aws

Entwicklerhandbuch

AWS HealthLake

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

AWS HealthLake: Entwicklerhandbuch

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

Die Handelsmarken und die Handelsaufmachung von Amazon dürfen nicht in einer Weise in Verbindung mit nicht von Amazon stammenden Produkten oder Services verwendet werden, durch die Kunden irregeführt werden könnten oder Amazon in schlechtem Licht dargestellt oder diskreditiert werden könnte. Alle anderen Handelsmarken, die nicht Eigentum von Amazon sind, gehören den jeweiligen Besitzern, die möglicherweise zu Amazon gehören oder nicht, mit Amazon verbunden sind oder von Amazon gesponsert werden.

Table of Contents

Was ist AWS HealthLake? 1
Wichtiger Hinweis 2
Features 2
Zugehörige Services 3
Zugriff 4
HIPAA 5
Preisgestaltung 5
Erste Schritte 6
Konzepte 6
Autorisierungsstrategie 7
Integriertes NLP 7
Integrierte Analytik 7
Einrichtung 7
Melden Sie sich an für ein AWS-Konto 8
Konfigurieren Sie einen IAM-Benutzer oder eine IAM-Rolle 8
Fügen Sie einen Data Lake-Administratorbenutzer oder eine Rolle hinzu 11
S3-Buckets erstellen 12
Einen Datenspeicher erstellen 13
Richten Sie Importberechtigungen ein 13
Exportberechtigungen einrichten 16
Installiere das AWS CLI 20
Tutorial 21
Verwaltung von Datenspeichern 22
Einen Datenspeicher erstellen 22
Eigenschaften des Datenspeichers abrufen 30
Datenspeicher auflisten 34
Datenspeicher kennzeichnen 37
Taggen eines Datenspeichers 38
Tags für einen Datenspeicher auflisten 41
Entmarkierung eines Datenspeichers 45
Löschen eines Datenspeichers 48
Verwaltung von FHIR-Abonnements 52
Wie funktionieren FHIR-Abonnements 52
Zentrale Komponenten 53

Abonnement-Themen ... 53
Subscriptions (Abonnements) ... 53
Benachrichtigungskanäle ... 54
Payloads für Benachrichtigungen ... 54
Best Practices ... 55
Abonnement-Lebenszyklus ... 55
Ein Abonnement erstellen ... 57
Beispiele für Abonnement-Payloads ... 60
Beispiele für Payloads für Benachrichtigungen ... 65
Auf der Suche nach Abonnements ... 70
Benachrichtigungen filtern ... 73
FHIR-Daten importieren ... 76
Einen Importjob starten ... 78
Eigenschaften von Importaufträgen abrufen ... 83
Importaufträge auflisten ... 87
Verwaltung von FHIR-Ressourcen ... 92
Erstellen einer Ressource ... 94
Eine Ressource lesen ... 97
Lesen des Ressourcenverlaufs ... 99
Lesen des versionsspezifischen Verlaufs ... 102
Eine Ressource aktualisieren ... 104
Bedingtes Update ... 107
Konfiguration der Validierungsebene für Ressourcenupdates ... 108
Eine Ressource ändern ... 109
Unterstützte PATCH-Formate ... 109
Usage ... 110
JSON-Patch-Format ... 110
FHIRPath Patch-Format ... 113
Anforderungsheader ... 115
Beispielantwort ... 115
Behavior ... 116
Fehlerbehandlung ... 116
Zusammenfassung der Funktionen ... 117
Einschränkungen ... 117
Weitere Ressourcen ... 117
Bündelung von Ressourcen ... 118

Bündeln Sie als unabhängige Einheiten ... 122
Bedingt PUTs ... 126
Als eine Einheit bündeln ... 130
Konfiguration der Validierungsebene für Bundles ... 133
Eingeschränkte Unterstützung für den Bundle-Typ „Nachricht“ ... 135
Asynchrone Transaktionen ... 137
Eine Ressource löschen ... 146
Bedingtes Löschen für FHIR ... 148
Idempotenz und Parallelität ... 150
Idempotenzschlüssel ... 150
ETag in AWS HealthLake ... 152
FHIR-Ressourcen werden durchsucht ... 154
Mit GET suchen ... 154
GET-Suchbeispiele ... 157
Mit POST suchen ... 159
Beispiele für die POST-Suche ... 161
Konsistenzstufen für die Suche ... 164
Konsistenzstufen ... 164
Verwendungsbeispiel ... 164
Best Practices ... 165
Exportieren von FHIR-Daten ... 167
Einen Exportjob starten ... 167
Eigenschaften von Exportaufträgen abrufen ... 172
Exportaufträge auflisten ... 176
Codebeispiele ... 182
Grundlagen ... 183
Aktionen ... 183
Integration ... 230
Natürliche Sprachverarbeitung ... 230
NLP-Bibliotheken ... 231
Verwendung von FHIR APIs ... 233
Suchparameter ... 233
Beispielanfragen ... 236
SQL-Index und Abfrage ... 253
Erste Schritte ... 254
Abfragen mit SQL ... 258

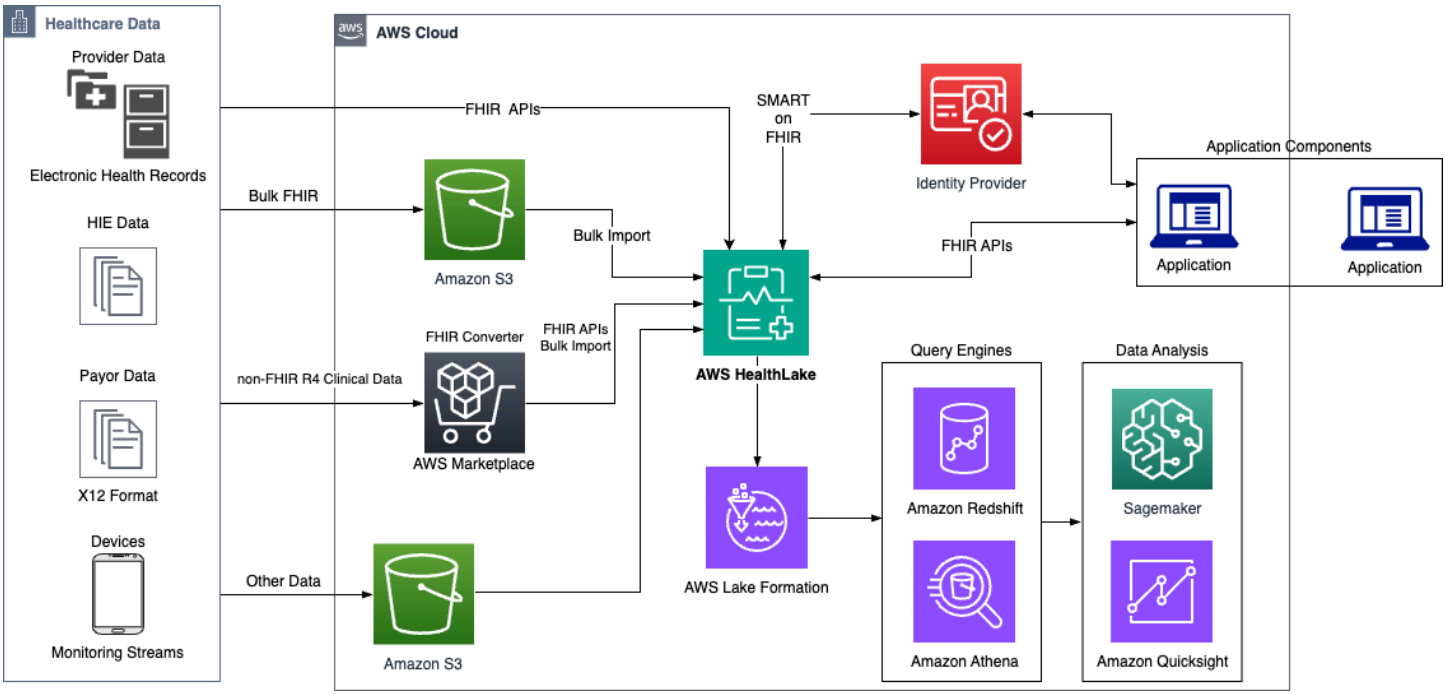
Beispielabfragen 265
Überwachen 272
CloudTrail (API-Aufrufe) 273
AWS HealthLake Informationen in CloudTrail 273
Grundlegendes zu AWS HealthLake Einträgen in Protokolldateien 275
CloudWatch (Metriken) 276
HealthLake Metriken anzeigen 279
Erstellen eines -Alarms 279
EventBridge (Ereignisse) 280
HealthLake Ereignisse wurden gesendet an EventBridge 281
HealthLake Struktur des Ereignisses 282
Sicherheit 297
Datenschutz 298
Verschlüsselung im Ruhezustand 299
AWS-eigener KMS-Schlüssel 299
Kundenseitig verwaltete KMS-Schlüssel 300
Erstellen eines kundenseitig verwalteten Schlüssels 300
Erforderliche IAM-Berechtigungen für die Verwendung eines vom Kunden verwalteten KMS-Schlüssels 302
Verschlüsselung während der Übertragung 309
Identity and Access Management 309
Zielgruppe 309
Authentifizierung mit Identitäten 310
Verwalten des Zugriffs mit Richtlinien 311
Wie AWS HealthLake funktioniert mit IAM 313
Beispiele für identitätsbasierte Richtlinien 319
AWS verwaltete Richtlinien 322
Fehlerbehebung 327
Compliance-Validierung 329
Sicherheit der Infrastruktur 330
Infrastructure as Code 330
HealthLake und CloudFormation Vorlagen 330
Erfahren Sie mehr über CloudFormation 331
VPC-Endpunkte 331
Überlegungen zu HealthLake VPC-Endpunkten 332
Erstellen eines VPC-Schnittstellen-Endpunkts für; HealthLake 332

Erstellen einer VPC-Endpunktrichtlinie für HealthLake 332
Best Practices 333
Ausfallsicherheit 334
Referenz 335
SMART auf FHIR 335
Erste Schritte 336
Authentifizierung 339
OAuth 2.0-Bereiche 341
Token-Validierung 344
Feinkörnige Autorisierung 357
Discovery-Dokument 358
Beispiel für eine Anfrage 359
FHIR R4 360
Erklärung zur Leistungsfähigkeit 361
Profilvalidierungen 362
Ressourcentypen 369
Suchparameter 372
$ Operationen 385
Compliance 538
CMS 539
HealthLake 545
Endpunkte und Kontingente 545
Vorinstallierte Datentypen 557
Beispielprojekte 558
Fehlerbehebung 558
Arbeitet mit AWS SDKs 567
Versionen 569
..... dxcvi

Was ist AWS HealthLake?

AWS HealthLake ist ein HIPAA-fähiger Dienst zum Speichern, Analysieren und Teilen von Gesundheitsdaten in der Cloud unter Verwendung der R4-Spezifikation Fast Healthcare Interoperability Resources (FHIR). HealthLake Zu den Anwendungsfällen gehören:

- Gesundheitsdaten für Unternehmen — Verwalten und teilen Sie FHIR R4-Gesundheitsdaten direkt von dort aus und sorgen AWS Cloud Sie gleichzeitig für hohe Leistung und Verfügbarkeit.
- Interoperabilität im Gesundheitswesen — Unterstützen Sie die Einhaltung des 21st Century Cures Act für den Patientenzugriff durch einen vollständig verwalteten FHIR-Datenspeicher durch Kunden.
- Natürliche Sprachverarbeitung (NLP) — Nutzen Sie integrierte NLP-Modelle, um aussagekräftige medizinische Informationen aus unstrukturierten Gesundheitsdaten zu extrahieren.
- Multimodale Analyse — Kombinieren Sie HealthLake Daten mit Daten und AWS HealthImaging AWS HealthOmics Daten, um Erkenntnisse für die Präzisionsmedizin zu gewinnen.



Themen

- Wichtiger Hinweis
- Eigenschaften von AWS HealthLake

- Verwandte Dienstleistungen AWS
- Zugreifen AWS HealthLake
- HIPAA-Eignung und Datensicherheit
- Preisgestaltung

Wichtiger Hinweis

AWS HealthLake ist kein Ersatz für professionelle medizinische Beratung, Diagnose oder Behandlung und ist nicht dazu bestimmt, Krankheiten oder Gesundheitsprobleme zu heilen, zu behandeln, zu mildern, zu verhindern oder zu diagnostizieren. Sie sind dafür verantwortlich, im Rahmen der Verwendung von Produkten Dritter, die als Grundlage für klinische Entscheidungen dienen AWS HealthLake, Untersuchungen am Menschen durchzuführen, auch in Verbindung mit Produkten von Drittanbietern. AWS HealthLake darf nur in der Patientenversorgung oder in klinischen Szenarien verwendet werden, nachdem es von geschultem medizinischem Fachpersonal mit fundiertem medizinischem Urteilsvermögen überprüft wurde.

Eigenschaften von AWS HealthLake

AWS HealthLake bietet die folgenden Funktionen.

Importieren Sie FHIR R4-Gesundheitsdaten

> Mit der HealthLake nativen Importaktion können Sie Ihre FHIR-Daten einfach von einem Amazon S3 S3-Bucket in einen HealthLake Datenspeicher migrieren, einschließlich klinischer Notizen, Laborberichte, Versicherungsansprüche und mehr. HealthLake unterstützt die FHIR R4-Spezifikation für den Austausch von Gesundheitsdaten. Bei Bedarf können Sie mit einem AWS HealthLake Partner zusammenarbeiten, um Ihre Gesundheitsdaten in das FHIR R4-Format zu konvertieren.

Speichern Sie Gesundheitsdaten auf sichere, konforme und überprüfbare Weise

> Ein HealthLake Datenspeicher hilft dabei, Gesundheitsdaten zu indexieren, sodass sie abgefragt werden können. Der Datenspeicher bietet einen vollständigen Überblick über die Krankengeschichte jedes Patienten in chronologischer Reihenfolge und erleichtert den Informationsaustausch mithilfe der FHIR R4-Spezifikation. Und er läuft ständig, um Ihren Index auf dem neuesten Stand zu halten, und bietet Ihnen die Möglichkeit, die Informationen

jederzeit mithilfe von standardmäßigen FHIR R4-Interaktionen mit stabilem Primärspeicher und Indexskalierung zu durchsuchen.

Nutzen Sie den transaktionalen FHIR-Server

Nutzen Sie FHIR APIs für die standardmäßige Ressourcenvalidierung, SMART für die FHIR-Autorisierung und die FHIR-API-Exportfunktionen für Massendaten, um die Vereinheitlichung und Analyse Ihrer Daten zu unterstützen, um die Betriebskosten zu senken und die Entscheidungsfindung zu verbessern. HealthLake unterstützt die Konformität der Kunden mit den neuesten ONC- und CMS-Regulierungsstandards, darunter: HL7 FHIR R4 APIs, FHIR Bulk Data Access, US Core IG STU, HL7 SMART App Launch Framework IG, OAuth 2.0 und OpenID Connect.

Transformieren Sie unstrukturierte medizinische Daten mithilfe von NLP

Integrierte medizinische Verarbeitung natürlicher Sprache (NLP) transformiert alle medizinischen Rohtextdaten in einem Datenspeicher, um aussagekräftige Informationen aus unstrukturierten HealthLake Gesundheitsdaten zu verstehen und daraus zu extrahieren. Mit integriertem medizinischem NLP können Sie automatisch Entitäten, Entitätsbeziehungen, Entitätsmerkmale und geschützte Gesundheitsinformationen (PHI) aus Ihrem medizinischen Text extrahieren. Die NLP-extrahierten Entitäten werden als native FHIR R4-Ressourcen in einem HealthLake Datenspeicher gespeichert und können über FHIR R4 APIs oder Amazon Athena (SQL) abgerufen werden.

Verwandte Dienstleistungen AWS

AWS HealthLake bietet eine enge Integration mit anderen AWS Diensten. Kenntnisse der folgenden Dienste sind hilfreich, um sie voll auszuschöpfen HealthLake.

- AWS Identity and Access Management— Verwenden Sie IAM, um Identitäten und den Zugriff auf Ressourcen sicher zu HealthLake verwalten.
- Amazon Simple Storage Service — Verwenden Sie Amazon S3 als Staging-Bereich für den Import von DICOM-Daten in. HealthLake
- AWS CloudTrail— Wird verwendet CloudTrail , um HealthLake Benutzeraktivitäten und API-Nutzung zu verfolgen.
- Amazon CloudWatch — Wird CloudWatch zur Beobachtung und Überwachung von HealthLake Ressourcen verwendet.

- AWS CloudFormation— Wird verwendet CloudFormation , um IaC-Vorlagen (Infrastructure as Code) zu implementieren, in HealthLake denen Ressourcen erstellt werden.
- AWS PrivateLink— Verwenden Sie Amazon VPC, um Konnektivität zwischen HealthLake und Amazon Virtual Private Cloud herzustellen, ohne Daten dem Internet zugänglich zu machen.
- Amazon EventBridge — Wird verwendet, EventBridge um skalierbare, ereignisgesteuerte Anwendungen zu erstellen, indem Regeln erstellt werden, die HealthLake Ereignisse an Ziele weiterleiten.
- AWS Lake Formation— Verwenden Sie Lake Formation, um HealthLake Daten für Analysen und maschinelles Lernen zentral zu verwalten, zu sichern und gemeinsam zu nutzen.
- Amazon Athena — Verwenden Sie Athena, um HealthLake Daten mit SQL abzufragen, um eine tiefere Analyse zu ermöglichen.

Zugreifen AWS HealthLake

Sie können AWS HealthLake mit dem AWS-Managementkonsole, AWS Command Line Interface und dem zugreifen AWS SDKs. Dieses Handbuch enthält Verfahrensanweisungen für das AWS-Managementkonsole und sowie Codebeispiele für AWS CLI und AWS SDKs.

AWS Command Line Interface (AWS CLI)

Das AWS CLI enthält Befehle für eine Vielzahl von AWS Produkten und wird unter Windows, Mac und Linux unterstützt. Weitere Informationen finden Sie im AWS Command Line Interface - Benutzerhandbuch.

AWS SDKs

AWS SDKs stellt Bibliotheken, Codebeispiele und andere Ressourcen für Softwareentwickler bereit. Diese Bibliotheken bieten grundlegende Funktionen, mit denen Aufgaben wie das kryptografische Signieren Ihrer Anfragen, das Wiederholen von Anfragen und das Behandeln von Fehlerantworten automatisiert werden können. Weitere Informationen finden Sie unter Tools, auf denen Sie aufbauen können. AWS

AWS-Managementkonsole

Das AWS-Managementkonsole bietet eine webbasierte Benutzeroberfläche für die Verwaltung HealthLake und die zugehörigen Ressourcen. Wenn Sie sich für ein AWS Konto registriert haben, können Sie sich bei der HealthLake Konsole anmelden.

HIPAA-Eignung und Datensicherheit

Dies ist ein HIPAA-berechtigter Service. Weitere Informationen AWS zum US-amerikanischen Health Insurance Portability and Accountability Act von 1996 (HIPAA) und zur Nutzung von AWS Diensten zur Verarbeitung, Speicherung und Übertragung geschützter Gesundheitsinformationen (PHI) finden Sie unter HIPAA Overview.

Verbindungen, die PHI und personenbezogene Daten (PII) HealthLake enthalten, müssen verschlüsselt sein. Standardmäßig HealthLake verwenden alle Verbindungen HTTPS über TLS. HealthLake speichert verschlüsselte Kundeninhalte und arbeitet nach dem Modell der AWS gemeinsamen Verantwortung.

Preisgestaltung

HealthLake Preisinformationen finden Sie unter AWS HealthLake Preise. Verwenden Sie den HealthLake Preisrechner, um die Kosten zu schätzen.

Erste Schritte mit AWS HealthLake

Um mit der Nutzung zu beginnen AWS HealthLake, richten Sie ein AWS Konto ein und erstellen Sie einen AWS Identity and Access Management Benutzer. Um die AWS CLIoder die verwenden zu können AWS SDKs, müssen Sie sie installieren und konfigurieren.

> **Note**
>
> Das Referenz Kapitel dieses Handbuchs enthält unterstützende Inhalte für SMART auf FHIR, FHIR R4 und. AWS HealthLake Informationen zu SMART finden Sie beispielsweise in der FHIR-Konfiguration, den unterstützten FHIR-Profilvalidierungen und Endpunkten. HealthLake

Nachdem Sie sich mit den HealthLake Konzepten und der Einrichtung vertraut gemacht haben, steht Ihnen ein kurzes Tutorial mit Codebeispielen zur Verfügung, das Ihnen den Einstieg erleichtern soll.

Themen

- AWS HealthLake Konzepte
- Einrichtung AWS HealthLake
- AWS HealthLake Tutorial

AWS HealthLake Konzepte

Die folgenden Begriffe und Konzepte sind von zentraler Bedeutung für Ihr Verständnis und Ihre Verwendung von AWS HealthLake.

Konzepte

- Strategie zur Autorisierung von Datenspeichern
- Integriertes NLP
- Integrierte Analytik

Strategie zur Autorisierung von Datenspeichern

Ein HealthLake Datenspeicher ist ein Speicher für FHIR R4-Gesundheitsdaten, die sich in einem einzigen Speicher befinden. AWS-Region HealthLake unterstützt die folgenden Autorisierungsstrategien für Datenspeicher.

- SigV4-Autorisierung — HealthLake autorisiert FHIR-API-Aufrufe mithilfe der AWS Signature Version 4-Autorisierung (Sigv4).
- SMART bei FHIR-Autorisierung — HealthLake autorisiert FHIR-API-Aufrufe mithilfe von SMART (Substituable Medical Applications and Reusable Technologies) im Rahmen der FHIR-Autorisierung.

Weitere Informationen finden Sie unter Einen HealthLake Datenspeicher erstellen.

Integriertes NLP

AWS HealthLake lässt sich in HIPAA-fähige Bibliotheken zur Verarbeitung natürlicher Sprache (NLP) integrieren, um aussagekräftige Gesundheitsdaten aus unstrukturiertem medizinischem Text zu extrahieren. Die NLP-Bibliotheken identifizieren medizinische Entitäten wie Erkrankungen, Medikamente, Dosierungen, Tests, Behandlungen und Verfahren. Sie erkennen Beziehungen zwischen den Entitäten und verknüpfen sie mit Bibliotheken für medizinische Ontologie wie ICD-10-CM und. RxNorm Weitere Informationen finden Sie unter Integrierte Verarbeitung natürlicher Sprache (NLP) für HealthLake.

Integrierte Analytik

AWS HealthLake geht über FHIR hinaus `search` und `bundle` APIs bietet integrierte Analysen für die Abfrage und Analyse großer Mengen an Gesundheitsdaten. Generiert beim Import HealthLake automatisch Tabellen für den SQL-Index und die SQL-Abfrage. Auf diese Weise können Sie umsetzbare Erkenntnisse aus komplexen Gesundheitsdaten gewinnen, ohne dass umfangreiche Datenentwicklungsarbeiten erforderlich sind. Weitere Informationen erhalten Sie unter HealthLake Daten mit Amazon Athena abfragen und AWS HealthLake Beispielprojekte.

Einrichtung AWS HealthLake

In diesem Kapitel richten Sie mit dem die erforderlichen Berechtigungen ein, AWS-Managementkonsole um mit der Verwendung AWS HealthLake und Erstellung eines Datenspeichers

zu beginnen. Um Berechtigungen für die Erstellung eines Datenspeichers einzurichten, erstellen Sie einen IAM-Benutzer oder eine IAM-Rolle, bei der es sich um einen Data HealthLake Lake-Administrator und -Administrator handelt. Sie machen diesen Benutzer in AWS Lake Formation zum Data Lake-Administrator. Der Data Lake-Administrator gewährt Lake Formation Zugriff auf Ressourcen, die für die Verwendung von Amazon Athena zur Abfrage eines Datenspeichers erforderlich sind. Nachdem Sie einen HealthLake Datenspeicher erstellt haben, können Sie Berechtigungen für den Import und Export von Dateien einrichten.

Themen

- Melden Sie sich an für ein AWS-Konto
- Konfigurieren Sie einen zu verwendenden IAM-Benutzer oder eine IAM-Rolle HealthLake (IAM-Administrator)
- Einen Benutzer oder eine Rolle als Data Lake-Administrator in Lake Formation hinzufügen (IAM-Administrator)
- S3-Buckets erstellen
- Einen Datenspeicher erstellen
- Berechtigungen für Importaufträge einrichten
- Berechtigungen für Exportaufträge einrichten
- Installiere das AWS CLI

Melden Sie sich an für ein AWS-Konto

Um loszulegen AWS, benötigen Sie eine AWS-Konto. Informationen zum Erstellen eines AWS-Konto finden Sie unter Erste Schritte mit einem AWS-Konto im AWS -Kontenverwaltung Referenzhandbuch.

Konfigurieren Sie einen zu verwendenden IAM-Benutzer oder eine IAM-Rolle HealthLake (IAM-Administrator)

> Persona: IAM-Administrator
>
> Ein Benutzer, der IAM-Benutzer und -Rollen erstellen und Data Lake-Administratoren hinzufügen kann.

Diese Schritte in diesem Thema müssen von einem IAM-Administrator ausgeführt werden.

Um Ihren HealthLake Datenspeicher mit Athena zu verbinden, müssen Sie einen IAM-Benutzer oder eine IAM-Rolle erstellen, bei der es sich um einen Data Lake-Administrator und einen HealthLake Administrator handelt. Dieser neue Benutzer oder diese neue Rolle gewährt Zugriff auf Ressourcen, die sich in einem Datenspeicher über AWS Lake Formation befinden, und die `AmazonHealthLakeFullAccess` AWS verwaltete Richtlinie wird dem Benutzer oder der Rolle hinzugefügt.

> **Important**
>
> Ein IAM-Benutzer oder eine IAM-Rolle, die ein Data Lake-Administrator ist, kann keine neuen Data Lake-Administratoren erstellen. Um einen weiteren Data Lake-Administrator hinzuzufügen, müssen Sie einen IAM-Benutzer oder eine IAM-Rolle verwenden, der bzw. der Zugriff gewährt `AdministratorAccess` wurde.

Um einen Administrator zu erstellen

1. Fügen Sie die von **`AmazonHealthlakeFullAccess`** IAM AWS verwaltete Richtlinie einem Benutzer oder einer Rolle in Ihrer Organisation hinzu.

   Wenn Sie mit der Erstellung eines IAM-Benutzers nicht vertraut sind, finden Sie weitere Informationen unter Erstellen eines IAM-Benutzers und Überblick über AWS IAM-Richtlinien im IAM-Benutzerhandbuch.
2. Gewähren Sie dem IAM-Benutzer oder der IAM-Rolle Zugriff auf AWS Lake Formation.

   - Fügen Sie einem Benutzer oder einer Rolle in Ihrer Organisation die folgende von IAM AWS verwaltete Richtlinie hinzu: **`AWSLakeFormationDataAdmin`**

     > **Note**
     >
     > Die `AWSLakeFormationDataAdmin` Richtlinie gewährt Zugriff auf alle Ressourcen von AWS Lake Formation. Es wird empfohlen, immer die Mindestberechtigungen zu verwenden, die für die Ausführung Ihrer Aufgabe erforderlich sind. Weitere Informationen finden Sie unter Bewährte Methoden für IAM im IAM-Benutzerhandbuch.
3. Fügen Sie dem Benutzer oder der Rolle die folgende Inline-Richtlinie hinzu. Weitere Informationen finden Sie unter Inline-Richtlinien im IAM-Benutzerhandbuch.

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": "*"
        }
    ]
}
```

Weitere Informationen zu dieser `AWSLakeFormationDataAdmin` Richtlinie finden Sie unter Lake Formation Personas and IAM Permissions Reference im AWS Lake Formation Developer Guide.

Einen Benutzer oder eine Rolle als Data Lake-Administrator in Lake Formation hinzufügen (IAM-Administrator)

> **Note**
>
> Dieser Schritt ist erforderlich, wenn Sie integrierenSQL-Index und Abfrage.

Als Nächstes muss der IAM-Administrator den Benutzer oder die Rolle hinzufügen, die im vorherigen Schritt als Data Lake-Administrator in Lake Formation erstellt wurden.

Um einen IAM-Benutzer oder eine IAM-Rolle als Data Lake-Administrator hinzuzufügen

1. Öffnen Sie die AWS Lake Formation Formation-Konsole: https://console.aws.amazon.com/lakeformation/

   > **Note**
   >
   > Wenn Sie Lake Formation zum ersten Mal besuchen, wird ein Dialogfeld Willkommen bei Lake Formation angezeigt, in dem Sie aufgefordert werden, einen Lake Formation-Administrator zu definieren.

2. Weisen Sie dem neuen Benutzer oder der neuen Rolle die Rolle eines AWS Lake Formation Data Lake-Administrators zu.
   - Option 1: Wenn Sie das Dialogfeld Willkommen bei Lake Formation erhalten haben.
     1. Wählen Sie Weitere AWS Benutzer oder Rollen hinzufügen aus.
     2. Wählen Sie den Abwärtspfeil (▼).
     3. Wählen Sie den HealthLake Administrator aus, der auch Lake Formation-Administratoren sein soll.
     4. Wählen Sie Erste Schritte.
   - Option 2: Verwenden Sie den Navigationsbereich (☰).
     1. Wählen Sie den Navigationsbereich (☰).
     2. Wählen Sie unter Berechtigungen die Option Administrative Rollen und Aufgaben aus.

3. Wählen Sie im Abschnitt Data Lake-Administratoren die Option Administratoren auswählen aus.
4. Wählen Sie im Dialogfeld Data Lake-Administratoren verwalten den Abwärtspfeil (▼) aus.
5. Wählen Sie als Nächstes die HealthLake Administratoren, Benutzer oder Rollen aus, die Sie auch Lake Formation-Administratoren sein möchten, oder suchen Sie nach ihnen.
6. Wählen Sie Speichern.

3. Ändern Sie die Standardsicherheitseinstellungen, die von Lake Formation verwaltet werden sollen. Die HealthLake Datenspeicherressourcen müssen von Lake Formation und nicht von IAM verwaltet werden. Informationen zur Aktualisierung finden Sie unter Ändern des Standardberechtigungsmodells im AWS Lake Formation Developer Guide.

S3-Buckets erstellen

Um FHIR R4-Daten in zwei Amazon S3-Buckets zu importieren AWS HealthLake, werden zwei Amazon S3 S3-Buckets empfohlen. Der Amazon S3 S3-Eingabe-Bucket enthält die zu importierenden FHIR-Daten und HealthLake liest aus diesem Bucket. Der Amazon S3 S3-Ausgabe-Bucket speichert die Verarbeitungsergebnisse des Importjobs und HealthLake schreibt (Logs) in diesen Bucket.

> **Note**
>
> Aufgrund der AWS Identity and Access Management (IAM-) Richtlinie müssen Ihre Amazon S3 S3-Bucket-Namen eindeutig sein. Weitere Informationen finden Sie unter Benennungsregeln für Buckets im Benutzerhandbuch zu Amazon Simple Storage Service.

Für die Zwecke dieses Handbuchs geben wir die folgenden Amazon S3 S3-Eingabe- und Ausgabe-Buckets an, wenn wir später in diesem Abschnitt Importberechtigungen einrichten.

- Eingabe-Bucket: `arn:aws:s3:::amzn-s3-demo-source-bucket`
- Ausgabe-Bucket: `arn:aws:s3:::amzn-s3-demo-logging-bucket`

Weitere Informationen finden Sie unter Erstellen eines Buckets im Amazon S3 S3-Benutzerhandbuch.

Einen Datenspeicher erstellen

Ein HealthLake Datenspeicher ist ein Repository mit FHIR R4-Daten, die sich in einer einzigen Region befinden. AWS Ein AWS Konto kann null oder viele Datenspeicher haben. HealthLake unterstützt zwei Autorisierungsstrategien für Datenspeicher.

> **Wichtig**
>
> Bevor Sie einen HealthLake Datenspeicher erstellen, sollten Sie die Service Control Policies (SCPs) in Ihrer AWS Organisation überprüfen, die die Erstellung oder Verwaltung von HealthLake Ressourcen einschränken könnten. SCPs können die erfolgreiche Erstellung von HealthLake Datenspeichern verhindern, selbst wenn Ihre IAM-Berechtigungen korrekt eingerichtet sind.
>
> A `datastoreID` wird generiert, wenn Sie einen HealthLake Datenspeicher erstellen. Sie müssen die verwendendatastoreID, wenn Sie die Importberechtigungen später in diesem Abschnitt einrichten.

Informationen zum Erstellen eines HealthLake Datenspeichers finden Sie unterEinen HealthLake Datenspeicher erstellen.

Berechtigungen für Importaufträge einrichten

Bevor Sie Dateien in einen Datenspeicher importieren, müssen Sie die HealthLake Erlaubnis für den Zugriff auf Ihre Eingabe- und Ausgabe-Buckets in Amazon S3 erteilen. Um HealthLake Zugriff zu gewähren, erstellen Sie eine IAM-Servicerolle für HealthLake, fügen der Rolle eine Vertrauensrichtlinie hinzu, um Rollenübernahmeberechtigungen zu gewähren HealthLake , und fügen der Rolle eine Berechtigungsrichtlinie hinzu, die ihr Zugriff auf Ihre Amazon S3 S3-Buckets gewährt.

Wenn Sie einen Importauftrag erstellen, geben Sie den Amazon-Ressourcennamen (ARN) dieser Rolle für die an`DataAccessRoleArn`. Weitere Informationen zu IAM-Rollen und Vertrauensrichtlinien finden Sie unter IAM-Rollen.

Nachdem Sie die Berechtigungen eingerichtet haben, können Sie Dateien mit einem Importauftrag in Ihren Datenspeicher importieren. Weitere Informationen finden Sie unter Einen FHIR-Importjob starten.

Um Importberechtigungen einzurichten

1. Falls dies noch nicht geschehen ist, erstellen Sie einen Amazon S3 S3-Ziel-Bucket für Ausgabeprotokolldateien. Der Amazon S3 S3-Bucket muss sich in derselben AWS Region wie der Service befinden, und Block Public Access muss für alle Optionen aktiviert sein. Weitere Informationen finden Sie unter Blockieren des öffentlichen Zugriffs mit Amazon S3. Für die Verschlüsselung muss auch ein KMS-Schlüssel Amazon-owned oder ein kundeneigener KMS-Schlüssel verwendet werden. Weitere Informationen zur Verwendung von KMS-Schlüsseln finden Sie unter Amazon Key Management Service.
2. Erstellen Sie eine Datenzugriffs-Servicerolle für HealthLake und erteilen Sie dem HealthLake Service die Erlaubnis, diese zu übernehmen. Beachten Sie dabei die folgende Vertrauensrichtlinie. HealthLake verwendet dies, um den Amazon S3 S3-Ausgabe-Bucket zu schreiben.

   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. Fügen Sie der Datenzugriffsrolle eine Berechtigungsrichtlinie hinzu, die ihr den Zugriff auf den Amazon S3 S3-Bucket ermöglicht. `amzn-s3-demo-bucket`Ersetzen Sie es durch den Namen Ihres Buckets.

   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"
    }]
}
```

Berechtigungen für Exportaufträge einrichten

Bevor Sie Dateien aus einem Datenspeicher exportieren, müssen Sie die HealthLake Erlaubnis für den Zugriff auf Ihren Ausgabe-Bucket in Amazon S3 erteilen. Um HealthLake Zugriff zu gewähren, erstellen Sie eine IAM-Servicerolle fürHealthLake, fügen der Rolle eine Vertrauensrichtlinie hinzu, um Rollenübernahmeberechtigungen zu gewähren HealthLake , und fügen der Rolle eine Berechtigungsrichtlinie hinzu, die ihr Zugriff auf Ihren Amazon S3 S3-Bucket gewährt.

Wenn Sie bereits eine Rolle für erstellt haben HealthLake, können Sie sie wiederverwenden und ihr die zusätzlichen Berechtigungen für Ihren Amazon S3 S3-Exportbucket gewähren, die in diesem Thema aufgeführt sind. Weitere Informationen zu IAM-Rollen und Vertrauensrichtlinien finden Sie unter IAM-Richtlinien und -Berechtigungen.

Wichtig

HealthLake unterstützt sowohl native SDK-Exportanfragen als auch den FHIR R4-Vorgang. `$export` Je nachdem, für welche Export-API Sie sich entscheiden, müssen separate IAM-Aktionen bereitgestellt werden. Auf diese Weise können Sie die Verwaltung `allow` und die `deny` Berechtigungen getrennt voneinander verwalten. Wenn Sie sowohl den HealthLake SDK- als auch den FHIR-REST-API-Export einschränken möchten, müssen Sie den einzelnen IAM-Aktionen die Zugriffsrechte verweigern. Änderungen der IAM-Benutzerberechtigungen sind nicht erforderlich, wenn Sie Benutzern vollen Zugriff auf gewähren. HealthLake

Verwenden AWS CLI und AWS SDKs:

Die folgenden systemeigenen HealthLake Aktionen sind für den Export von Daten aus einem Datenspeicher mithilfe der AWS SDKs AWS CLI und verfügbar:

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

Verwendung von FHIR-APIs:

Die folgenden IAM-Aktionen sind für den Export von Daten aus einem HealthLake Datenspeicher und für das Abbrechen (Löschen) eines Exportauftrags mithilfe der FHIR-Operation verfügbar: `$export`

POST:

- StartFHIRExportJobWithPost

GET:

- StartFHIRExportJobWithGet
- DescribeFHIRExportJobWithGet
- GetExportedFile

DELETE:

- CancelFHIRExportJobWithDelete

Der Benutzer oder die Rolle, der bzw. die Berechtigungen einrichtet, muss berechtigt sein, Rollen und Richtlinien zu erstellen und Rollen Richtlinien zuzuordnen. Die folgende IAM-Richtlinie gewährt diese Berechtigungen.

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"
                }
            }
        }
    ]
}
```

Um Exportberechtigungen einzurichten

1. Falls dies noch nicht geschehen ist, erstellen Sie einen Amazon S3 S3-Ziel-Bucket für die Daten, die Sie aus Ihrem Datenspeicher exportieren möchten. Der Amazon S3 S3-Bucket muss sich in derselben AWS-Region wie der Service befinden, und Block Public Access muss für alle Optionen aktiviert sein. Weitere Informationen finden Sie unter Blockieren des öffentlichen Zugriffs mit Amazon S3. Für die Verschlüsselung muss auch ein KMS-Schlüssel Amazon-owned oder ein kundeneigener KMS-Schlüssel verwendet werden. Weitere Informationen zur Verwendung von KMS-Schlüsseln finden Sie unter Amazon Key Management Service.
2. Falls Sie dies noch nicht getan haben, erstellen Sie eine Datenzugriffs-Servicerolle für HealthLake und erteilen Sie dem HealthLake Service die Erlaubnis, diese zu übernehmen. Beachten Sie dabei die folgende Vertrauensrichtlinie. HealthLakeverwendet dies, um den Amazon S3 S3-Ausgabe-Bucket zu schreiben. Wenn Sie bereits einen in erstellt habenBerechtigungen für Importaufträge einrichten, können Sie ihn wiederverwenden und ihm im nächsten Schritt Berechtigungen für Ihren Amazon S3 S3-Bucket erteilen.

   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. Fügen Sie der Datenzugriffsrolle eine Berechtigungsrichtlinie hinzu, die ihr den Zugriff auf Ihren Amazon S3 S3-Ausgabe-Bucket ermöglicht. `amzn-s3-demo-bucket`Ersetzen Sie es durch den Namen Ihres Buckets.

   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"
    }]
}
```

Installiere das AWS CLI

Der AWS CLI ist erforderlich, um die Eigenschaften von HealthLake Import- und Exportaufträgen zu beschreiben und aufzulisten. Sie können diese Informationen auch mithilfe von HealthLake SDKs anfordern.

Um das einzurichten AWS CLI

1. Herunterladen und Konfigurieren von AWS CLI. Eine Anleitung finden Sie unter den folgenden Themen im AWS Command Line Interface -Benutzerhandbuch.

   - Installation oder Aktualisierung der neuesten Version von AWS CLI
   - Erste Schritte mit dem AWS CLI

2. Fügen Sie der AWS CLI `config` Datei ein benanntes Profil für den Administrator hinzu. Sie verwenden dieses Profil, wenn Sie die AWS CLI Befehle ausführen. Gemäß dem Sicherheitsprinzip der geringsten Rechte empfehlen wir Ihnen, eine separate IAM-Rolle mit spezifischen Rechten für die auszuführenden Aufgaben zu erstellen. Weitere Informationen zu benannten Profilen finden Sie im AWS Command Line Interface Benutzerhandbuch unter Konfiguration und Einstellungen für Anmeldeinformationsdateien.

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

3. Überprüfen Sie das Setup mit dem folgenden `help` Befehl.

   ```
   aws healthlake help
   ```

Wenn der richtig konfiguriert AWS CLI ist, sehen Sie eine kurze Beschreibung AWS HealthLake und eine Liste der verfügbaren Befehle.

AWS HealthLake Tutorial

Zielsetzung

In diesem Tutorial importieren Sie FHIR R4-Daten mithilfe HealthLake systemeigener Aktionen in einen HealthLake Datenspeicher. Als Nächstes verwalten (erstellen, lesen, aktualisieren, löschen) Sie eine FHIR-Ressource mithilfe von FHIR. RESTful APIs Zum Abschluss des Tutorials exportieren Sie FHIR-Daten mithilfe systemeigener Aktionen. HealthLake

Voraussetzungen

Alle unter aufgeführten Verfahren Einrichtung sind erforderlich, um dieses Tutorial abzuschließen.

Schritte des Tutorials

1. Starten Sie den FHIR-Importjob
2. Holen Sie sich die Eigenschaften des FHIR-Importauftrags
3. FHIR-Ressource erstellen
4. Lesen Sie die FHIR-Ressource
5. Aktualisieren Sie die FHIR-Ressource
6. FHIR-Ressource löschen
7. Exportieren Sie FHIR-Daten
8. Datenspeicher löschen

Verwaltung von Datenspeichern mit AWS HealthLake

Mit AWS HealthLake erstellen und verwalten Sie Datenspeicher für FHIR R4-Ressourcen. Wenn Sie einen HealthLake Datenspeicher erstellen, wird ein FHIR-Daten-Repository über einen API-Endpunkt verfügbar gemacht. RESTful Sie können wählen, ob Sie Synthea Open-Source-FHIR R4-Gesundheitsdaten in Ihren Datenspeicher importieren (vorab laden) möchten, wenn Sie ihn erstellen. Weitere Informationen finden Sie unter Vorinstallierte Datentypen.

> **Wichtig**
>
> HealthLake unterstützt zwei Arten von Autorisierungsstrategien für FHIR-Datenspeicher: AWS SIGv4 oder SMART auf FHIR. Sie müssen eine der Autorisierungsstrategien wählen, bevor Sie einen HealthLake FHIR-Datenspeicher erstellen. Weitere Informationen finden Sie unter Strategie zur Autorisierung von Datenspeichern.

Um die FHIR-bezogenen Funktionen (Verhaltensweisen) eines aktiven HealthLake Datenspeichers zu ermitteln, rufen Sie dessen Capability Statement auf.

In den folgenden Themen wird beschrieben, wie Sie mithilfe von HealthLake Cloud-nativen Aktionen FHIR-Datenspeicher mithilfe von, und erstellen, beschreiben, auflisten, taggen und löschen können. AWS CLI AWS SDKs AWS-Managementkonsole

Themen

- Einen HealthLake Datenspeicher erstellen
- Eigenschaften des HealthLake Datenspeichers abrufen
- HealthLake Datenspeicher auflisten
- HealthLake Datenspeicher kennzeichnen
- Löschen eines HealthLake Datenspeichers

Einen HealthLake Datenspeicher erstellen

Wird verwendet`CreateFHIRDatastore`, um einen AWS HealthLake Datenspeicher zu erstellen, der der FHIR R4-Spezifikation entspricht. HealthLake Datenspeicher werden zum Importieren, Verwalten, Suchen und Exportieren von FHIR-Daten verwendet. Sie können wählen, ob Sie Synthea-

Open-Source-FHIR R4-Gesundheitsdaten in Ihren Datenspeicher importieren (vorab laden) möchten, wenn Sie ihn erstellen. Weitere Informationen finden Sie unter Vorinstallierte Datentypen.

> **Wichtig**
>
> HealthLake unterstützt zwei Arten von Autorisierungsstrategien für FHIR-Datenspeicher: AWS SIGv4 oder SMART auf FHIR. Sie müssen eine der Autorisierungsstrategien wählen, bevor Sie einen HealthLake FHIR-Datenspeicher erstellen. Weitere Informationen finden Sie unter Strategie zur Autorisierung von Datenspeichern.

Wenn Sie einen HealthLake Datenspeicher erstellen, wird ein FHIR-Datenrepository über einen RESTful API-Endpunkt verfügbar gemacht. Nachdem Sie Ihren HealthLake Datenspeicher erstellt haben, können Sie dessen Capability Statement anfordern, um alle zugehörigen FHIR-bezogenen Fähigkeiten (Verhaltensweisen) zu finden.

Die folgenden Menüs enthalten Beispiele für AWS CLI und AWS SDKs und ein Verfahren für. AWS-Managementkonsole Weitere Informationen finden Sie unter `CreateFHIRDatastore` in der AWS HealthLake -API-Referenz.

Um einen HealthLake Datenspeicher zu erstellen

Wählen Sie ein Menü, das Ihren Zugriffspräferenzen für entspricht AWS HealthLake.

AWS CLI und SDKs

CLI

AWS CLI

Beispiel 1: Erstellen Sie einen SIGV4-fähigen Datenspeicher HealthLake

Das folgende `create-fhir-datastore` Beispiel zeigt, wie Sie einen neuen Datenspeicher in erstellen. AWS HealthLake

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

Ausgabe:

```
{
    "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)"
}
```

Beispiel 2: Erstellen Sie einen SMART-Datenspeicher auf einem FHIR-fähigen Datenspeicher HealthLake

Das folgende `create-fhir-datastore` Beispiel zeigt, wie Sie einen neuen SMART-on-FHIR-fähigen Datenspeicher in erstellen. 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
```

Inhalt von `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\"]}"
}
```

Ausgabe:

```
{
    "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)"
}
```

- Einzelheiten zur API finden Sie unter Create FHIRDatastore in AWS CLI Command Reference.

Python

SDK für 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
```

Der folgende Code zeigt ein Beispiel für Parameter für einen SMART-Datenspeicher auf einem FHIR-fähigen Datenspeicher 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
            )
```

- API-Einzelheiten finden Sie unter Create FHIRDatastore in AWS SDK for Python (Boto3) API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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.
```

- Einzelheiten zur API finden Sie unter API-Referenz zum Erstellen FHIRDatastore im AWS SDK für SAP ABAP.

> **Beispiel für die Verfügbarkeit**
>
> Sie können nicht finden, was Sie brauchen? Fordern Sie über den Link Feedback geben in der rechten Seitenleiste dieser Seite ein Codebeispiel an.

AWS Konsole

> **Hinweis**
>
> Das folgende Verfahren erstellt einen HealthLake Datenspeicher mit AWS SigV4-Autorisierung. Die HealthLake Konsole unterstützt nicht die Erstellung eines SMART-Datenspeichers auf FHIR.

Um einen HealthLake Datenspeicher mit AWS SigV4-Autorisierung zu erstellen

1. Melden Sie sich auf der Seite Datenspeicher erstellen in der HealthLake Konsole an.
2. Wählen Sie Datenspeicher erstellen.
3. Geben Sie im Bereich Datenspeicher-Einstellungen für den Namen des Datenspeichers einen Namen an.
4. (Optional) Aktivieren Sie im Bereich Datenspeicher-Einstellungen für Beispieldaten vorab laden das Kontrollkästchen, um Synthea-Daten vorab zu laden. Synthea-Daten sind ein Open-Source-Beispieldatensatz. Weitere Informationen finden Sie unter Synthea vorinstallierte Datentypen für HealthLake.
5. Wählen Sie im Abschnitt Datenspeicher-Verschlüsselung entweder AWS-eigenen Schlüssel verwenden (Standard) oder Anderen AWS-KMS-Schlüssel auswählen (erweitert).
6. Im Abschnitt Tags — optional können Sie Ihrem Datenspeicher Tags hinzufügen. Weitere Informationen zum Taggen Ihres Datenspeichers finden Sie unterHealthLake Datenspeicher kennzeichnen.

7. Wählen Sie Datenspeicher erstellen aus.

   Der Status Ihres Datenspeichers ist auf der Seite Datenspeicher verfügbar.

Eigenschaften des HealthLake Datenspeichers abrufen

Wird verwendet`DescribeFHIRDatastore`, um Eigenschaften für einen AWS HealthLake Datenspeicher abzurufen. Die folgenden Menüs enthalten eine Vorgehensweise für das AWS-Managementkonsole und Codebeispiele für AWS CLI und AWS SDKs. Weitere Informationen finden Sie unter `DescribeFHIRDatastore` in der AWS HealthLake -API-Referenz.

Um Eigenschaften für einen HealthLake Datenspeicher abzurufen

Wählen Sie ein Menü, das Ihren Zugriffspräferenzen für entspricht AWS HealthLake.

AWS CLI und SDKs

CLI

AWS CLI

So beschreiben Sie einen FHIR-Datenspeicher

Das folgende `describe-fhir-datastore` Beispiel zeigt, wie Sie die Eigenschaften eines Datenspeichers in finden AWS HealthLake.

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

Ausgabe:

```
{
    "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
        }
    }
}
```

- Einzelheiten zur API finden Sie unter Beschreibung FHIRDatastore in der AWS CLI Befehlsreferenz.

Python

SDK für 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
```

- Einzelheiten zur API finden Sie unter FHIRDatastoreDescribe in AWS SDK for Python (Boto3) API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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.
```

- Einzelheiten zur API finden Sie unter Describe FHIRDatastore in der API-Referenz zum AWS SDK für SAP ABAP.

> **Beispiel für die Verfügbarkeit**
>
> Sie können nicht finden, was Sie brauchen? Fordern Sie über den Link Feedback geben in der rechten Seitenleiste dieser Seite ein Codebeispiel an.

AWS Konsole

1. Melden Sie sich auf der Seite Datenspeicher in der HealthLake Konsole an.
2. Wählen Sie einen Datenspeicher aus.

   Die Seite mit den Datenspeicher-Details wird geöffnet und alle Eigenschaften des HealthLake Datenspeichers sind verfügbar.

HealthLake Datenspeicher auflisten

`ListFHIRDatastores`Dient zum Auflisten aller HealthLake Datenspeicher in einem Benutzerkonto, unabhängig vom Status des Datenspeichers. Die folgenden Menüs enthalten eine Vorgehensweise für das AWS-Managementkonsole und Codebeispiele für AWS CLI und AWS SDKs. Weitere Informationen finden Sie unter `ListFHIRDatastores` in der AWS HealthLake -API-Referenz.

Um alle HealthLake Datenspeicher aufzulisten

Wählen Sie ein Menü, das Ihren Zugriffspräferenzen für entspricht AWS HealthLake.

AWS CLI und SDKs

CLI

AWS CLI

So listen Sie FHIR-Datenspeicher auf

Das folgende `list-fhir-datastores` Beispiel zeigt, wie der Befehl verwendet wird und wie Benutzer Ergebnisse nach dem Status des Datenspeichers in filtern können AWS HealthLake.

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

Ausgabe:

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

- Einzelheiten zur API finden Sie unter Liste FHIRDatastores in der AWS CLI Befehlsreferenz.

Python

SDK für 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
```

- Einzelheiten zur API finden Sie unter Liste FHIRDatastores in der AWS API-Referenz zum SDK for Python (Boto3).

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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.
```

- Einzelheiten zur API finden Sie unter Liste FHIRDatastores in der API-Referenz zum AWS SDK für SAP ABAP.

> **Beispiel für die Verfügbarkeit**
>
> Sie können nicht finden, was Sie brauchen? Fordern Sie über den Link Feedback geben in der rechten Seitenleiste dieser Seite ein Codebeispiel an.

AWS Konsole

- Melden Sie sich auf der Seite Datenspeicher in der HealthLake Konsole an.

  Alle HealthLake Datenspeicher sind im Abschnitt Datenspeicher aufgeführt.

HealthLake Datenspeicher kennzeichnen

Sie können HealthLake Datenspeichern Metadaten in Form von Tags zuweisen. Jedes Tag ist eine Bezeichnung, die aus einem benutzerdefinierten Schlüssel und Wert besteht. Mithilfe von Tags können Sie Datenspeicher verwalten, identifizieren, organisieren, suchen und filtern.

> **Wichtig**
>
> Speichern Sie keine geschützten Gesundheitsinformationen (PHI), personenbezogenen Daten (PII) oder andere vertrauliche oder sensible Informationen in Tags. Tags sind nicht für private oder sensible Daten gedacht.

In den folgenden Themen wird beschrieben, wie HealthLake Tagging-Operationen mithilfe von AWS-Managementkonsole AWS CLI, und verwendet werden. AWS SDKs Weitere Informationen finden Sie im Handbuch unter AWS Ressourcen taggen.Allgemeine AWS-Referenz

Themen

- Einen Datenspeicher taggen HealthLake
- Tags für einen HealthLake Datenspeicher auflisten
- Entmarkierung eines Datenspeichers HealthLake

Einen Datenspeicher taggen HealthLake

Wird verwendet`TagResource`, um einen HealthLake Datenspeicher zu kennzeichnen. Die folgenden Menüs enthalten eine Vorgehensweise für das AWS-Managementkonsole und Codebeispiele für AWS CLI und AWS SDKs. Weitere Informationen finden Sie unter `TagResource` in der AWS HealthLake -API-Referenz.

Um einen HealthLake Datenspeicher zu taggen

Wählen Sie ein Menü, das Ihren Zugriffspräferenzen für entspricht AWS HealthLake.

AWS CLI und SDKs

CLI

AWS CLI

So fügen Sie ein Tag zum Datenspeicher hinzu

Im folgenden Beispiel für `tag-resource` wird gezeigt, wie Sie ein Tag zu einem Datenspeicher hinzufügen.

```
aws healthlake tag-resource \
```

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

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

.

- Einzelheiten zur API finden Sie TagResourcein der AWS CLI Befehlsreferenz.

Python

SDK für 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
```

- Einzelheiten zur API finden Sie TagResourcein AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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.
```

- Einzelheiten zur API finden Sie TagResourcein der API-Referenz zum AWS SDK für SAP ABAP.

> **Beispiel für die Verfügbarkeit**
>
> Sie können nicht finden, was Sie brauchen? Fordern Sie über den Link Feedback geben in der rechten Seitenleiste dieser Seite ein Codebeispiel an.

AWS Konsole

1. Melden Sie sich auf der Seite Datenspeicher in der HealthLake Konsole an.
2. Wählen Sie einen Datenspeicher aus.

   Die Seite mit den Details zum Datenspeicher wird geöffnet.
3. Wählen Sie im Abschnitt Tags die Option Tags verwalten aus.

   Die Seite „Tags verwalten“ wird geöffnet.
4. Wählen Sie Neues Tag hinzufügen aus.
5. Geben Sie einen Schlüssel und einen Wert ein (optional).
6. Wählen Sie Speichern.

Tags für einen HealthLake Datenspeicher auflisten

Wird verwendet`ListTagsForResource`, um Tags für einen HealthLake Datenspeicher aufzulisten. Die folgenden Menüs enthalten eine Vorgehensweise für das AWS-Managementkonsole und Codebeispiele für AWS CLI und AWS SDKs. Weitere Informationen finden Sie unter `ListTagsForResource` in der AWS HealthLake -API-Referenz.

Um Tags für einen HealthLake Datenspeicher aufzulisten

Wählen Sie ein Menü, das Ihren Zugriffspräferenzen für entspricht AWS HealthLake.

AWS CLI und SDKs

CLI

AWS CLI

So listen Sie Tags für einen Datenspeicher auf

Im folgenden Beispiel für `list-tags-for-resource` werden die Tags aufgelistet, die mit dem angegebenen Datenspeicher verknüpft sind:

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

Ausgabe:

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

- Einzelheiten zur API finden Sie ListTagsForResourcein der AWS CLI Befehlsreferenz.

Python

SDK für 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
```

- Einzelheiten zur API finden Sie ListTagsForResourcein AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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.
```

- Einzelheiten zur API finden Sie ListTagsForResourcein der API-Referenz zum AWS SDK für SAP ABAP.

> **Beispiel für die Verfügbarkeit**
>
> Sie können nicht finden, was Sie brauchen? Fordern Sie über den Link Feedback geben in der rechten Seitenleiste dieser Seite ein Codebeispiel an.

AWS Konsole

1. Melden Sie sich auf der Seite Datenspeicher in der HealthLake Konsole an.
2. Wählen Sie einen Datenspeicher aus.

   Die Seite mit den Details zum Datenspeicher wird geöffnet. Im Abschnitt Tags werden alle Datenspeicher-Tags aufgelistet.

Entmarkierung eines Datenspeichers HealthLake

Wird verwendet`UntagResource`, um ein Tag aus einem HealthLake Datenspeicher zu entfernen. Die folgenden Menüs enthalten eine Vorgehensweise für das AWS-Managementkonsole und Codebeispiele für AWS CLI und AWS SDKs. Weitere Informationen finden Sie unter `UntagResource` in der AWS HealthLake -API-Referenz.

Um die Markierung eines HealthLake Datenspeichers aufzuheben

Wählen Sie ein Menü, das Ihren Zugriffspräferenzen für entspricht AWS HealthLake.

AWS CLI und SDKs

CLI

AWS CLI

So entfernen Sie Tags aus einem Datenspeicher.

Das folgende Beispiel für `untag-resource` zeigt, wie Sie Tags aus einem Datenspeicher entfernen.

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

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie UntagResourcein der AWS CLI Befehlsreferenz.

Python

SDK für 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
```

- Einzelheiten zur API finden Sie UntagResourcein AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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.
```

- Einzelheiten zur API finden Sie UntagResourcein der API-Referenz zum AWS SDK für SAP ABAP.

> **Beispiel für die Verfügbarkeit**
>
> Sie können nicht finden, was Sie brauchen? Fordern Sie über den Link Feedback geben in der rechten Seitenleiste dieser Seite ein Codebeispiel an.

AWS Konsole

1. Melden Sie sich auf der Seite Datenspeicher in der HealthLake Konsole an.
2. Wählen Sie einen Datenspeicher aus.

   Die Seite mit den Details zum Datenspeicher wird geöffnet.
3. Wählen Sie im Abschnitt Tags die Option Tags verwalten aus.

   Die Seite „Tags verwalten“ wird geöffnet.
4. Wählen Sie neben dem Tag, das Sie entfernen möchten, die Option Entfernen aus.
5. Wählen Sie Speichern.

Löschen eines HealthLake Datenspeichers

Wird verwendet`DeleteFHIRDatastore`, um einen HealthLake Datenspeicher zu löschen. Die folgenden Menüs enthalten eine Vorgehensweise für das AWS-Managementkonsole und Codebeispiele für AWS CLI und AWS SDKs. Weitere Informationen finden Sie unter `DeleteFHIRDatastore` in der AWS HealthLake -API-Referenz.

Um einen HealthLake Datenspeicher zu löschen

Wählen Sie ein Menü, das Ihren Zugriffspräferenzen für entspricht AWS HealthLake.

AWS CLI und SDKs

CLI

AWS CLI

So löschen Sie einen FHIR-Datenspeicher

Das folgende `delete-fhir-datastore` Beispiel zeigt, wie Sie einen Datenspeicher und seinen gesamten Inhalt löschen AWS HealthLake.

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

Ausgabe:

```
{
    "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)"
}
```

- Einzelheiten zur FHIRDatastoreAWS CLI API finden Sie unter Löschen in der Befehlsreferenz.

Python

SDK für Python (Boto3)

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

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


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

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

```
            )
            raise
```

- API-Einzelheiten finden Sie unter Delete FHIRDatastore in AWS SDK for Python (Boto3) API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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.
```

- Einzelheiten zur API finden Sie unter Delete FHIRDatastore in der API-Referenz zum AWS SDK für SAP ABAP.

> **Beispiel für die Verfügbarkeit**
>
> Sie können nicht finden, was Sie brauchen? Fordern Sie über den Link Feedback geben in der rechten Seitenleiste dieser Seite ein Codebeispiel an.

AWS Konsole

1. Melden Sie sich auf der Seite Datenspeicher in der HealthLake Konsole an.
2. Wählen Sie einen Datenspeicher aus.

   Die Seite mit den Details zum Datenspeicher wird geöffnet.
3. Wählen Sie Löschen aus.

   Die Seite Datenspeicher löschen wird geöffnet.
4. Um das Löschen des Datenspeichers zu bestätigen, geben Sie den Namen des Datenspeichers in das Texteingabefeld ein.
5. Wählen Sie Löschen aus.

Verwaltung von FHIR-Abonnements in AWS HealthLake

AWS HealthLake unterstützt FHIR-Abonnements, sodass Sie in Echtzeit Benachrichtigungen erhalten können, wenn sich bestimmte Gesundheitsdaten ändern. Diese Funktion implementiert das themenbasierte Abonnementmodell von FHIR R5 Backport und bietet eine verbesserte Skalierbarkeit und Flexibilität gegenüber dem herkömmlichen FHIR R4-Abonnementmodell.

Mit FHIR-Abonnements können Sie ereignisgesteuerte Anwendungen für das Gesundheitswesen entwickeln, die sofort auf Änderungen der klinischen Daten reagieren und so zeitnahe Interventionen, automatisierte Arbeitsabläufe und eine verbesserte Pflegekoordination ermöglichen.

Themen

- So funktionieren FHIR-Abonnements
- Zentrale Komponenten
- Best Practices
- Der FHIR-Abonnement-Lebenszyklus mit AWS HealthLake
- Erstellen eines FHIR-Abonnements mit AWS HealthLake
- Suche nach FHIR-Abonnements mit AWS HealthLake
- Benachrichtigungen filtern mit AWS HealthLake

So funktionieren FHIR-Abonnements

FHIR-Abonnements basieren auf einem themenbasierten Modell, bei dem: HealthLake

1. Erstellen Sie Themen, um Ereignisse zu definieren: Erstellen Sie Abonnementthemen, die Ereignisse angeben, die Benachrichtigungen auslösen können
2. Sie abonnieren: Erstellen Sie Abonnements für diese Themen mit bestimmten Filterkriterien
3. HealthLake überwacht: Der Dienst sucht kontinuierlich nach Ereignissen, die Ihren Kriterien entsprechen
4. Zugestellte Benachrichtigungen: Es treten CWhen übereinstimmende Ereignisse auf. Benachrichtigungen HealthLake werden über den von Ihnen ausgewählten Kanal zugestellt

Zentrale Komponenten

FHIR-Abonnements bestehen aus den folgenden Komponenten.

Abonnement-Themen

Abonnementthemen bilden die Grundlage des Benachrichtigungssystems und definieren:

- Ereignisse auslösen: Welche Änderungen lösen Benachrichtigungen aus (zum Beispiel: Erstellung, Aktualisierung, Löschung von Ressourcen)
- Verfügbare Filter: Welche Filteroptionen stehen Abonnenten zur Verfügung
- Inhalt der Benachrichtigungen: Welche Daten sind in Benachrichtigungen enthalten

In der folgenden Tabelle sind die gängigen Thementypen aufgeführt.

Ereignistyp	Description	Häufige Anwendungsfälle
Erstellung von Ressourcen	Wird ausgelöst, wenn Ressourcen erstellt werden	Registrierung neuer Patienten , Aufzeichnung neuer Beobachtungen
Ressourcenaktualisierungen	Wird ausgelöst, wenn Ressourcen geändert werden	Statusänderungen, klinische Updates
Löschen von Ressourcen	Wird ausgelöst, wenn Ressourcen gelöscht werden	Prüfung und Nachverfolgung der Einhaltung von Vorschrif ten

Subscriptions (Abonnements)

Ein Abonnement ist Ihre Anfrage, Benachrichtigungen für bestimmte Ereignisse zu erhalten, die in einem Abonnementthema definiert sind. Jedes Abonnement beinhaltet:

- Themenreferenz: Gibt an, welches Abonnement-Thema Sie abonnieren

- Filter: Kriterien zur Auswahl, bei welchen Ereignissen Benachrichtigungen generiert werden

- Kanalkonfiguration: Wo und wie sollen Benachrichtigungen zugestellt werden
- Payload-Präferenzen: Welcher Detaillierungsgrad sollte in Benachrichtigungen enthalten sein

Benachrichtigungskanäle

HealthLake unterstützt die folgenden Benachrichtigungskanäle:

Kanaltyp	Anwendungsfälle	
EventBridge	Unternehmensintegrationen, serverlose Workflows, serviceübergreifende Orchestrierung AWS	
REST Hook	Direkte Endpunktbenachrichtigungen, Systemintegration von Drittanbietern	

Payloads für Benachrichtigungen

Wählen Sie je nach Bedarf den passenden Payload-Typ aus:

Typ der Nutzlast	Description	Sicherheitsüberlegungen
Nur Ausweis	Enthält nur Ressourcen-Identifikatoren	Minimale PHI-Exposition
Vollständige Ressource	Enthält den vollständigen Ressourceninhalt mit einer maximalen Größe von 256 KB. Wenn die Größe größer als 256 KB ist, wird sie wieder auf „Nur ID“ umgestellt	Enthält PHI; Überprüfen Sie die sichere Handhabung

Best Practices

Leistungsoptimierung

- Verwenden Sie gezielte Filter: Grenzen Sie Ihre Kriterien ein, um nur wichtige Benachrichtigungen zu erhalten
- Wählen Sie geeignete Payload-Typen: Verwenden Sie nach Möglichkeit Payloads, die nur IDs enthalten, um eine bessere Leistung zu erzielen
- Implementieren Sie effiziente Empfänger: Stellen Sie sicher, dass die Benachrichtigungsempfänger Nachrichten schnell verarbeiten

Sicherheitsüberlegungen

- Sichere Endpunkte: Implementieren Sie die richtige Authentifizierung für REST-Hook-Endpunkte
- PHI-Schutz: Seien Sie vorsichtig mit Payloads für vollständige Ressourcen, da diese PHI enthalten
- Zugriffskontrolle: Beschränken Sie die Abonnementerstellung nur auf autorisierte Benutzer

Operative Exzellenz

- Legen Sie die entsprechenden Enddaten fest: Verwenden Sie Enddaten für temporäre Abonnements
- Überwachen Sie den Abonnementstatus: Überprüfen Sie regelmäßig den Status Ihrer Abonnements
- Implementieren Sie die Fehlerbehandlung: Gestalten Sie Ihre Anwendungen so, dass sie Fehler bei der Zustellung von Benachrichtigungen bewältigen

Der FHIR-Abonnement-Lebenszyklus mit AWS HealthLake

Folgen Sie diesen Schritten, um den Lebenszyklus eines FHIR-Abonnements zu verstehen:

1. Erstellen eines `SubscriptionTopic`
   - Erstellen Sie ein `SubscriptionTopic` mit Status `"unknown"`
2. Erstellen eines `Subscription`
   - Erstellen Sie eine `Subscription` mit Status `"requested"`

- HealthLake validiert die Konfiguration `Subscription`
- `Subscription`muss auf ein bereits vorhandenes Thema verweisen (das Thema muss den Status habenunknown,`draft`,`active`).

3. Aktivierung
   - Falls gültig, HealthLake aktualisiert der Status von `Subscription` auf `"active"`
   - Wenn das angegebene Thema beim Erstellen eines `Subscription` im Status war`"unknown"`, wird der Status auf den Status HealthLake aktualisiert, `"active"` sobald das Abonnement ebenfalls aktiv ist
   - Die erfolgreiche Erstellung von Abonnements dauert in der Regel 5-10 Minuten
   - Wenn die Erstellung `Subscription` nicht erfolgreich ist, ändert sich der Status dahin, `error` wo Sie einen DELETE-Vorgang ausführen und dann erneut versuchen sollten, ein Abonnement zu erstellen. In dem `"error"` Feld in der Abonnementressource können Sie nachlesen, warum das Abonnement nicht erfolgreich erstellt wurde.
4. Aufnahme während des Abonnements `active`
5. Während `Subscription` ist `active`
   - HealthLake überwacht Ereignisse, die Ihren Kriterien entsprechen
   - Bei Übereinstimmungen werden Benachrichtigungen an den konfigurierten Endpunkt gesendet
6. Fehlerbehandlung
   - HealthLake versucht es 14 Tage lang erneut und hört dann bei diesen Ereignissen auf, es erneut zu versuchen
7. Deaktivierung
   - A `Subscription` kann deaktiviert werden durch:

     Einstellung eines Enddatums (automatische Deaktivierung)

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

Die Ressource wird `Subscription` gelöscht

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

Erstellen eines FHIR-Abonnements mit AWS HealthLake

Die folgende Anleitung zeigt Ihnen, wie Sie ein FHIR-Abonnement mit erstellen. AWS HealthLake

Um ein FHIR-Abonnement zu erstellen

1. Erstellen Sie einen `SubscriptionTopic`.

   Beispiel für eine Ressource zum Thema Abonnement:

```
{
  "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. Bereiten Sie Ihren Benachrichtigungsendpunkt vor (benutzerdefinierter Kanal). Die folgenden Schritte sind erforderlich, um sicherzustellen, dass der Endpunkt Benachrichtigungen erhält

   Bei Verwendung von REST Hook

   - Vertrauen `events.amazonaws.com` Sie Ihrer KMS-Schlüsselrichtlinie, wenn Sie den CM_CMK-Datenspeicher verwenden.
   - Wenn Sie einen CM_CMK-Datenspeicher verwenden, müssen Sie Ihrem KMS-Schlüssel das Tag mit dem Wert von hinzufügen `EventBridgeApiDestinations` `true`
   - HealthLake verwendet, OAuth um Ihren REST-Hook-Endpunkt zu authentifizieren. Daher müssen Sie beim Erstellen eines REST-Hook-Abonnements eine Client-ID, ein Client-Secret und den Channel angeben. oAuth-endpoint-url _type.extension [*].

   Beispiel für eine KMS-Schlüsselrichtlinie bei Verwendung des CM_CMK-Datenspeichers:

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

Bei der Verwendung EventBridge

- Vertrauen `events.amazonaws.com` Sie Ihrer KMS-Schlüsselrichtlinie, wenn Sie den CM_CMK-Datenspeicher verwenden.
- Stellen Sie sicher, dass Ihre EventBridge Ressourcenrichtlinie dem Service Principal vertraut. `healthlake.amazonaws.com`
- Wenn Sie CM_CMK und EventBridge als Endpunkt verwenden, stellen Sie sicher, dass Sie Ihren EventBridge Bus mit demselben KMS-Schlüssel wie den KMS-Schlüssel für den Datenspeicher verschlüsseln.
- Stellen Sie sicher, dass Ihr EventBridge Bus mindestens eine Regel hat, die den Ereignissen entspricht, die von generiert wurden. HealthLake

Beispiel für eine Ressourcenrichtlinie für den EventBridge Kanalbus:

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

Beispiel für ein EventBridge Regelereignismuster zum Empfangen von Ereignissen von: HealthLake

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

> **Note**
>
> HealthLake unterstützt 2 Quellen:
>
> - `“healthlake”`: Nur für Abonnements.
> - `“aws.healthlake”`: Um HealthLake Serviceereignisse zu empfangen.
>
> Wird `“healthlake”` als Quelle verwendet, wenn Sie eine Regel für FHIR Subscriptions-Event-Busse erstellen.

3. Erstellen Sie Ihr `Subscription`

   Reichen Sie eine Abonnement-Ressource ein mit:

   - Status: `"requested"`
   - Verweis auf die von Ihnen gewählte `SubscriptionTopic` ID
   - Kriterien filtern. Weitere Informationen finden Sie unter Filtern von Benachrichtigungen für unterstützte Filter.
   - Kanalkonfiguration

Beispiele für Abonnement-Payloads

Die folgenden Codebeispiele zeigen, wie Abonnement-Payloads erstellt werden.

EventBridge

Abonnement mit `event-bridge` Kanal und **`id-only`** Payload-Typ.

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

Abonnement mit `event-bridge` Endpunkt und `full-resource` Payload-Typ.

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

Hook testen

Abonnement mit `rest-hook` Endpunkt und `id-only` Payload-Typ.

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

Abonnement mit `rest-hook` Kanal und **`full-resource`** Payload-Typ.

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

Beispiele für Payloads für Benachrichtigungen

Während ein Abonnement erstellt wird, HealthLake überprüft es, ob das Abonnement erfolgreich eingerichtet wurde, indem es ein Handshake-Paket an Ihren konfigurierten Kanal sendet. Die folgende Payload ist ein Beispiel für ein Handshake-Paket.

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

Beispiel für ein Benachrichtigungspaket nur mit 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"
              }
            ]
          }
        }
      ]
    }
  }
}
```

Beispiel für ein Benachrichtigungspaket mit allen Ressourcen.

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

Versionierung von Ereignissen

HealthLake unterstützt standardmäßig den FHIR-Verlauf.

Um zu erfahren, welche Version der Ressource Sie in Ihrem Benachrichtigungspaket erhalten haben:

- Vollständige Ressource: Da vollständige Ressourcenpakete die gesamte Ressource enthalten, wird die Version `entry[*]` für jede Ressource, die sich im Paket befindet, in der enthalten ist, enthalten.
- id-only: Bundles enthalten keine Ressourceninformationen. HealthLakeenthält die Version, die über das Feld abgeglichen und in das Paket aufgenommen wurde. `entry[0].notificationEvent[*].additionalContext` Dieses Feld hat das Format`<ResourceType>/<ResourceId>/_history/<Version Id>`. Weitere Informationen finden Sie im Feld AdditionalContext im Beispiel einer Payload, die nur ID enthält.

Erkennung von Duplikaten bei Ereignissen

HealthLakeDie FHIR-Abonnementfunktion garantiert mindestens eine Lieferung. Das bedeutet, dass Sie dieselbe Veranstaltung möglicherweise mehrmals erhalten, entweder im selben Paket oder in einem anderen Paket. Um Duplikate zu identifizieren, wird für jedes Ereignis im Benachrichtigungs-Paket eine eindeutige ID HealthLake bereitgestellt. `entry[0].notificationEvent[*].id`

Diese ID ist für die spezifische Version des Ereignisses, die zugeordnet und zugestellt wurde, eindeutig. Wenn beispielsweise dieselbe Begegnung zweimal aktualisiert wird und beide Aktualisierungen den Filterkriterien entsprechen, erhalten Sie zwei separate Ereignisse mit derselben Encounter-Referenz. Sie werden dasselben`notificationEvent[*].focus`, aber eine einzigartige haben`notificationEvent[*].id`. Darüber hinaus können diese Ereignisse in separaten Paketen oder innerhalb desselben Benachrichtigungspakets gesendet werden.

Suche nach FHIR-Abonnements mit AWS HealthLake

`Subscription`und `SubscriptionTopic` Ressourcen können ebenfalls durchsucht werden. HealthLake unterstützt alle gängigen Suchparameter für Abonnements und SubscriptionTopic Ressourcen.

Darüber hinaus unterstützen wir zusätzliche Suchfunktionen über die folgenden Parameter:

Abonnement

Suchparameter	Description	Beispiel
contact	Suchen Sie im Feld subscription.contact in der R4-Core-Spezifikation	`Subscription?contact=phone`
criteria	Suchen Sie im Feld Subscription.Criteria in der R4-Core-Spezifikation	`Subscription?criteria=[baseUrl]/datastore/[datastoreId]/r4/SubscriptionTopic/[topicId]`
Nutzlast	Suchen Sie im Feld Subscription.Channel.Payload in der R4-Core-Spezifikation	`Subscription?payload=application/fhir+json`
Status	Suchen Sie im Feld Subscription.Status in der R4-Core-Spezifikation	`Subscription?status=error`
type	Suchen Sie im Feld Subscription.Channel.Type in der R4-Core-Spezifikation	`Subscription?topic=event-bridge`
URL	Suchen Sie nach dem Feld Subscription.Channel.Endpoint in der R4-Core-Spezifikation	`Subscription?url=[Subscription.channel.endpoint]`

Suchparameter	Description	Beispiel
Filterkriterien	Suchen Sie im Feld für die Kriterienerweiterung mit der URL "http://hl7“. org/fhir/uv/subscriptions-backport/StructureDefinition/backport-filter-criteria“ als Zeichenfolge	`Subscription?filter-criteria=Encounter?`
benutzerdefinierter Kanal	Suchen Sie im Erweiterungsfeld für den benutzerdefinierten Kanaltyp mit der URL "http://hl7. org/fhir/uv/subscriptions-backport/StructureDefinition/backport-channel-type“ als Codierung Beispiel für eine Abonnement-Payload	`Subscription?custom-channel=[System]|[code]`
Payload-Typ	Suchen Sie im Erweiterungsfeld für den Payload-Typ, in dem Sie angeben, wie die Payload formatiert ist (entweder oder) `id-only` `full-resource`	`Subscription?payload-type=full-resource`
Thema	Suchen Sie im Feld subscription.criteria, dem das Thema hinzugefügt wurde	`Subscription?topic=[topicId]`

SubscriptionTopic

Suchparameter	Description	Beispiel
date	Suchen Sie im Datumsfeld in der SubscriptionTopic Ressource	`SubscriptionTopic?date=[SubscriptionTopic.date]`
derived-or-self	Suchen Sie entweder nach den `derivedFrom` Feldern `url` oder in der SubscriptionTopic Ressource	`SubscriptionTopic?derived-or-self=[SubscriptionTopic.url | SubscriptionTopic.derivedFrom]`
Bezeichner	Suchen Sie im Feld SubscriptionTopic .identifier	`SubscriptionTopic?identifier=[SubscriptionTopic.identifier]`
Ressource	Suchen Sie im Feld SubscriptionTopic .resourceTrigger.Resource	`SubscriptionTopic?resource=Encounter`
Status	Suchen Sie nach dem Status des SubscriptionTopic	`SubscriptionTopic?status=unknown`
Titel	Suchen Sie im Titel nach einem SubscriptionTopic	`SubscriptionTopic?title=admission`
Beschreibung des Triggers	Suchen Sie auf .resourceTrigger.Description SubscriptionTopic	`SubscriptionTopic.trigger-description=resource moving to state 'in-progress'`
URL	Suchen Sie auf der URL nach SubscriptionTopic	`SubscriptionTopic?url=[SubscriptionTopic.url]`

Suchparameter	Description	Beispiel
version	Suchen Sie nach der Version für SubscriptionTopic	`SubscriptionTopic?version=1`

Benachrichtigungen filtern mit AWS HealthLake

Verfeinern Sie Ihre Benachrichtigungen mithilfe der standardmäßigen FHIR-Suchparameter in Ihren `Subscription` Kriterien.

Unterstützte Filter

Die folgende Tabelle zeigt eine Liste der unterstützten Filterkriterien, die Abonnements HealthLake unterstützen.

