
S3Uri	Sì	Parte di un OutputDataConfigoggetto. L'URI Amazon S3 del bucket di destinazione in cui verranno scaricati i dati esportati.	s3://amzn-s3-demo-bucket/ **EXPORT-JOB** /
KmsKeyId	Sì	Parte di un oggetto. OutputDataConfig L'ARN della AWS KMS chiave utilizzata per proteggere il bucket Amazon S3.	arn:aws:kms: **region-of-bucket:123456789012** :key/**1234abcd-12ab-34cd-56ef-1234567890ab**

Example: corpo di una richiesta di descrizione dell'esportazione effettuata utilizzando l'API REST FHIR

In caso di successo, riceverai la seguente risposta 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 Operazione FHIR per HealthLake

L'`$inquire`operazione consente di verificare lo stato di una richiesta di autorizzazione preventiva inviata in precedenza. Questa operazione implementa la Guida all'implementazione del Da Vinci Prior Authorization Support (PAS), che fornisce un flusso di lavoro standardizzato basato su FHIR per recuperare l'attuale decisione di autorizzazione.

Come funziona

- Invia richiesta: invii un pacchetto FHIR contenente il reclamo che desideri verificare e le informazioni di supporto
- Cerca: HealthLake cerca il corrispondente ClaimResponse nel tuo archivio dati
- Recupera: viene recuperato lo stato di autorizzazione più recente
- Rispondi: ricevi una risposta immediata con lo stato dell'autorizzazione corrente (in coda, approvata, negata, ecc.)

> **Note**
>
> `$inquire`è un'operazione di sola lettura che recupera lo stato di autorizzazione esistente. Non modifica o aggiorna alcuna risorsa nell'archivio dati.

Endpoint API

```
POST /datastore/{datastoreId}/r4/Claim/$inquire
Content-Type: application/fhir+json
```

Struttura della richiesta

Requisiti del pacchetto

La tua richiesta deve essere una risorsa FHIR Bundle con:

- Bundle.type: Deve essere `"collection"`
- bundle.entry: deve contenere esattamente una risorsa Claim con:
  - `use = "preauthorization"`
  - `status = "active"`
- Risorse referenziate: tutte le risorse a cui fa riferimento il reclamo devono essere incluse nel pacchetto

> **Interrogazione per esempio**
>
> Le risorse del tuo pacchetto di input fungono da modello di ricerca. HealthLake utilizza le informazioni fornite per individuare il corrispondente ClaimResponse.

Risorse obbligatorie

Risorsa	Cardinalità	Profilo	Description
Reclamo	1	Richiesta di reclamo PAS	L'autorizzazione preventiva su cui stai chiedendo informazioni
La paziente	1	Paziente beneficiario PAS	Informazioni demografiche sui pazienti
Organizzazione (assicuratore)	1	Organizzazione assicurativa PAS	Compagnia assicurativa
Organizzazione (fornitore)	1	Organizzazione richiedente PAS	Operatore sanitario che ha inviato la richiesta

Criteri di ricerca importanti

HealthLake ricerche per l' ClaimResponse utilizzo:

- Riferimento del paziente riportato nel reclamo
- Riferimento dell'assicuratore riportato nel reclamo
- Riferimento al fornitore riportato nel reclamo
- Data di creazione del reclamo (come filtro temporale)

> **Solo domande specifiche per il paziente**
>
> Tutte le richieste devono essere legate a un paziente specifico. Non sono consentite domande a livello di sistema senza identificazione del paziente.

Richiesta di esempio

```
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 della risposta

Risposta riuscita (200 OK)

Riceverai un pacchetto PAS Inquiry Response contenente:

- ClaimResponsecon lo stato di autorizzazione corrente; multiplo ClaimResponsese corrisponde ai criteri di ricerca
- Tutte le risorse originali contenute nella tua richiesta (riportate all'indietro)
- Timestamp di quando è stata assemblata la risposta

Possibili esiti ClaimResponse

Outcome	Description
queued	La richiesta di autorizzazione è ancora in attesa di revisione

Outcome	Description
`complete`	La decisione di autorizzazione è stata presa (verifica se `disposition` approvata/negata)
`error`	Si è verificato un errore durante l'elaborazione
`partial`	Autorizzazione parziale concessa


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

Risposte agli errori

400 Richiesta non valida

Restituito quando il formato della richiesta non è valido o la convalida fallisce.

```
{
```

```
    "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 - Autorizzazione negata

Restituito quando le credenziali di autenticazione sono mancanti o non valide.

```
{
    "resourceType": "OperationOutcome",
    "issue": [
        {
            "severity": "error",
            "code": "forbidden",
            "diagnostics": "Invalid authorization header"
        }
    ]
}
```

403 Non consentito

Restituito quando l'utente autenticato non è autorizzato ad accedere alla risorsa richiesta.

```
{
    "resourceType": "OperationOutcome",
    "issue": [
        {
            "severity": "error",
            "code": "exception",
            "diagnostics": "Insufficient SMART scope permissions."
        }
    ]
}
```

400 Quando non ne viene trovato nessuno

Restituito quando non ClaimResponse viene trovata alcuna corrispondenza per la richiesta.

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

4.1.5 Tipo di supporto non supportato

Restituito quando l'intestazione Content-Type non è application/fhir+json.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "value",
    "diagnostics": "Incorrect MIME-type. Update request Content-Type header."
  }]
}
```

429 Troppe richieste

Restituito quando vengono superati i limiti di velocità.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "throttled",
    "diagnostics": "Rate limit exceeded. Please retry after some time."
  }]
}
```

Regole di convalida

HealthLake esegue una convalida completa della richiesta:

Convalida del pacchetto

- Deve essere conforme al profilo PAS Inquiry Request Bundle
- `Bundle.type`deve essere `"collection"`
- Deve contenere esattamente una risorsa Claim
- Tutte le risorse referenziate devono essere incluse nel pacchetto

Convalida del reclamo

- Deve essere conforme al profilo PAS Claim Inquiry
- `Claim.use`deve essere `"preauthorization"`
- `Claim.status`deve essere `"active"`
- Campi obbligatori:`patient`,`insurer`,`provider`, `created`

Convalida delle risorse

- Tutte le risorse devono essere conformi ai rispettivi profili di richiesta PAS
- Devono essere presenti le risorse di supporto necessarie (paziente, organizzazione assicurativa, organizzazione fornitrice)
- I riferimenti incrociati devono essere validi e risolvibili all'interno del pacchetto

Specifiche prestazionali

Metrica	Specifiche
Limite di conteggio delle risorse	500 risorse per pacchetto
Limite di dimensione del pacchetto	Massimo 5 MB

Autorizzazioni richieste

Per utilizzare l'`$inquire`operazione, assicurati che il tuo ruolo IAM abbia:

- `healthlake:InquirePreAuthClaim`- Per richiamare l'operazione

SMART su FHIR Scopes

Ambiti minimi richiesti:

- SMART v1: `user/ClaimResponse.read`
- SMART versione 2: `user/ClaimResponse.s`

Note importanti sull'implementazione

Comportamento di ricerca

Quando invii una richiesta, HealthLake cerca ClaimResponse utilizzando:

- Riferimento del paziente riportato nel reclamo inserito
- Riferimento dell'assicuratore riportato nella Richiesta inserita
- Riferimento al fornitore riportato nel reclamo inserito
- Data di creazione in base alla richiesta di immissione (come filtro temporale)

Corrispondenze multiple: se più ClaimResponses corrispondono ai criteri di ricerca, HealthLake restituisce tutti i risultati corrispondenti. È necessario utilizzare il `ClaimResponse.created` timestamp più recente per identificare lo stato più recente.

Reclami aggiornati

Se hai inviato più aggiornamenti alla stessa autorizzazione preventiva (ad esempio, Claim v1.1, v1.2, v1.3), l'`$inquire`operazione recupererà quelli ClaimResponse associati alla versione più recente in base ai criteri di ricerca forniti.

Operazione di sola lettura

L'`$inquire`operazione:

- Recupera lo stato di autorizzazione esistente
- Restituisce il più recente ClaimResponse
- Non modifica o aggiorna alcuna risorsa
- Non crea nuove risorse

- Non attiva una nuova elaborazione delle autorizzazioni

Esempio di workflow

Tipico flusso di lavoro di richiesta di autorizzazione preventiva

```
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"
```

Operazioni correlate

- `Claim/$submit`- Invia una nuova richiesta di autorizzazione preventiva o aggiornane una esistente
- `Patient/$everything`- Recupera i dati completi del paziente per il contesto di autorizzazione preventiva

Recupero dei dettagli concettuali con **`$lookup`**

AWS HealthLake ora supporta l'`$lookup`operazione per CodeSystem le risorse, consentendoti di recuperare dettagli su un concetto specifico in un sistema di codice fornendo informazioni identificative come il codice. Questa operazione è particolarmente utile quando è necessario:

- Recuperare informazioni dettagliate su codici medici specifici
- Convalida i significati e le proprietà dei codici
- Accedi alle definizioni e alle relazioni dei concetti
- Supporta il processo decisionale clinico con dati terminologici accurati

Utilizzo

L'`$lookup`operazione può essere richiamata sulle CodeSystem risorse utilizzando i metodi GET e POST:

Operazioni supportate

```
GET [base]/CodeSystem/$lookup?system=http://snomed.info/
sct&code=73211009&version=20230901
POST [base]/CodeSystem/$lookup
```

Parametri supportati

HealthLake supporta un sottoinsieme di parametri FHIR R4: `$lookup`

Parametro	Tipo	Campo obbligatorio	Description
`code`	code	Sì	Il codice concettuale che stai cercando (ad esempio, «71620000" in SNOMED CT)
`system`	uri	Sì	L'URL canonico del sistema di codici (ad esempio, "http://snomed.info/sct «)
`version`	stringa	No	Versione specifica del sistema di codice

Esempi

Richiesta GET

```
GET [base]/CodeSystem/$lookup?system=http://snomed.info/
sct&code=71620000&version=2023-09
```

Richiesta 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"
    }
  ]
}
```

Risposta di esempio

L'operazione restituisce una risorsa Parameters contenente i dettagli del concetto:

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

Parametri di risposta

La risposta include i seguenti parametri, se disponibili:

Parametro	Tipo	Description
`name`	stringa	Nome del sistema di codici
`version`	stringa	Versione del sistema di codici
`display`	stringa	Visualizza il nome del concetto
`designation`	BackboneElement	Rappresentazioni aggiuntive per questo concetto.

Parametro	Tipo	Description
`property`	BackboneElement	Proprietà aggiuntive del concetto (definizione, relazioni, ecc.)

Comportamento

L'`$lookup`operazione:

1. Convalida i parametri richiesti (`codeesystem`)
2. Cerca il concetto all'interno del sistema di codice specificato memorizzato nel datastore
3. Restituisce informazioni dettagliate sul concetto, tra cui nome visualizzato, designazioni e proprietà.
4. Supporta ricerche specifiche per versione quando viene fornito il parametro `version`
5. Funziona solo su sistemi di codice archiviati in modo esplicito nel datastore HealthLake

Gestione errori

L'operazione gestisce le seguenti condizioni di errore:

- 400 Bad Request: `$lookup` operazione non valida (richiesta non conforme o parametri obbligatori mancanti)
- 404 Not Found: Sistema di codice non trovato o codice non trovato nel sistema di codice specificato

Avvertenze

In questa versione, non sono supportati:

- `$lookup`operazione mediante chiamata a server terminologici esterni
- `$lookup`operazione su CodeSystems gestita da HealthLake ma non archiviata in modo esplicito nel datastore

Per ulteriori informazioni sulle specifiche `$lookup` operative, consultate la documentazione di FHIR R4. CodeSystem `$lookup`

$member-addoperazione per HealthLake

L'$member-addoperazione FHIR aggiunge un membro (paziente) a una risorsa del Gruppo, in particolare a una lista di attribuzione dei membri. Questa operazione fa parte della DaVinci Member Attribution Implementation Guide e supporta il processo di riconciliazione per la gestione delle attribuzioni dei membri.

Operazione Endpoint

```
POST [base]/datastore/{datastoreId}/r4/Group/{groupId}/$member-add
Content-Type: application/json
```

Parameters

L'operazione accetta una risorsa FHIR Parameters con le seguenti combinazioni di parametri:

Opzioni dei parametri

È possibile utilizzare una delle seguenti combinazioni di parametri:

Opzione 1: Member ID + Provider NPI

- `memberId` + `providerNpi`
- `memberId` + `providerNpi` + `attributionPeriod`

Opzione 2: riferimento per il paziente e riferimento per il fornitore

- `patientReference` + `providerReference`
- `patientReference` + `providerReference` + `attributionPeriod`

Dettagli dei parametri

MemberID (opzionale)

L'identificatore del membro da aggiungere al gruppo.

Tipo: identificatore

Sistema: sistema di identificazione del paziente

```
{
  "name": "memberId",
  "valueIdentifier": {
    "system": "http://example.org/patient-id",
    "value": "patient-new"
  }
}
```

ProviderNPI (opzionale)

Il National Provider Identifier (NPI) del provider attribuito.

Tipo: identificatore

Sistema: http://terminology.hl7. org/CodeSystem/NPI

```
{
  "name": "providerNpi",
  "valueIdentifier": {
    "system": "http://terminology.hl7.org/CodeSystem/NPI",
    "value": "1234567890"
  }
}
```

Riferimento per il paziente (opzionale)

Riferimento diretto alla risorsa per il paziente da aggiungere.

Tipo: riferimento

```
{
  "name": "patientReference",
  "valueReference": {
    "reference": "Patient/patient-123"
  }
}
```

ProviderReference (opzionale)

Riferimento diretto alla risorsa del provider.

Tipo: riferimento

```
{
  "name": "providerReference",
  "valueReference": {
    "reference": "Practitioner/provider-456"
  }
}
```

Periodo di attribuzione (opzionale)

Il periodo di tempo durante il quale il paziente viene attribuito al fornitore.

Tipo: Periodo

```
{
  "name": "attributionPeriod",
  "valuePeriod": {
    "start": "2024-07-15",
    "end": "2025-07-14"
  }
}
```

Esempi di richieste

Utilizzo dell'ID membro e del provider NPI

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "memberId",
      "valueIdentifier": {
        "system": "http://example.org/patient-id",
        "value": "patient-new"
      }
    },
    {
      "name": "providerNpi",
      "valueIdentifier": {
        "system": "http://terminology.hl7.org/CodeSystem/NPI",
        "value": "1234567890"
      }
    },
```

```
    {
      "name": "attributionPeriod",
      "valuePeriod": {
        "start": "2024-07-15",
        "end": "2025-07-14"
      }
    }
  ]
}
```

Utilizzo dei riferimenti di pazienti e fornitori

```
{
  "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 della risposta

Risposta di aggiunta riuscita

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

Risposte agli errori

Sintassi di richiesta non valida

Stato HTTP: 400 Richiesta errata

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "invalid",
      "details": {
        "text": "Invalid parameter combination provided"
      }
    }
  ]
}
```

Risorsa non trovata

Stato HTTP: 404 non trovato

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
```

```
      "code": "not-found",
      "details": {
        "text": "Resource not found."
      }
    }
  ]
}
```

Conflitto di versione

Stato HTTP: conflitto 409

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "conflict",
      "details": {
        "text": "Resource version conflict detected"
      }
    }
  ]
}
```

Stato di attribuzione non valido

Stato HTTP: 422 Entità non processabile

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

Regole aziendali

Convalida dello stato di attribuzione

L'operazione può essere eseguita solo quando lo stato di attribuzione del gruppo è:

- `draft`
- `open`

Le operazioni non sono consentite quando lo stato è`final`.

