aws

開発者ガイド

Amazon Cloud Directory

Amazon Cloud Directory: 開発者ガイド

Table of Contents

Amazon Cloud Directory とは?

Amazon クラウドディレクトリは、AWS の可用性が高いマルチテナントディレクトリベースのストアです。これらのディレクトリは、アプリケーションに必要な数億のオブジェクトに自動的に拡張されます。これにより、運用スタッフは、ディレクトリインフラストラクチャを管理ではなく、ビジネスを推進するアプリケーションの開発とデプロイに集中できます。従来のディレクトリシステムとは異なり、クラウドディレクトリは単一の固定階層でディレクトリオブジェクトを整理することを制限しません。

クラウドディレクトリを使用すると、ディレクトリオブジェクトを複数の階層に編成することができ、多くの組織ピボットやディレクトリ情報間の関係をサポートします。たとえば、ユーザーのディレクトリは、構造、場所、およびプロジェクト所属の報告に基づいて階層表示を提供することができます。同様に、デバイスのディレクトリは、その製造元、現在の所有者、および物理的な場所に基づいて、複数の階層表示を持つことができます。

本質的には、クラウドディレクトリは、開発者のための基本的な構成要素を提供する、専用のグラフベースのディレクトリストアです。クラウドのディレクトリを使用すると、開発者は以下の操作を実行します。

- デプロイメント、グローバル規模、可用性、パフォーマンスを気にすることなく、簡単にディレクトリベースのアプリケーションを作成できます
- ユーザーおよびグループの管理、アクセス権限またはポリシー管理、デバイスレジストリ、顧客管理、アドレス帳、アプリケーションまたは製品カタログを提供するアプリケーションを構築します
- 新しいディレクトリオブジェクトを定義するか、アプリケーションのニーズに合わせて既存のタイプを拡張し、書き込みに必要なコードを減らします。
- クラウドディレクトリのレイヤー化アプリケーションの複雑さを軽減します。
- 時間の経過とともにスキーマ情報の進化を管理し、将来の消費者との互換性を保証します。

クラウドディレクトリには、クラウドディレクトリベースのディレクトリに格納されているさまざまなオブジェクトやポリシーにアクセスする一連の API 操作が含まれています。使用可能なオペレーションのリストについては、「」を参照してください。Amazon Cloud Directory API のアクション。操作のリストと、各 API アクションを実行するための必要な権限については、「Amazon Cloud Directory API のアクセス許可: アクション、リソース、条件リファレンス」 を参照してください。

サポートされている Cloud Directory リージョンのリストについては、「」を参照してください。AWS リージョンとエンドポイントドキュメント内) を参照してください。その他のリソースについては、「Cloud Directory リソース」を参照してください。

Cloud Directory にないもの

クラウドディレクトリは、ディレクトリインフラストラクチャを管理または移行する IT 管理者向けのディレクトリサービスではありません。

はじめに

この入門演習では、スキーマを作成します。You then choose to create a directory from that same schema or from any of the sample schemas that are available in the AWS Directory Service console. 必須ではありませんが、コンソールを使用する前に「クラウドディレクトリの主要概念の理解」を確認し、コア機能や用語について理解されておくことをお勧めします。

トピック

- スキーマの作成
- Amazon クラウドディレクトリを作成する
- Cloud Directory インターフェイス VPC エンドポイントの使用

スキーマの作成

Amazon Cloud Directory では、スキーマの作成に準拠した JSON ファイルをアップロードすることができます。新しいスキーマを作成するには、一から独自の JSON ファイルを作成するか、コンソールに表示されている既存のスキーマのいずれかをダウンロードします。次に、カスタムのスキーマとしてアップロードします。詳細については、「カスタムスキーマ」を参照してください。

また、Cloud Directory API を使用して、スキーマの作成、削除、ダウンロード、一覧表示、公開、更新、アップグレードを行うことができます。スキーマ API オペレーションの詳細については、「Amazon クラウドディレクトリの API リファレンスガイド」を参照してください。

任意の方法に応じて、以下のいずれかの手順を選択します。

カスタムスキーマを作成するには

1. 左AWS Directory Service コンソールナビゲーションペインのクラウドのディレクトリ] で、スキーマ。
2. すべての新しいスキーマを定義する JSON ファイルを作成します。JSON ファイルを整形する方法の詳細については、「JSON スキーマ形式」を参照してください。
3. コンソールで、[] を選択します。今すぐアップロード。
4. 左今すぐアップロードダイアログで、スキーマの名前を入力します。
5. Selectファイルを選択] を選択し、作成したばかりの新しい JSON ファイルを選択して、[] を選択します。オープン。

6. [Upload] を選択します。これにより、新しいスキーマがスキーマライブラリに追加され、開発ステータス。スキーマの状態に関する詳細については、「スキーマのライフサイクル」を参照してください。