Typen von Suchparametern	FHIR-Datentyp	Modifikatoren	Unterstützte Präfixe
Zeichenfolge	Zeichenfolge HumanName Adresse	exakt, enthält fehlende	
Token	boolesch Code Zeichenfolge Codierung CodeableConcept Kennung ContactPoint id	nicht, Text, fehlt	

Typen von Suchparametern	FHIR-Datentyp	Modifikatoren	Unterstützte Präfixe
Zahl	Ganzzahl Dezimalwert Positive Tinte Int ohne Vorzeichen	fehlt	„eq“, „ne“, „gt“, „lt“, „ge“, „le“, „sa“, „eb“, „ap“
Date	date dateTime Instant- Zeitraum Timing		„eq“, „ne“, „gt“, „lt“, „ge“, „le“, „sa“, „eb“, „ap“
Quantity (Menge)	Quantity (Menge) Geld Bereich SimpleQuantity		„eq“, „ne“, „gt“, „lt“, „ge“, „le“, „sa“, „eb“, „ap“
Referenz	Referenz	fehlt, Kennung, Typ	
URI	uri URL kanonisch uid oid	fehlt	

Beispiel unterstützter Filter

Die folgende Tabelle enthält Beispiele für unterstützte Filterkriterien, die HealthLake für Abonnements gelten:

Zweck	Filterkriterien	Description
Patientenspezifische Beobachtungen	`Observation?patient=Patient/[id]&status=final`	Lassen Sie sich benachrichtigen, wenn die Beobachtungen für einen bestimmten Patienten abgeschlossen sind
Patientenspezifische Beobachtungen	`Patient?birthdate=gt2021`	Erhalten Sie Benachrichtigungen, wenn Patienten, die nach 2021 geboren wurden, registriert oder aktualisiert werden
Patientenspezifische Beobachtungen	`Condition?code=http://snomed.info/sct|39065001`	Erhalten Sie Benachrichtigungen über Erkrankungen mit bestimmten SNOMED-Codes
Patientenspezifische Beobachtungen	`Observation?code=http://loinc.org|8480-6&value-quantity=gt160`	Erhalten Sie Benachrichtigungen über hohe systolische Blutdruckwerte

Import von FHIR-Daten mit AWS HealthLake

Nach dem Erstellen eines HealthLake Datenspeichers besteht der nächste Schritt darin, Dateien aus einem Amazon Simple Storage Service (S3) -Bucket zu importieren. Sie können einen FHIR-Importauftrag mit dem AWS-Managementkonsole, AWS CLI, oder AWS SDKs starten. Verwenden Sie native AWS HealthLake Aktionen, um FHIR-Importaufträge zu starten, zu beschreiben und aufzulisten.

> **Wichtig**
>
> HealthLake unterstützt die FHIR R4-Spezifikation für den Austausch von Gesundheitsdaten. Bei Bedarf können Sie mit einem AWS HealthLake Partner zusammenarbeiten, um Ihre Gesundheitsdaten vor dem Import in das FHIR R4-Format zu konvertieren.

Wenn Sie einen FHIR-Importauftrag starten, geben Sie einen Amazon S3 S3-Bucket-Eingabespeicherort, einen Amazon S3 S3-Bucket-Ausgabespeicherort (für Ergebnisse der Auftragsverarbeitung), eine IAM-Rolle, die HealthLake Zugriff auf Ihre Amazon S3 S3-Buckets gewährt, und einen kundeneigenen Schlüssel an. AWS AWS Key Management Service Weitere Informationen finden Sie unter Berechtigungen für Importaufträge einrichten.

> **Note**
>
> Sie können Importaufträge in eine Warteschlange stellen. Die asynchronen Importaufträge werden nach dem FIFO-Verfahren (First In First Out) verarbeitet. Sie können Jobs auf die gleiche Weise in die Warteschlange stellen, wie Sie Importaufträge starten. Wenn gerade einer in Bearbeitung ist, wird er einfach in die Warteschlange gestellt. Sie können FHIR-Ressourcen erstellen, lesen, aktualisieren oder löschen, während ein Importauftrag ausgeführt wird.

HealthLake generiert eine `manifest.json` Datei für jeden FHIR-Importauftrag. Die Datei beschreibt sowohl die Erfolge als auch die Misserfolge eines FHIR-Importjobs. HealthLake gibt die `manifest.json` Datei in den Amazon S3 S3-Bucket aus, der beim Starten eines FHIR-Importauftrags angegeben wurde. Protokolldateien sind in zwei Ordnern mit dem Namen `SUCCESS` und `FAILURE` organisiert. Verwenden Sie die `manifest.json` Datei als ersten Schritt bei der Behebung eines fehlgeschlagenen Importauftrags, da sie Einzelheiten zu jeder Datei enthält.

```
{
    "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
}
```


Konfiguration der Validierungsebene für Importe

Wenn Sie einen FHIR-Importjob starten, können Sie optional eine angeben, `ValidationLevel` die auf jede Ressource angewendet werden soll. AWS HealthLake unterstützt derzeit die folgenden Validierungsstufen:

- `strict`: Ressourcen werden anhand des Profilelements der Ressource oder der R4-Spezifikation, falls kein Profil vorhanden ist, validiert. Dies ist die Standardvalidierungsebene für AWS HealthLake.
- `structure-only`: Ressourcen werden anhand von R4 validiert, wobei alle referenzierten Profile ignoriert werden.

- `minimal`: Ressourcen werden minimal validiert, wobei bestimmte R4-Regeln ignoriert werden. Ressourcen, die die erforderlichen Strukturprüfungen nicht bestehen, search/analytics werden aktualisiert und enthalten nun eine Warnung zur Prüfung.

Beim Import mit der `minimal` Validierungsebene können zusätzliche Protokolldateien in einem Ordner mit dem Namen generiert werden`SUCCESS_WITH_SEARCH_VALIDATION_FAILURES`. Ressourcen in den Protokolldateien dieses Ordners wurden in Ihren Datenspeicher aufgenommen, obwohl die suchbezogenen Validierungsprüfungen nicht bestanden haben. Dies bedeutet, dass bestimmte Aspekte Ihrer FHIR-Ressource laut FHIR ungültig waren und falsch formatierte Felder möglicherweise nicht durchsuchbar sind. An diese Ressourcen wird ein `extension` Anhang angehängt, in dem der Fehler beschrieben wird.

Themen

- Einen FHIR-Importjob starten
- Eigenschaften von FHIR-Importaufträgen abrufen
- Auflisten von FHIR-Importaufträgen

Einen FHIR-Importjob starten

Wird verwendet`StartFHIRImportJob`, um einen FHIR-Importauftrag in einen HealthLake Datenspeicher zu starten. Die folgenden Menüs enthalten eine Vorgehensweise für das AWS-Managementkonsole und Codebeispiele für AWS CLI und AWS SDKs. Weitere Informationen finden Sie unter `StartFHIRImportJob` in der AWS HealthLake -API-Referenz.

> **Wichtig**
>
> HealthLake unterstützt die FHIR R4-Spezifikation für den Austausch von Gesundheitsdaten. Bei Bedarf können Sie mit einem AWS HealthLake Partner zusammenarbeiten, um Ihre Gesundheitsdaten vor dem Import in das FHIR R4-Format zu konvertieren.

So starten Sie einen FHIR-Importjob

Wählen Sie ein Menü, das Ihren Zugriffspräferenzen für entspricht. AWS HealthLake

AWS CLI und SDKs

CLI

AWS CLI

So starten Sie einen FHIR-Importjob

Das folgende `start-fhir-import-job` Beispiel zeigt, wie Sie einen FHIR-Importauftrag mit AWS HealthLake starten.

```
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)"
```

Ausgabe:

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

- API-Einzelheiten finden Sie unter FHIRImportJob starten in der AWS CLI Befehlsreferenz.

Python

SDK für 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
```

- Einzelheiten zur API finden Sie unter Start FHIRImport Job in AWS SDK for Python (Boto3) API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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_hls3configuration(
              iv_s3uri = iv_job_output_s3_uri
              iv_kmskeyid = iv_kms_key_id
            )
          )
          iv_dataaccessrolearn = iv_data_access_role_arn
          iv_datastoreid = iv_datastore_id
```

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

- API-Details finden Sie unter Referenz zum Starten FHIRImport eines Job im AWS SDK für SAP ABAP API.

> **Beispiel für die Verfügbarkeit**
>
> Sie können nicht finden, was Sie brauchen? Fordern Sie über den Link Feedback geben in der rechten Seitenleiste dieser Seite ein Codebeispiel an.

AWS Konsole

1. Melden Sie sich auf der Seite Datenspeicher in der HealthLake Konsole an.
2. Wählen Sie einen Datenspeicher aus.
3. Wählen Sie Importieren aus.

   Die Importseite wird geöffnet.
4. Geben Sie im Bereich Eingabedaten die folgenden Informationen ein:

   - Speicherort der Eingabedaten in Amazon S3

5. Geben Sie im Abschnitt Ausgabedateien importieren die folgenden Informationen ein:

   - Speicherort der Importausgabedateien in Amazon S3
   - Verschlüsselung der Importausgabedateien

6. Wählen Sie im Abschnitt Zugriffsberechtigungen die Option Bestehende IAM-Dienstrolle verwenden und wählen Sie die Rolle im Menü Servicerollenname aus, oder wählen Sie IAM-Rolle erstellen aus.
7. Wählen Sie Daten importieren.

> **Note**
>
> Wählen Sie beim Import im Banner oben auf der Seite die Option Job-ID kopieren aus. Sie können die verwenden `JobID`, um die Eigenschaften des Importauftrags anzufordern, indem Sie die verwenden AWS CLI. Weitere Informationen finden Sie unter Eigenschaften von FHIR-Importaufträgen abrufen.

Eigenschaften von FHIR-Importaufträgen abrufen

Wird verwendet`DescribeFHIRImportJob`, um die Eigenschaften von FHIR-Importaufträgen abzurufen. Die folgenden Menüs enthalten eine Vorgehensweise für das AWS-Managementkonsole und Codebeispiele für AWS CLI und AWS SDKs. Weitere Informationen finden Sie unter `DescribeFHIRImportJob` in der AWS HealthLake -API-Referenz.

Um die Eigenschaften von FHIR-Importaufträgen abzurufen

Wählen Sie ein Menü, das Ihren Zugriffspräferenzen für entspricht AWS HealthLake.

AWS CLI und SDKs

CLI

AWS CLI

So beschreiben Sie einen FHIR-Importjob

Das folgende `describe-fhir-import-job` Beispiel zeigt, wie Sie die Eigenschaften eines FHIR-Importjobs mithilfe AWS HealthLake von lernen können.

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

Ausgabe:

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

- Einzelheiten zur API finden Sie unter FHIRImportJob beschreiben in der AWS CLI Befehlsreferenz.

Python

SDK für 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
```

- Einzelheiten zur API finden Sie unter FHIRImportDescribe Job in AWS SDK for Python (Boto3) API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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.
```

- Einzelheiten zur API finden Sie unter FHIRImportDescribe Job in AWS SDK for SAP ABAP API-Referenz.

> **Beispiel für die Verfügbarkeit**
>
> Sie können nicht finden, was Sie brauchen? Fordern Sie über den Link Feedback geben in der rechten Seitenleiste dieser Seite ein Codebeispiel an.

AWS Konsole

> **Note**
>
> Informationen zum FHIR-Importauftrag sind in der HealthLake Konsole nicht verfügbar. Verwenden Sie stattdessen die Option AWS CLI with, `DescribeFHIRImportJob` um

Eigenschaften von Importaufträgen anzufordern, wie JobStatusz. Weitere Informationen finden Sie im AWS CLI Beispiel auf dieser Seite.

Auflisten von FHIR-Importaufträgen

Dient `ListFHIRImportJobs` zum Auflisten von FHIR-Importaufträgen für einen aktiven HealthLake Datenspeicher. Die folgenden Menüs enthalten eine Vorgehensweise für das AWS-Managementkonsole und Codebeispiele für AWS CLI und AWS SDKs. Weitere Informationen finden Sie unter `ListFHIRImportJobs` in der AWS HealthLake -API-Referenz.

Um FHIR-Importaufträge aufzulisten

Wählen Sie ein Menü, das Ihren Zugriffspräferenzen für entspricht AWS HealthLake.

AWS CLI und SDKs

CLI

AWS CLI

So listen Sie alle FHIR-Importjobs auf

Das folgende Beispiel für `list-fhir-import-jobs` zeigt, wie Sie mit dem Befehl eine Liste aller Importjobs anzeigen, die mit einem Konto verknüpft sind.

```
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)
```

Ausgabe:

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

- Einzelheiten zur API finden Sie unter FHIRImportJobs in der AWS CLI Befehlsreferenz auflisten.

Python

SDK für 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
```

- Einzelheiten zur API finden Sie unter List FHIRImport Jobs in AWS SDK for Python (Boto3) API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.
```

- Einzelheiten zur API finden Sie in der Referenz zum FHIRImportAuflisten von Aufträgen im AWS SDK für SAP ABAP API.

Beispiel für die Verfügbarkeit

Sie können nicht finden, was Sie brauchen? Fordern Sie über den Link Feedback geben in der rechten Seitenleiste dieser Seite ein Codebeispiel an.

AWS Konsole

Note

Informationen zum FHIR-Importauftrag sind in der HealthLake Konsole nicht verfügbar. Verwenden Sie stattdessen das „ AWS CLI with“, `ListFHIRImportJobs` um alle FHIR-Importaufträge aufzulisten. Weitere Informationen finden Sie im AWS CLI Beispiel auf dieser Seite.

Verwaltung von FHIR-Ressourcen in AWS HealthLake

Verwenden Sie FHIR RESTful R4-API-Interaktionen, um FHIR-Ressourcen in einem Datenspeicher zu verwalten. HealthLake In den folgenden Abschnitten HealthLake werden alle FHIR RESTful R4-API-Interaktionen beschrieben, die für die FHIR-Ressourcenverwaltung verfügbar sind. Informationen zu den Funktionen des HealthLake Datenspeichers und zu den unterstützten Teilen der FHIR-Spezifikation finden Sie unter. FHIR R4-Fähigkeitserklärung für AWS HealthLake

> **Note**
>
> Die in diesem Kapitel aufgeführten FHIR-Interaktionen entsprechen dem HL7 FHIR R4-Standard für den Austausch von Gesundheitsdaten. Da es sich um Repräsentationen von HL7 FHIR-Diensten handelt, werden sie nicht über und angeboten. AWS CLI AWS SDKs Weitere Informationen finden Sie unter RESTful API in der FHIR RESTful R4-API-Dokumentation.

In der folgenden Tabelle sind FHIR R4-Interaktionen aufgeführt, die von unterstützt werden. AWS HealthLake Informationen zu den von unterstützten FHIR-Ressourcentypen finden Sie unter HealthLake. Ressourcentypen

FHIR R4-Interaktionen werden unterstützt von AWS HealthLake

Interaktionen	Description
Interaktionen mit dem gesamten System	
`capabilities`	Holen Sie sich eine Leistungsbeschreibung für das System. Siehe FHIR R4-Fähigkeitserklärung für AWS HealthLake.
`batch`	Aktualisieren, erstellen oder löschen Sie eine Reihe von Ressourcen in einer einzigen Interaktion. Siehe Bündelung der FHIR-Ressourcen.
Interaktionen auf Typebene	
`create`	Erstellen Sie eine neue Ressource mit einer vom Server zugewiesenen ID. Siehe Eine FHIR-Ressource erstellen.

Interaktionen	Description
`search`	Suchen Sie anhand einiger Filterkriterien nach einem Ressourcentyp. Siehe FHIR-Ressourcen werden durchsucht.
`history`	Rufen Sie den Änderungsverlauf für einen bestimmten Ressourcentyp ab. Siehe Lesen der FHIR-Ressourcengeschichte.
Interaktionen auf Instanzebene	
`read`	Lesen Sie den aktuellen Status einer Ressource. Siehe Eine FHIR-Ressource lesen.
`history`	Lesen Sie den Änderungsverlauf für eine bestimmte Ressource. Siehe Lesen der FHIR-Ressourcengeschichte.
`vread`	Lesen Sie den Status einer bestimmten Version der Ressource. Siehe Lesen des versionsspezifischen FHIR-Ressourcenverlaufs.
`update`	Aktualisieren Sie eine Ressource anhand ihrer ID (oder erstellen Sie sie, wenn sie neu ist). Siehe Aktualisierung einer FHIR-Ressource.
`delete`	Löscht eine Ressource. Siehe Löschen einer FHIR-Ressource.

Themen

- Eine FHIR-Ressource erstellen
- Eine FHIR-Ressource lesen
- Lesen der FHIR-Ressourcengeschichte
- Aktualisierung einer FHIR-Ressource
- Ändern von Ressourcen mit PATCH-Operation
- Bündelung der FHIR-Ressourcen
- Löschen einer FHIR-Ressource
- Idempotenz und Parallelität

Eine FHIR-Ressource erstellen

Die `create` FHIR-Interaktion erstellt eine neue FHIR-Ressource in einem HealthLake Datenspeicher. Weitere Informationen finden Sie `create`in der FHIR RESTful R4-API-Dokumentation.

Um eine FHIR-Ressource zu erstellen

1. Sammeln HealthLake `region` und `datastoreId` schätzen. Weitere Informationen finden Sie unter Eigenschaften des Datenspeichers abrufen.
2. Ermitteln Sie den Typ der `Resource` zu erstellenden FHIR. Weitere Informationen finden Sie unter Ressourcentypen.
3. Konstruieren Sie mithilfe der gesammelten Werte für HealthLake `region` und `datastoreId` eine URL für die Anfrage. Geben Sie auch den zu erstellenden `Resource` FHIR-Typ an. Scrollen Sie über die Schaltfläche Kopieren, um den gesamten URL-Pfad im folgenden Beispiel anzuzeigen.

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

4. Konstruieren Sie einen JSON-Hauptteil für die Anfrage und geben Sie die FHIR-Daten für die neue Ressource an. Für dieses Verfahren verwenden wir eine `Patient` FHIR-Ressource. Speichern Sie die Datei also unter. `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. Senden Sie die Anforderung . Die `create` FHIR-Interaktion verwendet eine POST Anfrage mit entweder AWS Signature Version 4 oder SMART bei der FHIR-Autorisierung. In den folgenden Beispielen wird eine `Patient` FHIR-Ressource entweder HealthLake mithilfe von curl oder der Konsole erstellt. HealthLake Scrollen Sie über die Schaltfläche Kopieren, um ein vollständiges Beispiel anzuzeigen.

   SigV4

   SigV4-Autorisierung

   ```
   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

   Beispiel für die SMART-on-FHIR-Autorisierung für den `IdentityProviderConfiguration`Datentyp.

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

Der Anrufer kann im Autorisierungs-Lambda Berechtigungen zuweisen. Weitere Informationen finden Sie unter OAuth 2.0-Bereiche.

AWS Console

> **Note**
>
> Die HealthLake Konsole unterstützt nur die AWS SigV4-Autorisierung.

1. Melden Sie sich auf der Seite „Abfrage ausführen“ in der HealthLake Konsole an.
2. Treffen Sie im Abschnitt Abfrageeinstellungen die folgenden Optionen.
   - Datenspeicher-ID — Wählen Sie eine Datenspeicher-ID aus, um eine Abfragezeichenfolge zu generieren.
   - Abfragetyp — wählen Sie`Create`.
   - Ressourcentyp — wählen Sie den zu erstellenden FHIR-Ressourcentyp aus.
   - Hauptteil der Anfrage — Konstruieren Sie einen JSON-Hauptteil für die Anfrage und geben Sie die FHIR-Daten für die neue Ressource an.
3. Wählen Sie Abfrage ausführen.

Konfiguration der Validierungsebene für die Ressourcenerstellung

Wenn Sie eine FHIR-Ressource erstellen, können Sie optional einen `x-amzn-healthlake-fhir-validation-level` HTTP-Header angeben, um eine Validierungsebene für die Ressource zu konfigurieren. AWS HealthLake unterstützt derzeit die folgenden Validierungsstufen:

- `strict`: Ressourcen werden anhand des Profilelements der Ressource oder der R4-Spezifikation, falls kein Profil vorhanden ist, validiert. Dies ist die Standardvalidierungsebene für AWS HealthLake.

- `structure-only`: Ressourcen werden anhand von R4 validiert, wobei alle referenzierten Profile ignoriert werden.
- `minimal`: Ressourcen werden minimal validiert, wobei bestimmte R4-Regeln ignoriert werden. Ressourcen, die die erforderlichen Strukturprüfungen nicht bestehen, search/analytics werden aktualisiert und enthalten nun eine Warnung zur Prüfung.

Ressourcen, die mit der Mindestvalidierungsstufe erstellt wurden, können in einen Datenspeicher aufgenommen werden, obwohl die für die Suchindizierung erforderliche Überprüfung fehlgeschlagen ist. In diesem Fall werden die Ressourcen um eine Healthlake-spezifische Erweiterung erweitert, um die genannten Fehler zu dokumentieren:

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

Zusätzlich wird der folgende HTTP-Antwort-Header mit dem Wert „true“ hinzugefügt:

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

> **Note**
>
> Die aufgenommenen Daten, die gemäß der R4-Spezifikation fehlerhaft formatiert sind, können möglicherweise nicht wie erwartet durchsucht werden, wenn diese Fehler vorliegen.

Eine FHIR-Ressource lesen

Die `read` FHIR-Interaktion liest den aktuellen Status einer Ressource in einem HealthLake Datenspeicher. Weitere Informationen finden Sie `read`in der FHIR RESTful R4-API-Dokumentation.

Um eine FHIR-Ressource zu lesen

1. Sammeln HealthLake `region` und `datastoreId` schätzen. Weitere Informationen finden Sie unter Eigenschaften des Datenspeichers abrufen.

2. Ermitteln Sie den Typ von FHIR`Resource`, der gelesen werden soll, und erfassen Sie den zugehörigen `id` Wert. Weitere Informationen finden Sie unter Ressourcentypen.
3. Konstruieren Sie mithilfe der gesammelten Werte für HealthLake `region` und `datastoreId` eine URL für die Anfrage. Geben Sie auch den `Resource` FHIR-Typ und den zugehörigen `id` Typ an. Scrollen Sie über die Schaltfläche Kopieren, um den gesamten URL-Pfad im folgenden Beispiel anzuzeigen.

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

4. Senden Sie die Anforderung . Die `read` FHIR-Interaktion verwendet eine `GET` Anfrage mit entweder AWS Signature Version 4 oder SMART bei der FHIR-Autorisierung. Im folgenden `curl` Beispiel wird der aktuelle Status einer `Patient` FHIR-Ressource eingelesen. HealthLake Scrollen Sie über die Schaltfläche Kopieren, um das gesamte Beispiel anzuzeigen.

   SigV4

   SigV4-Autorisierung

```
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

   Beispiel für die SMART-on-FHIR-Autorisierung für den `IdentityProviderConfiguration`Datentyp.

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

Der Anrufer kann im Autorisierungs-Lambda Berechtigungen zuweisen. Weitere Informationen finden Sie unter OAuth 2.0-Bereiche.

AWS Console

1. Melden Sie sich auf der Seite „Abfrage ausführen“ in der HealthLake Konsole an.
2. Treffen Sie im Abschnitt Abfrageeinstellungen die folgenden Optionen.
   - Datenspeicher-ID — Wählen Sie eine Datenspeicher-ID aus, um eine Abfragezeichenfolge zu generieren.
   - Abfragetyp — wählen Sie`Read`.
   - Ressourcentyp — wählen Sie den FHIR-Ressourcentyp aus, der gelesen werden soll.
   - Ressourcen-ID — geben Sie die FHIR-Ressourcen-ID ein.
3. Wählen Sie Abfrage ausführen.

Lesen der FHIR-Ressourcengeschichte

Die `history` FHIR-Interaktion ruft den Verlauf einer bestimmten FHIR-Ressource in einem Datenspeicher ab. HealthLake Mithilfe dieser Interaktion können Sie feststellen, wie sich der Inhalt einer FHIR-Ressource im Laufe der Zeit verändert hat. Bei der Abstimmung mit Audit-Logs ist es auch nützlich, den Status einer Ressource vor und nach der Änderung zu überprüfen. Die FHIR-Interaktionen `create` und `delete` führen zu einer historischen Version der Ressource, die gespeichert werden soll. `update` Weitere Informationen finden Sie `history`in der FHIR RESTful R4-API-Dokumentation.

> **Note**
>
> Sie können sich `history` für bestimmte FHIR-Ressourcentypen abmelden. Um sich abzumelden, erstellen Sie einen Fall mit AWS Support Center Console. Um Ihren Fall zu erstellen, melden Sie sich bei Ihrem an AWS-Konto und wählen Sie Fall erstellen.

Um den FHIR-Ressourcenverlauf zu lesen

1. Sammeln HealthLake `region` und `datastoreId` schätzen. Weitere Informationen finden Sie unter Eigenschaften des Datenspeichers abrufen.
2. Ermitteln Sie den Typ von FHIR`Resource`, der gelesen werden soll, und erfassen Sie den zugehörigen `id` Wert. Weitere Informationen finden Sie unter Ressourcentypen.
3. Konstruieren Sie mithilfe der gesammelten Werte für HealthLake `region` und `datastoreId` eine URL für die Anfrage. Geben Sie auch den `Resource` FHIR-Typ, die zugehörigen `id` und optionalen Suchparameter an. Scrollen Sie über die Schaltfläche Kopieren, um den gesamten URL-Pfad im folgenden Beispiel anzuzeigen.

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

HealthLake unterstützte Suchparameter für die FHIR-Interaktion **`history`**

Parameter	Description
`_count : integer`	Die maximale Anzahl von Suchergebnissen auf einer Seite. Der Server gibt die angeforderte Anzahl oder die maximale Anzahl von Suchergebnissen zurück, die standardmäßig für den Datenspeicher zulässig sind, je nachdem, welcher Wert niedriger ist.
`_since : instant`	Schließt nur Ressourcenversionen ein, die zu oder nach dem angegebenen Zeitpunkt erstellt wurden.

Parameter	Description
`_at : date(Time)`	Schließt nur Ressourcenversionen ein, die zu einem bestimmten Zeitpunkt während des im Datums- und Uhrzeitwert angegebenen Zeitraums aktuell waren. Weitere Informationen finden Sie `date` in der HL7 RESTfulFHIR-API-Dokumentation.

4. Senden Sie die Anforderung . Die `history` FHIR-Interaktion verwendet eine `GET` Anfrage mit entweder AWS Signature Version 4 oder SMART bei der FHIR-Autorisierung. Im folgenden `curl` Beispiel wird der `_count` Suchparameter verwendet, um 100 historische Suchergebnisse pro Seite für eine `Patient` FHIR-Ressource in zurückzugeben. HealthLake Scrollen Sie über die Schaltfläche Kopieren, um das gesamte Beispiel anzuzeigen.

   SigV4

   SigV4-Autorisierung

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

   SMART on FHIR

   SMART on FHIR-Autorisierungsbeispiel für den `IdentityProviderConfiguration` Datentyp.

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

Der Anrufer kann im Autorisierungs-Lambda Berechtigungen zuweisen. Weitere Informationen finden Sie unter OAuth 2.0-Bereiche.

Der zurückgegebene Inhalt einer `history` Interaktion ist in einer FHIR-Ressource enthaltenBundle, wobei der Typ auf gesetzt ist. `history` Sie enthält den angegebenen Versionsverlauf, sortiert nach den ältesten Versionen zuletzt, und enthält gelöschte Ressourcen. Weitere Informationen finden Sie `Resource Bundle`in der FHIR R4-Dokumentation.

Lesen des versionsspezifischen FHIR-Ressourcenverlaufs

Die `vread` FHIR-Interaktion führt ein versionsspezifisches Lesen einer Ressource in einem Datenspeicher durch. HealthLake Mithilfe dieser Interaktion können Sie den Inhalt einer FHIR-Ressource so anzeigen, wie er zu einem bestimmten Zeitpunkt in der Vergangenheit war.

> **Note**
>
> Wenn Sie die `history` FHIR-Interaktion ohne verwendenvread, wird HealthLake immer die neueste Version der Metadaten der Ressource zurückgegeben.

HealthLake erklärt, dass es die Versionierung `CapabilityStatement.rest.resource.versioning`für jede unterstützte Ressource unterstützt. Alle HealthLake Datenspeicher enthalten `Resource.meta.versionId` (`vid`) für alle Ressourcen.

Wenn die `history` FHIR-Interaktion aktiviert ist (standardmäßig für Datenspeicher, die nach dem 25.10.2024 erstellt wurden, oder auf Anfrage für ältere Datenspeicher), enthält die `Bundle` Antwort

das `vid` als Teil des Elements. `location` Im folgenden Beispiel `vid` wird das als Zahl angezeigt. 1 Das vollständige Beispiel finden Sie unter Beispiel Bundle/Bundle-Response (JSON).

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

Um den versionsspezifischen FHIR-Ressourcenverlauf zu lesen

1. Sammeln HealthLake `region` und bewerten. `datastoreId` Weitere Informationen finden Sie unter Eigenschaften des Datenspeichers abrufen.
2. Ermitteln Sie den `Resource` FHIR-Typ, der gelesen und die zugehörigen `id` `vid` Werte gesammelt werden soll. Weitere Informationen finden Sie unter Ressourcentypen.
3. Konstruieren Sie anhand der für HealthLake und FHIR gesammelten Werte eine URL für die Anfrage. Scrollen Sie über die Schaltfläche Kopieren, um den gesamten URL-Pfad im folgenden Beispiel anzuzeigen.

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

4. Senden Sie die Anforderung . Die `history` FHIR-Interaktion verwendet eine `GET` Anfrage mit entweder AWS Signature Version 4 oder SMART bei der FHIR-Autorisierung. Die folgende `vread` Interaktion gibt eine einzelne Instanz mit dem für die `Patient` FHIR-Ressource angegebenen Inhalt für die Version der Ressourcenmetadaten zurück, die von der angegeben wurde. `vid` Scrollen Sie über die Schaltfläche Kopieren, um den gesamten URL-Pfad im folgenden Beispiel anzuzeigen.

   SigV4

   SigV4-Autorisierung

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

SMART on FHIR

SMART on FHIR-Autorisierungsbeispiel für den `IdentityProviderConfiguration`Datentyp.

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

Der Anrufer kann im Autorisierungs-Lambda Berechtigungen zuweisen. Weitere Informationen finden Sie unter OAuth 2.0-Bereiche.

Aktualisierung einer FHIR-Ressource

Die `update` FHIR-Interaktion erstellt eine neue aktuelle Version für eine vorhandene Ressource oder erstellt eine erste Version, wenn für die angegebene Ressource noch keine Ressource vorhanden ist. `id` Weitere Informationen finden Sie `update`in der FHIR RESTful R4-API-Dokumentation.

Um eine FHIR-Ressource zu aktualisieren

1. Sammeln HealthLake `region` und bewerten`datastoreId`. Weitere Informationen finden Sie unter Eigenschaften des Datenspeichers abrufen.

2. Ermitteln Sie den Typ des FHIR`Resource`, der aktualisiert werden soll, und erfassen Sie den zugehörigen `id` Wert. Weitere Informationen finden Sie unter Ressourcentypen.

3. Konstruieren Sie mithilfe der gesammelten Werte für HealthLake `region` und `datastoreId` eine URL für die Anfrage. Geben Sie auch den `Resource` FHIR-Typ und den zugehörigen `id` Typ an. Scrollen Sie über die Schaltfläche Kopieren, um den gesamten URL-Pfad im folgenden Beispiel anzuzeigen.

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

4. Erstellen Sie einen `JSON` Text für die Anfrage, in dem Sie angeben, welche FHIR-Datenaktualisierungen vorgenommen werden sollen. Speichern Sie die Datei für die Zwecke dieses Verfahrens unter. `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. Senden Sie die Anforderung . Die `update` FHIR-Interaktion verwendet eine `PUT` Anfrage mit entweder AWS Signature Version 4 oder SMART bei der FHIR-Autorisierung. Das folgende `curl` Beispiel aktualisiert eine `Patient` Ressource in. HealthLake Scrollen Sie über die Schaltfläche Kopieren, um das gesamte Beispiel anzuzeigen.

SigV4

SigV4-Autorisierung

```
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
```

Ihre Anfrage gibt einen `200` HTTP-Statuscode zurück, wenn eine bestehende Ressource aktualisiert wird, oder einen `201` HTTP-Statuscode, wenn eine neue Ressource erstellt wird.

SMART on FHIR

SMART on FHIR-Autorisierungsbeispiel für den [IdentityProviderConfiguration](#)Datentyp.

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

Der Anrufer kann im Autorisierungs-Lambda Berechtigungen zuweisen. Weitere Informationen finden Sie unter OAuth 2.0-Bereiche.

AWS Console

1. Melden Sie sich auf der Seite „Abfrage ausführen“ in der HealthLake Konsole an.
2. Treffen Sie im Abschnitt Abfrageeinstellungen die folgenden Optionen.

- Datenspeicher-ID — Wählen Sie eine Datenspeicher-ID aus, um eine Abfragezeichenfolge zu generieren.
- Abfragetyp — wählen Sie`Update (PUT)`.
- Ressourcentyp — Wählen Sie den FHIR-Ressourcentyp aus, den Sie aktualisieren oder erstellen möchten.
- Hauptteil der Anfrage — Konstruieren Sie einen JSON-Hauptteil für die Anfrage und geben Sie die FHIR-Daten an, mit denen die Ressource aktualisiert werden soll.

3. Wählen Sie Abfrage ausführen.

Aktualisierung der FHIR-Ressourcen auf der Grundlage von Bedingungen

Die bedingte Aktualisierung ermöglicht es Ihnen, eine vorhandene Ressource anhand einiger Identifikations-Suchkriterien und nicht anhand logischer FHIR `id` zu aktualisieren. Wenn der Server das Update verarbeitet, führt er eine Suche mithilfe seiner Standardsuchfunktionen für den Ressourcentyp durch, mit dem Ziel, eine `id` für die Anfrage logische Lösung zu finden.

Die Aktion, die der Server ergreift, hängt davon ab, wie viele Treffer er findet:

- Keine Treffer, keine **`id`** Angabe im Anfragetext: Der Server erstellt die FHIR-Ressource.
- Keine Treffer, **`id`** angegeben und die Ressource ist noch nicht vorhanden mit**`id`**: Der Server behandelt die Interaktion als Update as Create-Interaktion.
- Keine Treffer, **`id`** angegeben und bereits vorhanden: Der Server lehnt das Update mit einem `409 Conflict` Fehler ab.
- Ein Treffer, keine Ressource **`id`** bereitgestellt ODER (Ressource **`id`** bereitgestellt und sie entspricht der gefundenen Ressource): Der Server führt das Update anhand der passenden Ressource wie oben beschrieben durch. Wenn die Ressource aktualisiert wurde, MUSS der Server a `200 OK` zurückgeben.

- One Match, **`id`** bereitgestellte Ressource, entspricht aber nicht der gefundenen Ressource: Der Server gibt einen `409 Conflict` Fehler zurück, der darauf hinweist, dass die Client-ID-Spezifikation ein Problem war, vorzugsweise mit einem `OperationOutcome`
- Mehrere Treffer: Der Server gibt einen `412 Precondition Failed` Fehler zurück, der darauf hinweist, dass die Kriterien des Clients nicht selektiv genug waren, vorzugsweise mit einem OperationOutcome

Im folgenden Beispiel wird eine `Patient` Ressource aktualisiert, deren Name Peter ist, deren Geburtsdatum der 1. Januar 2000 ist und deren Telefonnummer 1234567890 ist.

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

Konfiguration der Validierungsebene für Ressourcenupdates

Bei der Aktualisierung einer FHIR-Ressource können Sie optional einen `x-amzn-healthlake-fhir-validation-level` HTTP-Header angeben, um eine Validierungsebene für die Ressource zu konfigurieren. AWS HealthLake unterstützt derzeit die folgenden Validierungsstufen:

- `strict`: Ressourcen werden anhand des Profilelements der Ressource oder der R4-Spezifikation, falls kein Profil vorhanden ist, validiert. Dies ist die Standardvalidierungsebene für AWS HealthLake.
- `structure-only`: Ressourcen werden anhand von R4 validiert, wobei alle referenzierten Profile ignoriert werden.
- `minimal`: Ressourcen werden minimal validiert, wobei bestimmte R4-Regeln ignoriert werden. Ressourcen, die die erforderlichen Strukturprüfungen nicht bestehen, search/analytics werden aktualisiert und enthalten nun eine Warnung zur Prüfung.

Ressourcen, die mit der Mindestvalidierungsstufe aktualisiert wurden, können in einen Datenspeicher aufgenommen werden, obwohl die für die Suchindizierung erforderliche Überprüfung fehlgeschlagen ist. In diesem Fall werden die Ressourcen um eine Healthlake-spezifische Erweiterung erweitert, um die genannten Fehler zu dokumentieren:

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

Zusätzlich wird der folgende HTTP-Antwort-Header mit dem Wert „true“ hinzugefügt:

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

> **Note**
>
> Beachten Sie, dass aufgenommene Daten, die gemäß der R4-Spezifikation falsch formatiert sind, möglicherweise nicht wie erwartet durchsucht werden können, wenn diese Fehler vorliegen.

Ändern von Ressourcen mit PATCH-Operation

AWS HealthLake unterstützt den PATCH-Vorgang für FHIR-Ressourcen, sodass Sie Ressourcen ändern können, indem Sie gezielt bestimmte Elemente hinzufügen, ersetzen oder löschen möchten, ohne die gesamte Ressource zu aktualisieren. Dieser Vorgang ist besonders nützlich, wenn Sie:

- Nehmen Sie gezielte Aktualisierungen großer Ressourcen vor
- Reduzieren Sie die Nutzung der Netzwerkbandbreite
- Führen Sie atomare Änderungen an bestimmten Ressourcenelementen durch
- Minimiere das Risiko, dass gleichzeitige Änderungen überschrieben werden
- Aktualisieren Sie Ressourcen im Rahmen von Batch- und Transaktions-Workflows

Unterstützte PATCH-Formate

AWS HealthLake unterstützt zwei Standard-PATCH-Formate:

JSON-Patch (RFC 6902)

Verwendet die JSON-Pointer-Syntax, um Elemente anhand ihrer Position in der Ressourcenstruktur als Ziel auszuwählen.

Inhaltstyp: `application/json-patch+json`

FHIRPath Patch (FHIR R4-Spezifikation)

Verwendet FHIRPath Ausdrücke, um Elemente anhand ihres Inhalts und ihrer Beziehungen gezielt anzusprechen, und bietet so einen FHIR-nativen Patchansatz.

Inhaltstyp: `application/fhir+json`

Usage

Direkte PATCH-Operationen

Der PATCH-Vorgang kann mit der PATCH-HTTP-Methode direkt auf FHIR-Ressourcen aufgerufen werden:

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

PATCH in Paketen

PATCH-Operationen können als Einträge in FHIR-Bundles des Typs enthalten sein `batch` oder ermöglichen es Ihnen`transaction`, Patch-Operationen mit anderen FHIR-Interaktionen (Erstellen, Lesen, Aktualisieren, Löschen) in einer einzigen Anfrage zu kombinieren.

- Transaktionspakete: Alle Einträge sind atomar erfolgreich oder schlagen fehl
- Batch-Bundles: Jeder Eintrag wird unabhängig verarbeitet

JSON-Patch-Format

Unterstützte Vorgänge

Operation	Description
`add`	Fügen Sie der Ressource einen neuen Wert hinzu
`remove`	Entferne einen Wert aus der Ressource
`replace`	Ersetzt einen vorhandenen Wert in der Ressource
`move`	Entferne einen Wert von einem Ort und füge ihn einem anderen hinzu

Operation	Description
`copy`	Kopiert einen Wert von einem Ort an einen anderen
`test`	Testen Sie, ob ein Wert an der Zielposition einem bestimmten Wert entspricht

Pfadsyntax

Der JSON-Patch verwendet die JSON-Pointer-Syntax (RFC 6901):

Beispiel für einen Pfad	Description
`/name/0/family`	Das Familienelement des Vornamens
`/telecom/-`	An das Telekommunikations-Array anhängen
`/active`	Aktives Element auf Stammebene
`/address/0/line/1`	Zweite Zeile der ersten Adresse

Beispiele

Direkte JSON-Patch-Anfrage mit mehreren Vorgängen

```
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"
  }
]
```

Direkte JSON-Patch-Anfrage mit einem einzigen Vorgang

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

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

JSON-Patch im Paket

Verwenden Sie eine Binärressource, die die Base64-kodierte JSON-Patch-Payload enthält:

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

FHIRPath Patch-Format

Unterstützte Vorgänge

Operation	Description
`add`	Fügt einer Ressource ein neues Element hinzu
`insert`	Fügt ein Element an einer bestimmten Position in einer Liste ein
`delete`	Ein Element aus einer Ressource entfernen
`replace`	Ersetzt den Wert eines vorhandenen Elements
`move`	Ordnen Sie Elemente innerhalb einer Liste neu an

Pfadsyntax

FHIRPath Patch verwendet FHIRPath Ausdrücke und unterstützt:

- Indexbasierter Zugriff: `Patient.name[0]`

- Filterung mit: `where()` `Patient.name.where(use = 'official')`
- Boolesche Logik: `Patient.telecom.where(system = 'phone' and use = 'work')`
- Funktionen zur Unterteilung:, `first()` `last()`
- Existenzprüfungen:`exists()`, `count()`
- Polymorphe Navigation: `0bservation.value`

Beispiele

Direkte Patch-Anfrage 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 im Paket

Verwenden Sie eine Parameter-Ressource als Eingangsressource mit`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"
    }
  }]
}
```

Anforderungsheader

Header	Erforderlich	Beschreibung
`Content-Type`	Ja	`application/json-patch+json` für JSON Patch oder `application/fhir+json` für FHIRPath Patch
`If-Match`	Nein	Versionsspezifisches bedingtes Update mit ETag

Beispielantwort

Der Vorgang gibt die aktualisierte Ressource mit neuen Versionsinformationen zurück:

```
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"
  }
}
```

Behavior

Die PATCH-Operation:

- Validiert die Patch-Syntax gemäß der entsprechenden Spezifikation (RFC 6902 für JSON Patch, FHIR R4 für Patch) FHIRPath
- Wendet Operationen atomar an — alle Operationen sind erfolgreich oder alle schlagen fehl
- Aktualisiert die Versions-ID der Ressource und erstellt einen neuen Verlaufseintrag
- Behält die ursprüngliche Ressource im Verlauf bei, bevor Änderungen übernommen werden
- Überprüft die FHIR-Ressourcenbeschränkungen nach dem Anwenden von Patches
- Unterstützt bedingte Aktualisierungen mithilfe des If-Match-Headers mit ETag

Fehlerbehandlung

Der Vorgang behandelt die folgenden Fehlerbedingungen:

- 400 Bad Request: Ungültige Patch-Syntax (fehlerhafte Anfrage oder fehlerhaftes Patch-Dokument)
- 404 Nicht gefunden: Ressource nicht gefunden (angegebene ID existiert nicht)
- 409-Konflikt: Versionskonflikt (gleichzeitige Updates oder veraltete Versions-ID angegeben)
- 422 Unverarbeitbare Entität: Patch-Operationen können nicht auf die angegebenen Ressourcenelemente angewendet werden

Zusammenfassung der Funktionen

Funktion	JSON-Patch	FHIRPath Aufnäher
Art des Inhalts	`application/json-patch+json`	`application/fhir+json`
Pfadformat	JSON-Zeiger (RFC 6901)	FHIRPath Ausdrücke
Direkte PATCH-API	Unterstützt	Unterstützt
Batch bündeln	Unterstützt (über Binary)	Unterstützt (über Parameter)
Transaktion bündeln	Unterstützt (per Binärdatei)	Unterstützt (über Parameter)
Operationen	hinzufügen, entfernen, ersetzen, verschieben, kopieren, testen	hinzufügen, einfügen, löschen, ersetzen, verschieben

Einschränkungen

- Bedingte PATCH-Operationen, die Suchbedingungen verwenden, werden nicht unterstützt
- JSON-Patch in Bundles muss Binärressourcen mit Base64-codiertem Inhalt verwenden
- FHIRPath Patches in Bundles müssen Parameter-Ressourcen verwenden

Weitere Ressourcen

Weitere Informationen zu PATCH-Vorgängen finden Sie unter:

- FHIR R4 PATCH-Dokumentation
- FHIR R4-Patch-Spezifikation FHIRPath
- RFC 6902 — JSON-Patch
- RFC 6901 — JSON-Zeiger

Bündelung der FHIR-Ressourcen

Ein FHIR `Bundle` ist ein Container für eine Sammlung von FHIR-Ressourcen in. AWS HealthLake AWS HealthLake unterstützt zwei Arten von Bundles mit unterschiedlichem Verarbeitungsverhalten.

`Batch`Bundles verarbeiten jede Ressource unabhängig voneinander. Wenn eine Ressource ausfällt, können die verbleibenden Ressourcen trotzdem erfolgreich sein. Jeder Vorgang wird einzeln verarbeitet, und die Verarbeitung wird auch dann fortgesetzt, wenn einige Operationen fehlschlagen. Verwenden Sie Batch-Pakete für Massenoperationen, bei denen ein teilweiser Erfolg akzeptabel ist, wie z. B. das Hochladen mehrerer Patientenakten, die nichts miteinander zu tun haben.

`Transaction`Bündel verarbeiten alle Ressourcen atomar als eine einzige Einheit. Entweder sind alle Ressourcenoperationen erfolgreich, oder es wird kein AWS HealthLake Commit ausgeführt. Verwenden Sie Transaktionspakete, wenn Sie eine garantierte referenzielle Integrität für alle verwandten Ressourcen benötigen, z. B. wenn Sie einen Patienten mit ähnlichen Beobachtungen und Erkrankungen erstellen möchten, bei denen alle Daten zusammen aufgezeichnet werden müssen.

Unterschiede zwischen Batch- und Transaktionspaketen

Feature	Batch	Transaktion
Verarbeitungsmodell	Jeder Vorgang ist unabhängig erfolgreich oder schlägt fehl.	Alle Operationen sind als eine einzige atomare Einheit erfolgreich oder fehlschlagen.
Fehlerbehandlung	Die Verarbeitung wird fortgesetzt, auch wenn einzelne Operationen fehlschlagen.	Das gesamte Paket schlägt fehl, wenn ein einzelner Vorgang fehlschlägt.
Reihenfolge der Ausführung	Die Ausführungsreihenfolge ist nicht garantiert.	Die Operationen werden in der angegebenen Reihenfolge verarbeitet.
Referentielle Integrität	Wird nicht betriebsübergreifend durchgesetzt.	Wird für lokal referenzierte Ressourcen innerhalb des Pakets durchgesetzt.

Feature	Batch	Transaktion
Am besten verwendet für	Massenoperationen, bei denen ein teilweiser Erfolg akzeptabel ist.	Verwandte Ressourcen, die zusammen erstellt oder aktualisiert werden müssen.

Sie können FHIR-Ressourcen desselben oder verschiedener Typen bündeln, und sie können eine Mischung aus FHIR-Vorgängen wie,`create`, `read` `update``delete`, und beinhalten. `patch` Weitere Informationen finden Sie unter Resource Bundle in der FHIR R4-Dokumentation.

Im Folgenden finden Sie Beispiele für Anwendungsfälle für jeden Bundle-Typ.

Batch-Pakete

- Laden Sie während der nächtlichen Datensynchronisierung mehrere Patientenakten aus verschiedenen Einrichtungen hoch, die nichts miteinander zu tun haben.
- Laden Sie historische Medikamentendaten in großen Mengen hoch, bei denen es bei einigen Aufzeichnungen zu Validierungsproblemen kommen könnte.
- Laden Sie Referenzdaten wie Organisationen und Praktiker, bei denen einzelne Fehler keine Auswirkungen auf andere Einträge haben.

Transaktionspakete

- Erstellen Sie einen Patienten mit entsprechenden Beobachtungen und Zuständen während einer Aufnahme in die Notaufnahme, bei der alle Daten zusammen aufgezeichnet werden müssen.
- Aktualisieren Sie die Medikamentenliste eines Patienten und die zugehörigen Allergie-Informationen, die konsistent bleiben müssen.
- Zeichnen Sie eine vollständige Begegnung mit dem Patienten, Beobachtungen, Verfahren und Abrechnungsinformationen in einer einzigen atomaren Einheit auf.

> **Important**
>
> Sowohl Batch- als auch Transaktionspakete verwenden dieselbe `Bundle` Ressourcenstruktur. Der einzige Unterschied ist der Wert des `type` Feldes.

Das folgende Beispiel zeigt ein Transaktionspaket mit mehreren Ressourcentypen und Vorgängen.

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

Bündelung von FHIR-Ressourcen als unabhängige Einheiten

Um FHIR-Ressourcen als unabhängige Einheiten zu bündeln

1. Sammeln HealthLake `region` und bewertendatastoreId. Weitere Informationen finden Sie unter Eigenschaften des Datenspeichers abrufen.

2. Konstruieren Sie mithilfe der gesammelten Werte für HealthLake `region` und eine URL für die Anfragedatastore`Id`. Geben Sie in der URL keinen FHIR-Ressourcentyp an. Scrollen Sie über die Schaltfläche Kopieren, um den gesamten URL-Pfad im folgenden Beispiel anzuzeigen.

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

3. Konstruieren Sie einen JSON-Hauptteil für die Anfrage und geben Sie jedes HTTP-Verb als Teil der `method` Elemente an. Im folgenden Beispiel wird eine `batch` Typinteraktion mit der `Bundle` Ressource verwendet, um neue `Medication` Ressourcen `Patient` zu erstellen. Alle erforderlichen Abschnitte sind entsprechend kommentiert. Speichern Sie die Datei für dieses Verfahren unter`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. Senden Sie die Anforderung . Der `Bundle` Batchtyp FHIR verwendet eine `POST` Anfrage mit entweder AWS Signature Version 4 oder SMART bei der FHIR-Autorisierung. Im folgenden Codebeispiel wird das `curl` Befehlszeilentool zu Demonstrationszwecken verwendet.

   SigV4

   SigV4-Autorisierung

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

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

SMART on FHIR

SMART on FHIR-Autorisierungsbeispiel für den `IdentityProviderConfiguration`Datentyp.

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

Der Anrufer kann im Autorisierungs-Lambda Berechtigungen zuweisen. Weitere Informationen finden Sie unter OAuth 2.0-Bereiche.

Der Server gibt eine Antwort zurück, in der die `Medication` Ressourcen `Patient` und die Ressourcen angezeigt werden, die als Ergebnis der `Bundle` Batch-Anfrage erstellt wurden.

PUTs Befristet in Paketen

AWS HealthLake unterstützt bedingte Aktualisierungen innerhalb von Bundles mit den folgenden Abfrageparametern:

> **Note**
>
> Bedingte PUTs werden nur in `batch` Bundles unterstützt. `Transaction`Bundles unterstützen Conditional nicht. PUTs

- `_id`(eigenständig)
- `_id`in Kombination mit einer der folgenden Optionen:
  - `_tag`
  - `_createdAt`
  - `_lastUpdated`

Wenn Sie Conditional PUTs in Bundles verwenden, werden die Abfrageparameter anhand vorhandener Ressourcen AWS HealthLake ausgewertet und auf der Grundlage der Vergleichsergebnisse Maßnahmen ergriffen.

Verhalten bei bedingten Aktualisierungen

Szenario	HTTP-Status	Ergriffene Maßnahme
Ressource ohne angegebene ID	201 erstellt	Erstellt immer eine neue Ressource.
Ressource mit neuer ID (keine Übereinstimmung)	201 erstellt	Erstellt eine neue Ressource mit der angegebenen ID.
Ressource mit vorhandener ID (einziger Treffer)	200 OK	Aktualisiert die passende Ressource.
Ressource mit vorhandener ID (Konflikt erkannt)	409 Conflict (409 Konflikt)	Gibt einen Fehler zurück. Es wurden keine Änderungen vorgenommen.
Ressource mit vorhandener ID (ID stimmt nicht überein)	400 Bad Request (400 Ungültige Anfrage)	Gibt einen Fehler zurück. Es wurden keine Änderungen vorgenommen.

Szenario	HTTP-Status	Ergriffene Maßnahme
Mehrere Ressourcen entsprechen den Bedingungen	412 Precondition Failed (412 Vorbedingung fehlgeschlagen)	Gibt einen Fehler zurück. Es wurden keine Änderungen vorgenommen.

Im folgenden Beispielpaket mit einer bedingten Aktualisierung wird die `Patient` Ressource mit der FHIR-ID nur `476` aktualisiert, wenn die Bedingung erfüllt `_lastUpdated=lt2025-04-20` ist.

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

Bündelung von FHIR-Ressourcen zu einer einzigen Einheit

Um FHIR-Ressourcen zu einer einzigen Einheit zu bündeln

1. Sammeln HealthLake `region` und `datastoreId` schätzen. Weitere Informationen finden Sie unter Eigenschaften des Datenspeichers abrufen.
2. Konstruieren Sie mithilfe der gesammelten Werte für HealthLake `region` und eine URL für die Anfrage`datastoreId`. Nehmen Sie den FHIR-Ressourcentyp `Bundle` als Teil der URL auf. Scrollen Sie über die Schaltfläche Kopieren, um den gesamten URL-Pfad im folgenden Beispiel anzuzeigen.

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

3. Konstruieren Sie einen JSON-Hauptteil für die Anfrage und geben Sie die FHIR-Ressourcen an, die gruppiert werden sollen. Im folgenden Beispiel werden zwei `Patient` Ressourcen gruppiert HealthLake. Speichern Sie die Datei für dieses Verfahren unter`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. Senden Sie die Anforderung . Der `Bundle` Dokumenttyp FHIR verwendet eine `POST` Anfrage mit dem AWS Signature Version 4-Signaturprotokoll. Im folgenden Codebeispiel wird das `curl` Befehlszeilentool zu Demonstrationszwecken verwendet.

   SigV4

   SigV4-Autorisierung

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

   SMART on FHIR

   SMART on FHIR-Autorisierungsbeispiel für den `IdentityProviderConfiguration`Datentyp.

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

Der Anrufer kann im Autorisierungs-Lambda Berechtigungen zuweisen. Weitere Informationen finden Sie unter OAuth 2.0-Bereiche.

Der Server gibt eine Antwort zurück, in der zwei `Patient` Ressourcen angezeigt werden, die als Ergebnis der `Bundle` Dokumenttypanforderung erstellt wurden.

Konfiguration der Validierungsebene für Bundles

Beim Bündeln von FHIR-Ressourcen können Sie optional einen `x-amzn-healthlake-fhir-validation-level` HTTP-Header angeben, um eine Validierungsebene für die Ressource zu konfigurieren. Diese Validierungsebene wird für alle Erstellungs- und Aktualisierungsanforderungen innerhalb des Bundles festgelegt. AWS HealthLake unterstützt derzeit die folgenden Validierungsstufen:

- `strict`: Ressourcen werden anhand des Profilelements der Ressource oder der R4-Spezifikation, falls kein Profil vorhanden ist, validiert. Dies ist die Standardvalidierungsebene für AWS HealthLake.
- `structure-only`: Ressourcen werden anhand von R4 validiert, wobei alle referenzierten Profile ignoriert werden.
- `minimal`: Ressourcen werden minimal validiert, wobei bestimmte R4-Regeln ignoriert werden. Ressourcen, die die erforderlichen Strukturprüfungen nicht bestehen, search/analytics werden aktualisiert und enthalten nun eine Warnung zur Prüfung.

Ressourcen, die mit der Mindestvalidierungsstufe gebündelt sind, können in einen Datenspeicher aufgenommen werden, obwohl die für die Suchindizierung erforderliche Überprüfung fehlgeschlagen

ist. In diesem Fall werden die Ressourcen mit einer Healthlake-spezifischen Erweiterung aktualisiert, um diese Fehler zu dokumentieren, und die Einträge in der Bundle-Antwort werden Ressourcen wie folgt enthalten: 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."
                        }
                    ]
                }
            }
        }
    ]
}
```

Zusätzlich wird der folgende HTTP-Antwort-Header mit dem Wert „true“ hinzugefügt:

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

> **Note**
>
> Beachten Sie, dass aufgenommene Daten, die gemäß der R4-Spezifikation falsch formatiert sind, möglicherweise nicht wie erwartet durchsucht werden können, wenn diese Fehler vorliegen.

Eingeschränkte Unterstützung für den Bundle-Typ „Nachricht“

HealthLake bietet eingeschränkte Unterstützung für den FHIR-Bundle-Typ `message` durch einen internen Konvertierungsprozess. Diese Unterstützung ist für Szenarien konzipiert, in denen Nachrichtenpakete nicht an der Quelle neu formatiert werden können, wie z. B. die Aufnahme von ADT-Feeds (Admission, Discharge, Transfer) aus älteren Krankenhaussystemen.

> **Warning**
>
> Diese Funktion erfordert eine explizite Zuteilung von AWS Benutzerkonten und erzwingt weder die Semantik noch die referenzielle Integrität von FHIR R4-Nachrichten. Wenden Sie sich an den AWS Support, um die Aktivierung für Ihr Konto zu beantragen, bevor Sie Nachrichtenpakete verwenden.

Hauptunterschiede zur standardmäßigen Nachrichtenverarbeitung

- Nachrichtenpakete (FHIR-Spezifikation): Der erste Eintrag muss ein Eintrag sein`MessageHeader`, der auf andere Ressourcen verweist. Ressourcen fehlen einzelne `request` Objekte, und das MessageHeader Ereignis bestimmt die Verarbeitungsaktionen.
- HealthLake Verarbeitung: Konvertiert Nachrichtenpakete in Batch-Bundles, indem jedem Ressourceneintrag automatisch PUT-Operationen zugewiesen werden. Ressourcen werden unabhängig voneinander verarbeitet, ohne dass die Nachrichtensemantik oder die referenzielle Integrität erzwungen werden.

Wichtige Einschränkungen

- FHIR R4-Nachrichtenspezifische Verarbeitungsregeln werden nicht durchgesetzt
- Keine ressourcenübergreifende Transaktionsintegrität
- Ressourcenübergreifende Referenzen werden nicht validiert

- Erfordert eine ausdrückliche Zulassungsliste für Konten

Beispiel für eine Struktur eines Nachrichtenpakets

```
{
              "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**
>
> Jede Ressource wird unabhängig gespeichert, als ob sie über einzelne PUT-Operationen übermittelt würde. Wenn eine vollständige FHIR-Messaging-Semantik oder eine Überprüfung der referentiellen Integrität erforderlich sind, verarbeiten Sie Nachrichtenpakete vorab oder implementieren Sie vor der Übermittlung eine Validierung auf Anwendungsebene.

Asynchrone Bundle-Transaktionen

AWS HealthLake unterstützt den asynchronen `Bundle` Typ`transaction`, mit dem Sie Transaktionen mit bis zu 500 Ressourcen einreichen können. Wenn Sie eine asynchrone Transaktion einreichen, wird sie zur Verarbeitung in eine HealthLake Warteschlange gestellt und sofort eine Abfrage-URL zurückgegeben. Sie können diese URL verwenden, um den Status zu überprüfen und die Antwort abzurufen. Dies folgt dem asynchronen FHIR-Bundle-Muster.

Wann sollten asynchrone Transaktionen verwendet werden

- Sie müssen mehr als 100 Ressourcen (synchrones Limit) in einer einzigen Transaktion einreichen.
- Sie möchten vermeiden, dass Ihre Anwendung blockiert wird, während Sie auf den Abschluss der Transaktionsverarbeitung warten.
- Sie müssen große Mengen verwandter Ressourcen mit besserem Durchsatz verarbeiten.

Important

Die Umfrageergebnisse sind nach Abschluss der Transaktion 90 Tage lang verfügbar. Nach diesem Zeitraum von 90 Tagen gibt die Abfrage-URL keine Ergebnisse mehr zurück. Entwerfen Sie Ihre Integration so, dass Ergebnisse in diesem Fenster abgerufen und gespeichert werden.

Note

Der synchrone `Bundle` Typ unterstützt `transaction` weiterhin bis zu 100 Ressourcen und ist der Standardverarbeitungsmodus. Wenn Sie einen `Bundle` Typ `transaction` mit mehr als 100 Ressourcen ohne `Prefer: respond-async` Header einreichen, wird ein `422 Unprocessable Entity` Fehler HealthLake zurückgegeben. Pakete mit Typ `batch` werden für die asynchrone Verarbeitung nicht unterstützt — nur `Bundle` Typen `transaction` können asynchron übermittelt werden (mit bis zu 500 Vorgängen).

> **Note**
>
> PATCHOperationen und Bedingungen PUTs werden in asynchronen Bundle-Transaktionen nicht unterstützt.

Eine asynchrone Transaktion einreichen

Um eine asynchrone Transaktion einzureichen, senden Sie eine POST Anfrage mit dem Header an den Datenspeicher-Endpunkt. `Prefer: respond-async` Das Paket muss den Typ `transaction` haben. Bundles mit Typ `batch` werden für die asynchrone Bundle-Verarbeitung nicht unterstützt.

HealthLake führt bei der Einreichung erste Validierungen für das Bundle durch. Wenn die Überprüfung erfolgreich ist, wird HTTP 202 Accepted mit einem `content-location` Antwortheader HealthLake zurückgegeben, der die Abfrage-URL enthält.

Um einen asynchronen Typ einzureichen **`Bundle transaction`**

1. Senden Sie eine POST Anfrage an den HealthLake Datenspeicher-Endpunkt.

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

2. Konstruieren Sie einen JSON-Hauptteil für die Anfrage mit dem Bundle-Typ`transaction`. Speichern Sie die Datei für dieses Verfahren unter`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. Senden Sie die Anfrage mit dem `Prefer: respond-async` Header. Der `Bundle` Transaktionstyp FHIR verwendet eine `POST` Anfrage mit entweder AWS Signature Version 4 oder SMART bei FHIR-Autorisierung. Im folgenden Codebeispiel wird das `curl` Befehlszeilentool zu Demonstrationszwecken verwendet.

   SigV4

   SigV4-Autorisierung

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

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

SMART on FHIR

SMART on FHIR-Autorisierungsbeispiel für den [IdentityProviderConfiguration](#)Datentyp.

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

Der Anrufer kann im Autorisierungs-Lambda Berechtigungen zuweisen. Weitere Informationen finden Sie unter OAuth 2.0-Bereiche.

4. Bei erfolgreicher Übermittlung gibt der Server HTTP 202 Accepted zurück. Der `content-location` Antwortheader enthält die Abfrage-URL. Der Antworttext ist eine `OperationOutcome` Ressource.

```
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"
            ]
        }
    ]
}
```

Abfrage des Transaktionsstatus

Nachdem Sie eine asynchrone Transaktion eingereicht haben, verwenden Sie die Abfrage-URL aus dem `content-location` Antwort-Header, um den Transaktionsstatus zu überprüfen. Senden Sie eine `GET` Anfrage an die Abfrage-URL.

> **Note**
>
> Für SMART auf FHIR-fähigen Datenspeichern muss das Autorisierungstoken `read` Berechtigungen für den `Transaction` Ressourcentyp enthalten, um den Transaktionsstatus abzufragen. Weitere Informationen zu SMART auf FHIR-Bereichen finden Sie unter. SMART auf FHIR OAuth 2.0-Bereichen, unterstützt von HealthLake

Senden Sie eine `GET` Anfrage an die Abfrage-URL. Im folgenden Beispiel wird das `curl` Befehlszeilentool verwendet.

SigV4

SigV4-Autorisierung

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

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

SMART on FHIR

SMART bei FHIR-Autorisierung. Das Autorisierungstoken muss `read` Berechtigungen für den `Transaction` Ressourcentyp enthalten.

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

In der folgenden Tabelle werden die möglichen Antworten beschrieben.

Antwortcodes für Abfragen

HTTP-Status	Bedeutung	Antworttext
202 Akzeptiert	Die Transaktion befindet sich in der Warteschlange	`OperationOutcome` mit der Diagnose „EINGEREICHT“
202 Akzeptiert	Die Transaktion wird bearbeitet	`OperationOutcome` mit der Diagnose „IN_PROGRESS“
200 OK	Die Transaktion wurde erfolgreich abgeschlossen	`Bundle`mit Typ `transaction-response`
4xx/5xx	Die Transaktion ist fehlgeschlagen	`OperationOutcome` mit Fehlerdetails

Die folgenden Beispiele zeigen die einzelnen Antworttypen.

Transaktion in Warteschlange gestellt (202)

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

Transaktionsverarbeitung (202)

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

Transaktion abgeschlossen (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"
            }
        }
    ]
}
```

Die Transaktion ist fehlgeschlagen (4xx/5xx)

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

Bestellung wird bearbeitet

Asynchrone Pakete des Typs `transaction` befinden sich in der Warteschlange, werden aber nicht in strikter Reihenfolge verarbeitet. HealthLake optimiert die Verarbeitung auf der Grundlage der verfügbaren Kapazität und der Systemlast.

> **Important**
>
> Verlassen Sie sich nicht darauf, dass Transaktionen in der Reihenfolge verarbeitet werden, in der sie eingereicht wurden. Wenn Sie beispielsweise Transaktion A um 10:00 Uhr und Transaktion B um 10:01 Uhr einreichen, könnte Transaktion B vor Transaktion A abgeschlossen werden. Gestalten Sie Ihre Anwendung so, dass sie:
>
> - Erledigen Sie den out-of-order Abschluss.
> - Verwenden Sie die Abfrage-URL, um jede Transaktion unabhängig zu verfolgen.

- Implementieren Sie eine Sequenzierung auf Anwendungsebene, wenn die Reihenfolge für Ihren Anwendungsfall von Bedeutung ist.

Kontingente und Drosselung

Die folgenden Kontingente und Ratenbegrenzungen gelten für asynchrone Transaktionen.

Kontingente für asynchrone Transaktionen

Kontingent	Wert	Einstellbar
Maximale Anzahl von Vorgängen pro asynchroner Transaktion	500	Nein
Maximale Anzahl ausstehender Transaktionen pro Datenspeicher	500	Ja

- Für asynchrone Transaktionen gelten dieselben API-Ratenlimits, die unter Servicekontingente definiert sind.
- Für Abfragen des Transaktionsstatus gelten dieselben API-Ratenbegrenzungen wie für Read (`GET`) -Operationen auf FHIR-Ressourcen.
- Wenn das Limit für ausstehende Transaktionen erreicht ist, wird bei nachfolgenden Einsendungen ein Fehler zurückgegeben, bis die vorhandenen Transaktionen abgeschlossen sind.

Fehlerbehandlung

Bei einem „Transaction“ -Bundle werden alle im Paket enthaltenen FHIR-Ressourcen als atomare Operation verarbeitet. Alle Ressourcen in der Operation müssen erfolgreich sein, oder es werden keine Operationen im Paket verarbeitet.

Fehler lassen sich in zwei Kategorien einteilen: Übertragungsfehler, die synchron HealthLake zurückgegeben werden, und Verarbeitungsfehler, die Sie durch Abfragen abrufen.

Fehler bei der Übermittlung

HealthLake validiert das Paket bei der Übermittlung und gibt Fehler synchron zurück, bevor die Transaktion in die Warteschlange gestellt wird. Zu den Übermittlungsfehlern gehören Fehler bei der

Validierung ungültiger FHIR-Ressourcen, nicht unterstützte Ressourcentypen, Überschreitung der Obergrenze von 500 Vorgängen und die Verwendung des `Prefer: respond-async` Headers mit Batch-Bundles. Wenn das Limit für ausstehende Transaktionen für den Datenspeicher erreicht wurde, HealthLake wird a zurückgegeben. `ThrottlingException` Wenn ein Übertragungsfehler auftritt, wird die Transaktion nicht in die Warteschlange gestellt.

Fehler bei der Verarbeitung

Verarbeitungsfehler treten auf, nachdem die Transaktion in die Warteschlange gestellt wurde, und werden über die Abfrage-URL zurückgegeben. Dazu gehören Transaktionskonflikte, bei denen eine Ressource, die Teil der Transaktion ist, durch einen anderen Vorgang geändert wurde, und Serverfehler während der Verarbeitung. Wenn ein Verarbeitungsfehler auftritt, werden keine Ressourcenmutationen für Ressourcen in der Transaktion vorgenommen. Die Abfrage-URL gibt eine `OperationOutcome` mit den Fehlerdetails zurück.

Löschen einer FHIR-Ressource

Die `delete` FHIR-Interaktion entfernt eine vorhandene FHIR-Ressource aus einem HealthLake Datenspeicher. Weitere Informationen finden Sie `delete`in der FHIR RESTful R4-API-Dokumentation.

Um eine FHIR-Ressource zu löschen

1. Sammeln HealthLake `region` und bewertendatastoreId. Weitere Informationen finden Sie unter Eigenschaften des Datenspeichers abrufen.
2. Ermitteln Sie den FHIR-Typ, der gelöscht `Resource` werden soll, und erfassen Sie den zugehörigen `id` Wert. Weitere Informationen finden Sie unter Ressourcentypen.
3. Konstruieren Sie mithilfe der gesammelten Werte für HealthLake `region` und `datastoreId` eine URL für die Anfrage. Geben Sie auch den `Resource` FHIR-Typ und den zugehörigen `id` Typ an. Scrollen Sie über die Schaltfläche Kopieren, um den gesamten URL-Pfad im folgenden Beispiel anzuzeigen.

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

4. Senden Sie die Anforderung . Die `delete` FHIR-Interaktion verwendet eine `DELETE` Anfrage mit entweder AWS Signature Version 4 oder SMART bei der FHIR-Autorisierung. Im folgenden `curl`

Beispiel wird eine vorhandene `Patient` FHIR-Ressource aus einem HealthLake Datenspeicher entfernt. Scrollen Sie über die Schaltfläche Kopieren, um das gesamte Beispiel anzuzeigen.

SigV4

SigV4-Autorisierung

```
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'
```

Der Server gibt einen `204` HTTP-Statuscode zurück, der bestätigt, dass die Ressource aus dem HealthLake Datenspeicher entfernt wurde. Wenn eine Löschanforderung fehlschlägt, erhalten Sie eine `400` Reihe von HTTP-Statuscodes, die angeben, warum die Anforderung fehlgeschlagen ist.

SMART on FHIR

SMART on FHIR-Autorisierungsbeispiel für den `IdentityProviderConfiguration`Datentyp.

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

Der Anrufer kann im Autorisierungs-Lambda Berechtigungen zuweisen. Weitere Informationen finden Sie unter OAuth 2.0-Bereiche.

AWS Console

1. Melden Sie sich auf der Seite „Abfrage ausführen“ in der HealthLake Konsole an.
2. Treffen Sie im Abschnitt Abfrageeinstellungen die folgenden Optionen.
   - Datenspeicher-ID — Wählen Sie eine Datenspeicher-ID aus, um eine Abfragezeichenfolge zu generieren.
   - Abfragetyp — wählen Sie`Delete`.
   - Ressourcentyp — Wählen Sie den zu löschenden FHIR-Ressourcentyp aus.
   - Ressourcen-ID — geben Sie die FHIR-Ressourcen-ID ein.
3. Wählen Sie Abfrage ausführen.

Löschen von FHIR-Ressourcen auf der Grundlage von Bedingungen

Bedingtes Löschen ist besonders nützlich, wenn Sie die spezifische FHIR-Ressourcen-ID nicht kennen, aber über andere identifizierende Informationen zu der Ressource verfügen, die Sie löschen möchten.

Mit dem bedingten Löschen können Sie eine vorhandene Ressource anhand von Suchkriterien und nicht anhand der logischen FHIR-ID löschen. Wenn der Server die Löschanforderung verarbeitet, führt er eine Suche mithilfe von Standardsuchfunktionen für den Ressourcentyp durch, um eine einzelne logische ID für die Anforderung aufzulösen.

So funktioniert bedingtes Löschen

Die Aktion des Servers hängt davon ab, wie viele Treffer er findet:

1. Keine Treffer: Der Server versucht einen normalen Löschvorgang und reagiert entsprechend (404 Not Found für eine nicht existierende Ressource, 204 Kein Inhalt für bereits gelöschte Ressource)

2. Eine Übereinstimmung: Der Server führt eine normale Löschung der entsprechenden Ressource durch
3. Mehrere Treffer: Gibt den Fehler 412 Precondition Failed zurück, der darauf hinweist, dass die Kriterien des Clients nicht selektiv genug waren

Reaktionsszenarien

AWS HealthLake verarbeitet bedingte Löschvorgänge mit den folgenden Antwortmustern:

Erfolgreiche Operationen

- Wenn Ihre Suchkriterien erfolgreich eine einzelne aktive Ressource identifizieren, gibt das System nach Abschluss des Löschvorgangs 204 Kein Inhalt zurück, genau wie bei Standardlöschvorgängen.

ID-basiertes bedingtes Löschen

Beim Ausführen eines bedingten Löschvorgangs auf der `id` Grundlage zusätzlicher Parameter (`createdAt`,`tag`, oder`_lastUpdated`):

- 204 Kein Inhalt: Die Ressource wurde bereits gelöscht
- 404 Nicht gefunden: Die Ressource existiert nicht
- 409 Konflikt: Die ID stimmt überein, aber andere Parameter stimmen nicht überein

Non-ID-Based Bedingtes Löschen

Wenn nicht `id` angegeben oder wenn andere Parameter verwendet werden als`createdAt`,`tag`, oder`_lastUpdated`:

- 404 Nicht gefunden: Keine Treffer gefunden

Konfliktsituationen

Verschiedene Szenarien führen zu den Antworten 412 Precondition Failed:

- Mehrere Ressourcen entsprechen Ihren Suchkriterien (die Kriterien sind nicht spezifisch genug)
- Versionskonflikte bei der Verwendung von ETag Headern mit `If-Match`

- Ressourcenaktualisierungen, die zwischen Such- und Löschvorgängen auftreten

Beispiel für ein erfolgreiches bedingtes Löschen

Im folgenden Beispiel wird eine Patientenressource anhand bestimmter Kriterien gelöscht:

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

Mit dieser Anfrage wird eine Patientenressource gelöscht, wobei:

- Der Name ist „Peter“
- Geburtsdatum ist der 1. Januar 2000
- Die Telefonnummer ist 1234567890

Bewährte Methoden

1. Verwenden Sie spezifische Suchkriterien, um Mehrfachübereinstimmungen und 412-Fehler zu vermeiden.
2. Ziehen Sie ETag Header für die Versionskontrolle in Betracht, wenn dies für gleichzeitige Änderungen erforderlich ist.
3. Gehen Sie angemessen mit Fehlerreaktionen um:
   - Für 404: Verfeinern Sie Ihre Suchkriterien
   - Für 412: Spezifizieren Sie die Kriterien oder lösen Sie Versionskonflikte
4. Bereiten Sie sich auf Zeitkonflikte in Umgebungen mit hoher Parallelität vor, in denen Ressourcen zwischen Such- und Löschvorgängen geändert werden können.

Idempotenz und Parallelität

Idempotenzschlüssel

AWS HealthLake unterstützt Idempotenzschlüssel für `POST` FHIR-Operationen und bietet so einen robusten Mechanismus zur Sicherstellung der Datenintegrität bei der Ressourcenerstellung. Durch die Aufnahme einer eindeutigen UUID als Idempotenzschlüssel in den Anforderungsheader können

Anwendungen im Gesundheitswesen garantieren, dass jede FHIR-Ressource genau einmal erstellt wird, selbst in Szenarien mit Netzwerkinstabilität oder automatischen Wiederholungsversuchen.

Diese Funktion ist besonders wichtig für Gesundheitssysteme, in denen doppelte Patientenakten schwerwiegende Folgen haben könnten. Wenn eine Anfrage mit demselben Idempotenzschlüssel wie eine vorherige Anfrage eingeht, HealthLake wird die ursprüngliche Ressource zurückgegeben, anstatt ein Duplikat zu erstellen. Dies kann beispielsweise während einer Wiederholungsschleife oder aufgrund redundanter Anforderungspipelines auftreten. Die Verwendung des Idempotenzschlüssels ermöglicht HealthLake die Aufrechterhaltung der Datenkonsistenz und bietet gleichzeitig ein reibungsloses Erlebnis für Client-Anwendungen, die mit zeitweiligen Verbindungsproblemen umgehen.

Implementierung

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

Reaktionsszenarien

Erste Anfrage (201 erstellt)

- Neue Ressource wurde erfolgreich erstellt
- Die Antwort enthält die Ressourcen-ID

Doppelte Anfrage (409-Konflikt)

- Derselbe Idempotenzschlüssel erkannt
- Die ursprüngliche Ressource wurde zurückgegeben
- Es wurde keine neue Ressource erstellt

Ungültige Anfrage (400 Schlechte Anfrage)

- Fehlerhafte UUID
- Fehlende Pflichtfelder

Bewährte Methoden

- Generieren Sie eine eindeutige UUID für jede neue Ressourcenerstellung

- Speichern Sie Idempotenzschlüssel für die Wiederholungslogik
- Verwenden Sie ein konsistentes Schlüsselformat: UUID v4 wird empfohlen
- Implementieren Sie es in Client-Anwendungen, die die Ressourcenerstellung übernehmen

> **Note**
>
> Diese Funktion ist besonders nützlich für Gesundheitssysteme, die eine strikte Datengenauigkeit erfordern und doppelte Patientenakten verhindern müssen.

ETag in AWS HealthLake

AWS HealthLake dient ETags zur optimistischen Kontrolle der Parallelität in FHIR-Ressourcen und bietet so einen zuverlässigen Mechanismus zur Verwaltung gleichzeitiger Änderungen und zur Wahrung der Datenkonsistenz. An ETag ist ein eindeutiger Bezeichner, der für eine bestimmte Version einer Ressource steht und über HTTP-Header als Versionskontrollsystem fungiert. Beim Lesen oder Ändern von Ressourcen können ETags Anwendungen unbeabsichtigte Überschreibungen verhindern und die Datenintegrität sicherstellen, insbesondere in Szenarien mit potenziellen gleichzeitigen Aktualisierungen.

Beispiel für eine Implementierung

```
// 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
```

Reaktionsszenarien

Erfolgreicher Vorgang (200 OK oder 204 Kein Inhalt)

- ETag entspricht der aktuellen Version
- Der Vorgang wird wie vorgesehen fortgesetzt

Versionskonflikt (412-Vorbedingung fehlgeschlagen)

- ETag entspricht nicht der aktuellen Version
- Das Update wurde abgelehnt, um Datenverlust zu verhindern

Bewährte Methoden

- ETags In alle Aktualisierungs- und Löschvorgänge einbeziehen
- Implementieren Sie die Wiederholungslogik zur Behandlung von Versionskonflikten
- Verwenden Sie If-None-Match: * für Szenarien create-if-not-exists
- Überprüfen Sie immer die ETag Frische, bevor Sie Änderungen vornehmen

Dieses System zur Kontrolle der Parallelität ist für die Aufrechterhaltung der Integrität von Gesundheitsdaten unerlässlich, insbesondere in Umgebungen, in denen mehrere Benutzer oder Systeme auf dieselben Ressourcen zugreifen und diese ändern.

Suche nach FHIR-Ressourcen in AWS HealthLake

Verwenden Sie die `search`FHIR-Interaktion, um anhand einiger Filterkriterien nach einer Reihe von FHIR-Ressourcen in einem HealthLake Datenspeicher zu suchen. Die `search` Interaktion kann entweder mit einer `GET` Oder-Anfrage durchgeführt werden. `POST` Für Suchanfragen, die personenbezogene Daten (PII) oder geschützte Gesundheitsinformationen (PHI) beinhalten, wird empfohlen, `POST` Anfragen zu verwenden, da PII und PHI als Teil des Anfragetextes hinzugefügt und bei der Übertragung verschlüsselt werden.

> **Note**
>
> Die in diesem Kapitel beschriebene `search` FHIR-Interaktion entspricht dem HL7 FHIR R4-Standard für den Austausch von Gesundheitsdaten. Da es sich um eine Repräsentation eines HL7 FHIR-Dienstes handelt, wird er nicht über und angeboten. AWS CLI AWS SDKs Weitere Informationen finden Sie `search`in der FHIR RESTful R4-API-Dokumentation.