Prevenzione della duplicazione dei membri

Il sistema impedisce l'aggiunta di membri duplicati in base alla combinazione unica di:

- Identificatore del membro
- Identificativo del pagatore
- Identificatore di copertura

Convalida del periodo di copertura

Quando ne `attributionPeriod` viene fornita una, questa deve rientrare nei limiti del periodo di copertura del socio. Il sistema consentirà di:

- Cerca la risorsa Coverage del socio
- Usa la copertura più recente (VersionID più alto) se ne esistono più
- Verifica che il periodo di attribuzione non superi il periodo di copertura

Convalida del riferimento

Quando vengono forniti sia l'ID che il riferimento per la stessa risorsa (paziente o fornitore), il sistema verifica che corrispondano alla stessa risorsa.

Quando vengono forniti entrambi i campi ID e reference.identifier per la stessa risorsa (paziente o fornitore), viene generato un errore.

Autenticazione e autorizzazione

L'operazione richiede l'autorizzazione SMART on FHIR per:

- Autorizzazioni di lettura: per convalidare le risorse di pazienti, fornitori e gruppi
- Autorizzazioni di ricerca: per trovare risorse in base all'identificatore
- Aggiorna le autorizzazioni: per modificare la risorsa del gruppo

Comportamento operativo

Aggiornamenti delle risorse

- Aggiorna l'ID della versione della risorsa del gruppo
- Crea una voce di cronologia con lo stato originale della risorsa prima dell'operazione
- Aggiunge le informazioni sui membri all'array Group.member con:
  - Riferimento al paziente in entity.reference
  - Periodo di attribuzione in periodo
  - Informazioni sulla copertura e sul fornitore nei campi di estensione

Fasi di convalida

- Convalida dei parametri: garantisce combinazioni di parametri valide
- Esistenza di risorse: verifica l'esistenza di risorse per pazienti, fornitori e gruppi
- Stato di attribuzione: conferma che lo stato del gruppo consente modifiche
- Duplicate Check: impedisce l'aggiunta di membri esistenti
- Convalida della copertura: garantisce che il periodo di attribuzione rientri nei limiti di copertura

Limitazioni

- Tutte le risorse referenziate devono esistere all'interno dello stesso datastore
- L'operazione funziona solo con le risorse del Member Attribution List Group
- I periodi di attribuzione devono rientrare nei limiti di copertura
- Non è possibile modificare i gruppi con lo stato «finale»

`$member-match`operazione per HealthLake

AWS HealthLake ora supporta l'`$member-match`operazione Patient Resources, consentendo alle organizzazioni sanitarie di trovare l'identificatore univoco di un membro in diversi sistemi sanitari utilizzando informazioni demografiche e di copertura. Questa operazione è essenziale per raggiungere la conformità CMS e facilitare lo scambio sicuro payer-to-payer dei dati, mantenendo al contempo la privacy dei pazienti.

Questa operazione è particolarmente utile quando è necessario:

- Consentire lo scambio sicuro di dati sanitari tra le organizzazioni

- Mantieni la continuità dell'assistenza ai pazienti tra diversi sistemi
- Supporta i requisiti di conformità CMS
- Facilita l'identificazione accurata dei membri attraverso le reti sanitarie

Utilizzo

L'`$member-match`operazione può essere richiamata sulle risorse del paziente utilizzando il metodo POST:

```
POST [base]/Patient/$member-match
```

Parametri supportati

HealthLake supporta i seguenti parametri FHIR`$member-match`:

Parametro	Tipo	Obbligatorio	Predefinita	Description
MemberPatient	Paziente	Sì	—	Risorsa per il paziente contenente informazioni demografiche relative al membro da abbinare
CoverageToMatch	Copertura	Sì	—	Risorsa di copertura che verrà utilizzata per il confronto con i record esistenti
CoverageToLink	Copertura	No	—	Risorsa di copertura da collegare durante il processo di abbinamento
Consenso	Consenso	No	—	Risorsa di consenso a fini di autorizzazione

Esempi

Richiesta POST con parametri

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

```
}
```

Risposta di esempio

L'operazione restituisce una risorsa Parameters contenente i risultati corrispondenti:

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

Parametri di risposta

La risposta include i seguenti parametri quando viene trovata una corrispondenza:

Parametro	Tipo	Description
MemberIdentifier	Identificatore	L'identificatore univoco per il membro corrispondente
MemberId	Documentazione di riferimento	Riferimento alla risorsa Patient
Match Algoritm	Stringa	Tipo di algoritmo di corrispondenza utilizzato (EXACT_MATCH, STRONG_MATCH o DEMOGRAPHIC_MATCH)
Dettagli della partita	Stringa	Informazioni dettagliate sul processo di abbinamento
Campi corrispondenti	Stringa	Elenco di campi specifici che sono stati abbinati correttamente

Algoritmi di corrispondenza

L'`$member-match`API utilizza un approccio di abbinamento a più livelli per garantire un'identificazione accurata dei membri:

EXACT_MATCH

- Utilizza l'identificatore del paziente combinato con la copertura SubscriberId
- Fornisce il massimo livello di confidenza per l'abbinamento dei membri

STRONG_MATCH

- Utilizza Patient Identifier con informazioni minime sulla copertura
- Offre un'elevata affidabilità quando non vengono soddisfatti i criteri di corrispondenza esatti

ABBINAMENTO DEMOGRAFICO

- Si basa su informazioni demografiche di base
- Utilizzato quando non è possibile la corrispondenza basata su identificatori

Comportamento

L'operazione: `$member-match`

- Accetta come input i dati demografici dei pazienti, i dettagli sulla copertura e le informazioni facoltative sul consenso
- Restituisce un identificatore di membro univoco che può essere utilizzato per le interazioni successive
- Implementa la corrispondenza a più livelli (esatta, forte, demografica) per garantire l'identificazione accurata dei membri nei diversi sistemi sanitari
- Salva tutte le informazioni di consenso fornite per scopi di autorizzazione futuri
- Supporta lo scambio sicuro di payer-to-payer dati mantenendo al contempo la privacy del paziente
- Conforme ai requisiti CMS per lo scambio di dati sanitari

Autorizzazione

L'API utilizza il protocollo di autorizzazione SMART on FHIR con i seguenti ambiti richiesti:

- `system/Patient.read`
- `system/Coverage.read`
- `system/Organization.read`(condizionale)
- `system/Practitioner.read`(condizionale)
- `system/PractitionerRole.read`(condizionale)
- `system/Consent.write`(condizionale)

Gestione errori

L'operazione gestisce le seguenti condizioni di errore:

- `400 Bad Request`: `$member-match` operazione non valida (richiesta non conforme o parametri obbligatori mancanti)
- `422 Unprocessable Entity`: nessuna corrispondenza o più corrispondenze trovate

$member-removeoperazione per HealthLake

L'`$member-remove`operazione consente di rimuovere membri da una lista di attribuzione dei membri FHIR (risorsa di gruppo) in. AWS HealthLake Questa operazione fa parte della DaVinci Member Attribution Implementation Guide e supporta il processo di riconciliazione per la gestione delle attribuzioni dei membri.

Prerequisiti

- AWS HealthLake Datastore FHIR
- Autorizzazioni IAM appropriate per le operazioni HealthLake
- Elenco di attribuzione dei membri (risorsa di gruppo) in bozza o in stato aperto

Dettagli dell'operazione

Endpoint

```
POST /Group/{id}/$member-remove
```

Content Type

```
application/fhir+json
```

Parameters

L'operazione accetta una risorsa FHIR Parameters con i seguenti parametri opzionali:

Parametro	Cardinalità	Tipo	Description
memberId	0.1..	Identificatore	Identificativo aziendale del membro da rimuovere
ProviderNPI	0.1..	Identificatore	NPI del fornitore attribuito
Riferimento per il paziente	0.. 1	Documentazione di riferimento	Riferimento diretto alla risorsa per i pazienti
Riferimento del fornitore	0.1..	Documentazione di riferimento	Riferimento diretto alla risorsa del Fornitore (professionista o Practitio nerRole organizzazione)

Parametro	Cardinalità	Tipo	Description
Riferimento alla copertura	0.1..	Documentazione di riferimento	Riferimento alla risorsa Coverage

Combinazioni di parametri supportate

Sono supportate le seguenti combinazioni di parametri:

- `memberId`solo: rimuove tutte le attribuzioni per il membro specificato
- `memberId`+ `providerNpi` - Rimuove le attribuzioni per la specifica combinazione membro-provider
- `patientReference`solo: rimuove tutte le attribuzioni per il paziente specificato
- `patientReference`+ `providerReference` - Rimuove le attribuzioni per la combinazione specifica paziente-operatore
- `patientReferenceproviderReference`+ `coverageReference` - Rimuove l'attribuzione specifica in base al paziente, al fornitore e alla copertura

Esempio di richiesta

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "patientReference",
      "valueReference": {
        "reference": "Patient/12345"
      }
    },
    {
      "name": "providerReference",
      "valueReference": {
        "reference": "Practitioner/67890"
      }
    }
  ]
}
```

Risposta

Risposta riuscita


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


Comportamento

Requisiti di status

- L'operazione funziona solo su elenchi di attribuzione con stato o `draft` `open`
- Gli elenchi con `final` stato rifiuteranno l'operazione con un errore 422

Procedura di rimozione dei membri

- Bozze di elenchi di status: i membri sono contrassegnati come inattivi (`inactive: true`) e la loro `changeType` estensione è aggiornata a `changed`
- Liste di stato aperte: comportamento simile alla bozza di stato
- Elenchi di stato finali: l'operazione è rifiutata

Convalida

I riferimenti vengono convalidati per garantire che esistano nel HealthLake datastore

Se vengono forniti sia l'identificatore che il riferimento per lo stesso tipo di risorsa, devono corrispondere alla stessa risorsa

Le combinazioni di parametri vengono convalidate in base ai modelli supportati

Gestione errori

Risposte di errore comuni

Risorsa non trovata (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"]
    }
  ]
}
```

Stato finale dell'elenco di attribuzione (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"]
    }
  ]
}
```

Operazione non valida (400)

Restituito quando le combinazioni di parametri non sono valide o non sono corrette.

Trovate più corrispondenze (412)

Restituito quando i parametri forniti corrispondono a più membri nell'elenco di attribuzione.

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "processing",
      "diagnostics": "Multiple members found matching the criteria"
    }
  ]
}
```

Best practice

- Usa parametri specifici: quando possibile, utilizza la combinazione di parametri più specifica per evitare rimozioni involontarie
- Stato della lista di controllo: verifica lo stato dell'elenco di attribuzione prima di tentare le rimozioni
- Gestisci gli errori con garbo: implementa una corretta gestione degli errori per tutte le possibili condizioni di errore

- Convalida i riferimenti: assicurati che tutte le risorse referenziate esistano prima di effettuare la richiesta

Eliminazione delle risorse del comparto paziente con **$purge**

AWS HealthLake supporta l'`$purge`operazione, consentendo l'eliminazione permanente di tutte le risorse all'interno del compartimento del paziente. Questa operazione è particolarmente utile quando è necessario:

- Rimuovere tutti i dati associati a un paziente
- Rispetta le richieste di rimozione dei dati dei pazienti
- Gestisci il ciclo di vita dei dati dei pazienti
- Esegui una pulizia completa delle cartelle cliniche dei pazienti

Utilizzo

L'`$purge`operazione può essere richiamata sulle risorse del paziente:

```
POST [base]/Patient/[ID]/$purge?deleteAuditEvent=true
```

Parameters

Parametro	Tipo	Obbligatorio	Predefinita	Description
`deleteAuditEvent`	booleano	No	false	Se impostato su true, elimina gli eventi di controllo associati
`_since`	stringa	No	Ora di creazione del datastore	Una volta inserito, seleziona l'orario limite iniziale per trovare le risorse in base all'ora dell'ultima modifica. Non può essere utilizzato con start o end
`start`	stringa	No	Ora di creazione del datastore	Una volta inserito, seleziona l'orario limite per la ricerca delle risorse in base all'ora dell'ultima modifica. Può essere usato con fine

Parametro	Tipo	Obbligatorio	Predefinita	Description
end	stringa	No	Ora di invio del lavoro	Una volta inserito, seleziona l'orario limite finale per trovare le risorse in base all'ora dell'ultima modifica

Esempi

Richiesta di esempio

```
POST [base]/Patient/example-patient/$purge?deleteAuditEvent=true
```

Risposta di esempio

```
{
  "resourceType": "OperationOutcome",
  "id": "purge-job",
  "issue": [
    {
      "severity": "information",
      "code": "informational",
      "diagnostics": "Purge job started successfully. Job ID:
 12345678-1234-1234-1234-123456789012"
    }
  ]
}
```

Stato di un processo

Per verificare lo stato di un processo di eliminazione:

```
GET [base]/$purge/[jobId]
```

L'operazione restituisce informazioni sullo stato del lavoro:

```
{
      "datastoreId": "36622996b1fcecb7e12ee2ee085308d3",
      "jobId": "3dd1c7a5b6c0ef8c110f566eb87e2ef9",
```

```
    "status": "COMPLETED",
    "submittedTime": "2025-10-31T18:43:21.822Z"
}
```

Comportamento

L'`$purge`operazione:

1. Processi in modo asincrono per gestire più risorse
2. Mantiene le transazioni ACID per l'integrità dei dati
3. Fornisce il monitoraggio dello stato del lavoro con il conteggio delle eliminazioni delle risorse
4. Rimuove definitivamente tutte le risorse presenti nel compartimento del paziente
5. Include una registrazione di controllo completa delle attività di eliminazione
6. Supporta l'eliminazione selettiva degli eventi di controllo

Registrazione degli audit

Le `$purge` operazioni vengono registrate come Start FHIRBulk DeleteJob e Descrivi FHIRBulk DeleteJob con informazioni dettagliate sull'operazione.

Limitazioni

- Le risorse eliminate non verranno visualizzate nelle risposte di ricerca
- Le risorse eliminate potrebbero essere temporaneamente inaccessibili durante l'elaborazione
- Tutte le risorse presenti nel compartimento del paziente vengono rimosse definitivamente

`$questionnaire-package`Operazione FHIR per HealthLake

L'`$questionnaire-package`operazione recupera un pacchetto completo contenente un questionario FHIR e tutte le sue dipendenze necessarie per il rendering e l'elaborazione del questionario. Questa operazione implementa la Da Vinci Documentation Templates and Rules (DTR) Implementation Guide, che consente il rendering dinamico dei moduli per i requisiti di documentazione nei flussi di lavoro sanitari.

Come funziona

- Richiesta: inviate i parametri che identificano il questionario o i questionari necessari, insieme alla copertura e al contesto dell'ordine

- Recupera: HealthLake raccoglie il questionario e tutte le dipendenze (, librerie CQL, ecc.) ValueSets
- Package: Tutte le risorse sono raggruppate in un formato standardizzato
- Rispondi: Riceverai un pacchetto completo pronto per il rendering e la raccolta dei dati

Casi d'uso

- Documentazione di autorizzazione preventiva: raccoglie le informazioni cliniche necessarie per le richieste di autorizzazione preventiva
- Requisiti di copertura: raccogliere la documentazione necessaria per soddisfare i requisiti di copertura dei pagatori
- Clinical Data Exchange: struttura dei dati clinici da presentare ai pagatori
- Moduli dinamici: renderizza i questionari con dati precompilati sui pazienti e logica condizionale

Endpoint API

```
POST /datastore/{datastoreId}/r4/Questionnaire/$questionnaire-package
Content-Type: application/fhir+json
```

Parametri della richiesta

Parametri di input

Il corpo della richiesta deve contenere una risorsa FHIR Parameters con i seguenti parametri:

Parametro	Tipo	Cardinalità	Description
`coverage`	Copertura	1.. * (Obbligatorio)	Risorse/e di copertura per stabilire il membro e copertura della documentazione
`questionnaire`	canonico	0.. *	URL canonici per uno o più questionari specifici da restituire (la versione può includere)
`order`	Risorsa	0.. *	Ordina le risorse (DeviceRequest, ServiceRequest, MedicationRequest,

Parametro	Tipo	Cardinalità	Description
			Encounter, Appointment) per stabilire il contesto
`changedSince`	dateTime	0.1.	Se presenti, restituisce solo le risorse modificate dopo questo timestamp

Regole di convalida dei parametri

È necessario fornire almeno UNO dei seguenti elementi (oltre a quelli obbligatori`coverage`):

- Uno o più `questionnaire` canonici URLs
- Una o più risorse `order`

Combinazioni di richieste valide:

- `coverage` + `questionnaire`
- `coverage` + `order`
- `coverage` + `questionnaire` + `order`

Richiesta di esempio

```
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 della risposta

Risposta riuscita (200 OK)

L'operazione restituisce una risorsa FHIR Parameters contenente uno o più Package Bundle. Ogni pacchetto Package include:

Tipo di ingresso	Cardinalità	Description
Questionario	1	Il questionario da rendere
QuestionnaireResponse	0.1..	Risposta precompilata o parzialmente completata (se applicabile)
Libreria	0.. *	Librerie CQL contenenti logica condizionale e di precompilazione
ValueSet	0.. *	Espanso ValueSets (per scelte di risposta con <40 espansioni)

Example Esempio di risposta

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

Flusso di lavoro operativo

Come HealthLake elabora la tua richiesta

Quando chiami`$questionnaire-package`, HealthLake effettua le seguenti operazioni:

1. Identify Patient & Payer: estrae il paziente e l'organizzazione assicurativa dal parametro. `coverage`
2. Trova il questionario giusto:
   - Con **`questionnaire`** parametro: utilizza l'URL canonico che hai fornito
   - Con **`order`** parametro: corrisponde al codice dell'ordine (CPT/HCPCS/LOINC) e al pagatore per trovare il questionario appropriato
3. Raccogli le dipendenze: recupera automaticamente tutto ciò che serve per il rendering del questionario:
   - Librerie CQL - Logica per domande condizionali e preliminari
   - ValueSets- Scelte di risposta (espanse automaticamente se <40 opzioni)
   - QuestionnaireResponse- Eventuali risposte esistenti in corso o completate
4. Package tutto insieme:

- Raggruppa tutte le risorse (ogni risorsa è inclusa una sola volta)
- Filtra per `changedSince` timestamp, se fornito
- Aggiunge avvisi in `Outcome` caso di mancanza di risorse

Risultato: un pacchetto completo e autonomo pronto per il rendering.

Risposte agli errori

400 Richiesta non valida

Restituito quando la convalida della richiesta fallisce.

```
{
  "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 Dipendenza non riuscita

Restituito quando una risorsa dipendente non può essere recuperata.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "warning",
    "code": "not-found",
    "details": {
      "text": "Referenced Library 'http://example.org/fhir/Library/missing-library'
 could not be retrieved"
    }
  }]
}
```

401 - Autorizzazione negata

Restituito quando le credenziali di autenticazione sono mancanti o non valide.

403 Non consentito

Restituito quando l'utente autenticato non è autorizzato ad accedere alle risorse richieste.

406 Non accettabile

Restituito quando non è possibile fornire il tipo di contenuto richiesto.

409 Conflitto

Restituito in caso di conflitto di versione o concorrenza.

410 Andato

Restituito quando la risorsa richiesta è stata eliminata definitivamente.

429 Troppe richieste

Restituito quando vengono superati i limiti di velocità.

500 - Errore interno del server

Restituito quando si verifica un errore imprevisto del server.

501 non implementato

Restituito quando l'operazione richiesta non è ancora implementata.

Regole di convalida

Convalida dell'input

- `coverage`il parametro è obbligatorio (1.. * cardinalità)
- Almeno uno `questionnaire` o `order` deve essere fornito
- Tutte le risorse di copertura devono essere risorse FHIR valide
- Tutte le risorse dell'Ordine devono essere risorse FHIR valide
- Canonical URLs deve essere formattato correttamente
- `changedSince`deve essere un DateTime ISO 8601 valido

QuestionnaireResponse convalida

- `status`deve essere appropriato (`in-progress`,`completed`,`amended`)
- La struttura deve corrispondere al questionario di riferimento
- `basedOn`deve fare riferimento a risorse Order valide
- `subject`deve fare riferimento a risorse valide per i pazienti

Deduplicazione delle risorse

- Ogni risorsa appare solo una volta nel pacchetto
- Eccezione: è possibile includere entrambe versioni diverse della stessa risorsa
- Le risorse sono identificate in base all'URL e alla versione canonici

Specifiche prestazionali

Metrica	Specifiche
Limite di conteggio delle risorse	500 risorse per pacchetto
Limite di dimensione del pacchetto	Massimo 5 MB

Autorizzazioni richieste

Per utilizzare l'`$questionnaire-package`operazione, assicurati che il tuo ruolo IAM abbia:

- `healthlake:QuestionnairePackage`- Per richiamare l'operazione
- `healthlake:ReadResource`- Per recuperare il questionario e le risorse dipendenti
- `healthlake:SearchWithPost`- Per cercare risorse correlate QuestionnaireResponse

SMART su FHIR Scopes

Ambiti minimi richiesti:

- SMART v1: `user/Questionnaire.read user/Library.read user/ValueSet.read user/QuestionnaireResponse.read`

- SMART versione 2: `user/Questionnaire.rs user/Library.rs user/ValueSet.rs user/QuestionnaireResponse.rs`

Note importanti sull'implementazione

Strategia di recupero delle risorse

Priorità di identificazione del questionario:

- URL canonico (se il `questionnaire` parametro è stato fornito) - Priorità massima
- Analisi dell'ordine (se il `order` parametro è fornito):
  - Abbina i codici degli ordini (CPT, HCPCS, LOINC) alle politiche mediche dei paganti
  - Utilizza Coverage Payor per filtrare i questionari specifici per ciascun pagatore
  - Prendi in considerazione i codici dei motivi per un contesto aggiuntivo

Risoluzione delle dipendenze

Librerie CQL:

- Recuperato tramite l'`cqf-library`estensione sulle risorse del questionario
- Recupera ricorsivamente le librerie dipendenti tramite type `Library.relatedArtifact depends-on`
- Tutte le dipendenze della libreria sono incluse nel pacchetto

ValueSets:

- Si espandono automaticamente se contengono meno di 40 concetti
- ValueSets I più grandi sono inclusi senza espansione
- ValueSets a cui si fa riferimento sia nel Questionario che nelle risorse della Libreria sono incluse

QuestionnaireResponse prepopolazione

L'operazione può restituire un file QuestionnaireResponse con dati precompilati quando:

- Viene trovata una risposta esistente in corso o completata

- La logica CQL nelle librerie associate può estrarre dati dalle cartelle cliniche dei pazienti
- La risposta è collegata all'ordine e alla copertura pertinenti

Criteri di ricerca per QuestionnaireResponse:

Parametro di ricerca	Percorso FHIR	Description
`based-on`	`QuestionnaireResponse.basedOn`	Collegamenti a o ServiceRequest CarePlan
`patient`	`QuestionnaireResponse.subject`	Il paziente che è il soggetto
`questionnaire`	`QuestionnaireResponse.questionnaire`	Il questionario a cui si sta rispondendo

Filtraggio delle risorse modificato

Quando viene fornito il `changedSince` parametro:

- Sono incluse solo le risorse modificate dopo il timestamp specificato
- Se nessuna risorsa è stata modificata, ritorna `200 OK` con un pacchetto vuoto
- Utile per aggiornamenti incrementali e strategie di memorizzazione nella cache
- Il confronto dei timestamp utilizza il campo delle risorse `meta.lastUpdated`

Pacchetti multipli

L'operazione può restituire più Package Bundle quando:

- Vengono richiesti più questionari tramite canonical URLs
- Ordini multipli richiedono questionari diversi
- Sono applicabili versioni diverse dello stesso questionario

Ogni Package Bundle è autonomo con tutte le dipendenze necessarie.

Casi di utilizzo comune

Caso d'uso 1: documentazione di autorizzazione preventiva

Scenario: un fornitore deve raccogliere la documentazione per un'autorizzazione preventiva di ossigenoterapia domiciliare.

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

Risultato: restituisce un pacco con il questionario di ossigenoterapia, precompilato con i dati anagrafici dei pazienti e i codici di diagnosi dell'EHR.

Caso d'uso 2: recupero di una versione specifica del questionario

Scenario: un provider necessita di una versione specifica di un questionario per la conformità.

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "coverage",
      "resource": { /* Coverage resource */ }
    },
```

```
    {
      "name": "questionnaire",
      "valueCanonical": "http://example.org/fhir/Questionnaire/dme-request|3.1.0"
    }
  ]
}
```

Risultato: restituisce esattamente la versione 3.1.0 del questionario di richiesta DME con tutte le dipendenze.

Caso d'uso 3: verifica la presenza di aggiornamenti

Scenario: un provider desidera verificare se alcune risorse del questionario sono state aggiornate dall'ultimo recupero.

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

Risultato: restituisce solo le risorse che sono state modificate dopo il 1° marzo 2024 o un pacchetto vuoto se non è cambiato nulla.

Caso d'uso 4: ordini multipli

Scenario: un provider invia più richieste di assistenza che possono richiedere questionari diversi.

```
{
  "resourceType": "Parameters",
```

```
  "parameter": [
    {
      "name": "coverage",
      "resource": { /* Coverage resource */ }
    },
    {
      "name": "order",
      "resource": { /* ServiceRequest for imaging */ }
    },
    {
      "name": "order",
      "resource": { /* ServiceRequest for DME */ }
    }
  ]
}
```

Risultato: restituisce più Package Bundle, uno per ogni questionario applicabile.

Integrazione con Other Da Vinci IGs

Discovery dei requisiti di copertura (CRD)

Integrazione del flusso di lavoro:

- Il fornitore ordina un servizio nel proprio EHR
- Il CRD si blocca, verifica dei requisiti di copertura
- Il pagatore risponde indicando che è necessaria la documentazione
- Il provider chiama `$questionnaire-package` per recuperare il modulo di documentazione
- Il provider completa il questionario
- La documentazione viene inviata tramite PAS o CDex

Support di autorizzazione preventiva (PAS)

Integrazione del flusso di lavoro:

- Utilizzare `$questionnaire-package` per recuperare i requisiti di documentazione
- Completare il modulo QuestionnaireResponse con i dati clinici richiesti
- Inviare l'autorizzazione preventiva utilizzando `Claim/$submit` con il QuestionnaireResponse

- Controlla lo stato utilizzando `Claim/$inquire`

Data Exchange clinico (CDex)

Integrazione del flusso di lavoro:

- Il pagante richiede documentazione aggiuntiva per un reclamo
- Il fornitore utilizza `$questionnaire-package` per recuperare il modulo di raccolta dei dati strutturati
- Il fornitore completa il QuestionnaireResponse
- La documentazione viene inviata al pagatore tramite CDex il flusso di lavoro in allegato

Guida alla risoluzione dei problemi

Problema: nessun questionario restituito

Possibili cause:

- L'URL canonico non corrisponde a nessun questionario nell'archivio dati
- Il codice dell'ordine non corrisponde a nessun questionario nella politica medica del pagatore
- Il beneficiario della copertura non dispone di questionari associati

Soluzioni:

- Verifica che l'URL canonico sia corretto e che il questionario esista
- Verifica che i codici d'ordine (CPT/HCPCS) siano specificati correttamente
- Conferma che i questionari siano configurati per il pagatore o l'organizzazione

Problema: dipendenze mancanti nel pacchetto

Possibili cause:

- Libreria referenziata o ValueSet non esiste nell'archivio dati

- I riferimenti alla libreria sono interrotti o errati
- ValueSet espansione non riuscita

Soluzioni:

- Controlla il `Outcome` parametro per gli avvisi relativi alle risorse mancanti
- Verifica che tutte le risorse di riferimento esistano nel tuo archivio dati
- Assicurati che ValueSet URLs siano corretti e risolvibili

Problema: Package vuoto con ChangedSince

Possibili cause:

- Questo è il comportamento previsto: nessuna risorsa è stata modificata dopo il timestamp specificato

Soluzioni:

- Usa la versione cache del pacchetto
- Rimuovi il `changedSince` parametro per recuperare il pacchetto completo

Problema: QuestionnaireResponse non precompilato

Possibili cause:

- Nessun elemento esistente QuestionnaireResponse trovato
- La logica della libreria CQL non è in grado di estrarre i dati richiesti
- I dati del paziente sono mancanti o incompleti

Soluzioni:

- Questo è prevedibile: non tutti i questionari hanno una logica di prepopolazione
- Verificate che i dati del paziente esistano nell'archivio dati

- Rivedi la logica della libreria CQL per i requisiti di estrazione dei dati

Best practice

1. Usa Canonical URLs con le versioni

Specificate sempre i numeri di versione quando richiedete questionari specifici:

```
{
  "name": "questionnaire",
  "valueCanonical": "http://example.org/fhir/Questionnaire/dme|2.1.0"
}
```

Perché: garantisce la coerenza e previene modifiche impreviste quando i questionari vengono aggiornati.