HealthLake unterstützt eine Teilmenge der FHIR R4-Suchparameter. Weitere Informationen finden Sie unter FHIR R4-Suchparameter für HealthLake.

Themen

- FHIR-Ressourcen mit GET durchsuchen
- FHIR-Ressourcen mit POST durchsuchen
- Konsistenzstufen für die FHIR-Suche

FHIR-Ressourcen mit GET durchsuchen

Sie können `GET` Anfragen verwenden, um einen HealthLake Datenspeicher zu durchsuchen. HealthLake Unterstützt bei der Verwendung `GET` die Angabe von Suchparametern als Teil der URL, jedoch nicht als Teil eines Anforderungstexts. Weitere Informationen finden Sie unter FHIR R4-Suchparameter für HealthLake.

> **Wichtig**
>
> Bei Suchanfragen, die personenbezogene Daten (PII) oder geschützte Gesundheitsinformationen (PHI) beinhalten, empfehlen bewährte Sicherheitsverfahren die

Verwendung von POST Anfragen, da PII und PHI als Teil des Anfragetexts hinzugefügt und bei der Übertragung verschlüsselt werden. Weitere Informationen finden Sie unter FHIR-Ressourcen mit POST durchsuchen.

Auf das folgende Verfahren folgen Beispiele, die GET zum Durchsuchen eines HealthLake Datenspeichers verwendet werden.

Um einen HealthLake Datenspeicher zu durchsuchen mit **GET**

1. Sammeln HealthLake `region` und `datastoreId` Werte. Weitere Informationen finden Sie unter Eigenschaften des Datenspeichers abrufen.
2. Ermitteln Sie den Typ der FHIR-Ressource, nach der gesucht werden soll, und erfassen Sie den zugehörigen `id` Wert. Weitere Informationen finden Sie unter Ressourcentypen.
3. Konstruieren Sie mithilfe der gesammelten Werte für HealthLake `region` und `datastoreId` eine URL für die Anfrage. Geben Sie auch den `Resource` FHIR-Typ und die unterstützten Suchparameter an. Scrollen Sie über die Schaltfläche Kopieren, um den gesamten URL-Pfad im folgenden Beispiel anzuzeigen.

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

4. Senden Sie die GET Anfrage entweder mit AWS Signature Version 4 oder SMART bei FHIR-Autorisierung. Das folgende `curl` Beispiel gibt die Gesamtzahl der `Patient` Ressourcen in einem HealthLake Datenspeicher zurück. Scrollen Sie über die Schaltfläche Kopieren, um das gesamte Beispiel anzuzeigen.

   SigV4

   SigV4-Autorisierung

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

SMART on FHIR

SMART on FHIR-Autorisierungsbeispiel für den `IdentityProviderConfiguration`Datentyp.

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

Der Anrufer kann im Autorisierungs-Lambda Berechtigungen zuweisen. Weitere Informationen finden Sie unter OAuth 2.0-Bereiche.

AWS Console

> **Note**
>
> Die HealthLake Konsole unterstützt nur die SigV4-Autorisierung. Die SMART-on-FHIR-Autorisierung wird über AWS CLI und unterstützt. AWS SDKs

1. Melden Sie sich auf der Seite „Abfrage ausführen“ in der HealthLake Konsole an.

2. Treffen Sie im Abschnitt Abfrageeinstellungen die folgenden Optionen.

- Datenspeicher-ID — Wählen Sie eine Datenspeicher-ID aus, um eine Abfragezeichenfolge zu generieren.
- Abfragetyp — wählen Sie`Search with GET`.
- Ressourcentyp — Wählen Sie den FHIR-Ressourcentyp aus, nach dem gesucht werden soll.
- Suchparameter — Wählen Sie einen Suchparameter oder eine Kombination von Suchparametern aus, um Ihre Anfrage auf bestimmte Datensätze zu konzentrieren.

3. Wählen Sie Abfrage ausführen.

Beispiele: Suche mit GET

Die folgenden Registerkarten enthalten Beispiele für die Suche nach bestimmten FHIR-Ressourcentypen mit. `GET` Die Beispiele zeigen, wie Suchparameter in der Anfrage URLs angegeben werden.

> **Note**
>
> Die HealthLake Konsole unterstützt nur die SigV4-Autorisierung. Die SMART-on-FHIR-Autorisierung wird über AWS CLI und unterstützt. AWS SDKs
> HealthLake unterstützt eine Teilmenge der FHIR R4-Suchparameter. Weitere Informationen finden Sie unter Suchparameter.

Patient (age)

Obwohl das Alter in FHIR kein definierter Ressourcentyp ist, wird es als Element im Ressourcentyp erfasst. `Patient` Verwenden Sie das folgende Beispiel, um eine auf `Patient`Ressourcentypen `GET` basierende Suchanfrage zu stellen, indem Sie das Element BirthDate und den `eq` Suchkomparator verwenden, um nach Personen zu suchen, die im Jahr 1997 geboren wurden.

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

Condition

Verwenden Sie das folgende Beispiel, um eine GET Anfrage für den Condition Ressourcentyp zu stellen. Bei der Suche werden Bedingungen in Ihrem HealthLake Datenspeicher gefunden, die den medizinischen Code von SNOMED enthalten72892002, was übersetzt bedeutet. Normal pregnancy

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

DocumentationReference

Das folgende Beispiel zeigt, wie eine GET Anfrage für den DocumentReference Ressourcentyp für Patient (s) erstellt wird, bei denen eine Streptokokken-Diagnose gestellt wurde und denen auch Amoxicillin verschrieben wurde.

```
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

Verwenden Sie das folgende Beispiel, um eine GET Anfrage für den Ressourcentyp zu stellen. Location Mit der folgenden Suche werden Standorte in Ihrem HealthLake Datenspeicher gefunden, die den Stadtnamen Boston als Teil der Adresse enthalten.

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

Observation

Verwenden Sie das folgende Beispiel, um eine GET basierte Suchanfrage für den ObservationRessourcentyp zu stellen. Diese Suche verwendet den value-concept Suchparameter, um nach medizinischem Code zu suchen266919005, was übersetzt bedeutet. Never smoker

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

FHIR-Ressourcen mit POST durchsuchen

Sie können die `search`FHIR-Interaktion mit `POST` Anfragen verwenden, um einen HealthLake Datenspeicher zu durchsuchen. Bei der Verwendung `POST` werden Suchparameter entweder in der URL oder in einem Anfragetext HealthLake unterstützt, Sie können jedoch nicht beide in einer einzigen Anfrage verwenden.

> **Wichtig**
>
> Bei Suchanfragen, die personenbezogene Daten (PII) oder geschützte Gesundheitsinformationen (PHI) beinhalten, empfehlen bewährte Sicherheitsverfahren die Verwendung von `POST` Anfragen, da PII und PHI als Teil des Anfragetextes hinzugefügt und bei der Übertragung verschlüsselt werden.

Auf das folgende Verfahren folgen Beispiele, bei denen die `search` Interaktion mit FHIR R4 `POST` zum Durchsuchen eines Datenspeichers verwendet wird. HealthLake Die Beispiele zeigen, wie Suchparameter im JSON-Anforderungstext angegeben werden.

Um einen HealthLake Datenspeicher zu durchsuchen mit **`POST`**

1. Sammeln HealthLake `region` und `datastoreId` Werte. Weitere Informationen finden Sie unter Eigenschaften des Datenspeichers abrufen.
2. Ermitteln Sie den Typ der FHIR-Ressource, nach der gesucht werden soll, und erfassen Sie den zugehörigen `id` Wert. Weitere Informationen finden Sie unter Ressourcentypen.
3. Konstruieren Sie mithilfe der gesammelten Werte für HealthLake `region` und `datastoreId` eine URL für die Anfrage. Geben Sie auch den `Resource` FHIR-Typ und die `_search` Interaktion an. Scrollen Sie über die Schaltfläche Kopieren, um den gesamten URL-Pfad im folgenden Beispiel anzuzeigen.

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

4. Konstruieren Sie einen JSON-Hauptteil für die Anfrage und geben Sie die FHIR-Daten an, nach denen gesucht werden soll. Für dieses Verfahren suchen Sie nach `Observation` Ressourcen, um Patienten zu finden, die noch nie geraucht haben. Um den Status des medizinischen

Codes anzugebenNever smoker, legen Sie ihn value-concept=266919005 im JSON-Anforderungstext fest. Speichern Sie die Datei als search-observation.json.

```
value-concept=266919005
```

5. Senden Sie die Anforderung . Die search FHIR-Interaktion verwendet die GET Anfrage entweder mit AWS Signature Version 4 oder SMART bei der FHIR-Autorisierung.

> **Note**
>
> Wenn POST Sie eine Anfrage mit Suchparametern im Anfragetext stellen, verwenden Sie diese Option Content-Type: application/x-www-form-urlencoded als Teil des Headers.

Im folgenden curl Beispiel wird eine POST-basierte Suchanfrage für den Observation Ressourcentyp gestellt. Die Anfrage verwendet den value-conceptSuchparameter, um nach medizinischem Code zu suchen, der einen Wert 266919005 Never smoker angibt. Scrollen Sie über die Schaltfläche Kopieren, um das gesamte Beispiel anzuzeigen.

SigV4

SigV4-Autorisierung

```
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

Beispiel für die SMART-on-FHIR-Autorisierung für den IdentityProviderConfigurationDatentyp.

```
{
```

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

Der Anrufer kann im Autorisierungs-Lambda Berechtigungen zuweisen. Weitere Informationen finden Sie unter OAuth 2.0-Bereiche.

Beispiele: Suche mit POST

Die folgenden Registerkarten enthalten Beispiele für die Suche nach bestimmten FHIR-Ressourcentypen mit. `POST` Die Beispiele zeigen, wie Sie eine Anfrage in der URLs spezifizieren.

> **Note**
>
> Die HealthLake Konsole unterstützt nur die SigV4-Autorisierung. Die SMART-on-FHIR-Autorisierung wird über AWS CLI und unterstützt. AWS SDKs
> HealthLake unterstützt eine Teilmenge der FHIR R4-Suchparameter. Weitere Informationen finden Sie unter Suchparameter.

Patient (age)

Obwohl das Alter in FHIR kein definierter Ressourcentyp ist, wird es als Element im Ressourcentyp erfasst. `Patient` Verwenden Sie das folgende Beispiel, um eine `POST` basierte

Suchanfrage für den `Patient` Ressourcentyp zu stellen. Im folgenden Suchbeispiel wird der `eq` Suchkomparator verwendet, um nach Personen zu suchen, die 1997 geboren wurden.

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

Um das Jahr 1997 in der Suche anzugeben, fügen Sie dem Anfragetext das folgende Element hinzu.

```
birthdate=eq1997
```

Condition

Verwenden Sie Folgendes, um eine `POST` Anfrage für den `Condition` Ressourcentyp zu stellen. Bei dieser Suche werden Stellen in Ihrem HealthLake Datenspeicher gefunden, die den medizinischen Code enthalten`72892002`.

Sie müssen eine Anfrage-URL und einen Anfragetext angeben. Hier ist ein Beispiel für eine Anfrage-URL.

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

Um den medizinischen Code anzugeben, den Sie durchsuchen möchten, fügen Sie dem Anfragetext das folgende JSON-Element hinzu.

```
code=72892002
```

DocumentReference

Um die Ergebnisse der HealthLake integrierten Verarbeitung natürlicher Sprache (NLP) zu sehen, wenn Sie eine `POST` Anfrage für den `DocumentReference` Ressourcentyp stellen, formatieren Sie eine Anfrage wie folgt.

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

Informationen zur Angabe der `DocumentReference` Suchparameter, auf die verwiesen werden soll, finden Sie unterSuchparametertypen. Die folgende Abfragezeichenfolge verwendet mehrere

Suchparameter für die Suche nach Amazon Comprehend Medical API-Operationen, die zur Generierung der integrierten NLP-Ergebnisse verwendet werden.

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

Location

Verwenden Sie das folgende Beispiel, um eine `POST` Anfrage für den Ressourcentyp zu stellen. `Location` Bei der Suche werden Standorte in Ihrem HealthLake Datenspeicher gefunden, die den Stadtnamen Boston als Teil der Adresse enthalten.

Sie müssen eine Anfrage-URL und einen Anfragetext angeben. Hier ist ein Beispiel für eine Anfrage-URL.

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

Um dies `Boston` in der Suche zu spezifizieren, fügen Sie dem Anfragetext das folgende Element hinzu:

```
address=Boston
```

Observation

Verwenden Sie das folgende Beispiel, um eine `POST` basierte Suchanfrage für den `Observation` Ressourcentyp zu stellen. Die Suche verwendet den `value-concept` Suchparameter, um nach medizinischem Code zu suchen, der `266919005` darauf hinweist`Never smoker`.

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

Um den Status anzugeben`Never smoker`, der `value-concept=266919005` im Hauptteil der JSON-Datei festgelegt wird.

```
value-concept=266919005
```

Konsistenzstufen für die FHIR-Suche

HealthLakeDer Suchindex von AWS basiert auf einem Eventual Consistency Model für `GET` und `POST` mit SEARCH-Vorgängen. Bei eventueller Konsistenz schließen die Suchergebnisse Version N-1 der Ressource aus, wenn ein Suchindex-Update für eine Ressource aussteht, bis die Indexaktualisierung abgeschlossen ist.

AWS bietet HealthLake jetzt die Möglichkeit, auszuwählen, wie sich das Konsistenzmodell für aktualisierte Ressourcen verhalten soll. Entwickler können entweder „Eventual Consistency“, das oben beschriebene Standardverhalten, oder „Strong Consistency“ angeben. Strong Consistency ermöglicht es, die N-1-Version der Ressource für Ressourcen mit ausstehenden Aktualisierungen des Suchindexes in die Suchergebnisse aufzunehmen. Dies kann für Anwendungsszenarien verwendet werden, in denen alle Ressourcen für das Ergebnis benötigt werden, auch wenn die Aktualisierung des Suchindex noch nicht abgeschlossen ist. Kunden können dieses Verhalten mithilfe des `x-amz-fhir-history-consistency-level` Anforderungsheaders steuern.

Konsistenzstufen

Starke Konsistenz

: Legt fest`x-amz-fhir-history-consistency-level: strong`, dass alle übereinstimmenden Datensätze zurückgegeben werden, auch solche mit ausstehenden Aktualisierungen des Suchindexes. Verwenden Sie diese Option, wenn Sie unmittelbar nach Aktualisierungen nach Ressourcen suchen müssen.

Letztendliche Datenkonsistenz

: Legt fest`x-amz-fhir-history-consistency-level: eventual`, dass nur Datensätze zurückgegeben werden, für die die Aktualisierung des Suchindex abgeschlossen wurde. Dies ist das Standardverhalten, wenn keine Konsistenzstufe angegeben ist.

Verwendungsbeispiel

1. Gehen Sie beim Aktualisieren einer Ressource wie folgt vor:

```
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. Nachfolgende Suche:

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

Best Practices

- Verwenden Sie starke Konsistenz, wenn Sie sofort nach kürzlich aktualisierten Ressourcen suchen müssen
- Verwenden Sie Eventual-Konsistenz für allgemeine Abfragen, bei denen sofortige Sichtbarkeit nicht entscheidend ist
- Berücksichtigen Sie den Kompromiss zwischen sofortiger Sichtbarkeit und potenziellen Auswirkungen auf die Leistung

> **Note**
>
> Die Einstellung der Konsistenzstufe wirkt sich darauf aus, wie schnell aktualisierte Ressourcen in den Suchergebnissen angezeigt werden, hat jedoch keinen Einfluss auf die tatsächliche Speicherung der Ressourcen.
> Wenn der optionale `x-amz-fhir-history-consistency-level` Header auf „strong“ gesetzt wird, verdoppelt sich der Schreibkapazitätsverbrauch pro Ressource.
> Diese Funktion ist nur für Datenspeicher verfügbar, bei denen der Versionsverlauf aktiviert ist (bei allen Datenspeichern, die nach dem 25. Oktober 2024 erstellt wurden, ist sie standardmäßig aktiviert).

Exportieren von FHIR-Daten mit AWS HealthLake

Verwenden Sie native AWS HealthLake Aktionen, um FHIR-Exportaufträge zu starten, zu beschreiben und aufzulisten. Sie können Exportaufträge in eine Warteschlange stellen. Die asynchronen Exportaufträge werden nach dem FIFO-Verfahren (First In First Out) verarbeitet. Sie können Jobs genauso in die Warteschlange stellen, wie Sie Exportaufträge starten. Wenn gerade einer in Bearbeitung ist, wird er einfach in die Warteschlange gestellt. Sie können FHIR-Ressourcen erstellen, lesen, aktualisieren oder löschen, während ein Exportauftrag ausgeführt wird.

> **Note**
>
> Sie können FHIR-Daten auch mithilfe der FHIR R4-Operation aus einem HealthLake Datenspeicher exportieren. `$export` Weitere Informationen finden Sie unter HealthLake Daten mit FHIR exportieren `$export`.

Themen

- Einen FHIR-Exportjob starten
- Eigenschaften von FHIR-Exportaufträgen abrufen
- Auflisten von FHIR-Exportaufträgen

Einen FHIR-Exportjob starten

Wird verwendet`StartFHIRExportJob`, um einen FHIR-Exportauftrag aus einem HealthLake Datenspeicher zu starten. Die folgenden Menüs enthalten eine Vorgehensweise für das AWS-Managementkonsole und Codebeispiele für AWS CLI und AWS SDKs. Weitere Informationen finden Sie unter `StartFHIRExportJob` in der AWS HealthLake -API-Referenz.

> **Hinweis**
>
> HealthLake unterstützt die FHIR R4-Spezifikation für den Austausch von Gesundheitsdaten. Daher werden alle Gesundheitsdaten im FHIR R4-Format exportiert.

So starten Sie einen FHIR-Exportjob

Wählen Sie ein Menü, das Ihren Zugriffspräferenzen für entspricht. AWS HealthLake

AWS CLI und SDKs

CLI

AWS CLI

So starten Sie einen FHIR-Exportjob

Das folgende `start-fhir-export-job` Beispiel zeigt, wie Sie einen FHIR-Exportauftrag mit AWS HealthLake starten.

```
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)
```

Ausgabe:

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

- API-Einzelheiten finden Sie unter FHIRExportJob starten in der AWS CLI Befehlsreferenz.

Python

SDK für 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
```

- Einzelheiten zur API finden Sie unter Start FHIRExport Job in AWS SDK for Python (Boto3) API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

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

- API-Details finden Sie unter Referenz zum Starten FHIRExport eines Job im AWS SDK für SAP ABAP API.

> **Beispiel für die Verfügbarkeit**
>
> Sie können nicht finden, was Sie brauchen? Fordern Sie über den Link Feedback geben in der rechten Seitenleiste dieser Seite ein Codebeispiel an.

AWS Konsole

1. Melden Sie sich auf der Seite Datenspeicher in der HealthLake Konsole an.
2. Wählen Sie einen Datenspeicher aus.
3. Wählen Sie Export aus.

   Die Exportseite wird geöffnet.
4. Geben Sie im Abschnitt Ausgabedaten die folgenden Informationen ein:

   - Speicherort der Ausgabedaten in Amazon S3
   - Verschlüsselung der Ausgabe

5. Wählen Sie im Abschnitt Zugriffsberechtigungen die Option Bestehende IAM-Dienstrolle verwenden und wählen Sie die Rolle im Menü Rollenname aus, oder wählen Sie Create an IAM-Rolle aus.
6. Wählen Sie Daten exportieren aus.

> **Note**
>
> Wählen Sie beim Export im Banner oben auf der Seite die Option Job-ID kopieren aus. Sie können die verwenden `JobID`, um Exportauftragseigenschaften über die anzufordern AWS CLI. Weitere Informationen finden Sie unter Eigenschaften von FHIR-Exportaufträgen abrufen.

Eigenschaften von FHIR-Exportaufträgen abrufen

Wird verwendet`DescribeFHIRExportJob`, um Eigenschaften von Exportaufträgen aus einem HealthLake Datenspeicher abzurufen. Die folgenden Menüs enthalten eine Vorgehensweise für das AWS-Managementkonsole und Codebeispiele für AWS CLI und AWS SDKs. Weitere Informationen finden Sie unter `DescribeFHIRExportJob` in der AWS HealthLake -API-Referenz.

> **Hinweis**
>
> HealthLake unterstützt die FHIR R4-Spezifikation für den Austausch von Gesundheitsdaten. Daher werden alle Gesundheitsdaten im FHIR R4-Format exportiert.

So beschreiben Sie einen FHIR-Exportjob

Wählen Sie ein Menü, das Ihren Zugriffspräferenzen für entspricht. AWS HealthLake

AWS CLI und SDKs

CLI

AWS CLI

So beschreiben Sie einen FHIR-Exportjob

Das folgende `describe-fhir-export-job` Beispiel zeigt, wie Sie die Eigenschaften eines FHIR-Exportauftrags in AWS HealthLake finden.

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

Ausgabe:

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

- Einzelheiten zur API finden Sie unter FHIRExportJob beschreiben in der AWS CLI Befehlsreferenz.

Python

SDK für 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
```

- Einzelheiten zur API finden Sie unter FHIRExportDescribe Job in AWS SDK for Python (Boto3) API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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.
```

- Einzelheiten zur API finden Sie unter FHIRExportDescribe Job in AWS SDK for SAP ABAP API-Referenz.

> **Beispiel für die Verfügbarkeit**
>
> Sie können nicht finden, was Sie brauchen? Fordern Sie über den Link Feedback geben in der rechten Seitenleiste dieser Seite ein Codebeispiel an.

AWS Konsole

> **Note**
>
> Informationen zum FHIR-Exportauftrag sind auf der HealthLake Konsole nicht verfügbar. Verwenden Sie stattdessen die Option AWS CLI with, `DescribeFHIRExportJob` um Eigenschaften von Exportaufträgen anzufordern, wie `JobStatus`z. Weitere Informationen finden Sie im AWS CLI Beispiel auf dieser Seite.

Auflisten von FHIR-Exportaufträgen

Dient `ListFHIRExportJobs` zum Auflisten von FHIR-Exportaufträgen für einen HealthLake Datenspeicher. Die folgenden Menüs enthalten eine Vorgehensweise für das AWS-Managementkonsole und Codebeispiele für AWS CLI und AWS SDKs. Weitere Informationen finden Sie unter `ListFHIRExportJobs` in der AWS HealthLake -API-Referenz.

> **Hinweis**
>
> HealthLake unterstützt die FHIR R4-Spezifikation für den Austausch von Gesundheitsdaten. Daher werden alle Gesundheitsdaten im FHIR R4-Format exportiert.

Um FHIR-Exportaufträge aufzulisten

Wählen Sie ein Menü, das Ihren Zugriffspräferenzen für entspricht AWS HealthLake.

AWS CLI und SDKs

CLI

AWS CLI

So listen Sie alle FHIR-Exportjobs auf

Das folgende Beispiel für `list-fhir-export-jobs` zeigt, wie Sie mit dem Befehl eine Liste von Exportjobs anzeigen, die mit einem Konto verknüpft sind.

```
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)
```

Ausgabe:

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

```
}
```

- Einzelheiten zur API finden Sie unter FHIRExportAuflisten von Aufträgen in der AWS CLI Befehlsreferenz.

Python

SDK für 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
```

- Einzelheiten zur API finden Sie unter List FHIRExport Jobs in AWS SDK for Python (Boto3) API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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.
```

- Einzelheiten zur API finden Sie in der Referenz zum FHIRExportAuflisten von Aufträgen im AWS SDK für SAP ABAP API.

Beispiel für die Verfügbarkeit

Sie können nicht finden, was Sie brauchen? Fordern Sie über den Link Feedback geben in der rechten Seitenleiste dieser Seite ein Codebeispiel an.

AWS Konsole

Note

Informationen zum FHIR-Exportauftrag sind auf der HealthLake Konsole nicht verfügbar. Verwenden Sie stattdessen das „ AWS CLI with“, `ListFHIRExportJobs` um alle FHIR-Exportaufträge aufzulisten. Weitere Informationen finden Sie im AWS CLI Beispiel auf dieser Seite.

Codebeispiele für die HealthLake Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Verwendung HealthLake mit einem AWS Software Development Kit (SDK) funktioniert.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterVerwendung HealthLake mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Codebeispiele

- Grundlegende Beispiele für die HealthLake Verwendung AWS SDKs
  - Aktionen zur HealthLake Verwendung AWS SDKs
    - Verwendung CreateFHIRDatastore mit einem AWS SDK oder CLI
    - Verwendung DeleteFHIRDatastore mit einem AWS SDK oder CLI
    - Verwendung DescribeFHIRDatastore mit einem AWS SDK oder CLI
    - Verwendung DescribeFHIRExportJob mit einem AWS SDK oder CLI
    - Verwendung DescribeFHIRImportJob mit einem AWS SDK oder CLI
    - Verwendung ListFHIRDatastores mit einem AWS SDK oder CLI
    - Verwendung ListFHIRExportJobs mit einem AWS SDK oder CLI
    - Verwendung ListFHIRImportJobs mit einem AWS SDK oder CLI
    - Verwendung ListTagsForResource mit einem AWS SDK oder CLI
    - Verwendung StartFHIRExportJob mit einem AWS SDK oder CLI
    - Verwendung StartFHIRImportJob mit einem AWS SDK oder CLI
    - Verwendung TagResource mit einem AWS SDK oder CLI
    - Verwendung UntagResource mit einem AWS SDK oder CLI

Grundlegende Beispiele für die HealthLake Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS HealthLake with verwendet AWS SDKs werden.

Beispiele

- Aktionen zur HealthLake Verwendung AWS SDKs
  - Verwendung CreateFHIRDatastore mit einem AWS SDK oder CLI
  - Verwendung DeleteFHIRDatastore mit einem AWS SDK oder CLI
  - Verwendung DescribeFHIRDatastore mit einem AWS SDK oder CLI
  - Verwendung DescribeFHIRExportJob mit einem AWS SDK oder CLI
  - Verwendung DescribeFHIRImportJob mit einem AWS SDK oder CLI
  - Verwendung ListFHIRDatastores mit einem AWS SDK oder CLI
  - Verwendung ListFHIRExportJobs mit einem AWS SDK oder CLI
  - Verwendung ListFHIRImportJobs mit einem AWS SDK oder CLI
  - Verwendung ListTagsForResource mit einem AWS SDK oder CLI
  - Verwendung StartFHIRExportJob mit einem AWS SDK oder CLI
  - Verwendung StartFHIRImportJob mit einem AWS SDK oder CLI
  - Verwendung TagResource mit einem AWS SDK oder CLI
  - Verwendung UntagResource mit einem AWS SDK oder CLI

Aktionen zur HealthLake Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie einzelne HealthLake Aktionen mit ausführen können AWS SDKs. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS HealthLake -API-Referenz.

Beispiele

- Verwendung CreateFHIRDatastore mit einem AWS SDK oder CLI
- Verwendung DeleteFHIRDatastore mit einem AWS SDK oder CLI

- Verwendung DescribeFHIRDatastore mit einem AWS SDK oder CLI
- Verwendung DescribeFHIRExportJob mit einem AWS SDK oder CLI
- Verwendung DescribeFHIRImportJob mit einem AWS SDK oder CLI
- Verwendung ListFHIRDatastores mit einem AWS SDK oder CLI
- Verwendung ListFHIRExportJobs mit einem AWS SDK oder CLI
- Verwendung ListFHIRImportJobs mit einem AWS SDK oder CLI
- Verwendung ListTagsForResource mit einem AWS SDK oder CLI
- Verwendung StartFHIRExportJob mit einem AWS SDK oder CLI
- Verwendung StartFHIRImportJob mit einem AWS SDK oder CLI
- Verwendung TagResource mit einem AWS SDK oder CLI
- Verwendung UntagResource mit einem AWS SDK oder CLI

Verwendung **`CreateFHIRDatastore`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateFHIRDatastore` verwendet wird.

CLI

AWS CLI

Beispiel 1: Erstellen Sie einen SIGV4-fähigen Datenspeicher HealthLake

Das folgende `create-fhir-datastore` Beispiel zeigt, wie Sie einen neuen Datenspeicher in erstellen. AWS HealthLake

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

Ausgabe:

```
{
    "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)"
```

```
}
```

Beispiel 2: Erstellen Sie einen SMART-Datenspeicher auf einem FHIR-fähigen Datenspeicher HealthLake

Das folgende `create-fhir-datastore` Beispiel zeigt, wie Sie einen neuen SMART-on-FHIR-fähigen Datenspeicher in erstellen. 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
```

Inhalt von `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\"]}"
}
```

Ausgabe:

```
{
```

```
    "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)"
}
```

- Einzelheiten zur API finden Sie unter Create FHIRDatastore in AWS CLI Command Reference.

Python

SDK für 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
```

Der folgende Code zeigt ein Beispiel für Parameter für einen SMART-Datenspeicher auf einem FHIR-fähigen Datenspeicher 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
            )
```

- API-Einzelheiten finden Sie unter Create FHIRDatastore in AWS SDK for Python (Boto3) API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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.
```

- Einzelheiten zur API finden Sie unter API-Referenz zum Erstellen FHIRDatastore im AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterVerwendung HealthLake mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`DeleteFHIRDatastore`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteFHIRDatastore` verwendet wird.

CLI

AWS CLI

So löschen Sie einen FHIR-Datenspeicher

Das folgende `delete-fhir-datastore` Beispiel zeigt, wie Sie einen Datenspeicher und seinen gesamten Inhalt löschen AWS HealthLake.

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

Ausgabe:

```
{
    "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)"
}
```

- Einzelheiten zur FHIRDatastoreAWS CLI API finden Sie unter Löschen in der Befehlsreferenz.

Python

SDK für Python (Boto3)

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

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


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

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

- API-Einzelheiten finden Sie unter Delete FHIRDatastore in AWS SDK for Python (Boto3) API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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.
```

- Einzelheiten zur API finden Sie unter Delete FHIRDatastore in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterVerwendung HealthLake mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`DescribeFHIRDatastore`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeFHIRDatastore` verwendet wird.

CLI

AWS CLI

So beschreiben Sie einen FHIR-Datenspeicher

Das folgende `describe-fhir-datastore` Beispiel zeigt, wie Sie die Eigenschaften eines Datenspeichers in finden AWS HealthLake.

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

Ausgabe:

```
{
    "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
```

```
        }
    }
}
```

- Einzelheiten zur API finden Sie unter Beschreibung FHIRDatastore in der AWS CLI Befehlsreferenz.

Python

SDK für 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
```

- Einzelheiten zur API finden Sie unter FHIRDatastoreDescribe in AWS SDK for Python (Boto3) API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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.
```

- Einzelheiten zur API finden Sie unter Describe FHIRDatastore in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterVerwendung HealthLake mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`DescribeFHIRExportJob`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeFHIRExportJob` verwendet wird.

CLI

AWS CLI

So beschreiben Sie einen FHIR-Exportjob

Das folgende `describe-fhir-export-job` Beispiel zeigt, wie Sie die Eigenschaften eines FHIR-Exportauftrags in AWS HealthLake finden.

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

Ausgabe:

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

- Einzelheiten zur API finden Sie unter FHIRExportJob beschreiben in der AWS CLI Befehlsreferenz.

Python

SDK für 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
```

- Einzelheiten zur API finden Sie unter FHIRExportDescribe Job in AWS SDK for Python (Boto3) API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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.
```

- Einzelheiten zur API finden Sie unter FHIRExportDescribe Job in AWS SDK for SAP ABAP API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterVerwendung HealthLake mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`DescribeFHIRImportJob`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeFHIRImportJob` verwendet wird.

CLI

AWS CLI

So beschreiben Sie einen FHIR-Importjob

Das folgende `describe-fhir-import-job` Beispiel zeigt, wie Sie die Eigenschaften eines FHIR-Importjobs mithilfe AWS HealthLake von lernen können.

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

Ausgabe:

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

- Einzelheiten zur API finden Sie unter FHIRImportJob beschreiben in der AWS CLI Befehlsreferenz.

Python

SDK für 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
```

- Einzelheiten zur API finden Sie unter FHIRImportDescribe Job in AWS SDK for Python (Boto3) API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.
```

- Einzelheiten zur API finden Sie unter FHIRImportDescribe Job in AWS SDK for SAP ABAP API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterVerwendung HealthLake mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`ListFHIRDatastores`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListFHIRDatastores` verwendet wird.

CLI

AWS CLI

So listen Sie FHIR-Datenspeicher auf

Das folgende `list-fhir-datastores` Beispiel zeigt, wie der Befehl verwendet wird und wie Benutzer Ergebnisse nach dem Status des Datenspeichers filtern können AWS HealthLake.

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

Ausgabe:

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

- Einzelheiten zur API finden Sie unter Liste FHIRDatastores in der AWS CLI Befehlsreferenz.

Python

SDK für 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
```

- Einzelheiten zur API finden Sie unter Liste FHIRDatastores in der AWS API-Referenz zum SDK for Python (Boto3).

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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.
```

- Einzelheiten zur API finden Sie unter Liste FHIRDatastores in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterVerwendung HealthLake mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`ListFHIRExportJobs`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListFHIRExportJobs` verwendet wird.

CLI

AWS CLI

So listen Sie alle FHIR-Exportjobs auf

Das folgende Beispiel für `list-fhir-export-jobs` zeigt, wie Sie mit dem Befehl eine Liste von Exportjobs anzeigen, die mit einem Konto verknüpft sind.

```
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)
```

Ausgabe:

```
{
    "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)"
            }
```

```
        }
    ]
}
```

- Einzelheiten zur API finden Sie unter FHIRExportAuflisten von Aufträgen in der AWS CLI Befehlsreferenz.

Python

SDK für 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
```

- Einzelheiten zur API finden Sie unter List FHIRExport Jobs in AWS SDK for Python (Boto3) API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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.
```

- Einzelheiten zur API finden Sie in der Referenz zum FHIRExportAuflisten von Aufträgen im AWS SDK für SAP ABAP API.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterVerwendung HealthLake mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`ListFHIRImportJobs`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListFHIRImportJobs` verwendet wird.

CLI

AWS CLI

So listen Sie alle FHIR-Importjobs auf

Das folgende Beispiel für `list-fhir-import-jobs` zeigt, wie Sie mit dem Befehl eine Liste aller Importjobs anzeigen, die mit einem Konto verknüpft sind.

```
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)
```

Ausgabe:

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

- Einzelheiten zur API finden Sie unter FHIRImportAuflisten von Aufträgen in der AWS CLI Befehlsreferenz.

Python

SDK für 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
```

- Einzelheiten zur API finden Sie unter List FHIRImport Jobs in AWS SDK for Python (Boto3) API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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.
```

- Einzelheiten zur API finden Sie in der Referenz zum FHIRImportAuflisten von Aufträgen im AWS SDK für SAP ABAP API.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterVerwendung HealthLake mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`ListTagsForResource`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListTagsForResource` verwendet wird.

CLI

AWS CLI

So listen Sie Tags für einen Datenspeicher auf

Im folgenden Beispiel für `list-tags-for-resource` werden die Tags aufgelistet, die mit dem angegebenen Datenspeicher verknüpft sind:

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

Ausgabe:

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

```
    }
}
```

- Einzelheiten zur API finden Sie ListTagsForResourcein der AWS CLI Befehlsreferenz.

Python

SDK für 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
```

- Einzelheiten zur API finden Sie ListTagsForResourcein AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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.
```

- Einzelheiten zur API finden Sie ListTagsForResourcein der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterVerwendung HealthLake mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`StartFHIRExportJob`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `StartFHIRExportJob` verwendet wird.

CLI

AWS CLI

So starten Sie einen FHIR-Exportjob

Das folgende `start-fhir-export-job` Beispiel zeigt, wie Sie einen FHIR-Exportauftrag mit AWS HealthLake starten.

```
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)
```

Ausgabe:

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

- API-Einzelheiten finden Sie unter FHIRExportJob starten in der AWS CLI Befehlsreferenz.

Python

SDK für 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
```

- Einzelheiten zur API finden Sie unter Start FHIRExport Job in AWS SDK for Python (Boto3) API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

```
              iv_s3uri = iv_output_s3_uri
              iv_kmskeyid = iv_kms_key_id
            )
          )
          iv_dataaccessrolearn = iv_data_access_role_arn
          iv_datastoreid = iv_datastore_id
        ).
        DATA(lv_job_id) = oo_result->get_jobid( ).
        MESSAGE |Export job started with ID { lv_job_id }.| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllthrottlingex INTO DATA(lo_throttling_ex).
        lv_error = |Throttling error: { lo_throttling_ex->av_err_code }-
{ lo_throttling_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_throttling_ex.
      CATCH /aws1/cx_hllaccessdeniedex INTO DATA(lo_access_ex).
        lv_error = |Access denied: { lo_access_ex->av_err_code }-{ lo_access_ex-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_access_ex.
    ENDTRY.
```

- API-Details finden Sie unter Referenz zum Starten FHIRExport eines Job im AWS SDK für SAP ABAP API.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterVerwendung HealthLake mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`StartFHIRImportJob`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `StartFHIRImportJob` verwendet wird.

CLI

AWS CLI

So starten Sie einen FHIR-Importjob

Das folgende `start-fhir-import-job` Beispiel zeigt, wie Sie einen FHIR-Importjob mit AWS HealthLake starten.

```
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)"
```

Ausgabe:

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

- API-Einzelheiten finden Sie unter FHIRImportJob starten in der AWS CLI Befehlsreferenz.

Python

SDK für 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
```

- Einzelheiten zur API finden Sie unter Start FHIRImport Job in AWS SDK for Python (Boto3) API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

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

- API-Details finden Sie unter Referenz zum Starten FHIRImport eines Job im AWS SDK für SAP ABAP API.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterVerwendung HealthLake mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`TagResource`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `TagResource` verwendet wird.

CLI

AWS CLI

So fügen Sie ein Tag zum Datenspeicher hinzu

Im folgenden Beispiel für `tag-resource` wird gezeigt, wie Sie ein Tag zu einem Datenspeicher hinzufügen.

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

```
    --tags '[{"Key": "key1", "Value": "value1"}]'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

.

- Einzelheiten zur API finden Sie TagResourcein der AWS CLI Befehlsreferenz.

Python

SDK für 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
```

- Einzelheiten zur API finden Sie TagResourcein AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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.
```

- Einzelheiten zur API finden Sie TagResourcein der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterVerwendung HealthLake mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`UntagResource`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UntagResource` verwendet wird.

CLI

AWS CLI

So entfernen Sie Tags aus einem Datenspeicher.

Das folgende Beispiel für `untag-resource` zeigt, wie Sie Tags aus einem Datenspeicher entfernen.

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

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie UntagResourcein der AWS CLI Befehlsreferenz.

Python

SDK für 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
```

- Einzelheiten zur API finden Sie UntagResourcein AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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.
```

- Einzelheiten zur API finden Sie UntagResourcein der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterVerwendung HealthLake mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Integrieren AWS HealthLake

Die folgenden AWS Dienste lassen sich direkt in die integrierte Verarbeitung natürlicher Sprache, SQL-Abfragen und Data Warehousing integrieren, AWS HealthLake um sie zu ermöglichen.

- Amazon Comprehend Medical ist ein HIPAA-fähiger Service zur Verarbeitung natürlicher Sprache (NLP), der mithilfe von Bibliotheken für maschinelles Lernen aussagekräftige Gesundheitsdaten aus unstrukturiertem medizinischem Text extrahiert. HealthLake Weitere Informationen finden Sie im Amazon Comprehend Medical Developer Guide.
- Amazon Athena ist ein interaktiver Abfrageservice, mit dem Sie HealthLake Daten direkt in Amazon Simple Storage Service (Amazon S3) -Buckets mithilfe von Standard-SQL analysieren können. Weitere Informationen finden Sie im Amazon Athena Developer Guide.

Themen

- Integrierte Verarbeitung natürlicher Sprache (NLP) für HealthLake
- HealthLake Daten mit Amazon Athena abfragen

Integrierte Verarbeitung natürlicher Sprache (NLP) für HealthLake

AWS HealthLake bietet integrierte Bibliotheken zur Verarbeitung natürlicher Sprache (NLP) zum Analysieren, Identifizieren und Zuordnen unstrukturierter Daten, die in FHIR-Ressourcentypen gespeichert sind. `DocumentReference`

> **Wichtig**
>
> Integriertes NLP für HealthLake ist standardmäßig ausgeschaltet. Um es zu aktivieren, reichen Sie eine Support-Anfrage über AWS Support Center Console. Um Ihren Fall zu erstellen, melden Sie sich bei Ihrem an AWS-Konto und wählen Sie Kundenfall erstellen.

HealthLake integriertes NLP funktioniert, indem es die Amazon Comprehend Medical-`DetectEntities-V2`, `InferICD10-CM`, und `InferRxNorm` API-Aktionen aufruft. Die Aktionen hängen ihre Ergebnisse als Erweiterung an die Ressource an. `DocumentReference` Wenn die Amazon Comprehend Medical API-Aktionen Merkmale erkennen`SIGN`, die, und/oder `SYMPTOMDIAGNOSIS`, generieren sie eine `Linkage`FHIR-Ressource. Neue `Observation`

Ressourcen `Condition` und Ressourcen werden aus Entitäten erstellt, die mit den Merkmalen von`SIGN`, oder identifiziert wurden`SYMPTOM`, `DIAGNOSIS` und sie werden mithilfe der Ressource mit dem Quelldokument verknüpft. `Linkage`

> **Note**
>
> `GET`Anfragen für FHIR-Ressourcen, die durch HealthLake integriertes NLP generiert wurden, werden zwar unterstützt, die `search` FHIR-API-Funktionalität jedoch nicht. Weitere Informationen zur Suche nach NLP-Erweiterungen mithilfe HealthLake der Integration mit Athena finden Sie unter. SQL-Index und Abfrage

Inhalt

- HealthLake Integrierte NLP-Bibliotheken
- Verwendung von FHIR-REST-API-Interaktionen
- Suchparameter für integriertes NLP HealthLake
- HealthLake integrierte NLP-Beispielanfragen

HealthLake Integrierte NLP-Bibliotheken

HealthLake leitet mithilfe von Amazon Comprehend Medical Medical-Bibliotheken Daten ab, die im `DocumentReference` Ressourcentyp gefunden wurden. Die Amazon Comprehend Medical API-Funktionen *`DetectEntities-V2`* und *`InferRxNorm`* erkennen Erkrankungen als Merkmale. *`InferICD10-CM`* Jeder Vorgang bietet unterschiedliche Erkenntnisse.

> **Sprach-Support**
>
> Amazon Comprehend Medical API-Operationen erkennen nur medizinische Entitäten in englischsprachigen Texten.

- DetectEntities-V2: Prüft den klinischen Text für eine Vielzahl von medizinischen Entitäten und gibt spezifische Informationen zu ihnen zurück, z. B. die Kategorie der Entität, den Standort und den Konfidenzwert.
- Infer ICD1 0-CM: Erkennt Erkrankungen in einer Patientenakte als Entitäten und verknüpft diese Entitäten mit normalisierten Konzeptidentifikatoren in der ICD-10-CM-Wissensdatenbank des

National Center for Health Statistics der CDC mit Genehmigung der Weltgesundheitsorganisation (WHO).

- InferRxNorm: Erkennt Medikamente als Entitäten, die in einer Patientenakte aufgeführt sind, und verknüpft sie mit den normalisierten Konzeptidentifikatoren in der Datenbank der National Library of Medicine. RxNorm

Die unterstützten Eigenschaften für jeden API-Vorgang sind `SIGNSYMPTOM`, und. `DIAGNOSIS` Wenn Merkmale erkannt werden, werden sie als FHIR-konforme Erweiterungen an verschiedenen Stellen in Ihrem HealthLake Datenspeicher hinzugefügt.

Standorte, an denen Erweiterungen hinzugefügt werden.

- `DocumentReference`: Die Ergebnisse der Amazon Comprehend Medical API-Operationen werden jedem Dokument hinzugefügt, das innerhalb des `DocumentReference` Ressourcentyps gefunden wird. `extension` Die Ergebnisse in der Erweiterung sind in zwei Gruppen unterteilt. Sie finden sie in den Ergebnissen, die auf ihren basierenURL.
  - `http://healthlake.amazonaws.com/system-generated-resources/`
    - Dies sind Ressourcentypen, die von erstellt oder hinzugefügt wurden HealthLake.
  - `http://healthlake.amazonaws.com/aws-cm/`
    - Wo die Rohausgabe der Amazon Comprehend Medical API-Operationen zu Ihrem HealthLake Datenspeicher hinzugefügt wird.
- `Linkage`: Dieser Ressourcentyp wird entweder hinzugefügt oder als Ergebnis des integrierten NLP erstellt. Eine `GET` Anfrage zu einer bestimmten Ressource `Linkage` gibt eine Liste verknüpfter Ressourcen zurück. Suchen Sie nach dem `Linkage` hinzugefügten `"tag": [{"display": "SYSTEM_GENERATED"}]` Schlüssel-Wert-Paar HealthLake, um festzustellen, ob a hinzugefügt wurde. Weitere Informationen zu den FHIR-Spezifikationen für Linkage finden Sie Linkagein der FHIR R4-Dokumentation.
- FHIR-Ressourcentypen, die als Ergebnis der Abläufe von Amazon Comprehend Medical generiert wurden.
  - `Observation`: beinhaltet Ergebnisse der Amazon Comprehend Medical API-Aktionen `DetectEntities-V2` und `InferICD10-CM` wann die Merkmale oder sind`SIGN`. `SYMPTOM`
  - `Condition`: beinhaltet Ergebnisse der Amazon Comprehend Medical API-Aktionen `DetectEntities-V2` und `InferICD10-CM` wann das Merkmal vorhanden ist. `DIAGNOSIS`
  - `MedicationStatement`: beinhaltet Ergebnisse der Amazon Comprehend Medical API-Aktionen. `InferRxNorm`

Verwendung von FHIR-REST-API-Interaktionen

Standardmäßig werden Merkmale, die von den Amazon Comprehend Medical API-Vorgängen erkannt wurden, bei einer `GET` Anfrage nicht zurückgegeben. Um die Ergebnisse der integrierten NLP-Operationen zu sehen, müssen Sie einen `ID` für die folgenden FHIR-Ressourcentypen bekannten Typen angeben.

- `Linkage`
- `Observation`
- `Condition`
- `MedicationStatement`

Die Ergebnisse HealthLake integrierter NLP-Aktionen außerhalb des `DocumentReference` Ressourcentyps sind über eine `GET` Anfrage verfügbar, bei der die angegebene Anfrage `ID` bekanntermaßen Ergebnisse der Amazon Comprehend Medical API-Operationen enthält.

Suchparameter für integriertes NLP HealthLake

In der folgenden Tabelle sind die durchsuchbaren Attribute für HealthLake integriertes NLP aufgeführt.

Suchparameter für NLP HealthLake

Suchparameter	Findet Treffer für
DetECTE ntities-entity-category	Entitätskategorie innerhalb der DetectEntities Untererweiterung innerhalb der CM-Erweiterung AWS
DetECTE ntities-entity-text	Entitätstext innerhalb der DetectEntities Untererweiterung innerhalb der CM-Erweiterung AWS
DetECTE ntities-entity-type	Entitätstyp innerhalb der DetectEntities Untererweiterung innerhalb der CM-Erweiterung AWS
DetECTE ntities-entity-score	Entity Score innerhalb der DetectEntities Untererweiterung innerhalb der CM-Erweiterung AWS
Infer-ICD10 cm-entity-text	Entitätstext innerhalb der Infer ICD1 0CM-Unter erweiterung innerhalb der CM-Erweiterung AWS

Suchparameter	Findet Treffer für
infer-icd10 cm-entity-score	Entity Score innerhalb der Infer ICD1 0CM-Unter erweiterung innerhalb der CM-Erweiterung AWS
infer-icd-10 cm-entity-concept-code	Entitätskonzeptcode innerhalb der Infer ICD1 0CM-Untererweiterung innerhalb der CM-Erweiterung AWS
infer-icd10 cm-entity-concept-description	Beschreibung des Entitätskonzepts innerhalb der Infer ICD1 0CM-Untererweiterung innerhalb der CM-Erweiterung AWS
infer-icd10 cm-entity-concept-score	Entity Concept Score innerhalb der Infer ICD1 0CM-Untererweiterung innerhalb der CM-Erweiterung AWS
infer-rxnorm-entity-score	Entity Score innerhalb der InferRxNorm Untererwe iterung innerhalb der CM-Erweiterung AWS
infer-rxnorm-entity-text	Entitätstext innerhalb der InferRxNorm Untererwe iterung innerhalb der CM-Erweiterung AWS
infer-rxnorm-entity-concept-Code	Entitätskonzept-Code innerhalb der InferRxNorm Untererweiterung innerhalb der CM-Erweiterung AWS
infer-rxnorm-entity-concept-Beschreibung	Beschreibung des Entitätskonzepts innerhalb der InferRxNorm Untererweiterung innerhalb der CM-Erweiterung AWS
infer-rxnorm-entity-concept-Punktzahl	Entity Concept Score innerhalb der InferRxNorm Untererweiterung innerhalb der CM-Erweiterung AWS

HealthLake bietet eine spezielle Suche, um die Kriterien zu erfüllen, wenn sie `EntityText` Teil derselben Entität `EntityCategory` sind. In der folgenden Tabelle werden die speziellen Suchparameter beschrieben, die von unterstützt werden HealthLake.

Suchparameter

Suchparameter	Treffer wurden zurückgegeben
DetECTE ntities-entity-text-category	Wenn die DetectEntities Untererweiterung mindesten s eine Entität enthält, die sowohl mit EntityText als auch mit EntityCategory übereinstimmt.
DetecTe ntities-entity-type-score	Wenn es in der DetectEntities Untererweiterung mindestens eine Entität gibt, die sowohl dem EntityType als auch dem EntityScore entspricht.
Detecte ntities-entity-text-score	Wenn die Untererweiterung mindestens eine Entität enthält, die sowohl mit DetectEntities EntityText als auch mit EntityScore übereinstimmt.
DetecTe ntities-entity-text-type	Wenn die DetectEntities Untererweiterung mindesten s eine Entität enthält, die sowohl mit EntityText als auch mit EntityType übereinstimmt.
Detecte ntities-entity-category-score	Wenn es mindestens eine Entität gibt, die sowohl der EntityCategory als auch der EntityScore entspricht.
infer-icd10-Code cm-entity-text-concept	Wenn es mindestens eine Entität in der Infer ICD1 0CM-Untererweiterung gibt, die dem EntityTex t entspricht, und wenn es mindestens einen ConceptCode für diese Entität gibt, der dem Code entspricht.
infer-icd10 -Score cm-entity-text-concept	Wenn es mindestens eine Entität in der Infer ICD1 0CM-Untererweiterung gibt, die dem EntityText entspricht, und wenn es mindestens einen ConceptSc ore für diese Entität gibt, der der Punktzahl entspricht.
infer-icd10 -concept-score cm-entity-concept-description	Wenn es innerhalb der Entität in der Infer ICD1 0CM-Untererweiterung mindestens ein Konzept gibt, das der Konzeptbeschreibung und dem ConceptScore entspricht.

Suchparameter	Treffer wurden zurückgegeben
infer-rxnorm-entity-text-Konzeptcode	Wenn es in der InferRxNorm Untererweiterung mindestens eine Entität gibt, die dem EntityTex t entspricht, und wenn es mindestens einen ConceptCode für diese Entität gibt, der dem Code entspricht.
infer-rxnorm-entity-text-Konzept-Score	Wenn es in der InferRxNorm Untererweiterung mindestens eine Entität gibt, die dem EntityText entspricht, und es mindestens einen ConceptScore für diese Entität gibt, der der Punktzahl entspricht.
infer-rxnorm-entity-concept-description-concept-score	Wenn es innerhalb der Entität in der InferRxNorm Untererweiterung mindestens ein Konzept gibt, das der Konzeptbeschreibung und dem ConceptScore entspricht.

HealthLake integrierte NLP-Beispielanfragen

Beispiel 1: **`Patient`** Datensatz, der in einen HealthLake Datenspeicher aufgenommen wurde

Im Folgenden finden Sie ein Beispiel für eine klinische Notiz, die auf einer `Patient` Begegnung mit einem medizinischen Fachpersonal basiert.

> **Synthetische Daten**
>
> Der Text im folgenden Beispiel ist synthetischer Inhalt und enthält keine geschützten Gesundheitsinformationen (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)
```

Zur Erinnerung: Diese Informationen sind in der Ressource im Base64-Format codiert. `DocumentReference` Wenn dieses Dokument aufgenommen wurde HealthLake und die Amazon Comprehend Medical API-Operationen abgeschlossen sind, können Sie mit der `GET` Anfrage für den Ressourcentyp beginnen, um die Ergebnisse zu sehen. `DocumentReference`

```
GET https://https://healthlake.region.amazonaws.com/datastore/datastoreId/
r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/DocumentReference
```

Wenn die Amazon Comprehend Medical API-Operationen erfolgreich sind, suchen Sie nach diesen Schlüssel-Wert-Paaren in den folgenden Links `extension "url": "http://healthlake.amazonaws.com/aws-cm/"`

```
{
  "url": "http://healthlake.amazonaws.com/aws-cm/status/",
  "valueString": "SUCCESS"
 },
 {
  "url": "http://healthlake.amazonaws.com/aws-cm/message/",
  "valueString": "The AWS HealthLake integrated medical NLP operation was successful."
 }
```

Die folgenden Registerkarten zeigen Ihnen, wie die aufgenommenen Patientenakten je nach Ressourcentyp in Ihrem HealthLake Datenspeicher gemeldet werden.

DocumentReference

Um die Ergebnisse für einen einzelnen `DocumentReference` Ressourcentyp zu sehen, stellen Sie eine `GET` Anfrage, in `id` der die Daten einer bestimmten Ressource bereitgestellt werden.

```
GET https://https://healthlake.region.amazonaws.com/datastore/datastoreId/
r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/DocumentReference/0e938f03-da7f-4178-acd8-
eea9586c46ed
```

Bei Erfolg erhalten Sie einen `200` HTTP-Antwortcode und die folgende JSON-Antwort (die aus Gründen der Übersichtlichkeit gekürzt wurde).

Hier ist der `http://healthlake.amazonaws.com/system-generated-resources/` Teil. Sie können sehen, dass ein neuer hinzugefügt `Linkage/e366d29f-2c22-4c19-866e-09603937935a` wurde. Sie können auch sehen, wo auf HealthLake Inferenzen basierende Ergebnisse zu bestimmten `Observation` `Condition` Ressourcentypen hinzugefügt wurden.

Um zu sehen, wie diese Ressourcentypen geändert wurden, wählen Sie die entsprechenden Tabs.

```
{
  "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

Um die Ergebnisse für einen einzelnen `Linkage` Ressourcentyp zu sehen, stellen Sie eine `GET` Anfrage, in `ID` der die einer bestimmten Ressource angegeben ist.

```
GET https://https://healthlake.region.amazonaws.com/datastore/datastoreId/
r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/Linkage/e366d29f-2c22-4c19-866e-09603937935a
```

Bei Erfolg erhalten Sie einen `200` HTTP-Antwortcode und die folgende gekürzte JSON-Antwort.

Die Antwort enthält das `item` Element. Darin `"type": "source"` gibt das Schlüssel-Wert-Paar den spezifischen `DocumentReference` Eintrag an, mit dem das Schlüssel-Wert-Paar geändert wurde `Condition` und der unter dem `"type": "alternate"` Schlüssel-Wert-Paar `Observations` aufgeführt ist.

Sie sehen auch das *`meta`* Element und ein entsprechendes Schlüssel-Wert-Paar, was darauf hinweist*`"tag": [{"display": "SYSTEM_GENERATED"}]`*, dass diese Ressourcen von erstellt wurden. 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

Um die Ergebnisse für einen einzelnen `Observation` Ressourcentyp zu sehen, stellen Sie eine `GET` Anfrage, in `ID` der die einer bestimmten Ressource bereitgestellt wird.

```
GET https://https://healthlake.region.amazonaws.com/
datastore/datastoreId/r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/
Observation/e366d29f-2c22-4c19-866e-09603937935a
```

Die Ergebnisse der Amazon Comprehend Medical API-Operationen wurden um die folgenden Elemente ergänzt: `codemeta`, und. `modifierExtension`

`code`

Ein Element vom Typ. `CodeableConcept` Weitere Informationen finden Sie `CodeableConcept`in der FHIR R4-Dokumentation.

HealthLake hängt die folgenden drei Schlüssel-Wert-Paare an.

- `"system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/"`: Wo sich die URL auf einen bestimmten Amazon Comprehend Medical API-Vorgang bezieht. ICD1Inferieren Sie in diesem Fall 0CM.
- `"code": "A52.06"`: Wo `A52.06` ist der ICD-10-CM-Code, der das Konzept identifiziert, das in der Wissensdatenbank der Centers for Disease Control zu finden ist?
- `"display": "Other syphilitic heart involvement"`: Wo `"Other syphilitic heart involvement"` ist die lange Beschreibung des ICD-10-CM-Codes in der Ontologie.

Die folgende gekürzte JSON-Antwort enthält nur das Element. `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"
 }
```

Verwenden Sie das Element, um zu verstehen, ob das Modell sicher ist, dass der zugewiesene ICD-10-CM-Code korrekt ist. `modifierExtension`

`meta`

Das `meta` Element enthält Metadaten, die angeben, ob das `code` Element Details enthält, die durch die Amazon Comprehend Medical API-Operationen hinzugefügt wurden.

Die folgende gekürzte JSON-Antwort enthält nur das Element. `meta`

```
"meta": {
  "lastUpdated": "2022-10-21T19:38:30.879Z",
  "tag": [{
   "display": "SYSTEM_GENERATED"
  }]
 }
```

`modifierExtension`

Das `modifierExtension` Element enthält weitere Informationen zum Vertrauensgrad der zugewiesenen Codes, die im `code` Element gefunden wurden. Es enthält auch Schlüssel-Wert-Paare, die einen Link zurück zum Original, das zur Generierung der Ergebnisse DocumentReference verwendet wurde, und zum zugehörigen Linkage-Ressourcentyp bereitstellen.

Für jedes hinzugefügte `coding` Element sehen Sie ein `entity-score` und ein, das der ModifierExtension `entity-Concept-Score` hinzugefügt wurde. Für jeden Wert im Schlüssel-Wert-Paar wird eine Punktzahl angezeigt. Denn `entity-score` dieser Wert ist das Maß an Vertrauen, das Amazon Comprehend Medical in die Genauigkeit der Erkennung hat. Denn dieser Wert ist das Maß an Vertrauen`entity-Concept-Score`, das Amazon Comprehend Medical davon hat, dass das Unternehmen korrekt mit einem ICD-10-CM-Konzept verknüpft ist.

Die folgende gekürzte JSON-Antwort enthält nur das Element. `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"
   }
  }
 ]
```

Vollständige JSON-Antwort

```
{
  "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

Um die Ergebnisse für einen einzelnen `Condition` Ressourcentyp zu sehen, stellen Sie eine `GET` Anfrage, in `ID` der die einer bestimmten Ressource angegeben wird.

```
GET https://https://healthlake.region.amazonaws.com/datastore/datastoreId/
r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/Condition/b06d343d-
ddb8-4f36-82cb-853fcd434dfd
```

Die Ergebnisse der Amazon Comprehend Medical API-Operationen wurden um die folgenden Elemente ergänzt: `codemeta`, und. `modifierExtension`

`code`

Ein Element vom Typ. `CodeableConcept` Weitere Informationen finden Sie `CodeableConcept`in der FHIR R4-Dokumentation.

HealthLake hängt die folgenden drei Schlüssel-Wert-Paare an.

- `"system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/"`: Wo sich die URL auf einen bestimmten Amazon Comprehend Medical API-Vorgang bezieht. ICD1Inferieren Sie in diesem Fall 0CM.
- `"code": "I70.0"`: Wo `A52.06` ist der ICD-10-CM-Code, der das Konzept identifiziert, das in der Wissensdatenbank der Centers for Disease Control zu finden ist?
- `"display": "Atherosclerosis of aorta"`: Wo `"Other syphilitic heart involvement"` ist die lange Beschreibung des ICD-10-CM-Codes in der Ontologie.

Die folgende gekürzte JSON-Antwort enthält nur das Element. `code`

```
"code": {
  "coding":
  [
    {
   "system": "http://healthlake.amazonaws.com/aws-cm/infer-icd10/",
   "code": "I70.0",
   "display": "Atherosclerosis of aorta"
    }
  ],
  "text": "Atherosclerosis of aorta"
 }
```

Verwenden Sie das Element, um zu verstehen, ob das Modell sicher ist, dass der zugewiesene ICD-10-CM-Code korrekt ist. `modifierExtension`

`meta`

Das `meta` Element enthält Metadaten, die angeben, ob das `code` Element Details enthält, die durch die Amazon Comprehend Medical API-Operationen hinzugefügt wurden.

Die folgende gekürzte JSON-Antwort enthält nur das Element. `meta`

```
"meta": {
  "lastUpdated": "2022-10-21T19:38:30.877Z",
  "tag": [{
   "display": "SYSTEM_GENERATED"
  }]
 }
```

`modifierExtension`

Das `modifierExtension` Element enthält weitere Informationen zum Vertrauensgrad der zugewiesenen Codes, die im `code` Element gefunden wurden. Es enthält auch Schlüssel-Wert-Paare, die einen Link zurück zum Original, das zur Generierung der Ergebnisse DocumentReference verwendet wurde, und zum zugehörigen Linkage-Ressourcentyp bereitstellen.

Für jedes hinzugefügte `coding` Element sehen Sie ein `entity-score` und ein, das der ModifierExtension `entity-Concept-Score` hinzugefügt wurde. Für jeden Wert im Schlüssel-Wert-Paar wird eine Punktzahl angezeigt. Denn `entity-score` dieser Wert ist das Maß an

Vertrauen, das Amazon Comprehend Medical in die Genauigkeit der Erkennung hat. Denn dieser Wert ist das Maß an Vertrauen`entity-Concept-Score`, das Amazon Comprehend Medical davon hat, dass das Unternehmen korrekt mit einem ICD-10-CM-Konzept verknüpft ist.

Die folgende gekürzte JSON-Antwort enthält nur das Element. `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"
   }
  }
 ]
```

Vollständige JSON-Antwort

```
{
  "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"
 }
```

Beispiel 2: A**`DocumentReference`**, das den **`MedicationStatement`** Ressourcentyp enthält

Hier ist ein Beispiel für eine klinische Notiz, die auf der Begegnung eines Patienten mit einem Arzt basiert.

> ⚠ Synthetische Daten
>
> Der Text in diesem Beispiel ist synthetischer Inhalt und enthält keine geschützten Gesundheitsinformationen (PHI).

```
Tom is not prescribed Advil
```

Die folgenden Registerkarten zeigen, wie die aufgenommenen Patientenakten je nach Ressourcentyp in Ihrem HealthLake Datenspeicher gemeldet werden.

DocumentReference

Um die Ergebnisse für einen einzelnen `DocumentReference` Ressourcentyp zu sehen, stellen Sie eine `GET` Anfrage, in `ID` der die Daten einer bestimmten Ressource bereitgestellt werden.

```
GET https://https://healthlake.region.amazonaws.com/datastore/datastoreId/
r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/DocumentReference/c549125d-a218-421f-
b8bf-23614c5e796c
```

Bei Erfolg erhalten Sie einen `200` HTTP-Antwortcode und die folgende gekürzte JSON-Antwort.

Das Schlüssel-Wert-Paar,, gibt an`"url": "http://healthlake.amazonaws.com/system-generated-resources/"`, dass die darin enthaltenen Ressourcentypen durch Amazon Comprehend Medical API-Operationen hinzugefügt `extension` wurden. Sie können den neuen `Linkage` Ressourcentyp und mehrere Ressourcen sehen. `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

Um die Ergebnisse für einen einzelnen `Linkage` Ressourcentyp zu sehen, stellen Sie eine `GET` Anfrage, in `ID` der die einer bestimmten Ressource bereitgestellt wird.

```
GET https://https://healthlake.region.amazonaws.com/datastore/datastoreId/
r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/Linkage/394bb244-177b-4409-8657-26b20ed56dd7
```

Bei Erfolg erhalten Sie einen `200` HTTP-Antwortcode und die folgende JSON-Antwort.

Die Antwort enthält das `item` Element. Darin `"type": "source"` gibt das Schlüssel-Wert-Paar den spezifischen `DocumentReference` Eintrag an, der zur Änderung der `MedicationStatement` Ressourcentypen verwendet wird.

Sie können auch das `meta` Element und ein entsprechendes Schlüssel-Wert-Paar sehen, was darauf hinweist`"tag": [{"display": "SYSTEM_GENERATED"}]`, dass diese Ressourcen von erstellt wurden. 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

Um die Ergebnisse für einen einzelnen `MedicationStatement` Ressourcentyp zu sehen, stellen Sie eine `GET` Anfrage, in `ID` der die einer bestimmten Ressource bereitgestellt wird.

```
GET https://https://healthlake.region.amazonaws.com/
datastore/datastoreId/r4/eeb8005725ae22b35b4edbdc68cf2dfd/r4/
MedicationStatement/9a89b0d3-6681-45ca-9926-27951edce5c7
```

Der `MedicationStatement` Ressourcentyp ist der Ort, an dem die Ergebnisse des Amazon Comprehend Medical InferRxNorm API-Vorgangs gefunden werden. Die Ergebnisse werden um die folgenden Elemente ergänzt: `medicationCodeableConceptmeta`, und. `modifierExtension`

medicationCodeableConcept

Ein Element vom Typ`CodeableConcept`. Weitere Informationen finden Sie `CodeableConcept`in der FHIR R4-Dokumentation.

HealthLake hängt die folgenden drei Schlüssel-Wert-Paare an.

- `"system": ""http://healthlake.amazonaws.com/aws-cm/infer-rxnorm/`: Wo sich die URL auf einen bestimmten Amazon Comprehend Medical API-Vorgang bezieht. In diesem Fall. InferRxNorm
- `"code": "731533"`: Wo `731533` ist eine RxNorm Konzept-ID, auch bekannt als RxCUI.

- `"display": "ibuprofen 200 MG Oral Capsule [Advil]"`: Wo `ibuprofen 200 MG Oral Capsule [Advil]` ist die Beschreibung des Konzepts RxNorm .

Die folgende gekürzte JSON-Antwort enthält nur das `MedicationStatement` Element.

```
"medicationCodeableConcept": {
  "coding": [
   {
    "system": "http://healthlake.amazonaws.com/aws-cm/infer-rxnorm/",
    "code": "731533",
    "display": "ibuprofen 200 MG Oral Capsule [Advil]"
   }
  ]
 }
```

`meta`

Das `meta` Element enthält Metadaten, die angeben, ob das `code` Element Details enthält, die durch die Amazon Comprehend Medical API-Operationen hinzugefügt wurden.

Die folgende gekürzte JSON-Antwort enthält nur das Element. `meta`

```
"meta": {
  "lastUpdated": "2022-10-24T20:05:02.800Z",
  "tag": [
   {
    "display": "SYSTEM_GENERATED"
   }
  ]
 }
```

`modifierExtension`

Das `modifierExtension` Element enthält Schlüssel-Wert-Paare, die einen Link zurück zum Original, das zur Generierung der Ergebnisse `DocumentReference` verwendet wurde, und zum zugehörigen Linkage-Ressourcentyp bereitstellen.

```
"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"
    }
   }
]
```

HealthLake Daten mit Amazon Athena abfragen

Während eines HealthLake Importjobs werden verschachtelte FHIR-JSON-Daten einem ETL-Prozess unterzogen und im offenen Tabellenformat von Apache Iceberg gespeichert, wobei jeder FHIR-Ressourcentyp als einzelne Tabelle in Athena dargestellt wird. Auf diese Weise können Benutzer die FHIR-Daten mit SQL abfragen, ohne sie jedoch zuerst exportieren zu müssen. Dies ist wertvoll, da es Klinikern und Wissenschaftlern ermöglicht, FHIR-Daten abzufragen, um ihre Entscheidungen zu validieren oder ihre Forschung voranzutreiben. Weitere Informationen zur Funktionsweise von Apache Iceberg-Tabellen in Athena finden Sie unter Abfragen von Apache Iceberg-Tabellen im Athena-Benutzerhandbuch.

> **Note**
>
> HealthLake unterstützt die FHIR `read` R4-Interaktion mit Ihren HealthLake Daten in Athena. Weitere Informationen finden Sie unter Eine FHIR-Ressource lesen.

In den Themen dieses Abschnitts wird beschrieben, wie Sie Ihren HealthLake Datenspeicher mit Athena verbinden, wie Sie ihn mit SQL abfragen und wie Sie Ergebnisse für weitere Analysen mit anderen AWS Diensten verbinden.

Themen

- Erste Schritte mit Amazon Athena
- HealthLake Daten mit SQL abfragen
- Beispiel für SQL-Abfragen mit komplexer Filterung

Erste Schritte mit Amazon Athena

Für die Integration HealthLake mit Amazon Athena müssen Sie Berechtigungen einrichten. Dazu erstellen Sie einen Athena-Benutzer, eine Gruppe oder eine Rolle und gewähren ihnen Zugriff auf FHIR-Ressourcen, die sich in einem HealthLake Datenspeicher befinden.

- Einem Benutzer, einer Gruppe oder einer Rolle Zugriff auf einen HealthLake Datenspeicher gewähren (AWS Lake Formation Console)
- Einrichtung eines Athena-Kontos

Einem Benutzer, einer Gruppe oder einer Rolle Zugriff auf einen HealthLake Datenspeicher gewähren (AWS Lake Formation Console)

> Person: Administrator HealthLake
>
> Die HealthLake Administrator-Persona ist ein Data Lake-Administrator in AWS Lake Formation. Sie gewähren Zugriff auf HealthLake Datenspeicher in Lake Formation.

Für jeden erstellten Datenspeicher sind zwei Einträge in der AWS Lake Formation Formation-Konsole sichtbar. Ein Eintrag ist ein Ressourcenlink. Namen von Ressourcenlinks werden immer kursiv angezeigt. Jeder Ressourcenlink wird mit dem Namen und dem Besitzer der verknüpften gemeinsam genutzten Ressource angezeigt. Für alle HealthLake Datenspeicher ist der Besitzer der gemeinsam genutzten Ressource das HealthLake Dienstkonto. Der andere Eintrag ist der HealthLake Datenspeicher im HealthLake Dienstkonto. Die Schritte in diesem Verfahren verwenden den Datenspeicher, der die Ressourcenverknüpfung darstellt.

Weitere Informationen zu Ressourcenlinks finden Sie unter So funktionieren Ressourcenlinks in Lake Formation im AWS Lake Formation Developer Guide.

Damit ein Benutzer, eine Gruppe oder eine Rolle Daten in Athena abfragen kann, müssen Sie die Describe-Berechtigung für die Ressourcendatenbank erteilen. Anschließend müssen Sie den Tabellen die Optionen Select und Describe gewähren.

SCHRITT 1: So gewähren Sie DESCRIBE-Berechtigungen für eine HealthLake Data Store-Resource Link-Datenbank

1. Öffnen Sie die AWS Lake Formation Formation-Konsole: https://console.aws.amazon.com/lakeformation/
2. Wählen Sie in der primären Navigationsleiste Datenbanken aus.
3. Wählen Sie auf der Seite Datenbanken das Optionsfeld neben dem Namen des Datenspeichers, der kursiv gedruckt ist.
4. Wählen Sie Aktionen (▾).
5. Wählen Sie Grant (Erteilen).
6. Wählen Sie auf der Seite Datenberechtigungen gewähren unter Principals die Option IAM-Benutzer oder -Rollen aus.
7. Verwenden Sie unter IAM-Benutzer oder -Rollen den Abwärtspfeil (▾), oder suchen Sie nach dem IAM-Benutzer, der Rolle oder der Gruppe, zu der Sie in Athena Abfragen stellen möchten.
8. Wählen Sie auf der Karte LF-Tags oder Katalogressourcen die Option Benannte Datenkatalogressourcen aus.
9. Wählen Sie unter Datenbanken mit dem Abwärtspfeil (▾) die HealthLake Datenspeicher-Datenbank aus, auf die Sie gemeinsam zugreifen möchten.
10. Wählen Sie auf der Karte Berechtigungen für Ressourcenlinks unter Berechtigungen für Ressourcenverknüpfungen die Option Beschreiben aus.

Wenn die Erteilung erfolgreich war, wird das Banner „Genehmigung erteilt“ angezeigt. Um die soeben erteilte Berechtigung einzusehen, wählen Sie Data Lake-Berechtigungen aus. Suchen Sie den Benutzer, die Gruppe und die Rolle in der Tabelle. In der Spalte Berechtigungen wird Describe aufgeführt.

Jetzt müssen Sie Grant on target verwenden, um Select und Describe für alle Tabellen in der Datenbank zu gewähren.

SCHRITT 2: Gewähren Sie Zugriff auf alle Tabellen in einem HealthLake Datenspeicher-Ressourcenlink

1. Öffnen Sie die AWS Lake Formation Formation-Konsole: https://console.aws.amazon.com/lakeformation/
2. Wählen Sie in der primären Navigationsleiste Datenbanken aus.

3. Wählen Sie auf der Seite Datenbanken das Optionsfeld neben dem Namen des Datenspeichers, der kursiv gedruckt ist.
4. Wählen Sie Aktionen (▾).
5. Wählen Sie Grant on target aus.
6. Wählen Sie auf der Seite Datenberechtigungen gewähren unter Principals die Option IAM-Benutzer oder -Rollen aus.
7. Verwenden Sie unter IAM-Benutzer oder -Rollen den Abwärtspfeil (▾) oder suchen Sie nach dem IAM-Benutzer, der Gruppe oder der Rolle, zu der Sie in Athena Abfragen stellen möchten.
8. Wählen Sie unter LF-Tags oder Katalogressourcen die Option Benannte Datenkatalogressourcen aus.
9. Wählen Sie unter Datenbanken mit dem Abwärtspfeil (▾) die HealthLake Datenspeicher-Datenbank aus, auf die Sie Zugriff gewähren möchten.
10. Wählen Sie unter Tabellen die Option Alle Tabellen aus, um alle Tabellen für einen HealthLake Benutzer freizugeben.
11. Wählen Sie auf der Karte Tabellenberechtigungen unter Tabellenberechtigungen die Option Beschreiben und auswählen aus.
12. Wählen Sie Grant (Erteilen).

Nachdem Sie „Gewähren“ ausgewählt haben, wird das Erfolgsbanner „Berechtigungen gewähren“ angezeigt. Der angegebene Benutzer kann jetzt Abfragen an einem HealthLake Datenspeicher in Athena stellen.

Erste Schritte mit Athena

> **HealthLake Nutzer**
>
> Der HealthLake Benutzer verwendet die Athena-Konsole oder AWS SDKs fragt einen HealthLake Datenspeicher ab AWS CLI, der ihm vom HealthLake Administrator zur Verfügung gestellt wurde.

Um einen Datenspeicher mit Athena abzufragen, müssen Sie die folgenden drei Dinge tun.

- Gewähren Sie dem IAM-Benutzer oder der IAM-Rolle über Lake Formation Zugriff auf den HealthLake Datenspeicher. Weitere Informationen hierzu finden Sie unter Einem Benutzer, einer

Gruppe oder einer Rolle Zugriff auf einen HealthLake Datenspeicher gewähren (AWS Lake Formation Console).

- Erstellen Sie eine Arbeitsgruppe für Ihren HealthLake Datenspeicher.
- Benennen Sie einen Amazon S3 S3-Bucket zum Speichern Ihrer Abfrageergebnisse.

Um mit Athena zu beginnen, fügen Sie Ihrem Benutzer, Ihrer Gruppe oder Rolle die FullAccess AWS verwalteten Richtlinien AmazonAthenaFullAccessund AmazonS3 hinzu. Die Verwendung einer AWS verwalteten Richtlinie ist eine hervorragende Möglichkeit, um mit der Nutzung eines neuen Dienstes zu beginnen. Beachten Sie, dass von AWS verwaltete Richtlinien möglicherweise keine Least-Privilege-Berechtigungen für Ihre speziellen Anwendungsfälle gewähren, da sie für alle AWS-Kunden verfügbar sind. Gewähren Sie die Berechtigungen mit IAM Richtlinien nur die zum Ausführen einer Aufgabe erforderlich sind. Weitere Informationen zu IAM und der Anwendung von Least-Privilege-Berechtigungen finden Sie im IAM-Benutzerhandbuch unter Anwenden von Least-Privilege-Berechtigungen.

> **Important**
>
> Um einen HealthLake Datenspeicher in Athena abzufragen, müssen Sie die Athena-Engine Version 3 verwenden.

Arbeitsgruppen sind Ressourcen, und daher können Sie IAM-basierte Richtlinien verwenden, um den Zugriff auf bestimmte Arbeitsgruppen zu steuern. Weitere Informationen finden Sie im Athena-Benutzerhandbuch unter Verwendung von Arbeitsgruppen zur Kontrolle des Abfragezugriffs und der Kosten.

Weitere Informationen zur Einrichtung von Arbeitsgruppen finden Sie https://docs.aws.amazon.com/athena/latest/ug/workgroups-procedure.html im Athena-Benutzerhandbuch.

> **Note**
>
> Die Region, in der sich Ihr Amazon S3 S3-Bucket befindet, und die Athena-Konsole müssen übereinstimmen.

Bevor Sie eine Abfrage ausführen können, muss in Amazon S3 ein Speicherort für das Abfrageergebnis angegeben werden, oder Sie müssen eine Arbeitsgruppe verwenden, für die

ein Bucket angegeben wurde und deren Konfiguration die Client-Einstellungen überschreibt. Ausgabedateien werden automatisch für jede Abfrage gespeichert, die ausgeführt wird.

Weitere Informationen zur Angabe von Speicherorten für Abfrageergebnisse in der Athena-Konsole finden Sie unter Angeben eines Speicherorts für Abfrageergebnisse mithilfe der Athena-Konsole im Amazon Athena Athena-Benutzerhandbuch.

Beispiele für die Abfrage Ihres HealthLake Datenspeichers in Athena finden Sie unterHealthLake Daten mit SQL abfragen.

HealthLake Daten mit SQL abfragen

Wenn Sie Ihre FHIR-Daten in den HealthLake Datenspeicher importieren, werden die verschachtelten JSON-FHIR-Daten gleichzeitig einem ETL-Prozess unterzogen und im offenen Tabellenformat von Apache Iceberg in Amazon S3 gespeichert. Jeder FHIR-Ressourcentyp aus Ihrem HealthLake Datenspeicher wird in eine Tabelle konvertiert, in der er mit Amazon Athena abgefragt werden kann. Die Tabellen können einzeln oder als Gruppe mithilfe von SQL-basierten Abfragen abgefragt werden. Aufgrund der Struktur der Datenspeicher werden Ihre Daten als mehrere verschiedene Datentypen in Athena importiert. Weitere Informationen zum Erstellen von SQL-Abfragen, die auf diese Datentypen zugreifen können, finden Sie unter Abfrage-Arrays mit komplexen Typen und verschachtelten Strukturen im Amazon Athena Athena-Benutzerhandbuch.

> **Note**
>
> Alle Beispiele in diesem Thema verwenden fiktionalisierte Daten, die mit Synthea erstellt wurden. Weitere Informationen zum Erstellen eines mit Synthea-Daten vorinstallierten Datenspeichers finden Sie unter. Einen HealthLake Datenspeicher erstellen

Für jedes Element in einem Ressourcentyp definiert die FHIR-Spezifikation eine Kardinalität. Die Kardinalität eines Elements definiert die Unter- und Obergrenzen, wie oft dieses Element vorkommen kann. Bei der Erstellung einer SQL-Abfrage müssen Sie dies berücksichtigen. Schauen wir uns zum Beispiel einige Elemente im Ressourcentyp: Patient an.

- Element: Name Die FHIR-Spezifikation legt die Kardinalität als fest. `0..*`

  Das Element wird als Array erfasst.