2. Sfrutta ChangedSince per le prestazioni

Per i questionari a cui si accede di frequente, utilizza `changedSince` per ridurre al minimo il trasferimento dei dati:

```
{
  "name": "changedSince",
  "valueDateTime": "2024-03-10T15:30:00Z"
}
```

Perché: riduce la latenza e l'utilizzo della larghezza di banda recuperando solo le risorse aggiornate.

3. Includi informazioni complete sulla copertura

Fornisci dettagli completi sulla copertura per garantire la corretta selezione del questionario:

```
{
  "name": "coverage",
  "resource": {
    "resourceType": "Coverage",
    "beneficiary": { "reference": "Patient/123" },
    "payor": [{ "reference": "Organization/payer-abc" }],
    "class": [{ /* Group/plan information */ }]
  }
}
```

Perché: aiuta a HealthLake identificare i questionari e i requisiti specifici del pagatore.

Operazioni correlate

- `Claim/$submit`- Invia richieste di autorizzazione preventiva con documentazione completa
- `Claim/$inquire`- Verifica lo stato delle autorizzazioni precedenti inviate
- `ValueSet/$expand`- Espandi ValueSets per visualizzare le opzioni di risposta

`$submit`Operazione FHIR per HealthLake

L'`$submit`operazione consente di inviare elettronicamente richieste di autorizzazione preventiva ai pagatori per l'approvazione. Questa operazione implementa la Da Vinci Prior Authorization Support (PAS) Implementation Guide, che fornisce un flusso di lavoro standardizzato basato su FHIR per l'invio di autorizzazioni preventive.

Come funziona

- Invio: si invia un pacchetto FHIR contenente la richiesta di autorizzazione preventiva e i dati clinici di supporto
- Convalida: HealthLake convalida l'invio rispetto ai requisiti PAS
- Persiste: tutte le risorse sono archiviate nel tuo archivio dati HealthLake
- Rispondi: ricevi una risposta immediata con lo stato «in coda»
- Processo: la decisione di autorizzazione viene elaborata in modo asincrono dal pagatore

Endpoint API

```
POST /datastore/{datastoreId}/r4/Claim/$submit
Content-Type: application/fhir+json
```

Struttura della richiesta

Requisiti del pacchetto

La tua richiesta deve essere una risorsa FHIR Bundle con:

- Bundle.type: Deve essere `"collection"`
- bundle.entry: deve contenere esattamente una risorsa Claim con `use = "preauthorization"`

- Risorse referenziate: tutte le risorse a cui fa riferimento il reclamo devono essere incluse nel pacchetto

Risorse obbligatorie

Risorsa	Cardinalità	Profilo	Description
Reclamo	1	Reclamo PAS	La richiesta di autorizzazione preventiva
Paziente	1	Paziente PAS	Informazioni demografiche sui pazienti
Organizzazione (assicuratore)	1	Assicuratore PAS	Compagnia assicurativa
Organizzazione (fornitore)	1	Richiedente PAS	Fornitore di assistenza sanitaria che invia la richiesta
Copertura	1 o più	Copertura PAS	Dettagli della copertura assicurativa

Risorse opzionali

Risorsa	Cardinalità	Profilo	Description
Professionista	0 o più	Professionista PAS	Operatori sanitari
PractitionerRole	0 o più	PAS PractitionerRole	Ruoli del professionista
ServiceRequest	0 o più	PAS ServiceRequest	Servizi medici richiesti
DeviceRequest	0 o più	PAS DeviceRequest	Dispositivi medici richiesti

Risorsa	Cardinalità	Profilo	Description
MedicationRequest	0 o più	PAS MedicationRequest	Farmaci richiesti
DocumentReference	0 o più	PAS DocumentReference	Documentazione clinica di supporto

Richiesta di esempio

```
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 della risposta

Risposta di successo (200 OK)

Riceverai un pacchetto di risposta PAS contenente:

- ClaimResponsecon e `outcome: "queued"` `status: "active"`
- Tutte le risorse originali contenute nella tua richiesta
- Timestamp di conferma della ricezione

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

Risposte agli errori

400 Richiesta non valida

Restituito quando il formato della richiesta non è valido o non è valido.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "invalid",
    "diagnostics": "The provided payload was invalid and could not be parsed
 correctly."
  }]
}
```

412 Precondizione non riuscita

Restituito quando è già stata inviata la stessa richiesta di autorizzazione preventiva (è stato rilevato un invio duplicato).

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "processing",
    "diagnostics": "PreAuth Claim already exists"
  }]
}
```

Idempotenza

L'`$submit`operazione è idempotente. L'invio della stessa richiesta più volte non creerà richieste di autorizzazione preventiva duplicate. Riceverai invece un errore 412 che ti chiederà di utilizzare per `$inquire` verificare lo stato dell'invio originale.

422 Entità non processabile

Restituito quando la convalida FHIR fallisce.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "required",
    "diagnostics": "Bundle contains more than one preauthorization claim"
  }]
}
```

Regole di convalida

HealthLake esegue una convalida completa dell'invio:

Convalida del pacchetto

- Deve essere conforme al profilo PAS Request Bundle

- `Bundle.type`deve essere `"collection"`
- Può contenere più risorse per i reclami
- Tuttavia, deve contenere esattamente una risorsa Claim da utilizzare in preautorizzazione
  - E questa risorsa Claim deve essere la prima voce del pacchetto
- Tutte le risorse a cui si fa riferimento devono essere incluse nel pacchetto

Convalida del reclamo

- Deve essere conforme al profilo PAS Claim
- `Claim.use`deve essere `"preauthorization"`
- Campi obbligatori: `patient`,`insurer`,`provider`,`created`, `priority`
- Gli identificatori aziendali devono essere presenti e validi

Convalida delle risorse

- Tutte le risorse devono essere conformi ai rispettivi profili PAS
- Devono essere presenti le risorse di supporto necessarie (paziente, copertura, organizzazione)
- I riferimenti incrociati devono essere validi e risolvibili all'interno del pacchetto

Specifiche prestazionali

Metrica	Specifiche
Limite di dimensione del pacchetto	Massimo 5 MB
Limite di numero di risorse	500 risorse per pacchetto

Autorizzazioni richieste

Per utilizzare l'`$submit`operazione, è possibile utilizzare AWS Sigv4 o SMART su FHIR:

- Assicurati che il tuo ruolo IAM abbia: `healthlake:SubmitPreAuthClaim` - Per chiamare l'operazione

SMART su FHIR Scopes

Ambiti minimi richiesti:

- SMART v1: `user/Claim.write & <all_resourceTypes_in_Bundle>.write`
- SMART versione 2: `user/Claim.c & <all_resourceTypes_in_Bundle>.c or system/*.*`

Note importanti sull'implementazione

Persistenza delle risorse

- Tutte le voci del Bundle vengono memorizzate come risorse FHIR individuali nell'archivio dati
- I dati forniti dal cliente vengono conservati quando vengono forniti IDs
- La cronologia delle versioni viene conservata a fini di controllo
- Il rilevamento dei duplicati previene i conflitti di risorse

Comportamento di elaborazione

- Ogni invio valido restituisce esattamente ClaimResponse un `"queued"` risultato
- Gli invii non validi restituiscono codici di stato 400 o 422 con informazioni dettagliate sull'errore
- Gli errori di sistema restituiscono i codici di stato 5xx appropriati
- Tutti gli invii andati a buon fine restituiscono lo stato 200 con un messaggio Risolto ClaimResponse

Requisiti del pacchetto

- `Bundle.entry.fullUrl`i valori devono essere REST URLs o format `"urn:uuid:[guid]"`
- Tutti gli invii GUIDs devono essere unici (ad eccezione delle stesse istanze di risorse)
- Le risorse referenziate devono esistere all'interno del pacchetto o essere risolvibili

Operazioni correlate

- `Claim/$inquire`- Verifica lo stato di una richiesta di autorizzazione preventiva inviata
- `Patient/$everything`- Recupera i dati completi del paziente per il contesto di autorizzazione preventiva

Convalida delle risorse FHIR con **$validate**

AWS HealthLake ora supporta il `$validate` funzionamento delle risorse FHIR, consentendoti di convalidare una risorsa rispetto alle specifiche FHIR e di verificarne la conformità a un profilo specifico o a una definizione di risorsa di base senza eseguire alcuna operazione di storage. Questa operazione è particolarmente utile quando è necessario:

- Convalidare i requisiti di conformità FHIR CMS
- Testa le risorse prima di utilizzarle in produzione
- Fornisci feedback di convalida in tempo reale mentre gli utenti modificano i dati clinici
- Riduci gli invii di dati non validi per crearli e aggiornarli APIs

Utilizzo

L'`$validate`operazione può essere richiamata sulle risorse FHIR utilizzando i metodi POST:

Operazioni supportate

```
POST [base]/[type]/[id]/$validate
POST [base]/[type]/$validate
```

Payload supportati

Risorsa di parametri

HealthLake supporta i seguenti `$validate` parametri FHIR:

Parametro	Tipo	Campo obbligatorio	Description
`resource`	Risorsa	Sì	La risorsa da convalidare
`profile`	canonico	No	URL canonico del profilo con cui eseguire la convalida
`mode`	code	No	Modalità di convalida:, oppure `create` `update`

Risorsa diretta con parametri di interrogazione

Parametro	Tipo	Campo obbligatorio	Description
`profile`	canonico	No	URL canonico del profilo con cui eseguire la convalida
`mode`	code	No	Modalità di convalida:, oppure `create` `update`

Esempi

Richiesta POST di risorsa con payload ID e parametri

```
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"
    }
  ]
}
```

Richiesta POST per il tipo di risorsa e il payload dei parametri

```
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"
    }
  ]
}
```

Richiesta di risorsa POST con ID e payload diretto della risorsa

```
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"
}
```

Richiesta POST per tipo di risorsa e payload diretto della risorsa

```
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"
}
```

Risposta di esempio

L'operazione restituisce una OperationOutcome risorsa con risultati di convalida:

```
{
  "resourceType": "OperationOutcome",
```

```
  "issue": [
    {
      "severity": "information",
      "code": "informational",
      "diagnostics": "Validation successful"
    }
  ]
}
```

Esempio di risposta con errori di convalida

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

Comportamento

L'`$validate`operazione:

1. Convalida la risorsa rispetto alla specifica FHIR e alla definizione della risorsa di base
2. Verifica la conformità ai profili specificati quando viene fornito il parametro `profile`
3. Esegue la convalida in base alla modalità specificata (o) `create` `update`
4. Restituisce risultati di convalida dettagliati, inclusi errori, avvisi e messaggi informativi
5. Non esegue alcuna operazione di archiviazione, solo convalida
6. Restituisce HTTP 200 OK quando è possibile eseguire la convalida, indipendentemente dal fatto che vengano rilevati problemi di convalida

Modalità di convalida

- create: convalida la risorsa come se fosse in fase di creazione (nuova risorsa)
- update: convalida la risorsa come se fosse in fase di aggiornamento (risorsa esistente)

Gestione errori

L'operazione restituisce:

- 200 OK: la convalida è stata eseguita correttamente (indipendentemente dal risultato della convalida)
- 400 Bad Request: formato o parametri della richiesta non validi
- 404 Not Found: Tipo di risorsa o profilo non trovato

Per ulteriori informazioni sulle specifiche `$validate` operative, consulta la documentazione delle risorse `$validate` FHIR R4.

Riferimento di conformità per AWS HealthLake

AWS HealthLake offre funzionalità progettate per aiutarti a tracciare e segnalare l'utilizzo delle API in linea con i requisiti di interoperabilità CMS (Centers for Medicare & Medicaid Services). Queste funzionalità consentono di classificare le transazioni API in base alle categorie obbligatorie del CMS e di acquisire automaticamente le metriche di utilizzo per scopi di reporting sulla conformità.

⚠ Comprensione delle responsabilità in materia di conformità

L'utilizzo AWS HealthLake e i relativi endpoint di interoperabilità CMS non sono di per sé sufficienti per ottenere la conformità al CMS. L'utente è responsabile di:

- Mappatura corretta dei flussi di lavoro delle API sugli endpoint della categoria CMS appropriata in base ai casi d'uso specifici e agli obblighi normativi
- Implementazione di controlli di autenticazione e autorizzazione adeguati che soddisfino i requisiti CMS
- Garantire che le risorse e gli scambi di dati FHIR siano conformi alle normative CMS e alle guide di implementazione applicabili
- Configurazione e monitoraggio delle CloudWatch metriche per supportare le esigenze di rendicontazione della conformità
- Comprendere quali regole CMS si applicano alla propria organizzazione e implementare i controlli tecnici e operativi appropriati

AWS HealthLake fornisce l'infrastruttura e gli strumenti per supportare gli sforzi di conformità, ma è necessario utilizzare queste funzionalità in modo appropriato in base ai requisiti normativi specifici. Il semplice instradamento delle chiamate API attraverso questi endpoint non rende automaticamente l'applicazione conforme alle normative CMS.

Argomenti

- Funzionalità di conformità CMS

Funzionalità di conformità CMS

AWS HealthLake fornisce funzionalità che consentono di soddisfare i requisiti di interoperabilità e conformità dei CMS (Centers for Medicare & Medicaid Services). Queste funzionalità consentono di tenere traccia dell'utilizzo delle API per categoria CMS e successivamente di riportare le metriche di utilizzo ai fini della conformità.

Argomenti

- Endpoint di interoperabilità CMS
- Metriche migliorate CloudWatch per la conformità CMS

Endpoint di interoperabilità CMS

Panoramica di

HealthLake fornisce quattro endpoint di interoperabilità CMS che corrispondono alle categorie di API obbligatorie per il CMS. L'URL di base sottostante del tuo data store non cambia. HealthLake Questi endpoint forniscono semplicemente un modo per classificare e tracciare le chiamate API per scopi di reporting CMS.

Scopo

Lo scopo principale di questi endpoint di interoperabilità è consentire ai clienti di:

- Monitora facilmente le transazioni API per categoria CMS
- Segnala automaticamente le metriche di utilizzo per la conformità CMS
- Mantieni i flussi di lavoro FHIR esistenti con modifiche minime

Tutte le chiamate API funzionano allo stesso modo, indipendentemente dal fatto che si utilizzi l'endpoint di interoperabilità o l'endpoint FHIR standard: l'unica differenza è il modo in cui le transazioni sono classificate nelle metriche. CloudWatch

Endpoint di interoperabilità CMS supportati

Categoria CMS	Endpoint di interoperabilità	Esempio di utilizzo
Accesso per i pazienti	`/patientaccess/v2/r4`	`baseURL/patientaccess/v2/r4/Patient/123`
Accesso da parte del fornitore	`/provideraccess/v2/r4`	`baseURL/provideraccess/v2/r4/Observation?patient=123`
Da pagante a pagante	`/payertopayerdx/v2/r4`	`baseURL/payertopayerdx/v2/r4/Practitioner/456`
Servizio di autentica zione precedente	`/priorauthservice/v2/r4`	`baseURL/priorauthservice/v2/r4/ExplanationOfBenefit?patient=789`

Come funzionano gli endpoint di interoperabilità

Chiamata API standard: HealthLake

```
baseURL/resourceType/[id]
baseURL/resourceType?[parameters]
```

Con CMS Interoperability Endpoint:

```
baseURL/interoperability-endpoint/resourceType/[id]
baseURL/interoperability-endpoint/resourceType?[parameters]
```

Il percorso dell'endpoint di interoperabilità viene semplicemente inserito tra l'URL di base e il tipo di risorsa. Tutto ciò che segue il percorso dell'endpoint di interoperabilità rimane esattamente lo stesso delle chiamate API correnti.

Esempi di utilizzo

Esempio 1: API Patient Access

Chiamata API corrente (funziona ancora):

```
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 l'endpoint di interoperabilità Patient Access (per il tracciamento 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"
```

Punti chiave:

- L'URL di base rimane: `https://healthlake.us-east-1.amazonaws.com/datastore/abc123`
- Endpoint di interoperabilità inserito: `/patientaccess/v2/r4`

- Percorso delle risorse invariato: `/Patient/123`
- Entrambe le chiamate restituiscono risposte identiche
- La chiamata all'endpoint di interoperabilità viene tracciata automaticamente in `URIType=patient-access` CloudWatch
- Le operazioni POST, PUT, PATCH, DELETE funzionano in modo identico.
- Il corpo della richiesta rimane invariato.

Riferimento alla traduzione degli endpoint

Endpoint di interoperabilità	Si traduce in	Categoria CMS
`baseURL/patientaccess/v2/r4/Patient`	`baseURL/r4/Patient`	Accesso per i pazienti
`baseURL/provideraccess/v2/r4/Observation`	`baseURL/r4/Observation`	Accesso da parte del fornitore
`baseURL/payertopayerdx/v2/r4/Practitioner/456`	`baseURL/r4/Practitioner/456`	Data Exchange tra pagatori
`baseURL/priorauthservice/v2/r4/ExplanationOfBenefit?patient=789`	`baseURL/r4/ExplanationOfBenefit?patient=789`	Autorizzazione preventiva

Note importanti

- Nessuna differenza funzionale: gli endpoint di interoperabilità e gli endpoint FHIR standard restituiscono risposte identiche e supportano operazioni identiche
- URL di base invariato: l'endpoint del data store rimane lo stesso HealthLake
- Integrazione semplice: inserisci il percorso dell'endpoint di interoperabilità tra l'URL di base e il tipo di risorsa
- Tracciamento automatico: le CloudWatch metriche classificano automaticamente le chiamate in base all'endpoint di interoperabilità utilizzato

- Compatibilità con le versioni precedenti: le chiamate API esistenti senza endpoint di interoperabilità continuano a funzionare normalmente

Metriche migliorate CloudWatch per la conformità CMS

Panoramica di

Quando si utilizzano gli endpoint di interoperabilità CMS, emette HealthLake automaticamente CloudWatch metriche avanzate con dimensioni aggiuntive per supportare i requisiti di reporting CMS. Queste metriche tengono traccia dell'utilizzo dell'API in base all'identità del chiamante, all'applicazione e ai tipi di URI specifici del CMS senza alcuna configurazione aggiuntiva.

Come funziona

Quando effettui una chiamata API utilizzando un endpoint di interoperabilità:

```
# 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.)
```

Non è necessario alcun codice o configurazione aggiuntivi. Basta usare gli endpoint di interoperabilità e le metriche avanzate vengono acquisite automaticamente.

> **Note**
>
> Per gli archivi di dati Non-Smart on FHIR, la `URIType` dimensione viene comunque acquisita, consentendoti di tracciare l'utilizzo delle API per categoria CMS. `ClientId`Le dimensioni `Sub` and sono disponibili solo quando si utilizza l'autenticazione SMART on FHIR con token portatori contenenti queste attestazioni.

Nuove dimensioni metriche

Oltre alle dimensioni esistenti (`DatastoreId`,,`Operation`)`DatastoreType`, le seguenti dimensioni vengono aggiunte automaticamente quando si utilizzano gli endpoint di interoperabilità:

Dimensione	Description	Valori di esempio	Origine
URIType	Categoria di conformità CMS	`patient-access` , `provider-access` , `payer-to-payer` , `prior-authorization`	Determinato automaticamente dal percorso dell'endpoint di interoperabilità
Sub	Identità del chiamante	Identificatore utente/entità	Estratto dal token claim del portatore `sub`
ClientId	Identificatore dell'applicazione	`portal_app` , `ehr_system`	Estratto dalla rivendicazione del token al portatore `client_id`

Metriche disponibili

Tutte le HealthLake metriche esistenti ora includono le dimensioni aggiuntive quando si utilizzano gli endpoint di interoperabilità:

- CallCount- Numero totale di chiamate API
- Latenza: tempo di risposta dell'API in millisecondi
- UserErrors- Numero di 4xx errori del client
- SystemErrors- Numero di 5xx errori del server
- Throttles: Numero di richieste limitate
- SuccessfulRequests- Numero di chiamate API riuscite

Interrogazione delle metriche in CloudWatch

CloudWatch Esempio di query Insights

Interroga tutte le chiamate API Patient Access per applicazione:

```
SELECT SUM(CallCount)
FROM "AWS/HealthLake"
WHERE DatastoreId = '75c1cf9b0d71cd38fec8f7fb317c4c1a'
    AND URIType = 'patient-access'
GROUP BY ClientId
```

Supporto di riferimento per AWS HealthLake

Il seguente materiale di riferimento di supporto è disponibile per. AWS HealthLake

> **Note**
>
> Tutte le HealthLake azioni e i tipi di dati nativi sono descritti in un riferimento separato. Per ulteriori informazioni, consulta la documentazione di riferimento dell'API di AWS HealthLake .

Argomenti

- AWS HealthLake endpoint e quote
- Tipi di dati precaricati Synthea per HealthLake
- AWS HealthLake progetti di esempio
- Risoluzione dei problemi AWS HealthLake
- Utilizzo HealthLake con un AWS SDK

AWS HealthLake endpoint e quote

I seguenti argomenti contengono informazioni sugli endpoint e sulle AWS HealthLake quote di servizio.

Argomenti

- Endpoint di servizio
- Service Quotas

Endpoint di servizio

Un endpoint di servizio è un URL che identifica un host e una porta come punto di ingresso per un servizio Web. Ogni richiesta di servizio Web include un endpoint. La maggior parte AWS dei servizi fornisce endpoint per regioni specifiche per consentire una connettività più rapida. La tabella seguente elenca gli endpoint del servizio per. AWS HealthLake

Nome della regione	Regione	Endpoint	Protocollo	
US East (Ohio)	us-east-2	healthlake.us-east-2.amazonaws.com healthlake-fips.us-east-2.amazonaws.com	HTTPS HTTPS	
US East (N. Virginia)	us-east-1	healthlake.us-east-1.amazonaws.com healthlake-fips.us-east-1.amazonaws.com	HTTPS HTTPS	
Stati Uniti occidentali (Oregon)	us-west-2	healthlake.us-west-2.amazonaws.com healthlake-fips.us-west-2.amazonaws.com	HTTPS HTTPS	
Asia Pacifico (Mumbai)	ap-south-1	healthlake.ap-south-1.amazonaws.com	HTTPS	
Asia Pacifico (Sydney)	ap-southeast-2	healthlake.ap-southeast-2.amazonaws.com	HTTPS	
Canada (Centrale)	ca-central-1	healthlake.ca-central-1.amazonaws.com	HTTPS	
Europa (Irlanda)	eu-west-1	healthlake.eu-west-1.amazonaws.com	HTTPS	

Nome della regione	Regione	Endpoint	Protocollo	
Europa (Londra)	eu-west-2	healthlake.eu-west-2.amazonaws.com	HTTPS	

Service Quotas

Le quote di servizio sono definite come il valore massimo per le risorse, le azioni e gli elementi presenti nell'account AWS .

> **Note**
>
> Per le quote regolabili, puoi richiedere un aumento della quota utilizzando la console Service Quotas. Per ulteriori informazioni, consulta Richiesta di un aumento delle quote nella Guida per l'utente di Service Quotas.
> La velocità dell'API Sync Write aumenta proporzionalmente alla dimensione del payload, con ogni incremento di 1 KB che consuma capacità aggiuntiva (ad esempio, un payload di 4 KB utilizza una capacità di scrittura quadrupla). L'impostazione dell'`x-amz-fhir-history-consistency-level`intestazione opzionale per raddoppiare il consumo di capacità di scrittura per risorsa. `strong`
> Le risorse all'interno dei pacchetti seguono i read/write limiti standard basati sulla dimensione del payload di 1 KB. Le transazioni di tipo bundle consumano il doppio della capacità di scrittura rispetto a quelle di tipo batch, il che significa che i pacchetti batch possono elaborare il doppio delle risorse al secondo rispetto ai pacchetti di transazioni.

La tabella seguente elenca le quote predefinite per. AWS HealthLake

Nome	Predefinita	Adatta e	Description
Numero di abbonamenti attivi per account	Ogni regione supportata: 100	Sì	Il numero massimo di risorse attive in

Nome	Predefinita	Adatta e	Description
			abbonamento per account.
Numero di risorse in abbonamento attive per datastore	Ogni Regione supportata: 50	Sì	Il numero massimo di risorse attive in abbonamento per datastore.
Numero di attivi SubscriptionTopic per account	Ogni regione supportata: 100	Sì	Il numero massimo di SubscriptionTopic risorse attive per account.
Numero di SubscriptionTopic risorse attive per datastore	Ogni Regione supportata: 50	Sì	Il numero massimo di SubscriptionTopic risorse attive per datastore.
Numero di caratteri in una nota medica	Ogni regione supportata: 10.000	No	Il numero massimo di caratteri in una singola nota medica all'interno del tipo di DocumentR eference risorsa (POST/ PUT richieste).
Numero di lavori simultanei StartFHIR ExportJob	Ogni regione supportata: 1	No	Il numero massimo di lavori simultanei StartFHIRExportJob .
Numero di lavori simultanei StartFHIR ImportJob	Ogni regione supportata: 1	No	Il numero massimo di lavori simultanei StartFHIRImportJob .
Numero di archivi dati per account	Ogni regione supportata: 10	Sì	Il numero massimo predefinito di archivi dati attivi per account.

Nome	Predefinita	Adatta e	Description
Numero di file in un StartFHIRImportJob	Ogni regione supportata: 1.000.000	Sì	Il numero massimo di file in a StartFHIRImportJob.
Numero di risorse per pacchetto	Ogni regione supportata: 500	No	Il numero massimo di risorse consentite in una richiesta di pacchetto.
Frequenza delle CancelFHIRExportJob richieste che utilizzano DELETE per account	Ogni regione supportata: 1	No	Il numero massimo di CancelFHIRExportJob richieste che puoi effettuare utilizzando DELETE al secondo per account.
Frequenza di richieste CreateFhirDataStore per account	Ogni regione supportata: 1	No	Il numero massimo di richieste CreateFhirDataStore che puoi effettuare al minuto per account.
Frequenza di richieste DeleteFhirDataStore per account	Ogni regione supportata: 1	No	Il numero massimo di richieste DeleteFhirDataStore che puoi effettuare al minuto per account.
Tasso di richieste per account DescribeFHIRBulkDeleteJob	Ogni regione supportata: 10	Sì	Il numero massimo di DescribeFHIRBulkDeleteJob richieste che puoi effettuare al secondo per account.

Nome	Predefinita	Adatta e	Description
Frequenza di DescribeFHIRBulkDeleteJob richieste per archivio dati	Ogni regione supportata: 10	Sì	Il numero massimo di DescribeFHIRBulkDeleteJob richieste che è possibile effettuare al secondo per archivio dati.
Frequenza di richieste DescribeFHIRDataStore per account	Ogni regione supportata: 10	No	Il numero massimo di richieste DescribeFHIRDataStore che puoi effettuare al secondo per account.
Tasso di richieste per account DescribeFHIRExportJob	Ogni regione supportata: 10	No	Il numero massimo di DescribeFHIRExportJob richieste che puoi effettuare al secondo per account.
Frequenza di DescribeFHIRExportJob richieste utilizzando GET per account	Ogni regione supportata: 10	No	Il numero massimo di DescribeFHIRExportJob richieste tramite GET che puoi effettuare al secondo per account.
Tasso di DescribeFHIRImportJob richieste per account	Ogni regione supportata: 10	No	Il numero massimo di DescribeFHIRImportJob richieste che puoi effettuare al secondo per account.
Tasso di richieste Discovery per account	Ogni regione supportata: 10	No	Il numero massimo di richieste Discovery che puoi effettuare al minuto per account.