```
[{
```

```
 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
}]
```

Um in Athena zu sehen, wie ein Ressourcentyp aufgenommen wurde, suchen Sie unter Tabellen und Ansichten danach. Um auf Elemente in diesem Array zuzugreifen, können Sie die Punktnotation verwenden. Hier ist ein einfaches Beispiel, das auf die Werte für `given` und zugreifen würde`family`.

```
SELECT
    name[1].given as FirstName,
    name[1].family as LastName
FROM Patient
```

- Element: MaritalStatus Die FHIR-Spezifikation legt die Kardinalität als fest. `0..1`

  Dieses Element wird als JSON erfasst.

```
{
 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
}
```

Um in Athena zu sehen, wie ein Ressourcentyp aufgenommen wurde, suchen Sie unter Tabellen und Ansichten danach. Um auf Schlüssel-Wert-Paare im JSON zuzugreifen, können Sie die Punktnotation verwenden. Da es sich nicht um ein Array handelt, ist kein Array-Index erforderlich. Hier ist ein einfaches Beispiel, das auf den Wert für zugreifen würde`text`.

```
SELECT
    maritalstatus.text as MaritalStatus
FROM Patient
```

Weitere Informationen zum Zugreifen auf und Durchsuchen von JSON finden Sie unter JSON abfragen im Athena-Benutzerhandbuch.

Die Abfrageanweisungen der Athena Data Manipulation Language (DML) basieren auf Trino. Athena unterstützt nicht alle Funktionen von Trino, und es gibt signifikante Unterschiede. Weitere Informationen finden Sie unter DML-Abfragen, Funktionen und Operatoren im Amazon Athena Athena-Benutzerhandbuch.

Darüber hinaus unterstützt Athena mehrere Datentypen, auf die Sie bei der Erstellung von Abfragen Ihres HealthLake Datenspeichers stoßen können. Weitere Informationen zu Datentypen in Athena finden Sie unter Datentypen in Amazon Athena im Amazon Athena Athena-Benutzerhandbuch.

Weitere Informationen zur Funktionsweise von SQL-Abfragen in Athena finden Sie unter SQL-Referenz für Amazon Athena im Amazon Athena Athena-Benutzerhandbuch.

Jede Registerkarte zeigt Beispiele für die Suche nach den angegebenen Ressourcentypen und zugehörigen Elementen mit Athena.

Element: Extension

Das Element `extension` wird verwendet, um benutzerdefinierte Felder in einem Datenspeicher zu erstellen.

Dieses Beispiel zeigt Ihnen, wie Sie auf die Funktionen des `extension` Elements zugreifen, das im `Patient` Ressourcentyp gefunden wurde.

Wenn Ihr HealthLake Datenspeicher in Athena importiert wird, werden die Elemente eines Ressourcentyps unterschiedlich analysiert. Da es sich bei der Struktur um `element` eine Variable handelt, kann sie im Schema nicht vollständig spezifiziert werden. Um diese Variabilität zu handhaben, werden die Elemente innerhalb des Arrays als Zeichenketten übergeben.

In der Tabellenbeschreibung von finden Sie das Element`Patient`, das als `extension` beschrieben wird`array<string>`, was bedeutet, dass Sie mithilfe eines Indexwerts auf die Elemente des Arrays zugreifen können. Um auf die Elemente der Zeichenfolge zuzugreifen, müssen Sie jedoch verwenden`json_extract`.

Hier ist ein einzelner Eintrag aus dem `extension` Element in der Patiententabelle.

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

Obwohl dies gültiges JSON ist, behandelt Athena es als Zeichenfolge.

Dieses Beispiel für eine SQL-Abfrage zeigt, wie Sie eine Tabelle erstellen können, die die `patient-birthPlace` Elemente `patient-mothersMaidenName` und enthält. Um auf diese Elemente zuzugreifen, müssen Sie verschiedene Array-Indizes verwenden und `json_extract.`

```
SELECT
    extension[1],
    json_extract(extension[1], '$.valueString') AS MothersMaidenName,
    extension[2],
    json_extract(extension[2], '$.valueAddress.city') AS birthPlace
FROM patient
```

Weitere Informationen zu Abfragen, die JSON beinhalten, finden Sie unter Extrahieren von Daten aus JSON im Amazon Athena Athena-Benutzerhandbuch.

Element: birthDate (Age)

Alter ist kein Element des Ressourcentyps Patient in FHIR. Hier sind zwei Beispiele für Suchanfragen, die nach Alter filtern.

Da das Alter kein Element ist, verwenden wir das `birthDate` für die SQL-Abfragen. Um zu sehen, wie ein Element in FHIR aufgenommen wurde, suchen Sie unter Tabellen und Ansichten nach dem Tabellennamen. Sie können sehen, dass es vom Typ Zeichenfolge ist.

Beispiel 1: Berechnung eines Alterswerts

In dieser Beispiel-SQL-Abfrage verwenden wir ein integriertes SQL-Tool, `current_date year` um diese Komponenten zu extrahieren. Dann subtrahieren wir sie, um das tatsächliche Alter eines Patienten in einer Spalte mit dem Namen `age` zurückzugeben.

```
SELECT
 (year(current_date) - year(date(birthdate))) as age
FROM patient
```

Beispiel 2: Filterung nach Patienten, die früher geboren wurden `2019-01-01` und es sind`male`.

Die SQL-Abfrage zeigt Ihnen, wie Sie die `CAST` Funktion verwenden, um das `birthDate` Element als Typ `DATE` umzuwandeln, und wie Sie anhand von zwei Kriterien in der `WHERE` Klausel filtern. Da das Element standardmäßig als Typstring aufgenommen wird, müssen wir `CAST` es

als Typ eingebenDATE. Dann können Sie den < Operator verwenden, um es mit einem anderen Datum zu vergleichen. 2019-01-01 Mithilfe AND von können Sie der WHERE Klausel ein zweites Kriterium hinzufügen.

```
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

Dieses Beispiel zeigt Suchen nach Orten innerhalb des Ressourcentyps Standort, bei denen der Stadtname Attleboro lautet.

```
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

Der Ressourcentyp-Zustand speichert Diagnosedaten zu Problemen, die ein gewisses Maß an Besorgnis ausgelöst haben. HealthLakeDie integrierte medizinische Verarbeitung natürlicher Sprache (NLP) generiert neue Condition Ressourcen auf der Grundlage der im DocumentReference Ressourcentyp enthaltenen Angaben. Wenn neue Ressourcen generiert werden, wird das Tag HealthLake an das Element SYSTEM_GENERATED angehängt. meta Diese SQL-Beispielabfrage zeigt, wie Sie die Bedingungstabelle durchsuchen und Ergebnisse zurückgeben können, bei denen die SYSTEM_GENERATED Ergebnisse entfernt wurden.

Weitere Informationen HealthLake zur integrierten Verarbeitung natürlicher Sprache (NLP) finden Sie unterIntegrierte Verarbeitung natürlicher Sprache (NLP) für HealthLake.

```
SELECT *
```

```
FROM condition
WHERE meta.tag[1] is NULL
```

Sie können auch innerhalb eines bestimmten Zeichenkettenelements suchen, um Ihre Anfrage weiter zu filtern. Das `modifierextension` Element enthält Details darüber, welche `DocumentReference` Ressource zum Generieren einer Reihe von Bedingungen verwendet wurde. Auch hier müssen Sie verwenden, `json_extract` um auf die verschachtelten JSON-Elemente zuzugreifen, die als Zeichenfolge in Athena importiert werden.

Diese SQL-Beispielabfrage zeigt, wie Sie nach all dem suchen können`Condition`, was auf der Grundlage einer bestimmten Abfrage generiert wurde. `DocumentReference` Verwenden Sie diese Option`CAST`, um das JSON-Element als Zeichenfolge festzulegen, sodass Sie `LIKE` es zum Vergleichen verwenden können.

```
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

Der Ressourcentyp Observation speichert Messwerte und einfache Aussagen zu einem Patienten, einem Gerät oder einer anderen Person. HealthLakeDie integrierte Verarbeitung natürlicher Sprache (NLP) generiert neue `Observation` Ressourcen auf der Grundlage von Details, die in einer `DocumentReference` Ressource gefunden wurden. Diese SQL-Beispielabfrage enthält `WHERE meta.tag[1] is NULL` auskommentiert, was bedeutet, dass die `SYSTEM_GENERATED` Ergebnisse enthalten sind.

```
SELECT valueCodeableConcept.coding[1].code
FROM Observation
WHERE  valueCodeableConcept.coding[1].code = '266919005'
-- WHERE meta.tag[1] is NULL
```

Diese Spalte wurde als importiert `struct`. Daher können Sie mithilfe der Punktnotation auf die darin enthaltenen Elemente zugreifen.

Resource type: MedicationStatement

MedicationStatement ist ein FHIR-Ressourcentyp, mit dem Sie Informationen über Medikamente speichern können, die ein Patient eingenommen hat, einnimmt oder in future einnehmen wird. HealthLakeDie integrierte medizinische Verarbeitung natürlicher Sprache (NLP) generiert neue MedicationStatement Ressourcen auf der Grundlage von Dokumenten, die im DocumentReference Ressourcentyp gefunden wurden. Wenn neue Ressourcen generiert werden, wird das Tag HealthLake an das Element `SYSTEM_GENERATED` angehängt. `meta` Diese SQL-Beispielabfrage zeigt, wie eine Abfrage erstellt wird, die anhand eines einzelnen Patienten anhand seiner Kennung filtert und Ressourcen findet, die durch HealthLake das integrierte NLP hinzugefügt wurden.

```
SELECT *
FROM medicationstatement
WHERE meta.tag[1].display = 'SYSTEM_GENERATED' AND subject.reference =
 'Patient/0679b7b7-937d-488a-b48d-6315b8e7003b';
```

Weitere Informationen HealthLake zur integrierten Verarbeitung natürlicher Sprache (NLP) finden Sie unter. Integrierte Verarbeitung natürlicher Sprache (NLP) für HealthLake

Beispiel für SQL-Abfragen mit komplexer Filterung

Die folgenden Beispiele zeigen, wie Amazon Athena SQL-Abfragen mit komplexer Filterung verwendet werden, um FHIR-Daten aus einem HealthLake Datenspeicher zu finden.

Example Erstellen Sie Filterkriterien auf der Grundlage demografischer Daten

Bei der Erstellung einer Patientenkohorte ist es wichtig, die richtigen demografischen Daten der Patienten zu ermitteln. Diese Beispielabfrage zeigt, wie Sie die Trino-Punktnotation verwenden und `json_extract` Daten in Ihrem HealthLake Datenspeicher filtern können.

```
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;
```

Mit der Athena-Konsole können Sie die Ergebnisse weiter sortieren und herunterladen.

Example Erstellen Sie Filter für einen Patienten und die damit verbundenen Erkrankungen

Die folgende Beispielabfrage zeigt, wie Sie alle verwandten Erkrankungen für die in einem HealthLake Datenspeicher gefundenen Patienten suchen und sortieren können.

```
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;
```

Sie können die Athena-Konsole verwenden, um die Ergebnisse weiter zu sortieren oder sie zur weiteren Analyse herunterzuladen.

Example Erstellen Sie Filter für Patienten und die damit verbundenen Beobachtungen

Die folgende Beispielabfrage zeigt, wie alle zugehörigen Beobachtungen für Patienten, die in einem HealthLake Datenspeicher gefunden wurden, gesucht und sortiert werden.

```
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 Erstellen Sie Filterbedingungen für einen Patienten und die zugehörigen Verfahren

Die Verknüpfung von Verfahren mit Patienten ist ein wichtiger Aspekt des Gesundheitswesens. Die folgende SQL-Beispielabfrage zeigt, wie FHIR `Patient` und `Procedure` Ressourcentypen verwendet werden, um dies zu erreichen. Die folgende SQL-Abfrage gibt alle Patienten und die zugehörigen Verfahren zurück, die sich in Ihrem HealthLake Datenspeicher befinden.

```
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;
```

Sie können die Athena-Konsole verwenden, um die Ergebnisse für weitere Analysen herunterzuladen oder sie zu sortieren, um die Ergebnisse besser zu verstehen.

Example Erstellen Sie Filterbedingungen für einen Patienten und die zugehörigen Verschreibungen

Es ist wichtig, eine aktuelle Liste der Medikamente zu sehen, die Patienten einnehmen. Mit Athena können Sie eine SQL-Abfrage schreiben, die sowohl die Ressourcentypen als auch die `Patient` in Ihrem HealthLake Datenspeicher enthaltenen `MedicationRequest` Ressourcentypen verwendet.

Die folgende SQL-Abfrage verknüpft die in Athena importierten `MedicationRequest` Tabellen `Patient` und. Außerdem werden die Rezepte mithilfe der Punktnotation in ihre einzelnen Einträge unterteilt.

```
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
```

Sie können die Athena-Konsole verwenden, um die Ergebnisse zu sortieren oder sie zur weiteren Analyse herunterzuladen.

Example Sehen Sie sich die im **`MedicationStatement`** Ressourcentyp gefundenen Medikamente an

Die folgende Beispielabfrage zeigt Ihnen, wie Sie das verschachtelte JSON organisieren, das mithilfe von SQL in Athena importiert wurde. Die Abfrage verwendet das `meta` FHIR-Element, um anzugeben, wann ein Medikament durch HealthLake die integrierte Verarbeitung natürlicher Sprache (NLP) hinzugefügt wurde. Es wird auch verwendet`json_extract`, um nach Daten innerhalb des Arrays von JSON-Zeichenketten zu suchen. Weitere Informationen finden Sie unter Natürliche Sprachverarbeitung.

```
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;
```

Sie können die Athena-Konsole verwenden, um diese Ergebnisse herunterzuladen oder zu sortieren.

Example Filtern Sie nach einem bestimmten Krankheitstyp

Das Beispiel zeigt, wie Sie eine Gruppe von Patienten im Alter von 18 bis 75 Jahren finden können, bei denen Diabetes diagnostiziert wurde.

```
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%');
```

Jetzt können Sie die Ergebnisse mit der Athena-Konsole sortieren oder zur weiteren Analyse herunterladen.

Überwachung AWS HealthLake

Überwachung und Protokollierung sind wichtige Bestandteile der Aufrechterhaltung der Sicherheit, Zuverlässigkeit, Verfügbarkeit und Leistung von AWS HealthLake. AWS bietet die folgenden Dienste, um zu beobachten HealthLake, zu melden, wenn etwas nicht stimmt, und gegebenenfalls automatische Maßnahmen zu ergreifen.

- AWS CloudTrailerfasst API-Aufrufe und zugehörige Ereignisse, die von oder im Namen Ihres AWS Kontos getätigt wurden, und übermittelt die Protokolldateien an einen von Ihnen angegebenen Amazon S3 S3-Bucket. Sie können feststellen, welche Benutzer und Konten angerufen wurden AWS, von welcher Quell-IP-Adresse aus die Anrufe getätigt wurden und wann die Aufrufe erfolgten. Weitere Informationen finden Sie im AWS CloudTrail -Benutzerhandbuch.
- Amazon CloudWatch überwacht Ihre AWS Ressourcen und die Anwendungen, auf denen Sie laufen, AWS in Echtzeit. Sie können Kennzahlen erfassen und verfolgen, benutzerdefinierte Dashboards erstellen und Alarme festlegen, die Sie benachrichtigen oder Maßnahmen ergreifen, wenn eine bestimmte Metrik einen von Ihnen festgelegten Schwellenwert erreicht. Sie können beispielsweise die CPU-Auslastung oder andere Kennzahlen Ihrer EC2 Amazon-Instances CloudWatch verfolgen und bei Bedarf automatisch neue Instances starten. Weitere Informationen finden Sie im CloudWatch Amazon-Benutzerhandbuch.
- Amazon EventBridge ist ein serverloser Event-Bus-Service, der es einfach macht, Ihre Anwendungen mit Daten aus einer Vielzahl von Quellen zu verbinden. EventBridge liefert einen Stream von Echtzeitdaten aus Ihren eigenen Anwendungen, Software-as-a-Service (SaaS-) Anwendungen und AWS Diensten und leitet diese Daten an Ziele wie Lambda weiter. Auf diese Weise können Sie Ereignisse überwachen, die in Services auftreten, und ereignisgesteuerte Architekturen erstellen. Weitere Informationen finden Sie im EventBridge Amazon-Benutzerhandbuch.

Themen

- Protokollieren von HealthLake API-Aufrufen mit AWS CloudTrail
- HealthLake Metriken mit Amazon überwachen CloudWatch
- HealthLake Ereignisse mithilfe von Amazon überwachen EventBridge

Protokollieren von HealthLake API-Aufrufen mit AWS CloudTrail

AWS HealthLake ist in einen Dienst integriert AWS CloudTrail, der eine Aufzeichnung der Aktionen bereitstellt, die von einem Benutzer, einer Rolle oder einem AWS Dienst ausgeführt wurden HealthLake. CloudTrail erfasst alle API-Aufrufe HealthLake als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der HealthLake Konsole und Codeaufrufen für die HealthLake API-Operationen. Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Bereitstellung von CloudTrail Ereignissen an einen Amazon S3 S3-Bucket aktivieren, einschließlich Ereignissen für HealthLake. Wenn Sie keinen Trail konfigurieren, können Sie die neuesten Ereignisse trotzdem in der CloudTrail Konsole im Ereignisverlauf anzeigen. Anhand der von gesammelten Informationen können Sie die Anfrage ermitteln CloudTrail, an die die Anfrage gestellt wurde HealthLake, die IP-Adresse, von der aus die Anfrage gestellt wurde, wer die Anfrage gestellt hat, wann sie gestellt wurde, und weitere Details.

Weitere Informationen CloudTrail dazu finden Sie im AWS CloudTrail Benutzerhandbuch.

AWS HealthLake Informationen in CloudTrail

CloudTrail ist in Ihrem AWS Konto aktiviert, wenn Sie das Konto erstellen. Wenn Aktivitäten in auftreten HealthLake, wird diese Aktivität zusammen mit anderen AWS Serviceereignissen in der CloudTrail Ereignishistorie in einem Ereignis aufgezeichnet. Sie können die neusten Ereignisse in Ihr AWS -Konto herunterladen und dort suchen und anzeigen. Weitere Informationen finden Sie unter Ereignisse mit dem CloudTrail Ereignisverlauf anzeigen.

Für eine fortlaufende Aufzeichnung der Ereignisse in Ihrem AWS Konto, einschließlich der Ereignisse für HealthLake, erstellen Sie einen Trail. Ein Trail ermöglicht CloudTrail die Übermittlung von Protokolldateien an einen Amazon S3 S3-Bucket. Wenn Sie einen Pfad in der Konsole anlegen, gilt dieser für alle AWS-Regionen. Der Trail protokolliert Ereignisse aus allen Regionen der AWS Partition und übermittelt die Protokolldateien an den von Ihnen angegebenen Amazon S3 S3-Bucket. Darüber hinaus können Sie andere AWS Dienste konfigurieren, um die in den CloudTrail Protokollen gesammelten Ereignisdaten weiter zu analysieren und darauf zu reagieren. Weitere Informationen finden Sie hier:

- Übersicht zum Erstellen eines Trails
- CloudTrail Unterstützte Dienste und Integrationen
- Konfiguration von Amazon SNS SNS-Benachrichtigungen für CloudTrail
- Empfangen von CloudTrail Protokolldateien aus mehreren Regionen und Empfangen von CloudTrail Protokolldateien von mehreren Konten

Alle HealthLake Aktionen werden in der HealthLake API-Referenz CloudTrail und in diesem Entwicklerhandbuch für Aktionen, die mit der FHIR REST-API ausgeführt werden, protokolliert und dokumentiert. Aufrufe der folgenden Aktionen generieren beispielsweise Einträge in den CloudTrail Protokolldateien:

- `DescribeFHIRImportJob`
- `DescribeFHIRExportJob`
- `StartFHIRImportJob`
- `ListFHIRImportJobs`
- `StartFHIRExportJob`
- `ListFHIRExportJobs`
- `CreateFHIRDatastore`
- `ListFHIRDatastores`
- `DeleteFHIRDatastore`
- `DescribeFHIRDatastore`
- `UpdateResource`
- `CreateResource`
- `DeleteResource`
- `ReadResource`
- `GetCapabilities`
- `SearchWithGet`
- `SearchWithPost`

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Die Identitätsinformationen unterstützen Sie bei der Ermittlung der folgenden Punkte:

- Ob die Anfrage mit Root- oder AWS Identity and Access Management (IAM-) Benutzeranmeldedaten gestellt wurde.
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer gesendet wurde.
- Ob die Anfrage von einem anderen AWS Dienst gestellt wurde.

Weitere Informationen finden Sie unter CloudTrail userIdentity-Element.

Grundlegendes zu AWS HealthLake Einträgen in Protokolldateien

Ein Trail ist eine Konfiguration, die die Übertragung von Ereignissen als Protokolldateien an einen von Ihnen angegebenen Amazon S3 S3-Bucket ermöglicht. CloudTrail Protokolldateien enthalten einen oder mehrere Protokolleinträge. Ein Ereignis stellt eine einzelne Anforderung aus einer beliebigen Quelle dar und enthält Informationen über die angeforderte Aktion, Datum und Uhrzeit der Aktion, Anforderungsparameter usw. CloudTrail Protokolldateien sind kein geordneter Stack-Trace der öffentlichen API-Aufrufe, sodass sie nicht in einer bestimmten Reihenfolge angezeigt werden.

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die `CreateFHIRDatastore` Aktion demonstriert.


```
{
    "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"
}
```


HealthLake Metriken mit Amazon überwachen CloudWatch

Sie können die Überwachung HealthLake mithilfe von Amazon CloudWatch durchführen. Amazon sammelt Rohdaten und verarbeitet sie zu lesbaren, nahezu in Echtzeit verfügbaren Metriken. Diese Statistiken werden 15 Monate lang aufbewahrt, sodass Sie diese historischen Informationen nutzen und sich einen besseren Überblick über die Leistung Ihrer Webanwendung oder Ihres Dienstes verschaffen können. Sie können auch Alarme einrichten, die auf bestimmte Grenzwerte achten und Benachrichtigungen senden oder Aktivitäten auslösen, wenn diese Grenzwerte erreicht werden. Weitere Informationen finden Sie im CloudWatch Amazon-Benutzerhandbuch.

> **Note**
>
> Metriken werden für alle systemeigenen HealthLake Aktionen gemeldet.

In den folgenden Tabellen sind HealthLake Metriken und Dimensionen aufgeführt, an die gemeldet wurde CloudWatch. Jede Zahl wird als Frequenzzählung für einen vom Benutzer angegebenen Datenbereich dargestellt.

Die folgenden HealthLake Metriken werden gemeldet. CloudWatch

HealthLake Metriken, die gemeldet wurden CloudWatch

Metrik	Description
Anzahl der Anrufe	Die Anzahl der Anrufe an APIs. Dies kann für das Konto oder einen bestimmten Datenspeicher gemeldet werden. Einheiten: Anzahl Gültige Statistiken: Summe, Anzahl Dimensionen: Vorgang, Datenspeicher-ID, Datenspeichertyp
Erfolgreiche Anfragen	Die Anzahl der erfolgreichen API-Anfragen. Einheiten: Anzahl Gültige Statistiken: Summe, Durchschnitt Dimensionen: Vorgang, Datenspeicher-ID, Datenspeichertyp
Benutzerfehler	Die Anzahl der Anfragen, die aufgrund eines Benutzerfehlers fehlgeschlagen sind. Einheiten: Anzahl Gültige Statistiken: Summe, Durchschnitt Dimensionen: Vorgang, Datenspeicher-ID, Datenspeichertyp
Serverfehler	Die Anzahl der Anfragen, die aufgrund eines Serverfehlers fehlgeschlagen sind.

Metrik	Description
	Einheiten: Anzahl Gültige Statistiken: Summe, Durchschnitt Dimensionen: Vorgang, Datenspeicher-ID, Datenspeichertyp
Gedrosselte Anforderungen	Die Anzahl der Anfragen, die gedrosselt wurden. Diese Metrik ist nicht in der Anzahl der Benutzer- oder Serverfehler enthalten. Einheiten: Anzahl Gültige Statistiken: Summe, Durchschnitt Dimensionen: Vorgang, Datenspeicher-ID, Datenspeichertyp
Latenz	Die Zeit in Millisekunden, die für die Verarbeit ung der Benutzeranfrage benötigt wurde. Einheit: Millisekunden Gültige Statistiken: Minimum, Maximum, Summe, Durchschnitt Abmessungen: Vorgang, Datenspeicher-ID, Datenspeichertyp

Die folgenden HealthLake Dimensionen werden gemeldet CloudWatch.

HealthLake Die Abmessungen wurden gemeldet an CloudWatch

Dimension	Description
Operation	Der in der Anfrage verwendete API-Vorgang

Dimension	Description
DataStoreID	Die in der Anfrage verwendete Datenspeicher-ID
DataStoreType	Der Typ des in der Anfrage verwendeten Datenspeichers

Sie können Metriken für HealthLake mit der AWS-Managementkonsole AWS CLI, der oder der CloudWatch API abrufen. Sie können die CloudWatch API über eines der Amazon AWS Software Development Kits (SDKs) oder die CloudWatch API-Tools verwenden. In der HealthLake Konsole werden Diagramme angezeigt, die auf den Rohdaten der CloudWatch API basieren.

Sie müssen über die entsprechenden CloudWatch Berechtigungen für die Überwachung HealthLake verfügen CloudWatch. Weitere Informationen finden Sie unter Identitäts- und Zugriffsmanagement für Amazon CloudWatch im CloudWatch Amazon-Benutzerhandbuch.

HealthLake Metriken anzeigen

Um Metriken anzuzeigen (CloudWatch Konsole)

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die CloudWatchKonsole.
2. Wählen Sie Metriken, Alle Metriken und dann AWS/ HealthLake aus.
3. Wählen Sie die Dimension, den Namen einer Metrik und schließlich Add to graph (Dem Diagramm hinzufügen) aus.
4. Wählen Sie einen Wert für den Datumsbereich aus. Die Anzahl der Kennzahlen für den ausgewählten Zeitraum wird im Diagramm angezeigt.

Einen Alarm erstellen mit CloudWatch

Ein CloudWatch Alarm überwacht eine einzelne Metrik über einen bestimmten Zeitraum und führt eine oder mehrere Aktionen aus: das Senden einer Benachrichtigung an ein Amazon Simple Notification Service (SNS) -Thema oder an eine Auto Scaling Scaling-Richtlinie. Die Aktion oder Aktionen basieren auf dem Wert der Metrik im Verhältnis zu einem bestimmten Schwellenwert über eine von Ihnen angegebene Anzahl von Zeiträumen. CloudWatch kann Ihnen auch eine SNS-Nachricht senden, wenn sich der Status des Alarms ändert.

> **Note**
>
> CloudWatch Alarme lösen nur dann Aktionen aus, wenn sich der Status ändert und für den von Ihnen angegebenen Zeitraum andauert.

So zeigen Sie Metriken an (Konsole) CloudWatch

1. Melden Sie sich bei der CloudWatch-Konsole an.
2. Wählen Sie Alarms und dann Create Alarm.
3. Wählen Sie AWS/ HealthLake und wählen Sie dann eine Metrik aus.
4. Wählen Sie für Time Range den zu überwachenden Zeitbereich und dann Next.
5. Geben Sie Name (Name) und Description (Beschreibung) ein.
6. Wählen Sie für Whenever >= und geben Sie einen Maximalwert ein.
7. Wenn Sie eine E-Mail senden CloudWatch möchten, wenn der Alarmstatus erreicht ist, wählen Sie im Bereich Aktionen für Wann immer dieser Alarm die Option Status ist ALARM. Wählen Sie unter Benachrichtigung senden an eine Mailingliste aus oder wählen Sie Neue Liste und erstellen Sie eine neue Mailingliste.
8. Nutzen Sie die Alarmvorschau im Bereich Alarm Preview. Wenn Sie mit dem Alarm zufrieden sind, wählen Sie Create Alarm.

HealthLake Ereignisse mithilfe von Amazon überwachen EventBridge

Amazon EventBridge ist ein serverloser Service, der Ereignisse verwendet, um Anwendungskomponenten miteinander zu verbinden, sodass Sie leichter skalierbare, ereignisgesteuerte Anwendungen erstellen können. Die Grundlage von EventBridge ist die Erstellung von Regeln, die Ereignisse an Ziele weiterleiten. AWS HealthLake sorgt für eine dauerhafte Übertragung von Zustandsänderungen an EventBridge. Weitere Informationen finden Sie unter Was ist Amazon EventBridge? im EventBridge Amazon-Benutzerhandbuch.

> **Note**
>
> Informationen zum Senden von HealthLake Veranstaltungen an Amazon EventBridge finden Sie unter EventBridge Amazon-Integration für AWS HealthLake im AWS For Industries-Blog.

Topics

- HealthLake Ereignisse wurden gesendet an EventBridge
- HealthLake Struktur des Ereignisses

HealthLake Ereignisse wurden gesendet an EventBridge

In der folgenden Tabelle sind alle HealthLake Ereignisse aufgeführt, die EventBridge zur Verarbeitung gesendet wurden.

HealthLake Ereignistyp	Status
Ereignisse im Datenspeicher	
Datenspeicher erstellen	`CREATING`
Datenspeicher aktiv	`ACTIVE`
Datenspeicher wird gelöscht	`DELETING`
Datenspeicher wurde gelöscht	`DELETED`
Die Erstellung des Datenspeichers ist fehlgeschlagen	`CREATE_FAILED`
Weitere Informationen finden Sie unter `datastoreStatus` in der AWS HealthLake -API-Referenz.	
Auftragsereignisse importieren	
Job importieren übermittelt	`SUBMITTED`
Job in Bearbeitung importieren	`IN_PROGRESS`

HealthLake Ereignistyp	Status
Importjob wurde mit Fehlern abgeschlossen	`COMPLETED_WITH_ERRORS`
Importauftrag abgeschlossen	`COMPLETED`
Der Importauftrag ist fehlgeschlagen	`FAILED`
Weitere Informationen finden Sie unter `jobStatus` in der AWS HealthLake -API-Referenz.	
Auftragsereignisse exportieren	
Job exportieren übermittelt	`SUBMITTED`
Job in Bearbeitung exportieren	`IN_PROGRESS`
Exportauftrag wurde mit Fehlern abgeschlossen	`COMPLETED_WITH_ERRORS`
Exportauftrag abgeschlossen	`COMPLETED`
Exportauftrag fehlgeschlagen	`FAILED`
Weitere Informationen finden Sie unter `jobStatus` in der AWS HealthLake -API-Referenz.	

HealthLake Struktur des Ereignisses

HealthLake Ereignisse sind Objekte mit JSON-Struktur, die auch Metadatendetails enthalten. Sie können die Metadaten als Eingabe verwenden, um entweder ein Ereignis neu zu erstellen oder weitere Informationen zu erhalten. Alle zugehörigen Metadatenfelder sind in einer Tabelle unter den Codebeispielen in den folgenden Menüs aufgeführt. Weitere Informationen finden Sie unter Metadaten zu AWS Serviceereignissen im EventBridge Amazon-Benutzerhandbuch.

> **Note**
>
> Informationen zum Senden von HealthLake Veranstaltungen an Amazon EventBridge finden Sie unter EventBridge Amazon-Integration für AWS HealthLake im AWS For Industries-Blog.

Ereignisse im Datenspeicher

Data Store Creating

Bundesstaat - **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

Bundesland - **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

Bundesland - **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

Bundesland - **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/"
    }
}
```

Datenspeicher-Ereignisse — Metadaten-Beschreibungen

Name	Typ	Description
`version`	Zeichenfolge	Die Version des EventBridge Ereignisschemas.
`id`	Zeichenfolge	Die für jedes Ereignis generierte UUID der Version 4.
`detail-type`	Zeichenfolge	Der Ereignistyp, der gesendet wird.
`source`	Zeichenfolge	Identifiziert den Service, aus dem das Ereignis stammt.

Name	Typ	Description
`account`	Zeichenfolge	Die 12-stellige AWS-Konto-ID des Datenspeicher-Besitzers.
`time`	Zeichenfolge	Der Zeitpunkt, zu dem das Ereignis eingetreten ist.
`region`	Zeichenfolge	Identifiziert die AWS Region des Datenspeichers.
`resources`	Array (Zeichenfolge)	Ein JSON-Array, das den ARN des Datenspeichers enthält.
`detail`	object	Ein JSON-Objekt, das Informationen zum Ereignis enthält.
`detail.datastoreId`	Zeichenfolge	Die Datenspeicher-ID, die dem Statusänderungsereignis zugeordnet ist.
`detail.datastoreName`	Zeichenfolge	Der Name des Datenspeichers.
`detail.datastoreTypeVersion`	Zeichenfolge	Die FHIR-Version des Datenspeichers.
`detail.datastoreEndpoint`	Zeichenfolge	Der Endpunkt des Datenspeichers.

Job-Ereignisse importieren

Import Job Submitted

Bundesland - **`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

Bundesland - **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

Bundesland - **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

Bundesland - **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

Bundesland - **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/"
        }
    }
}
```

Job-Ereignisse importieren — Metadaten-Beschreibungen

Name	Typ	Description
`version`	Zeichenfolge	Die Version des EventBridge Ereignisschemas.
`id`	Zeichenfolge	Die für jedes Ereignis generierte UUID der Version 4.
`detail-type`	Zeichenfolge	Der Ereignistyp, der gesendet wird.
`source`	Zeichenfolge	Identifiziert den Service, aus dem das Ereignis stammt.
`account`	Zeichenfolge	Die 12-stellige AWS-Konto-ID des Datenspeicher-Besitzers.
`time`	Zeichenfolge	Der Zeitpunkt, zu dem das Ereignis eingetreten ist.
`region`	Zeichenfolge	Identifiziert die AWS Region des Datenspeichers.
`resources`	Array (Zeichenfolge)	Ein JSON-Array, das den ARN des Datenspeichers enthält.
`detail`	object	Ein JSON-Objekt, das Informationen zum Ereignis enthält.

Name	Typ	Description
`detail.jobId`	Zeichenfolge	Die Importauftrags-ID, die dem Statusänderungsereignis zugeordnet ist.
`detail.submitTime`	Zeichenfolge	Die Uhrzeit, zu der der Importauftrag gesendet wurde.
`detail.datastoreId`	Zeichenfolge	Der Datenspeicher, der das Statusänderungsereignis generiert hat.
`detail.inputDataConfig`	Zeichenfolge	Der Eingabepräfixpfad für den Amazon S3 S3-Bucket, der die zu importierenden FHIR-Dateien enthält.

Job-Ereignisse exportieren

Export Job Submitted

Bundesland - **`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

Bundesland - **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

Bundesland - **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

Bundesland - **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

Bundesland - **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/"
        }
    }
}
```

Job-Ereignisse exportieren — Metadaten-Beschreibungen

Name	Typ	Description
`version`	Zeichenfolge	Die Version des EventBridge Ereignisschemas.
`id`	Zeichenfolge	Die für jedes Ereignis generierte UUID der Version 4.
`detail-type`	Zeichenfolge	Der Ereignistyp, der gesendet wird.
`source`	Zeichenfolge	Identifiziert den Service, aus dem das Ereignis stammt.
`account`	Zeichenfolge	Die 12-stellige AWS-Konto-ID des Datenspeicher-Besitzers.
`time`	Zeichenfolge	Der Zeitpunkt, zu dem das Ereignis eingetreten ist.
`region`	Zeichenfolge	Identifiziert die AWS Region des Datenspeichers.
`resources`	Array (Zeichenfolge)	Ein JSON-Array, das den ARN des Datenspeichers enthält.
`detail`	object	Ein JSON-Objekt, das Informationen zum Ereignis enthält.
`detail.jobId`	Zeichenfolge	Die Exportauftrags-ID, die dem Statusänderungsereignis zugeordnet ist.
`detail.submitTime`	Zeichenfolge	Der Zeitpunkt, zu dem der Exportauftrag gesendet wurde.

Name	Typ	Description
`detail.datastoreId`	Zeichenfolge	Der Datenspeicher, der das Statusänderungsereignis generiert hat.
`detail.outputDataConfig`	Zeichenfolge	Der Ausgabepräfixpfad für den Amazon S3 S3-Bucket, der die zu exportierenden FHIR-Dateien enthält.

Sicherheit bei AWS HealthLake

Cloud-Sicherheit AWS hat höchste Priorität. Als AWS Kunde profitieren Sie von einer Rechenzentrums- und Netzwerkarchitektur, die darauf ausgelegt sind, die Anforderungen der sicherheitssensibelsten Unternehmen zu erfüllen.

Sicherheit ist eine gemeinsame Verantwortung von Ihnen AWS und Ihnen. Das Modell der geteilten Verantwortung beschreibt dies als Sicherheit der Cloud und Sicherheit in der Cloud:

- Die Sicherheit der Cloud AWS ist für den Schutz der Infrastruktur verantwortlich, die AWS Dienste in der AWS Cloud ausführt. AWS bietet Ihnen auch Dienste, die Sie sicher nutzen können. Third-party Prüfer testen und verifizieren regelmäßig die Wirksamkeit unserer Sicherheitsmaßnahmen im Rahmen der AWS . Weitere Informationen zu den geltenden Compliance-Programmen finden Sie unter AWS-Services in Umfang nach Compliance-Programm . HealthLake
- Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem AWS Service, den Sie nutzen. Sie sind auch für andere Faktoren verantwortlich, etwa für die Vertraulichkeit Ihrer Daten, für die Anforderungen Ihres Unternehmens und für die geltenden Gesetze und Vorschriften.

Diese Dokumentation hilft Ihnen zu verstehen, wie Sie das Modell der gemeinsamen Verantwortung bei der Nutzung anwenden können HealthLake. In den folgenden Themen erfahren Sie, wie Sie die Konfiguration vornehmen HealthLake , um Ihre Sicherheits- und Compliance-Ziele zu erreichen. Sie lernen auch, wie Sie andere AWS-Services nutzen können, die Sie bei der Überwachung und Sicherung Ihrer HealthLake Ressourcen unterstützen.

Topics

- Datenschutz in AWS HealthLake
- Verschlüsselung im REST für AWS HealthLake
- Verschlüsselung bei der Übertragung für AWS HealthLake
- Identitäts- und Zugriffsmanagement für AWS HealthLake
- Konformitätsprüfung für AWS HealthLake
- Sicherheit der Infrastruktur in AWS HealthLake
- AWS HealthLake Ressourcen erstellen mit AWS CloudFormation
- AWS HealthLake und Schnittstellen-VPC-Endpunkte ()AWS PrivateLink
- Bewährte Sicherheitsmethoden in AWS HealthLake

- Resilienz in AWS HealthLake

Datenschutz in AWS HealthLake

Das Modell der AWS gemeinsamen Verantwortung gilt für den Datenschutz in AWS HealthLake. Wie in diesem Modell beschrieben, AWS ist verantwortlich für den Schutz der globalen Infrastruktur, auf der alle Systeme laufen AWS Cloud. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden Sie unter Häufig gestellte Fragen zum Datenschutz . Informationen zum Datenschutz in Europa finden Sie im General Data Protection Regulation (GDPR) Center.

Aus Datenschutzgründen empfehlen wir, dass Sie Ihre AWS-Konto Anmeldeinformationen schützen und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einrichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor-Authentifizierung (MFA).
- Wird verwendet SSL/TLS , um mit AWS Ressourcen zu kommunizieren. Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein AWS CloudTrail. Informationen zur Verwendung von CloudTrail Pfaden zur Erfassung von AWS Aktivitäten finden Sie unter Arbeiten mit CloudTrail Pfaden im AWS CloudTrail Benutzerhandbuch.
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen darin enthaltenen Standardsicherheitskontrollen AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine API FIPS 140-3-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-3.

Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies gilt auch, wenn Sie

mit der Konsole, der API HealthLake oder den SDKs arbeiten oder diese anderweitig AWS-Services verwenden. AWS CLI AWS Alle Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie eine URL für einen externen Server bereitstellen, empfehlen wir dringend, keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL einzuschließen.

Verschlüsselung im REST für AWS HealthLake

HealthLake bietet standardmäßig Verschlüsselung, um sensible Kundendaten im Speicher zu schützen, indem ein serviceeigener AWS Key Management Service (AWS KMS) -Schlüssel verwendet wird. Customer-managed KMS-Schlüssel werden ebenfalls unterstützt und sind sowohl für den Import als auch für den Export von Dateien aus einem Datenspeicher erforderlich. Weitere Informationen zu Customer-managed KMS Key finden Sie unter Amazon Key Management Service. Kunden können bei der Erstellung eines Datenspeichers einen AWS-eigenen Customer-managed KMS-Schlüssel oder einen KMS-Schlüssel wählen. Die Verschlüsselungskonfiguration kann nicht geändert werden, nachdem ein Datenspeicher erstellt wurde. Wenn ein Datenspeicher einen AWS-eigenen KMS-Schlüssel verwendet, wird dieser als bezeichnet, AWS_OWNED_KMS_KEY und Sie sehen den spezifischen Schlüssel, der für die Verschlüsselung im Ruhezustand verwendet wird, nicht.

AWS-eigener KMS-Schlüssel

HealthLake verwendet diese Schlüssel standardmäßig, um potenziell vertrauliche Informationen wie personenbezogene Daten oder private Gesundheitsinformationen (PHI) im Ruhezustand automatisch zu verschlüsseln. AWS-eigene KMS-Schlüssel werden nicht in Ihrem Konto gespeichert. Sie sind Teil einer Sammlung von KMS-Schlüsseln, die AWS besitzt und für die Verwendung in mehreren AWS-Konten verwaltet. AWS-Services können AWS-eigene KMS-Schlüssel verwenden, um Ihre Daten zu schützen. Sie können AWS-eigene KMS-Schlüssel nicht anzeigen, verwalten, verwenden oder deren Verwendung überprüfen. Sie müssen jedoch keine Arbeit verrichten oder Programme ändern, um die Schlüssel zu schützen, mit denen Ihre Daten verschlüsselt werden.

Ihnen wird keine monatliche Gebühr oder Nutzungsgebühr berechnet, wenn Sie AWS-eigene KMS-Schlüssel verwenden, und diese werden nicht auf die AWS KMS-Kontingente für Ihr Konto angerechnet. Weitere Informationen finden Sie unter AWS-eigene Schlüssel.

Kundenseitig verwaltete KMS-Schlüssel

HealthLake unterstützt die Verwendung eines symmetrischen, vom Kunden verwalteten KMS-Schlüssels, den Sie erstellen, besitzen und verwalten, um eine zweite Verschlüsselungsebene über der vorhandenen AWS-eigenen Verschlüsselung hinzuzufügen. Da Sie die volle Kontrolle über diese Verschlüsselungsebene haben, können Sie beispielsweise folgende Aufgaben ausführen:

- Einrichtung und Pflege wichtiger Richtlinien, IAM-Richtlinien und Zuschüsse
- Kryptographisches Material mit rotierendem Schlüssel
- Aktivieren und Deaktivieren wichtiger Richtlinien
- Hinzufügen von -Tags
- Erstellen von Schlüsselaliasen
- Schlüssel für das Löschen von Schlüsseln planen

Sie können es auch verwenden CloudTrail , um die Anfragen nachzuverfolgen, die in Ihrem Namen HealthLake AWS KMS an gesendet werden. Es AWS KMS fallen zusätzliche Gebühren an. Weitere Informationen finden Sie unter Kundeneigene Schlüssel.

Erstellen eines kundenseitig verwalteten Schlüssels

Sie können einen symmetrischen, vom Kunden verwalteten Schlüssel mithilfe der AWS-Managementkonsole oder der AWS KMS APIs erstellen.

Folgen Sie den Schritten zur Erstellung eines symmetrischen kundenverwalteten Schlüssels im AWS Key Management Service Developer Guide.

Schlüsselrichtlinien steuern den Zugriff auf den vom Kunden verwalteten Schlüssel. Jeder vom Kunden verwaltete Schlüssel muss über genau eine Schlüsselrichtlinie verfügen, die aussagt, wer den Schlüssel wie verwenden kann. Wenn Sie Ihren kundenseitig verwalteten Schlüssel erstellen, können Sie eine Schlüsselrichtlinie angeben. Weitere Informationen finden Sie unter Verwaltung des Zugriffs auf vom Kunden verwaltete Schlüssel im AWS Key Management Service Developer Guide.

Um Ihren vom Kunden verwalteten Schlüssel mit Ihren HealthLake Ressourcen verwenden zu können, müssen kms: CreateGrant operations in der Schlüsselrichtlinie zulässig sein. Dadurch wird einem vom Kunden verwalteten Schlüssel ein Zuschuss hinzugefügt, der den Zugriff auf einen bestimmten KMS-Schlüssel steuert, wodurch ein Benutzer Zugriff auf die für kms:grant-Operationen

erforderlichen Funktionen erhält. HealthLake Weitere Informationen finden Sie unter Zuschüsse verwenden.

Um Ihren vom Kunden verwalteten KMS-Schlüssel mit Ihren HealthLake Ressourcen zu verwenden, müssen die folgenden API-Operationen in der Schlüsselrichtlinie zulässig sein:

- kms: CreateGrant fügt einem bestimmten, vom Kunden verwalteten KMS-Schlüssel Zuschüsse hinzu, der den Zugriff auf Grant-Operationen ermöglicht.
- kms: DescribeKey stellt die vom Kunden verwalteten Schlüsseldetails bereit, die zur Validierung des Schlüssels erforderlich sind. Dies ist für alle Operationen erforderlich.
- kms: GenerateDataKey bietet Zugriff auf verschlüsselte Ressourcen im Ruhezustand für alle Schreibvorgänge.
- kms:Decrypt bietet Zugriff auf Lese- oder Suchvorgänge für verschlüsselte Ressourcen.

Im Folgenden finden Sie ein Beispiel für eine Richtlinienanweisung, die es einem Benutzer ermöglicht, einen Datenspeicher zu erstellen und mit diesem zu interagieren, der mit diesem AWS HealthLake Schlüssel verschlüsselt ist:

```
"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"
            }
```

```
        }
    }
]
```

Erforderliche IAM-Berechtigungen für die Verwendung eines vom Kunden verwalteten KMS-Schlüssels

Beim Erstellen eines Datenspeichers mit aktivierter AWS KMS Verschlüsselung mithilfe eines vom Kunden verwalteten KMS-Schlüssels sind für den Benutzer oder die Rolle, die den HealthLake Datenspeicher erstellt, Berechtigungen sowohl für die Schlüsselrichtlinie als auch für die IAM-Richtlinie erforderlich.

Sie können den ViaService Bedingungsschlüssel kms: verwenden, um die Verwendung des KMS-Schlüssels auf Anfragen zu beschränken, die von HealthLake stammen.

Weitere Informationen zu wichtigen Richtlinien finden Sie unter Enabling IAM-Policies im AWS Key Management Service Developer Guide.

Der IAM-Benutzer, die IAM-Rolle oder das AWS-Konto, das Ihre Repositorys erstellt, muss über die Berechtigungen kms:CreateGrant, kms: GenerateDataKey und kms: sowie DescribeKey über die erforderlichen Berechtigungen verfügen. HealthLake

So HealthLake werden Zuschüsse in AWS KMS verwendet

HealthLake erfordert einen Zuschuss für die Nutzung Ihres vom Kunden verwalteten KMS-Schlüssels. Wenn Sie einen Datenspeicher erstellen, der mit einem vom Kunden verwalteten KMS-Schlüssel verschlüsselt ist, HealthLake erstellt er in Ihrem Namen einen Zuschuss, indem es eine CreateGrantAnfrage an AWS KMS sendet. Zuschüsse in AWS KMS werden verwendet, um HealthLake Zugriff auf einen KMS-Schlüssel in einem Kundenkonto zu gewähren.

Die Zuschüsse, HealthLake die in Ihrem Namen gewährt werden, sollten nicht widerrufen oder zurückgezogen werden. Wenn Sie die HealthLake Genehmigung zur Verwendung der AWS-KMS-Schlüssel in Ihrem Konto widerrufen oder zurückziehen, können Sie HealthLake nicht auf diese Daten zugreifen, neue FHIR-Ressourcen, die an den Datenspeicher übertragen werden, verschlüsseln oder sie entschlüsseln, wenn sie abgerufen werden. Wenn Sie einen Zuschuss für widerrufen oder zurückziehen HealthLake, erfolgt die Änderung sofort. Um die Zugriffsrechte zu entziehen, sollten Sie den Datenspeicher löschen, anstatt die Gewährung zu widerrufen. Wenn ein Datenspeicher gelöscht wird, werden die Zuschüsse HealthLake in Ihrem Namen zurückgezogen.

Überwachen Sie Ihre Verschlüsselungsschlüssel für HealthLake

Sie können CloudTrail damit die Anfragen verfolgen, die in Ihrem Namen HealthLake AWS KMS an gesendet werden, wenn Sie einen vom Kunden verwalteten KMS-Schlüssel verwenden. In den Protokolleinträgen im CloudTrail Protokoll wird healthlake.amazonaws.com im Feld UserAgent angezeigt, um Anfragen von deutlich zu unterscheiden. HealthLake

Bei den folgenden Beispielen handelt es sich um CloudTrail Ereignisse für CreateGrant, GenerateDataKey Decrypt und zur Überwachung von AWS KMS Vorgängen, die aufgerufen werden, DescribeKey um auf Daten zuzugreifen, die mit Ihrem vom Kunden HealthLake verwalteten Schlüssel verschlüsselt wurden.

Im Folgenden wird gezeigt, wie Sie CreateGrant den HealthLake Zugriff auf einen vom Kunden bereitgestellten KMS-Schlüssel ermöglichen, sodass HealthLake dieser KMS-Schlüssel zur Verschlüsselung aller gespeicherten Kundendaten verwendet werden kann.

Benutzer müssen keine eigenen Zuschüsse erstellen. HealthLake erstellt in Ihrem Namen einen Zuschuss, indem Sie eine CreateGrant Anfrage an AWS KMS senden. Zuschüsse AWS KMS werden verwendet, um HealthLake Zugriff auf einen AWS KMS Schlüssel in einem Kundenkonto zu gewähren.

```
         {
"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"
}
```

Die folgenden Beispiele zeigen, wie sichergestellt werden kann GenerateDataKey , dass der Benutzer vor dem Speichern über die erforderlichen Berechtigungen zum Verschlüsseln von Daten verfügt.


```
         {
"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"
}
```

Das folgende Beispiel zeigt, wie der Vorgang Decrypt HealthLake aufgerufen wird, um den gespeicherten verschlüsselten Datenschlüssel für den Zugriff auf die verschlüsselten Daten zu verwenden.

```
         {
    "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"
}
```

Das folgende Beispiel zeigt, wie der DescribeKey Vorgang HealthLake verwendet wird, um zu überprüfen, ob sich der AWS KMS Schlüssel im Besitz des AWS KMS Kunden in einem verwendbaren Zustand befindet, und um einem Benutzer bei der Fehlerbehebung zu helfen, falls er nicht funktioniert.

```
            {
    "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"
}
```

Weitere Informationen

Die folgenden Ressourcen bieten weitere Informationen zur Verschlüsselung von Daten im Ruhezustand.

Weitere Informationen zu den Grundkonzepten von AWS Key Management Service finden Sie in der AWS KMS Dokumentation.

Weitere Informationen zu bewährten Sicherheitsmethoden finden Sie in der AWS KMS Dokumentation.

Verschlüsselung bei der Übertragung für AWS HealthLake

AWS HealthLake verwendet TLS 1.2, um Daten bei der Übertragung über den öffentlichen Endpunkt und über Backend-Dienste zu verschlüsseln.

Identitäts- und Zugriffsmanagement für AWS HealthLake

AWS Identity and Access Management (IAM) hilft einem Administrator AWS-Service , den Zugriff auf Ressourcen sicher zu AWS kontrollieren. IAM-Administratoren kontrollieren, wer authentifiziert (angemeldet) und autorisiert werden kann (über Berechtigungen verfügt), um Ressourcen zu verwenden. HealthLake IAM ist ein Programm AWS-Service , das Sie ohne zusätzliche Kosten nutzen können.

Themen

- Zielgruppe
- Authentifizierung mit Identitäten
- Verwalten des Zugriffs mit Richtlinien
- Wie AWS HealthLake funktioniert mit IAM
- Beispiele für identitätsbasierte Richtlinien für AWS HealthLake
- AWS verwaltete Richtlinien für AWS HealthLake
- Problembehandlung bei AWS HealthLake Identität und Zugriff

Zielgruppe

Wie Sie AWS Identity and Access Management (IAM) verwenden, hängt von Ihrer Rolle ab:

- Servicebenutzer – Fordern Sie von Ihrem Administrator Berechtigungen an, wenn Sie nicht auf Features zugreifen können (siehe Problembehandlung bei AWS HealthLake Identität und Zugriff).
- Serviceadministrator – Bestimmen Sie den Benutzerzugriff und stellen Sie Berechtigungsanfragen (siehe Wie AWS HealthLake funktioniert mit IAM).
- IAM-Administrator – Schreiben Sie Richtlinien zur Zugriffsverwaltung (siehe Beispiele für identitätsbasierte Richtlinien für AWS HealthLake).

Authentifizierung mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten anmelden. Sie müssen sich als IAM-Benutzer authentifizieren oder eine IAM-Rolle annehmen. Root-Benutzer des AWS-Kontos

Sie können sich als föderierte Identität anmelden, indem Sie Anmeldeinformationen aus einer Identitätsquelle wie AWS IAM Identity Center (IAM Identity Center), Single Sign-On-Authentifizierung oder Anmeldeinformationen verwenden. Google/Facebook Weitere Informationen zum Anmelden finden Sie unter So melden Sie sich bei Ihrem AWS-Konto an im Benutzerhandbuch für AWS-Anmeldung .

AWS Bietet für den programmatischen Zugriff ein SDK und eine CLI zum kryptografischen Signieren von Anfragen. Weitere Informationen finden Sie unter AWS Signature Version 4 for API requests im IAM-Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie einen erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, dem sogenannten AWS-Konto Root-Benutzer, der vollständigen Zugriff auf alle AWS-Services Ressourcen hat. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Eine Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter Tasks that require root user credentials im IAM-Benutzerhandbuch.

Verbundidentität

Es hat sich bewährt, dass menschliche Benutzer für den Zugriff AWS-Services mithilfe temporärer Anmeldeinformationen einen Verbund mit einem Identitätsanbieter verwenden müssen.

Eine föderierte Identität ist ein Benutzer aus Ihrem Unternehmensverzeichnis, Ihrem Directory Service Web-Identitätsanbieter oder der AWS-Services mithilfe von Anmeldeinformationen

aus einer Identitätsquelle zugreift. Verbundene Identitäten übernehmen Rollen, die temporäre Anmeldeinformationen bereitstellen.

Für die zentrale Zugriffsverwaltung empfehlen wir AWS IAM Identity Center. Weitere Informationen finden Sie unter Was ist IAM Identity Center? im AWS IAM Identity Center -Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität mit bestimmten Berechtigungen für eine einzelne Person oder Anwendung. Wir empfehlen die Verwendung temporärer Anmeldeinformationen anstelle von IAM-Benutzern mit langfristigen Anmeldeinformationen. Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter Erfordern, dass menschliche Benutzer den Verbund mit einem Identitätsanbieter verwenden müssen, um AWS mithilfe temporärer Anmeldeinformationen darauf zugreifen zu können.

Eine IAM-Gruppe spezifiziert eine Sammlung von IAM-Benutzern und erleichtert die Verwaltung von Berechtigungen für große Gruppen von Benutzern. Weitere Informationen finden Sie unter Anwendungsfälle für IAM-Benutzer im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität mit spezifischen Berechtigungen, die temporäre Anmeldeinformationen bereitstellt. Sie können eine Rolle übernehmen, indem Sie von einer Benutzer- zu einer IAM-Rolle (Konsole) wechseln oder indem Sie eine AWS Oder-API-Operation AWS CLI aufrufen. Weitere Informationen finden Sie unter Methoden, um eine Rolle zu übernehmen im IAM-Benutzerhandbuch.

IAM-Rollen sind nützlich für den Verbundbenutzer-Zugriff, temporäre IAM-Benutzerberechtigungen, kontoübergreifenden Zugriff, serviceübergreifenden Zugriff und Anwendungen, die auf Amazon EC2 laufen. Weitere Informationen finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Verwalten des Zugriffs mit Richtlinien

Sie kontrollieren den Zugriff, AWS indem Sie Richtlinien erstellen und diese an AWS Identitäten oder Ressourcen anhängen. Eine Richtlinie definiert Berechtigungen, wenn sie mit einer Identität oder Ressource verknüpft sind. AWS bewertet diese Richtlinien, wenn ein Principal eine Anfrage stellt. Die meisten Richtlinien werden AWS als JSON-Dokumente gespeichert. Weitere Informationen zu JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Mit Hilfe von Richtlinien legen Administratoren fest, wer Zugriff auf was hat, indem sie definieren, welches Prinzipal welche Aktionen auf welchen Ressourcenund unter welchen Bedingungendurchführen darf.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator erstellt IAM-Richtlinien und fügt sie zu Rollen hinzu, die die Benutzer dann übernehmen können. IAM-Richtlinien definieren Berechtigungen unabhängig von der Methode, die zur Ausführung der Operation verwendet wird.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität (Benutzer, Gruppe oder Rolle) anfügen können. Diese Richtlinien steuern, welche Aktionen Identitäten für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können Inline-Richtlinien (direkt in eine einzelne Identität eingebettet) oder verwaltete Richtlinien (eigenständige Richtlinien, die mit mehreren Identitäten verbunden sind) sein. Informationen dazu, wie Sie zwischen verwalteten und Inline-Richtlinien wählen, finden Sie unter Choose between managed policies and inline policies im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele hierfür sind Vertrauensrichtlinien für IAM-Rollen und Amazon S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben.

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können AWS verwaltete Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Weitere Richtlinientypen

AWS unterstützt zusätzliche Richtlinientypen, mit denen die maximalen Berechtigungen festgelegt werden können, die durch gängigere Richtlinientypen gewährt werden:

- Berechtigungsgrenzen – Eine Berechtigungsgrenze legt die maximalen Berechtigungen fest, die eine identitätsbasierte Richtlinie einer IAM-Entität erteilen kann. Weitere Informationen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im -IAM-Benutzerhandbuch.

- Richtlinien zur Dienstkontrolle (SCPs) — Geben Sie die maximalen Berechtigungen für eine Organisation oder Organisationseinheit in an AWS Organizations. Weitere Informationen finden Sie unter Service-Kontrollrichtlinien im AWS Organizations -Benutzerhandbuch.
- Richtlinien zur Ressourcenkontrolle (RCPs) — Legen Sie die maximal verfügbaren Berechtigungen für Ressourcen in Ihren Konten fest. Weitere Informationen finden Sie im AWS Organizations Benutzerhandbuch unter Richtlinien zur Ressourcenkontrolle (RCPs).
- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die als Parameter übergeben werden, wenn Sie eine temporäre Sitzung für eine Rolle oder einen Verbundbenutzer erstellen. Weitere Informationen finden Sie unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn für eine Anfrage mehrere Arten von Richtlinien gelten, sind die sich daraus ergebenden Berechtigungen schwieriger zu verstehen. Informationen darüber, wie AWS bestimmt wird, ob eine Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, finden Sie unter Bewertungslogik für Richtlinien im IAM-Benutzerhandbuch.

Wie AWS HealthLake funktioniert mit IAM

Bevor Sie IAM zur Verwaltung des Zugriffs auf verwenden, sollten Sie sich darüber informieren HealthLake, mit welchen IAM-Funktionen Sie arbeiten können. HealthLake

IAM-Funktionen, die Sie mit verwenden können AWS HealthLake

IAM-Feature	HealthLake Unterstützung
Identitätsbasierte Richtlinien	Ja
Ressourcenbasierte Richtlinien	Nein
Richtlinienaktionen	Ja
Richtlinienressourcen	Ja
Bedingungsschlüssel für die Richtlinie	Ja
ACLs	Nein
ABAC (Tags in Richtlinien)	Ja

IAM-Feature	HealthLake Unterstützung
Temporäre Anmeldeinformationen	Ja
Prinzipalberechtigungen	Ja
Servicerollen	Ja
Service-verknüpfte Rollen	Nein

Einen allgemeinen Überblick darüber, wie HealthLake und andere AWS Dienste mit den meisten IAM-Funktionen funktionieren, finden Sie im IAM-Benutzerhandbuch unter AWS Dienste, die mit IAM funktionieren.

Identitätsbasierte Richtlinien für AWS HealthLake

Unterstützt Richtlinien auf Identitätsbasis: Ja

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien im IAM-Benutzerhandbuch.

Mit identitätsbasierten IAM-Richtlinien können Sie angeben, welche Aktionen und Ressourcen zugelassen oder abgelehnt werden. Darüber hinaus können Sie die Bedingungen festlegen, unter denen Aktionen zugelassen oder abgelehnt werden. Informationen zu sämtlichen Elementen, die Sie in einer JSON-Richtlinie verwenden, finden Sie in der IAM-Referenz für JSON-Richtlinienelemente im IAM-Benutzerhandbuch.

Beispiele für identitätsbasierte Richtlinien für AWS HealthLake

Beispiele für HealthLake identitätsbasierte Richtlinien finden Sie unter. Beispiele für identitätsbasierte Richtlinien für AWS HealthLake

Ressourcenbasierte Richtlinien finden Sie in AWS HealthLake

Unterstützt ressourcenbasierte Richtlinien: Nein

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Zu den Prinzipalen können Konten, Benutzer, Rollen, Verbundbenutzer oder gehören. AWS-Services

Um kontoübergreifenden Zugriff zu ermöglichen, können Sie ein gesamtes Konto oder IAM-Entitäten in einem anderen Konto als Prinzipal in einer ressourcenbasierten Richtlinie angeben. Weitere Informationen finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Richtlinienaktionen für AWS HealthLake

Unterstützt Richtlinienaktionen: Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das Element `Action` einer JSON-Richtlinie beschreibt die Aktionen, mit denen Sie den Zugriff in einer Richtlinie zulassen oder verweigern können. Nehmen Sie Aktionen in eine Richtlinie auf, um Berechtigungen zur Ausführung des zugehörigen Vorgangs zu erteilen.

Eine Liste der HealthLake Aktionen finden Sie unter Aktionen definiert von AWS HealthLake in der Serviceautorisierungsreferenz.

Bei Richtlinienaktionen wird vor der Aktion das folgende Präfix HealthLake verwendet:

```
healthlake
```

Um mehrere Aktionen in einer einzigen Anweisung anzugeben, trennen Sie jede Aktion durch ein Komma.

```
"Action": [
      "healthlake:action1",
      "healthlake:action2"
```

```
        ]
```

Beispiele für HealthLake identitätsbasierte Richtlinien finden Sie unter. Beispiele für identitätsbasierte Richtlinien für AWS HealthLake

Politische Ressourcen für AWS HealthLake

Unterstützt Richtlinienressourcen: Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das JSON-Richtlinienelement `Resource` gibt die Objekte an, auf welche die Aktion angewendet wird. Als Best Practice geben Sie eine Ressource mit dem zugehörigen Amazon-Ressourcennamen (ARN) an. Verwenden Sie für Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, einen Platzhalter (*), um anzugeben, dass die Anweisung für alle Ressourcen gilt.

```
"Resource": "*"
```

Eine Liste der HealthLake Ressourcentypen und ihrer ARNs Eigenschaften finden Sie unter Resources defined by AWS HealthLake in der Service Authorization Reference. Informationen zu den Aktionen, mit denen Sie den ARN jeder Ressource angeben können, finden Sie unter Aktionen definiert von AWS HealthLake.

Beispiele für HealthLake identitätsbasierte Richtlinien finden Sie unter. Beispiele für identitätsbasierte Richtlinien für AWS HealthLake

Bedingungsschlüssel für Richtlinien für AWS HealthLake

Unterstützt servicespezifische Richtlinienbedingungsschlüssel: Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das Element `Condition` gibt an, wann Anweisungen auf der Grundlage definierter Kriterien ausgeführt werden. Sie können bedingte Ausdrücke erstellen, die Bedingungsoperatoren

verwenden, z. B. ist gleich oder kleiner als, damit die Bedingung in der Richtlinie mit Werten in der Anforderung übereinstimmt. Eine Übersicht aller AWS globalen Bedingungsschlüssel finden Sie unter Kontextschlüssel für AWS globale Bedingungen im IAM-Benutzerhandbuch.

Eine Liste der HealthLake Bedingungsschlüssel finden Sie unter Bedingungsschlüssel für AWS HealthLake in der Service Authorization Reference. Informationen zu den Aktionen und Ressourcen, mit denen Sie einen Bedingungsschlüssel verwenden können, finden Sie unter Aktionen definiert von AWS HealthLake.

Beispiele für HealthLake identitätsbasierte Richtlinien finden Sie unter. Beispiele für identitätsbasierte Richtlinien für AWS HealthLake

Zugriffskontrolllisten () ACLs in AWS HealthLake

Unterstützt ACLs: Nein

Zugriffskontrolllisten (ACLs) steuern, welche Principals (Kontomitglieder, Benutzer oder Rollen) über Zugriffsberechtigungen für eine Ressource verfügen. ACLs ähneln ressourcenbasierten Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Attributbasierte Zugriffskontrolle (ABAC) mit AWS HealthLake

Unterstützt ABAC (Tags in Richtlinien): Ja

Die attributbasierte Zugriffskontrolle (ABAC) ist eine Autorisierungsstrategie, bei der Berechtigungen basierend auf Attributen, auch als Tags bezeichnet, definiert werden. Sie können Tags an IAM-Entitäten und AWS -Ressourcen anhängen und dann ABAC-Richtlinien entwerfen, die Operationen zulassen, wenn das Tag des Prinzipals mit dem Tag auf der Ressource übereinstimmt.

Um den Zugriff auf der Grundlage von Tags zu steuern, geben Sie im Bedingungselement einer Richtlinie Tag-Informationen an, indem Sie die Schlüssel `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`*, oder Bedingung `aws:TagKeys` verwenden.

Wenn ein Service alle drei Bedingungsschlüssel für jeden Ressourcentyp unterstützt, lautet der Wert für den Service Ja. Wenn ein Service alle drei Bedingungsschlüssel für nur einige Ressourcentypen unterstützt, lautet der Wert Teilweise.

Weitere Informationen zu ABAC finden Sie unter Definieren von Berechtigungen mit ABAC-Autorisierung im IAM-Benutzerhandbuch. Um ein Tutorial mit Schritten zur Einstellung von ABAC anzuzeigen, siehe Attributbasierte Zugriffskontrolle (ABAC) verwenden im IAM-Benutzerhandbuch.

Verwenden temporärer Anmeldeinformationen mit AWS HealthLake

Unterstützt temporäre Anmeldeinformationen: Ja

Temporäre Anmeldeinformationen ermöglichen kurzfristigen Zugriff auf AWS Ressourcen und werden automatisch erstellt, wenn Sie einen Verbund verwenden oder die Rollen wechseln. AWS empfiehlt, temporäre Anmeldeinformationen dynamisch zu generieren, anstatt langfristige Zugriffsschlüssel zu verwenden. Weitere Informationen finden Sie unter Temporäre Anmeldeinformationen in IAM und AWS-Services , die mit IAM funktionieren im IAM-Benutzerhandbuch.

Serviceübergreifende Prinzipalberechtigungen für AWS HealthLake

Unterstützt Forward Access Sessions (FAS): Ja

Forward Access Sessions (FAS) verwenden die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, in Kombination mit der Anforderung, Anfragen an nachgelagerte Dienste AWS-Service zu stellen. Einzelheiten zu den Richtlinien für FAS-Anforderungen finden Sie unter Zugriffssitzungen weiterleiten.

Servicerollen für AWS HealthLake

Unterstützt Servicerollen: Ja

Eine Servicerolle ist eine IAM-Rolle, die ein Service annimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.

Informationen zu Servicerollen und den Inline-Richtlinien, die für den vollen Zugriff auf erforderlich sind AWS HealthLake, finden Sie unterEinrichtung AWS HealthLake.

> **Warning**
>
> Das Ändern der Berechtigungen für eine Servicerolle kann zu HealthLake Funktionseinschränkungen führen. Bearbeiten Sie Servicerollen nur, HealthLake wenn Sie dazu eine Anleitung erhalten.

Dienstbezogene Rollen für AWS HealthLake

Unterstützt serviceverknüpfte Rollen: Ja

Eine dienstbezogene Rolle ist eine Art von Servicerolle, die mit einer verknüpft ist. AWS-Service Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Dienstbezogene Rollen werden in Ihrem Dienst angezeigt AWS-Konto und gehören dem Dienst. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.

Details zum Erstellen oder Verwalten von serviceverknüpften Rollen finden Sie unter AWS -Services, die mit IAM funktionieren. Suchen Sie in der Tabelle nach einem Service mit einem `Yes` in der Spalte Service-linked role (Serviceverknüpfte Rolle). Wählen Sie den Link Yes (Ja) aus, um die Dokumentation für die serviceverknüpfte Rolle für diesen Service anzuzeigen.

Beispiele für identitätsbasierte Richtlinien für AWS HealthLake

Benutzer und Rollen haben standardmäßig nicht die Berechtigung, HealthLake-Ressourcen zu erstellen oder zu ändern. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen.

Informationen dazu, wie Sie unter Verwendung dieser beispielhaften JSON-Richtliniendokumente eine identitätsbasierte IAM-Richtlinie erstellen, finden Sie unter Erstellen von IAM-Richtlinien (Konsole) im IAM-Benutzerhandbuch.

Einzelheiten zu Aktionen und Ressourcentypen, die von definiert wurden HealthLake, einschließlich des Formats von ARNs für die einzelnen Ressourcentypen, finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für AWS HealthLake in der Referenz zur Serviceautorisierung.

Themen

- Best Practices für Richtlinien
- Verwenden der Konsole AWS HealthLake
- Zugreifen auf einen AWS HealthLake Datenspeicher in Amazon Athena
- Benutzern die Berechtigung zur Anzeige eigener Berechtigungen erteilen

Best Practices für Richtlinien

Identitätsbasierte Richtlinien legen fest, ob jemand HealthLake Ressourcen in Ihrem Konto erstellen, darauf zugreifen oder sie löschen kann. Dies kann zusätzliche Kosten für Ihr verursachen AWS-Konto. Beachten Sie beim Erstellen oder Bearbeiten identitätsbasierter Richtlinien die folgenden Richtlinien und Empfehlungen:

- Erste Schritte mit AWS verwalteten Richtlinien und Umstellung auf Berechtigungen mit den geringsten Rechten — Verwenden Sie die AWS verwalteten Richtlinien, die Berechtigungen für viele gängige Anwendungsfälle gewähren, um damit zu beginnen, Ihren Benutzern und Workloads Berechtigungen zu gewähren. Sie sind in Ihrem verfügbar. AWS-Konto Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom AWS Kunden verwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind. Weitere Informationen finden Sie unter Von AWS verwaltete Richtlinien oder Von AWS verwaltete Richtlinien für Auftragsfunktionen im IAM-Benutzerhandbuch.
- Anwendung von Berechtigungen mit den geringsten Rechten – Wenn Sie mit IAM-Richtlinien Berechtigungen festlegen, gewähren Sie nur die Berechtigungen, die für die Durchführung einer Aufgabe erforderlich sind. Sie tun dies, indem Sie die Aktionen definieren, die für bestimmte Ressourcen unter bestimmten Bedingungen durchgeführt werden können, auch bekannt als die geringsten Berechtigungen. Weitere Informationen zur Verwendung von IAM zum Anwenden von Berechtigungen finden Sie unter Richtlinien und Berechtigungen in IAM im IAM-Benutzerhandbuch.
- Verwenden von Bedingungen in IAM-Richtlinien zur weiteren Einschränkung des Zugriffs – Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und Ressourcen zu beschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben, um festzulegen, dass alle Anforderungen mithilfe von SSL gesendet werden müssen. Sie können auch Bedingungen verwenden, um Zugriff auf Serviceaktionen zu gewähren, wenn diese für einen bestimmten Zweck verwendet werden AWS-Service, z. CloudFormation B. Weitere Informationen finden Sie unter IAM-JSON-Richtlinienelemente: Bedingung im IAM-Benutzerhandbuch.
- Verwenden von IAM Access Analyzer zur Validierung Ihrer IAM-Richtlinien, um sichere und funktionale Berechtigungen zu gewährleisten – IAM Access Analyzer validiert neue und vorhandene Richtlinien, damit die Richtlinien der IAM-Richtliniensprache (JSON) und den bewährten IAM-Methoden entsprechen. IAM Access Analyzer stellt mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen zur Verfügung, damit Sie sichere und funktionale Richtlinien erstellen können. Weitere Informationen finden Sie unter Richtlinienvalidierung mit IAM Access Analyzer im IAM-Benutzerhandbuch.
- Multi-Faktor-Authentifizierung (MFA) erforderlich — Wenn Sie ein Szenario haben, das IAM-Benutzer oder einen Root-Benutzer in Ihrem System erfordert AWS-Konto, aktivieren Sie MFA für zusätzliche Sicherheit. Um MFA beim Aufrufen von API-Vorgängen anzufordern, fügen Sie Ihren Richtlinien MFA-Bedingungen hinzu. Weitere Informationen finden Sie unter Sicherer API-Zugriff mit MFA im IAM-Benutzerhandbuch.

Weitere Informationen zu bewährten Methoden in IAM finden Sie unter Best Practices für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Verwenden der Konsole AWS HealthLake

Um auf die AWS HealthLake Konsole zugreifen zu können, benötigen Sie ein Mindestmaß an Berechtigungen. Diese Berechtigungen müssen es Ihnen ermöglichen, Details zu den HealthLake Ressourcen in Ihrem aufzulisten und anzuzeigen AWS-Konto. Wenn Sie eine identitätsbasierte Richtlinie erstellen, die strenger ist als die mindestens erforderlichen Berechtigungen, funktioniert die Konsole nicht wie vorgesehen für Entitäten (Benutzer oder Rollen) mit dieser Richtlinie.

Sie müssen Benutzern, die nur die API AWS CLI oder die AWS API aufrufen, keine Mindestberechtigungen für die Konsole gewähren. Stattdessen sollten Sie nur Zugriff auf die Aktionen zulassen, die der API-Operation entsprechen, die die Benutzer ausführen möchten.

Um vollen Zugriff auf zu erhalten HealthLake, fügen Sie einem IAM-Benutzer oder einer IAM-Rolle die folgenden Richtlinien hinzu: `AmazonHealthLakeFullAccess` und`AWSLakeFormationDataAdmin`. Sie müssen auch die HealthLake Inline-Richtlinie anhängen, bei der es sich um eine Servicerolle handelt. Eine Servicerolle ist eine IAM-Rolle, die ein Service annimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch. Hinweise zur Inline-Richtlinie, mit der die erforderliche Servicerolle erstellt wird, finden Sie unterEinrichtung AWS HealthLake. Sie müssen auch die AWS Lake Formation Konsole oder CLI verwenden, um Ihren HealthLake Administrator als AWS Lake Formation Data Lake-Administrator zuzuweisen. Weitere Informationen finden Sie unter Einrichtung AWS HealthLake.

Zugreifen auf einen AWS HealthLake Datenspeicher in Amazon Athena

Wenn Sie Benutzern und Rollen Zugriff auf die HealthLake Datenspeicher in gewähren möchten Amazon Athena, fügen Sie der Rolle oder dem Benutzer die folgenden IAM-Richtlinien hinzu: `AmazonAthenaFullAccess` und`AmazonS3FullAccess`. `Select`und `Describe` Berechtigungen sind auch für Tabellen erforderlich, die von verwaltet werden AWS Lake Formation. AWS Lake Formation Tabellenberechtigungen werden von einem AWS Lake Formation Administrator in der AWS Lake Formation Konsole oder über die CLI erteilt. Weitere Informationen finden Sie unter Einrichtung AWS HealthLake.

Benutzern die Berechtigung zur Anzeige eigener Berechtigungen erteilen

In diesem Beispiel wird gezeigt, wie Sie eine Richtlinie erstellen, die IAM-Benutzern die Berechtigung zum Anzeigen der eingebundenen Richtlinien und verwalteten Richtlinien gewährt, die ihrer Benutzeridentität angefügt sind. Diese Richtlinie umfasst Berechtigungen zum Ausführen dieser Aktion auf der Konsole oder programmgesteuert mithilfe der API AWS CLI oder 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 verwaltete Richtlinien für AWS HealthLake

Eine AWS verwaltete Richtlinie ist eine eigenständige Richtlinie, die von erstellt und verwaltet wird AWS. AWS Verwaltete Richtlinien sind so konzipiert, dass sie Berechtigungen für viele gängige Anwendungsfälle bereitstellen, sodass Sie damit beginnen können, Benutzern, Gruppen und Rollen Berechtigungen zuzuweisen.

Beachten Sie, dass AWS verwaltete Richtlinien für Ihre speziellen Anwendungsfälle möglicherweise keine Berechtigungen mit den geringsten Rechten gewähren, da sie allen AWS Kunden zur Verfügung stehen. Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom Kunden verwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind.

Sie können die in AWS verwalteten Richtlinien definierten Berechtigungen nicht ändern. Wenn die in einer AWS verwalteten Richtlinie definierten Berechtigungen AWS aktualisiert werden, wirkt sich das Update auf alle Prinzidentitäten (Benutzer, Gruppen und Rollen) aus, denen die Richtlinie zugeordnet ist. AWS aktualisiert eine AWS verwaltete Richtlinie höchstwahrscheinlich, wenn eine neue Richtlinie eingeführt AWS-Service wird oder neue API-Operationen für bestehende Dienste verfügbar werden.

Weitere Informationen finden Sie unter Von AWS verwaltete Richtlinien im IAM-Benutzerhandbuch.

AWS verwaltete Richtlinie: AmazonHealthLakeFullAccess

Die `AmazonHealthLakeFullAccess` Richtlinie bietet vollen Zugriff auf HealthLake. Wenn diese Richtlinie ihrem Benutzer oder ihrer Rolle zugewiesen ist, können Benutzer HealthLake damit auf Daten zugreifen, sie abfragen, importieren und exportieren HealthLake. Um viele allgemeine Aktionen in ausführen zu können HealthLake, müssen Sie dem Benutzer oder der Rolle zusätzliche Richtlinien hinzufügen. Weitere Informationen finden Sie unter HealthLake Operationen Einrichtung AWS HealthLake und Berechtigungen.

Sie können die `AmazonHealthLakeFullAccess`-Richtlinie an Ihre IAM-Identitäten anfügen.

Diese Richtlinie gewährt Administrator- und Mitwirkendenberechtigungen, mit HealthLake denen Benutzer und Rollen Abfragen, Suchen, Importieren und Exportieren durchführen können. Außerdem können Aktionen im Namen der Benutzer und Rollen ausgeführt werden, die über diese Berechtigungen verfügen. HealthLake

Details zu Berechtigungen

Diese Richtlinie beinhaltet die folgende Aussage.