Nome	Predefinita	Adatta e	Description
Frequenza di richieste GET per account	Ogni regione supportata: 6.000	Sì	Il numero massimo di richieste GET che è possibile effettuare al secondo per account.
Frequenza di richieste GET per archivio dati	Ogni regione supportata: 3.000	Sì	Il numero massimo di richieste GET che è possibile effettuare al secondo per data store. Gli archivi dati creati prima del 8/21 /2023 saranno limitati a 100 richieste al secondo.
Tasso di GetCapabilities richieste per account	Ogni regione supportata: 10	No	Il numero massimo di GetCapabilities richieste che puoi effettuare al secondo per account.
Frequenza di GetExportedFile richieste per datastore	Ogni regione supportata: 10	No	Il numero massimo di GetExportedFile richieste che è possibile effettuare al secondo per datastore.
Frequenza di richieste ListFHIRD ataStores per account	Ogni regione supportata: 10	No	Il numero massimo di richieste ListFHIRD ataStores che puoi effettuare al secondo per account.
Tasso di richieste per account ListFHIRExportJobs	Ogni regione supportata: 10	No	Il numero massimo di ListFHIRExportJobs richieste che puoi effettuare al secondo per account.

Nome	Predefinita	Adatta e	Description
Tasso di ListFHIRImportJobs richieste per account	Ogni regione supportata: 10	No	Il numero massimo di ListFHIRImportJobs richieste che puoi effettuare al secondo per account.
Tasso di ListTagsforResource richieste per account	Ogni regione supportata: 10	No	Il numero massimo di ListTagsforResource richieste che puoi effettuare al secondo per account.
Tasso di SearchEverything richieste per account	Ogni regione supportata: 10	Sì	Il numero massimo di SearchEverything richieste che puoi effettuare al secondo per account.
Frequenza di SearchEverything richieste per archivio dati	Ogni regione supportata: 10	Sì	Il numero massimo di SearchEverything richieste che è possibile effettuare al secondo per archivio dati.
Tasso di StartFHIRBulkDeleteJob richieste per account	Ogni regione supportata: 10	Sì	Il numero massimo di StartFHIRBulkDeleteJob richieste che puoi effettuare al secondo per account.
Frequenza di StartFHIRBulkDeleteJob richieste per archivio dati	Ogni regione supportata: 10	Sì	Il numero massimo di StartFHIRBulkDeleteJob richieste che è possibile effettuare al secondo per archivio dati.

Nome	Predefinita	Adattabile	Description
Tasso di StartFHIRBulkMemberMatchJob richieste per account	Ogni regione supportata: 10	Sì	Il numero massimo di StartFHIRBulkMemberMatchJob richieste che puoi effettuare al secondo per account.
Frequenza di StartFHIRBulkMemberMatchJob richieste per archivio dati	Ogni regione supportata: 10	Sì	Il numero massimo di StartFHIRBulkMemberMatchJob richieste che è possibile effettuare al secondo per archivio dati.
Tasso di StartFHIRExportJob richieste per account	Ogni regione supportata: 1	No	Il numero massimo di StartFHIRExportJob richieste che puoi effettuare al secondo per account.
Frequenza di StartFHIRExportJob richieste utilizzando GET per account	Ogni regione supportata: 1	No	Il numero massimo di StartFHIRExportJob richieste tramite GET che puoi effettuare al secondo per account.
Frequenza di StartFHIRExportJob richieste che utilizzano POST per account	Ogni regione supportata: 1	No	Il numero massimo di StartFHIRExportJob richieste tramite POST che puoi effettuare al secondo per account.
Tasso di StartFHIRImportJob richieste per account	Ogni regione supportata: 25	No	Il numero massimo di StartFHIRImportJob richieste che puoi effettuare al secondo per account.

Nome	Predefinita	Adatta e	Description
Tasso di TagResource richieste per account	Ogni regione supportata: 10	No	Il numero massimo di TagResource richieste che puoi effettuare al secondo per account.
Tasso di UntagResource richieste per account	Ogni regione supportata: 10	No	Il numero massimo di UntagResource richieste che puoi effettuare al secondo per account.
Tasso di ValidateResource richieste per account	Ogni regione supportata: 2.000	Sì	Il numero massimo di ValidateResource richieste che puoi effettuare al secondo per account. Gli archivi dati creati prima 8/21 del 2023 saranno limitati a 300 richieste al secondo.
Frequenza di ValidateResource richieste per archivio dati	Ogni regione supportata: 1.000	Sì	Il numero massimo di ValidateResource richieste che è possibile effettuare al secondo per archivio dati. Gli archivi dati creati prima 8/21 del 2023 saranno limitati a 300 richieste al secondo.
Frequenza di richieste di scrittura per account	Ogni regione supportata: 6.000	Sì	Il numero massimo di richieste CREATE|UPDATE|DELETE che puoi effettuare al secondo per account.

Nome	Predefinita	Adatta e	Description
Frequenza di richieste di scrittura per archivio dati	Ogni regione supportata: 3.000	Sì	Il numero massimo di richieste CREATE|UPDATE|DELETE che è possibile effettuare al secondo per data store. Gli archivi dati creati prima del 8/21 2023 saranno limitati a 300 richieste al secondo.
Frequenza delle richieste di ricerca utilizzando GET per account	Ogni Regione supportata: 200	Sì	Il numero massimo di richieste di ricerca tramite GET che puoi effettuare al secondo per account.
Frequenza delle richieste di ricerca utilizzando GET per archivio dati	Ogni regione supportata: 100	Sì	Il numero massimo di richieste di ricerca tramite GET che è possibile effettuare al secondo per archivio dati.
Frequenza delle richieste di ricerca utilizzando POST per account	Ogni Regione supportata: 200	Sì	Il numero massimo di richieste di ricerca tramite POST che puoi effettuare al secondo per account.
Frequenza delle richieste di ricerca che utilizzano POST per archivio dati	Ogni regione supportata: 100	Sì	Il numero massimo di richieste di ricerca tramite POST che è possibile effettuare al secondo per archivio dati.

Nome	Predefinita	Adatta e	Description
Dimensione del singolo file importato	Ogni regione supportata: 50 GB	No	La dimensione massima (in GB) di un singolo file incluso in un StartFHIR ImportJob.
Transazioni totali del pacchetto asincrono in coda per datastore	Ogni regione supportata: 500	Sì	Il numero massimo di transazioni in bundle asincrone in coda per datastore in un dato momento.
Totale dei lavori di esportazione in blocco in coda per datastore	Ogni regione supportata: 25	Sì	Il numero massimo di lavori di esportazione in blocco in coda per datastore in un dato momento.
Totale dei lavori di importazione in blocco in coda per datastore	Ogni regione supportata: 25	Sì	Il numero massimo di lavori di importazione in blocco in coda per datastore in un dato momento.
Dimensione totale del processo di importazione	Ogni regione supportata: 5.000 Gigabyte	Sì	La dimensione massima (in GB) di tutti i file inclusi nel processo di importazione.

Tipi di dati precaricati Synthea per HealthLake

HealthLake supporta solo SYNTHEA come tipo di dati precaricato. Synthea è un generatore sintetico per pazienti che modella la storia medica. `Patient` È ospitato in un repository Git open source che consente di generare una risorsa conforme HealthLake a FHIR R4 in `Bundle` modo che gli utenti possano testare i modelli senza utilizzare i dati effettivi dei pazienti.

I seguenti tipi di risorse sono disponibili negli archivi dati precaricati. HealthLake Per ulteriori informazioni sul precaricamento degli archivi HealthLake dati con i dati Synthea, vedere. Creazione di un archivio HealthLake dati

> **Note**
>
> Per visualizzare un elenco completo delle risorse FHIR HealthLake R4 supportate, vedere. Tipi di risorse supportati da FHIR R4 per HealthLake

Tipi di risorse Synthea FHIR supportati da HealthLake

AllergyIntolerance	Location (Ubicazione)
CarePlan	MedicationAdministration
CareTeam	MedicationRequest
Richiedi	Osservazione
Condizione	Organizzazione
Dispositivo	Paziente
DiagnosticReport	Professionista
Incontro	PractitionerRole
ExplanationofBenefit	Procedura
ImagingStudy	Provenienza
Immunizzazione	

AWS HealthLake progetti di esempio

Per approfondire l'analisi, puoi utilizzare HealthLake altri AWS servizi, come dimostrato nei seguenti esempi di post di blog.

HealthLake analisi integrata

- Applicazioni per la salute della popolazione con AWS HealthLake — Parte 1: Analisi e monitoraggio con Amazon Quick.
- Creazione di modelli predittivi di malattie utilizzando Amazon SageMaker AI con AWS HealthLake dati normalizzati.
- Crea una ricerca cognitiva e un grafico delle conoscenze sanitarie utilizzando i servizi di AWS intelligenza artificiale.

HealthLake monitoraggio degli eventi

- EventBridge Integrazione Amazon con AWS HealthLake.

Risoluzione dei problemi AWS HealthLake

I seguenti argomenti forniscono consigli per la risoluzione di errori e problemi che potrebbero verificarsi durante l'utilizzo della console AWS CLI AWS SDKs, o HealthLake . Se trovi un problema che non è elencato in questa sezione, utilizza il pulsante Fornisci feedback nella barra laterale destra di questa pagina per segnalarlo.

Argomenti

- Azioni relative all'archivio dati
- Azioni di importazione
- FHIR APIs
- Integrazioni NLP
- Integrazioni SQL

Azioni relative all'archivio dati

Problema: quando tento di creare un archivio HealthLake dati, ricevo il seguente errore:

```
AccessDeniedException: Insufficient Lake Formation permission(s): Required Database on
 Catalog
```

Il 14 novembre 2022, HealthLake ha aggiornato le autorizzazioni IAM richieste per creare un nuovo data store. Per ulteriori informazioni, consulta Configura un utente o un ruolo IAM da utilizzare HealthLake (amministratore IAM).

Problema: quando si crea un HealthLake data store utilizzando AWS SDKs, lo stato di creazione del data store restituisce un'eccezione o uno stato sconosciuto.

Aggiorna l' AWS SDK alla versione più recente se le tue chiamate `DescribeFHIRDatastore` o le chiamate `ListFHIRDatastores` API restituiscono un'eccezione o uno stato sconosciuto del data store.

Azioni di importazione

Problema: posso ancora utilizzarli HealthLake se i miei dati non sono in formato FHIR R4?

Solo i dati in formato FHIR R4 possono essere importati in un archivio dati. HealthLake Per un elenco di partner che possono contribuire a trasformare i dati sanitari esistenti in formato FHIR R4, consulta Partners.AWS HealthLake

Problema: perché il mio processo di importazione FHIR non è riuscito?

Un processo di importazione riuscito genererà una cartella con i risultati (registro di output) in `.ndjson` formato, tuttavia, i singoli record potrebbero non essere importati. In tal caso, verrà generata una seconda `FAILURE` cartella con un manifesto di record che non è stato possibile importare. Per ulteriori informazioni, consulta Importazione di dati FHIR con AWS HealthLake.

Per analizzare il motivo per cui un processo di importazione non è riuscito, utilizza l'`DescribeFHIRImportJob`API per analizzare il JobProperties. Si consiglia quanto segue:

- Se lo stato è `FAILED` ed è presente un messaggio, gli errori sono correlati a parametri del lavoro, come la dimensione dei dati di input o il numero di file di input, che superano le HealthLake quote.
- Se lo stato del processo di importazione è`COMPLETED_WITH_ERRORS`, controllate il file manifesto per informazioni su quali file non sono stati importati correttamente. `manifest.json`
- Se lo stato del processo di importazione è uguale `FAILED` e non è presente alcun messaggio, vai alla posizione di output del lavoro per accedere al file manifest,`manifest.json`.

Per ogni file di input, esiste un file di output non riuscito con il nome del file di input per ogni risorsa che non riesce a importare. Le risposte contengono il numero di riga (lineID) corrispondente alla posizione dei dati di input, l'oggetto di risposta FHIR (UpdateResourceResponse) e il codice di stato (statusCode) della risposta.

Un file di output di esempio potrebbe essere simile al seguente:

{"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}
```

L'esempio precedente mostra che si sono verificati errori sulle righe 3, 4, 7, 9, 15 delle linee di input corrispondenti del file di input. Per ognuna di queste righe, le spiegazioni sono le seguenti:

- Nella riga 3, la risposta spiega che il contenuto `resourceType` fornito nella riga 3 del file di input non è valido.
- Alla riga 5, la risposta spiega che c'è un errore di convalida FHIR nella riga 5 del file di input.
- Alla riga 7, la risposta spiega che c'è un problema di convalida se viene `resourceId` fornito come input.
- Alla riga 9, la risposta spiega che il file di input deve contenere un ID di risorsa valido.
- Alla riga 15, la risposta del file di input è che il file non è in un formato JSON valido.

FHIR APIs

Problema: come posso implementare l'autorizzazione per il RESTful APIs FHIR?

Determinate la Strategia di autorizzazione all'archiviazione dei dati modalità di utilizzo.

Per creare l'autorizzazione SigV4 utilizzando il AWS SDK per Python (Boto3), create uno script simile all'esempio seguente.

```
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: perché ricevo *`AccessDenied`* errori quando utilizzo il FHIR RESTful APIs per un archivio dati crittografato con una chiave KMS gestita dal cliente?

Le autorizzazioni sia per le chiavi gestite dal cliente che per le politiche IAM sono necessarie per consentire a un utente o un ruolo di accedere a un archivio dati. Un utente deve disporre delle autorizzazioni IAM richieste per utilizzare una chiave gestita dal cliente. Se un utente ha revocato o ritirato una concessione che HealthLake consentiva di utilizzare la chiave KMS gestita dal cliente, HealthLake restituirà un errore. `AccessDenied`

HealthLake deve disporre dell'autorizzazione per accedere ai dati dei clienti, crittografare le nuove risorse FHIR importate in un archivio dati e decrittografare le risorse FHIR quando vengono richieste. Per ulteriori informazioni, vedere Risoluzione dei problemi relativi alle autorizzazioni. AWS KMS

Problema: un'operazione dell'*POST*API FHIR sull' HealthLake utilizzo di un documento da 10 MB restituisce l'errore. `413 Request Entity Too Large`

AWS HealthLake ha un limite di 5 MB per le API di creazione e aggiornamento sincrono per evitare un aumento delle latenze e dei timeout. Puoi importare documenti di grandi dimensioni, fino a 164 MB, utilizzando il tipo di risorsa che utilizza l'`Binary`API Bulk Import.

Integrazioni NLP

Problema: come posso attivare la funzionalità integrata HealthLake di elaborazione del linguaggio naturale?

A partire dal 14 novembre 2022, il comportamento predefinito degli archivi HealthLake dati è cambiato.