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 verwaltete Richtlinie: AmazonHealthLakeReadOnlyAccess

`AmazonHealthLakeReadOnlyAccess`Die Richtlinie gewährt nur Lesezugriff und Berechtigungen für HealthLake und verwandte Ressourcen in anderen AWS Diensten. Wenden Sie diese Richtlinie auf Benutzer an, denen Sie die Möglichkeit gewähren möchten, HealthLake Datenspeicher abzufragen und anzuzeigen, aber nicht die Möglichkeit, sie zu erstellen oder zu ändern.

Sie können die `AmazonHealthLakeReadOnlyAccess`-Richtlinie an Ihre IAM-Identitäten anfügen.

Diese Richtlinie gewährt *`read-only`* Berechtigungen, mit denen Benutzer und Rollen Abfragen durchführen können HealthLake.

Details zu Berechtigungen

Diese Richtlinie beinhaltet die folgende Aussage.

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 Operationen und Genehmigungen

In der folgenden Tabelle sind typische Operationen HealthLake und die zu ihrer Ausführung erforderlichen Berechtigungen aufgeführt.

HealthLake Operationen	Erforderliche Berechtigungen
Erstellen Sie einen Datenspeicher in HealthLake	`AmazonHealthLakeFullAccess` `AmazonLakeFormationDataAdmin` , Inline-Richtlinien und AWS Lake Formation Administratorberechtigungen, verwaltet von AWS Lake Formation
Löschen Sie einen Datenspeicher in HealthLake	`AmazonHealthLakeFullAccess` `AmazonLakeFormationDataAdmin` , Inline-Richtlinien und AWS Lake Formation Administratorberechtigungen, verwaltet von AWS Lake Formation
Einen Datenspeicher auflisten, durchsuchen oder abfragen in HealthLake	`AmazonHealthLakeReadOnlyAccess`
Fragen Sie einen Datenspeicher ab mit Amazon Athena	`AmazonAthenaFullAccess` `AmazonS3FullAccess` , AWS Lake Formation `Select` und `Describe` Berechtigungen für Tabellen, die verwaltet werden von AWS Lake Formation
Daten importieren von HealthLake	Siehe Berechtigungen für Importaufträge einrichten.
Daten exportieren von HealthLake	Siehe Berechtigungen für Exportaufträge einrichten.

HealthLake Aktualisierungen der AWS verwalteten Richtlinien

Hier finden Sie Informationen zu Aktualisierungen AWS verwalteter Richtlinien HealthLake ab dem Zeitpunkt, zu dem dieser Dienst mit der Erfassung dieser Änderungen begann. Abonnieren Sie den RSS-Feed auf der Seite HealthLake Dokumentenverlauf, um automatische Benachrichtigungen über Änderungen an dieser Seite zu erhalten.

Änderungen	Beschreibung	Date
AmazonHealthLakeFullAccess	`AmazonHealthLakeFullAccess` Eine Richtlinie ist erforderlich, um vollen Zugriff auf zu gewähren HealthLake.	14. November 2022
AmazonHealthLakeReadOnlyAccess	`AmazonHealthLakeReadOnlyAccess` Für den schreibgeschützten Zugriff auf ist eine Richtlinie erforderlich. HealthLake	14. November 2022
HealthLake hat begonnen, Änderungen zu verfolgen	HealthLake hat begonnen, Änderungen für die AWS verwalteten Richtlinien zu verfolgen.	14. November 2022

Problembehandlung bei AWS HealthLake Identität und Zugriff

Verwenden Sie die folgenden Informationen, um häufig auftretende Probleme zu diagnostizieren und zu beheben, die bei der Arbeit mit HealthLake und IAM auftreten können.

Themen

- Ich bin nicht berechtigt, eine Aktion durchzuführen in AWS HealthLake
- Ich bin nicht berechtigt, iam durchzuführen: PassRole
- Ich möchte Personen außerhalb meines AWS Kontos den Zugriff auf meine AWS HealthLake Ressourcen ermöglichen

Ich bin nicht berechtigt, eine Aktion durchzuführen in AWS HealthLake

Wenn Ihnen AWS-Managementkonsole mitgeteilt wird, dass Sie nicht berechtigt sind, eine Aktion durchzuführen, müssen Sie sich an Ihren Administrator wenden, um Unterstützung zu erhalten. Ihr Administrator ist die Person, die Ihnen Ihren Benutzernamen und Ihr Passwort zur Verfügung gestellt hat.

Der folgende Beispielfehler tritt auf, wenn der `mateojackson` IAM-Benutzer versucht, die Konsole zum Anzeigen von Details zu einer fiktiven *`my-example-widget`*-Ressource zu verwenden, jedoch nicht über `healthlake:`*`GetWidget`*-Berechtigungen verfügt.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 healthlake:GetWidget on resource: my-example-widget
```

In diesem Fall bittet Mateo seinen Administrator um die Aktualisierung seiner Richtlinien, um unter Verwendung der Aktion *`my-example-widget`* auf die Ressource `healthlake:`*`GetWidget`* zugreifen zu können.

Ich bin nicht berechtigt, iam durchzuführen: PassRole

Wenn Sie die Fehlermeldung erhalten, dass Sie nicht zum Durchführen der `iam:PassRole`-Aktion autorisiert sind, müssen Ihre Richtlinien aktualisiert werden, um eine Rolle an HealthLake übergeben zu können.

Einige AWS-Services ermöglichen es Ihnen, eine bestehende Rolle an diesen Dienst zu übergeben, anstatt eine neue Servicerolle oder eine dienstverknüpfte Rolle zu erstellen. Hierzu benötigen Sie Berechtigungen für die Übergabe der Rolle an den Dienst.

Der folgende Beispielfehler tritt auf, wenn ein IAM-Benutzer mit dem Namen `marymajor` versucht, die Konsole zu verwenden, um eine Aktion in HealthLake auszuführen. Die Aktion erfordert jedoch, dass der Service über Berechtigungen verfügt, die durch eine Servicerolle gewährt werden. Mary besitzt keine Berechtigungen für die Übergabe der Rolle an den Dienst.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In diesem Fall müssen die Richtlinien von Mary aktualisiert werden, um die Aktion `iam:PassRole` ausführen zu können.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich möchte Personen außerhalb meines AWS Kontos den Zugriff auf meine AWS HealthLake Ressourcen ermöglichen

Sie können eine Rolle erstellen, mit der Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation auf Ihre Ressourcen zugreifen können. Sie können festlegen, wem die Übernahme

der Rolle anvertraut wird. Für Dienste, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten (ACLs) unterstützen, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen darüber, ob diese Funktionen HealthLake unterstützt werden, finden Sie unter. Wie AWS HealthLake funktioniert mit IAM
- Informationen dazu, wie Sie Zugriff auf Ihre Ressourcen gewähren können, AWS-Konten die Ihnen gehören, finden Sie im IAM-Benutzerhandbuch unter Gewähren des Zugriffs auf einen IAM-Benutzer in einem anderen AWS-Konto , den Sie besitzen.
- Informationen dazu, wie Sie Dritten Zugriff auf Ihre Ressourcen gewähren können AWS-Konten, finden Sie AWS-Konten im IAM-Benutzerhandbuch unter Gewähren des Zugriffs für Dritte.
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter Gewähren von Zugriff für extern authentifizierte Benutzer (Identitätsverbund) im IAM-Benutzerhandbuch.
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Konformitätsprüfung für AWS HealthLake

Third-party Prüfer bewerten die Sicherheit und Einhaltung von Vorschriften im AWS HealthLake Rahmen mehrerer AWS Compliance-Programme. HealthLake Dazu gehört auch HIPAA.

Informationen darüber, ob AWS-Service ein in den Geltungsbereich bestimmter Compliance-Programme fällt, finden Sie unter AWS-Services Umfang nach Compliance-Programm unter Umfang nach Compliance-Programm AWS-Services das Compliance-Programm aus, an dem Sie interessiert sind. Allgemeine Informationen finden Sie unter AWS Compliance-Programme AWS .

Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter Berichte herunterladen unter .

Ihre Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. Weitere Informationen zu Ihrer Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services finden Sie in der AWS Sicherheitsdokumentation.

Sicherheit der Infrastruktur in AWS HealthLake

Als verwalteter Service AWS HealthLake ist er durch die AWS globalen Netzwerksicherheitsverfahren geschützt, die im Whitepaper Amazon Web Services: Sicherheitsprozesse im Überblick beschrieben sind.

Sie verwenden AWS veröffentlichte API-Aufrufe für den Zugriff HealthLake über das Netzwerk. Kunden müssen Transport Layer Security (TLS) 1.0 oder neuer unterstützen. Wir empfehlen TLS 1.2 oder neuer. Clients müssen außerdem Cipher Suites mit Perfect Forward Secrecy (PFS) wie Ephemeral (DHE) oder Elliptic Curve Ephemeral Diffie-Hellman (ECDHE) unterstützen. Diffie-Hellman Die meisten modernen Systemen wie Java 7 und höher unterstützen diese Modi.

Außerdem müssen Anforderungen mit einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel signiert sein, der einem IAM-Prinzipal zugeordnet ist. Alternativ können Sie mit AWS -Security-Token-Service (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anforderungen zu signieren.

AWS HealthLake Ressourcen erstellen mit AWS CloudFormation

AWS HealthLake ist in einen Service integriert AWS CloudFormation, der Ihnen hilft, Ihre AWS Ressourcen zu modellieren und einzurichten, sodass Sie weniger Zeit mit der Erstellung und Verwaltung Ihrer Ressourcen und Infrastruktur verbringen müssen. Sie erstellen eine Vorlage, die alle AWS benötigten Ressourcen beschreibt und diese Ressourcen für Sie CloudFormation bereitstellt und konfiguriert.

Wenn Sie sie verwenden CloudFormation, können Sie Ihre Vorlage wiederverwenden, um Ihre HealthLake Ressourcen konsistent und wiederholt einzurichten. Beschreiben Sie Ihre Ressourcen einmal und stellen Sie dann dieselben Ressourcen immer wieder in mehreren AWS-Konten Regionen bereit.

HealthLake und CloudFormation Vorlagen

Um Ressourcen für und zugehörige Dienste bereitzustellen HealthLake und zu konfigurieren, müssen Sie sich mit CloudFormation Vorlagen auskennen. Vorlagen sind formatierte Textdateien in JSON oder YAML. Diese Vorlagen beschreiben die Ressourcen, die Sie in Ihren CloudFormation Stacks bereitstellen möchten. Wenn Sie mit JSON oder YAML nicht vertraut sind, können Sie CloudFormation Designer verwenden, um Ihnen die ersten Schritte mit Vorlagen zu erleichtern.

CloudFormation Weitere Informationen finden Sie unter Was ist CloudFormation -Designer? im AWS CloudFormation -Benutzerhandbuch.

> **Note**
>
> AWS HealthLake unterstützt das Erstellen von Datenspeichern mit. CloudFormation Weitere Informationen, einschließlich Beispielen für JSON- und YAML-Vorlagen für die Bereitstellung von HealthLake Datenspeichern, finden Sie in der Referenz zum AWS HealthLake Ressourcentyp im AWS CloudFormation Benutzerhandbuch.

Erfahren Sie mehr über CloudFormation

Weitere Informationen CloudFormation dazu finden Sie in den folgenden Ressourcen:

- AWS CloudFormation
- AWS CloudFormation Benutzerhandbuch
- CloudFormation API Referenz
- AWS CloudFormation Benutzerhandbuch für die Befehlszeilenschnittstelle

AWS HealthLake und Schnittstellen-VPC-Endpunkte ()AWS PrivateLink

Sie können eine private Verbindung zwischen Ihrer VPC und AWS HealthLake durch die Erstellung eines Schnittstellen-VPC-Endpunkts herstellen. Schnittstellen-VPC-Endpunkte basieren auf einer Technologie AWS PrivateLink, die Sie für den privaten Zugriff verwenden können HealthLake, APIs ohne ein Internet-Gateway, ein NAT-Gerät, eine VPN-Verbindung oder Direct Connect eine Verbindung. Instances in Ihrer VPC benötigen keine öffentlichen IP-Adressen für die Kommunikation mit HealthLake;. APIs Der Verkehr zwischen Ihrer VPC und HealthLake; verlässt das Amazon-Netzwerk nicht.

Jeder Schnittstellenendpunkt wird durch eine oder mehrere Elastic-Network-Schnittstellen in Ihren Subnetzen dargestellt.

Weitere Informationen finden Sie unter Interface VPC Endpoints (AWS PrivateLink) im Amazon VPC-Benutzerhandbuch.

Überlegungen zu HealthLake VPC-Endpunkten

Bevor Sie einen Schnittstellen-VPC-Endpunkt für einrichten HealthLake, sollten Sie die Eigenschaften und Einschränkungen der Schnittstellen-Endpunkte im Amazon VPC-Benutzerhandbuch lesen.

HealthLake unterstützt Aufrufe aller API-Aktionen von Ihrer VPC aus.

Erstellen eines VPC-Schnittstellen-Endpunkts für; HealthLake

Sie können einen VPC-Endpunkt für den HealthLake; -Service entweder mit der Amazon VPC-Konsole oder mit AWS Command Line Interface ()AWS CLI erstellen. Weitere Informationen finden Sie unter Erstellung eines Schnittstellenendpunkts im Benutzerhandbuch für Amazon VPC.

Erstellen Sie einen VPC-Endpunkt für HealthLake; mit dem folgenden Dienstnamen:

- com.amazonaws. *region*. Gesundheitssee

Wenn Sie privates DNS für den Endpunkt aktivieren, können Sie API-Anfragen an die HealthLake Verwendung des Standard-DNS-Namens für die Region stellen. Beispiel, *healthlake.us-east-1.amazonaws.com*.

Weitere Informationen finden Sie unter Zugriff auf einen Service über einen Schnittstellenendpunkt im Benutzerhandbuch für Amazon VPC.

Erstellen einer VPC-Endpunktrichtlinie für HealthLake

Sie können eine Endpunktrichtlinie an Ihren VPC-Endpunkt anhängen, der den Zugriff auf HealthLake steuert. Die Richtlinie gibt die folgenden Informationen an:

- Prinzipal, der die Aktionen ausführen kann.
- Aktionen, die ausgeführt werden können
- Die Ressourcen, für die Aktionen ausgeführt werden können.

Weitere Informationen finden Sie unter Steuerung des Zugriffs auf Services mit VPC-Endpunkten im Amazon-VPC-Benutzerhandbuch.

Beispiel: VPC-Endpunktrichtlinie für Aktionen HealthLake

Im Folgenden finden Sie ein Beispiel für eine Endpunktrichtlinie für HealthLake. Wenn diese Richtlinie an einen Endpunkt angehängt ist, gewährt sie allen Prinzipalen auf allen Ressourcen Zugriff auf die HealthLake `CreateFHIRDatastore` Aktion.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "healthlake:create-fhir-datastore"
         ],
         "Resource":"*"
      }
   ]
}
```

Bewährte Sicherheitsmethoden in AWS HealthLake

AWS HealthLake bietet eine Reihe von Sicherheitsfunktionen, die Sie bei der Entwicklung und Implementierung Ihrer eigenen Sicherheitsrichtlinien berücksichtigen sollten. Die folgenden bewährten Methoden sind allgemeine Richtlinien und keine vollständige Sicherheitslösung. Da diese bewährten Methoden für Ihre Umgebung möglicherweise nicht angemessen oder ausreichend sind, sollten Sie sie als hilfreiche Überlegungen und nicht als bindend ansehen.

- Implementieren Sie den Zugriff mit den geringsten Rechten.
- Verwenden Sie nach Möglichkeit Customer-Managed-Keys (CMKs), um Ihre Daten zu verschlüsseln. Weitere Informationen zu CMKs finden Sie unter Amazon Key Management Service.
- Verwenden Sie Search with POST, nicht Search with GET, wenn Sie in Ihrem Datenspeicher nach PHI oder PII suchen.
- Beschränken Sie den Zugriff auf sensible und wichtige Prüfungsfunktionen.
- Verwenden Sie beim Erstellen von Ressourcen über die Aktualisierungs- oder Massenimport-APIs keine PHI oder PII, einschließlich der Namen von Datenspeichern und Aufträgen, in sichtbaren Feldern oder in der logischen FHIR-ID (LID).
- Verwenden Sie beim Senden von Erstellungs-, Lese-, Aktualisierungs-, Lösch- oder Suchanfragen kein PHI im HTTP-Header.

- Aktivieren Sie AWS CloudTrail diese Option, um die AWS HealthLake Nutzung zu überwachen und sicherzustellen, dass keine unerwarteten Aktivitäten auftreten.
- Lesen Sie die bewährten Methoden für die sichere Verwendung von Amazon S3 S3-Buckets. Weitere Informationen finden Sie unter Bewährte Sicherheitsmethoden im Amazon S3 S3-Benutzerhandbuch.

Resilienz in AWS HealthLake

Die AWS globale Infrastruktur basiert auf AWS Regionen und Availability Zones. AWS Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die über Netzwerke mit niedriger Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Zonen ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Falls Sie Ihre Daten oder Anwendungen über größere geografische Distanzen hinweg replizieren müssen, verwenden Sie lokale AWS -Regionen. Eine AWS lokale Region ist ein einzelnes Rechenzentrum, das eine bestehende AWS Region ergänzen soll. Wie alle AWS Regionen sind AWS lokale Regionen vollständig von anderen AWS Regionen isoliert.

Weitere Informationen zu AWS Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.

AWS HealthLake Referenz

Das folgende unterstützende Referenzmaterial ist für SMART auf FHIR, FHIR und verfügbar. AWS HealthLake

> **Note**
>
> Alle systemeigenen HealthLake Aktionen und Datentypen werden in einer separaten Referenz beschrieben. Weitere Informationen finden Sie in der AWS HealthLake -API-Referenz.

Themen

- SMART on FHIR-Unterstützung für AWS HealthLake
- FHIR R4-Unterstützung für AWS HealthLake
- Compliance-Referenz für AWS HealthLake
- Support-Referenz für AWS HealthLake

SMART on FHIR-Unterstützung für AWS HealthLake

Ein auf FHIR basierender HealthLake Datenspeicher mit austauschbaren medizinischen Anwendungen und wiederverwendbaren Technologien (SMART) ermöglicht den Zugriff auf SMART auf FHIR-konformen Anwendungen. HealthLake Der Datenzugriff erfolgt durch Authentifizierung und Autorisierung von Anfragen mithilfe eines Autorisierungsservers eines Drittanbieters. Anstatt die Benutzeranmeldedaten über zu verwalten AWS Identity and Access Management, verwenden Sie einen SMART-On-FHIR-kompatiblen Autorisierungsserver.

> **Note**
>
> HealthLake unterstützt SMART auf den FHIR-Versionen 1.0 und 2.0. Weitere Informationen zu diesen Frameworks finden Sie unter SMART App Launch in der FHIR R4-Dokumentation. HealthLake Datenspeicher unterstützen die folgenden Authentifizierungs- und Autorisierungs-Frameworks für SMART bei FHIR-Anfragen:
>
> - OpenID (AuthN): zur Authentifizierung der Person oder der Client-Anwendung, wer (oder was) sie vorgibt zu sein.

- OAuth 2.0 (AuthZ): zur Autorisierung, auf welche FHIR-Ressourcen in Ihrem HealthLake Datenspeicher eine authentifizierte Anfrage lesen oder schreiben kann. Dies wird durch die Bereiche definiert, die auf Ihrem Autorisierungsserver eingerichtet sind.

Mit dem oder können Sie einen SMART on FHIR-fähigen Datenspeicher erstellen. AWS CLI AWS SDKs Weitere Informationen finden Sie unter Einen HealthLake Datenspeicher erstellen.

Themen

- Erste Schritte mit SMART auf FHIR
- HealthLake Authentifizierungsanforderungen für SMART auf FHIR
- SMART auf FHIR OAuth 2.0-Bereichen, unterstützt von HealthLake
- Token-Validierung mit AWS Lambda
- Verwendung detaillierter Autorisierung mit einem SMART on FHIR-fähigen Datenspeicher HealthLake
- Das SMART on FHIR Discovery-Dokument wird abgerufen
- Eine FHIR-REST-API-Anfrage für einen HealthLake SMART-fähigen Datenspeicher stellen

Erste Schritte mit SMART auf FHIR

In den folgenden Themen wird beschrieben, wie Sie mit SMART bei der FHIR-Autorisierung für beginnen. AWS HealthLake Sie beinhalten die Ressourcen, die Sie in Ihrem AWS Konto bereitstellen müssen, die Erstellung eines SMART on FHIR-fähigen HealthLake Datenspeichers und ein Beispiel dafür, wie eine SMART on FHIR-Clientanwendung mit einem Autorisierungsserver und einem Datenspeicher interagiert. HealthLake

Themen

- Ressourcen für SMART auf FHIR einrichten
- Arbeitsablauf für Client-Anwendungen für SMART auf FHIR

Ressourcen für SMART auf FHIR einrichten

In den folgenden Schritten wird definiert, wie SMART-Anfragen von FHIR bearbeitet werden HealthLake und welche Ressourcen benötigt werden, um erfolgreich zu sein. Die folgenden Elemente arbeiten in einem Workflow zusammen, um eine SMART-on-FHIR-Anfrage zu erstellen:

- Der Endbenutzer: In der Regel ein Patient oder ein Arzt, der eine SMART on FHIR-Anwendung eines Drittanbieters verwendet, um auf Daten in einem Datenspeicher zuzugreifen. HealthLake
- Die SMART on FHIR-Anwendung (als Client-Anwendung bezeichnet): Eine Anwendung, die auf Daten zugreifen möchte, die sich im Datenspeicher befinden. HealthLake
- Der Autorisierungsserver: Ein OpenID Connect-kompatibler Server, der Benutzer authentifizieren und Zugriffstoken ausstellen kann.
- Der HealthLake Datenspeicher: Ein SMART on FHIR-fähiger HealthLake Datenspeicher, der eine Lambda-Funktion verwendet, um auf FHIR-REST-Anfragen zu antworten, die ein Bearer-Token bereitstellen.

Damit diese Elemente zusammenarbeiten können, müssen Sie die folgenden Ressourcen erstellen.

> **Note**
>
> Wir empfehlen, Ihren SMART on FHIR-fähigen HealthLake Datenspeicher zu erstellen, nachdem Sie den Autorisierungsserver eingerichtet, die erforderlichen Bereiche darauf definiert und eine AWS Lambda Funktion zur Token-Introspektion erstellt haben.

1. Richten Sie einen Autorisierungsserver-Endpunkt ein

Um das SMART on FHIR-Framework verwenden zu können, müssen Sie einen Autorisierungsserver eines Drittanbieters einrichten, der FHIR-REST-Anfragen validieren kann, die in einem Datenspeicher gestellt wurden. Weitere Informationen finden Sie unter HealthLake Authentifizierungsanforderungen für SMART auf FHIR.

2. Definieren Sie Bereiche auf Ihrem Autorisierungsserver, um die Zugriffsebenen für HealthLake Datenspeicher zu kontrollieren

Das SMART on FHIR-Framework verwendet OAuth Bereiche, um zu bestimmen, auf welche FHIR-Ressourcen eine authentifizierte Anfrage Zugriff hat und in welchem Umfang. Die Definition von Bereichen ist eine Möglichkeit, beim Design die geringsten Rechte zu berücksichtigen. Weitere Informationen finden Sie unter SMART auf FHIR OAuth 2.0-Bereichen, unterstützt von HealthLake.

3. Richten Sie eine AWS Lambda Funktion ein, die eine Token-Introspektion durchführen kann

Eine FHIR-REST-Anfrage, die von der Client-Anwendung auf einem SMART on FHIR-fähigen Datenspeicher gesendet wird, enthält ein JSON-Web-Token (JWT). Weitere Informationen finden Sie unter Ein JWT dekodieren.

4. Erstellen Sie einen SMART-on-FHIR-fähigen Datenspeicher HealthLake

Um einen SMART on HealthLake FHIR-Datenspeicher zu erstellen, müssen Sie einen angeben. `IdentityProviderConfiguration` Weitere Informationen finden Sie unter Einen HealthLake Datenspeicher erstellen.

Arbeitsablauf für Client-Anwendungen für SMART auf FHIR

Im folgenden Abschnitt wird erklärt, wie Sie im Kontext von SMART auf FHIR eine Client-Anwendung starten und eine erfolgreiche FHIR-REST-Anfrage für einen HealthLake Datenspeicher stellen.

1. Stellen Sie mithilfe der Client-Anwendung eine **`GET`** Anfrage an den Known Uniform Resource Identifier

Eine SMART-fähige Client-Anwendung muss eine `GET` Anfrage stellen, um die Autorisierungsendpunkte Ihres HealthLake Datenspeichers zu finden. Dies erfolgt über eine URI-Anfrage (Known Uniform Resource Identifier). Weitere Informationen finden Sie unter Das SMART on FHIR Discovery-Dokument wird abgerufen.

2. Zugriff und Geltungsbereiche anfordern

Die Client-Anwendung verwendet den Autorisierungsendpunkt des Autorisierungsservers, sodass sich der Benutzer anmelden kann. Dieser Prozess authentifiziert den Benutzer. Bereiche werden verwendet, um zu definieren, auf welche FHIR-Ressourcen in Ihrem HealthLake Datenspeicher eine Client-Anwendung zugreifen kann. Weitere Informationen finden Sie unter SMART auf FHIR OAuth 2.0-Bereichen, unterstützt von HealthLake.

3. Zugriffstoken

Nachdem der Benutzer nun authentifiziert wurde, erhält eine Client-Anwendung ein JWT-Zugriffstoken vom Autorisierungsserver. Dieses Token wird bereitgestellt, wenn die Client-Anwendung eine FHIR-REST-Anfrage an sendet. HealthLake Weitere Informationen finden Sie unter Token-Validierung.

4. Stellen Sie eine FHIR-REST-API-Anfrage auf SMART auf einem FHIR-fähigen Datenspeicher HealthLake

Die Client-Anwendung kann jetzt mithilfe des vom Autorisierungsserver bereitgestellten Zugriffstokens eine FHIR-REST-API-Anfrage an einen HealthLake Datenspeicher-Endpunkt senden. Weitere Informationen finden Sie unter Eine FHIR-REST-API-Anfrage für einen HealthLake SMART-fähigen Datenspeicher stellen.

5. Validieren Sie das JWT-Zugriffstoken

Verwenden Sie eine Lambda-Funktion, um das in der FHIR-REST-Anfrage gesendete Zugriffstoken zu validieren. Weitere Informationen finden Sie unter Token-Validierung mit AWS Lambda.

HealthLake Authentifizierungsanforderungen für SMART auf FHIR

Um auf FHIR-Ressourcen in einem SMART-on-FHIR-fähigen HealthLake Datenspeicher zuzugreifen, muss eine Client-Anwendung von einem OAuth 2.0-kompatiblen Autorisierungsserver autorisiert werden und ein OAuth Bearer-Token als Teil einer FHIR-REST-API-Anforderung vorlegen. Um den Endpunkt des Autorisierungsservers zu finden, verwenden Sie das HealthLake SMART on FHIR Discovery-Dokument über einen Uniform Resource Identifier. `Well-Known` Weitere Informationen zu diesem Prozess finden Sie unter Das SMART on FHIR Discovery-Dokument wird abgerufen.

Wenn Sie einen HealthLake SMART-on-FHIR-Datenspeicher erstellen, müssen Sie den Endpunkt des Autorisierungsservers und den Token-Endpunkt im `metadata` Element der `CreateFHIRDatastore` Anfrage definieren. Weitere Informationen zur Definition des `metadata` Elements finden Sie unterEinen HealthLake Datenspeicher erstellen.

Mithilfe der Endpunkte des Autorisierungsservers authentifiziert die Client-Anwendung einen Benutzer beim Autorisierungsdienst. Nach der Autorisierung und Authentifizierung wird ein JSON Web Token (JWT) vom Autorisierungsdienst generiert und an die Client-Anwendung übergeben. Dieses Token enthält FHIR-Ressourcenbereiche, die die Client-Anwendung verwenden darf, was wiederum einschränkt, auf welche Daten der Benutzer zugreifen kann. Wenn der Startbereich angegeben wurde, enthält die Antwort optional diese Details. Weitere Informationen zu den SMART on FHIR-Bereichen, die von unterstützt werden HealthLake, finden Sie unter. SMART auf FHIR OAuth 2.0-Bereichen, unterstützt von HealthLake

Mithilfe des vom Autorisierungsserver gewährten JWT führt eine Client-Anwendung FHIR-REST-API-Aufrufe an einen SMART on FHIR-fähigen Datenspeicher durch. HealthLake Um das JWT zu validieren und zu dekodieren, müssen Sie eine Lambda-Funktion erstellen. HealthLake ruft diese Lambda-Funktion in Ihrem Namen auf, wenn eine FHIR-REST-API-Anfrage empfangen wird. Ein Beispiel für eine Lambda-Starterfunktion finden Sie unterToken-Validierung mit AWS Lambda.

Elemente des Autorisierungsservers, die zum Erstellen eines SMART on FHIR-fähigen HealthLake Datenspeichers erforderlich sind

In der `CreateFHIRDatastore` Anfrage müssen Sie den Autorisierungsendpunkt und den Token-Endpunkt als Teil des `metadata` Elements im `IdentityProviderConfiguration` Objekt angeben. Sowohl der Autorisierungsendpunkt als auch der Token-Endpunkt sind erforderlich. Ein Beispiel dafür, wie dies in einer `CreateFHIRDatastore` Anfrage angegeben wird, finden Sie unterEinen HealthLake Datenspeicher erstellen.

Erforderliche Ansprüche, um eine FHIR-REST-API-Anfrage in einem SMART on FHIR-fähigen Datenspeicher abzuschließen HealthLake

Ihre AWS Lambda Funktion muss die folgenden Behauptungen enthalten, damit es sich um eine gültige FHIR-REST-API-Anfrage in einem SMART on FHIR-fähigen Datenspeicher handelt. HealthLake

- `nbf`: Anspruch (nicht vorher) — Der Anspruch „nbf“ (nicht vorher) gibt den Zeitpunkt an, bis zu dem das JWT NICHT zur Bearbeitung akzeptiert werden DARF. Die Bearbeitung des Antrags mit der Angabe „Nbf“ setzt voraus, dass der aktuelle Wert hinter dem im date/time Antrag „Nbf“ date/time genannten Wert liegt oder diesem entspricht. Die von uns bereitgestellte Lambda-Beispielfunktion konvertiert `iat` von der Serverantwort in`nbf`.
- `exp`: (Ablaufzeit) -Anspruch — Der Anspruch „exp“ (Ablaufzeit) gibt die Ablaufzeit an oder nach der das JWT nicht zur Bearbeitung akzeptiert werden darf.
- `isAuthorized`: Ein boolescher Wert, der auf gesetzt ist. `True` Zeigt an, dass die Anfrage auf dem Autorisierungsserver autorisiert wurde.
- `aud`: (Zielgruppenanspruch) — Der Anspruch „aud“ (Zielgruppe) identifiziert die Empfänger, für die das JWT bestimmt ist. Dies muss ein SMART on FHIR-fähiger HealthLake Datenspeicher-Endpunkt sein.
- `scope`: Dies muss mindestens ein Bereich sein, der sich auf FHIR-Ressourcen bezieht. Dieser Bereich ist auf Ihrem Autorisierungsserver definiert. Weitere Informationen zu den von akzeptierten Bereichen im Zusammenhang mit FHIR-Ressourcen finden Sie unter HealthLake. Die Ressourcen von SMART auf FHIR umfassen folgende Bereiche HealthLake

SMART auf FHIR OAuth 2.0-Bereichen, unterstützt von HealthLake

HealthLake verwendet OAuth 2.0 als Autorisierungsprotokoll. Wenn Sie dieses Protokoll auf Ihrem Autorisierungsserver verwenden, können Sie HealthLake Datenspeicherberechtigungen (Erstellen, Lesen, Aktualisieren, Löschen und Suchen) für FHIR-Ressourcen definieren, auf die eine Client-Anwendung Zugriff hat.

Das SMART on FHIR-Framework definiert eine Reihe von Bereichen, die vom Autorisierungsserver angefordert werden können. Beispielsweise sollte eine Client-Anwendung, die nur darauf ausgelegt ist, dass Patienten ihre Laborergebnisse oder ihre Kontaktdaten einsehen können, nur berechtigt sein, Bereiche anzufordern`read`.

> **Note**
>
> HealthLake bietet Unterstützung für SMART auf FHIR V1 und V2, wie unten beschrieben. SMART auf FHIR `AuthorizationStrategy`wird bei der Erstellung Ihres Datenspeichers auf einen der folgenden drei Werte gesetzt:
>
> - `SMART_ON_FHIR_V1`— Support nur für SMART auf FHIR V1, einschließlich `read` (Lesen/Suchen) und `write` (create/update/delete) Berechtigungen.
> - `SMART_ON_FHIR`— Support für SMART auf FHIR V1 und V2, einschließlich`create`,`read`, `updatedelete`, und `search` Berechtigungen.
> - `AWS_AUTH`— Die AWS HealthLake Standard-Autorisierungsstrategie; nicht mit SMART auf FHIR verbunden.

Umfang der eigenständigen Markteinführung

HealthLake unterstützt den Bereich des eigenständigen Startmodus`launch/patient`.

Im eigenständigen Startmodus fordert eine Client-Anwendung Zugriff auf die klinischen Daten des Patienten an, da der Benutzer und der Patient der Client-Anwendung nicht bekannt sind. Daher fordert die Autorisierungsanfrage der Client-Anwendung ausdrücklich die Rückgabe des Patientenbereichs an. Nach erfolgreicher Authentifizierung gibt der Autorisierungsserver ein Zugriffstoken aus, das den angeforderten Patientenstartbereich enthält. Der benötigte Patientenkontext wird zusammen mit dem Zugriffstoken in der Antwort des Autorisierungsservers bereitgestellt.

Unterstützte Bereiche für den Startmodus

Scope	Description
`launch/patient`	Ein Parameter in einer OAuth 2.0-Autorisierungsanfrage, der anfordert, dass Patientendaten in der Autorisierungsantwort zurückgegeben werden.

Die Ressourcen von SMART auf FHIR umfassen folgende Bereiche HealthLake

HealthLake definiert drei Ebenen von SMART für FHIR-Ressourcenbereiche.

- `patient`Bereiche gewähren Zugriff auf spezifische Daten über einen einzelnen Patienten.
- `user`Bereiche gewähren Zugriff auf bestimmte Daten, auf die ein Benutzer zugreifen kann.
- `system`Bereiche gewähren Zugriff auf alle FHIR-Ressourcen, die sich im HealthLake Datenspeicher befinden.

In den folgenden Abschnitten ist die Syntax für die Erstellung von FHIR-Ressourcenbereichen mit SMART auf FHIR V1 oder SMART auf FHIR V2 aufgeführt.

> **Note**
>
> Die Autorisierungsstrategie von SMART auf FHIR wird bei der Erstellung Ihres Datenspeichers festgelegt. Weitere Informationen finden Sie unter `AuthorizationStrategy` in der AWS HealthLake -API-Referenz.

SMART auf FHIR V1-Bereichen, unterstützt von HealthLake

Bei der Verwendung von SMART auf FHIR V1 gilt die allgemeine Syntax für die Erstellung von FHIR-Ressourcenbereichen wie folgt. HealthLake Scrollen Sie über die Schaltfläche Kopieren, um den gesamten URL-Pfad im folgenden Beispiel anzuzeigen.

```
('patient' | 'user' | 'system') '/' (fhir-resource | '*') '.' ('read' | 'write' | '*')
```

SMART auf FHIR v1 unterstützte Autorisierungsbereiche

Syntax des Geltungsbereichs	Beispiel für einen Geltungsbereich	Ergebnis
`patient/(fhir-resource | '*').('read' | 'write' | '*')`	`patient/AllergyIntolerance.*`	Die Client-Anwendung für Patienten hat Lese-/Schreibzugriff auf Instanzebene auf alle aufgezeichneten Allergien.
`user/(fhir-resource | '*').('read' | 'write' | '*')`	`user/Observation.read`	Die Benutzer-Client-Anwendung hat Zugriff auf Instanzebene auf read/write alle aufgezeichneten Beobachtungen.
`system/('read' | 'write' | *)`	`system/*.*`	Die System-Client-Anwendung hat read/write Zugriff auf alle FHIR-Ressourcendaten.

SMART auf FHIR V2-Bereichen, unterstützt von HealthLake

Bei der Verwendung von SMART auf FHIR V2 gilt die allgemeine Syntax für die Erstellung von FHIR-Ressourcenbereichen wie folgt. HealthLake Scrollen Sie über die Schaltfläche Kopieren, um den gesamten URL-Pfad im folgenden Beispiel anzuzeigen.

```
('patient' | 'user' | 'system') '/' (fhir-resource | '*') '.' ('c' | 'r' | 'u' | 'd' |
 's')
```

> **Note**
>
> Um SMART auf FHIR V2 zu verwenden, müssen Sie den Wert `permission-v2`in die `capabilities` Metadatenzeichenfolge übergeben, die ein Mitglied des `IdentityProviderConfiguration`Datentyps ist.
> HealthLake unterstützt detaillierte Bereiche. Weitere Informationen finden Sie unter Unterstützte detaillierte Bereiche im FHIR US Core Implementation Guide.

SMART auf FHIR V2 unterstützte Autorisierungsbereiche

Syntax des Geltungsbereichs	Beispiel für einen V1-Bereich	Ergebnis
`patient/Observation.rs`	`user/Observation.read`	Erlaubnis zum Lesen und Durchsuchen der `Observation` Ressource für den aktuellen Patienten.
`system/*.cruds`	`system/*.*`	Die System-Client-Anwendung hat vollen create/read/update/delete/search Zugriff auf alle FHIR-Ressourcendaten.

Token-Validierung mit AWS Lambda

Wenn Sie einen HealthLake SMART on FHIR-fähigen Datenspeicher erstellen, müssen Sie den ARN der AWS Lambda Funktion in der `CreateFHIRDatastore` Anfrage angeben. Der ARN der Lambda-Funktion wird im `IdentityProviderConfiguration` Objekt mithilfe des `IdpLambdaArn` Parameters angegeben.

Sie müssen die Lambda-Funktion erstellen, bevor Sie Ihren SMART on FHIR-fähigen Datenspeicher erstellen. Sobald Sie den Datenspeicher erstellt haben, kann der Lambda-ARN nicht mehr geändert

werden. Verwenden Sie die `DescribeFHIRDatastore` API-Aktion, um den Lambda-ARN zu sehen, den Sie bei der Erstellung des Datenspeichers angegeben haben.

Damit eine FHIR-REST-Anfrage auf einem SMART on FHIR-fähigen Datenspeicher erfolgreich ist, muss Ihre Lambda-Funktion Folgendes tun:

- Gibt in weniger als 1 Sekunde eine Antwort an den HealthLake Datenspeicher-Endpunkt zurück.
- Dekodieren Sie das Zugriffstoken, das im Autorisierungsheader der von der Client-Anwendung gesendeten REST-API-Anforderung bereitgestellt wird.
- Weisen Sie eine IAM-Servicerolle zu, die über ausreichende Berechtigungen verfügt, um die FHIR-REST-API-Anfrage auszuführen.
- Die folgenden Ansprüche sind erforderlich, um eine FHIR-REST-API-Anfrage abzuschließen. Weitere Informationen hierzu finden Sie unter Erforderliche Ansprüche.
  - `nbf`
  - `exp`
  - `isAuthorized`
  - `aud`
  - `scope`

Wenn Sie mit Lambda arbeiten, müssen Sie zusätzlich zu Ihrer Lambda-Funktion eine Ausführungsrolle und eine ressourcenbasierte Richtlinie erstellen. Die Ausführungsrolle einer Lambda-Funktion ist eine IAM-Rolle, die der Funktion die Berechtigung erteilt, auf AWS-Services und -Ressourcen zuzugreifen, die zur Laufzeit benötigt werden. Die von Ihnen bereitgestellte ressourcenbasierte Richtlinie muss es ermöglichen HealthLake , Ihre Funktion in Ihrem Namen aufzurufen.

In den Abschnitten dieses Themas werden eine Beispielanforderung von einer Client-Anwendung und eine dekodierte Antwort, die zum Erstellen einer AWS Lambda-Funktion erforderlichen Schritte und das Erstellen einer ressourcenbasierten Richtlinie beschrieben, die Folgendes voraussetzen kann. HealthLake

- Teil 1: Eine Lambda-Funktion erstellen
- Teil 2: Erstellen einer HealthLake Servicerolle, die von der AWS Lambda-Funktion verwendet wird
- Teil 3: Aktualisierung der Ausführungsrolle der Lambda-Funktion
- Teil 4: Hinzufügen einer Ressourcenrichtlinie zu Ihrer Lambda-Funktion

- Teil 5: Bereitstellung von Parallelität für Ihre Lambda-Funktion

Eine AWS Lambda-Funktion erstellen

Die in diesem Thema erstellte Lambda-Funktion wird ausgelöst, wenn Anfragen an einen SMART on FHIR-fähigen Datenspeicher HealthLake empfangen werden. Die Anfrage von der Client-Anwendung enthält einen REST-API-Aufruf und einen Autorisierungsheader, der ein Zugriffstoken enthält.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
Authorization: Bearer i8hweunweunweofiwweoijewiwe
```

Die Lambda-Beispielfunktion in diesem Thema verwendet AWS Secrets Manager , um Anmeldeinformationen für den Autorisierungsserver zu verschleiern. Wir empfehlen dringend, Anmeldedaten für den Autorisierungsserver nicht direkt in einer Lambda-Funktion anzugeben.

Example Validierung einer FHIR-REST-Anfrage, die ein Autorisierungsträgertoken enthält

Die Lambda-Beispielfunktion zeigt Ihnen, wie Sie eine FHIR-REST-Anfrage validieren, die an einen SMART on FHIR-fähigen Datenspeicher gesendet wurde. step-by-stepsAnweisungen zur Implementierung dieser Lambda-Funktion finden Sie unterErstellen einer Lambda-Funktion mit dem AWS-Managementkonsole.

Wenn die FHIR-REST-API-Anforderung keinen gültigen Datenspeicher-Endpunkt, kein Zugriffstoken und keinen gültigen REST-Vorgang enthält, schlägt die Lambda-Funktion fehl. Weitere Informationen zu den erforderlichen Autorisierungsserverelementen finden Sie unter. Erforderliche Ansprüche

```
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()
```

Erstellen einer Lambda-Funktion mit dem AWS-Managementkonsole

Beim folgenden Verfahren wird davon ausgegangen, dass Sie die Servicerolle, die Sie bei der Bearbeitung einer FHIR-REST-API-Anfrage in einem SMART on FHIR-fähigen Datenspeicher übernehmen möchten HealthLake , bereits erstellt haben. Wenn Sie die Servicerolle nicht erstellt haben, können Sie trotzdem die Lambda-Funktion erstellen. Sie müssen den ARN der Servicerolle hinzufügen, damit die Lambda-Funktion funktioniert. Weitere Informationen zum Erstellen einer Servicerolle und deren Angabe in der Lambda-Funktion finden Sie unter Erstellen einer HealthLake Servicerolle zur Verwendung in der AWS Lambda-Funktion, die zum Dekodieren eines JWT verwendet wird

Um eine Lambda-Funktion ()AWS-Managementkonsole zu erstellen

1. Öffnen Sie die Seite Funktionen der Lambda-Konsole.
2. Wählen Sie Funktion erstellen.
3. Wählen Sie Verfassen von Grund auf aus.
4. Geben Sie unter Basisinformationen einen Funktionsnamen ein. Wählen Sie unter Runtime eine Python-basierte Runtime aus.
5. Wählen Sie für Execution role (Ausführungsrolle) die Option Create a new role with basic Lambda permissions (Neue Rolle mit grundlegenden Lambda-Berechtigungen erstellen) aus.

   Lambda erstellt eine Ausführungsrolle, die der Funktion die Berechtigung zum Hochladen von Protokollen auf Amazon CloudWatch erteilt. Die Lambda-Funktion übernimmt die Ausführungsrolle, wenn Sie Ihre Funktion aufrufen, und verwendet die Ausführungsrolle, um Anmeldeinformationen für das AWS SDK zu erstellen.
6. Wählen Sie die Registerkarte Code und fügen Sie die Lambda-Beispielfunktion hinzu.

   Wenn Sie die Servicerolle für die zu verwendende Lambda-Funktion noch nicht erstellt haben, müssen Sie sie erstellen, bevor die Lambda-Beispielfunktion funktioniert. Weitere Informationen zum Erstellen einer Servicerolle für die Lambda-Funktion finden Sie unterErstellen einer HealthLake Servicerolle zur Verwendung in der AWS Lambda-Funktion, die zum Dekodieren eines JWT verwendet wird.

   ```
   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()
```

Die Ausführungsrolle einer Lambda-Funktion ändern

Nachdem Sie die Lambda-Funktion erstellt haben, müssen Sie die Ausführungsrolle aktualisieren, sodass sie die erforderlichen Berechtigungen zum Aufrufen von Secrets Manager enthält. In Secrets Manager hat jedes Secret, das Sie erstellen, einen ARN. Um die geringste Berechtigung anzuwenden, sollte die Ausführungsrolle nur Zugriff auf die Ressourcen haben, die für die Ausführung der Lambda-Funktion benötigt werden.

Sie können die Ausführungsrolle einer Lambda-Funktion ändern, indem Sie in der IAM-Konsole danach suchen oder in der Lambda-Konsole Konfiguration auswählen. Weitere Informationen zur Verwaltung Ihrer Ausführungsrolle für Lambda-Funktionen finden Sie unterLambda-Ausführungsrolle.

Example Lambda-Funktionsausführungsrolle, die Zugriff gewährt auf **`GetSecretValue`**

Durch das Hinzufügen der IAM-Aktion `GetSecretValue` zur Ausführungsrolle wird die erforderliche Berechtigung erteilt, damit die Lambda-Beispielfunktion funktioniert.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:secret-name-DKodTA"
```

```
        }
    ]
}
```

Zu diesem Zeitpunkt haben Sie eine Lambda-Funktion erstellt, mit der Sie das Zugriffstoken validieren können, das als Teil der FHIR-REST-Anfrage an Ihren SMART on FHIR-fähigen Datenspeicher gesendet wurde.

Erstellen einer HealthLake Servicerolle zur Verwendung in der AWS Lambda-Funktion, die zum Dekodieren eines JWT verwendet wird

Persona: IAM-Administrator

Ein Benutzer, der IAM-Richtlinien hinzufügen oder entfernen und neue IAM-Identitäten erstellen kann.

Service – -Rolle

Eine Servicerolle ist eine IAM-Rolle, die ein Service annimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.

Nachdem das JSON Web Token (JWT) dekodiert wurde, benötigt Lambda die Autorisierung, um auch einen IAM-Rollen-ARN zurückzugeben. Diese Rolle muss über die erforderlichen Berechtigungen verfügen, um die REST-API-Anfrage auszuführen. Andernfalls schlägt sie aufgrund unzureichender Berechtigungen fehl.

Bei der Einrichtung einer benutzerdefinierten Richtlinie mithilfe von IAM empfiehlt es sich, die erforderlichen Mindestberechtigungen zu gewähren. Weitere Informationen finden Sie unter Anwenden von Berechtigungen mit den geringsten Rechten im IAM-Benutzerhandbuch.

Das Erstellen einer HealthLake Servicerolle, die in der Autorisierungs-Lambda-Funktion festgelegt werden soll, erfordert zwei Schritte.

- Zunächst müssen Sie eine IAM-Richtlinie erstellen. Die Richtlinie muss den Zugriff auf die FHIR-Ressourcen spezifizieren, für die Sie Bereiche auf dem Autorisierungsserver bereitgestellt haben.

- Zweitens müssen Sie die Servicerolle erstellen. Wenn Sie die Rolle erstellen, legen Sie eine Vertrauensbeziehung fest und fügen die Richtlinie hinzu, die Sie in Schritt eins erstellt haben. Die Vertrauensstellung wird als HealthLake Dienstprinzipal bezeichnet. In diesem Schritt müssen Sie einen HealthLake Datenspeicher-ARN und eine AWS Konto-ID angeben.

Eine neue IAM-Richtlinie erstellen

Die Bereiche, die Sie auf Ihrem Autorisierungsserver definieren, bestimmen, auf welche FHIR-Ressourcen ein authentifizierter Benutzer in einem Datenspeicher zugreifen kann. HealthLake

Die von Ihnen erstellte IAM-Richtlinie kann an die von Ihnen definierten Bereiche angepasst werden.

Die folgenden Aktionen können im `Action` Element einer IAM-Richtlinienerklärung definiert werden. Für jede `Action` in der Tabelle können Sie eine `Resource types` definieren. In HealthLake einem Datenspeicher befindet sich der einzige unterstützte Ressourcentyp, der im `Resource` Element einer IAM-Berechtigungsrichtlinie definiert werden kann.

Einzelne FHIR-Ressourcen sind keine Ressource, die Sie als Element in einer IAM-Berechtigungsrichtlinie definieren können.

Aktionen, definiert von HealthLake

Aktionen	Beschreibung	Zugrif bene	Ressourcentyp (erforderlich)
CreateRes ource	Erteilt die Erlaubnis, eine Ressource zu erstellen	Schre	Datenspeicher-ARN: arn:aws:healthlake : ::datastore/fhir/ **your-region 111122223 333 your-datastore-id**
DeleteRes ource	Gewährt die Berechtig ung zum Löschen von Ressourcen	Schre	Datenspeicher-ARN: arn:aws:healthlake : ::datastore/fhir/ **your-region 111122223 333 your-datastore-id**
ReadResou rce	Gewährt die Berechtig ung zum Lesen von Ressourcen	Leser	Datenspeicher-ARN: arn:aws:healthlake : ::datastore/fhir/ **your-region 111122223 333 your-datastore-id**
SearchWit hGet	Gewährt die Berechtig ung, Ressourcen mit	Leser	Datenspeicher-ARN: arn:aws:healthlake : ::datastore/fhir/ **your-region 111122223 333 your-datastore-id**

Aktionen	Beschreibung	Zugrif bene	Ressourcentyp (erforderlich)
	der GET-Methode zu durchsuchen		
SearchWithPost	Gewährt die Berechtigung, Ressourcen mit der POST-Methode zu durchsuchen	Leser	Datenspeicher-ARN: arn:aws:healthlake: ::datastore/fhir/ **your-region 111122223333 your-datastore-id**
Starten FHIRExport JobWithPost	Erteilt die Erlaubnis, einen FHIR-Exportauftrag mit GET zu starten	Schre	Datenspeicher-ARN: arn:aws:healthlake: ::datastore/fhir/ **your-region 111122223333 your-datastore-id**
UpdateResource	Gewährt die Berechtigung zum Aktualisieren von Ressourcen	Schre	Datenspeicher-ARN: arn:aws:healthlake: ::datastore/fhir/ **your-region 111122223333 your-datastore-id**

Um zu beginnen, können Sie verwenden. `AmazonHealthLakeFullAccess` Diese Richtlinie würde Lese-, Schreib-, Such- und Exportvorgänge für alle FHIR-Ressourcen in einem Datenspeicher zulassen. Um nur Leseberechtigungen für die Verwendung eines Datenspeichers zu gewähren. `AmazonHealthLakeReadOnlyAccess`

Weitere Informationen zum Erstellen einer benutzerdefinierten Richtlinie mithilfe von AWS-Managementkonsole AWS CLI, oder IAM SDKs finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

Eine Servicerolle für HealthLake (IAM-Konsole) erstellen

Gehen Sie wie folgt vor, um eine Servicerolle zu erstellen. Wenn Sie einen Dienst erstellen, müssen Sie auch eine IAM-Richtlinie festlegen.

Um die Servicerolle für HealthLake (IAM-Konsole) zu erstellen

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die IAM-Konsole unter. https://console.aws.amazon.com/iam/

2. Wählen Sie im Navigationsbereich der IAM Console Roles (Rollen) aus.
3. Wählen Sie anschließend Create role (Rolle erstellen) aus.
4. Wählen Sie auf der Seite Vertrauensentität auswählen die Option Benutzerdefinierte Vertrauensrichtlinie aus.
5. Aktualisieren Sie anschließend unter Benutzerdefinierte Vertrauensrichtlinie die Beispielrichtlinie wie folgt. **`your-account-id`**Ersetzen Sie es durch Ihre Kontonummer und fügen Sie den ARN des Datenspeichers hinzu, den Sie für Ihre Import- oder Exportaufträge verwenden möchten.

   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. Wählen Sie anschließend Weiter.
7. Wählen Sie auf der Seite „Berechtigungen hinzufügen“ die Richtlinie aus, die der HealthLake Dienst annehmen soll. Um Ihre Richtlinie zu finden, suchen Sie unter Berechtigungsrichtlinien danach.
8. Wählen Sie dann Richtlinie anhängen aus.
9. Geben Sie dann auf der Seite Name, Überprüfung und Erstellung unter Rollenname einen Namen ein.

10. (Optional) Fügen Sie anschließend unter Beschreibung eine kurze Beschreibung für Ihre Rolle hinzu.
11. Geben Sie möglichst einen Rollennamen oder ein Rollennamen-Suffix ein, mit dem der Zweck dieser Rolle einfach zu erkennen ist. Rollennamen müssen innerhalb Ihres AWS-Konto-Kontos eindeutig sein. Sie werden nicht nach Groß- und Kleinschreibung unterschieden. z. B. können Sie keine Rollen erstellen, die **`PRODROLE`** bzw. **`prodrole`** heißen. Da möglicherweise verschiedene Entitäten auf die Rolle verweisen, kann der Rollenname nach der Erstellung nicht bearbeitet werden.
12. Überprüfen Sie die Rollendetails und wählen Sie dann Rolle erstellen aus.

Informationen zur Angabe der Rolle ARN in der Lambda-Beispielfunktion finden Sie unterEine AWS Lambda-Funktion erstellen.

Lambda-Ausführungsrolle

Die Ausführungsrolle einer Lambda-Funktion ist eine IAM-Rolle, die der Funktion die Berechtigung zum Zugriff auf AWS Dienste und Ressourcen erteilt. Diese Seite enthält Informationen zum Erstellen, Anzeigen und Verwalten der Ausführungsrolle einer Lambda-Funktion.

Standardmäßig erstellt Lambda eine Ausführungsrolle mit minimalen Berechtigungen, wenn Sie eine neue Lambda-Funktion mit dem erstellen. AWS-Managementkonsole Informationen zur Verwaltung der in der Ausführungsrolle gewährten Berechtigungen finden Sie unter Erstellen einer Ausführungsrolle in der IAM-Konsole im Lambda Developer Guide.

Die in diesem Thema vorgestellte Lambda-Beispielfunktion verwendet Secrets Manager, um die Anmeldeinformationen des Autorisierungsservers zu verschleiern.

Wie bei jeder IAM-Rolle, die Sie erstellen, ist es wichtig, die bewährte Methode mit den geringsten Rechten zu befolgen. Während der Entwicklungsphase können Sie manchmal Berechtigungen gewähren, die über das hinausgehen, was erforderlich ist. Bevor Sie Ihre Funktion in der Produktionsumgebung veröffentlichen, sollten Sie als bewährte Methode die Richtlinie so anpassen, dass sie nur die erforderlichen Berechtigungen enthält. Weitere Informationen finden Sie unter Anwenden der geringsten Rechte im IAM-Benutzerhandbuch.

Erlaube HealthLake , deine Lambda-Funktion auszulösen

Um die Lambda-Funktion in Ihrem Namen aufrufen zu HealthLake können, müssen Sie Folgendes tun:

- Sie müssen den ARN der Lambda-Funktion, die Sie in der Anfrage aufrufen HealthLake möchten, `IdpLambdaArn` gleich setzen. `CreateFHIRDatastore`
- Sie benötigen eine ressourcenbasierte Richtlinie, die es Ihnen ermöglicht HealthLake , die Lambda-Funktion in Ihrem Namen aufzurufen.

Wenn es eine FHIR-REST-API-Anfrage auf einem SMART on FHIR-fähigen Datenspeicher HealthLake empfängt, benötigt es Berechtigungen, um die Lambda-Funktion aufzurufen, die bei der Erstellung des Datenspeichers in Ihrem Namen angegeben wurde. Um HealthLake Zugriff zu gewähren, verwenden Sie eine ressourcenbasierte Richtlinie. Weitere Informationen zum Erstellen einer ressourcenbasierten Richtlinie für eine Lambda-Funktion finden Sie unter Zulassen, dass ein AWS Dienst eine Lambda-Funktion aufruft im Entwicklerhandbuch.AWS Lambda

Bereitstellung von Parallelität für Ihre Lambda-Funktion

> **Important**
>
> HealthLake erfordert, dass die maximale Laufzeit für Ihre Lambda-Funktion weniger als eine Sekunde (1000 Millisekunden) beträgt.
> Wenn Ihre Lambda-Funktion das Laufzeitlimit überschreitet, erhalten Sie eine TimeOutAusnahme.

Um zu vermeiden, dass diese Ausnahme auftritt, empfehlen wir, die bereitgestellte Parallelität zu konfigurieren. Wenn Sie Provisioned Concurrency (Bereitgestellte Gleichzeitigkeit) vor einer Erhöhung der Aufrufe zuweisen, können Sie sicherstellen, dass alle Anforderungen von initialisierten Instances mit geringer Latenz verarbeitet werden. Weitere Informationen zur Konfiguration der bereitgestellten Parallelität finden Sie unter Konfiguration der bereitgestellten Parallelität im Lambda Developer Guide

Um die durchschnittliche Laufzeit Ihrer Lambda-Funktion derzeit zu sehen, verwenden Sie die Monitoring-Seite für Ihre Lambda-Funktion in der Lambda-Konsole. Standardmäßig bietet die Lambda-Konsole ein Dauer-Diagramm, das Ihnen die durchschnittliche, minimale und maximale Zeit anzeigt, die Ihr Funktionscode mit der Verarbeitung eines Ereignisses verbringt. Weitere Informationen zur Überwachung von Lambda-Funktionen finden Sie unter Überwachungsfunktionen in der Lambda-Konsole im Lambda Developer Guide.

Wenn Sie bereits Parallelität für Ihre Lambda-Funktion bereitgestellt haben und diese überwachen möchten, finden Sie weitere Informationen unter Überwachung der Parallelität im Lambda Developer Guide.

Verwendung detaillierter Autorisierung mit einem SMART on FHIR-fähigen Datenspeicher HealthLake

Geltungsbereiche allein bieten Ihnen nicht die nötige Genauigkeit darüber, auf welche Daten ein Anforderer in einem Datenspeicher zugreifen darf. Die Verwendung einer differenzierten Autorisierung ermöglicht ein höheres Maß an Spezifität bei der Gewährung des Zugriffs auf einen SMART-on-FHIR-fähigen Datenspeicher. HealthLake Um eine differenzierte Autorisierung zu verwenden, legen Sie `True` im Parameter Ihrer Anfrage den `FineGrainedAuthorizationEnabled` Wert „gleich“ fest. `IdentityProviderConfiguration` `CreateFHIRDatastore`

Wenn Sie die differenzierte Autorisierung aktiviert haben, gibt Ihr Autorisierungsserver `id_token` zusammen mit dem Zugriffstoken einen `fhirUser` Bereich zurück. Dadurch können Informationen über den Benutzer von der Client-Anwendung abgerufen werden. Die Client-Anwendung sollte den `fhirUser` Anspruch als URI einer FHIR-Ressource behandeln, die den aktuellen Benutzer repräsentiert. Mögliche Werte sind `Patient`, `Practitioner` oder `RelatedPerson`. Die Antwort des Autorisierungsservers beinhaltet auch einen `user/` Bereich, der definiert, auf welche Daten der Benutzer zugreifen kann. Dabei wird die Syntax verwendet, die für Bereiche definiert ist, die sich auf FHIR-ressourcenspezifische Bereiche beziehen:

```
user/(fhir-resource | '*').('read' | 'write' | '*')
```

Im Folgenden finden Sie Beispiele dafür, wie eine differenzierte Autorisierung verwendet werden kann, um FHIR-Ressourcentypen im Zusammenhang mit dem Datenzugriff weiter zu spezifizieren.

- Wann bestimmt `fhirUser` eine differenzierte Autorisierung`Practitioner`, auf welche Patientengruppe der Benutzer zugreifen kann. Der Zugang `fhirUser` ist nur für Patienten erlaubt, auf die sich der Patient `fhirUser` als Allgemeinmediziner bezieht.

  ```
  Patient.generalPractitioner : [{Reference(Practitioner)}]
  ```

- Wann `fhirUser` ist ein `Patient` oder `RelatedPerson` und der Patient, auf den in der Anfrage verwiesen wird, unterscheidet sich von der Autorisierung`fhirUser`, zu der der angefragte Patient

Zugang `fhirUser` hat. Der Zugriff ist zulässig, wenn in der angeforderten `Patient` Ressource eine Beziehung angegeben ist.

```
Patient.link.other : {Reference(Patient|RelatedPerson)}
```

Das SMART on FHIR Discovery-Dokument wird abgerufen

SMART definiert ein Discovery-Dokument, das es Kunden ermöglicht, den Autorisierungsendpunkt zu ermitteln, URLs und bietet Funktionen, die ein HealthLake Datenspeicher unterstützt. Diese Informationen helfen Kunden dabei, Autorisierungsanfragen an den richtigen Endpunkt weiterzuleiten und Autorisierungsanfragen zu erstellen, die der HealthLake Datenspeicher unterstützt.

Damit eine Client-Anwendung erfolgreich eine FHIR-REST-Anfrage stellen kann HealthLake, muss sie die vom HealthLake Datenspeicher definierten Autorisierungsanforderungen erfassen. Ein Bearer-Token (Autorisierung) ist nicht erforderlich, damit diese Anfrage erfolgreich ist.

Um das Discovery-Dokument für einen HealthLake Datenspeicher anzufordern

1. Sammeln HealthLake `region` und `datastoreId` Werte. Weitere Informationen finden Sie unter Eigenschaften des Datenspeichers abrufen.
2. Konstruieren Sie mithilfe der gesammelten Werte für HealthLake `region` und eine URL für die Anfrage`datastoreId`. An `/.well-known/smart-configuration` den Endpunkt der URL anhängen. Scrollen Sie über die Schaltfläche Kopieren, um den gesamten URL-Pfad im folgenden Beispiel anzuzeigen.

   ```
   https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/.well-known/smart-configuration
   ```

3. Senden Sie die Anfrage mithilfe des AWS Signaturprotokolls GET mit Signature Version 4. Scrollen Sie über die Schaltfläche Kopieren, um das gesamte Beispiel anzuzeigen.

   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'
```

Das Discovery-Dokument für den HealthLake Datenspeicher wird als JSON-Blob zurückgegeben, in dem Sie das `authorization_endpoint` und das `token_endpoint` sowie die Spezifikationen und definierten Funktionen für den Datenspeicher finden.

```
{
    "authorization_endpoint": "https://oidc.example.com/authorize",
    "token_endpoint": "https://oidc.example.com/oauth/token",
    "capabilities": [
        "launch-ehr",
        "client-public"
    ]
}
```

`authorization_endpoint`Sowohl das als auch das `token_endpoint` sind erforderlich, um eine Client-Anwendung zu starten.

- Autorisierungsendpunkt — Die URL, die zur Autorisierung einer Client-Anwendung oder eines Benutzers benötigt wird.
- Token-Endpunkt — Der Endpunkt des Autorisierungsservers, mit dem die Client-Anwendung kommuniziert.

Eine FHIR-REST-API-Anfrage für einen HealthLake SMART-fähigen Datenspeicher stellen

Sie können FHIR-REST-API-Anfragen in einem HealthLake SMART-on-FHIR-fähigen Datenspeicher stellen. Das folgende Beispiel zeigt eine Anfrage von einer Client-Anwendung, die ein JWT im Autorisierungsheader enthält, und wie Lambda die Antwort dekodieren sollte. Nachdem die Anforderung der Client-Anwendung autorisiert und authentifiziert wurde, muss sie ein Bearer-Token vom Autorisierungsserver erhalten. Verwenden Sie das Bearer-Token im Autorisierungsheader, wenn Sie eine FHIR-REST-API-Anforderung an einen SMART-on-FHIR-fähigen Datenspeicher senden. HealthLake

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/[ID]
Authorization: Bearer auth-server-provided-bearer-token
```

Da im Autorisierungsheader ein Bearer-Token gefunden wurde und keine AWS IAM-Identität erkannt wurde, wird die Lambda-Funktion HealthLake aufgerufen, die bei der Erstellung des SMART on FHIR-fähigen HealthLake Datenspeichers angegeben wurde. Wenn das Token erfolgreich von Ihrer Lambda-Funktion dekodiert wurde, wird die folgende Beispielantwort an gesendet. 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
}
```

FHIR R4-Unterstützung für AWS HealthLake

AWS HealthLake unterstützt die FHIR R4-Spezifikation für den Austausch von Gesundheitsdaten. In den folgenden Abschnitten finden Sie unterstützende Informationen dazu, HealthLake wie Sie mithilfe der FHIR R4-Spezifikation FHIR R4-Ressourcen in Ihrem HealthLake Datenspeicher verwalten und durchsuchen können. RESTful APIs

Themen

- FHIR R4-Fähigkeitserklärung für AWS HealthLake
- FHIR-Profilvalidierungen für HealthLake
- FHIR R4 unterstützte Ressourcentypen für HealthLake
- FHIR R4-Suchparameter für HealthLake
- FHIR R4 für $operations HealthLake

FHIR R4-Fähigkeitserklärung für AWS HealthLake

Um die FHIR-bezogenen Funktionen (Verhaltensweisen) eines aktiven HealthLake Datenspeichers zu ermitteln, müssen Sie dessen Capability Statement abrufen. Die Capability-Anweisung wird als Aussage über die tatsächliche Serverfunktionalität oder als Aussage über die erforderliche oder gewünschte Serverimplementierung verwendet. Die `capabilities`FHIR-Interaktion ruft Informationen über die Funktionen des HealthLake Datenspeichers ab und gibt an, welche Teile der FHIR-Spezifikation er unterstützt. HealthLake validiert die FHIR-Ressourcentypen anhand der FHIR R4-Ressource. `StructureDefinition`

Um die Capability Statement für einen Datenspeicher abzurufen HealthLake

1. Sammeln HealthLake `region` und `datastoreId` Werte. Weitere Informationen finden Sie unter Eigenschaften des Datenspeichers abrufen.
2. Konstruieren Sie mithilfe der gesammelten Werte für HealthLake `region` und eine URL für die Anfrage`datastoreId`. Nehmen Sie auch das `metadata` FHIR-Element in die URL auf. Scrollen Sie über die Schaltfläche Kopieren, um den gesamten URL-Pfad im folgenden Beispiel anzuzeigen.

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

3. Senden Sie die Anforderung . Die `capabilities` FHIR-Interaktion verwendet eine `GET` Anfrage mit dem AWS Signature Version 4-Signaturprotokoll. Im folgenden `curl` Beispiel wird die Capability-Anweisung für den durch den `datastoreId` angegebenen HealthLake Datenspeicher abgerufen. Scrollen Sie über die Schaltfläche Kopieren, um das gesamte Beispiel anzuzeigen.

   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'
   ```

Sie erhalten einen `200` HTTP-Antwortcode und die Capability Statement für Ihren HealthLake Datenspeicher. Weitere Informationen finden Sie `CapabilityStatement`in der FHIR R4-Dokumentation.

FHIR-Profilvalidierungen für HealthLake

AWS HealthLake unterstützt die FHIR R4-Basisspezifikation. In der FHIR R4-Basisspezifikation sind FHIR-Profile enthalten. Profile werden für einen FHIR-Ressourcentyp verwendet, um mithilfe von and/or Einschränkungserweiterungen für den Basisressourcentyp eine spezifischere Definition eines Ressourcentyps zu definieren. Ein FHIR-Profil kann beispielsweise Pflichtfelder wie Erweiterungen und Wertesätze identifizieren. Eine Ressource kann mehrere Profile unterstützen. Alle HealthLake Datenspeicher unterstützen die Verwendung von FHIR-Profilen.

> **Note**
>
> Das Hinzufügen eines FHIR-Profils ist nicht erforderlich, wenn Daten zu einem HealthLake Datenspeicher hinzugefügt werden. Wenn beim Hinzufügen oder Aktualisieren einer Ressource kein FHIR-Profil angegeben wird, wird die Ressource nur anhand des FHIR R4-Basisschemas validiert.
> FHIR-Profile, denen FHIR-Ressourcen entsprechen, sind in Ressourcen enthalten, bevor sie importiert werden. HealthLake Daher werden FHIR-Profile während des Imports validiert. HealthLake

FHIR-Profile sind in einem Implementierungsleitfaden spezifiziert. Ein FHIR-Implementierungsleitfaden (IG) besteht aus einer Reihe von Anweisungen, die beschreiben, wie der FHIR-Standard für einen bestimmten Zweck verwendet werden kann. HealthLake validiert die in den folgenden Implementierungsleitfäden definierten FHIR-Profile.

FHIR-Profile werden unterstützt von AWS HealthLake

Name	Ve	Implementierungsleitfaden	Funktion	US Ea (Ol	US Os (Nc Vir gin	US We (Or	Asi Pa ifik (M	Asi Pa ifik (Sy	Ca (Ce	Eu (Irl	Europa (London)
US-Kern	3.1	http://hl7.org/fhir/us/core/STU3.1.1/	Standard	X	X	X	X	X	X	X	X
US-Kern	4.0	https://hl7.org/fhir/us/core/STU4/index.html	Unterstützt	X	X	X	X	X	X	X	X
US-Kern	5.0	https://hl7.org/fhir/us/core/STU5.0.1/index.html	Unterstützt	X	X	X	X	X	X	X	X
US-Kern	6.1	https://hl7.org/fhir/us/core/STU6.1/index.html	Unterstützt	X	X	X	X	X	X	X	X
US-Kern	7.0	https://hl7.org/fhir/us/core/STU7/	Unterstützt	X	X	X	X	X	X	X	X
Britischer Kern	2.0	https://simplifier.net/guide/uk-core-implementation-guide-stu2/Home/ProfilesandExtensions/ProfilesIndex?version=2.0.1	Unterstützt	X	X	X	X			X	X
CARIN Blauer Knopf	1.1	http://hl7.org/fhir/us/carin-bb/STU1.1/	Standard	X	X	X	X	X	X	X	X

Name	Ve	Implementierungsleitfaden	Funktion	US Ea (Ol	US Os (Nc Vir gin	US We (Or	Asi Pa ifik (Mu	Asi Pa ifik (Sy	Ca (Ce	Eu (Irl	Europa (London)
CARIN Blauer Knopf	1.0 2.0 2.1	https://hl7.org/fhir/us/carin-bb/history.html	Unterstützt	X	X	X	X	X	X	X	X
Da Vinci Payer Data Exchange	1.0	https://hl7.org/fhir/us/davinci-pdex/	Standard	X	X	X	X	X	X	X	X
Da Vinci Payer Data Exchange	2.0 2.1	https://hl7.org/fhir/us/davinci-pdex/history.html	Unterstützt	X	X	X	X	X	X	X	X
Austausch von Patientenakten von Da Vinci (HREx)	0.2	https://hl7.org/fhir/us/davinci-hrex/2020Sep/	Standard	X	X	X	X	X	X	X	X
DaVinci PDEX Plan Net	1.1	https://hl7.org/fhir/us/davinci-pdex-plan-net/STU1.1/	Standard	X	X	X	X	X	X	X	X
DaVinci PDEX Plan Net	1.0	https://hl7.org/fhir/us/davinci-pdex-plan-net/STU1/	Unterstützt	X	X	X	X	X	X	X	X

Name	Ve	Implementierungsle itfaden	Funktion	US Ea (Ol	US Os (Nc Vir gin	US We (Or	Asi Pa ifik (Mu	Asi Pa ifik (Sy	Ca (Ce	Eu (Irl	Europa (London)
DaVinci Arzneimit telformul ierung in den USA (Payer Data Exchange, PDeX)	1.1	https://hl7.org/fhir/us/davinci-drug-formulary/STU1.1/	Standard	X	X	X	X	X	X	X	X
DaVinci Arzneimit telformul ierung in den USA (Payer Data Exchange, PDeX)	1.0 2.0 2.1	https://hl7.org/fhir/us/davinci-drug-formulary/history.html	Unterstützt	X	X	X	X	X	X	X	X
Klinische r Data Exchange von Da Vinci (CDex)	2.1	https://build.fhir.org/ig/HL7/davinci-ecdx/index.html	Standard	X	X	X	X	X	X	X	X

Name	Ve	Implementierungsleitfaden	Funktion	US Ea (Ol	US Os (Nc Vir gin	US We (Or	Asi Pa ifik (Mu	Asi Pa ifik (Sy	Ca (Ce	Eu (Irl	Europa (London)
Support durch vorherige Autorisierung von Da Vinci (PAS) FHIR IG	2.1	https://hl7.org/fhir/us/davinci-pas/	Standard	X	X	X	X	X	X	X	X
NCQA HEDIS® - Implementierungsleitfaden	0.3	https://www.ncqa.org/resources/hedis-ig-resource-page/	Standard	X	X	X	X			X	X
Internationale Patientenübersicht (IPS)	2.0 Sti mn	https://hl7.org/fhir/uv/ips/2024Sep/	Standard	X	X	X	X	X	X	X	X
Qualitätsmaßnahme	5.0	https://registry.fhir.org/package/hl7.fhir.us.cqfmeasures%7C5.0.0	Standard	X	X	X	X			X	X
Berichterstattung über Genomik	3.0	https://build.fhir.org/ig/HL7/genomics-reporting/index.html	Standard	X	X	X	X			X	X

Name	Ve	Implementierungsleitfaden	Funktion	US Ea (Ol	US Os (Nc Vir gin	US We (Or	Asi Pa ifik (Mu	Asi Pa ifik (Sy	Ca (Ce	Eu (Irl	Europa (London)
Ayushman Bharat Digital Mission (ABDM) der Nationalen Gesundheitsbehörde	2.0	https://www.nrces.in/ndhm/fhir/r4/index.html	Standard	X	X	X	X			X	X
CA Core+	1.1	https://simplifier.net/ca-core	Unterstützt						X		
CA:eReC Pan-Canadian e Referral-eConsult	1.1	https://simplifier.net/CA-eReC/~introduction	Unterstützt						X		
Patientenübersicht kanadische Ausgabe - (PS-CA)	2.1	https://simplifier.net/PS-CA-R1/~introduction	Unterstützt						X		
Digitales Repositorium Health Gesundheitsmedikamente in Ontario	4.0	https://simplifier.net/ca-on-dhdr-r4/~introduction	Unterstützt						X		

Name	Ve	Implementierungsleitfaden	Funktion	US Ea (Ol	US Os (Nc Vir gin	US We (Or	Asi Pa ifik (Mu	Asi Pa ifik (Sy	Ca (Ce	Eu (Irl	Europa (London)
AU-Kern	1.0	https://hl7.org.au/fhir/core/	Unterstützt					X			
Magentus-Praxismanagement	1.2	https://fhir-versions.dev.geniesolutions.io/1.2.16/downloads.html	Unterstützt					X			

Validierung der in einer Ressource angegebenen FHIR-Profile

Damit ein FHIR-Profil validiert werden kann, fügen Sie es mithilfe der im `profile` Implementierungsleitfaden angegebenen Profil-URL dem Element der einzelnen Ressourcen hinzu.

FHIR-Profile werden validiert, wenn Sie Ihrem Datenspeicher eine neue Ressource hinzufügen. Um eine neue Ressource hinzuzufügen, können Sie den `StartFHIRImportJob` API-Vorgang verwenden, eine `POST` Anfrage zum Hinzufügen einer neuen Ressource stellen `PUT` oder eine vorhandene Ressource aktualisieren.

Example— Um zu sehen, auf welches FHIR-Profil in einer Ressource verwiesen wird

Die Profil-URL wird dem `profile` Element im `"meta" : "profile"` Schlüssel-Wert-Paar hinzugefügt. Diese Ressource wurde aus Gründen der Übersichtlichkeit gekürzt.

```
{
    "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— Wie referenziert man ein nicht standardmäßig unterstütztes FHIR-Profil

Um anhand eines unterstützten, nicht standardmäßigen Profils zu validieren, fügen Sie dem Element die versionierte Profil-URL hinzu. `meta.profile` Die versionierte URL enthält die URL des Basisprofils, gefolgt von einem senkrechten Strich (`|`) und der Versionsnummer. Diese Beispielressource wurde aus Gründen der Übersichtlichkeit gekürzt.

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

Sie können auch sowohl die versionierte URL als auch die URL des Basisprofils angeben. Beide Formate sind gültig.

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

FHIR R4 unterstützte Ressourcentypen für HealthLake

In der folgenden Tabelle sind die FHIR R4-Ressourcentypen aufgeführt, die von unterstützt werden. AWS HealthLake Weitere Informationen finden Sie unter Ressourcenindex in der FHIR R4-Dokumentation.

FHIR R4-Ressourcentypen werden unterstützt von HealthLake

Account	DetectedIssue	Rechnung	Praktiker
ActivityDefinition	Gerät	Bibliothek	PractitionerRole
AdverseEvent	DeviceDefinition	Verknüpfung	Verfahren
AllergyIntolerance	DeviceMetric	Auflisten	Herkunft
Ernennung	DeviceUseStatement	Speicherort	Fragebogen
AppointmentResponse	DeviceRequest	Measure (Maß)	QuestionnaireResponse
AuditEvent - Siehe Hinweis	DiagnosticReport	MeasureReport	RelatedPerson
Binär	DocumentManifest	Medien	RequestGroup
BodyStructure	DocumentReference	Medikamente	ResearchStudy
Paket — siehe Hinweis	EffectEvidenceSynthesis	MedicationAdministration	ResearchSubject
CapabilityStatement	Begegnung	MedicationDispense	RiskAssessment
CarePlan	Endpoint	MedicationKnowledge	RiskEvidenceSynthesis
CareTeam	EpisodeOfCare	MedicationRequest	Plan
ChargeItem	EnrollmentRequest	MedicationStatement	ServiceRequest
ChargeItemDefinition	EnrollmentResponse	MessageHeader	Schlitz
Antrag	ExplanationOfBenefit	MolecularSequence	Exemplar
ClaimResponse	FamilyMemberHistory	NutritionOrder	StructureDefinition
Kommunikation	Flag	Beobachtung	StructureMap

CommunicationRequest	Ziel	OperationOutcome - Siehe Hinweis	Substanz
Composition	Group (Gruppieren)	Organisation	SupplyDelivery
ConceptMap	GuidanceResponse	OrganizationAffiliation	SupplyRequest
Bedingung	HealthcareService	Parameter — siehe Hinweis	Aufgabe
Zustimmung	ImagingStudy	Patientin	ValueSet
Vertrag	Immunisierung	PaymentNotice	VisionPrescription
Abdeckung	ImmunizationEvaluation	PaymentReconciliation	VerificationResult - Siehe Hinweis
CoverageEligibilityRequest	ImmunizationRecommendation	Person	
CoverageEligibilityResponse	InsurancePlan	PlanDefinition	

⚠ FHIR-Spezifikationen und HealthLake

- Sie `GET` können keine `POST` Anfragen mit FHIR `OperationOutcome` und `Parameters` Ressourcentypen stellen.
- AuditEvent— Eine AuditEvent Ressource kann erstellt oder gelesen, aber nicht aktualisiert oder gelöscht werden.
- Bundle — Es gibt mehrere Möglichkeiten, Bundle-Anfragen zu HealthLake verwalten. Weitere Details finden Sie unter Bündelung der FHIR-Ressourcen.
- VerificationResult— Dieser Ressourcentyp wird nur für Datenspeicher unterstützt, die nach dem 09. Dezember 2023 erstellt wurden.

FHIR R4-Suchparameter für HealthLake

Verwenden Sie die `search`FHIR-Interaktion, um anhand einiger Filterkriterien eine Reihe von FHIR-Ressourcen in einem HealthLake Datenspeicher zu durchsuchen. Die `search` Interaktion kann entweder mit einer `GET` Oder-Anfrage durchgeführt werden. `POST` Für Suchanfragen, die personenbezogene Daten (PII) oder geschützte Gesundheitsinformationen (PHI) beinhalten, wird empfohlen, `POST` Anfragen zu verwenden, da PII und PHI als Teil des Anfragetextes hinzugefügt und bei der Übertragung verschlüsselt werden.

> **Note**
>
> Die in diesem Kapitel beschriebene `search` FHIR-Interaktion entspricht dem HL7 FHIR R4-Standard für den Austausch von Gesundheitsdaten. Da es sich um eine Repräsentation eines HL7 FHIR-Dienstes handelt, wird er nicht über und angeboten. AWS CLI AWS SDKs Weitere Informationen finden Sie `search`in der FHIR RESTful R4-API-Dokumentation.
> Sie können HealthLake Datenspeicher auch mit SQL mithilfe von Amazon Athena abfragen. Weitere Informationen finden Sie unter Integrieren.

HealthLake unterstützt die folgende Teilmenge der FHIR R4-Suchparameter. Weitere Informationen finden Sie unter FHIR R4-Suchparameter für HealthLake.

Unterstützte Suchparametertypen

Die folgende Tabelle zeigt die unterstützten Suchparametertypen in HealthLake.

Unterstützte Suchparametertypen

Suchparameter	Description
_id	Ressourcen-ID (keine vollständige URL)
_Letzte Aktualisierung	Datum der letzten Aktualisierung. Die Genauigkeit der Grenzen liegt im Ermessen des Servers.
_tag	Suche nach einem Ressourcen-Tag.
_Profil	Sucht nach allen Ressourcen, die mit einem Profil gekennzeichnet sind.

Suchparameter	Description
_Sicherheit	Sucht nach Sicherheitslabels, die auf diese Ressource angewendet wurden.
_Quelle	Sucht danach, woher die Ressource stammt.
_text	Suchen Sie nach der Erzählung der Ressource.
createdAt	Suchen Sie nach der benutzerdefinierten Erweiterung CreatedAt.

> **Note**
>
> Die folgenden Suchparameter werden nur für Datenspeicher unterstützt, die nach dem 09. Dezember 2023 erstellt wurden: _security, _source, _text, createdAt.

Die folgende Tabelle zeigt Beispiele dafür, wie Abfragezeichenfolgen auf der Grundlage bestimmter Datentypen für einen bestimmten Ressourcentyp geändert werden können. Aus Gründen der Übersichtlichkeit wurden Sonderzeichen in der Beispielspalte nicht codiert. Um eine erfolgreiche Abfrage durchzuführen, stellen Sie sicher, dass die Abfragezeichenfolge richtig codiert wurde.

Beispiele für Suchparameter

Typen von Suchparametern	Details	Beispiele

Zahl	Sucht in einer angegebenen Ressource nach einem numerischen Wert. Signifikante Zahlen werden beobachtet. Die Anzahl der signifikanten Ziffern ist spezifisch für den Suchparameterwert, ausgenommen führende Nullen. Vergleichspräfixe sind zulässig.	`[parameter]=100`
		`[parameter]=1e2`
		`[parameter]=lt100`

Typen von Suchparametern	Details	Beispiele
Datum/ DateTime	Sucht nach einem bestimmten Datum oder einer bestimmten Uhrzeit. Das erwartete Format istyyyy-mm-ddThh:mm:ss[Z|(+|-)hh:mm] , kann aber variieren. Akzeptiert die folgenden Datentypen: datedateTime,instant,PeriundTiming. Weitere Informationen zur Verwendung dieser Datentypen bei Suchen finden Sie unter Datum in der FHIR RESTful R4-API-Dokumentation. Vergleichspräfixe sind zulässig.	[parameter]=eq2013-01-14 [parameter]=gt2013-01-14T10:00 [parameter]=ne2013-01-14
Zeichenfolge	Sucht nach einer Zeichenfolge unter Berücksichtigung von Groß- und Kleinschreibung. Unterstützt beide HumanName Address Typen. Weitere Informationen finden Sie in den HumanName Datentypeinträgen und den AddressDatentypeinträgen in der FHIR R4-Dokumentation. Die erweiterte Suche wird mithilfe :text von Modifikatoren unterstützt.	[base]/Patient?given=eve [base]/Patient?given:contains=eve

Typen von Suchparametern	Details	Beispiele
Token	Sucht nach einer close-to-exact Übereinstimmung mit einer Zeichenfolge und wird häufig mit einem Paar medizinischer Codewerte verglichen. Die Berücksichtigung von Groß- und Kleinschreibung hängt vom Codesystem ab, das bei der Erstellung einer Abfrage verwendet wird. Subsumtionsbasierte Abfragen können dazu beitragen, Probleme im Zusammenhang mit der Berücksichtigung von Groß- und Kleinschreibung zu reduzieren. Aus Gründen der Übersichtlichkeit wurde das | nicht codiert.	`[parameter]=[system]|[code]` : `[system]` Bezieht sich hier auf ein Kodierungssystem und `[code]` bezieht sich auf den Codewert, der in diesem spezifischen System gefunden wurde. `[parameter]=[code]` : Hier entspricht Ihre Eingabe entweder einem Code oder einem System. `[parameter]=|[code]` : Hier entspricht Ihre Eingabe einem Code, und die Systemeigenschaft hat keinen Bezeichner.

Zusammengesetzt	Sucht mithilfe der Modifikatoren $ und der , Operation nach mehreren Parametern innerhalb eines einzigen Ressourcentyps.
	
	Vergleichspräfixe sind zulässig.
		`/Patient?language=FR,NL&language=EN`
		
		`Observation?component-code-value-quantity=http://loinc.org|8480-6$lt60`
		
		`[base]/Group?characteristic-value=gender$mixed`

Typen von Suchparametern	Details	Beispiele

Quantity (Menge)	Sucht nach einer Zahl, einem System und einem Code als Werte. Eine Zahl ist erforderlich, System und Code sind jedoch optional. Basiert auf dem Datentyp Menge. Weitere Informationen finden Sie unter Menge in der FHIR R4-Dokumentation.
	
	Verwendet die folgende angenommene Syntax `[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`

Referenz	Sucht nach Verweisen auf andere Ressourcen.	`[base]/Observation?subject=Patient/23`
		
		`test`

URI	Sucht nach einer Zeichenfolge, die eine bestimmte Ressource eindeutig identifiziert.	`[base]/ValueSet?url=http://acme.org/fhir/ValueSet/123`
Spezial	Suchanfragen basieren auf integrierten medizinischen NLP-Erweiterungen.	

Erweiterte Suchparameter werden unterstützt von HealthLake

HealthLake unterstützt die folgenden erweiterten Suchparameter.

Name	Beschreibung	Beispiel	Funktion
`_includ`	Wird verwendet, um anzufordern, dass zusätzlic he Ressourcen in einer Suchanfrage zurückgegeben werden. Es gibt Ressource n zurück, auf die von der Zielressourceninstanz verwiesen wird.	`Encounter ?_include =Encounte r:subject`	
`_revinc de`	Wird verwendet, um anzufordern, dass zusätzlic he Ressourcen in einer Suchanfrage zurückgegeben werden. Es gibt Ressource n zurück, die auf die primäre Ressourceninstanz verweisen.	`Patient?_ id=` **`patient-identifier`** `&_revincl ude=Encou nter:patient`	
`_summar`	Die Zusammenfassung kann verwendet werden, um eine Teilmenge der Ressource anzufordern.	`Patient?_ summary=text`	Die folgenden Zusammenf assungsparameter werden unterstützt:`_summary= true` ,, `_summary= false _summary= text` ,`_summary=data` .
`_elemen`	Fordert an, dass ein bestimmter Satz von Elementen als Teil einer Ressource in den Suchergeb nissen zurückgegeben wird.	`Patient?_ elements= identifie r,active, link`	

Name	Beschreibung	Beispiel	Funktion
`_total`	Gibt die Anzahl der Ressourcen zurück, die den Suchparametern entsprechen.	`Patient?_total=accurate`	Support `_total=accurate` ,`_total=none` .
`_sort`	Geben Sie die Sortierreihenfolge der zurückgegebenen Suchergebnisse mithilfe einer durch Kommas getrennten Liste an. Das `-` Präfix kann für jede Sortierregel in der kommagetrennten Liste verwendet werden, um die absteigende Reihenfolge anzugeben.	`Observation?_sort=status,-date`	Support Sortierung nach Feldern mit Typen`Number, String, Quantity, Token, URI, Reference` . Sortieren nach `Date` wird nur für Datenspeicher unterstützt, die nach dem 09. Dezember 2023 erstellt wurden. Support bis zu 5 Sortierregeln.
`_count`	Steuern Sie, wie viele Ressourcen pro Seite des Suchpakets zurückgegeben werden.	`Patient?_count=100`	Die maximale Seitengröße beträgt 100.
`chainin`	Suchen Sie nach Elementen von Ressourcen, auf die verwiesen wird. Der `.` leitet die verkettete Suche auf das Element in der referenzierten Ressource weiter.	`DiagnosticReport?subject:Patient.name=peter`	
`reverse chainin (_has)`	Sucht auf der Grundlage der Ressourcenelemente, die auf sie verweisen, nach einer Ressource.	`Patient?_has:Observation:patient:code=1234-5`	

`_include`

Durch die Verwendung `_include` in einer Suchabfrage können auch zusätzliche angegebene FHIR-Ressourcen zurückgegeben werden. Wird verwendet`_include`, um Ressourcen einzubeziehen, die weiterverknüpft sind.

Example— Wird verwendet_include, um die Patienten oder die Gruppe von Patienten zu finden, bei denen Husten diagnostiziert wurde

Sie würden nach dem `Condition` Ressourcentyp suchen, indem Sie den Diagnosecode für Husten `_include` angeben und dann angeben, dass auch die `subject` Diagnose zurückgegeben werden soll. Beim `Condition` Ressourcentyp `subject` handelt es sich entweder um den Ressourcentyp für den Patienten oder den Ressourcentyp für die Gruppe.

Aus Gründen der Übersichtlichkeit wurden Sonderzeichen im Beispiel nicht codiert. Um eine erfolgreiche Abfrage durchzuführen, stellen Sie sicher, dass die Abfragezeichenfolge richtig codiert wurde.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
Condition?code=49727002&_include=Condition:subject
```

`_include:iterate`Modifikator

Der `_include:iterate` Modifikator ermöglicht die rekursive Einbeziehung referenzierter Ressourcen auf zwei Ebenen. Zum Beispiel

```
GET /ServiceRequest?
identifier=025C0931195&_include=ServiceRequest:requester&_include:iterate=PractitionerRole:prac
```

gibt das zurück ServiceRequest, was ihm zugeordnet ist PractitionerRole (über die Referenz des Anfragenden), und schließt dann rekursiv den Practitioner ein, auf den dieser verweist. PractitionerRole Dieser Modifikator ist für alle Ressourcentypen in verfügbar. HealthLake

`_include=*`Modifikator

Der `_include=*` Modifikator ist ein Platzhalter, der automatisch alle Ressourcen einschließt, auf die in den Suchergebnissen direkt verwiesen wird. Zum Beispiel

```
GET /ServiceRequest?specimen.accession=12345&_include=*
```

gibt den Treffer ServiceRequest zusammen mit allen Ressourcen zurück, auf die er verweist (wie Patient, Arzt, Probe usw.), ohne dass jeder Referenzpfad einzeln angegeben werden muss. Dieser Modifikator ist für alle Ressourcentypen in verfügbar. HealthLake

`_revinclude`

Durch die Verwendung `_revinclude` in einer Suchabfrage können auch zusätzliche angegebene FHIR-Ressourcen zurückgegeben werden. Wird verwendet`_revinclude`, um Ressourcen einzubeziehen, die rückwärts verknüpft sind.

Example— Wird verwendet, _revinclude um verwandte Ressourcentypen für Begegnung und Beobachtung einzubeziehen, die mit einem bestimmten Patienten verknüpft sind

Um diese Suche durchzuführen, würden Sie zunächst die Person definieren, `Patient` indem Sie ihre Kennung im `_id` Suchparameter angeben. Anschließend würden Sie mithilfe der Struktur `Encounter:patient` und `Observation:patient` zusätzliche FHIR-Ressourcen angeben.

Aus Gründen der Übersichtlichkeit wurden die Sonderzeichen im Beispiel nicht codiert. Um eine erfolgreiche Abfrage durchzuführen, stellen Sie sicher, dass die Abfragezeichenfolge richtig codiert wurde.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
Patient?_id=patient-
identifier&_revinclude=Encounter:patient&_revinclude=Observation:patient
```

`_summary`

Die Verwendung `_summary` in einer Suchabfrage ermöglicht es dem Benutzer, eine Teilmenge der FHIR-Ressource anzufordern. Es kann einen der folgenden Werte enthalten:. `true, text, data, false` Alle anderen Werte werden als ungültig behandelt. Die zurückgegebenen Ressourcen werden `'SUBSETTED'` in meta.tag mit gekennzeichnet, um darauf hinzuweisen, dass die Ressourcen unvollständig sind.

- `true`: Gibt alle unterstützten Elemente zurück, die in der Basisdefinition der Ressource (n) als „Zusammenfassung“ gekennzeichnet sind.
- `text`: Gibt nur die Elemente 'text', 'id', 'meta' und nur die obligatorischen Elemente der obersten Ebene zurück.
- `data`: Gibt alle Teile außer dem Element „Text“ zurück.

- `false`: Gibt alle Teile der Ressource (n) zurück

Kann in einer einzelnen Suchanfrage `_summary=text` nicht mit `_include` oder mit `_revinclude` Suchparametern kombiniert werden.

Example— Ruft das „Text“ -Element der Patientenressourcen in einem Datenspeicher ab.

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

`_elements`

Durch die Verwendung `_elements` in einer Suchabfrage können bestimmte FHIR-Ressourcenelemente angefordert werden. Die zurückgegebenen Ressourcen werden `'SUBSETTED'` in meta.tag mit gekennzeichnet, um darauf hinzuweisen, dass die Ressourcen unvollständig sind.

Der `_elements` Parameter besteht aus einer durch Kommas getrennten Liste von Basiselementnamen, z. B. Elementen, die auf der Stammebene der Ressource definiert sind. Es dürfen nur die aufgelisteten Elemente zurückgegeben werden. Wenn `_elements` Parameterwerte ungültige Elemente enthalten, ignoriert der Server sie und gibt obligatorische Elemente und gültige Elemente zurück.

`_elements`gilt nicht für eingeschlossene Ressourcen (zurückgegebene Ressourcen, deren Suchmodus ist`include`).

Kann in einer einzelnen Suchanfrage `_elements` nicht mit `_summary` Suchparametern kombiniert werden.

Example— Rufen Sie die Elemente „Identifier“, „Active“ und „Link“ von Patientenressourcen in Ihrem HealthLake Datenspeicher ab.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?
_elements=identifier,active,link
```

`_total`

`_total`Bei Verwendung in einer Suchabfrage wird die Anzahl der Ressourcen zurückgegeben, die den angeforderten Suchparametern entsprechen. HealthLake gibt die Gesamtzahl der

übereinstimmenden Ressourcen (zurückgegebene Ressourcen, deren Suchmodus ist`match`) in `Bundle.total` der Suchantwort zurück.

`_total`unterstützt die `accurate` none Parameterwerte. `_total=estimate`wird nicht unterstützt. Alle anderen Werte werden als ungültig behandelt. `_total`gilt nicht für die eingeschlossenen Ressourcen (zurückgegebene Ressourcen, deren Suchmodus ist`include`).

Example— Ruft die Gesamtzahl der Patientenressourcen in einem Datenspeicher ab:

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

`_sort`

Durch die Verwendung `_sort` in der Suchabfrage werden die Ergebnisse in einer bestimmten Reihenfolge angeordnet. Die Ergebnisse werden auf der Grundlage der durch Kommas getrennten Liste von Sortierregeln in Prioritätsreihenfolge sortiert. Bei den Sortierregeln sollte es sich um gültige Suchparameter handeln. Alle anderen Werte werden als ungültig behandelt.

In einer einzigen Suchanfrage können Sie bis zu 5 Sortier-Suchparameter verwenden. Sie können optional ein `-` Präfix verwenden, um die absteigende Reihenfolge anzugeben. Der Server sortiert standardmäßig in aufsteigender Reihenfolge.

Die unterstützten Parametertypen für die Sortiersuche sind:`Number, String, Date, Quantity, Token, URI, Reference`. Wenn sich ein Suchparameter auf ein verschachteltes Element bezieht, wird dieser Suchparameter für die Sortierung nicht unterstützt. Beispielsweise bezieht sich die Suche nach „Name“ des Ressourcentyps Patient auf Patienten.Ein Name-Element mit HumanName Datentyp wird als verschachtelt betrachtet. Daher wird die Sortierung von Patientenressourcen nach „Name“ nicht unterstützt.

Example— Rufen Sie Patientenressourcen in einem Datenspeicher ab und sortieren Sie sie in aufsteigender Reihenfolge nach dem Geburtsdatum:

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

`_count`

Der Parameter `_count` ist als Anweisung an den Server definiert, wie viele Ressourcen auf einer einzigen Seite zurückgegeben werden sollen.

Die maximale Seitengröße ist 100. Alle Werte über 100 sind ungültig. `_count=0`wird nicht unterstützt.

Example— Suchen Sie nach der Patientenressource und legen Sie die Größe der Suchseite auf 25 fest:

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?_count=25
```

Chaining and Reverse Chaining(_has)

Verkettung und umgekehrte Verkettung in FHIR bieten eine effizientere und kompaktere Möglichkeit, miteinander verbundene Daten zu erhalten, wodurch die Notwendigkeit mehrerer separater Abfragen reduziert wird und der Datenabruf für Entwickler und Benutzer komfortabler wird.

Wenn eine Rekursionsebene mehr als 100 Ergebnisse zurückgibt, HealthLake wird 4xx zurückgegeben, um den Datenspeicher vor Überlastung und mehrfachen Paginierungen zu schützen.

Example— Verkettung — Ruft alle Daten ab DiagnosticReport , die sich auf einen Patienten beziehen, wobei der Patientenname Peter ist.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/DiagnosticReport?
subject:Patient.name=peter
```

Example— Umgekehrte Verkettung — Ruft Patientenressourcen ab, wobei auf die Patientenressource durch mindestens eine Beobachtung verwiesen wird, wobei die Beobachtung den Code 1234 hat und wobei sich die Beobachtung auf die Patientenressource im Patientensuchparameter bezieht.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient?
_has:Observation:patient:code=1234
```

Unterstützte Suchmodifikatoren

Suchmodifikatoren werden mit Feldern verwendet, die auf Zeichenfolgen basieren. Alle verwendeten Suchmodifikatoren verwenden boolesche Logik HealthLake . Sie könnten beispielsweise angeben`:contains`, dass ein größeres Zeichenkettenfeld eine kleine Zeichenfolge enthalten soll, damit es in Ihre Suchergebnisse aufgenommen wird.

Unterstützte Suchmodifikatoren

Suchmodifikator	Typ
: fehlt	Alle Parameter außer `Composite`
:exakt	Zeichenfolge
:enthält	Zeichenfolge
:nicht	Token
:Text	Token
:Bezeichner	Referenz
:unten	URI

Unterstützte Suchvergleicher

Sie können Suchkomparatoren verwenden, um die Art der Übereinstimmung in einer Suche zu steuern. Sie können Komparatoren verwenden, wenn Sie nach Zahlen-, Datums- und Mengenfeldern suchen. In der folgenden Tabelle sind Suchkomparatoren und ihre Definitionen aufgeführt, die von unterstützt werden. HealthLake

Unterstützte Suchvergleicher

Suchkomparator	Description
eq	Der Wert für den Parameter in der Ressource entspricht dem angegebenen Wert.
Eins	Der Wert für den Parameter in der Ressource entspricht nicht dem angegebenen Wert.
gt	Der Wert für den Parameter in der Ressource ist größer als der angegebene Wert.
lt	Der Wert für den Parameter in der Ressource ist kleiner als der angegebene Wert.

Suchkomparator	Description
ge	Der Wert für den Parameter in der Ressource ist größer oder gleich dem angegebenen Wert.
le	Der Wert für den Parameter in der Ressource ist kleiner oder gleich dem angegebenen Wert.
sa	Der Wert für den Parameter in der Ressource beginnt nach dem angegebenen Wert.
eb	Der Wert für den Parameter in der Ressource endet vor dem angegebenen Wert.

FHIR-Suchparameter werden nicht unterstützt von HealthLake

HealthLake unterstützt alle FHIR-Suchparameter mit Ausnahme der in der folgenden Tabelle aufgeführten. Eine vollständige Liste der FHIR-Suchparameter finden Sie in der FHIR-Suchparameter-Registrierung.

Suchparameter werden nicht unterstützt

Zusammenstellung des Pakets	Lage in der Nähe
Bundle-ID	C onsent-source-reference
Bundle-Botschaft	Vertragspatient
Typ des Pakets	Inhalt der Ressource
Zeitstempel des Pakets	Ressourcen-Abfrage

FHIR R4 für **`$operations`** HealthLake

FHIR $-Operationen (auch „Dollar-Operationen“ genannt) sind spezielle serverseitige Funktionen, die über die standardmäßigen CRUD-Operationen (`Create`,, `ReadUpdate`,`Delete`) in der FHIR-Spezifikation hinausgehen. Diese Operationen werden durch das Präfix „$“ identifiziert und ermöglichen komplexe Verarbeitungs-, Datentransformations- und Massenoperationen, deren

Ausführung mit standardmäßigen REST-API-Aufrufen schwierig oder ineffizient wäre. $ Operationen können auf Systemebene, Ressourcentypebene oder auf bestimmten Ressourceninstanzen aufgerufen werden und bieten so flexible Möglichkeiten zur Interaktion mit FHIR-Daten. AWS HealthLake unterstützt mehrere FHIR. `$operations` Weitere Informationen finden Sie auf den einzelnen Seiten unten.

Themen

- FHIR R4-Betrieb $attribution-status für HealthLake
- Ressourcentypen löschen mit $bulk-delete
- $bulk-member-match-Operation für HealthLake
- FHIR R4-Betrieb $confirm-attribution-list für HealthLake
- FHIR R4-Betrieb $davinci-data-export für HealthLake
- Generierung klinischer Dokumente mit $document
- Dauerhaftes Entfernen von Ressourcen mit $erase
- Patientendaten abrufen mit Patient/$everything
- ValueSet Codes abrufen mit $expand
- HealthLake Daten mit FHIR exportieren $export
- $inquireFHIR-Operation für HealthLake
- Konzeptdetails abrufen mit $lookup
- $member-addOperation für HealthLake
- $member-matchOperation für HealthLake
- $member-removeOperation für HealthLake
- Ressourcen für die Patientenabteilung entfernen mit $purge
- $questionnaire-packageFHIR-Operation für HealthLake
- $submitFHIR-Operation für HealthLake
- Validierung von FHIR-Ressourcen mit $validate

FHIR R4-Betrieb **`$attribution-status`** für HealthLake

Ruft den Zuordnungsstatus für ein bestimmtes Mitglied ab und gibt ein Paket zurück, das alle Zuordnungsressourcen enthält, die sich auf den Patienten beziehen. Dieser Vorgang ist Teil der Implementierung der FHIR Member Attribution (ATR) List IG 2.1.0.

Endpoint

```
POST [base]/Group/[id]/$attribution-status
```

Anforderungsparameter

Die Operation akzeptiert die folgenden optionalen Parameter:

Parameter	Typ	Description
memberId	Kennung	Das MemberId Mitglied, für das der Zuordnung sstatus angefordert wird
Referenz des Patienten	Referenz	Verweis auf die Patientenressource im System des Herstellers

> **Note**
>
> Zu Validierungszwecken `patientReference` kann entweder `memberId` oder beide bereitgestellt werden.

Beispielanforderung

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "memberId",
      "valueIdentifier": {
        "system": "http://example.org",
        "value": "MBR123456789"
      }
    },
    {
      "name": "patientReference",
      "valueReference": {
        "reference": "Patient/patient-123",
        "display": "John Doe"
      }
```

```
    }
  ]
}
```

Antwort

Gibt ein Paket mit Attributionsressourcen zurück, die sich auf den Patienten beziehen:

Ressource	Kardinalität	Speicherort
Patientin	1..1	Gruppe.Mitglied.Entität
Abdeckung	0.. 1	group.member.extension: Deckungsreferenz
Organization/Practitioner/PractitionerRole	0.. 1	group.member.extension: AttributedProvider
Beliebige Ressource	0.. 1	group.member.extension: Assoziierte Daten

Beispielantwort

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

Fehlerbehandlung

Der Vorgang behandelt die folgenden Fehlerbedingungen:

Fehler	HTTP-Status	Description
Ungültige Operation sanforderung	400	Fehlerhafte Anforderungsparameter oder Struktur
Die Gruppenre ssource wurde nicht gefunden	404	Die angegebene Gruppen-ID ist nicht vorhanden

Fehler	HTTP-Status	Description
Die Patienten ressource wurde nicht gefunden	404	Die angegebene Patientenreferenz ist nicht vorhanden

Autorisierung und Sicherheit

Anforderungen an SMART Scope

Kunden müssen über die entsprechenden Rechte verfügen, um Gruppenressourcen und zugehörige Zuordnungsressourcen lesen zu können

Die standardmäßigen FHIR-Autorisierungsmechanismen gelten für alle Operationen

Ressourcentypen löschen mit **`$bulk-delete`**

AWS HealthLake unterstützt den `$bulk-delete` Vorgang und ermöglicht das Löschen aller Ressourcen eines bestimmten Typs in einem Datenspeicher. Dieser Vorgang ist besonders nützlich, wenn Sie:

- Führen Sie saisonale Audits und Aufräumarbeiten durch
- Verwalten Sie den Datenlebenszyklus in großem Umfang
- Entfernen Sie bestimmte Ressourcentypen
- Halten Sie die Richtlinien zur Datenspeicherung ein

Usage

Der `$bulk-delete` Vorgang kann mit POST-Methoden aufgerufen werden:

```
POST [base]/[ResourceType]/$bulk-delete?isHardDelete=false&deleteAuditEvent=true
```

Parameters

Parameter	Typ	Erforderlich	Standard	Description
`isHardDelete`	Boolean	Nein	false	Wenn der Wert wahr ist, werden Ressourcen dauerhaft aus dem Speicher entfernt
`deleteAuditEvent`	boolesch	Nein	true	Wenn der Wert wahr ist, werden die zugehörigen Prüfereignisse gelöscht
`_since`	Zeichenfolge	Nein	Erstellungszeit des Datenspeichers	Wenn diese Option eingegeben wird, wird der Startgrenzwert für die Suche nach Ressourcen auf der Grundlage ihrer letzten Änderung ausgewählt. Kann nicht mit Start oder Ende verwendet werden
`start`	Zeichenfolge	Nein	Erstellungszeit des Datenspeichers	Wenn diese Option eingegeben wird, wird der Annahmeschluss für die Suche nach Ressourcen auf der Grundlage ihrer letzten Änderung ausgewählt. Kann mit Ende verwendet werden
`end`	Zeichenfolge	Nein	Zeit für die Einreichung des Job	Wenn diese Option eingegeben wird, wird der letzte Annahmeschluss für die Suche nach Ressourcen auf der Grundlage ihrer letzten Änderung ausgewählt

Beispiele

Beispielanforderung

```
POST [base]/Observation/$bulk-delete?isHardDelete=false
```

Beispielantwort

```
{
      "jobId": "jobId",
      "jobStatus": "SUBMITTED"
    }
```

Aufgabenstatus

Um den Status eines Massenlöschauftrags zu überprüfen:

```
GET [base]/$bulk-delete/[jobId]
```

Der Vorgang gibt Informationen zum Auftragsstatus zurück:

```
{
      "datastoreId": "datastoreId",
      "jobId": "jobId",
      "status": "COMPLETED",
      "submittedTime": "2025-10-09T15:09:51.336Z"
    }
```

Behavior

Die `$bulk-delete` Operation:

1. Wird asynchron verarbeitet, um große Ressourcenmengen zu handhaben
2. Sorgt für die Datenintegrität bei ACID-Transaktionen
3. Ermöglicht die Nachverfolgung des Auftragsstatus anhand der Anzahl gelöschter Ressourcen
4. Unterstützt sowohl den weichen als auch den harten Löschmodus
5. Beinhaltet eine umfassende Auditprotokollierung der Löschaktivitäten
6. Ermöglicht das selektive Löschen von historischen Versionen und Prüfereignissen

Protokollierung von Prüfungen

Der `$bulk-delete` Vorgang wird als Start FHIRBulk DeleteJob und Beschreibung protokolliert und FHIRBulk DeleteJob enthält detaillierte Informationen zum Vorgang.

Einschränkungen

- Wenn diese Option auf „true“ gesetzt `isHardDelete` ist, werden hartgelöschte Ressourcen nicht in Suchergebnissen oder `_history` Abfragen angezeigt.
- Auf Ressourcen, die durch diesen Vorgang gelöscht werden, kann während der Verarbeitung möglicherweise vorübergehend nicht zugegriffen werden
- Die Speicherbelegung wird nur für historische Versionen angepasst — deleteVersionHistory =false passt den Datenspeicherspeicher nicht an

`$bulk-member-match-Operation` für HealthLake

AWS HealthLake unterstützt den `$bulk-member-match` Vorgang zur asynchronen Verarbeitung von Matchanfragen für mehrere Mitglieder. Dieser Vorgang ermöglicht es Organisationen im Gesundheitswesen, die eindeutigen Identifikatoren von Hunderten von Mitgliedern in verschiedenen Gesundheitssystemen effizient abzugleichen, indem demografische Informationen und Informationen zur Deckung in einer einzigen Sammelanfrage verwendet werden. Diese Funktion ist für den groß angelegten Datenaustausch zwischen Kostenträgern, für die Umstellung von Mitgliedern und die Einhaltung der CMS-Anforderungen unerlässlich und entspricht der FHIR-Spezifikation.

> **Note**
>
> Die `$bulk-member-match` Operation basiert auf einer zugrunde liegenden FHIR-Spezifikation, die sich derzeit in der Erprobung befindet und sich ändern kann. Im Zuge der Weiterentwicklung der Spezifikation werden das Verhalten und die Schnittstelle dieser API entsprechend aktualisiert. Entwicklern wird empfohlen, die HealthLake AWS-Versionshinweise und die entsprechenden Aktualisierungen der FHIR-Spezifikation zu verfolgen, um über alle Änderungen, die sich auf ihre Integrationen auswirken könnten, auf dem Laufenden zu bleiben.

Dieser Vorgang ist besonders nützlich, wenn Sie:

- Führen Sie während der offenen Anmeldezeiträume einen maßstabsgetreuen Abgleich der Mitglieder durch
- Erleichtern Sie den Massenwechsel zwischen den Kostenträgern
- Support umfangreicher CMS-Compliance-Datenaustauschanforderungen
- Passen Sie Mitgliedskohorten aus allen Gesundheitsnetzwerken effizient an

- Reduzieren Sie API-Aufrufe und verbessern Sie die betriebliche Effizienz bei Matching-Szenarien mit hohem Volumen

Usage

Der `$bulk-member-match` Vorgang ist ein asynchroner Vorgang, der mit der POST-Methode für Gruppenressourcen aufgerufen wird:

```
POST [base]/Group/$bulk-member-match
```

Nachdem Sie eine Sammelabgleichsanfrage eingereicht haben, können Sie den Auftragsstatus wie folgt abfragen:

```
GET [base]/$bulk-member-match-status/{jobId}
```

Unterstützte Parameter

HealthLake unterstützt die folgenden `$bulk-member-match` FHIR-Parameter:

Parameter	Typ	Erforderlich	Description
`MemberPatient`	Patient	Ja	Patientenressource mit demografischen Informationen für das abzugleichende Mitglied.
`CoverageToMatch`	Abdeckung	Ja	Informationsquelle, die für den Abgleich mit vorhandenen Datensätzen verwendet wird.
`CoverageToLink`	Abdeckung	Nein	Coverage-Ressource, die während des Abgleichs verknüpft werden soll.
`Consent`	Zustimmung	Ja	Die Zustimmungsressource für Autorisierungszwecke wird ebenfalls gespeichert. Dies unterscheidet sich von einzelnen Vorgängen`$member-match` , bei denen keine Zustimmung erforderlich ist.

POST-Anfrage zur Einreichung eines Match-Jobs für mehrere Mitglieder

Das folgende Beispiel zeigt eine POST-Anfrage zum Einreichen eines Match-Jobs für mehrere Mitglieder. Jedes Mitglied ist in einen `MemberBundle` Parameter eingeschlossen, der die erforderlichen `MemberPatientCoverageToMatch`, und `Consent` Ressourcen sowie einen optionalen Wert enthält`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"
              }
            ]
          }
        }
      ]
    }
  ]
}
```

Antwort auf den Auftrag mit Ausgabe abgeschlossen

Wenn der Job abgeschlossen ist, enthält die Antwort Job-Metadaten und eine FHIR-Parameter-Ressource, die drei Gruppenressourcen enthält, die die Vergleichsergebnisse kategorisieren.

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

Wie werden Mitglieder HealthLake in Ausgabegruppen eingeteilt

Jedes in einer `$bulk-member-match` Anfrage eingereichte Mitglied wird über eine sequentielle Pipeline bewertet. Das Ergebnis jedes Schritts bestimmt, welcher Output-Gruppe das Mitglied zugeordnet wird.

1. Strukturvalidierung — Entspricht das den MemberBundle erforderlichen Profilen? Bei einem Fehler: Fehler (in keiner Gruppe).
2. Patientenabgleich — Kann ich Patienten HealthLake finden, die den eingereichten demografischen Daten entsprechen? Bei einem Fehler: NonMatchedMembers.
3. Bestätigung des Versicherungsschutzes — Kann auf genau einen Patienten mit gültigem Versicherungsschutz HealthLake eingegrenzt werden CoverageToMatch? Bei einem Fehler: NonMatchedMembers.
4. Bewertung der Einwilligung - Ist die eingereichte Einwilligung derzeit ehrenhaft? (Status = aktiv, Zeitraum deckt das aktuelle Datum ab, der Ausführende kann bestätigt werden). Bei einem Fehler: ConsentConstrainedMembers.
5. Erfolg — Alle Prüfungen wurden bestanden. Die Zustimmung wird im Datenspeicher gespeichert. Mitglied hinzugefügt. MatchedMembers

Grundprinzip: Ein Mitglied kann nur an einem Zielort erscheinen. Der erste fehlgeschlagene Schritt bestimmt die Platzierung. Mitglieder, die die Stufen 2 oder 3 nicht bestehen, werden niemals in diese Gruppe ConsentConstrainedMembers aufgenommen. Diese Gruppe ist ausschließlich für Mitglieder bestimmt, die erfolgreich gematcht haben, deren Zustimmung jedoch nicht eingehalten werden kann.

Einzelheiten zur Bewertung der Einwilligung (Schritt 4):

- Test 1 — Zustimmungsstatus: `Consent.status` Entspricht „aktiv“? Wenn nicht → ConsentConstrainedMembers.
- Test 2 — Bereitstellungszeitraum: `provision.period` Deckt das aktuelle Datum ab? Wenn das aktuelle Datum vor `period.start` oder nach dem liegt `period.end` → ConsentConstrainedMembers.
- Test 3 — Überprüfung durch den Ausführenden: Kann die `Consent.performer` Referenz validiert werden? Wenn die referenzierte Ressource nicht im Datenspeicher gefunden wird oder nicht mit dem entsprechenden Patienten verknüpft ist →. ConsentConstrainedMembers

Alle Prüfungen müssen bestanden werden, damit das Mitglied aufgenommen MatchedMembers und die Zustimmung gespeichert wird.

Verhalten bei der Anpassung an die Deckung

Beim Mitgliederabgleich `CoverageToMatch` wird nur anhand des Datenspeichers des antwortenden Zahlers validiert. `CoverageToLink`gehört dem new/requesting Zahler und wird nicht anhand des Datenspeichers des alten Zahlers validiert. Die Aufnahme `CoverageToLink` in die Anfrage hat keinen Einfluss auf die passenden Ergebnisse.

Jede Kombination aus Patient und Versicherungsschutz in der Anfrage wird unabhängig bearbeitet. Derselbe Patient kann mehrfach mit unterschiedlichen Versicherungsplänen eingereicht werden, und jeder Eintrag erhält sein eigenes Ergebnis, das auf seinem spezifischen Versicherungsschutz basiert.

Umgang mit Referenzen durch den Consent Performer

Der neue Zahler kann eine vorläufige oder lokale Patientenreferenz einsenden `Consent.performer` (z. B. dieselbe Referenznummer, die in verwendet wurde`Consent.patient`). HealthLake löst diese Referenzen automatisch auf:

- `Consent.performer`Enthält dieselbe lokale Referenz wie`Consent.patient`, HealthLake ersetzt sie nach erfolgreichem Abgleich durch die tatsächlich zugeordnete Patientenreferenz.

- HealthLake unterstützt Ausführende Referenzen der Typen Patient RelatedPerson, Praktiker und Organisation (sowohl direkte Verweise als auch Verweise auf logische Identifikatoren). PractitionerRole
- Schlägt die Überprüfung des Ausführenden fehl (die Ressource wurde nicht gefunden oder ist dem entsprechenden Patienten nicht zugeordnet), wird das Mitglied hinzugefügt, ConsentConstrainedMembers anstatt einen Fehler zurückzugeben.

Ressourcen der Output-Gruppe

Der abgeschlossene Job gibt eine Parameter-Ressource zurück, die drei Gruppenressourcen enthält:

MatchedMembers Group (Gruppieren)

Enthält Patientenreferenzen für alle Mitglieder, die erfolgreich zugeordnet wurden und deren Einwilligung zum Zeitpunkt der Anfrage aktiv und gültig ist. Die Zustimmungsressource wird für jedes zugeordnete Mitglied erstellt und im Datenspeicher gespeichert. Diese Gruppe wird im Datenspeicher instanziiert und kann direkt mit verwendet werden. `$davinci-data-export`

NonMatchedMembers Group (Gruppieren)

Enthält Verweise auf Mitglieder, bei denen keine eindeutige Übereinstimmung gefunden wurde. Ein Mitglied wird hier platziert, wenn kein Patient im Datenspeicher den angegebenen demografischen Daten entspricht, kein gültiger Versicherungsschutz für einen passenden Patientenkandidaten besteht oder wenn mehrere Patienten den demografischen Daten entsprechen und mehrere Patienten einen gültigen Versicherungsschutz haben (mehrdeutig).

ConsentConstrainedMembers Group (Gruppieren)

Enthält Patientenreferenzen für Mitglieder, die erfolgreich zugeordnet wurden (demografische Daten und Versicherungsschutz bestätigt), deren Zustimmung jedoch zum Zeitpunkt der Anfrage nicht berücksichtigt werden kann. Die Zustimmungsressource wird nicht für Mitglieder gespeichert, deren Einwilligung eingeschränkt ist. Die identifizierte Mitgliedsidentität (MemberIdentifier und MemberId) ist weiterhin enthalten, sodass der anfragende Zahler weiß, für wen die Einschränkung gilt.

Das `Group.quantity` Feld enthält die Gesamtzahl der Mitglieder in jeder ihrer jeweiligen Gruppen.

Referenzen der Gruppenmitglieder:

- `Group.member.entity.reference`— Für MatchedMembers und ConsentConstrainedMembers, enthält die Patienten-ID des entsprechenden Mitglieds im System des antwortenden Kostenträgers. Für NonMatchedMembers, verweist auf den enthaltenen Eingabepatienten.
- `Group.member.entity.extension (base-ext-match-parameters)`— Enthält die Patienten-ID aus der ursprünglichen Eingabeanfrage (die vom anfragenden Zahler übermittelte ID, abgeleitet von`Patient.id`,`Coverage.beneficiary.reference`, oder`Consent.patient.reference`).

Consent-Patient Verknüpfung

> **Important**
>
> In der gespeicherten Einwilligungsressource wird die Patientenreferenz genau so gespeichert, wie sie vom anfragenden Zahler übermittelt wurde. HealthLake aktualisiert das Patientenfeld der Einwilligung nicht automatisch so, dass es auf den entsprechenden Patienten im Empfangsdatenspeicher verweist.

Um eine gespeicherte Einwilligung mit dem entsprechenden Patienten zu verknüpfen, verwenden Sie die Jobausgabe: Jedes Mitglied der MatchedMembers Gruppe `member.entity.reference` zeigt mit einem Zeichen auf den entsprechenden Patienten und mit einem `member.entity.extension` (`base-ext-match-parameters`) auf den enthaltenen Eingabepatienten. Cross-reference diese mit dem Patientenfeld der Einwilligung, um das Mapping in Ihrer Anwendungsebene zu erstellen.

Was wird gespeichert und was ist vorübergehend

In der folgenden Tabelle wird dokumentiert, was während der `$bulk-member-match` Verarbeitung im Datenspeicher HealthLake verbleibt und was nur in der Job-Antwort enthalten ist:

Ressource	Gespeichert?	Über REST abfragbar?	Hinweise
MemberPatient (Eingabe)	Nein	Nein	Wird nur für den Abgleich verwendet; nicht dauerhaft
CoverageToMatch (Eingabe)	Nein	Nein	Wird nur zur Bestätigung der Deckung verwendet

Ressource	Gespeichert?	Über REST abfragbar?	Hinweise
CoverageToLink (Eingabe)	Nein	Nein	Nicht anhand des Datenspeichers validiert; gehört dem neuen Zahler
Zustimmung (übereinstimmende Mitglieder)	Ja	Ja — GET [base] / Consent/ {id}	Wird so gespeichert, wie es vom anfragenden Zahler eingegangen ist
Zustimmung (eingeschränkte Mitglieder)	Nein	Nein	Nicht gespeichert. Die Identität des Mitglieds ist immer noch in der Antwort enthalten.
MatchedMembers Gruppe (Ausgabe)	Ja	Ja — GET [base] / Group/ {id}	Instanziiert; nutzbar mit $davinci-data-export
NonMatchedMembers Gruppe	Nein	Nein	Nur Antwort auf den Job
ConsentConstrained Members Gruppe	Nein	Nein	Nur Antwort auf den Job

Integration mit $davinci-data-export

Die von zurückgegebene MatchedMembers Gruppenressource `$bulk-member-match` kann direkt mit dem `$davinci-data-export` Vorgang zum Abrufen von Massendaten von Mitgliedern verwendet werden:

```
POST [base]/Group/{matched-group-id}/$davinci-data-export
GET [base]/Group/{matched-group-id}
```

Diese Integration ermöglicht effiziente Workflows, bei denen Sie zuerst übereinstimmende Mitglieder in großen Mengen identifizieren und dann ihre vollständigen Gesundheitsdaten mithilfe der resultierenden Gruppenressource exportieren.

Verwenden Sie $member-remove vor dem Export

Wenn Sie nach dem Abgleich bestimmte Mitglieder vom Export ausschließen müssen (z. B. wenn ein Mitglied die Zustimmung zwischen dem Abgleich und dem Export widerruft), verwenden `$member-remove` Sie dies für die Gruppe. MatchedMembers

> **Important**
>
> Wenn Sie ein Mitglied über entfernen, wird das Mitglied in der Gruppe als inaktiv `$member-remove` markiert, inaktive Mitglieder werden jedoch `$davinci-data-export` erst ausgeschlossen, nachdem die Gruppe auf den Status „Endgültig“ aktualisiert wurde. Wenn Sie eine Gruppe aufrufen`$davinci-data-export`, die immer noch den Standardstatus hat, werden entfernte Mitglieder möglicherweise immer noch in den Exportergebnissen angezeigt.

Arbeitsablauf:

1. `POST [base]/Group/{id}/$member-remove`— Mitglieder als inaktiv markieren
2. `PUT [base]/Group/{id}`— den Gruppenstatus auf „endgültig“ aktualisieren
3. `POST [base]/Group/{id}/$davinci-data-export`— Der Export schließt jetzt entfernte Mitglieder aus

Leistungsmerkmale

Der `$bulk-member-match` Vorgang ist für die Verarbeitung großer Datenmengen konzipiert und läuft asynchron:

- Parallelität: Maximal 5 gleichzeitige Operationen pro Datenspeicher.
- Skalierbarkeit: Verarbeitet bis zu 500 Mitglieder pro Anfrage (Nutzlastlimit von 5 MB).
- Parallele Operationen: Kompatibel mit gleichzeitigen Import-, Export- oder Massenlöschvorgängen.

Autorisierung

Die API verwendet SMART auf dem FHIR-Autorisierungsprotokoll mit den folgenden erforderlichen Bereichen:

- `system/Patient.read`— Erforderlich, um Patientenressourcen zu suchen und abzugleichen.
- `system/Coverage.read`— Erforderlich, um die Versicherungsinformationen zu überprüfen.

- `system/Group.write`— Erforderlich, um Ressourcen für Ergebnisgruppen zu erstellen.
- `system/Organization.read`— Befriedigend, erforderlich, wenn sich der Versicherungsschutz auf Organisationen bezieht.
- `system/Practitioner.read`— Befriedigend, erforderlich, wenn sich der Versicherungsschutz auf Praktiker bezieht.
- `system/PractitionerRole.read`— Befristet, erforderlich, wenn sich der Versicherungsschutz auf die Rollen von Praktikern bezieht.
- `system/Consent.write`— Befriedigend, erforderlich, wenn Ressourcen für die Einwilligung bereitgestellt werden.

Der Vorgang unterstützt auch die AWS IAM Signature Version 4 (Sigv4) -Autorisierung für programmatischen Zugriff.

Regeln für die Validierung

In Schritt 1 werden jeweils MemberBundle die folgenden Validierungsregeln angewendet. Mitglieder, die die Überprüfung nicht bestehen, werden als Fehler gemeldet und erscheinen in keiner Ausgabegruppe.

MemberPatient

Feld	Wie HealthLake benutzt es	Die Überprüfung ist fehlgesch lagen, wenn...
`name.family`	Demografische Suche	Fehlen
`name.given`	Demografische Suche	Fehlt (mindestens eine erforderl ich)
`birthDate`	Demografische Suche	Fehlen
`gender`	Demografische Suche; falls nicht, wurde die Erweiterung Birth Sex verwendet	Weder Geschlecht noch Geburtsgeschlecht vorhanden (hrex-pat-1)
`identifier`	Wird in die Suche aufgenommen, sofern vorhanden; verbessert das Selbstvertrauen	Verursacht niemals Fehler (optional)

CoverageToMatch / CoverageToLink

Feld	Wie HealthLake benutzt es	Die Überprüfung ist fehlgesch lagen, wenn...
`status`	Bestätigt, dass die Berichter stattung umsetzbar ist	Fehlen
`beneficiary`	Verbindet den Versicherungsschut z mit einem Patientenkandidaten	Fehlen
`payor`	Begriffsklärung, wenn mehrere Kandidaten existieren	Fehlt oder es gibt mehr als einen Zahler
`relationship`	Bestätigt die Beziehung zwischen Abonnent und Begünstigtem	Fehlen
`identifier (MB) or subscriberId`	Primärer Begriffsklärungsschlüssel	Keiner von beiden anwesend

Zustimmung

Feld	Wie HealthLake benutzt es	Die Überprüfung ist fehlgesch lagen, wenn...
`scope`	Bestätigt, dass der Geltungsb ereich der Einwilligung die Privatsphäre des Patienten ist	Fehlender oder kein Datenschu tzcode für Patienten
`category`	Bestätigt die Klassifizierung der Offenlegung	Fehlen
`patient`	Identifiziert den Einwilligungssubje kt	Fehlen
`performer`	Identifiziert, wer zustimmt	Fehlen

Feld	Wie HealthLake benutzt es	Die Überprüfung ist fehlgesch lagen, wenn...
`sourceReference`	Dokumentiert die Quelle der Zustimmung	Fehlen
`policy.uri`	Legt den Umfang der gemeinsam en Nutzung von Daten fest	Fehlt oder URI endet nicht mit #regular oder #sensitive
`provision.type`	Muss laut HREx Consent-Profil „Permit“ sein	„Zulassen“ fehlt oder nicht (einschli eßlich „verweigern“)
`provision .period`	In Schritt 4 im Hinblick auf eine Prüfung mit eingeschränktem Einverständnis bewertet	Fehlt oder nicht start/end
`status`	In Schritt 4 bewertet (NICHT Schritt 1)	Führt niemals zu einem Fehler in Schritt 1 — HealthLake akzeptiert jeden gültigen Status und bewertet in Schritt 4

> **Note**
>
> Das HREx Consent-Profil definiert den Status mit dem festen Wert „aktiv“. HealthLake Diese Einschränkung wird bewusst gelockert, sodass eine Zustimmung, die nicht aktiv ist, eine aussagekräftige Einstufung (ConsentConstrainedMembers) erhält und nicht pauschal abgewiesen wird.

Übereinstimmendes Verhalten

- Patientensuche (Schritt 2) — HealthLake sucht mit `name.family` + `name.given` (exakt, ohne Berücksichtigung von Groß- und Kleinschreibung), `birthDate` (exakt), `gender` (exakt; Geburtsgeschlecht wird verwendet, wenn das Geschlecht fehlt) und `identifier` (falls vorhanden, optional).
- Disambiguierung des Erfassungsbereichs (Schritt 3) — Wenn mehrere Patientenkandidaten gefunden werden, `CoverageToMatch` wird diese Methode verwendet, um auf einen Patienten

einzugrenzen. Eine Abdeckung ist „gültig“, wenn im Datenspeicher eine aktive Coverage-Ressource vorhanden ist, die mit `subscriberId` oder `identifier` (Typ MB) UND übereinstimmt. `payor`

- Bewertung der Zustimmung (Schritt 4) — Wird erst nach einem erfolgreichen eindeutigen Match durchgeführt. Einzelheiten zur Bewertung der Einwilligung finden Sie weiter oben.

Fehlerbehandlung

Der Vorgang behandelt die folgenden Fehlerbedingungen:

- 400 Bad Request: Ungültiges Anforderungsformat, fehlende erforderliche Parameter oder die Nutzlast überschreitet die Größenbeschränkungen (500 Mitglieder oder 5 MB).
- 422 Unverarbeitbare Entität: Verarbeitungsfehler bei der Auftragsausführung.
- Fehler einzelner Mitglieder: Wenn ein bestimmtes Mitglied die Verarbeitung nicht durchführt, wird der Vorgang mit den verbleibenden Mitgliedern fortgesetzt und enthält Fehlerdetails in der NonMatchedMembers Gruppe mit den entsprechenden Ursachencodes. Beispiel: Wenn `MemberBundle` bei einem Patienten der `birthDate` Parameter fehlt, wird der folgende Fehler zurückgegeben:

```
"errors": [
  {
    "memberIndex": 1,
    "jsonBlob": {
      "resourceType": "OperationOutcome",
      "issue": [
        {
          "severity": "error",
          "code": "invalid",
          "diagnostics": "MemberPatient.birthDate is required"
        }
      ],
      "statusCode": 400
    }
  }
]
```

Fehlerdetails sind über den Endpunkt für die Statusabfrage verfügbar und beinhalten:

- `numberOfMembersWithCustomerError`: Anzahl der Mitglieder mit Überprüfungs- oder Eingabefehlern.
- `numberOfMembersWithServerError`: Anzahl der Mitglieder mit serverseitigen Verarbeitungsfehlern.

Zugehörige -Vorgänge

- the section called “$member-match”— Vorgang zum Abgleich einzelner Mitglieder.
- the section called “$davinci-data-export”— Massendatenexport mithilfe von Gruppenressourcen.
- the section called “$ Operationen”— Vollständige Liste der unterstützten Operationen.

FHIR R4-Betrieb **`$confirm-attribution-list`** für HealthLake

Zeigt einem Hersteller an, dass der Verbraucher keine weiteren Änderungen an der Attributionsliste vornehmen muss. Bei diesem Vorgang wird die Zuordnungsliste abgeschlossen, indem inaktive Mitglieder aus der Liste entfernt, der Status auf „endgültig“ geändert und der Vorgang bestätigt wird. Dieser Vorgang ist Teil der Implementierung der FHIR Member Attribution (ATR) List IG 2.1.0.

Endpoint

```
POST [base]/Group/[id]/$confirm-attribution-list
```

Anforderung

Es sind keine Parameter erforderlich.

```
POST [base]/Group/[id]/$confirm-attribution-list
```

Antwort

Gibt HTTP 201 mit einer Bestätigungsnachricht zurück.

Beispielantwort

```
HTTP Status Code: 201

{
```

```
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "message",
      "valueString": "Accepted."
    }
  ]
}
```

Gruppenstatus nach Bestätigung

Nach erfolgreicher Bestätigung wird der Status der Zuordnungsliste der Gruppenressource auf „final“ gesetzt:

```
{
  "resourceType": "Group",
  "id": "fullexample",
  "extension": [
    {
      "url": "http://hl7.org/fhir/us/davinci-atr/StructureDefinition/ext-
attributionListStatus",
      "valueCode": "final"
    }
  ]
  // ... other Group properties
}
```

Fehlerbehandlung

Der Vorgang behandelt die folgenden Fehlerbedingungen:

Fehler	HTTP-Status	Description
Ungültige Operation sanforderung	400	Fehlerhafte Anforderungsparameter oder Struktur
Die Gruppenre ssource wurde nicht gefunden	404	Die angegebene Gruppen-ID ist nicht vorhanden

Autorisierung und Sicherheit

Anforderungen an SMART Scope

Kunden müssen über die entsprechenden Rechte verfügen, um Gruppenressourcen und zugehörige Zuordnungsressourcen lesen zu können

Denn `$confirm-attribution-list` Clients müssen auch über Schreibrechte verfügen, um Gruppenressourcen zu ändern

Die standardmäßigen FHIR-Autorisierungsmechanismen gelten für alle Operationen

FHIR R4-Betrieb **`$davinci-data-export`** für HealthLake

Bei dem `$davinci-data-export` Vorgang handelt es sich um einen asynchronen FHIR-Vorgang, aus dem Sie Gesundheitsdaten exportieren können. AWS HealthLake Dieser Vorgang unterstützt mehrere Exporttypen, darunter Member Attribution (ATR), PDex Provider Access und Member Access. Payer-to-Payer APIs Es handelt sich um eine spezielle Version des `$export` Standard-FHIR-Vorgangs, die darauf ausgelegt ist, die Anforderungen der DaVinci Implementierungsleitfäden zu erfüllen.

Wichtigste Funktionen

- Asynchrone Verarbeitung: Folgt dem standardmäßigen asynchronen FHIR-Anforderungsmuster
- Export auf Gruppenebene: Exportiert Daten für Mitglieder innerhalb einer bestimmten Gruppenressource
- Mehrere Exporttypen: Unterstützt ATR (Member Attribution), PDex Provider Access und Payer-to-Payer Member Access APIs
- Umfassender Profilsupport: Beinhaltet US Core, CARIN Blue Button und PDex Profile
- Flexibles Filtern: Unterstützt das Filtern nach Patienten, Ressourcentypen und Zeiträumen
- NDJSON-Ausgabe: Stellt Daten im durch Zeilenumbruch getrennten JSON-Format bereit

Endpunkt des Vorgangs

```
GET [base]/Group/[id]/$davinci-data-export
POST [base]/Group/[id]/$davinci-data-export
```

Anforderungsparameter

Parameter	Kardinalität	Description
`patient`	0.. *	Bestimmte Mitglieder, deren Daten exportiert werden sollen. Wenn dieser Wert weggelassen wird, werden alle Mitglieder der Gruppe exportiert.
`_type`	0.. 1	Kommagetrennte Liste der zu exportierenden FHIR-Ressourcentypen. Wenn nicht angegeben, sind alle unterstützten Ressourcentypen für den angegebenen Exporttyp enthalten. Bei ATR-Exporten werden standardmäßig die 8 Attribution-Ressourcentypen verwendet. Bei PDex Exporten umfasst dies alle Ressourcentypen für die Zuordnung sowie die Ressourcentypen für klinische Daten und Schadensfälle aus US Core, CARIN Blue Button und Profilen. PDex
`_since`	0.. 1	Schließt nur Ressourcen ein, die nach diesem Datum und dieser Uhrzeit aktualisiert wurden.
`_until`	0.. 1	Schließt nur Ressourcen ein, die vor diesem Datum und dieser Uhrzeit aktualisiert wurden.
`exportType`	0.. 1	Art des auszuführenden Exports. Zulässige Werte: `hl7.fhir.us.davinci-atr`, `hl7.fhir.us.davinci-pdex` , `hl7.fhir.us.davinci-pdex.p2p` , `hl7.fhir.us.davinci-pdex.member` . Standard: `hl7.fhir.us.davinci-atr` .
`_includeEOB2xWoFinancial`	0.. 1	Gibt an, ob CARIN BB ExplanationOfBenefit 2.x-Ressourcen eingeschlossen werden sollen, deren Finanzdaten entfernt wurden. Standard: `false`.

Parameter	Kardinalität	Description
`_security`	0.. *	Filtert exportierte Ressourcen nach `meta.security` Codierungswerten. Verwenden Sie das `system|code` Format (Pipe-Zeichen muss URL-kodiert sein als). `%7C` Wenn mehrere Werte angegeben werden, müssen die Ressourcen mit allen übereinst immen (UND der Semantik). Verwenden Sie `system|` (Trailing Pipe, kein Code), um beliebigen Code aus einem bestimmten System abzugleichen.
`_tag`	0.. *	Filtert exportierte Ressourcen nach `meta.tag` Codierungswerten. Verwendet dasselbe `system|code` Format und dieselbe UND-Semantik wie`_security` . Wenn `_security` sowohl als auch angegeben `_tag` sind, müssen die Ressourcen beiden Filtern entsprechen.

Unterstützte Ressourcentypen

Die unterstützten Ressourcentypen hängen vom angegebenen Exporttyp ab. Für ATR-Exporte werden die folgenden Ressourcentypen unterstützt:

- `Group`
- `Patient`
- `Coverage`
- `RelatedPerson`
- `Practitioner`
- `PractitionerRole`
- `Organization`
- `Location`

Für PDex Exporte (Anbieterzugriff und Mitgliederzugriff) werden zusätzlich zu den vorherigen Typen alle Ressourcentypen für klinische Zwecke und Reklamationen unterstützt. Payer-to-Payer Eine vollständige Liste der unterstützten Ressourcentypen finden Sie im US Core Implementation Guide (STU 6.1), im CARIN Blue Button Implementation Guide und im Da Vinci Prior Authorization Support Implementation Guide.

Exporttypen

Der `$davinci-data-export` Vorgang unterstützt die folgenden Exporttypen. Sie geben den Exporttyp mithilfe des `exportType` Parameters an.

Exporttyp	Zweck	Umfang der Daten	Zeitliches Limit
`hl7.fhir.us.davinci-atr`	Liste der Zuordnung von Mitgliedern	Ressourcen im Zusammenhang mit der Zuordnung	Keine
`hl7.fhir.us.davinci-pdex`	API für den Anbieterzugriff	Klinische Daten und Reklamationsdaten für zugewiesene Patienten	5 Jahre
`hl7.fhir.us.davinci-pdex.p2p`	Payer-to-Payer Austausch	Historische Mitgliederdaten für Versicherungsübergänge	5 Jahre
`hl7.fhir.us.davinci-pdex.member`	API für den Mitgliederzugriff	Eigene Gesundheitsdaten des Mitglieds	5 Jahre

> **Note**
>
> Bei PDex Exporten gilt das Zeitlimit von 5 Jahren nicht für ATR-Ressourcentypen (`GroupPatient`,`Coverage`,`RelatedPerson`,`Practitioner`,, `PractitionerRoleOrganization`,`Location`). Diese Ressourcen sind unabhängig vom Alter immer enthalten.

ATR (hl7.fhir.us.davinci-atr)

Mit dem ATR-Exporttyp können Sie Daten aus der Mitgliedsattributionsliste exportieren. Verwenden Sie diesen Exporttyp, um Ressourcen im Zusammenhang mit der Zuordnung für Mitglieder innerhalb einer Gruppe abzurufen. Weitere Informationen finden Sie unter Da Vinci ATR-Exportvorgang.

Unterstützte Ressourcentypen

`Group`, `Patient`, `Coverage`, `RelatedPerson`, `Practitioner`, `PractitionerRole`, `Organization`, `Location`

Temporale Filterung

Es wird keine zeitliche Filterung angewendet. Alle passenden Ressourcen werden unabhängig vom Datum exportiert.

PDex Exporttypen

Alle PDex Exporttypen verwenden dieselben unterstützten Profile und dieselbe Filterlogik. Weitere Informationen finden Sie in der Da Vinci PDex Provider Access API. Die folgenden Profile werden unterstützt:

- US Core 3.1.1, 6.1.0 und 7.0.0
- PDex Vorherige Autorisierung (wird für den Mitgliederzugriff nicht unterstützt)
- CARIN BB 2.x Basisprofile: stationär stationär, ambulant stationär, beruflich, oral, pharmazeutisch NonClinician

Bei PDex Exporten werden die Ressourcen für klinische Daten und Reklamationen automatisch für jeden Patienten in der Gruppe ermittelt. Sie müssen in der Gruppenressource nicht explizit auf diese Ressourcen verweisen. Der Vorgang sucht nach allen Ressourcen der Patientenabteilung (wie`Observation`,,`Condition`, und`ExplanationOfBenefit`) `Coverage` `RelatedPerson``MedicationRequest`, die zu den zugewiesenen Patienten gehören. Nur die Typen `Patient``Group`,, und non-patient-compartment ATR (`Practitioner`,, `PractitionerRole``Organization`,`Location`) erfordern ausdrückliche Verweise in der Gruppe.

Anbieterzugriff (`hl7.fhir.us.davinci-pdex`)

Ermöglicht netzinternen Anbietern das Abrufen von Patientendaten für zugewiesene Patienten.

Payer-to-Payer (`hl7.fhir.us.davinci-pdex.p2p`)

Ermöglicht den Datenaustausch zwischen Kostenträgern, wenn ein Patient die Versicherung wechselt.

Zugang für Mitglieder () `hl7.fhir.us.davinci-pdex.member`

Ermöglicht Mitgliedern den Zugriff auf ihre eigenen Gesundheitsdaten. Dieser Exporttyp kann Finanzdaten in den Ressourcen für Schadensfälle enthalten.

Profilunterstützung und Inklusionslogik

Bei PDex Exporten verwendet der `$davinci-data-export` Vorgang Profildeklarationen im `meta.profile` Element, um zu bestimmen, welche Ressourcen in den Export aufgenommen werden sollen.

ExplanationOfBenefit Umgang mit Ressourcen

`ExplanationOfBenefit`(EOB) Ressourcen werden auf der Grundlage ihrer `meta.profile` Deklarationen von PDex Exporten eingeschlossen oder ausgeschlossen:

- ExplanationOfBenefit Ressourcen mit einem CARIN BB 1.x-Profil sind vom Export ausgeschlossen.
- ExplanationOfBenefit Ressourcen ohne `meta.profile` Set sind vom Export ausgeschlossen.
- ExplanationOfBenefit Ressourcen mit einem CARIN BB 2.x Basisprofil sind immer enthalten.
- ExplanationOfBenefit Ressourcen mit einem CARIN BB 2.x-Profil, das Finanzdaten enthält, sind standardmäßig ausgeschlossen. Wenn diese `_includeEOB2xWoFinancial=true` Option gesetzt ist, werden sie in den gelöschten Finanzdaten enthalten und die Ressource wird in das entsprechende Basisprofil umgewandelt.
- ExplanationOfBenefit Ressourcen mit einem Profil „ PDex Vorherige Autorisierung“ sind immer enthalten.

Transformation von Finanzdaten

Wenn Sie diese Einstellung vornehmen`_includeEOB2xWoFinancial=true`, transformiert der Vorgang die ExplanationOfBenefit Ressourcen von CARIN BB 2.x in die entsprechenden Basisprofile, indem Finanzdaten entfernt werden. Beispielsweise wird eine Ressource in eine `C4BB ExplanationOfBenefit Oral` Ressource umgewandelt`C4BB ExplanationOfBenefit Oral Basis`, wodurch Finanzdaten gemäß der FHIR-Spezifikation aus dem Datensatz entfernt werden.

Die folgenden Finanzdatenelemente werden bei der Transformation entfernt:

- Alle Elemente werden in Scheiben geschnitten `total`
- Alle `adjudication` Elemente mit Slice `amounttype`
- Alle `item.adjudication` Elemente mit Mengenangaben

Der Vorgang aktualisiert auch die Profilmetadaten während der Transformation:

- `meta.profile`wird auf die kanonische URL des Basisprofils aktualisiert
- Die Version wurde auf die CARIN BB 2.x Basisversion aktualisiert
- Bestehende Ressourcen im Datenspeicher werden nicht geändert
- Exportierte Ressourcen werden nicht dauerhaft im Datenspeicher gespeichert

Regeln für die Profilerkennung

Bei diesem Vorgang werden die folgenden Regeln verwendet, um Profile zu erkennen und zu validieren:

- Die Versionserkennung basiert auf dem `meta.profile` kanonischen URLs
- Eine Ressource ist enthalten, wenn EINES ihrer deklarierten Profile den Exportkriterien entspricht
- Die Profilvalidierung erfolgt während der Exportverarbeitung

Fünfjährige zeitliche Filterung für Exporte PDex

HealthLake Wendet für alle PDex Exporttypen einen Zeitfilter für fünf Jahre an, der darauf basiert, wann die Ressource zuletzt aktualisiert wurde. Der Zeitfilter gilt für alle Ressourcen mit Ausnahme der folgenden zentralen Attributionsressourcentypen, die unabhängig vom Alter immer exportiert werden:

- `Patient`
- `Coverage`
- `Organization`
- `Practitioner`
- `PractitionerRole`
- `RelatedPerson`

- `Location`
- `Group`

Diese administrativen und demografischen Ressourcen sind ausgenommen, da sie den wesentlichen Kontext für die exportierten Daten bieten. ATR-Exporte unterliegen keiner zeitlichen Filterung.

Beispielanforderungen

Die folgenden Beispiele zeigen, wie Exportaufträge für verschiedene Exporttypen gestartet werden.

ATR-Export

```
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"
    }
  }
}
```

Provider Access-Export mit Entfernung von ExplanationOfBenefit Finanzdaten

```
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 exportieren

```
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
```

Export mit Mitgliederzugriff für einen bestimmten Patienten

```
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
```

Beispielantwort

```
{
  "datastoreId": "eaee622d8406b41eb86c0f4741201ff9",
  "jobStatus": "SUBMITTED",
  "jobId": "48d7b91dae4a64d00d54b70862f33f61"
}
```

Beziehungen zu Ressourcen

Der Vorgang exportiert Ressourcen auf der Grundlage ihrer Beziehungen innerhalb der Mitgliedsattributionsliste:

```
Group (Attribution List)
### Patient (Members)
### Coverage # RelatedPerson (Subscribers)
### Practitioner (Attributed Providers)
### PractitionerRole # Location
### Organization (Attributed Providers)
```

> **Note**
>
> Das obige Diagramm der Ressourcenbeziehungen gilt für ATR-Exporte. Bei PDex Exporten werden Ressourcen für klinische Daten und Reklamationen automatisch bei der Suche nach Patienten gefunden, sodass keine ausdrücklichen Verweise in der Gruppenressource erforderlich sind.

Quellen der Ressourcen

Ressource	Standort der Quelle	Description
`Patient`	`Group.member.entity`	Die Patienten, die Mitglieder der Zuordnungsliste sind
`Coverage`	`Group.member.extension:coverageReference`	Versicherungsschutz, der zur Patientenmitgliedschaft führte
`Organization`	`Group.member.extension:attributedProvider`	Organizations, denen Patienten zugeschrieben werden
`Practitioner`	`Group.member.extension:attributedProvider`	Einzelne Ärzte, denen die Patienten zugeordnet sind
`PractitionerRole`	`Group.member.extension:attributedProvider`	Rollen von Praktikern, denen Patienten zugeschrieben werden
`RelatedPerson`	`Coverage.subscriber`	Abonnenten der Berichterstattung
`Location`	`PractitionerRole.location`	Standorte, die mit den Rollen von Praktikern in Verbindung stehen
`Group`	Eingabeendpunkt	Die Zuordnungsliste selbst

Verwaltung von Aufträgen

Überprüfen Sie den Jobstatus

```
GET [base]/export/[job-id]
```

Auftrag abbrechen

```
DELETE [base]/export/[job-id]
```

Auftragslebenszyklus

- `SUBMITTED`- Der Job wurde empfangen und in die Warteschlange gestellt

- `IN_PROGRESS`- Der Job wird aktiv bearbeitet
- `COMPLETED`- Job erfolgreich abgeschlossen, Dateien zum Herunterladen verfügbar
- `FAILED`- Job ist auf einen Fehler gestoßen

Ausgabeformat

- Dateiformat: NDJSON (Newline Delimited JSON)
- Dateiorganisation: Separate Dateien für jeden Ressourcentyp
- Dateierweiterung: .ndjson
- Speicherort: Spezifizierter S3-Bucket und Pfad

Fehlerbehandlung

Der Vorgang gibt HTTP 400 Bad Request mit einer OperationOutcome der folgenden Bedingungen zurück:

Fehler bei der Autorisierung

Die in angegebene IAM-Rolle `DataAccessRoleArn` verfügt nicht über ausreichende Berechtigungen, um den Exportvorgang durchzuführen. Die vollständige Liste der erforderlichen S3- und KMS-Berechtigungen finden Sie unter Berechtigungen für Exportaufträge einrichten.

Fehler bei der Parametervalidierung

- Der `patient` Parameter ist nicht formatiert als `Patient/id,Patient/id,...`
- Eine oder mehrere Patientenreferenzen sind ungültig oder gehören nicht zur angegebenen Gruppe
- Der `exportType` Parameterwert ist kein unterstützter Exporttyp
- Der `_type` Parameter enthält Ressourcentypen, die für den angegebenen Exporttyp nicht unterstützt werden
- Dem `_type` Parameter fehlen die erforderlichen Ressourcentypen (`GroupPatient`,,`Coverage`) für den `hl7.fhir.us.davinci-atr` Exporttyp
- Der `_includeEOB2xWoFinancial` Parameterwert ist kein gültiger boolescher Wert

Fehler bei der Ressourcenvalidierung

- Die angegebene Gruppenressource ist nicht im Datenspeicher vorhanden

- Die angegebene Gruppenressource hat keine Mitglieder
- Ein oder mehrere Gruppenmitglieder verweisen auf Patientenressourcen, die nicht im Datenspeicher vorhanden sind

Sicherheit und Autorisierung

- Es gelten die standardmäßigen FHIR-Autorisierungsmechanismen
- Die Datenzugriffsrolle muss über die erforderlichen IAM-Berechtigungen für S3- und KMS-Operationen verfügen. Die vollständige Liste der erforderlichen Berechtigungen finden Sie unter Berechtigungen für Exportaufträge einrichten.

Bewährte Methoden

- Auswahl des Ressourcentyps: Fordern Sie nur die Ressourcentypen an, die Sie benötigen, um die Exportgröße und die Verarbeitungszeit zu minimieren
- Zeitbasierte Filterung: Verwenden Sie den `_since` Parameter für inkrementelle Exporte
- Patientenfilterung: Verwenden Sie den `patient` Parameter, wenn Sie nur Daten für bestimmte Mitglieder benötigen
- Auftragsüberwachung: Überprüfen Sie regelmäßig den Auftragsstatus für große Exporte
- Fehlerbehandlung: Implementieren Sie die richtige Wiederholungslogik für fehlgeschlagene Jobs
- Erkennung von Zeitfiltern: Bei PDex Exporten sollten Sie bei der Auswahl von Ressourcentypen den Zeitfilter für 5 Jahre in Betracht ziehen
- Entfernung von Finanzdaten: Verwenden Sie diese `_includeEOB2xWoFinancial=true` Option, wenn Sie Forderungsdaten ohne Finanzinformationen benötigen
- Profilverwaltung: Stellen Sie sicher, dass Ressourcen über die entsprechenden Profildeklarationen verfügen, überprüfen Sie sie vor der Aufnahme anhand von Zielprofilen und verwenden Sie die Profilversionsverwaltung, um das Exportverhalten zu steuern

Einschränkungen

- In dem Parameter können maximal 500 Patienten angegeben werden `patient`
- Der Export ist nur auf Operationen auf Gruppenebene beschränkt
- Unterstützt nur die vordefinierten Ressourcentypen für jeden Exporttyp
- Die Ausgabe erfolgt immer im NDJSON-Format

- PDex Die Exporte sind auf klinische Daten und Reklamationsdaten für einen Zeitraum von 5 Jahren begrenzt
- Die Transformation von Finanzdaten gilt nur für CARIN BB 2.x-Profile ExplanationOfBenefit

Weitere Ressourcen

- Zuordnungsliste für Da Vinci-Mitglieder IG
- Da Vinci Payer Data Exchange AG
- CARIN Consumer Directed Payer Data Exchange IG
- Leitfaden zur Core-Implementierung in den USA
- FHIR-Spezifikation für den Zugriff auf Massendaten

Generierung klinischer Dokumente mit **`$document`**

AWS HealthLake unterstützt jetzt den `$document` Vorgang für Zusammenstellungsressourcen, sodass Sie ein vollständiges klinisches Dokument erstellen können, indem Sie die Zusammenstellung mit all ihren referenzierten Ressourcen in einem einzigen zusammenhängenden Paket bündeln. Dieser Vorgang ist unverzichtbar für Anwendungen im Gesundheitswesen, die:

- Erstellen Sie standardisierte klinische Dokumente
- Tauschen Sie komplette Patientenakten aus
- Bewahren Sie umfassende klinische Unterlagen auf
- Generieren Sie Berichte, die den gesamten relevanten Kontext enthalten

Usage

Der `$document` Vorgang kann für Composition-Ressourcen mit den Methoden GET und POST aufgerufen werden:

Unterstützte Vorgänge

```
GET/POST [base]/Composition/[id]/$document
```

Unterstützte Parameter

HealthLake unterstützt den folgenden FHIR-Parameter`$document`:

Parameter	Typ	Erforderlich	Standard	Description
`persist`	Boolean	Nein	false	Boolescher Wert, der angibt, ob der Server das generierte Dokumente npaket speichern soll

Beispiele

GET-Anfrage

```
GET [base]/Composition/180f219f-97a8-486d-99d9-ed631fe4fc57/$document?persist=true
```

POST-Anfrage mit Parametern

```
POST [base]/Composition/180f219f-97a8-486d-99d9-ed631fe4fc57/$document
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "persist",
      "valueBoolean": true
    }
  ]
}
```

Beispielantwort

Die Operation gibt eine Bundle-Ressource vom Typ „Dokument“ zurück, die die Komposition und alle referenzierten Ressourcen enthält:

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

Behavior

Die `$document` Operation:

1. Nimmt die angegebene Kompositionsressource als Grundlage für das Dokument
2. Identifiziert alle Ressourcen, auf die in der Komposition direkt verwiesen wird, und ruft sie ab
3. Verpackt die Komposition und alle Ressourcen, auf die verwiesen wird, in ein Paket vom Typ „Dokument“
4. Speichert das generierte Dokumentpaket im Datenspeicher, wenn der Parameter persist auf true gesetzt ist
5. Identifiziert Ressourcen, auf die in der Zusammensetzung indirekt verwiesen wird, und ruft sie ab, um eine umfassende Dokumentgenerierung zu ermöglichen

Der `$document` Vorgang unterstützt derzeit das Abrufen von Ressourcenreferenzen im folgenden Format:

1. `GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Resource/id`
2. Ressource/ID

Nicht unterstützte Ressourcenverweise innerhalb der Kompositionsressource werden aus dem generierten Dokument herausgefiltert.

Fehlerbehandlung

Der Vorgang behandelt die folgenden Fehlerbedingungen:

- 400 Bad Request: Ungültiger `$document` Vorgang (nicht konforme Anfrage) oder wenn das resultierende Dokument die FHIR-Validierung aufgrund herausgefilterter Verweise nicht besteht, wenn persist auf true gesetzt ist
- 404 Nicht gefunden: Die Kompositionsressource wurde nicht gefunden

Weitere Informationen zur `$document` Operationsspezifikation finden Sie in der FHIR `$document` R4-Kompositionsdokumentation.

Dauerhaftes Entfernen von Ressourcen mit **`$erase`**

AWS HealthLake unterstützt den `$erase` Vorgang und ermöglicht das dauerhafte Löschen einer bestimmten Ressource und ihrer historischen Versionen. Dieser Vorgang ist besonders nützlich, wenn Sie:

- Einzelne Ressourcen dauerhaft entfernen
- Löschen Sie bestimmte Versionsverläufe
- Verwalten Sie die Lebenszyklen einzelner Ressourcen
- Erfüllen Sie die spezifischen Anforderungen zur Datenlöschung

Usage

Der `$erase` Vorgang kann auf zwei Ebenen aufgerufen werden:

Ebene der Ressourceninstanz

```
POST [base]/[ResourceType]/[ID]/$erase?deleteAuditEvent=true
```

Versionsspezifische Ebene

```
POST [base]/[ResourceType]/[ID]/_history/[VersionID]/$erase
```

Parameters

Parameter	Typ	Erforderlich	Standard	Description
`deleteAuditEvent`	Boolean	Nein	false	Wenn der Wert wahr ist, werden die zugehörigen Prüfereignisse gelöscht

Beispiele

Beispielanforderung

```
POST [base]/Patient/example-patient/$erase
```

Beispielantwort

```
{
      "jobId": "5df47e2f51ff3c731847678cb8cad48e",
      "jobStatus": "SUBMITTED"
    }
```

Aufgabenstatus

Um den Status eines Löschauftrags zu überprüfen:

```
GET [base]/$erase/[jobId]
```

Der Vorgang gibt Informationen zum Auftragsstatus zurück:

```
{
      "datastoreId": "36622996b1fcecb7e12ee2ee085308d3",
      "jobId": "5df47e2f51ff3c731847678cb8cad48e",
```

```
    "status": "COMPLETED",
    "submittedTime": "2025-10-30T16:39:24.160Z"
}
```

Behavior

Die `$erase` Operation:

1. Wird asynchron verarbeitet, um die Datenintegrität sicherzustellen
2. Verwaltet ACID-Transaktionen
3. Ermöglicht die Nachverfolgung des Jobstatus
4. Löscht die angegebene Ressource und ihre Versionen dauerhaft
5. Beinhaltet eine umfassende Auditprotokollierung der Löschaktivitäten
6. Unterstützt das selektive Löschen von Prüfereignissen

Protokollierung von Prüfungen

Der `$erase` Vorgang protokolliert wie DeleteResource mit Benutzer-ID, Zeitstempel und Ressourcendetails.