Archivi dati attuali: tutti gli archivi di HealthLake dati attuali smetteranno di utilizzare l'elaborazione del linguaggio naturale (NLP) su risorse con codifica `DocumentReference` base64. Ciò significa che `DocumentReference` le nuove risorse non verranno analizzate utilizzando l'NLP e non verranno generate nuove risorse in base al testo del tipo di risorsa. `DocumentReference` Per `DocumentReference` le risorse esistenti, i dati e le risorse generati tramite la PNL rimangono, ma non verranno aggiornati dopo il 20 febbraio 2023.

Nuovi archivi HealthLake dati: gli archivi dati creati dopo il 20 febbraio 2023 non eseguiranno l'elaborazione del linguaggio naturale (NLP) su risorse con codifica base64. `DocumentReference`

Per attivare l'integrazione HealthLake NLP, crea un caso di supporto utilizzando. AWS Support Center Console Per creare il tuo caso, accedi al tuo Account AWS, quindi scegli Crea caso. Per ulteriori informazioni sulla creazione di un caso e sulla gestione dei casi, consulta Creazione di casi di supporto e gestione dei casi nella Guida per l'Supporto utente.

Problema: >Come posso trovare *`DocumentReference`* risorse che non possono essere elaborate con la PNL integrata?

Se una `DocumentReference` risorsa non è valida, HealthLake fornisce un'estensione che indica un errore di convalida anziché inserirlo nell'output di PNL medico integrato. Per trovare **`DocumentReference`** le risorse che hanno portato a un errore di convalida durante l'elaborazione NLP, è possibile utilizzare la **`search`** funzione FHIR con la chiave di ricerca e il valore HealthLake di ricerca VALIDATION_ERROR. cm-decoration-status Questa ricerca elencherà tutte le `DocumentReference` risorse che hanno portato a errori di convalida, insieme a un messaggio

di errore che descrive la natura dell'errore. La struttura del campo di estensione in quelle `DocumentReference` risorse con errori di convalida sarà simile all'esempio seguente.

```
"extension": [
          {
              "extension": [
                  {
                      "url": "http://healthlake.amazonaws.com/aws-cm/status/",
                      "valueString": "VALIDATION_ERROR"
                  },
                  {
                      "url": "http://healthlake.amazonaws.com/aws-cm/message/",
                      "valueString": "Resource led to too many nested objects after NLP
 operation processed the document. 10937 nested objects exceeds the limit of 10000."
                  }
              ],
              "url": "http://healthlake.amazonaws.com/aws-cm/"
          }
    ]
```

> **Note**
>
> A `VALIDATION_ERROR` può verificarsi anche se la decorazione NLP crea più di 10.000 oggetti annidati. Quando ciò accade, il documento deve essere suddiviso in documenti più piccoli prima dell'elaborazione.

Integrazioni SQL

Problema: Perché ottengo un Lake Formation *`permissions error: lakeformation:PutDataLakeSettings`* quando aggiungo un nuovo amministratore di data lake?