Einschränkungen

- `$erased`Die Ressource wird nicht in Suchergebnissen oder `_history` Abfragen angezeigt.
- Auf Ressourcen, die gelöscht werden, kann während der Verarbeitung möglicherweise vorübergehend nicht zugegriffen werden
- Die Speicherbelegung wird sofort angepasst, wenn Ressourcen dauerhaft entfernt werden

Patientendaten abrufen mit **`Patient/$everything`**

Die `Patient/$everything` Operation wird verwendet, um eine `Patient` FHIR-Ressource zusammen mit allen anderen Ressourcen, die damit zusammenhängen, abzufragen. `Patient` Die Operation kann verwendet werden, um einem Patienten Zugriff auf seine gesamte Patientenakte zu gewähren oder es einem Anbieter zu ermöglichen, einen Massendatendownload für einen Patienten durchzuführen. HealthLakeunterstützt `Patient/$everything` einen bestimmten Patienten`id`.

`Patient/$everything`ist eine FHIR-REST-API-Operation, die wie in den folgenden Beispielen gezeigt aufgerufen werden kann.

GET request

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything
```

> **Note**
>
> Die antwortenden Ressourcen sind nach Ressourcentyp und Ressource sortiert. `id`
> Die Antwort ist immer mit gefülltBundle`.total`.

`Patient/$everything`-Parameter

HealthLake unterstützt die folgenden Abfrageparameter

Parameter	Details
start	Ruft alle `Patient` Daten ab einem bestimmten Startdatum ab.
end	Ruft alle `Patient` Daten vor einem bestimmten Enddatum ab.
since	Alle `Patient` Daten werden nach einem bestimmten Datum aktualisiert.
_typ	Ruft `Patient` Daten für bestimmte Ressourcentypen ab.
_count	`Patient`Daten abrufen und Seitengröße angeben.

Example- Ruft alle Patientendaten ab einem bestimmten Startdatum ab

`Patient/$everything`kann den `start` Filter verwenden, um nur Daten nach einem bestimmten Datum abzufragen.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?start=2024-03-15T00:00:00.000Z
```

Example- Ruft alle Patient Daten vor einem bestimmten Enddatum ab

Patient $everything kann den end Filter verwenden, um nur Daten vor einem bestimmten Datum abzufragen.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?end=2024-03-15T00:00:00.000Z
```

Example- Alle Patient Daten werden nach einem bestimmten Datum aktualisiert

`Patient/$everything`kann den `since` Filter verwenden, um nur Daten abzufragen, die nach einem bestimmten Datum aktualisiert wurden.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?since=2024-03-15T00:00:00.000Z
```

Example- Ruft Patient Daten für bestimmte Ressourcentypen ab

Patient $everything kann den `_type` Filter verwenden, um bestimmte Ressourcentypen anzugeben, die in die Antwort aufgenommen werden sollen. In einer durch Kommas getrennten Liste können mehrere Ressourcentypen angegeben werden.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?_type=Observation,Condition
```

Example- Patient Daten abrufen und Seitengröße angeben

Patient $everything kann das verwenden`_count`, um die Seitengröße einzustellen.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/Patient/id/
$everything?_count=15
```

`Patient/$everything`start und Attribute **`end`**

HealthLake unterstützt die folgenden Ressourcenattribute für die Parameter `Patient/$everything` `start` und `end` query.

Ressource	"Resource"-Element
Account	Account.ServicePeriod.Start
AdverseEvent	AdverseEvent. Datum

Ressource	"Resource"-Element
AllergyIntolerance	AllergyIntolerance. Aufgezeichnetes Datum
Termin	Termin. Start
AppointmentResponse	AppointmentResponse. starten
AuditEvent	AuditEvent.Zeitraum.Beginn
Basic	Basic. Erstellt
BodyStructure	NO_DATE
CarePlan	CarePlan.Zeitraum.Beginn
CareTeam	CareTeam.Zeitraum.Beginn
ChargeItem	ChargeItem. occurrenceDateTime, ChargeItem .occurrencePeriod.Start, .occurrenceTiming.Event ChargeItem
Antrag	Claim.BillablePeriod.Start
ClaimResponse	ClaimResponse. erstellt
ClinicalImpression	ClinicalImpression. Datum
Kommunikation	Mitteilung. Gesendet
CommunicationRequest	CommunicationRequest. occurrenceDateTime,. CommunicationRequest Vorkommniszeit.Start

Ressource	"Resource"-Element
Composition	Zusammensetzung.Datum
Bedingung	Zustand. Datum der Aufzeichnung
Zustimmung	Zustimmung.DateTime
Abdeckung	Geltungsbereich. Zeitraum. Start
CoverageEligibilityRequest	CoverageEligibilityRequest. erstellt
CoverageEligibilityResponse	CoverageEligibilityResponse. erstellt
DetectedIssue	DetectedIssue. identifiziert
DeviceRequest	DeviceRequest. Verfasst am
DeviceUseStatement	DeviceUseStatement. Aufgenommen am
DiagnosticReport	DiagnosticReport. effektiv
DocumentManifest	DocumentManifest. geschaffen
DocumentReference	DocumentReference.kontext.period.start
Begegnung	Begegnung. Zeit. Start

Ressource	"Resource"-Element
EnrollmentRequest	EnrollmentRequest. erstellt
EpisodeOfCare	EpisodeOfCare.Zeitraum.Start
ExplanationOfBenefit	ExplanationOfBenefit. Abrechnungszeitraum.Start
FamilyMemberHistory	NO_DATE
Flag	Flag.Period.Start
Ziel	Ziel.StatusDatum
Group (Gruppieren)	KEIN_DATUM
ImagingStudy	ImagingStudy. gestartet
Immunisierung	Impfung. Aufgenommen
ImmunizationEvaluation	ImmunizationEvaluation. Datum
ImmunizationRecommendation	ImmunizationRecommendation.datum
Rechnung	Rechnung.Datum
Auflisten	Liste.Datum

Ressource	"Resource"-Element
MeasureReport	MeasureReport.Zeitraum.Beginn
Medien	Medien/ausgegeben
MedicationAdministration	MedicationAdministration. wirksam
MedicationDispense	MedicationDispense. Wenn vorbereitet
MedicationRequest	MedicationRequest. Verfasst am
MedicationStatement	MedicationStatement. Datum des Eintrags
MolecularSequence	NO_DATE
NutritionOrder	NutritionOrder. dateTime
Beobachtung	Beobachtung. Wirksam
Patient	KEIN DATUM
Person	KEIN_DATUM
Verfahren	Verfahren. Durchgeführt
Herkunft	Herkunft. Vorkommender Zeitraum. Beginn, Herkunft. occurredDateTime
QuestionnaireResponse	QuestionnaireResponse. verfasst

Ressource	"Resource"-Element
RelatedPerson	KEIN_DATUM
RequestGroup	RequestGroup. Autor am
ResearchSubject	ResearchSubject. Zeitraum
RiskAssessment	RiskAssessment. occurrenceDateTime,. RiskAssessment Vorkommniszeit.Start
Plan	Schedule.PlanningHorizon
ServiceRequest	ServiceRequest. Verfasst am
Exemplar	Exemplar. Uhrzeit des Empfangs
SupplyDelivery	SupplyDelivery. occurrenceDateTime, SupplyDelivery .occurrencePeriod. Start, .occurrenceTiming.Event SupplyDelivery
SupplyRequest	SupplyRequest. Verfasst am
VisionPrescription	VisionPrescription. Datum, an dem geschrieben wurde

ValueSet Codes abrufen mit **$expand**

AWS HealthLake unterstützt jetzt den `$expand` Vorgang für diejenigen ValueSets , die von Ihnen als Kunde aufgenommen wurden, sodass Sie die vollständige Liste der in diesen ValueSet Ressourcen enthaltenen Codes abrufen können. Dieser Vorgang ist besonders nützlich, wenn Sie:

- Rufen Sie zu Validierungszwecken alle möglichen Codes ab
- Verfügbare Optionen in Benutzeroberflächen anzeigen
- Führen Sie umfassende Codelookups innerhalb eines bestimmten Terminologiekontextes durch

Usage

Der `$expand` Vorgang kann mit den Methoden GET und POST für ValueSet Ressourcen aufgerufen werden:

Unterstützte Vorgänge

```
GET/POST [base]/ValueSet/[id]/$expand
GET [base]/ValueSet/$expand?url=http://example.com
POST [base]/ValueSet/$expand
```

Unterstützte Parameter

HealthLake unterstützt eine Teilmenge von FHIR R4-Parametern: `$expand`

Parameter	Typ	Erforderlich	Description
`url`	uri	Nein	Kanonische URL des zu erweiterenden ValueSet
`id`	id	Nein	ValueSet zu erweiterende Ressourcen-ID (für GET- oder POST-Operationen)
`filter`	Zeichenfolge	Nein	Filtert das Ergebnis der Codeerweiterung
`count`	Ganzzahl	Nein	Anzahl der zurückzugebenden Codes
`offset`	Ganzzahl	Nein	Anzahl der übereinstimmenden Codes, die vor der Rückkehr übersprungen werden müssen. Gilt nach dem Filtern und nur für die passenden Codes, nicht für den vollständigen, ungefilterten Inhalt des Originals ValueSet

Beispiele

GET-Anfrage nach ID

```
GET [base]/ValueSet/example-valueset/$expand
```

GET-Anfrage per URL mit Filter

```
GET [base]/ValueSet/$expand?url=http://example.com/ValueSet/my-
valueset&filter=male&count=5
```

POST-Anfrage mit Parametern (nach ID)

```
POST [base]/ValueSet/example-valueset/$expand
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "count",
      "valueInteger": 10
    },
    {
      "name": "filter",
      "valueString": "admin"
    }
  ]
}
```

POST-Anfrage mit Parametern (per 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
    }
  ]
```

```
}
```

Beispielantwort

Die Operation gibt eine ValueSet Ressource mit einem `expansion` Element zurück, das die erweiterten Codes enthält:

```
{
  "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"
      }
    ]
```

```
  }
}
```

Die Antwort enthält:

- expansion.total: Gesamtzahl der Codes im erweiterten Bereich ValueSet
- expansion.contains: Array erweiterter Codes mit ihren System-, Code- und Anzeigewerten
- expansion.parameter: In der Erweiterungsanforderung verwendete Parameter

Weitere Informationen zur `$expand` Operationsspezifikation finden Sie in der FHIR R4-Dokumentation. ValueSet `$expand`

HealthLake Daten mit FHIR exportieren **`$export`**

Mithilfe der Operation FHIR $export können Sie HealthLake Daten in großen Mengen aus Ihrem Datenspeicher exportieren. HealthLake unterstützt die `$export` Verwendung `POST` von FHIR und Anfragen. `GET` Um eine Exportanfrage mit stellen zu können`POST`, benötigen Sie einen IAM-Benutzer, eine Gruppe oder eine IAM-Rolle mit den erforderlichen Berechtigungen, die Sie im `$export` Rahmen der Anfrage angeben und die gewünschten Parameter in den Anfragetext aufnehmen können.

> **Note**
>
> Alle mit FHIR gestellten HealthLake Exportanfragen `$export` werden im `ndjson` Format zurückgegeben und in einen Amazon S3 S3-Bucket exportiert, in dem jedes Amazon S3 S3-Objekt nur einen einzigen FHIR-Ressourcentyp enthält.
> Sie können Exportanfragen im Rahmen der AWS Konto-Servicekontingente in die Warteschlange stellen. Weitere Informationen finden Sie unter Servicekontingente.

HealthLake unterstützt die folgenden drei Arten von Massenexport-Endpunktanforderungen.

HealthLake **`$export`**Massentypen

Exporttyp	Description	Syntax
System (System)	Exportieren Sie alle Daten vom HealthLake FHIR-Server.	`POST https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/$export`

Alle Patienten	Exportieren Sie alle Daten zu allen Patienten, einschließlich der Ressourcentypen, die dem Ressourcentyp Patient zugeordnet sind.	`POST https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/Patient/$export`
		
		`GET https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/Patient/$export`

Gruppe von Patienten	Exportieren Sie alle Daten, die sich auf eine Patientengruppe beziehen, für die eine Gruppen-ID angegeben wurde.	`POST https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/Group/ id/$export`
		
		`GET https://healthlake. region.amazonaws.com/datastore/ datastoreId /r4/Group/ id/$export`

Bevor Sie beginnen

Erfüllen Sie die folgenden Anforderungen, um eine Exportanfrage mithilfe der FHIR REST-API für HealthLake zu stellen.

- Sie müssen einen Benutzer, eine Gruppe oder eine Rolle eingerichtet haben, die über die erforderlichen Berechtigungen verfügt, um die Exportanfrage zu stellen. Weitere Informationen hierzu finden Sie unter Autorisieren einer Anfrage `$export`.
- Sie müssen eine Servicerolle erstellt haben, die HealthLake Zugriff auf den Amazon S3 S3-Bucket gewährt, in den Ihre Daten exportiert werden sollen. Die Servicerolle muss auch HealthLake als Dienstprinzipal angegeben werden. Weitere Informationen zum Einrichten von Berechtigungen finden Sie unterBerechtigungen für Exportaufträge einrichten.

Autorisieren einer Anfrage **`$export`**

Um eine erfolgreiche Exportanfrage mit der FHIR REST-API zu stellen, autorisieren Sie Ihren Benutzer, Ihre Gruppe oder Rolle entweder mit IAM oder .0. OAuth2 Sie müssen auch über eine Servicerolle verfügen.

Autorisieren einer Anfrage mithilfe von IAM

Wenn Sie eine `$export` Anfrage stellen, müssen für den Benutzer, die Gruppe oder die Rolle IAM-Aktionen in der Richtlinie enthalten sein. Weitere Informationen finden Sie unter Berechtigungen für Exportaufträge einrichten.

Autorisieren einer Anfrage mit SMART auf FHIR (2.0) OAuth

Wenn Sie eine `$export` Anfrage in einem SMART on FHIR-fähigen HealthLake Datenspeicher stellen, müssen Ihnen die entsprechenden Bereiche zugewiesen sein. Weitere Informationen finden Sie unter Die Ressourcen von SMART auf FHIR umfassen folgende Bereiche HealthLake.

> **Note**
>
> FHIR `$export` mit `GET` Anfragen erfordert dieselbe Authentifizierungsmethode oder dasselbe Trägertoken (im Fall von SMART auf FHIR), um den Export anzufordern und Dateien abzurufen. Mit FHIR `$export` mit exportierte Dateien `GET` stehen 48 Stunden lang zum Download zur Verfügung.

Eine **`$export`** Anfrage stellen

In diesem Abschnitt werden die erforderlichen Schritte beschrieben, die Sie ergreifen müssen, wenn Sie eine Exportanfrage mithilfe der FHIR REST-API stellen.

Um zu vermeiden, dass Ihr AWS Konto versehentlich belastet wird, empfehlen wir, Ihre Anfragen zu testen, indem Sie eine `POST` Anfrage stellen, ohne die `$export` Syntax anzugeben.

Um die Anfrage zu stellen, müssen Sie wie folgt vorgehen:

1. Geben Sie `$export` in der `POST` Anfrage-URL einen unterstützten Endpunkt an.
2. Geben Sie die erforderlichen Header-Parameter an.
3. Geben Sie einen Anforderungstext an, der die erforderlichen Parameter definiert.

Schritt 1: Geben Sie **`$export`** in der **`POST`** Anfrage-URL einen unterstützten Endpunkt an.

HealthLake unterstützt drei Arten von Massenexport-Endpunktanfragen. Um eine Massenexportanfrage zu stellen, müssen Sie eine `POST` Anfrage auf einem der drei unterstützten Endpunkte stellen. Die folgenden Beispiele zeigen, wo `$export` in der Anforderungs-URL angegeben werden muss.

- `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`

Sie können die folgenden unterstützten Suchparameter in der `POST` Anforderungszeichenfolge verwenden.

Unterstützte Suchparameter

HealthLake unterstützt die folgenden Suchmodifikatoren in Massenexportanfragen.

Die folgenden Beispiele enthalten Sonderzeichen, die vor dem Absenden Ihrer Anfrage codiert werden müssen.

Name	Erforderlich?	Description	Beispiel
`_outputFormat`	Nein	Das Format für die angeforderten Massendatendateien, die generiert werden sollen. Zulässige Werte sind`application/fhir+ndjson`, `application/ndjson`, `ndjson`.	

Name	Erforderlich?	Description	Beispiel
`_type`	Nein	Eine Folge von durch Kommas getrennten FHIR-Ressourcentypen, die Sie in Ihren Exportauftrag aufnehmen möchten. Wir empfehlen die Aufnahme, `_type` da sich dies negativ auf die Kosten auswirken kann, wenn alle Ressourcen exportiert werden.	`&_type=MedicationStatement,Observation`
`_since`	Nein	Ressourcentypen, die am oder nach dem Datums- und Zeitstempel geändert wurden. Wenn für einen Ressourcentyp kein Datum der letzten Aktualisierung angegeben ist, wird er in Ihre Antwort aufgenommen.	`&_since=2024-05-09T00%3A00%3A00Z`

Name	Erforderlich?	Description	Beispiel
`_until`	Nein	Ressourcentypen, die am oder vor dem Datumszeitstempel geändert wurden. Wird in Kombination mit verwendet `_since`, um einen bestimmten Zeitraum für den Export zu definieren. Wenn für einen Ressourcentyp kein Datum der letzten Aktualisierung angegeben ist, wird er von Ihrer Antwort ausgeschlossen.	`&_until=2024-12-31T23%3A59%3A59Z`

Name	Erforderlich?	Description	Beispiel
`_security`	Nein	Filtert exportierte Ressourcen nach `meta.security` Codierungswerten. Verwenden Sie das `system|code` Format. Wenn mehrere Werte angegeben werden, müssen die Ressourcen allen Werten (UND der Semantik) entsprechen. Verwenden Sie `system|` (Trailing Pipe, kein Code), um beliebigen Code aus einem bestimmten System abzugleichen.	`&_security=https://myorg.com/tenant%7Cclinic-A`
`_tag`	Nein	Filtert exportierte Ressourcen nach `meta.tag` Codierungswerten. Verwendet dasselbe `system|code` Format und dieselbe UND-Semantik wie `_security`. Wenn `_security` sowohl als auch `_tag` angegeben sind, müssen die Ressourcen beiden Filtern entsprechen.	`&_tag=https://myorg.com/dept%7Ccardiology`

Schritt 2: Geben Sie die erforderlichen Header-Parameter an

Um eine Exportanforderung mithilfe der FHIR REST-API zu stellen, müssen Sie die folgenden Header-Parameter angeben.

- Inhaltstyp: `application/fhir+json`
- Bevorzugen Sie: `respond-async`

Als Nächstes müssen Sie die erforderlichen Elemente im Anfragetext angeben.

Schritt 3: Geben Sie einen Anforderungstext an, der die erforderlichen Parameter definiert.

Die Exportanforderung erfordert auch einen Hauptteil im `JSON` Format. Der Körper kann die folgenden Parameter enthalten.

Key (Schlüssel)	Erforderlich?	Description	Wert
DataAccessRoleArn	Ja	Ein ARN einer HealthLake Servicerolle. Die verwendete Servicerolle muss HealthLake als Dienstprinzipal angegeben werden.	`arn:aws:iam:: 444455556666 :role/your-healthlake-service-role`
JobName	Nein	Der Name der Exportanforderung.	**your-export-job-name**
S3Uri	Ja	Teil eines OutputDataConfig Schlüssels. Die S3-URI des Ziel-Buckets, in den Ihre exportierten Daten heruntergeladen werden.	`s3://amzn-s3-demo-bucket/` **EXPORT-JOB** `/`
KmsKeyId	Ja	Teil eines OutputDataConfig Schlüssels.	`arn:aws:kms:` **region-of-**

Key (Schlüssel)	Erforderlich?	Description	Wert
		Der ARN des AWS KMS Schlüssels, der zur Sicherung des Amazon S3 S3-Buckets verwendet wurde.	**bucket:123456789012** :key/**1234abcd-12ab-34cd-56ef-1234567890ab**

Example Hauptteil einer Exportanfrage, die mit der FHIR REST-API gestellt wurde

Um eine Exportanfrage mithilfe der FHIR REST-API zu stellen, müssen Sie einen Hauptteil angeben, wie im Folgenden dargestellt.

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

Wenn Ihre Anfrage erfolgreich ist, erhalten Sie die folgende Antwort.

Header der Antwort

```
content-location: https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/
export/your-export-request-job-id
```

Hauptteil der Antwort

```
{
  "datastoreId": "your-data-store-id",
  "jobStatus": "SUBMITTED",
  "jobId": "your-export-request-job-id"
```

```
}
```

Verwaltung Ihrer Exportanfrage

Nachdem Sie eine erfolgreiche Exportanfrage gestellt haben, können Sie die Anfrage verwalten, indem `$export` Sie den Status einer aktuellen Exportanfrage beschreiben und `$export` eine aktuelle Exportanfrage stornieren.

Wenn Sie eine Exportanfrage mithilfe der REST-API stornieren, wird Ihnen nur der Teil der Daten in Rechnung gestellt, der bis zum Absenden der Stornierungsanforderung exportiert wurde.

In den folgenden Themen wird beschrieben, wie Sie den Status einer aktuellen Exportanfrage abrufen oder diese stornieren können.

Eine Exportanfrage stornieren

Um eine Exportanfrage zu stornieren, stellen Sie eine `DELETE` Anfrage und geben Sie die Job-ID in der Anfrage-URL an.

```
DELETE https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/export/your-export-request-job-id
```

Wenn Ihre Anfrage erfolgreich ist, erhalten Sie Folgendes.

```
{
  "exportJobProperties": {
    "jobId": "your-original-export-request-job-id",
    "jobStatus": "CANCEL_SUBMITTED",
    "datastoreId": "your-data-store-id"
  }
}
```

Wenn Ihre Anfrage nicht erfolgreich ist, erhalten Sie Folgendes.

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "not-supported",
```

```
      "diagnostics": "Interaction not supported."
    }
  ]
}
```

Beschreibung einer Exportanfrage

Um den Status einer Exportanfrage abzurufen, stellen Sie eine `GET` Anfrage mit `export` und Ihr**`export-request-job-id`**.

```
GET https://healthlake.region.amazonaws.com/datastore/datastoreId/r4/export/your-export-request-id
```

Die JSON-Antwort wird ein `ExportJobProperties` Objekt enthalten. Sie kann die folgenden Schlüssel:Wert-Paare enthalten.

Name	Erforderlich?	Description	Wert
DataAccessRoleArn	Nein	Ein ARN einer HealthLake Servicerolle. Die verwendete Servicerolle muss HealthLake als Dienstprinzipal angegeben werden.	`arn:aws:iam:: 444455556666 :role/your-healthlake-service-role`
SubmitTime	Nein	Das Datum, zu dem ein Exportauftrag gesendet wurde.	`Apr 21, 2023 5:58:02`
EndTime	Nein	Die Uhrzeit, zu der ein Exportauftrag abgeschlossen wurde.	`Apr 21, 2023 6:00:08 PM`
JobName	Nein	Der Name der Exportanforderung.	**`your-export-job-name`**
JobStatus	Nein		Folgende sind gültige Werte:

Name	Erforderlich?	Description	Wert
			`SUBMITTED | IN_PROGRESS | COMPLETED_WITH_ERRORS | COMPLETED | FAILED`
S3Uri	Ja	Teil eines OutputDataConfigObjekts. Die Amazon S3 S3-URI des Ziel-Buckets, in den Ihre exportierten Daten heruntergeladen werden.	`s3://amzn-s3-demo-bucket/` **`EXPORT-JOB /`**
KmsKeyId	Ja	Teil eines OutputDataConfigObjekts. Der ARN des AWS KMS Schlüssels, der zur Sicherung des Amazon S3 S3-Buckets verwendet wurde.	`arn:aws:kms:` **`region-of-bucket:123456789012`** `:key/`**`1234abcd-12ab-34cd-56ef-1234567890ab`**

Example: Hauptteil einer Exportanfrage zur Beschreibung, die mithilfe der FHIR REST-API gestellt wurde

Bei Erfolg erhalten Sie die folgende JSON-Antwort.

```
{
  "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",
  }
}
```

$inquireFHIR-Operation für HealthLake

Mit `$inquire` diesem Vorgang können Sie den Status einer zuvor eingereichten vorherigen Autorisierungsanfrage überprüfen. Dieser Vorgang implementiert den Da Vinci Prior Authorization Support (PAS) Implementation Guide, der einen standardisierten FHIR-basierten Workflow zum Abrufen der aktuellen Autorisierungsentscheidung bietet.

Funktionsweise

- Anfrage einreichen: Sie senden ein FHIR-Paket, das den Antrag, den Sie überprüfen möchten, und unterstützende Informationen enthält
- Suchen: HealthLake sucht ClaimResponse in Ihrem Datenspeicher nach dem entsprechenden
- Abrufen: Der neueste Autorisierungsstatus wird abgerufen
- Antworten: Sie erhalten sofort eine Antwort mit dem aktuellen Autorisierungsstatus (in der Warteschlange, genehmigt, verweigert usw.)

> **Note**
>
> `$inquire`ist ein schreibgeschützter Vorgang, der den vorhandenen Autorisierungsstatus abruft. Es werden keine Ressourcen in Ihrem Datenspeicher geändert oder aktualisiert.

API-Endpunkt

```
POST /datastore/{datastoreId}/r4/Claim/$inquire
Content-Type: application/fhir+json
```

Struktur der Anfrage

Anforderungen an das Paket

Bei Ihrer Anfrage muss es sich um eine FHIR-Bundle-Ressource handeln mit:

- Bundle.type: Muss sein `"collection"`
- bundle.entry: Muss genau eine Claim-Ressource enthalten mit:
  - `use = "preauthorization"`
  - `status = "active"`
- Referenzierte Ressourcen: Alle Ressourcen, auf die im Antrag verwiesen wird, müssen im Paket enthalten sein

> **Beispielweise abfragen**
>
> Die Ressourcen in Ihrem Eingabepaket dienen als Suchvorlage. HealthLake verwendet die bereitgestellten Informationen, um das entsprechende zu finden ClaimResponse.

Erforderliche -Ressourcen

Ressource	Kardinalität	Profil	Description
Anspruch erheben	1	Anfrage zum PAS-Anspruch	Die vorherige Genehmigung, nach der Sie sich erkundigen
Patient	1	PAS-Begünstigter Patient	Demografische Informationen zum Patienten
Organisation (Versicherer)	1	PAS-Versichererorganisation	Versicherungsgesellschaft
Organisation (Anbieter)	1	PAS anfordernde Organisation	Gesundheitsdienstleister, der die Anfrage eingereicht hat

Wichtige Suchkriterien

HealthLake sucht nach der ClaimResponse Verwendung von:

- Referenz des Patienten aus dem Antrag
- Referenz des Versicherers aus dem Antrag
- Referenz des Anbieters aus dem Antrag
- Erstellungsdatum des Antrags (als Zeitfilter)

> **Nur patientenspezifische Anfragen**
>
> Alle Anfragen müssen sich auf einen bestimmten Patienten beziehen. Systemweite Abfragen ohne Patientenidentifikation sind nicht zulässig.

Beispielanforderung

```
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"
      }
    }
  ]
}
```

Reaktionsformat

Erfolgreiche Antwort (200 OK)

Sie erhalten ein Paket mit Antworten auf Anfragen von PAS, das Folgendes enthält:

- ClaimResponsemit dem aktuellen Autorisierungsstatus; mehrere ClaimResponse, wenn er den Suchkriterien entspricht
- Alle Originalressourcen aus Ihrer Anfrage (zurückgesendet)
- Zeitstempel, zu dem die Antwort zusammengestellt wurde

Mögliche Ergebnisse ClaimResponse

Ergebnis	Description
`queued`	Die Prüfung der Autorisierungsanfrage steht noch aus
`complete`	Die Autorisierungsentscheidung wurde getroffen (auf `disposition` Genehmigt/Abgelehnt prüfen)
`error`	Bei der Verarbeitung ist ein Fehler aufgetreten

Ergebnis	Description
`partial`	Teilweise Autorisierung erteilt


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

Fehlermeldungen

400 Bad Request (400 Ungültige Anfrage)

Wird zurückgegeben, wenn das Anforderungsformat ungültig ist oder die Überprüfung fehlschlägt.

```
{
    "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 Nicht autorisiert

Wird zurückgegeben, wenn Anmeldeinformationen fehlen oder ungültig sind.

```
{
    "resourceType": "OperationOutcome",
    "issue": [
        {
            "severity": "error",
            "code": "forbidden",
            "diagnostics": "Invalid authorization header"
        }
    ]
}
```

403 Forbidden

Wird zurückgegeben, wenn der authentifizierte Benutzer nicht berechtigt ist, auf die angeforderte Ressource zuzugreifen.

```
{
    "resourceType": "OperationOutcome",
    "issue": [
        {
            "severity": "error",
            "code": "exception",
            "diagnostics": "Insufficient SMART scope permissions."
        }
    ]
}
```

400 Wenn keine gefunden wurde

Wird zurückgegeben, wenn keine Übereinstimmung ClaimResponse für die Anfrage gefunden wurde.

```
{
  "resourceType": "OperationOutcome",
```

```
  "issue": [{
    "severity": "error",
    "code": "not-found",
    "diagnostics": "Resource not found. No ClaimResponse found from the input Claim
 that matches the specified Claim properties patient, insurer, provider, and created(at
 Bundle.entry[0].resource)"
  }]
}
```

415 Medientyp wird nicht unterstützt

Wird zurückgegeben, wenn der Content-Type-Header nicht application/fhir+json lautet.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "value",
    "diagnostics": "Incorrect MIME-type. Update request Content-Type header."
  }]
}
```

429 Zu viele Anfragen

Wird zurückgegeben, wenn die Ratenlimits überschritten werden.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "throttled",
    "diagnostics": "Rate limit exceeded. Please retry after some time."
  }]
}
```

Regeln für die Validierung

HealthLake führt eine umfassende Validierung Ihrer Anfrage durch:

Validierung des Pakets

- Muss dem Profil des PAS Inquiry Request Bundle entsprechen
- `Bundle.type`muss sein `"collection"`

- Muss genau eine Claim-Ressource enthalten
- Alle Ressourcen, auf die verwiesen wird, müssen im Paket enthalten sein

Validierung des Antrags

- Muss dem PAS-Anforderungsprofil entsprechen
- `Claim.use`muss sein `"preauthorization"`
- `Claim.status`muss sein `"active"`
- Erforderliche Felder:`patient`,`insurer`,`provider`, `created`

Validierung der Ressourcen

- Alle Ressourcen müssen ihren jeweiligen PAS-Anfrageprofilen entsprechen
- Erforderliche unterstützende Ressourcen müssen vorhanden sein (Patient, Versichererorganisation, Anbieterorganisation)
- Querverweise müssen gültig und innerhalb des Pakets lösbar sein

Leistungsspezifikationen

Metrik	Spezifikation
Limit für die Anzahl der Ressourcen	500 Ressourcen pro Paket
Größenbeschränkung für Pakete	Maximal 5 MB

Erforderliche Berechtigungen

Um den `$inquire` Vorgang verwenden zu können, stellen Sie sicher, dass Ihre IAM-Rolle über Folgendes verfügt:

- `healthlake:InquirePreAuthClaim`- Um den Vorgang aufzurufen

SMART auf FHIR-Zielfernrohren

Erforderliche Mindestbereiche:

- SMART Version 1: `user/ClaimResponse.read`
- SMART v2: `user/ClaimResponse.s`

Wichtige Hinweise zur Implementierung

Suchverhalten

Wenn Sie eine Anfrage einreichen, HealthLake sucht nach der ClaimResponse Verwendung von:

- Patientenreferenz aus dem eingegebenen Antrag
- Referenz des Versicherers aus dem eingegebenen Anspruch
- Anbieterreferenz aus dem Eingabeanspruch
- Erstellungsdatum aus dem eingegebenen Claim (als Zeitfilter)

Mehrere Treffer: Wenn mehrere ClaimResponses Treffer Ihren Suchkriterien entsprechen, werden alle passenden Ergebnisse HealthLake zurückgegeben. Sie sollten den neuesten `ClaimResponse.created` Zeitstempel verwenden, um den neuesten Status zu ermitteln.

Aktualisierte Ansprüche

Wenn Sie mehrere Aktualisierungen für dieselbe vorherige Autorisierung eingereicht haben (z. B. Claim v1.1, v1.2, v1.3), ruft der `$inquire` Vorgang anhand der angegebenen Suchkriterien die der neuesten Version ClaimResponse zugeordnete Version ab.

Schreibgeschützter Vorgang

Die `$inquire` Operation:

- Ruft den vorhandenen Autorisierungsstatus ab
- Gibt den neuesten zurück ClaimResponse
- Ändert oder aktualisiert keine Ressourcen
- Erstellt keine neuen Ressourcen
- Löst keine neue Autorisierungsverarbeitung aus

Beispiel für einen Arbeitsablauf

Typischer Arbeitsablauf für Anfragen zur vorherigen Autorisierung

```
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"
```

Zugehörige -Vorgänge

- `Claim/$submit`- Reichen Sie eine neue vorherige Autorisierungsanfrage ein oder aktualisieren Sie eine bestehende
- `Patient/$everything`- Rufen Sie umfassende Patientendaten für den Kontext einer vorherigen Autorisierung ab

Konzeptdetails abrufen mit **`$lookup`**

AWS HealthLake unterstützt jetzt den `$lookup` Vorgang für CodeSystem Ressourcen, sodass Sie Details zu einem bestimmten Konzept in einem Codesystem abrufen können, indem Sie identifizierende Informationen wie den zugehörigen Code angeben. Diese Operation ist besonders nützlich, wenn Sie:

- Rufen Sie detaillierte Informationen zu bestimmten medizinischen Codes ab
- Überprüfen Sie die Bedeutungen und Eigenschaften von Codes
- Greifen Sie auf Konzeptdefinitionen und Beziehungen zu
- Support Sie klinische Entscheidungen mit genauen Terminologiedaten

Usage

Der `$lookup` Vorgang kann für CodeSystem Ressourcen sowohl mit der GET- als auch mit der POST-Methode aufgerufen werden:

Unterstützte Vorgänge

```
GET [base]/CodeSystem/$lookup?system=http://snomed.info/
sct&code=73211009&version=20230901
POST [base]/CodeSystem/$lookup
```

Unterstützte Parameter

HealthLake unterstützt eine Teilmenge von FHIR R4-Parametern: `$lookup`

Parameter	Typ	Erforderlich	Description
`code`	Code	Ja	Der Konzeptcode, nach dem Sie suchen (z. B. „71620000" in SNOMED CT)
`system`	uri	Ja	Die kanonische URL des Codesystems (z. B. "http://snomed.info/sct „)
`version`	Zeichenfolge	Nein	Spezifische Version des Codesystems

Beispiele

GET-Anfrage

```
GET [base]/CodeSystem/$lookup?system=http://snomed.info/
sct&code=71620000&version=2023-09
```

POST-Anfrage

```
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"
    }
  ]
}
```

Beispielantwort

Die Operation gibt eine Parameter-Ressource zurück, die die Konzeptdetails enthält:

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

Antwortparameter

Die Antwort enthält, sofern verfügbar, die folgenden Parameter:

Parameter	Typ	Description
`name`	Zeichenfolge	Name des Codesystems
`version`	Zeichenfolge	Version des Codesystems
`display`	Zeichenfolge	Anzeigename des Konzepts
`designation`	BackboneElement	Zusätzliche Darstellungen für dieses Konzept.
`property`	BackboneElement	Zusätzliche Eigenschaften des Konzepts (Definition, Beziehungen usw.)

Behavior

Die `$lookup` Operation:

1. Validiert die erforderlichen Parameter (`codeundsystem`)
2. Sucht innerhalb des angegebenen Codesystems, das im Datenspeicher gespeichert ist, nach dem Konzept
3. Gibt detaillierte Konzeptinformationen zurück, einschließlich Anzeigenamen, Bezeichnungen und Eigenschaften.
4. Unterstützt versionsspezifische Suchvorgänge, wenn der Parameter angegeben wird `version`
5. Funktioniert nur auf Codesystemen, die explizit im Datenspeicher gespeichert sind HealthLake

Fehlerbehandlung

Der Vorgang behandelt die folgenden Fehlerbedingungen:

- 400 Schlechte Anfrage: Ungültiger `$lookup` Vorgang (fehlerhafte Anfrage oder fehlende erforderliche Parameter)
- 404 Nicht gefunden: Das Codesystem wurde nicht gefunden oder der Code wurde im angegebenen Codesystem nicht gefunden

Einschränkungen

In dieser Version wird Folgendes nicht unterstützt:

- `$lookup`Betrieb durch Aufrufen externer Terminologieserver
- `$lookup`Vorgang wird vom Datenspeicher CodeSystems verwaltet HealthLake , aber nicht explizit im Datenspeicher gespeichert

Weitere Informationen zur `$lookup` Operationsspezifikation finden Sie in der FHIR R4-Dokumentation. CodeSystem `$lookup`

`$member-add`Operation für HealthLake

Die `$member-add` FHIR-Operation fügt einer Gruppenressource, insbesondere einer Mitgliedsattributionsliste, ein Mitglied (Patient) hinzu. Dieser Vorgang ist Teil des Leitfadens zur Implementierung der DaVinci Mitgliederzuweisung und unterstützt den Abgleichprozess bei der Verwaltung von Mitgliederzuordnungen.

Endpunkt des Vorgangs

```
POST [base]/datastore/{datastoreId}/r4/Group/{groupId}/$member-add
Content-Type: application/json
```

Parameters

Der Vorgang akzeptiert eine FHIR-Parameter-Ressource mit den folgenden Parameterkombinationen:

Parameter-Optionen

Sie können eine der folgenden Parameterkombinationen verwenden:

Option 1: Mitglieds-ID + Anbieter-NPI

- `memberId` + `providerNpi`
- `memberId` + `providerNpi` + `attributionPeriod`

Option 2: Patientenreferenz + Anbieterreferenz

- `patientReference` + `providerReference`
- `patientReference` + `providerReference` + `attributionPeriod`

Einzelheiten zu den Parametern

Mitglieds-ID (optional)

Die ID des Mitglieds, das der Gruppe hinzugefügt werden soll.

Typ: Bezeichner

System: Patientenidentifikationssystem

```
{
  "name": "memberId",
  "valueIdentifier": {
    "system": "http://example.org/patient-id",
    "value": "patient-new"
  }
```

```
}
```

Anbieter NPI (optional)

Der National Provider Identifier (NPI) des zugewiesenen Anbieters.

Typ: Identifier

System: http://terminology.hl7. org/CodeSystem/NPI

```
{
  "name": "providerNpi",
  "valueIdentifier": {
    "system": "http://terminology.hl7.org/CodeSystem/NPI",
    "value": "1234567890"
  }
}
```

Patientenreferenz (optional)

Direkter Verweis auf die Patientenressource, die hinzugefügt werden soll.

Typ: Referenz

```
{
  "name": "patientReference",
  "valueReference": {
    "reference": "Patient/patient-123"
  }
}
```

Anbieterreferenz (optional)

Direkter Verweis auf die Provider-Ressource.

Typ: Referenz

```
{
  "name": "providerReference",
  "valueReference": {
    "reference": "Practitioner/provider-456"
  }
}
```

Zeitraum der Zuordnung (optional)

Der Zeitraum, für den der Patient dem Arzt zugeordnet ist.

Typ: Zeitraum

```
{
  "name": "attributionPeriod",
  "valuePeriod": {
    "start": "2024-07-15",
    "end": "2025-07-14"
  }
}
```

Beispiele anfordern

Mitglieds-ID und Anbieter-NPI verwenden

```
{
  "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"
      }
    }
```

```
  ]
}
```

Verwendung von Patienten- und Anbieterreferenzen

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

Reaktion-Format

Erfolgreiche Antwort auf den Zusatz

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

Fehlermeldungen

Ungültige Anforderungssyntax

HTTP-Status: 400 Ungültige Anfrage

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "invalid",
      "details": {
        "text": "Invalid parameter combination provided"
      }
    }
  ]
}
```

Ressource nicht gefunden

HTTP-Status: 404 nicht gefunden

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "not-found",
      "details": {
        "text": "Resource not found."
      }
    }
  ]
}
```

Versionskonflikt

HTTP-Status: 409-Konflikt

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "conflict",
      "details": {
        "text": "Resource version conflict detected"
      }
    }
  ]
}
```

Ungültiger Zuordnungsstatus

HTTP-Status: 422 Unverarbeitbare Entität

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

Geschäftsregeln

Überprüfung des Zuordnungsstatus

Der Vorgang kann nur ausgeführt werden, wenn der Zuordnungsstatus der Gruppe wie folgt lautet:

- `draft`

- `open`

Operationen sind nicht zulässig, wenn der Status lautet`final`.

Verhinderung doppelter Mitglieder

Das System verhindert das Hinzufügen doppelter Mitglieder auf der Grundlage der einzigartigen Kombination von:

- Mitglieds-ID
- Kennung des Zahlers
- Kennung des Versicherungsschutzes

Validierung des Deckungszeitraums

Wenn eine bereitgestellt `attributionPeriod` wird, muss sie innerhalb der Grenzen des Versicherungszeitraums des Mitglieds liegen. Das System führt folgende Aktionen aus:

- Suchen Sie nach der Ressource Versicherungsschutz des Mitglieds
- Verwenden Sie das neueste Coverage (höchste versionId), falls mehrere existieren
- Stellen Sie sicher, dass der Zuweisungszeitraum den Deckungszeitraum nicht überschreitet

Validierung von Referenzen

Wenn sowohl die ID als auch die Referenz für dieselbe Ressource (Patient oder Anbieter) angegeben werden, überprüft das System, ob sie derselben Ressource entsprechen.

Wenn sowohl die Felder ID als auch reference.identifier für dieselbe Ressource (Patient oder Anbieter) angegeben werden, wird ein Fehler ausgegeben.

Authentifizierung und Autorisierung

Für den Vorgang ist eine SMART-On-FHIR-Autorisierung erforderlich für:

- Leseberechtigungen — Um die Ressourcen von Patienten, Anbietern und Gruppen zu validieren
- Suchberechtigungen — Um Ressourcen anhand ihrer ID zu finden
- Berechtigungen aktualisieren — Um die Gruppenressource zu ändern

Betriebsverhalten

Aktualisierungen der Ressourcen

- Aktualisiert die Versions-ID der Gruppenressource

- Erstellt einen Verlaufseintrag mit dem ursprünglichen Ressourcenstatus vor dem Vorgang
- Fügt dem Group.Member-Array Mitgliedsinformationen hinzu mit:
  - Patientenreferenz in entity.reference
  - Zuordnungszeitraum im Zeitraum
  - Reichweite und Anbieterinformationen in Erweiterungsfeldern

Schritte zur Validierung

- Parametervalidierung — Stellt gültige Parameterkombinationen sicher
- Vorhandensein von Ressourcen — Überprüft, ob Patienten-, Anbieter- und Gruppenressourcen vorhanden sind
- Zuordnungsstatus — Bestätigt, dass der Gruppenstatus Änderungen zulässt
- Doppelte Prüfung — verhindert das Hinzufügen vorhandener Mitglieder
- Deckungsüberprüfung — Stellt sicher, dass der Zuweisungszeitraum innerhalb der Deckungsgrenzen liegt

Einschränkungen

- Alle Ressourcen, auf die verwiesen wird, müssen sich im selben Datenspeicher befinden
- Der Vorgang funktioniert nur mit Ressourcen der Member Attribution List Group
- Die Zuordnungszeiträume müssen innerhalb der Deckungsgrenzen liegen
- Gruppen mit dem Status „Endgültig“ können nicht geändert werden

`$member-match`Operation für HealthLake

AWS HealthLake unterstützt jetzt die `$member-match` Operation für Patientenressourcen und ermöglicht es Organisationen im Gesundheitswesen, anhand demografischer Daten und Versicherungsdaten die eindeutige Kennung eines Mitglieds in verschiedenen Gesundheitssystemen zu ermitteln. Dieser Vorgang ist unerlässlich, um die CMS-Konformität zu erreichen und den sicheren payer-to-payer Datenaustausch zu ermöglichen und gleichzeitig die Privatsphäre der Patienten zu wahren.

Dieser Vorgang ist besonders nützlich, wenn Sie:

- Ermöglichen Sie den sicheren Austausch von Gesundheitsdaten zwischen Organisationen
- Sorgen Sie für die Kontinuität der Patientenversorgung über verschiedene Systeme hinweg

- Support der CMS-Compliance-Anforderungen
- Erleichtern Sie die genaue Identifizierung von Mitgliedern in allen Gesundheitsnetzwerken

Usage

Der `$member-match` Vorgang kann mithilfe der POST-Methode für Patientenressourcen aufgerufen werden:

```
POST [base]/Patient/$member-match
```

Unterstützte Parameter

HealthLake unterstützt die folgenden FHIR-Parameter`$member-match`:

Parameter	Typ	Erforderlich	Standard	Description
MemberPatient	Patient	Ja	—	Patientenressource mit demografischen Informationen für das zuzuordnende Mitglied
CoverageToMatch	Abdeckung	Ja	—	Informationsquelle, die für den Abgleich mit vorhandenen Datensätzen verwendet wird
CoverageToLink	Abdeckung	Nein	—	Coverage-Ressource, die während des Abgleichs verknüpft werden soll
Zustimmung	Zustimmung	Nein	—	Zustimmungsressource für Autorisierungszwecke

Beispiele

POST-Anfrage mit Parametern

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

Beispielantwort

Die Operation gibt eine Parameter-Ressource zurück, die die passenden Ergebnisse enthält:

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

Antwortparameter

Die Antwort enthält die folgenden Parameter, wenn eine Übereinstimmung gefunden wird:

Parameter	Typ	Description
MemberIdentifier	Kennung	Die eindeutige Kennung für das übereinst immende Mitglied
MemberId	Referenz	Verweis auf die Patientenressource
Match-Algorithmus	Zeichenfolge	Typ des verwendeten Match-Algorithmus (EXACT_MATCH, STRONG_MATCH oder DEMOGRAPHIC_MATCH)
Einzelheiten zum Spiel	Zeichenfolge	Detaillierte Informationen zum Matching-Prozess
Übereinstimmende Felder	Zeichenfolge	Liste der spezifischen Felder, die erfolgreich abgeglichen wurden

Passende Algorithmen

Die `$member-match` API verwendet einen mehrstufigen Matching-Ansatz, um eine genaue Identifizierung der Mitglieder zu gewährleisten:

EXACT_MATCH

- Verwendet die Patienten-ID in Kombination mit dem Versicherungsschutz SubscriberId
- Bietet das höchste Konfidenzniveau für die Zuordnung von Mitgliedern

STRONG_MATCH

- Verwendet eine Patienten-ID mit Informationen zur Mindestabdeckung
- Bietet ein hohes Maß an Sicherheit, wenn die genauen Übereinstimmungskriterien nicht erfüllt werden

DEMOGRAPHIC_MATCH

- Verlässt sich auf grundlegende demografische Informationen
- Wird verwendet, wenn ein identifikationsbasierter Abgleich nicht möglich ist

Behavior

Die `$member-match` Operation:

- Akzeptiert demografische Daten des Patienten, Angaben zum Versicherungsschutz und optionale Einwilligungsinformationen als Eingabe
- Gibt eine eindeutige Mitgliedskennung zurück, die für nachfolgende Interaktionen verwendet werden kann
- Implementiert einen mehrstufigen Abgleich (exakt, aussagekräftig, demografisch), um eine genaue Identifizierung der Mitglieder in verschiedenen Gesundheitssystemen zu gewährleisten
- Speichert alle bereitgestellten Einwilligungsinformationen für future Autorisierungszwecke
- Unterstützt den sicheren payer-to-payer Datenaustausch unter Wahrung der Privatsphäre der Patienten
- Entspricht den CMS-Anforderungen für den Austausch von Gesundheitsdaten

Autorisierung

Die API verwendet SMART auf dem FHIR-Autorisierungsprotokoll mit den folgenden erforderlichen Bereichen:

- `system/Patient.read`
- `system/Coverage.read`
- `system/Organization.read`(bedingt)
- `system/Practitioner.read`(bedingt)
- `system/PractitionerRole.read`(bedingt)
- `system/Consent.write`(bedingt)

Fehlerbehandlung

Der Vorgang behandelt die folgenden Fehlerbedingungen:

- `400 Bad Request`: Ungültiger `$member-match` Vorgang (fehlerhafte Anfrage oder fehlende erforderliche Parameter)
- `422 Unprocessable Entity`: Keine übereinstimmenden oder mehrere Treffer gefunden

$member-remove Operation für HealthLake

Mit `$member-remove` diesem Vorgang können Sie Mitglieder aus einer FHIR-Mitgliederzuordnungsliste (Gruppenressource) in entfernen. AWS HealthLake Dieser Vorgang ist Teil des Leitfadens zur Implementierung der DaVinci Mitgliederzuweisung und unterstützt den Abgleichprozess bei der Verwaltung von Mitgliederzuweisungen.

Voraussetzungen

- AWS HealthLake FHIR-Datenspeicher
- Entsprechende IAM-Berechtigungen für Operationen HealthLake
- Zuordnungsliste für Mitglieder (Gruppenressource) im Status Entwurf oder offen

Einzelheiten des Vorgangs

Endpoint

```
POST /Group/{id}/$member-remove
```

Inhaltstyp

```
application/fhir+json
```

Parameters

Der Vorgang akzeptiert eine FHIR-Parameter-Ressource mit den folgenden optionalen Parametern:

Parameter	Kardinalität	Typ	Description
memberId	0.. 1	Kennung	Geschäfts-ID des zu entfernenden Mitglieds
Anbieter/NPI	0.. 1	Kennung	NPI des zugewiesenen Anbieters
Referenz des Patienten	0.. 1	Referenz	Direkter Verweis auf die Patienten ressource
Referenz des Anbieters	0.. 1	Referenz	Direkter Verweis auf die Anbieter-Ressource (Praktiker oder Organisat ion) PractitionerRole

Parameter	Kardinalität	Typ	Description
Referenz zum Geltungsbereich	0.. 1	Referenz	Verweis auf die Coverage-Ressource

Unterstützte Parameterkombinationen

Die folgenden Parameterkombinationen werden unterstützt:

- `memberId`nur — Entfernt alle Zuweisungen für das angegebene Mitglied
- `memberId`+ `providerNpi` — Entfernt Zuweisungen für die spezifische Kombination aus Mitglied und Anbieter
- `patientReference`nur — Entfernt alle Zuordnungen für den angegebenen Patienten
- `patientReference`+ `providerReference` — Entfernt Zuordnungen für die spezifische Kombination aus Patient und Anbieter
- `patientReference`+ `providerReference` + `coverageReference` — Entfernt die spezifische Zuordnung auf der Grundlage von Patient, Anbieter und Versicherungsschutz

Anforderungsbeispiel

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "patientReference",
      "valueReference": {
        "reference": "Patient/12345"
      }
    },
    {
      "name": "providerReference",
      "valueReference": {
        "reference": "Practitioner/67890"
      }
    }
  ]
}
```

Antwort

Erfolgreiche Antwort

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

Behavior

Anforderungen an den Status

- Die Operation funktioniert nur bei Attributionslisten mit Status oder `draft` `open`
- Listen mit `final` Status lehnen den Vorgang mit einem 422-Fehler ab

Verfahren zur Entfernung von Mitgliedern

- Entwürfe von Statuslisten: Mitglieder werden als inaktiv markiert (`inactive: true`) und ihre `changeType` Erweiterung wird aktualisiert auf `changed`
- Statuslisten öffnen: Ähnliches Verhalten wie beim Entwurfsstatus
- Endgültige Statuslisten: Der Vorgang wurde abgelehnt

Validierung

Verweise werden überprüft, um sicherzustellen, dass sie im HealthLake Datenspeicher vorhanden sind

Wenn sowohl der Bezeichner als auch der Verweis für denselben Ressourcentyp angegeben werden, müssen sie derselben Ressource entsprechen

Parameterkombinationen werden gemäß den unterstützten Mustern validiert

Fehlerbehandlung

Häufige Fehlerantworten

Ressource nicht gefunden (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"]
    }
  ]
}
```

Endgültiger Status der Attributionsliste (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"]
    }
  ]
}
```

Ungültiger Vorgang (400)

Wird zurückgegeben, wenn Parameterkombinationen ungültig oder falsch formatiert sind.

Es wurden mehrere Treffer gefunden (412)

Wird zurückgegeben, wenn die angegebenen Parameter mit mehreren Mitgliedern in der Attributionsliste übereinstimmen.

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "processing",
      "diagnostics": "Multiple members found matching the criteria"
    }
  ]
}
```

Bewährte Methoden

- Spezifische Parameter verwenden: Verwenden Sie nach Möglichkeit die spezifischste Parameterkombination, um unbeabsichtigte Löschungen zu vermeiden
- Status der Liste überprüfen: Überprüfen Sie den Status der Zuordnungsliste, bevor Sie versuchen, Dateien zu entfernen

- Gehen Sie ordnungsgemäß mit Fehlern um: Implementieren Sie eine angemessene Fehlerbehandlung für alle möglichen Fehlerbedingungen
- Referenzen validieren: Stellen Sie sicher, dass alle referenzierten Ressourcen vorhanden sind, bevor Sie die Anfrage stellen

Ressourcen für die Patientenabteilung entfernen mit **$purge**

AWS HealthLake unterstützt den `$purge` Vorgang und ermöglicht das dauerhafte Löschen aller Ressourcen in der Abteilung eines Patienten. Diese Operation ist besonders nützlich, wenn Sie:

- Entfernen Sie alle mit einem Patienten verknüpften Daten
- Erfüllen Sie Anfragen zur Entfernung von Patientendaten
- Verwalten Sie den Lebenszyklus von Patientendaten
- Führen Sie eine umfassende Bereinigung der Patientenakten durch

Usage

Der `$purge` Vorgang kann in den Patientenressourcen aufgerufen werden:

```
POST [base]/Patient/[ID]/$purge?deleteAuditEvent=true
```

Parameters

Parameter	Typ	Erforderlich	Standard	Description
`deleteAuditEvent`	Boolean	Nein	false	Wenn der Wert wahr ist, werden die zugehörigen Prüfereignisse gelöscht
`_since`	Zeichenfolge	Nein	Erstellungszeit des Datenspeichers	Wenn diese Option eingegeben wird, wird der Startgrenzwert für die Suche nach Ressourcen auf der Grundlage ihrer letzten Änderung ausgewählt. Kann nicht mit Start oder Ende verwendet werden
`start`	Zeichenfolge	Nein	Erstellungszeit des	Wenn diese Option eingegeben wird, wird der Annahmeschluss für die Suche

Parameter	Typ	Erforderlich	Standard	Description
			Datenspeichers	nach Ressourcen auf der Grundlage ihrer letzten Änderung ausgewählt. Kann mit Ende verwendet werden
end	Zeichenfolge	Nein	Zeit für die Einreichung des Job	Wenn diese Option eingegeben wird, wird der letzte Annahmeschluss für die Suche nach Ressourcen auf der Grundlage ihrer letzten Änderung ausgewählt

Beispiele

Beispielanforderung

```
POST [base]/Patient/example-patient/$purge?deleteAuditEvent=true
```

Beispielantwort

```
{
  "resourceType": "OperationOutcome",
  "id": "purge-job",
  "issue": [
    {
      "severity": "information",
      "code": "informational",
      "diagnostics": "Purge job started successfully. Job ID:
 12345678-1234-1234-1234-123456789012"
    }
  ]
}
```

Aufgabenstatus

Um den Status eines Löschauftrags zu überprüfen:

```
GET [base]/$purge/[jobId]
```

Der Vorgang gibt Informationen zum Auftragsstatus zurück:

```
{
      "datastoreId": "36622996b1fcecb7e12ee2ee085308d3",
      "jobId": "3dd1c7a5b6c0ef8c110f566eb87e2ef9",
      "status": "COMPLETED",
      "submittedTime": "2025-10-31T18:43:21.822Z"
    }
```

Behavior

Die `$purge` Operation:

1. Wird asynchron verarbeitet, um mehrere Ressourcen zu verwalten
2. Sorgt für die Datenintegrität bei ACID-Transaktionen
3. Ermöglicht die Nachverfolgung des Auftragsstatus anhand der Anzahl gelöschter Ressourcen
4. Entfernt dauerhaft alle Ressourcen im Patientenraum
5. Beinhaltet eine umfassende Auditprotokollierung der Löschaktivitäten
6. Unterstützt das selektive Löschen von Prüfereignissen

Protokollierung von Prüfungen

Der `$purge` Vorgang wird als Start FHIRBulk DeleteJob und Beschreibung protokolliert und FHIRBulk DeleteJob enthält detaillierte Informationen zum Vorgang.

Einschränkungen

- Gelöschte Ressourcen werden nicht in den Suchantworten angezeigt
- Auf Ressourcen, die gelöscht werden, kann während der Verarbeitung möglicherweise vorübergehend nicht zugegriffen werden
- Alle Ressourcen in der Patientenabteilung werden dauerhaft entfernt

`$questionnaire-package`FHIR-Operation für HealthLake

Bei diesem `$questionnaire-package` Vorgang wird ein umfangreiches Paket abgerufen, das einen FHIR-Fragebogen und all seine Abhängigkeiten enthält, die für das Rendern und Verarbeiten des Fragebogens erforderlich sind. Bei diesem Vorgang wird der Implementierungsleitfaden für Da

Vinci Documentation Templates and Rules (DTR) implementiert, der ein dynamisches Rendern von Formularen für Dokumentationsanforderungen in Arbeitsabläufen im Gesundheitswesen ermöglicht.

Funktionsweise

- Anfrage: Sie senden Parameter, die die benötigten Fragebögen sowie den Umfang und den Kontext der Bestellung angeben
- Abrufen: HealthLake sammelt den Fragebogen und alle Abhängigkeiten (ValueSets, CQL-Bibliotheken usw.)
- Package: Alle Ressourcen sind in einem standardisierten Format gebündelt
- Antworten: Sie erhalten ein Komplettpaket, das zum Rendern und zur Datenerfassung bereit ist

Anwendungsfälle

- Unterlagen zur vorherigen Autorisierung: Sammeln Sie die erforderlichen klinischen Informationen für vorherige Autorisierungsanfragen
- Deckungsanforderungen: Sammeln Sie die Unterlagen, die zur Erfüllung der Deckungsanforderungen des Kostenträgers erforderlich sind
- Klinischer Data Exchange: Strukturierung klinischer Daten für die Einreichung an Kostenträger
- Dynamische Formulare: Rendern Sie Fragebögen mit vorab ausgefüllten Patientendaten und bedingter Logik

API-Endpunkt

```
POST /datastore/{datastoreId}/r4/Questionnaire/$questionnaire-package
Content-Type: application/fhir+json
```

Anforderungsparameter

Eingabeparameter

Der Anfragetext muss eine FHIR-Parameter-Ressource mit den folgenden Parametern enthalten:

Parameter	Typ	Kardinalität	Description
`coverage`	Abdeckung	1.. * (Erforder lich)	Ressource (n) zur Bestimmung des Mitglieds und des Versicher ungsschutzes für die Dokumentation
`questionn aire`	kanonisch	0.. *	Kanonische URL (s) für bestimmte Fragebögen, die zurückgegeben werden sollen (kann Version enthalten)
`order`	Ressource	0.. *	Ordnen Sie Ressourcen (DeviceRe quest, ServiceRequest Medicatio nRequest, Begegnung, Termin) an, um den Kontext festzulegen
`changedSi nce`	dateTime	0.1	Falls vorhanden, werden nur Ressourcen zurückgegeben, die nach diesem Zeitstempel geändert wurden

Regeln für die Parametervalidierung

Mindestens EINE der folgenden Angaben muss angegeben werden (zusätzlich zu den erforderlichen Angaben`coverage`):

- Eine oder mehrere `questionnaire` kanonische URLs
- Eine oder mehrere Ressourcen `order`

Gültige Anforderungskombinationen:

- `coverage` + `questionnaire`
- `coverage` + `order`
- `coverage` + `questionnaire` + `order`

Beispielanforderung

```
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"
    }
  ]
}
```

Reaktionsformat

Erfolgreiche Antwort (200 OK)

Die Operation gibt eine FHIR-Parameter-Ressource zurück, die ein oder mehrere Paketpakete enthält. Jedes Package beinhaltet:

Art des Eintrags	Kardinalität	Description
Fragebogen	1	Der auszufüllende Fragebogen
QuestionnaireRespo nse	0.. 1	Vorab ausgefüllte oder teilweise ausgefüllte Antwort (falls zutreffend)
Bibliothek	0.. *	CQL-Bibliotheken mit Logik vor der Befüllung und bedingter Logik
ValueSet	0.. *	Erweitert ValueSets (für Antwortoptionen mit <40 Erweiterungen)

Example Beispielantwort


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

Arbeitsablauf bei der Bedienung

Wie wird Ihre Anfrage HealthLake bearbeitet

HealthLake Führt bei `$questionnaire-package` Ihrem Anruf die folgenden Schritte aus:

1. Patient und Kostenträger identifizieren: Extrahiert den Patienten und die Versicherungsorganisation aus Ihrem `coverage` Parameter.
2. Finden Sie den richtigen Fragebogen:
   - Mit **`questionnaire`** Parameter: Verwendet die von Ihnen angegebene kanonische URL
   - Mit **`order`** Parameter: Stimmt mit dem Bestellcode (CPT/HCPCS/LOINC) und dem Zahler überein, um den passenden Fragebogen zu finden
3. Abhängigkeiten erfassen: Ruft automatisch alles ab, was zum Rendern des Fragebogens benötigt wird:
   - CQL-Bibliotheken — Logik für Fragen vor der Eingabe und für bedingte Fragen
   - ValueSets- Antwortoptionen (werden automatisch erweitert, wenn weniger als 40 Optionen verfügbar sind)
   - QuestionnaireResponse- Alle vorhandenen Antworten, die gerade bearbeitet oder abgeschlossen sind
4. Alles zusammen verpacken:
   - Bündelt alle Ressourcen (jede Ressource ist nur einmal enthalten)
   - Filtert nach `changedSince` Zeitstempel, falls angegeben
   - Fügt Warnungen hinzu`Outcome`, wenn Ressourcen fehlen

Ergebnis: Ein vollständiges, eigenständiges Paket, das zum Rendern bereit ist.

Fehlermeldungen

400 Bad Request (400 Ungültige Anfrage)

Wird zurückgegeben, wenn die Überprüfung der Anfrage fehlschlägt.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "required",
    "details": {
      "text": "At least one of 'questionnaire' or 'order' must be provided along with
 'coverage'"
    }
```

```
  }]
}
```

424 Fehlgeschlagene Abhängigkeit

Wird zurückgegeben, wenn eine abhängige Ressource nicht abgerufen werden kann.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "warning",
    "code": "not-found",
    "details": {
      "text": "Referenced Library 'http://example.org/fhir/Library/missing-library'
 could not be retrieved"
    }
  }]
}
```

401 Nicht autorisiert

Wird zurückgegeben, wenn Anmeldeinformationen fehlen oder ungültig sind.

403 Forbidden

Wird zurückgegeben, wenn der authentifizierte Benutzer nicht berechtigt ist, auf die angeforderten Ressourcen zuzugreifen.

406 Nicht akzeptabel

Wird zurückgegeben, wenn der angeforderte Inhaltstyp nicht bereitgestellt werden kann.

409 Conflict (409 Konflikt)

Wird zurückgegeben, wenn ein Versions- oder Parallelitätskonflikt vorliegt.

410 Verschwunden

Wird zurückgegeben, wenn die angeforderte Ressource dauerhaft gelöscht wurde.

429 Zu viele Anfragen

Wird zurückgegeben, wenn die Ratenlimits überschritten werden.

500 Internal Server Error

Wird zurückgegeben, wenn ein unerwarteter Serverfehler auftritt.

501 Nicht implementiert

Wird zurückgegeben, wenn der angeforderte Vorgang noch nicht implementiert ist.

Validierungsregeln

Überprüfung von Eingaben

- `coverage`Parameter ist erforderlich (1.. * Kardinalität)
- Mindestens einer von `questionnaire` oder `order` muss angegeben werden
- Alle Coverage-Ressourcen müssen gültige FHIR-Ressourcen sein
- Bei allen Ressourcen der Bestellung muss es sich um gültige FHIR-Ressourcen handeln
- Canonical URLs muss ordnungsgemäß formatiert sein
- `changedSince`muss eine gültige ISO 8601 dateTime sein

QuestionnaireResponse Validierung

- `status`muss angemessen sein (`in-progress`,`completed`,`amended`)
- Die Struktur muss mit dem referenzierten Fragebogen übereinstimmen
- `basedOn`muss auf gültige Ressourcen der Bestellung verweisen
- `subject`muss auf gültige Patientenressourcen verweisen

Deduplizierung von Ressourcen

- Jede Ressource erscheint nur einmal im Paket
- Ausnahme: Verschiedene Versionen derselben Ressource können beide enthalten sein
- Ressourcen werden anhand ihrer kanonischen URL und Version identifiziert

Leistungsspezifikationen

Metrik	Spezifikation
Limit für die Anzahl der Ressourcen	500 Ressourcen pro Paket

Metrik	Spezifikation
Größenbeschränkung für Pakete	Maximal 5 MB

Erforderliche Berechtigungen

Um den `$questionnaire-package` Vorgang verwenden zu können, stellen Sie sicher, dass Ihre IAM-Rolle über Folgendes verfügt:

- `healthlake:QuestionnairePackage`- Um den Vorgang aufzurufen
- `healthlake:ReadResource`- Um den Fragebogen und die zugehörigen Ressourcen abzurufen
- `healthlake:SearchWithPost`- Um nach QuestionnaireResponse und verwandten Ressourcen zu suchen

SMART auf FHIR-Zielfernrohren

Erforderliche Mindestbereiche:

- SMART Version 1: `user/Questionnaire.read user/Library.read user/ValueSet.read user/QuestionnaireResponse.read`
- SMART v2: `user/Questionnaire.rs user/Library.rs user/ValueSet.rs user/QuestionnaireResponse.rs`

Wichtige Hinweise zur Implementierung

Strategie zum Abruf von Ressourcen

Priorität bei der Identifizierung des Fragebogens:

- Kanonische URL (falls der `questionnaire` Parameter angegeben wurde) — Höchste Priorität
- Auftragsanalyse (falls der `order` Parameter angegeben wurde):
  - Ordnen Sie die Bestellcodes (CPT, HCPCS, LOINC) den Krankenversicherungen des Zahlers zu
  - Verwenden Sie Coverage Payor, um zahlerspezifische Fragebögen zu filtern
  - Erwägen Sie Ursachencodes für zusätzlichen Kontext

Auflösung von Abhängigkeiten

CQL-Bibliotheken:

- Abgerufen über die `cqf-library` Erweiterung zu den Fragebogen-Ressourcen
- Ruft abhängige Bibliotheken rekursiv mit dem Typ ab `Library.relatedArtifact` `depends-on`
- Alle Bibliotheksabhängigkeiten sind im Paket enthalten

ValueSets:

- Wird automatisch erweitert, wenn sie weniger als 40 Konzepte enthalten
- Größere ValueSets sind ohne Erweiterung enthalten
- ValueSets Die Ressourcen, auf die sowohl im Fragebogen als auch in der Bibliothek verwiesen wird, sind enthalten

QuestionnaireResponse Vorbevölkerung

Die Operation kann in folgenden Fällen eine QuestionnaireResponse mit bereits ausgefüllten Daten zurückgeben:

- Es wurde eine bereits in Bearbeitung befindliche oder abgeschlossene Antwort gefunden
- Die CQL-Logik in zugehörigen Bibliotheken kann Daten aus Patientenakten extrahieren
- Die Antwort ist mit der jeweiligen Reihenfolge und dem Umfang verknüpft

Suchkriterien für QuestionnaireResponse:

Suchparameter	FHIR-Pfad	Description
`based-on`	`QuestionnaireResponse.basedOn`	Links zu oder ServiceRequest CarePlan
`patient`	`QuestionnaireResponse.subject`	Der Patient, der das Subjekt ist

Suchparameter	FHIR-Pfad	Description
`questionnaire`	`QuestionnaireResponse.questionnaire`	Der Fragebogen wird beantwortet

Die Filterung der Ressourcen wurde geändert

Wenn der `changedSince` Parameter bereitgestellt wird:

- Nur Ressourcen, die nach dem angegebenen Zeitstempel geändert wurden, sind enthalten
- Wenn sich keine Ressourcen geändert haben, wird ein leeres Paket zurückgegeben `200 OK`
- Nützlich für inkrementelle Updates und Caching-Strategien
- Der Zeitstempelvergleich verwendet ein Ressourcenfeld `meta.lastUpdated`

Mehrere Paketpakete

Der Vorgang kann mehrere Paketpakete zurückgeben, wenn:

- Über Canonical werden mehrere Fragebögen angefordert URLs
- Für mehrere Bestellungen sind unterschiedliche Fragebögen erforderlich
- Es gelten verschiedene Versionen desselben Fragebogens

Jedes Package ist eigenständig mit allen erforderlichen Abhängigkeiten.

Häufige Anwendungsfälle

Anwendungsfall 1: Dokumentation zur vorherigen Autorisierung

Szenario: Ein Anbieter muss vor der Genehmigung Unterlagen für eine Sauerstofftherapie zu Hause sammeln.

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

Ergebnis: Gibt ein Paket mit dem Fragebogen zur Sauerstofftherapie zurück, der bereits mit den Patientendaten und Diagnosecodes aus dem EHR ausgefüllt ist.

Anwendungsfall 2: Rufen Sie eine bestimmte Version des Fragebogens ab

Szenario: Ein Anbieter benötigt zur Einhaltung der Vorschriften eine bestimmte Version eines Fragebogens.

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "coverage",
      "resource": { /* Coverage resource */ }
    },
    {
      "name": "questionnaire",
      "valueCanonical": "http://example.org/fhir/Questionnaire/dme-request|3.1.0"
    }
  ]
}
```

Ergebnis: Gibt genau Version 3.1.0 des DME-Anforderungsfragebogens mit allen Abhängigkeiten zurück.

Anwendungsfall 3: Nach Updates suchen

Szenario: Ein Anbieter möchte überprüfen, ob Fragebogenressourcen seit dem letzten Abruf aktualisiert wurden.

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

Ergebnis: Gibt nur Ressourcen zurück, die nach dem 1. März 2024 geändert wurden, oder ein leeres Paket, falls sich nichts geändert hat.

Anwendungsfall 4: Mehrere Bestellungen

Szenario: Ein Anbieter reicht mehrere Serviceanfragen ein, für die möglicherweise unterschiedliche Fragebögen erforderlich sind.

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "coverage",
      "resource": { /* Coverage resource */ }
    },
    {
      "name": "order",
      "resource": { /* ServiceRequest for imaging */ }
    },
    {
```

```
      "name": "order",
      "resource": { /* ServiceRequest for DME */ }
    }
  ]
}
```

Ergebnis: Gibt mehrere Paketpakete zurück, eines für jeden zutreffenden Fragebogen.

Integration mit anderen Da Vinci IGs

Ermittlung der Deckungsanforderungen (CRD)

Workflow-Integration:

- Der Anbieter bestellt einen Service in seinem EHR
- Der CRD-Haken löst aus und überprüft die Deckungsanforderungen
- Der Zahler gibt an, dass Unterlagen erforderlich sind
- Der Anbieter ruft `$questionnaire-package` an, um das Dokumentationsformular abzurufen
- Der Anbieter füllt den Fragebogen aus
- Die Dokumentation wird über PAS eingereicht oder CDex

Support bei vorheriger Autorisierung (PAS)

Workflow-Integration:

- Wird verwendet`$questionnaire-package`, um Dokumentationsanforderungen abzurufen
- Vervollständigen Sie das QuestionnaireResponse mit den erforderlichen klinischen Daten
- Reichen Sie die vorherige Genehmigung ein und verwenden Sie `Claim/$submit` dabei das ausgefüllte QuestionnaireResponse
- Überprüfen Sie den Status mit `Claim/$inquire`

Klinischer Data Exchange (CDex)

Workflow-Integration:

- Der Zahler fordert zusätzliche Unterlagen für einen Anspruch an

- Der Anbieter verwendet`$questionnaire-package`, um das Formular zur Erfassung strukturierter Daten abzurufen
- Der Anbieter füllt das aus QuestionnaireResponse
- Die Dokumentation wird dem Zahler über den CDex Anhangs-Workflow übermittelt

Anleitung zur Fehlerbehebung

Problem: Es wurde kein Fragebogen zurückgesendet

Mögliche Ursachen:

- Die kanonische URL stimmt mit keinem Fragebogen im Datenspeicher überein
- Der Bestellcode entspricht keinem Fragebogen in der Krankenversicherung des Zahlers
- Dem Versicherungsnehmer sind keine Fragebögen zugeordnet

Lösungen:

- Stellen Sie sicher, dass die kanonische URL korrekt ist und der Fragebogen existiert
- Vergewissern Sie sich, dass die Bestellcodes (CPT/HCPCS) korrekt angegeben sind
- Vergewissern Sie sich, dass die Kostenträgerorganisation Fragebögen konfiguriert hat

Problem: Fehlende Abhängigkeiten im Package

Mögliche Ursachen:

- Referenzierte Bibliothek oder ValueSet existiert nicht im Datenspeicher
- Bibliotheksverweise sind defekt oder falsch
- ValueSet Die Erweiterung ist fehlgeschlagen

Lösungen:

- Überprüfen Sie den `Outcome` Parameter auf Warnungen zu fehlenden Ressourcen
- Stellen Sie sicher, dass alle referenzierten Ressourcen in Ihrem Datenspeicher vorhanden sind

- Stellen Sie sicher ValueSet URLs , dass sie korrekt und lösbar sind

Problem: Leeres Package mit ChangedSince

Mögliche Ursachen:

- Dies ist erwartetes Verhalten — seit dem angegebenen Zeitstempel wurden keine Ressourcen geändert

Lösungen:

- Verwenden Sie die zwischengespeicherte Version des Pakets
- Entfernen Sie den `changedSince` Parameter, um das vollständige Paket abzurufen

Problem: QuestionnaireResponse Nicht vorab ausgefüllt

Mögliche Ursachen:

- Keine vorhandenen QuestionnaireResponse gefunden
- Die Logik der CQL-Bibliothek konnte die erforderlichen Daten nicht extrahieren
- Patientendaten fehlen oder sind unvollständig

Lösungen:

- Das ist zu erwarten — nicht alle Fragebögen basieren auf einer Präpopulationslogik
- Prüfen Sie, ob Patientendaten im Datenspeicher vorhanden sind
- Überprüfen Sie die Anforderungen an die Datenextraktion in der Logik der CQL-Bibliothek

Best Practices

1. Verwenden Sie Canonical URLs mit Versionen

Geben Sie immer Versionsnummern an, wenn Sie bestimmte Fragebögen anfordern:

```
{
```

```
  "name": "questionnaire",
  "valueCanonical": "http://example.org/fhir/Questionnaire/dme|2.1.0"
}
```

Warum: Sorgt für Konsistenz und verhindert unerwartete Änderungen, wenn Fragebögen aktualisiert werden.

2. Nutzen Sie ChangedSince für mehr Leistung

Verwenden Sie für häufig aufgerufene Fragebögen Folgendes, `changedSince` um die Datenübertragung zu minimieren:

```
{
  "name": "changedSince",
  "valueDateTime": "2024-03-10T15:30:00Z"
}
```

Warum: Reduziert die Latenz und die Bandbreitennutzung, indem nur aktualisierte Ressourcen abgerufen werden.

3. Vollständige Deckungsinformationen einschließen

Geben Sie umfassende Angaben zum Umfang an, um sicherzustellen, dass der Fragebogen korrekt ausgewählt wird:

```
{
  "name": "coverage",
  "resource": {
    "resourceType": "Coverage",
    "beneficiary": { "reference": "Patient/123" },
    "payor": [{ "reference": "Organization/payer-abc" }],
    "class": [{ /* Group/plan information */ }]
  }
}
```

Warum: Hilft bei der HealthLake Identifizierung von zahlerspezifischen Fragebögen und Anforderungen.

Zugehörige -Vorgänge

- `Claim/$submit`- Reichen Sie vorherige Autorisierungsanfragen mit ausgefüllten Unterlagen ein

- `Claim/$inquire`- Überprüfen Sie den Status der eingereichten vorherigen Genehmigungen
- `ValueSet/$expand`- Erweitern Sie ValueSets die Antwortmöglichkeiten

`$submit`FHIR-Operation für HealthLake

Dieser `$submit` Vorgang ermöglicht es Ihnen, vorherige Autorisierungsanträge elektronisch zur Genehmigung an die Zahler einzureichen. Dieser Vorgang implementiert den Da Vinci Prior Authorization Support (PAS) Implementation Guide, der einen standardisierten FHIR-basierten Workflow für vorherige Autorisierungsanträge bietet.

Funktionsweise

- Einreichen: Sie senden ein FHIR-Paket, das Ihre vorherige Autorisierungsanfrage und unterstützende klinische Daten enthält
- Bestätigen: HealthLake validiert die Einreichung anhand der PAS-Anforderungen
- Fortbestehen: Alle Ressourcen werden in Ihrem HealthLake Datenspeicher gespeichert
- Antworten: Sie erhalten sofort eine Antwort mit dem Status „In der Warteschlange“
- Prozess: Die Autorisierungsentscheidung wird vom Zahler asynchron verarbeitet

API-Endpunkt

```
POST /datastore/{datastoreId}/r4/Claim/$submit
Content-Type: application/fhir+json
```

Struktur der Anfrage

Anforderungen an das Paket

Bei Ihrer Anfrage muss es sich um eine FHIR-Bundle-Ressource handeln mit:

- Bundle.type: Muss sein `"collection"`
- bundle.entry: Muss genau eine Claim-Ressource enthalten mit `use = "preauthorization"`
- Referenzierte Ressourcen: Alle Ressourcen, auf die der Anspruch verweist, müssen im Paket enthalten sein

Erforderliche -Ressourcen

Ressource	Kardinalität	Profil	Description
Anspruch erheben	1	PAS-Anspruch	Die vorherige Autorisierungsanfr age
Patientin	1	PAS-Patient	Demografische Informationen zum Patienten
Organisation (Versicherer)	1	PAS-Versicherer	Versicherungsgesellschaft
Organisation (Anbieter)	1	PAS-Anforderer	Gesundheitsdienstleister, der die Anfrage einreicht
Deckung	1 oder mehr	PAS-Abdeckung	Einzelheiten zum Versicher ungsschutz

Optionale Ressourcen

Ressource	Kardinalität	Profil	Description
Praktiker	0 oder mehr	PAS-Praktiker	Praktiker im Gesundheitswesen
Practitio nerRole	0 oder mehr	PAS PractitionerRole	Rollen von Praktikern
ServiceRe quest	0 oder mehr	PAS ServiceRequest	Angeforderte medizinische Leistungen
DeviceRequest	0 oder mehr	PAS DeviceRequest	Angeforderte medizinische Geräte
Medicatio nRequest	0 oder mehr	PAS MedicationRequest	Angeforderte Medikamente
DocumentR eference	0 oder mehr	PAS DocumentR eference	Unterstützung der klinischen Dokumentation

Beispielanforderung

```
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"
    }
  }]
}
```

Reaktionsformat

Erfolgreiche Antwort (200 OK)

Sie erhalten ein PAS-Antwortpaket mit:

- ClaimResponsemit `outcome: "queued"` und `status: "active"`
- Alle Originalressourcen aus Ihrer Anfrage
- Zeitstempel zur Bestätigung des Empfangs

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

Fehlermeldungen

400 Bad Request (400 Ungültige Anfrage)

Wird zurückgegeben, wenn das Anforderungsformat ungültig oder falsch formatiert ist.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "invalid",
    "diagnostics": "The provided payload was invalid and could not be parsed
 correctly."
  }]
}
```

412 Precondition Failed (412 Vorbedingung fehlgeschlagen)

Wird zurückgegeben, wenn dieselbe vorherige Autorisierungsanfrage bereits eingereicht wurde (doppelte Einreichung erkannt).