Se il tuo utente o ruolo IAM contiene la policy `AWSLakeFormationDataAdmin` AWS gestita, non puoi aggiungere nuovi amministratori di data lake. Riceverai un errore contenente quanto segue:

```
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 policy AWS gestita `AdministratorAccess` è necessaria per aggiungere un utente o un ruolo IAM come amministratore del data lake AWS Lake Formation. Se l'utente o il ruolo IAM contiene anche `AWSLakeFormationDataAdmin` l'azione avrà esito negativo. La policy `AWSLakeFormationDataAdmin` AWS gestita contiene un rifiuto esplicito per il funzionamento dell'API AWS Lake Formation,. `PutDataLakeSetting` Anche gli amministratori con accesso completo all' AWS utilizzo della policy `AdministratorAccess` gestita possono essere limitati dalla policy. `AWSLakeFormationDataAdmin`

Problema: come posso migrare un HealthLake data store esistente per utilizzare l'integrazione con Amazon Athena SQL?

HealthLake gli archivi di dati creati prima del 14 novembre 2022 sono funzionali, ma non sono interrogabili in Athena utilizzando SQL. Per interrogare un data store preesistente con Athena, devi prima migrarlo in un nuovo data store.

Per migrare i HealthLake dati in un nuovo data store

1. Crea un nuovo data store.
2. Esporta i dati dal bucket preesistente in un bucket Amazon S3.
3. Importa i dati nel nuovo data store dal bucket Amazon S3.

> **Note**
>
> L'esportazione di dati in un bucket Amazon S3 comporta un costo aggiuntivo. Il costo aggiuntivo dipende dalla dimensione dei dati esportati.

Problema: quando si crea un nuovo HealthLake data store per l'integrazione SQL, lo stato del data store non cambia da*`Creating`*.

Se provi a creare un nuovo HealthLake data store e lo stato del tuo data store non cambia da Creazione, devi aggiornare Athena per utilizzare. AWS Glue Data Catalog Per ulteriori informazioni, consulta l'aggiornamento al AWS Glue Data Catalog step-by-step nella Guida per l'utente di Amazon Athena.

Dopo aver aggiornato correttamente AWS Glue Data Catalog, puoi creare un data store. HealthLake

Per rimuovere un vecchio archivio HealthLake dati, crea una richiesta di supporto utilizzando. AWS Support Center Console Per creare il tuo caso, accedi al tuo Account AWS, quindi scegli Crea caso.

Per ulteriori informazioni, consulta Creazione di casi di supporto e gestione dei casi nella Guida Supporto per l'utente.

Problema: la console Athena non funziona dopo l'importazione dei dati in un nuovo archivio dati HealthLake

Dopo aver importato i dati in un nuovo archivio HealthLake dati, i dati potrebbero non essere disponibili per l'uso immediato. Questo serve per consentire ai dati di essere inseriti nelle tabelle di Apache Iceberg. Riprova in un secondo momento.

Problema: come posso collegare i risultati di ricerca in Athena ad altri AWS servizi?

Quando si condividono i risultati di ricerca di Athena con altri AWS servizi, possono verificarsi problemi quando li si utilizza `json_extract[1]` come parte di una query di ricerca SQL. Per risolvere questo problema, è necessario eseguire l'aggiornamento a`CATVAR`.

Potresti riscontrare questo problema quando provi a creare risultati di salvataggio, una tabella (statica) o una vista (dinamica).

Utilizzo HealthLake con un AWS SDK

AWS i kit di sviluppo software (SDKs) sono disponibili per molti linguaggi di programmazione più diffusi. Ogni SDK fornisce un'API, esempi di codice e documentazione che facilitano agli sviluppatori la creazione di applicazioni nel loro linguaggio preferito.

Documentazione sugli SDK	Esempi di codice
AWS SDK per C++	AWS SDK per C++ esempi di codice
AWS CLI	AWS CLI esempi di codice
AWS SDK per Go	AWS SDK per Go esempi di codice
AWS SDK per Java	AWS SDK per Java esempi di codice
AWS SDK per JavaScript	AWS SDK per JavaScript esempi di codice
AWS SDK per Kotlin	AWS SDK per Kotlin esempi di codice
AWS SDK per .NET	AWS SDK per .NET esempi di codice

Documentazione sugli SDK	Esempi di codice
AWS SDK per PHP	AWS SDK per PHP esempi di codice
AWS Strumenti per PowerShell	AWS Strumenti per PowerShell esempi di codice
AWS SDK per Python (Boto3)	AWS SDK per Python (Boto3) esempi di codice
AWS SDK per Ruby	AWS SDK per Ruby esempi di codice
AWS SDK per Rust	AWS SDK per Rust esempi di codice
AWS SDK per SAP ABAP	AWS SDK per SAP ABAP esempi di codice
AWS SDK per Swift	AWS SDK per Swift esempi di codice

Esempio di disponibilità

Non riesci a trovare quello che ti serve? Richiedi un esempio di codice utilizzando il link Fornisci un feedback nella parte inferiore di questa pagina.

AWS HealthLake rilasci

La tabella seguente mostra quando sono state rilasciate funzionalità e aggiornamenti per AWS HealthLake. Per ulteriori informazioni su una versione, consulta l'argomento collegato.

Modifica	Descrizione	Data
Sicurezza e filtraggio dei tag per $export e $ davinci-data-export	Le `$davinci-data-export` operazioni `$export` and ora supportano `_security` e `_tag` interrogano i parametri per filtrare le risorse esportate in base `meta.security` ai valori di codifica. `meta.tag` Ciò consente l'esportazione multi-tenant e il controllo granulare degli accessi utilizzando il formato FHIR standard. `system|code`	30 aprile 2026
$ operazione bulk-member-match	AWS HealthLake ora supporta l'`$bulk-member-match` operazione per l'elaborazione asincrona delle richieste di abbinamento di più membri. Questa operazione consente alle organizzazioni sanitarie di abbinare in modo efficient e gli identificatori univoci di centinaia di membri in diversi sistemi sanitari utilizzando informazioni demografiche e di copertura in un'unica richiesta di massa. • Gestisce fino a 500 membri per richiesta con un	1 aprile 2026

	massimo di 5 operazioni simultanee per archivio dati • Risultati suddivisi in MatchedMembers NonMatchedMembers, e gruppi ConsentConstrained Members • Si integra con i flussi di lavoro `$davinci-data-export` di dati di end-to-end massa Per ulteriori informazioni, consulta the section called “$bulk-member-match”.	
Transazioni in bundle asincrone	AWS HealthLake ora supporta il `Bundle` tipo asincrono `transaction` , che consente di inviare transazioni con un massimo di 500 risorse. HealthLake mette in coda la transazione per l'elaborazione e restituisce immediatamente un URL di polling per controlla re lo stato e recuperare i risultati. Per ulteriori informazi oni, consulta Transazioni di bundle asincrone.	24 marzo 2026

Patch Support nelle operazioni in bundle	HealthLake ora supporta Bundle Patch, abilitando FHIRPath sia i bundle JSON Patch for Transaction e Batch, sia FHIRPath l'API Patch. Questa funzionalità consente ai clienti di applicare patch alle risorse direttamente all'interno delle operazioni Bundle senza richiedere la sostituzione completa delle risorse, ridurre le dimensioni del payload, semplificare i flussi di lavoro di integrazione e consentire un'acquisizione più rapida dei dati.	20 marzo 2026
DaVinci PDex Tipi di esportazione per $ davinci-data-export	L'`$davinci-data-export` operazione ora supporta i tipi di PDex esportazione per Provider Access e Member Access APIs. Payer-to-Payer • Logica di inclusione delle risorse basata sul profilo ExplanationOfBenefit • Trasformazione dei dati finanziari con il parametro `_includeEOB2xWoFinancial` • Filtro temporale quinquennale sui dati clinici e sui reclami	20 marzo 2026

_until Parametro in $export & $ davinci-data-export	Parametro di filtraggio temporale per le operazioni di esportazione	26 febbraio 2026
_include il parametro di ricerca	HealthLake ora supporta include: * e include:iterate	26 febbraio 2026
Endpoint di interoperabilità CMS	Questa funzionalità consente di tenere traccia dell'utilizzo dell'API per categoria CMS e successivamente di riportare le metriche di utilizzo a fini di conformità.	26 febbraio 2026
Bundle Message Type Support	Supporto limitato per le risorse del pacchetto FHIR con tipo di messaggio	26 febbraio 2026
È stato aggiunto il supporto per nuovi IGs	AWS HealthLake ha ampliato il supporto delle FHIR Implementation Guides (IGs) per CMS 0057F: • Support per CARIN Blue Button 2.0.0 e 2.1.0 • Supporto per Da Vinci Payer Data Exchange 2.0.0 e 2.1.0 • Supporto per DaVinci Payer Data Exchange (PDex) US Drug Formulary 2.0.1 e 2.1.0 • Supporto per Da Vinci Clinical Data Exchange (CDex) 2.1.0 • Supporto per Da Vinci Prior Authorization Support (PAS) FHIR IG 2.1.0	26 febbraio 2026

Operazione $submit	L'`$submit`operazione consente di inviare elettroni camente richieste di autorizza zione preventiva ai pagatori per l'approvazione.	26 febbraio 2026
Operazione $questionnaire-package	L'`$questionnaire-pac kage` operazione recupera un pacchetto completo contenent e un questionario FHIR e tutte le sue dipendenze necessari e per il rendering e l'elabora zione del questionario.	26 febbraio 2026
Operazione $inquire	L'`$inquire`operazione consente di verificare lo stato di una richiesta di autorizza zione preventiva inviata in precedenza.	26 febbraio 2026
Operazione $member-remove	L'operazione $member-r emove consente di rimuovere membri da una lista di attribuzi one dei membri FHIR (risorsa di gruppo) in. AWS HealthLake	12 novembre 2025
Operazione $member-match	AWS HealthLake ora supporta l'operazione $member-match per le risorse per i pazienti, che consente alle organizzazioni sanitarie di trovare l'identificatore univoco di un membro in diversi sistemi sanitari utilizzando informazioni demografiche e di copertura.	12 novembre 2025

Operazione $member-add	L'operazione FHIR $member-add aggiunge un membro (paziente) a una risorsa del gruppo, in particolare a una lista di attribuzione dei membri.	12 novembre 2025
$ davinci-data-export Operazione	L'davinci-data-export operazione $ è un'operazione FHIR asincrona che consente l'esportazione dei dati della Member Attribution List da. AWS HealthLake	12 novembre 2025
$ confirm-attribution-list Operazione	Indica a un produttore che il Consumatore non ha più modifiche da apportare all'Elenco di attribuzione, completando l'elenco di attribuzione rimuovendo i membri inattivi e modificando lo stato in «finale».	12 novembre 2025
Operazione $attribution-status	Recupera lo stato di attribuzione per un membro specifico , restituendo un pacchetto contenente tutte le risorse di attribuzione relative al paziente.	12 novembre 2025

Abbonamenti FHIR	HealthLake supporta gli abbonamenti FHIR, che consentono di ricevere notifiche in tempo reale quando si verificano modifiche specifiche ai dati sanitari e di creare flussi di lavoro basati sugli eventi.	30 ottobre 2025
Espansione della regione a Montreal, Canada	HealthLake è disponibile nella regione Canada (Montreal). Per ulteriori informazioni, consulta Service Endpoints.	17 ottobre 2025

È stato aggiunto il supporto per nuovi IGs	AWS HealthLake ha esteso il supporto delle FHIR Implementation Guides (IGs) per il seguente mercato canadese: • Il profilo FHIR di CA Core +Canadian Baseline definisce gli elementi di dati fondamentali e i vincoli per l'interoperabilità tra i sistemi sanitari canadesi. • CA:Erec Pan-Canadian eConsultStandardized eReferral-Specificazione FHIR per referenze e consulenze elettroniche tra operatori sanitari in tutto il Canada. • Patient Summary Canadian Edition (PS-CA) Adattamento canadese dell'International Patient Summary (IPS) per la condivisione di informazioni essenziali sulla salute dei pazienti tra le strutture di assistenza. • Profilo FHIR RepositoryOntario specifico per Ontario Digital Health Drug per lo scambio standardizzato di dati su farmaci e prescrizioni all'interno del sistema sanitario provinciale.	17 ottobre 2025

Supporto IG specifico per regione	HealthLake ora supporta specifiche regioni IGs. Per ulteriori informazioni, consulta le convalide dei profili.	8 ottobre 2025
Operazione con patch	HealthLake consente di modificare elementi specifici delle risorse FHIR utilizzan do le operazioni JSON Patch senza aggiornare l'intera risorsa.	18 agosto 2025
Operazione $validate	HealthLake consente di convalidare le risorse FHIR rispetto alle specifiche e ai profili senza eseguire operazioni di archiviaz ione, restituendo risultati di convalida dettagliati.	18 agosto 2025
Operazione $purge	HealthLake include funzional ità per eliminare definitiv amente tutte le risorse all'inter no dello scomparto del paziente dal datastore.	18 agosto 2025
Operazione $lookup	HealthLake offre la possibili tà di recuperare informazioni dettagliate su un concetto specifico in a CodeSystem fornendo il codice e l'identif icatore di sistema.	18 agosto 2025

Operazione $expand	HealthLake ora consente di espandere ValueSet le risorse per recuperare l'elenco completo dei codici contenuti in Customer-ingered. ValueSets	18 agosto 2025
Operazione $erase	HealthLake ora offre la cancellazione permanente di una risorsa specifica e di tutte le sue versioni storiche dal datastore.	18 agosto 2025
Operazione $document	HealthLake supporta la generazione di documenti clinici completi raggruppa ndo una risorsa Composition con tutte le relative risorse di riferimento in un unico pacchetto di documenti.	18 agosto 2025
supporta la generazione di documenti clinici completi raggruppando una risorsa Composition con tutte le sue risorse di riferimento in un unico pacchetto di documenti . ----sep----:below Search Modifier	HealthLake introduce la ricerca di valori URI gerarchic amente inferiori all'URI specificato in un sistema terminologico.	8 agosto 2025

Eliminazione condizionale	HealthLake ora supporta FHIR Conditional Delete, che consente alle organizza zioni sanitarie di eliminare una risorsa esistente in base a criteri di ricerca anziché in base a un ID FHIR logico. Per ulteriori informazioni, consulta Eliminazione delle risorse FHIR in base alle condizioni.	7 luglio 2025
È stato aggiunto il supporto per nuove IGs	AWS HealthLake ha ampliato il supporto delle guide di implementazione FHIR (IGs) per quanto segue: • US Core 7.0.0 che specifica come utilizzare FHIR per implementare lo standard USCDI 4.0 • Guida all'implementazion e di UK Core 2.0.1 per fornire linee guida all'imple mentazione FHIR in tutto il Regno Unito	7 luglio 2025
Espansione della regione a Dublino, Irlanda	HealthLake è disponibile nella regione UE (Dublino). Per ulteriori informazioni, consulta Service Endpoints.	9 giugno 2025

Transazioni di tipo bundle	HealthLake ora supporta il pacchetto FHIR di tipo «Transaction», che consente alle organizzazioni sanitarie di inviare più risorse come un'unica operazione atomica. Ciò consente flussi di lavoro di scambio e integrazione dei dati più efficienti. Ad esempio, un operatore sanitario può ora aggiornare la cartella del paziente, l'elenco dei farmaci e l'appuntamento in un'unica transazione, riducendo la complessità e i potenziali errori. Per ulteriori informazioni, vedere Bundling FHIR Resources.	28 aprile 2025

È stato aggiunto il supporto per il nuovo IGs	AWS HealthLake AWS HealthLake ha ampliato il supporto delle guide di implementazione FHIR (IGs) per quanto segue: • Guida all'implementazione HEDIS® di NCQA (0.3.1): supporta la misurazione e la reportistica della qualità per l'Healthcare Effectiveness Data and Information Set (HEDIS). • International Patient Summary (IPS) (2.0.0): consente lo scambio di informazioni sanitarie essenziali per supportare la continuità dell'assistenza ai pazienti. • Quality Measure (5.0.0): supporta la rappresen tazione e lo scambio di definizioni e dati delle misure di qualità. • Rapporti genomici (3.0.0): facilita lo scambio di dati e report genomici.	28 aprile 2025

Chiavi di idempotenza	HealthLake ora supporta le chiavi di idempotenza per le operazioni FHIR POST, fornendo un robusto meccanismo per garantire l'integrità dei dati durante la creazione delle risorse. Per ulteriori informazioni, vedere Idempotenza e concorrenza.	18 aprile 2025
Coerenza della cronologia FHIR	HealthLake ora supporta una forte coerenza per gli archivi di dati abilitati alla cronologia tramite la nuova `x-amz-fhir-history-consistency-level` intestazione. Se impostati su «forte», i risultati della ricerca FHIR includono tutti i record indicizzati indipendentemente dallo stato dell'aggiornamento. Per ulteriori informazioni, consulta FHIR Search Consistency Levels.	18 aprile 2025

Etag e 'if-match'	HealthLake ora fornisce il supporto ETag, che consente ai clienti di utilizzare l'intestazione 'If-Match' per garantire aggiornamenti idempotenti. Ciò aiuta a mantenere l'integrità dei dati prevenendo le sovrascritture accidentali durante gli aggiornamenti simultanei. Ciò è particolarmente utile in ambienti sanitari ad alto volume in cui più sistemi potrebbero tentare di aggiornare lo stesso record contemporaneamente. HealthLakePer ulteriori ETag informazioni, consulta AWS.	18 aprile 2025

Condizionale PUTs nei pacchetti	HealthLake ora supporta gli aggiornamenti condizionali per i pacchetti FHIR, offrendo alle organizzazioni sanitarie una maggiore flessibilità nella gestione e nell'aggiornamento dei propri dati. I clienti possono ora specificare criteri per creare, aggiornare o eliminare risorse in modo condizionale come parte di una transazione Bundle. Ciò semplifica i processi di sincronizzazione dei dati tra i sistemi e riduce la necessità di una logica complessa lato client. Per ulteriori informazioni, consulta Conditional PUTs in Bundles.	18 aprile 2025

SMART su cannocchiali FHIR V2	HealthLake supporta SMART su ambiti FHIR V2 per la creazione, la lettura, l'aggiornamento, l'eliminazione e la ricerca di risorse FHIR. Per ulteriori informazioni, vedere SMART on FHIR resource scopes for. HealthLake • Gli ambiti SMART on FHIR V2 sono disponibili per tutti gli archivi HealthLake dati creati dopo il 22/01/2025. Se il tuo data store è stato creato prima di questa data, puoi inviare un ticket di supporto per abilitare gli oscilloscopi SMART on FHIR V2. Crea un caso utilizzando. AWS Support Center Console Per creare il tuo caso, accedi al tuo Account AWS e scegli Crea caso.	22 gennaio 2025
FHIR US Core Profile, versione 6.1.0	HealthLake supporta la versione 6.1.0 del FHIR US Core Profile. Per ulteriori informazioni, vedere Convalide del profilo FHIR per. HealthLake	22 gennaio 2025
FHIR $export con GET	HealthLake supporta FHIR con`$export`. `GET` Per ulteriori informazioni, vedere Esportazione di HealthLake dati con FHIR. `$export`	22 gennaio 2025

Guida per sviluppatori Refactored con esempi di codice testati	HealthLake introduce una guida per sviluppatori rielaborata con esempi di codice testati per azioni native e SDK. AWS CLI AWS Inoltre, sono ora disponibili procedure per tutte le interazioni API FHIR supportate. Per ulteriori informazioni, consulta Esempi di codice e Gestione delle risorse FHIR.	18 dicembre 2024

FHIR `history` e interazioni `vread`	HealthLake supporta l'`history`interazione FHIR per recuperare la cronologia di una particolare risorsa e l'`vread`interazione per eseguire una lettura specifica della versione di una risorsa. Per ulteriori informazioni, vedere Leggere la cronologia delle risorse FHIR. • `history`La risorsa FHIR è abilitata per impostazione predefinita in tutti gli archivi HealthLake dati creati dopo il 25/10/2024. Se il tuo data store è stato creato prima di questa data, puoi inviare un ticket di supporto per abilitare l'interazione FHIR. `history` Crea un caso utilizzando AWS Support Center Console. Per creare il tuo caso, accedi al tuo Account AWS e scegli Crea caso.	25 ottobre 2024

Operazione FHIR `Patient/$everything`	HealthLake supporta l'`Patient/$everything` operazione FHIR per la ricerca di una `Patient` risorsa e di tutte le risorse correlate. Utilizzando questa operazione, è possibile accedere all'intera cartella clinica del paziente o scaricare `Patient` dati in blocco. Per ulteriori informazioni, vedere Acquisizione dei dati dei pazienti con `Patient/$everything`. • FHIR `Patient/$everything` è abilitato per impostazione predefinita in tutti gli archivi HealthLake dati creati dopo il 27/02/2024. Se il tuo data store è stato creato prima di questa data, puoi inviare un ticket di supporto per abilitare l'operazione. `Patient/$everything` Crea un caso utilizzando AWS Support Center Console. Per creare il tuo caso, accedi al tuo Account AWS e scegli Crea caso.	27 febbraio 2024

Risorsa FHIR `VerificationResult`	HealthLake supporta il tipo di `VerificationResult` risorsa FHIR per descrivere i requisiti di convalida, le fonti, lo stato e le date per uno o più elementi. Per ulteriori informazioni, vedere Tipi di risorse FHIR R4 per. HealthLake	9 dicembre 2023

Operazione FHIR `$export`	HealthLake supporta l'`$export`operazione FHIR per l'esportazione di dati sanitari in blocco da un archivio dati. HealthLake Per ulteriori informazioni, vedere Esportazione HealthLake di dati con FHIR. `$export` • FHIR `$export` è abilitato per impostazione predefinita in tutti gli archivi HealthLake dati creati dopo il 1° giugno 2023. Se il tuo data store è stato creato prima di questa data, puoi inviare un ticket di supporto per abilitare l'`$export`operazione. Crea un caso utilizzando AWS Support Center Console. Per creare il tuo caso, accedi al tuo Account AWS e scegli Crea caso. • HealthLake gli archivi di dati creati prima del 23/01/06/ supportano solo le richieste di `$export` lavoro per le esportazioni a livello di sistema. • HealthLake gli archivi di dati creati prima del 06/01/23 non supportano l'acquisi zione dello stato di un FHIR `$export` utilizzando una richiesta sull'endpoint del data store. `GET`	1 giugno 2023

Supporto SMART su FHIR	HealthLake aggiunge il supporto per SMART sull'auto rizzazione FHIR. Per ulteriori informazioni, consulta SMART on FHIR support per. AWS HealthLake	31 maggio 2023
Convalide del profilo FHIR	HealthLake supporta le convalide dei profili FHIR per definire definizioni di tipi di risorse specifici utilizzando estensioni di vincoli and/or sui tipi di risorse di base. Per ulteriori informazioni, consulta Convalide dei profili.	31 maggio 2023
Regione Asia Pacifico (Mumbai)	HealthLake è disponibile nella regione Asia Pacifico (Mumbai). Per ulteriori informazioni, consulta Service Endpoints.	4 aprile 2023

L'elaborazione del linguaggio naturale è disattivata per impostazione predefinita	HealthLake ha disattivato l'elaborazione integrata del linguaggio naturale (NLP) su tutti gli archivi di dati a partire dal 20 febbraio 2023. Puoi inviare un ticket di supporto per attivare la funzionalità NLP integrata. Crea un caso utilizzando. AWS Support Center Console Per creare il tuo caso, accedi al tuo Account AWS e scegli Crea caso. Per ulteriori informazioni sulla PNL integrata, consulta Integrazione della PNL con. HealthLake •	20 febbraio 2023

Indice e interrogazione SQL con Amazon Athena	HealthLake supporta l'interrogazione di dati FHIR con SQL utilizzando Amazon Athena. Per ulteriori informazioni, consulta Interrogazione dei HealthLake dati con Amazon Athena. • La funzionalità di interrogazione SQL è abilitata per impostazione predefinita in tutti gli archivi di HealthLake dati creati dopo il 14/11/2022. Se il tuo data store è stato creato prima di questa data, puoi inviare un ticket di supporto per abilitare la funzionalità di query SQL. Crea un caso utilizzando AWS Support Center Console. Per creare il tuo caso, accedi al tuo Account AWS e scegli Crea caso. • Con la funzionalità di interrogazione SQL, le impostazioni IAM per l'accesso HealthLake devono essere aggiornate. Per creare archivi HealthLake dati e concedere l'accesso ad essi in Athena, devi aggiungere la policy `AWSLakeFormationDataAdmin` gestita al tuo utente, gruppo o ruolo IAM. Puoi utilizzare la	14 novembre 2022

	`AWSLakeFormationDataAdmin` policy per creare amministratori di data lake e concedere l'accesso ai data store in Athena. Per ulteriori informazioni, consulta Configurare un utente o un ruolo IAM.	
La dimensione totale dei processi di importazione è aumentata	HealthLake aggiorna il `Total import job size` file per una `StartFHIRImportJob` richiesta a 500 GB. Per ulteriori informazioni, consultare Service quotas.	3 ottobre 2022
risorsa FHIR `Bundle`	HealthLake supporta il tipo di `Bundle` risorsa FHIR per l'elaborazione simultane a di più risorse FHIR. Per ulteriori informazioni, vedere Raggruppamento di risorse FHIR.	5 agosto 2022
Aggiornamenti delle quote per le interazioni FHIR	HealthLake aggiorna le quote per le interazioni di gestione delle risorse FHIR. Per ulteriori informazioni, consultare Service quotas.	16 luglio 2022
Parametro di `_include` ricerca FHIR	HealthLake aggiunge il supporto per il parametro di `_include` ricerca FHIR per restituire risorse aggiuntive in una `search` richiesta. Per ulteriori informazioni, vedere Parametri di ricerca avanzati.	16 luglio 2022

AWS HealthLake è disponibile a livello generale	HealthLake è generalmente disponibile in tutte le regioni supportate. Per ulteriori informazioni, consulta Service endpoints.	15 luglio 2021

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.