```
{
  "resourceType": "OperationOutcome",
```

```
  "issue": [{
    "severity": "error",
    "code": "processing",
    "diagnostics": "PreAuth Claim already exists"
  }]
}
```

> **Idempotenz**
>
> Die `$submit` Operation ist idempotent. Wenn Sie dieselbe Anfrage mehrmals einreichen, werden keine doppelten vorherigen Autorisierungsanfragen erstellt. Stattdessen erhalten Sie eine 412-Fehlermeldung, in der Sie aufgefordert werden, den Status Ihrer ursprünglichen Einreichung `$inquire` zu überprüfen.

4.2.2 Unverarbeitbare Entität

Wird zurückgegeben, wenn die FHIR-Validierung fehlschlägt.

```
{
  "resourceType": "OperationOutcome",
  "issue": [{
    "severity": "error",
    "code": "required",
    "diagnostics": "Bundle contains more than one preauthorization claim"
  }]
}
```

Regeln für die Validierung

HealthLake führt eine umfassende Validierung Ihrer Einreichung durch:

Validierung des Pakets

- Muss dem PAS Request Bundle-Profil entsprechen
- `Bundle.type`muss sein `"collection"`
- Kann mehrere Claim-Ressourcen enthalten
- Muss jedoch genau eine Claim-Ressource mit Vorautorisierung enthalten
  - Und diese Claim-Ressource muss der erste Eintrag im Paket sein

- Alle Ressourcen, auf die verwiesen wird, müssen im Paket enthalten sein

Validierung des Anspruchs

- Muss dem PAS-Antragsprofil entsprechen
- `Claim.use`muss sein `"preauthorization"`
- Erforderliche Felder:`patient`,`insurer`,`provider`,`created`, `priority`
- Geschäftskennungen müssen vorhanden und gültig sein

Validierung der Ressourcen

- Alle Ressourcen müssen ihren jeweiligen PAS-Profilen entsprechen
- Erforderliche unterstützende Ressourcen müssen vorhanden sein (Patient, Versicherungsschutz, Organisation)
- Querverweise müssen gültig und innerhalb des Pakets lösbar sein

Leistungsspezifikationen

Metrik	Spezifikation
Größenbeschränkung für Pakete	Maximal 5 MB
Limit für die Anzahl der Ressourcen	500 Ressourcen pro Paket

Erforderliche Berechtigungen

Um den `$submit` Vorgang zu verwenden, kann man entweder AWS Sigv4 oder SMART auf FHIR verwenden:

- Stellen Sie sicher, dass Ihre IAM-Rolle Folgendes hat: `healthlake:SubmitPreAuthClaim` - Um den Vorgang aufzurufen

SMART auf FHIR-Zielfernrohren

Erforderliche Mindestumfänge:

- SMART Version 1: `user/Claim.write & <all_resourceTypes_in_Bundle>.write`
- SMART v2: `user/Claim.c & <all_resourceTypes_in_Bundle>.c or system/*.*`

Wichtige Hinweise zur Implementierung

Persistenz der Ressourcen

- Alle Bundle-Einträge werden als einzelne FHIR-Ressourcen in Ihrem Datenspeicher gespeichert
- Vom Kunden bereitgestellte Daten bleiben erhalten, wenn IDs sie bereitgestellt werden
- Der Versionsverlauf wird zu Prüfungszwecken aufbewahrt
- Die Erkennung von Duplikaten verhindert Ressourcenkonflikte

Verhalten bei der Verarbeitung

- Jede gültige Einreichung führt zu genau einer ClaimResponse mit `"queued"` Ergebnis
- Bei ungültigen Einsendungen werden die Statuscodes 400 oder 422 mit detaillierten Fehlerinformationen zurückgegeben
- Bei Systemfehlern werden die entsprechenden 5xx-Statuscodes zurückgegeben
- Alle erfolgreichen Einsendungen geben den Status 200 zurück und geben den Status „Ausstehend“ zurück ClaimResponse

Anforderungen für das Paket

- `Bundle.entry.fullUrl`Die Werte müssen entweder REST URLs oder `"urn:uuid:[guid]"` Format sein
- Alle GUIDs müssen bei allen Einsendungen eindeutig sein (mit Ausnahme derselben Ressourceninstanzen)
- Ressourcen, auf die verwiesen wird, müssen im Paket vorhanden sein oder auflösbar sein

Zugehörige -Vorgänge

- `Claim/$inquire`- Fragen Sie den Status einer zuvor eingereichten Autorisierungsanfrage ab
- `Patient/$everything`- Rufen Sie umfassende Patientendaten für den Kontext einer vorherigen Autorisierung ab

Validierung von FHIR-Ressourcen mit **`$validate`**

AWS HealthLake unterstützt jetzt den `$validate` Vorgang für FHIR-Ressourcen, sodass Sie eine Ressource anhand der FHIR-Spezifikation validieren und ihre Konformität mit einem bestimmten Profil oder einer Basisressourcendefinition überprüfen können, ohne Speichervorgänge durchführen zu müssen. Dieser Vorgang ist besonders nützlich, wenn Sie:

- Überprüfen Sie die FHIR CMS-Compliance-Anforderungen
- Testen Sie Ressourcen, bevor Sie sie in der Produktion verwenden
- Geben Sie Feedback zur Validierung in Echtzeit, während Benutzer klinische Daten bearbeiten
- Reduzieren Sie die Anzahl ungültiger Dateneinreichungen bei der Erstellung und Aktualisierung APIs

Usage

Der `$validate` Vorgang kann mit POST-Methoden auf FHIR-Ressourcen aufgerufen werden:

Unterstützte Vorgänge

```
POST [base]/[type]/[id]/$validate
POST [base]/[type]/$validate
```

Unterstützte Payloads

Ressource „Parameter“

HealthLake unterstützt die folgenden `$validate` FHIR-Parameter:

Parameter	Typ	Erforderlich	Description
`resource`	Ressource	Ja	Die zu validierende Ressource
`profile`	kanonisch	Nein	Kanonische URL des Profils, gegen das validiert werden soll
`mode`	Code	Nein	Validierungsmodus:`create`, oder `update`

Direkte Ressource mit Abfrageparametern

Parameter	Typ	Erforderlich	Description
`profile`	kanonisch	Nein	Kanonische URL des Profils, gegen das validiert werden soll
`mode`	Code	Nein	Validierungsmodus:`create`, oder `update`

Beispiele

POST-Anfrage für eine Ressource mit ID und Payload mit Parametern

```
POST [base]/Patient/example-patient/$validate
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "resource",
      "resource": {
        "resourceType": "Patient",
        "id": "example-patient",
        "name": [
          {
            "family": "Smith",
            "given": ["John"]
          }
        ],
        "gender": "male",
        "birthDate": "1990-01-01"
      }
    },
    {
      "name": "profile",
      "valueCanonical": "http://hl7.org/fhir/us/core/StructureDefinition/us-core-
patient"
    },
    {
```

```
      "name": "mode",
      "valueString": "create"
    }
  ]
}
```

POST-Anforderung für die Payload des Ressourcentyps und der Parameter

```
POST [base]/Patient/$validate
Content-Type: application/fhir+json

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "resource",
      "resource": {
        "resourceType": "Patient",
        "name": [
          {
            "family": "Doe",
            "given": ["Jane"]
          }
        ],
        "gender": "female",
        "birthDate": "1985-05-15"
      }
    },
    {
      "name": "profile",
      "valueCanonical": "http://hl7.org/fhir/us/core/StructureDefinition/us-core-
patient"
    },
    {
      "name": "mode",
      "valueString": "update"
    }
  ]
}
```

POST-Anfrage für eine Ressource mit ID und direkter Ressourcennutzlast

```
POST [base]/Patient/example-patient/$validate?profile=http://hl7.org/fhir/us/core/
StructureDefinition/us-core-patient&mode=create
Content-Type: application/fhir+json

{
    "resourceType": "Patient",
    "id": "example-patient",
    "name": [
        {
        "family": "Smith",
        "given": ["John"]
        }
    ],
    "gender": "male",
    "birthDate": "1990-01-01"
}
```

POST-Anfrage für den Ressourcentyp und die direkte Ressourcennutzlast

```
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"
}
```

Beispielantwort

Der Vorgang gibt eine OperationOutcome Ressource mit Überprüfungsergebnissen zurück:

```
{
  "resourceType": "OperationOutcome",
```

```
  "issue": [
    {
      "severity": "information",
      "code": "informational",
      "diagnostics": "Validation successful"
    }
  ]
}
```

Beispielantwort mit Validierungsfehlern

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

Behavior

Die `$validate` Operation:

1. Validiert die Ressource anhand der FHIR-Spezifikation und der Basisressourcendefinition
2. Überprüft die Konformität mit den angegebenen Profilen, wenn der `profile` Parameter angegeben wird
3. Validiert basierend auf dem angegebenen Modus (oder) `create` `update`
4. Gibt detaillierte Überprüfungsergebnisse zurück, einschließlich Fehlern, Warnungen und Informationsmeldungen
5. Führt keine Speichervorgänge durch — nur Validierung
6. Gibt HTTP 200 OK zurück, wenn die Validierung durchgeführt werden kann, unabhängig davon, ob Validierungsprobleme gefunden wurden

Validierungsmodi

- create: Validiert die Ressource so, als ob sie erstellt würde (neue Ressource)
- update: Validiert die Ressource so, als ob sie aktualisiert würde (bestehende Ressource)

Fehlerbehandlung

Die Operation gibt Folgendes zurück:

- 200 OK: Die Validierung wurde erfolgreich durchgeführt (unabhängig vom Überprüfungsergebnis)
- 400 Schlechte Anfrage: Ungültiges Anforderungsformat oder ungültige Parameter
- 404 Nicht gefunden: Ressourcentyp oder Profil nicht gefunden

Weitere Informationen zur `$validate` Operationsspezifikation finden Sie in der FHIR `$validate` R4-Ressourcendokumentation.

Compliance-Referenz für AWS HealthLake

AWS HealthLake bietet Funktionen, mit denen Sie die API-Nutzung gemäß den Interoperabilitätsanforderungen von CMS (Centers for Medicare & Medicaid Services) verfolgen und melden können. Diese Funktionen ermöglichen es Ihnen, API-Transaktionen nach CMS-vorgeschriebenen Kategorien zu kategorisieren und automatisch Nutzungskennzahlen für Compliance-Berichtszwecke zu erfassen.

⚠ Verstehen Sie Ihre Compliance-Pflichten

Die Verwendung AWS HealthLake und die zugehörigen CMS-Interoperabilitätsendpunkte allein reichen nicht aus, um die CMS-Konformität zu erreichen. Sie sind verantwortlich für:

- Ordnen Sie Ihre API-Workflows auf der Grundlage Ihrer spezifischen Anwendungsfälle und regulatorischen Verpflichtungen korrekt den entsprechenden Endpunkten der CMS-Kategorie zu
- Implementierung geeigneter Authentifizierungs- und Autorisierungskontrollen, die den CMS-Anforderungen entsprechen
- Stellen Sie sicher, dass Ihre FHIR-Ressourcen und der Datenaustausch den geltenden CMS-Vorschriften und Implementierungsleitfäden entsprechen
- Konfiguration und Überwachung der CloudWatch Kennzahlen zur Unterstützung Ihrer Anforderungen an die Compliance-Berichterstattung
- Verstehen Sie, welche CMS-Regeln für Ihr Unternehmen gelten, und implementieren Sie die entsprechenden technischen und betrieblichen Kontrollen

AWS HealthLake stellt die Infrastruktur und die Tools zur Unterstützung Ihrer Compliance-Bemühungen bereit. Sie müssen diese Funktionen jedoch entsprechend Ihren spezifischen regulatorischen Anforderungen einsetzen. Durch die einfache Weiterleitung von API-Aufrufen über diese Endpunkte entspricht Ihre Anwendung nicht automatisch den CMS-Vorschriften.

Themen

- CMS-Compliance-Funktionen

CMS-Compliance-Funktionen

AWS HealthLake bietet Funktionen, mit denen Sie die Interoperabilitäts- und Compliance-Anforderungen von CMS (Centers for Medicare & Medicaid Services) erfüllen können. Mit diesen Funktionen können Sie die API-Nutzung nach CMS-Kategorien verfolgen und anschließend Nutzungskennzahlen zu Compliance-Zwecken melden.

Themen

- Endpunkte der CMS-Interoperabilität
- Verbesserte CloudWatch Metriken für die CMS-Konformität

Endpunkte der CMS-Interoperabilität

-Übersicht

HealthLake bietet vier CMS-Interoperabilitätsendpunkte, die den vom CMS vorgeschriebenen API-Kategorien entsprechen. Die zugrunde liegende Basis-URL Ihres HealthLake Datenspeichers ändert sich nicht. Diese Endpunkte bieten lediglich eine Möglichkeit, Ihre API-Aufrufe für CMS-Berichtszwecke zu kategorisieren und nachzuverfolgen.

Zweck

Der Hauptzweck dieser Interoperabilitätsendpunkte besteht darin, Kunden Folgendes zu ermöglichen:

- Einfache Nachverfolgung von API-Transaktionen nach CMS-Kategorien
- Melden Sie automatisch Nutzungskennzahlen zur Einhaltung der CMS-Vorschriften
- Behalten Sie bestehende FHIR-Workflows mit minimalen Änderungen bei

Alle API-Aufrufe funktionieren identisch, unabhängig davon, ob Sie den Interoperabilitäts-Endpunkt oder den Standard-FHIR-Endpunkt verwenden. Der einzige Unterschied besteht darin, wie die Transaktionen in Metriken kategorisiert werden. CloudWatch

Unterstützte CMS-Interoperabilitätsendpunkte

CMS-Kategorie	Interoperabilitäts-	Beispielverwendung
Zugang für Patienten	`/patientaccess/v2/r4`	`baseURL/patientaccess/v2/r4/Patient/123`
Zugang zum Anbieter	`/provideraccess/v2/r4`	`baseURL/provideraccess/v2/r4/Observation?patient=123`
Zahler zu Zahler	`/payertopayerdx/v2/r4`	`baseURL/payertopayerdx/v2/r4/Practitioner/456`
Vorheriger Authentifizierungsdienst	`/priorauthservice/v2/r4`	`baseURL/priorauthservice/v2/r4/ExplanationOfBenefit?patient=789`

Wie funktionieren Interoperabilitätsendpunkte

HealthLake Standard-API-Aufruf:

```
baseURL/resourceType/[id]
baseURL/resourceType?[parameters]
```

Mit CMS-Interoperabilitätsendpunkt:

```
baseURL/interoperability-endpoint/resourceType/[id]
baseURL/interoperability-endpoint/resourceType?[parameters]
```

Der Pfad zum Interoperabilitätsendpunkt wird einfach zwischen Ihrer Basis-URL und dem Ressourcentyp eingefügt. Alles, was hinter dem Pfad zum Interoperabilitätsendpunkt steht, bleibt genau so wie bei Ihren aktuellen API-Aufrufen.

Verwendungsbeispiele

Beispiel 1: API für den Patientenzugriff

Aktueller API-Aufruf (funktioniert immer noch):

```
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"
```

Mit dem Interoperabilitätsendpunkt Patient Access (für CMS-Tracking):

```
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"
```

Die wichtigsten Punkte:

- Die Basis-URL bleibt bestehen: `https://healthlake.us-east-1.amazonaws.com/datastore/abc123`
- Interoperabilitäts-Endpunkt eingefügt: `/patientaccess/v2/r4`

- Ressourcenpfad unverändert: `/Patient/123`
- Beide Anrufe geben identische Antworten zurück
- Der Anruf am Interoperabilitäts-Endpunkt wird automatisch unter `URIType=patient-access` in nachverfolgt CloudWatch
- Die Operationen POST, PUT, PATCH und DELETE funktionieren identisch.
- Der Text der Anfrage bleibt unverändert.

Referenz zur Endpunkt-Übersetzung

Interoperabilitäts-	Übersetzt in	CMS-Kategorie
`baseURL/patientaccess/v2/r4/Patient`	`baseURL/r4/Patient`	Zugang für Patienten
`baseURL/provideraccess/v2/r4/Observation`	`baseURL/r4/Observation`	Zugang zum Anbieter
`baseURL/payertopayerdx/v2/r4/Practitioner/456`	`baseURL/r4/Practitioner/456`	Data Exchange von Zahler zu Zahler
`baseURL/priorauthservice/v2/r4/ExplanationOfBenefit?patient=789`	`baseURL/r4/ExplanationOfBenefit?patient=789`	Vorherige Autorisierung

Wichtige Hinweise

- Kein funktionaler Unterschied: Interoperabilitätsendpunkte und Standard-FHIR-Endpunkte geben identische Antworten zurück und unterstützen identische Operationen
- Unveränderte Basis-URL: Ihr HealthLake Datenspeicher-Endpunkt bleibt derselbe
- Einfache Integration: Fügen Sie den Pfad zum Interoperabilitäts-Endpunkt zwischen Ihrer Basis-URL und dem Ressourcentyp ein
- Automatisches Tracking: Die CloudWatch Metriken kategorisieren Anrufe automatisch nach dem verwendeten Interoperabilitätsendpunkt

- Abwärtskompatibel: Bestehende API-Aufrufe ohne Interoperabilitätsendpunkte funktionieren weiterhin normal

Verbesserte CloudWatch Metriken für die CMS-Konformität

-Übersicht

Wenn Sie die CMS-Interoperabilitätsendpunkte verwenden, HealthLake werden automatisch erweiterte CloudWatch Messwerte mit zusätzlichen Dimensionen ausgegeben, um die CMS-Berichtsanforderungen zu erfüllen. Diese Metriken verfolgen die API-Nutzung nach Anruferidentität, Anwendung und CMS-spezifischen URI-Typen, ohne dass eine zusätzliche Konfiguration erforderlich ist.

Funktionsweise

Wenn Sie einen API-Aufruf über einen Interoperabilitäts-Endpunkt tätigen:

```
# 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.)
```

Es ist kein zusätzlicher Code oder eine zusätzliche Konfiguration erforderlich. Verwenden Sie einfach die Interoperabilitätsendpunkte, und die erweiterten Metriken werden automatisch erfasst.

> **Note**
>
> Bei Datenspeichern ohne SMART on FHIR wird die `URIType` Dimension trotzdem erfasst, sodass Sie die API-Nutzung nach CMS-Kategorien verfolgen können. Die `ClientId` Dimensionen `Sub` und sind nur verfügbar, wenn SMART für die FHIR-Authentifizierung mit Inhaber-Tokens verwendet wird, die diese Ansprüche enthalten.

Neue metrische Dimensionen

Zusätzlich zu den vorhandenen Dimensionen (`DatastoreId`,`DatastoreType`,`Operation`) werden die folgenden Dimensionen automatisch hinzugefügt, wenn Interoperabilitätsendpunkte verwendet werden:

Dimension	Description	Beispielwerte	Quelle
URIType	CMS-Konformitätskategorie	`patient-access` , `provider-access` , `payer-to-payer` , `prior-authorization`	Wird automatisch anhand des Interoperabilitäts-Endpunktpfads
Sub	Identität des Anrufers	Benutzer-/Entitäts-ID	Aus dem Inhaber-Token-Anspruch extrahiert `sub`
ClientId	Anwendungs-Identifikator	`portal_app` , `ehr_system`	Aus dem Inhaber-Token-Anspruch extrahiert `client_id`

Verfügbare Metriken

Alle vorhandenen HealthLake Metriken enthalten jetzt die zusätzlichen Dimensionen bei der Verwendung von Interoperabilitätsendpunkten:

- CallCount- Gesamtzahl der API-Aufrufe
- Latenz — API-Antwortzeit in Millisekunden
- UserErrors- Anzahl der 4xx-Client-Fehler
- SystemErrors- Anzahl der 5xx Serverfehler
- Drosseln — Anzahl der gedrosselten Anfragen
- SuccessfulRequests- Anzahl der erfolgreichen API-Aufrufe

Abfragen von Metriken in CloudWatch

CloudWatch Beispiel für eine Insights-Abfrage

Alle API-Aufrufe für den Patientenzugriff nach Anwendung abfragen:

```
SELECT SUM(CallCount)
FROM "AWS/HealthLake"
WHERE DatastoreId = '75c1cf9b0d71cd38fec8f7fb317c4c1a'
    AND URIType = 'patient-access'
GROUP BY ClientId
```

Support-Referenz für AWS HealthLake

Das folgende unterstützende Referenzmaterial ist verfügbar für AWS HealthLake.

> **Note**
>
> Alle systemeigenen HealthLake Aktionen und Datentypen werden in einer separaten Referenz beschrieben. Weitere Informationen finden Sie in der AWS HealthLake -API-Referenz.

Themen

- AWS HealthLake Endpunkte und Kontingente
- Synthea vorinstallierte Datentypen für HealthLake
- AWS HealthLake Beispielprojekte
- Problembehebung AWS HealthLake
- Verwendung HealthLake mit einem AWS SDK

AWS HealthLake Endpunkte und Kontingente

Die folgenden Themen enthalten Informationen zu AWS HealthLake Dienstendpunkten und -kontingenten.

Themen

- Service-Endpunkte

- Servicekontingente

Service-Endpunkte

Ein Dienstendpunkt ist eine URL, die einen Host und einen Port als Einstiegspunkt für einen Webdienst identifiziert. Jede Webserviceanforderung umfasst einen Endpunkt. Die meisten AWS Dienste bieten Endpunkte für bestimmte Regionen, um eine schnellere Konnektivität zu ermöglichen. In der folgenden Tabelle sind die Dienstendpunkte für aufgeführt. AWS HealthLake

Name der Region	Region	Endpunkt	Protocol (Protokoll)
USA Ost (Ohio)	us-east-2	healthlake.us-east-2.amazonaws.com	HTTPS
		healthlake-fips.us-east-2.amazonaws.com	HTTPS
USA Ost (Nord-Virginia)	us-east-1	healthlake.us-east-1.amazonaws.com	HTTPS
		healthlake-fips.us-east-1.amazonaws.com	HTTPS
USA West (Oregon)	us-west-2	healthlake.us-west-2.amazonaws.com	HTTPS
		healthlake-fips.us-west-2.amazonaws.com	HTTPS
Asien-Pazifik (Mumbai)	ap-south-1	healthlake.ap-south-1.amazonaws.com	HTTPS
Asien-Pazifik (Sydney)	ap-southeast-2	healthlake.ap-southeast-2.amazonaws.com	HTTPS
Kanada (Zentral)	ca-central-1	healthlake.ca-central-1.amazonaws.com	HTTPS
Europa (Irland)	eu-west-1	healthlake.eu-west-1.amazonaws.com	HTTPS

Name der Region	Region	Endpunkt	Protocol (Protokoll)	
Europa (London)	eu-west-2	healthlake.eu-west-2.amazonaws.com	HTTPS	

Servicekontingente

Dienstkontingente sind als der Höchstwert für Ressourcen, Aktionen und Elemente in Ihrem AWS Konto definiert.

> **Note**
>
> Für anpassbare Kontingente können Sie über die Service Quotas-Konsole eine Erhöhung des Kontingents beantragen. Weitere Informationen finden Sie unter Beantragen einer Kontingenterhöhung im Service-Quotas-Benutzerhandbuch.
> Die Sync-Write-API-Rate steigt proportional mit der Größe der Nutzlast, wobei jedes 1-KB-Inkrement zusätzliche Kapazität verbraucht (z. B. verbraucht eine 4-KB-Payload das Vierfache der Schreibkapazität). Wenn Sie den optionalen `x-amz-fhir-history-consistency-level` Header auf einstellen, `strong` verdoppelt sich der Schreibkapazitätsverbrauch pro Ressource.
> Ressourcen in Paketen unterliegen den read/write Standardgrenzwerten, die auf einer Nutzlastgröße von 1 KB basieren. Eine Transaktion vom Typ Bundle verbraucht doppelt so viel Schreibkapazität wie eine Transaktion vom Typ Batch, was bedeutet, dass Batch-Bundles doppelt so viele Ressourcen pro Sekunde verarbeiten können wie Transaktionspakete.

In der folgenden Tabelle sind die Standardkontingente für aufgeführt. AWS HealthLake

Name	Standard	Anpas	Description
Anzahl der aktiven Abonnements pro Konto	Jede unterstützte Region: 100	Yes (Ja)	Die maximale Anzahl aktiver Abonnemen tressourcen pro Konto.

Name	Standard	Anpas	Description
Anzahl der aktiven Abonnemen tressourcen pro Datenspeicher	Jede unterstützte Region: 50	Ja	Die maximale Anzahl aktiver Abonnemen tressourcen pro Datenspeicher.
Anzahl der aktiven pro Konto Subscript ionTopic	Jede unterstützte Region: 100	Yes (Ja)	Die maximale Anzahl aktiver SubscriptionTopic Ressourcen pro Konto.
Anzahl der aktiven SubscriptionTopic Ressourcen pro Datenspeicher	Jede unterstützte Region: 50	Ja	Die maximale Anzahl aktiver Subscript ionTopic Ressourcen pro Datenspeicher.
Anzahl der Zeichen in einer medizinis chen Notiz	Jede unterstützte Region: 10 000	Nein	Die maximale Anzahl von Zeichen in einer individue llen medizinischen Notiz innerhalb des DocumentR eference Ressourcentyps (POST/PUT Anfragen).
Anzahl gleichzeitiger Jobs StartFHIR ExportJob	Jede unterstützte Region: 1	Nein	Die maximale Anzahl gleichzeitiger Jobs StartFHIRExportJob .
Anzahl gleichzeitiger Jobs StartFHIR ImportJob	Jede unterstützte Region: 1	Nein	Die maximale Anzahl gleichzeitiger Jobs StartFHIRImportJob .
Anzahl der Datenspeicher pro Konto	Jede unterstützte Region: 10	Yes (Ja)	Die standardmäßige maximale Anzahl aktiver Datenspeicher pro Konto.
Anzahl der Dateien in einem StartFHIR ImportJob	Jede unterstützte Region: 1 000 000	Ja	Die maximale Anzahl von Dateien in StartFHIR ImportJob a.

Name	Standard	Anpas	Description
Anzahl der Ressourcen pro Paket	Jede unterstützte Region: 500	Nein	Die maximale Anzahl von Ressourcen, die in einer Bundle-Anfrage zulässig sind.
Rate der CancelFHIRExportJob Anfragen, die DELETE pro Konto verwenden	Jede unterstützte Region: 1	Nein	Die maximale Anzahl von CancelFHIRExportJob Anfragen mit DELETE, die Sie pro Sekunde und Konto stellen können.
Rate der CreateFhirDataStore-Anfragen pro Konto	Jede unterstützte Region: 1	Nein	Die maximale Anzahl von CreateFhirDataStore-Anfragen, die Sie pro Minute und Konto stellen können.
Rate der DeleteFhirDataStore-Anfragen pro Konto	Jede unterstützte Region: 1	Nein	Die maximale Anzahl von DeleteFhirDataStore-Anfragen, die Sie pro Minute und Konto stellen können.
DescribeFHIRBulkDeleteJob Rate der Anfragen pro Konto	Jede unterstützte Region: 10	Yes (Ja)	Die maximale Anzahl von DescribeFHIRBulkDeleteJob Anfragen, die Sie pro Sekunde und Konto stellen können.
Rate der DescribeFHIRBulkDeleteJob Anfragen pro Datenspeicher	Jede unterstützte Region: 10	Yes (Ja)	Die maximale Anzahl von DescribeFHIRBulkDeleteJob Anfragen, die Sie pro Sekunde pro Datenspeicher stellen können.

Name	Standard	Anpas	Description
Rate der DescribeFhirDataStore-Anfragen pro Konto	Jede unterstützte Region: 10	Nein	Die maximale Anzahl von DescribeFhirDataStore-Anfragen, die Sie pro Sekunde und Konto stellen können.
Rate der Anfragen pro Konto DescribeFHIRExportJob	Jede unterstützte Region: 10	Nein	Die maximale Anzahl von DescribeFHIRExportJob Anfragen, die Sie pro Sekunde und Konto stellen können.
Rate der DescribeFHIRExportJob Anfragen mit GET pro Konto	Jede unterstützte Region: 10	Nein	Die maximale Anzahl von DescribeFHIRExportJob Anfragen mit GET, die Sie pro Sekunde und Konto stellen können.
Rate der DescribeFHIRImportJob Anfragen pro Konto	Jede unterstützte Region: 10	Nein	Die maximale Anzahl von DescribeFHIRImportJob Anfragen, die Sie pro Sekunde und Konto stellen können.
Rate der Discovery-Anfragen pro Konto	Jede unterstützte Region: 10	Nein	Die maximale Anzahl von Discovery-Anfragen, die Sie pro Minute und Konto stellen können.
Rate der GET-Anfragen pro Konto	Jede unterstützte Region: 6 000	Ja	Die maximale Anzahl von GET-Anfragen, die Sie pro Sekunde und Konto stellen können.

Name	Standard	Anpas	Description
Rate der GET-Anfragen pro Datenspeicher	Jede unterstützte Region: 3 000	Ja	Die maximale Anzahl von GET-Anfragen, die Sie pro Sekunde und Datenspeicher stellen können. Datenspeicher, die vor 8/21 /2023 erstellt wurden, sind auf 100 Anfragen pro Sekunde begrenzt.
Rate der GetCapabilities Anfragen pro Konto	Jede unterstützte Region: 10	Nein	Die maximale Anzahl von GetCapabilities Anfragen, die Sie pro Sekunde und Konto stellen können.
Rate der GetExportedFile Anfragen pro Datenspeicher	Jede unterstützte Region: 10	Nein	Die maximale Anzahl von GetExportedFile Anfragen, die Sie pro Sekunde pro Datenspeicher stellen können.
Rate der ListFhirDataStores-Anfragen pro Konto	Jede unterstützte Region: 10	Nein	Die maximale Anzahl von ListFhirDataStores -Anfragen, die Sie pro Sekunde und Konto stellen können.
Rate der Anfragen pro Konto ListFHIRExportJobs	Jede unterstützte Region: 10	Nein	Die maximale Anzahl von ListFHIRExportJobs Anfragen, die Sie pro Sekunde und Konto stellen können.

Name	Standard	Anpas	Description
Rate der ListFHIRImportJobs Anfragen pro Konto	Jede unterstützte Region: 10	Nein	Die maximale Anzahl von ListFHIRImportJobs Anfragen, die Sie pro Sekunde und Konto stellen können.
Rate der ListTagsforResource Anfragen pro Konto	Jede unterstützte Region: 10	Nein	Die maximale Anzahl von ListTagsforResource Anfragen, die Sie pro Sekunde und Konto stellen können.
Rate der SearchEverything Anfragen pro Konto	Jede unterstützte Region: 10	Yes (Ja)	Die maximale Anzahl von SearchEverything Anfragen, die Sie pro Sekunde und Konto stellen können.
Rate der SearchEverything Anfragen pro Datenspeicher	Jede unterstützte Region: 10	Yes (Ja)	Die maximale Anzahl von SearchEverything Anfragen, die Sie pro Sekunde pro Datenspeicher stellen können.
Rate der StartFHIRBulkDeleteJob Anfragen pro Konto	Jede unterstützte Region: 10	Yes (Ja)	Die maximale Anzahl von StartFHIRBulkDeleteJob Anfragen, die Sie pro Sekunde und Konto stellen können.
Rate der StartFHIRBulkDeleteJob Anfragen pro Datenspeicher	Jede unterstützte Region: 10	Yes (Ja)	Die maximale Anzahl von StartFHIRBulkDeleteJob Anfragen, die Sie pro Sekunde pro Datenspeicher stellen können.

Name	Standard	Anpas	Description
Rate der StartFHIRBulkMembe rMatchJob Anfragen pro Konto	Jede unterstützte Region: 10	Yes (Ja)	Die maximale Anzahl von StartFHIRBulkMembe rMatchJob Anfragen, die Sie pro Sekunde und Konto stellen können.
Rate der StartFHIRBulkMembe rMatchJob Anfragen pro Datenspeicher	Jede unterstützte Region: 10	Yes (Ja)	Die maximale Anzahl von StartFHIRBulkMembe rMatchJob Anfragen, die Sie pro Sekunde pro Datenspeicher stellen können.
Rate der StartFHIRExportJob Anfragen pro Konto	Jede unterstützte Region: 1	Nein	Die maximale Anzahl von StartFHIRExportJob Anfragen, die Sie pro Sekunde und Konto stellen können.
Rate der StartFHIRExportJob Anfragen mit GET pro Konto	Jede unterstützte Region: 1	Nein	Die maximale Anzahl von StartFHIRExportJob Anfragen mit GET, die Sie pro Sekunde und Konto stellen können.
Rate der StartFHIRExportJob Anfragen mit POST pro Konto	Jede unterstützte Region: 1	Nein	Die maximale Anzahl von StartFHIRExportJob Anfragen mit POST, die Sie pro Sekunde und Konto stellen können.
Rate der StartFHIRImportJob Anfragen pro Konto	Jede unterstützte Region: 25	Nein	Die maximale Anzahl von StartFHIRImportJob Anfragen, die Sie pro Sekunde und Konto stellen können.

Name	Standard	Anpas	Description
Rate der TagResource Anfragen pro Konto	Jede unterstützte Region: 10	Nein	Die maximale Anzahl von TagResource Anfragen, die Sie pro Sekunde und Konto stellen können.
Rate der UntagResource Anfragen pro Konto	Jede unterstützte Region: 10	Nein	Die maximale Anzahl von UntagResource Anfragen, die Sie pro Sekunde und Konto stellen können.
Rate der ValidateResource Anfragen pro Konto	Jede unterstützte Region: 2.000	Ja	Die maximale Anzahl von ValidateResource Anfragen, die Sie pro Sekunde und Konto stellen können. Datenspeicher, die vor 8/21 /2023 erstellt wurden, sind auf 300 Anfragen pro Sekunde begrenzt.
Rate der ValidateResource Anfragen pro Datenspeicher	Jede unterstützte Region: 1 000	Ja	Die maximale Anzahl von ValidateResource Anfragen, die Sie pro Sekunde pro Datenspeicher stellen können. Datenspeicher, die vor 8/21 /2023 erstellt wurden, sind auf 300 Anfragen pro Sekunde begrenzt.

Name	Standard	Anpas	Description
Rate der WRITE-Anfragen pro Konto	Jede unterstützte Region: 6 000	Ja	Die maximale Anzahl von CREATE|UPDATE|DELETE-Anfragen, die Sie pro Sekunde und Konto stellen können.
Rate der WRITE-Anfragen pro Datenspeicher	Jede unterstützte Region: 3 000	Ja	Die maximale Anzahl von CREATE|UPDATE|DELETE-Anfragen, die Sie pro Sekunde und Datenspeicher stellen können. Datenspeicher, die vor 8/21 /2023 erstellt wurden, sind auf 300 Anfragen pro Sekunde begrenzt.
Rate der Suchanfragen mit GET pro Konto	Jede unterstützte Region: 200	Ja	Die maximale Anzahl von Suchanfragen mit GET, die Sie pro Sekunde und Konto stellen können.
Rate der Suchanfragen mit GET pro Datenspeicher	Jede unterstützte Region: 100	Yes (Ja)	Die maximale Anzahl von Suchanfragen mit GET, die Sie pro Sekunde und Datenspeicher stellen können.
Rate der Suchanfragen mit POST pro Konto	Jede unterstützte Region: 200	Ja	Die maximale Anzahl von Suchanfragen mit POST, die Sie pro Sekunde und Konto stellen können.

Name	Standard	Anpas	Description
Rate der Suchanfragen mit POST pro Datenspeicher	Jede unterstützte Region: 100	Yes (Ja)	Die maximale Anzahl von Suchanfragen mit POST, die Sie pro Sekunde und Datenspeicher stellen können.
Größe der einzelnen importierten Datei	Jede unterstützte Region: 50 Gigabyte	Nein	Die maximale Größe (in GB) einer einzelnen Datei, die in einer enthalten ist StartFHIR ImportJob.
Gesamtzahl der asynchronen Bundle-Transaktionen in der Warteschlange pro Datenspeicher	Jede unterstützte Region: 500	Ja	Die maximale Anzahl von asynchronen Bundle-Transaktionen in der Warteschlange pro Datenspeicher zu einem bestimmten Zeitpunkt.
Gesamtzahl der in der Warteschlange befindlichen Massenexportaufträge pro Datenspeicher	Jede unterstützte Region: 25	Ja	Die maximale Anzahl von Massenexportaufträgen in der Warteschlange pro Datenspeicher zu einem bestimmten Zeitpunkt.
Gesamtzahl der in der Warteschlange befindlichen Massenimportaufträge pro Datenspeicher	Jede unterstützte Region: 25	Ja	Die maximale Anzahl von Massenimportaufträgen in der Warteschlange pro Datenspeicher zu einem bestimmten Zeitpunkt.
Gesamtgröße des Importauftrags	Jede unterstützte Region: 5.000 Gigabyte	Ja	Die maximale Größe (in GB) aller im Importauftrag enthaltenen Dateien.

Synthea vorinstallierte Datentypen für HealthLake

HealthLake unterstützt nur SYNTHEA als vorinstallierten Datentyp. Synthea ist ein synthetischer Patientengenerator, der die Krankengeschichte modelliert. `Patient` Es wird in einem Open-Source-Git-Repository gehostet, das es ermöglicht, eine FHIR R4-konforme Ressource HealthLake zu generieren, `Bundle` sodass Benutzer Modelle testen können, ohne tatsächliche Patientendaten zu verwenden.

Die folgenden Ressourcentypen sind in vorinstallierten Datenspeichern verfügbar. HealthLake Weitere Informationen zum Vorladen von HealthLake Datenspeichern mit Synthea-Daten finden Sie unter. Einen HealthLake Datenspeicher erstellen

> **Note**
>
> Eine vollständige Liste der von HealthLake -unterstützten FHIR R4-Ressourcen finden Sie unter. FHIR R4 unterstützte Ressourcentypen für HealthLake

Synthea FHIR-Ressourcentypen werden unterstützt von HealthLake

AllergyIntolerance	Speicherort
CarePlan	MedicationAdministration
CareTeam	MedicationRequest
Antrag	Beobachtung
Bedingung	Organisation
Gerät	Patient
DiagnosticReport	Praktiker
Begegnung	PractitionerRole
ExplanationofBenefit	Verfahren
ImagingStudy	Herkunft
Immunisierung	

AWS HealthLake Beispielprojekte

Um Ihre Analyse weiter voranzutreiben, können Sie es HealthLake zusammen mit anderen AWS Diensten verwenden, wie in den folgenden Beispielen für Blogbeiträge gezeigt.

HealthLake integrierte Analytik

- Anwendungen zur Bevölkerungsgesundheit mit AWS HealthLake — Teil 1: Analyse und Überwachung mit Amazon Quick.
- Erstellung prädiktiver Krankheitsmodelle mithilfe von Amazon SageMaker AI mit AWS HealthLake normalisierten Daten.
- Erstellen Sie mithilfe von AWS KI-Diensten eine kognitive Suche und ein Diagramm mit Gesundheitswissen.

HealthLake Überwachung von Ereignissen

- EventBridge Amazon-Integration mit AWS HealthLake.

Problembehebung AWS HealthLake

Die folgenden Themen enthalten Hinweise zur Problembehandlung bei Fehlern und Problemen, die bei der Verwendung der HealthLake Konsole AWS CLI AWS SDKs, oder auftreten können. Wenn Sie ein Problem finden, das in diesem Abschnitt nicht aufgeführt ist, verwenden Sie die Schaltfläche Feedback geben in der rechten Seitenleiste dieser Seite, um es zu melden.

Themen

- Aktionen im Datenspeicher
- Aktionen importieren
- FHIR APIs
- NLP-Integrationen
- SQL-Integrationen

Aktionen im Datenspeicher

Problem: Wenn ich versuche, einen HealthLake Datenspeicher zu erstellen, erhalte ich die folgende Fehlermeldung:

```
AccessDeniedException: Insufficient Lake Formation permission(s): Required Database on
 Catalog
```

Am 14. November 2022 HealthLake wurden die erforderlichen IAM-Berechtigungen zum Erstellen eines neuen Datenspeichers aktualisiert. Weitere Informationen finden Sie unter Konfigurieren Sie einen zu verwendenden IAM-Benutzer oder eine IAM-Rolle HealthLake (IAM-Administrator).

Problem: Beim Erstellen eines HealthLake Datenspeichers mithilfe von gibt der AWS SDKs Erstellungsstatus des Datenspeichers einen Ausnahmestatus oder einen unbekannten Status zurück.

Aktualisieren Sie Ihr AWS SDK auf die neueste Version, wenn Ihre Aufrufe `DescribeFHIRDatastore` oder `ListFHIRDatastores` API-Aufrufe eine Ausnahme oder einen unbekannten Datenspeicherstatus zurückgeben.

Aktionen importieren

Problem: Kann ich es trotzdem verwenden, HealthLake wenn meine Daten nicht im FHIR R4-Format vorliegen?

Nur FHIR R4-formatierte Daten können in einen Datenspeicher importiert werden. HealthLake Eine Liste von Partnern, die bei der Umwandlung vorhandener Gesundheitsdaten in das FHIR R4-Format helfen können, finden Sie unter Partner.AWS HealthLake

Problem: Warum ist mein FHIR-Importjob fehlgeschlagen?

Bei einem erfolgreichen Importauftrag wird ein Ordner mit Ergebnissen (Ausgabeprotokoll) im `.ndjson` Format generiert. Einzelne Datensätze können jedoch möglicherweise nicht importiert werden. In diesem Fall wird ein zweiter `FAILURE` Ordner mit einem Manifest der Datensätze generiert, die nicht importiert werden konnten. Weitere Informationen finden Sie unter Import von FHIR-Daten mit AWS HealthLake.

Um zu analysieren, warum ein Importauftrag fehlgeschlagen ist, verwenden Sie die `DescribeFHIRImportJob` API zur Analyse von JobProperties. Folgendes wird empfohlen:

- Wenn der Status lautet `FAILED` und eine Meldung vorhanden ist, hängen die Fehler damit zusammen, dass Jobparameter wie die Größe der Eingabedaten oder die Anzahl der Eingabedateien die HealthLake Kontingente überschreiten.
- Lautet der Status des `COMPLETED_WITH_ERRORS` Importauftrags, finden Sie in der Manifestdatei Informationen darüber`manifest.json`, welche Dateien nicht erfolgreich importiert wurden.

- Wenn der Status des Importauftrags lautet `FAILED` und keine Meldung vorhanden ist, wechseln Sie zum Ausgabeort des Jobs, um auf die Manifestdatei zuzugreifen`manifest.json`.

Für jede Eingabedatei gibt es eine Fehlerausgabedatei mit dem Namen der Eingabedatei für jede Ressource, die nicht importiert werden kann. Die Antworten enthalten die Zeilennummer (lineID), die der Position der Eingabedaten entspricht, das FHIR-Antwortobjekt (UpdateResourceResponse) und den Statuscode (statusCode) der Antwort.

Eine Beispielausgabedatei könnte der folgenden ähneln:

```
{"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}
```


Das obige Beispiel zeigt, dass in den Zeilen 3, 4, 7, 9, 15 in den entsprechenden Eingabezeilen der Eingabedatei Fehler aufgetreten sind. Für jede dieser Zeilen lauten die Erklärungen wie folgt:

- In Zeile 3 wird in der Antwort erklärt, dass die `resourceType` Angaben in Zeile 3 der Eingabedatei nicht gültig sind.
- In Zeile 5 wird in der Antwort erklärt, dass in Zeile 5 der Eingabedatei ein FHIR-Validierungsfehler vorliegt.
- In Zeile 7 wird in der Antwort erklärt, dass ein Validierungsproblem bei der Eingabe `resourceId` vorliegt.

- In Zeile 9 wird in der Antwort erklärt, dass die Eingabedatei eine gültige Ressourcen-ID enthalten muss.
- In Zeile 15 lautet die Antwort der Eingabedatei, dass die Datei kein gültiges JSON-Format hat.

FHIR APIs

Problem: Wie implementiere ich die Autorisierung für die RESTful APIs FHIR?

Ermitteln Sie, welche verwendet Strategie zur Autorisierung von Datenspeichern werden soll.

Um eine SigV4-Autorisierung mit dem zu erstellen AWS SDK für Python (Boto3), erstellen Sie ein Skript, das dem folgenden Beispiel ähnelt.

```
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())
```

Problem: Warum erhalte ich *`AccessDenied`* Fehler, wenn ich FHIR RESTful APIs für einen Datenspeicher verwende, der mit einem vom Kunden verwalteten KMS-Schlüssel verschlüsselt ist?

Damit ein Benutzer oder eine Rolle auf einen Datenspeicher zugreifen kann, sind Berechtigungen sowohl für vom Kunden verwaltete Schlüssel als auch für IAM-Richtlinien erforderlich. Ein Benutzer muss über die erforderlichen IAM-Berechtigungen verfügen, um einen vom Kunden verwalteten Schlüssel verwenden zu können. Wenn ein Benutzer eine Erteilung widerruft oder zurückzieht, mit der die HealthLake Erlaubnis zur Verwendung des vom Kunden verwalteten KMS-Schlüssels erteilt wurde, HealthLake wird ein `AccessDenied` Fehler zurückgegeben.

HealthLake muss über die erforderlichen Berechtigungen verfügen, um auf Kundendaten zuzugreifen, neue in einen Datenspeicher importierte FHIR-Ressourcen zu verschlüsseln und die FHIR-Ressourcen zu entschlüsseln, wenn sie angefordert werden. Weitere Informationen finden Sie unter Problembehandlung bei Berechtigungen. AWS KMS

Problem: Ein *`POST`* FHIR-API-Vorgang zur HealthLake Verwendung eines 10-MB-Dokuments gibt den *`413 Request Entity Too Large`* Fehler zurück.

AWS HealthLake hat ein synchrones Create and Update API-Limit von 5 MB, um erhöhte Latenzen und Timeouts zu vermeiden. Sie können große Dokumente (bis zu 164 MB) mithilfe des `Binary` Ressourcentyps mithilfe der Bulk Import API aufnehmen.

NLP-Integrationen

Problem: Wie aktiviere ich die integrierte HealthLake Funktion zur Verarbeitung natürlicher Sprache?

Ab dem 14. November 2022 hat sich das Standardverhalten von HealthLake Datenspeichern geändert.

Aktuelle Datenspeicher: Alle aktuellen HealthLake Datenspeicher werden die Verarbeitung natürlicher Sprache (NLP) für `DocumentReference` Base64-kodierte Ressourcen einstellen. Das bedeutet, dass neue `DocumentReference` Ressourcen nicht mit NLP analysiert werden und dass keine neuen Ressourcen auf der Grundlage von Text im Ressourcentyp generiert werden. `DocumentReference` Für bestehende `DocumentReference` Ressourcen bleiben die über NLP generierten Daten und Ressourcen erhalten, sie werden jedoch nach dem 20. Februar 2023 nicht aktualisiert.

Neue Datenspeicher: HealthLake Datenspeicher, die nach dem 20. Februar 2023 erstellt wurden, führen auf `DocumentReference` Base64-codierten Ressourcen keine Verarbeitung natürlicher Sprache (Natural Language Processing, NLP) durch.

Um die HealthLake NLP-Integration zu aktivieren, erstellen Sie einen Support-Fall mit. AWS Support Center Console Um Ihren Fall zu erstellen, melden Sie sich bei Ihrem AWS-Konto an und wählen Sie

dann Fall erstellen. Weitere Informationen zur Erstellung eines Falls und zur Fallverwaltung finden Sie im Support Benutzerhandbuch unter Erstellen von Supportanfragen und Fallmanagement.

Problem: >Wie finde ich *`DocumentReference`* Ressourcen, die mit integriertem NLP nicht verarbeitet werden konnten?

Wenn eine `DocumentReference` Ressource nicht gültig ist, wird eine Erweiterung HealthLake bereitgestellt, die auf einen Validierungsfehler hinweist, anstatt sie in der integrierten medizinischen NLP-Ausgabe bereitzustellen. Um **`DocumentReference`** Ressourcen zu finden, die bei der NLP-Verarbeitung zu einem Validierungsfehler geführt haben, können Sie die **`search`** FHIR-Funktion mit dem Suchschlüssel cm-decoration-statusund dem Suchwert VALIDATION_ERROR verwenden HealthLake. Diese Suche listet alle `DocumentReference` Ressourcen auf, die zu Validierungsfehlern geführt haben, zusammen mit einer Fehlermeldung, die die Art des Fehlers beschreibt. Die Struktur des Erweiterungsfeldes in den `DocumentReference` Ressourcen mit Validierungsfehlern ähnelt dem folgenden Beispiel.

```
"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` kann auch auftreten, wenn bei der NLP-Dekoration mehr als 10.000 verschachtelte Objekte erstellt werden. In diesem Fall muss das Dokument vor der Verarbeitung in kleinere Dokumente aufgeteilt werden.

SQL-Integrationen

Problem: Warum erhalte ich eine Lake Formation, *`permissions error: lakeformation:PutDataLakeSettings`* wenn ich einen neuen Data Lake-Administrator hinzufüge?

Wenn Ihr IAM-Benutzer oder Ihre IAM-Rolle die `AWSLakeFormationDataAdmin` AWS verwaltete Richtlinie enthält, können Sie keine neuen Data Lake-Administratoren hinzufügen. Sie erhalten eine Fehlermeldung, die Folgendes enthält:

```
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
```

Die AWS verwaltete Richtlinie `AdministratorAccess` ist erforderlich, um einen IAM-Benutzer oder eine IAM-Rolle als AWS Lake Formation Data Lake-Administrator hinzuzufügen. Wenn Ihr IAM-Benutzer oder Ihre IAM-Rolle ebenfalls enthält, schlägt `AWSLakeFormationDataAdmin` die Aktion fehl. Die `AWSLakeFormationDataAdmin` AWS verwaltete Richtlinie enthält eine ausdrückliche Ablehnung des AWS Lake Formation Formation-API-Vorgangs,`PutDataLakeSetting`. Selbst Administratoren mit uneingeschränktem AWS Zugriff auf die `AdministratorAccess` verwaltete Richtlinie können durch die `AWSLakeFormationDataAdmin` Richtlinie eingeschränkt werden.

Problem: Wie migriere ich einen vorhandenen HealthLake Datenspeicher, um die Amazon Athena SQL-Integration zu verwenden?

HealthLake Datenspeicher, die vor dem 14. November 2022 erstellt wurden, funktionieren, können in Athena jedoch nicht mit SQL abgefragt werden. Um einen bereits vorhandenen Datenspeicher mit Athena abzufragen, müssen Sie ihn zunächst in einen neuen Datenspeicher migrieren.

Um Ihre HealthLake Daten in einen neuen Datenspeicher zu migrieren

1. Erstellen Sie einen neuen Datenspeicher.
2. Exportieren Sie die Daten aus dem bereits vorhandenen in einen Amazon S3 S3-Bucket.
3. Importieren Sie die Daten aus dem Amazon S3 S3-Bucket in den neuen Datenspeicher.

> **Note**
>
> Das Exportieren von Daten in einen Amazon S3 S3-Bucket ist mit einem Aufpreis verbunden. Die zusätzliche Gebühr hängt von der Größe der Daten ab, die Sie exportieren.

Problem: Beim Erstellen eines neuen HealthLake Datenspeichers für die SQL-Integration ändert sich der Status des Datenspeichers nicht von*`Creating`*.

Wenn Sie versuchen, einen neuen HealthLake Datenspeicher zu erstellen, und Ihr Datenspeicherstatus sich nicht von Creating ändert, müssen Sie Athena aktualisieren, um den AWS Glue Data Catalog verwenden zu können. Weitere Informationen finden Sie unter Upgrade auf den AWS Glue-Datenkatalog step-by-step im Amazon Athena Athena-Benutzerhandbuch.

Nach dem erfolgreichen Upgrade von können Sie einen HealthLake Datenspeicher erstellen. AWS Glue Data Catalog

Um einen alten HealthLake Datenspeicher zu entfernen, erstellen Sie einen Support-Fall mit AWS Support Center Console. Um Ihren Fall zu erstellen, melden Sie sich bei Ihrem AWS-Konto an und wählen Sie dann Kundenfall erstellen aus. Weitere Informationen finden Sie im Support Benutzerhandbuch unter Erstellen von Supportanfragen und Fallmanagement.

Problem: Die Athena-Konsole funktioniert nach dem Import von Daten in einen neuen HealthLake Datenspeicher nicht

Nachdem Sie Daten in einen neuen HealthLake Datenspeicher importiert haben, stehen die Daten möglicherweise nicht sofort zur Verfügung. Dies dient dazu, genügend Zeit für die Aufnahme der Daten in Apache Iceberg-Tabellen zu haben. Versuchen Sie es zu einem späteren Zeitpunkt erneut.

Problem: Wie verbinde ich Suchergebnisse in Athena mit anderen AWS Diensten?

Wenn Sie Ihre Suchergebnisse von Athena mit anderen AWS Diensten teilen, können Probleme auftreten, wenn Sie sie `json_extract[1]` als Teil einer SQL-Suchabfrage verwenden. Um dieses Problem zu beheben, müssen Sie auf `CATVAR` aktualisieren.

Dieses Problem tritt möglicherweise auf, wenn Sie versuchen, Speicherergebnisse, eine Tabelle (statisch) oder eine Ansicht (dynamisch) zu erstellen.

Verwendung HealthLake mit einem AWS SDK

AWS Software Development Kits (SDKs) sind für viele beliebte Programmiersprachen verfügbar. Jedes SDK bietet eine API, Codebeispiele und Dokumentation, die es Entwicklern erleichtern, Anwendungen in ihrer bevorzugten Sprache zu erstellen.

SDK-Dokumentation	Codebeispiele
AWS SDK für C++	AWS SDK für C++ Codebeispiele
AWS CLI	AWS CLI Code-Beispiele
AWS SDK für Go	AWS SDK für Go Code-Beispiele
AWS SDK für Java	AWS SDK für Java Code-Beispiele
AWS SDK für JavaScript	AWS SDK für JavaScript Code-Beispiele
AWS SDK für Kotlin	AWS SDK für Kotlin Code-Beispiele
AWS SDK für .NET	AWS SDK für .NET Code-Beispiele
AWS SDK für PHP	AWS SDK für PHP Code-Beispiele
AWS -Tools für PowerShell	AWS -Tools für PowerShell Code-Beispiele
AWS SDK für Python (Boto3)	AWS SDK für Python (Boto3) Code-Beispiele
AWS SDK für Ruby	AWS SDK für Ruby Code-Beispiele
AWS SDK für Rust	AWS SDK für Rust Code-Beispiele
AWS SDK für SAP ABAP	AWS SDK für SAP ABAP Code-Beispiele
AWS SDK für Swift	AWS SDK für Swift Code-Beispiele

Beispiel für die Verfügbarkeit

Sie können nicht finden, was Sie brauchen? Fordern Sie ein Codebeispiel an, indem Sie unten den Link Provide feedback (Feedback geben) auswählen.

AWS HealthLake Veröffentlichungen

Die folgende Tabelle zeigt, für AWS HealthLake wann Funktionen und Updates veröffentlicht wurden. Weitere Informationen zu einer Version finden Sie im verlinkten Thema.

Änderung	Beschreibung	Datum
Sicherheit und Tagfilterung für $export und $ davinci-data-export	Die `$davinci-data-export` Operationen `$export` und unterstützen jetzt Parameter zum Filtern exportierter Ressourcen nach `_security` und nach `meta.tag` Codierung swerten `meta.security` und `_tag` fragen diese ab. Dies ermöglicht den mehrinstanzenfähigen Export und eine detaillierte Zugriffskontrolle unter Verwendung des `system|code` Standard-FHIR-Formats.	30. April 2026
$ Betrieb bulk-member-match	AWS HealthLake unterstützt jetzt den `$bulk-member-match` Vorgang für die asynchrone Verarbeitung mehrerer Mitgliederabgleichsanfragen. Dieser Vorgang ermöglicht es Organisationen im Gesundheitswesen, die eindeutigen Identifikatoren von Hunderten von Mitgliedern in verschiedenen Gesundheitssystemen effizient abzugleichen, indem demografische	1. April 2026

Informationen und Informationen zum Versicherungsschutz in einer einzigen Sammelanfrage verwendet werden.

- Verarbeitet bis zu 500 Mitglieder pro Anfrage mit bis zu 5 gleichzeitigen Vorgängen pro Datenspeicher
- Die Ergebnisse sind in MatchedMembers Gruppen NonMatchedMembers, und ConsentConstrained Members unterteilt
- Lässt sich in Workflows `$davinci-data-export` für end-to-end Massendaten integrieren

Weitere Informationen finden Sie unter the section called “$bulk-member-match”.

Asynchrone Bundle-Transaktionen	AWS HealthLake unterstützt jetzt den asynchronen `Bundle` Typ`transaction` , sodass Sie Transaktionen mit bis zu 500 Ressourcen einreichen können. HealthLake stellt die Transaktion zur Verarbeitung in eine Warteschlange und gibt sofort eine Abfrage-URL zurück, um den Status zu überprüfen und Ergebnisse abzurufen. Weitere Informationen finden Sie unter Asynchrone Bundle-Transaktionen.	24. März 2026
Patch-Unterstützung im Bundle-Betrieb	HealthLake unterstützt jetzt Bundle Patch FHIRPath und aktiviert sowohl JSON Patch für Transaction- und Batch-Bundles als auch FHIRPath für die Patch-API. Diese Funktion ermöglicht es Kunden, Ressourcen direkt im Bundle-Betrieb zu patchen, ohne dass die Ressourcen vollständig ausgetauscht werden müssen, wodurch die Größe der Nutzlast reduziert, Integrationsworkflows vereinfacht und eine schnellere Datenaufnahme ermöglicht wird.	20. März 2026

DaVinci PDex Exporttypen für $ davinci-data-export	Der `$davinci-data-export` Vorgang unterstützt jetzt PDex Exporttypen für Provider Access Payer-to-Payer, und Member Access APIs. • Profilbasierte Inklusionslogik für Ressourcen ExplanationOfBenefit • Transformation von Finanzdaten mit dem Parameter `_includeE0B2xWoFinancial` • Zeitlicher Filter für klinische Daten und Schadensdaten über einen Zeitraum von 5 Jahren	20. März 2026
Parameter _until in $export & $ davinci-data-export	Temporaler Filterparameter für Exportvorgänge	26. Februar 2026
_include Suchparameter	HealthLake unterstützt jetzt include: * und include:iterate	26. Februar 2026
Endpunkte für die CMS-Interoperabilität	Mit dieser Funktion können Sie die API-Nutzung nach CMS-Kategorien verfolgen und anschließend Nutzungsknnzahlen zu Compliance-Zwecken melden.	26. Februar 2026
Support für Bundle-Nachrichtentypen	Eingeschränkte Unterstützung für FHIR-Bundle-Ressourcen mit Nachrichtentyp	26. Februar 2026

Unterstützung für neue hinzugefügt IGs	AWS HealthLake hat die Unterstützung seiner FHIR Implementation Guides (IGs) für CMS 0057F erweitert: • Support für CARIN Blue Button 2.0.0 und 2.1.0 • Support für Da Vinci Payer Data Exchange 2.0.0 und 2.1.0 • Support für DaVinci Payer Data Exchange (PDex) US Drug Formulary 2.0.1 & 2.1.0 • Support für Da Vinci Clinical Data Exchange (CDex) 2.1.0 • Support für Da Vinci Prior Authorization Support (PAS) FHIR IG 2.1.0	26. Februar 2026
$submit Vorgang	Dieser `$submit` Vorgang ermöglicht es Ihnen, vorherige Autorisierungsanfragen elektronisch zur Genehmigung an die Zahler zu senden.	26. Februar 2026
Vorgang des $questionnaire-Paketes	Der `$questionnaire-package` Vorgang ruft ein umfangreiches Paket ab, das einen FHIR-Fragebogen und all seine Abhängigk eiten enthält, die für das Rendern und Verarbeiten des Fragebogens erforderlich sind.	26. Februar 2026

$ Vorgang anfragen	Mit `$inquire` diesem Vorgang können Sie den Status einer zuvor eingereic hten Autorisierungsanfrage überprüfen.	26. Februar 2026
Vorgang zum Entfernen von $member	Die Operation $member-r emove ermöglicht es Ihnen, Mitglieder aus einer FHIR-Mitg liedsattributionsliste (Gruppenr essource) in zu entfernen. AWS HealthLake	12. November 2025
Aktion „$member-match“	AWS HealthLake unterstützt jetzt die Operation $member-match für Patientenressource n, sodass Gesundheitsorganis ationen anhand demografi scher Daten und Versicher ungsdaten die eindeutige Kennung eines Mitglieds in verschiedenen Gesundhei tssystemen ermitteln können.	12. November 2025
Vorgang $member hinzufügen	Die Operation FHIR $member-add fügt einer Gruppenre ssource, insbesondere einer Mitgliedsattributionsliste, ein Mitglied (Patient) hinzu.	12. November 2025
$ davinci-data-export Betrieb	Die davinci-data-export Operation $ ist eine asynchron e FHIR-Operation, die den Export von Daten aus der Mitgliedsattributionsliste ermöglicht. AWS HealthLake	12. November 2025

$ confirm-attribution-list Betrieb	Zeigt einem Hersteller an, dass der Verbraucher keine weiteren Änderungen an der Zuordnungsliste vornehmen muss. Die Zuordnungsliste wird finalisiert, indem inaktive Mitglieder entfernt und der Status auf „endgültig“ geändert wird.	12. November 2025
Vorgang mit dem $attribution-status	Ruft den Zuordnungsstatus für ein bestimmtes Mitglied ab und gibt ein Paket zurück, das alle Zuordnungsressourcen enthält, die sich auf den Patienten beziehen.	12. November 2025
FHIR-Abonnements	HealthLake unterstützt FHIR-Abonnements, sodass Sie in Echtzeit Benachrichtigungen erhalten, wenn sich bestimmte Gesundheitsdaten ändern, und ereignisgesteuerte Workflows einrichten können.	30. Oktober 2025
Erweiterung der Region nach Montréal, Kanada	HealthLake ist in der Region Kanada (Montreal) verfügbar. Weitere Informationen finden Sie unter Dienstendpunkte.	17. Oktober 2025

Unterstützung für neue hinzugefügt IGs	AWS HealthLake hat die Unterstützung seiner FHIR Implementation Guides (IGs) für den folgenden kanadischen Markt erweitert: • CA Core+Canadian Baseline-FHIR-Profil, in dem die wichtigsten Datenelemente und Einschränkungen für die Interoperabilität zwischen den kanadischen Gesundheitssystemen definiert sind. • CA:EREC Pan-Canadian eReferral- eConsultStandardized FHIR-Spezifikation für elektronische Überweisungen und Konsultationen zwischen Gesundheitsdienstleistern in ganz Kanada. • Patient Summary Canadian Edition (PS-CA) Kanadische Adaption der International Patient Summary (IPS) für den Austausch wichtiger Patienteninformationen in allen Versorgungseinrichtungen. • Ontario Digital Health RepositoryOntario Drug-spezifisches FHIR-Profil für den standardisierten Austausch von Medikamenten- und Verschrei	17. Oktober 2025

	bungsdaten innerhalb des Gesundheitssystems der Provinz.	
Regionalspezifische IG-Unterstützung	HealthLake unterstützt jetzt regionsspezifisch IGs. Weitere Informationen finden Sie unter Profilvalidierungen.	8. Oktober 2025
Betrieb des Patches	HealthLake ermöglicht das Ändern bestimmter Elemente von FHIR-Ressourcen mithilfe von JSON-Patch-Vorgängen, ohne die gesamte Ressource zu aktualisieren.	18. August 2025
$ Vorgang validieren	HealthLake ermöglicht die Validierung von FHIR-Ressourcen anhand der Spezifikation und der Profile, ohne Speicheroperationen durchzuführen, und gibt detaillierte Überprüfungsergebnisse zurück.	18. August 2025
$ Löschvorgang	HealthLake beinhaltet Funktionen zum dauerhaften Löschen aller Ressourcen innerhalb einer Patientenabteilung aus dem Datenspeicher.	18. August 2025

$lookup-Vorgang	HealthLake bietet die Möglichkeit, detaillierte Informationen zu einem bestimmten Konzept in einem abzurufen, CodeSystem indem der Code und die System-ID bereitgestellt werden.	18. August 2025
$ Betrieb erweitern	HealthLake ermöglicht es nun, ValueSet Ressourcen zu erweitern, um die vollständige Liste der Codes abzurufen, die in den vom Kunden ValueSets aufgenommenen Codes enthalten sind.	18. August 2025
$ Vorgang löschen	HealthLake bietet jetzt die Möglichkeit, eine bestimmte Ressource und all ihre historischen Versionen dauerhaft aus dem Datenspeicher zu löschen.	18. August 2025
$ Vorgang des Dokuments	HealthLake unterstützt die Generierung vollständiger klinischer Dokumente, indem eine Kompositionsressource mit all ihren referenzierten Ressourcen in einem einzigen Dokumentenpaket gebündelt wird.	18. August 2025
18. August 2025 ----sep----:unten Suchmodifikator	HealthLake führt in einem Terminologiesystem die Suche nach URI-Werten ein, die hierarchisch unter dem angegebenen URI liegen.	8. August 2025

Bedingtes Löschen	HealthLake unterstützt jetzt FHIR Conditional Delete, sodass Organisationen im Gesundheitswesen eine vorhandene Ressource anhand von Suchkriterien und nicht anhand der logischen FHIR-ID löschen können. Weitere Informationen finden Sie unter Löschen von FHIR-Ressourcen auf der Grundlage von Bedingungen.	7. Juli 2025
Unterstützung für neue hinzugefügt IGs	AWS HealthLake hat die Unterstützung seiner FHIR Implementation Guides (IGs) für Folgendes erweitert: • US Core 7.0.0, das spezifiziert, wie FHIR zur Implementierung des USCDI 4.0-Standards verwendet wird • UK Core 2.0.1 Implementierungsleitfaden zur Bereitstellung britischer Leitlinien zur FHIR-Implementierung	7. Juli 2025
Erweiterung der Region nach Dublin, Irland	HealthLake ist in der Region EU (Dublin) verfügbar. Weitere Informationen finden Sie unter Dienstendpunkte.	9. Juni 2025

Transaktionen vom Typ Bundle	HealthLake unterstützt jetzt den FHIR-Bundle-Typ „Transaktion“, sodass Gesundheitsorganisationen mehrere Ressourcen als eine einzige atomare Operation einreichen können. Dies ermöglicht effizientere Workflows für Datenaustausch und Integration. Beispielsweise kann ein Gesundheitsdienstleister jetzt eine Patientenakte, eine Medikamentenliste und einen Termin in einer einzigen Transaktion aktualisieren, wodurch die Komplexität und potenzielle Fehler reduziert werden. Weitere Informationen finden Sie unter Bündelung von FHIR-Ressourcen.	28. April 2025

Unterstützung für neue hinzugefügt IGs	AWS HealthLake AWS HealthLake hat die Unterstützung seiner FHIR Implementation Guides (IGs) für Folgendes erweitert: • NCQA HEDIS® Implementierungsleitfaden (0.3.1): Unterstützt die Qualitätsmessung und Berichterstattung für den Healthcare Effectiveness Data and Information Set (HEDIS). • International Patient Summary (IPS) (2.0.0): Ermöglicht den Austausch wichtiger Gesundheitsinformationen, um die Kontinuität der Patientenversorgung zu gewährleisten. • Quality Measure (5.0.0): Unterstützt die Darstellung und den Austausch von Definitionen und Daten zu Qualitätsmaßstäben. • Genomics Reporting (3.0.0): Erleichtert den Austausch von Genomdaten und Berichten.	28. April 2025

Schlüssel zur Idempotenz	HealthLake unterstützt jetzt Idempotenzschlüssel für FHIR-POST-Operationen und bietet so einen robusten Mechanismus zur Sicherstellung der Datenintegrität bei der Ressourcenerstellung. Weitere Informationen finden Sie unter Idempotenz und Parallelität.	18. April 2025
Konsistenz der FHIR-Historie	HealthLake unterstützt jetzt über den neuen `x-amz-fhir-history-consistency-level` Header eine hohe Konsistenz für Datenspeicher, für die der Verlauf aktiviert ist. Wenn diese Option auf „stark“ gesetzt ist, enthalten die FHIR-Suchergebnisse alle indizierten Datensätze, unabhängig vom Aktualisierungsstatus. Weitere Informationen finden Sie unter FHIR-Suchkonsistenzstufen.	18. April 2025

Etag und 'if-match'	HealthLake bietet jetzt ETag-Unterstützung, sodass Clients den 'If-Match'-Header verwenden können, um idempotente Aktualisierungen sicherzustellen. Dies trägt zur Aufrechterhaltung der Datenintegrität bei, indem versehentliche Überschre ibungen bei gleichzeitigen Aktualisierungen verhindert werden. Dies ist besonders nützlich in Umgebungen mit hohem Volumen im Gesundhei tswesen, in denen mehrere Systeme versuchen könnten, denselben Datensatz gleichzei tig zu aktualisieren. Weitere Informationen finden Sie unter ETag in AWS HealthLake.	18. April 2025

PUTs Befriedigend in Paketen	HealthLake unterstützt jetzt bedingte Aktualisierungen für FHIR-Pakete, sodass Organisationen im Gesundheitswesen ihre Daten flexibler verwalten und aktualisieren können. Kunden können jetzt Kriterien angeben, um Ressourcen im Rahmen einer Bundle-Transaktion bedingt zu erstellen, zu aktualisieren oder zu löschen. Dies vereinfacht die Datensynchronisierung zwischen Systemen und reduziert den Bedarf an komplexer clientseitiger Logik. Weitere Informationen finden Sie unter Conditional PUTs in Bundles.	18. April 2025

SMART auf FHIR V2-Zielfernrohren	HealthLake unterstützt SMART auf FHIR V2-Bereichen zum Erstellen, Lesen, Aktualisieren, Löschen und Durchsuchen von FHIR-Ressourcen. Weitere Informationen finden Sie unter SMART auf FHIR-Ressourcenbereichen für. HealthLake • SMART-Bereiche auf FHIR V2 sind für alle HealthLake Datenspeicher verfügbar, die nach dem 22.01.2025 erstellt wurden. Wenn Ihr Datenspeicher vor diesem Datum erstellt wurde, können Sie ein Supportticket einreichen, um SMART auf FHIR V2-Bereichen zu aktivieren. Erstellen Sie einen Fall mit. AWS Support Center Console Um Ihren Fall zu erstellen, melden Sie sich bei Ihrem an AWS-Konto und wählen Sie Fall erstellen aus.	22. Januar 2025
FHIR US-Kernprofil, Version 6.1.0	HealthLake unterstützt Version 6.1.0 des FHIR US Core Profile. Weitere Informationen finden Sie unter FHIR-Profilvalidierungen für. HealthLake	22. Januar 2025

FHIR $export mit GET	HealthLake unterstützt FHIR mit`$export`. `GET` Weitere Informationen finden Sie unter HealthLake Daten mit `$export` FHIR exportieren.	22. Januar 2025
Überarbeitetes Entwicklerhandbuch mit getesteten Codebeispielen	HealthLake stellt ein überarbeitetes Entwicklerhandbuch mit getesteten Codebeispielen für native Aktionen und SDK-Aktionen vor. AWS CLI AWS Darüber hinaus sind jetzt Verfahren für alle unterstützten FHIR-API-Interaktionen verfügbar. Weitere Informationen finden Sie unter Codebeispiele und Verwaltung von FHIR-Ressourcen.	18. Dezember 2024

FHIR `history` und Interaktionen `vread`	HealthLake unterstützt die `history` FHIR-Interaktion zum Abrufen des Verlaufs einer bestimmten Ressource und die `vread` Interaktion zum Ausführen eines versionsspezifischen Lesevorgangs einer Ressource. Weitere Informationen finden Sie unter Lesen des FHIR-Ressourcenverlaufs. • Die FHIR-Ressource `history` ist standardmäßig für alle HealthLake Datenspeicher aktiviert, die nach dem 25.10.2024 erstellt wurden. Wenn Ihr Datenspeicher vor diesem Datum erstellt wurde, können Sie ein Supportticket einreichen, um die `history` FHIR-Interaktion zu aktivieren. Erstellen Sie einen Fall mit AWS Support Center Console. Um Ihren Fall zu erstellen, melden Sie sich bei Ihrem an AWS-Konto und wählen Sie Fall erstellen aus.	25. Oktober 2024

FHIR-Vorgang `Patient/$everything`	HealthLake unterstützt den `Patient/$everything` FHIR-Vorgang für die Suche nach einer `Patient` Ressource und all ihren zugehörigen Ressourcen. Mit diesem Vorgang können Sie auf die gesamte Patienten akte zugreifen oder `Patient` Daten in großen Mengen herunterladen. Weitere Informationen finden Sie unter Patientendaten abrufen mit `Patient/$everything` . • FHIR `Patient/$everything` ist standardmäßig für alle HealthLake Datenspeicher aktiviert, die nach dem 27.02.2024 erstellt wurden. Wenn Ihr Datenspeicher vor diesem Datum erstellt wurde, können Sie ein Supportticket einreiche n, um den `Patient/$everything` Vorgang zu aktivieren. Erstellen Sie einen Fall mit AWS Support Center Console. Um Ihren Fall zu erstellen, melden Sie sich bei Ihrem an AWS-Konto und wählen Sie Fall erstellen aus.	27. Februar 2024

FHIR-Ressource `VerificationResult`	HealthLake unterstützt den `VerificationResult` FHIR-Ressourcentyp zur Beschreibung von Validierungsanforderungen, Quellen, Status und Daten für ein oder mehrere Elemente. Weitere Informationen finden Sie unter FHIR R4-Ressourcentypen für HealthLake.	09. Dezember 2023

FHIR-Vorgang `$export`	HealthLake unterstützt den `$export` FHIR-Vorgang für den Massenexport von Gesundheitsdaten aus einem HealthLake Datenspeicher. Weitere Informationen finden Sie unter HealthLake Daten mit `$export` FHIR exportieren. • FHIR `$export` ist standardmäßig für alle HealthLake Datenspeicher aktiviert, die nach dem 1. Juni 2023 erstellt wurden. Wenn Ihr Datenspeicher vor diesem Datum erstellt wurde, können Sie ein Supportticket einreichen, um den `$export` Vorgang zu aktivieren. Erstellen Sie einen Fall mit AWS Support Center Console. Um Ihren Fall zu erstellen, melden Sie sich bei Ihrem an AWS-Konto und wählen Sie Fall erstellen aus. • HealthLake Datenspeicher, die vor dem 01.06.23 erstellt wurden, unterstützen nur `$export` Jobanfragen für systemweite Exporte. • HealthLake Datenspeicher, die vor dem 01.06.23 erstellt wurden, unterstützen nicht das Abrufen des	01. Juni 2023

	Status einer FHIR `$export` mithilfe einer `GET` Anfrage am Endpunkt des Datenspeichers.	
SMART auf FHIR-Unterstützung	HealthLake fügt Unterstützung für SMART bei der FHIR-Autorisierung hinzu. Weitere Informationen finden Sie unter SMART on FHIR-Unterstützung für. AWS HealthLake	31. Mai 2023
Validierungen von FHIR-Profilen	HealthLake unterstützt FHIR-Profilvalidierungen zur Definition bestimmter Ressourcentypdefinitionen mithilfe von and/or Einschränkungserweiterungen für Basisressourcentypen. Weitere Informationen finden Sie unter Profilvalidierungen.	31. Mai 2023
Region Asien-Pazifik (Mumbai)	HealthLake ist in der Region Asien-Pazifik (Mumbai) verfügbar. Weitere Informationen finden Sie unter Dienstendpunkte.	4. April 2023

Die Verarbeitung natürlicher Sprache ist standardmäßig deaktiviert	HealthLake hat am 20. Februar 2023 die integrierte Verarbeitung natürlicher Sprache (NLP) in allen Datenspeichern deaktiviert. Sie können ein Supportticket einreichen, um die integrierte NLP-Funktionalität zu aktivieren. Erstellen Sie einen Fall mit AWS Support Center Console. Um Ihren Fall zu erstellen, melden Sie sich bei Ihrem an AWS-Konto und wählen Sie Fall erstellen aus. Weitere Informationen über integriertes NLP finden Sie unter Integration von NLP mit. HealthLake •	20. Februar 2023

SQL-Index und Abfrage mit Amazon Athena	HealthLake unterstützt das Abfragen von FHIR-Daten mit SQL mithilfe von Amazon Athena. Weitere Informationen finden Sie unter Abfragen von HealthLake Daten mit Amazon Athena. • Die SQL-Abfragefunktion ist standardmäßig für alle HealthLake Datenspeicher aktiviert, die nach dem 14.11.2022 erstellt wurden. Wenn Ihr Datenspeicher vor diesem Datum erstellt wurde, können Sie ein Supportticket einreichen, um die SQL-Abfragefunktion zu aktivieren. Erstellen Sie einen Fall mit AWS Support Center Console. Um Ihren Fall zu erstellen, melden Sie sich bei Ihrem an AWS-Konto und wählen Sie Fall erstellen aus. • Bei Verwendung der SQL-Abfragefunktion HealthLake müssen die IAM-Einstellungen für den Zugriff aktualisiert werden. Um in Athena sowohl HealthLake Datenspeicher zu erstellen als auch Zugriff darauf zu gewähren, müssen Sie die `AWSLakeFormationDataAdmin` verwaltete	14. November 2022

	Richtlinie zu Ihrem IAM-Benutzer, Ihrer Gruppe oder Rolle hinzugefügt haben. Sie können die `AWSLakeFormationDataAdmin` Richtlinie verwenden, um Data Lake-Administratoren zu erstellen und Zugriff auf Datenspeicher in Athena zu gewähren. Weitere Informationen finden Sie unter Konfigurieren eines IAM-Benutzers oder einer IAM-Rolle.	
Die Gesamtgröße der Importaufträge wurde erhöht	HealthLake aktualisiert den `Total import job size` für eine `StartFHIRImportJob` Anfrage auf 500 GB. Weitere Informationen finden Sie unter Servicekontingente.	3. Oktober 2022
FHIR-Ressource `Bundle`	HealthLake unterstützt den `Bundle` Ressourcentyp FHIR für die gleichzeitige Verarbeitung mehrerer FHIR-Ressourcen. Weitere Informationen finden Sie unter Bündelung von FHIR-Ressourcen.	5. August 2022
Kontingentaktualisierungen für FHIR-Interaktionen	HealthLake aktualisiert die Kontingente für FHIR-Ressourcenmanagement-Interaktionen. Weitere Informationen finden Sie unter Servicekontingente.	16. Juli 2022

_includeFHIR-Such parameter	HealthLake fügt Unterstützung für den `_include` FHIR-Such parameter hinzu, um zusätzlic he Ressourcen in einer `search` Anfrage zurückzug eben. Weitere Informationen finden Sie unter Erweiterte Suchparameter.	16. Juli 2022
AWS HealthLake ist allgemein verfügbar	HealthLake ist allgemein in allen unterstützten Regionen verfügbar. Weitere Informati onen finden Sie unter Dienstendpunkte.	15. Juli 2021

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.