コンソールに表示されている既存のスキーマに基づいてカスタムスキーマを作成するには

1. 左AWS Directory Service コンソールナビゲーションペインのクラウドのディレクトリ] で、スキーマ。
2. スキーマが表示されているテーブルで、コピーするスキーマの近くのオプションを選択します。
3. [Actions] を選択します。
4. 選択スキーマのダウンロード。
5. JSON ファイルの名前を変更し、必要に応じて編集してから、ファイルを保存します。JSON ファイルを整形する方法の詳細については、「JSON スキーマ形式」を参照してください。
6. コンソールで、[] を選択します。今すぐアップロード] を選択し、編集した JSON ファイルを選択して、[] を選択します。オープン。

   これにより、新しいスキーマがスキーマライブラリに追加され、開発ステータス。スキーマの状態に関する詳細については、「スキーマのライフサイクル」を参照してください。

Amazon クラウドディレクトリを作成する

Amazon クラウドディレクトリにディレクトリを作成する前に、まず AWS Directory Service にスキーマを適用する必要があります。スキーマなしでディレクトリを作成することはできず、通常はスキーマが 1 つ適用されます。ただし、クラウドディレクトリ API 操作を使用して、ディレクトリに追加のスキーマを適用します。詳細については、『Amazon Cloud Directory API リファレンスガイド』の「ApplySchema」を参照してください。

Cloud Directory を作成するには

1. 左AWS Directory Service コンソールナビゲーションペインのクラウドのディレクトリ] で、[ディレクトリ。
2. 選択Cloud Directory をセットアップする。
3. []新しいディレクトリに適用するスキーマを選択してください[] に、わかりやすいディレクトリ名 (など) を入力します。`User Repository`[] をクリックし、以下のいずれかのオプションを選択します。

- 管理スキーマ
- サンプルスキーマ
- カスタムスキーマ

サンプルスキーマとカスタムスキーマは、開発状態 (デフォルトで)。スキーマの状態に関する詳細については、「スキーマのライフサイクル」を参照してください。スキーマをディレクトリに適用するには、スキーマを発行済み状態に変換する必要があります。コンソールを使用してサンプルスキーマを正常に発行するには、次の操作に対する権限が必要です。

- `clouddirectory:Get*`
- `clouddirectory:List*`
- `clouddirectory:CreateSchema`
- `clouddirectory:CreateDirectory`
- `clouddirectory:PutSchemaFromJson`
- `clouddirectory:PublishSchema`
- `clouddirectory:DeleteSchema`

サンプルスキーマは AWS によって提供される読み取り専用テンプレートであるため、直接発行することはできません。代わりに、サンプルスキーマに基づいてディレクトリを作成することを選択すると、コンソールは選択したサンプルスキーマの一時コピーを作成し、それを作成し開発状態。次に、その開発スキーマのコピーが作成され、発行済み状態にされます。発行されると、開発スキーマが削除されるため、サンプルスキーマを発行するときに、`DeleteSchema` アクションが必要になります。

4. [Next] を選択します。
5. ディレクトリ情報を確認し、必要な変更を加えます。情報が正しい場合は、[作成] を選択します。

Cloud Directory インターフェイス VPC エンドポイントの使用

Amazon Virtual Private Cloud (Amazon VPC) を使用して AWS リソースをホストする場合は、VPC とCloud Directory の間にプライベート接続を確立できます。この接続を使用すると、Cloud Directory はパブリックインターネットを経由せずに、VPC のリソースと通信できます。

Amazon VPC は AWS のサービスで、お客様の定義する仮想ネットワークで AWS リソースを起動するために使用できる AWS のサービスです。VPC を使用すると、IP アドレス範囲、サブネット、ルートテーブル、ネットワークゲートウェイなどのネットワーク設定を制御できます。VPC をCloud Directory に接続するには、インターフェイス VPC エンドポイントCloud Directory 用. このエンドポイントは、インターネットゲートウェイ、ネットワークアドレス変換 (NAT) インスタンス、または VPN 接続を必要とせず、信頼性が高くスケーラブルな Cloud Directory への接続を提供します。詳細については、「」を参照してください。Amazon VPC とは?()Amazon VPC ユーザーガイド。

インターフェイス VPC エンドポイントは AWS PrivateLink を利用しています。これは、elastic network interface とプライベート IP アドレスを使用して AWS のサービス間のプライベート通信を可能にする AWS のテクノロジーです。詳細については、「」を参照してください。AWS サービス用 AWS PrivateLink。

次のステップは Amazon VPC のユーザー向けです。詳細については、「」を参照してください。Amazon VPC の使用開始()Amazon VPC ユーザーガイド。

Availability

現在、Cloud Directory は、次のリージョンで VPC エンドポイントをサポートしています。

- 米国東部 (オハイオ)
- 米国東部 (バージニア北部)
- 米国西部 (オレゴン)
- アジアパシフィック (シンガポール)
- アジアパシフィック (シドニー)
- カナダ (中部)
- 欧州 (フランクフルト)
- 欧州 (アイルランド)
- 欧州 (ロンドン)
- AWS GovCloud (米国西部)

Cloud Directory 用の VPC を作成する

VPC で Cloud Directory の使用を開始するには、Amazon VPC コンソールを使用して、Cloud Directory の VPC エンドポイントを作成します。詳細については、「インターフェイスエンドポイントの作成」を参照してください。

- を使用する場合サービスのカテゴリ] で、AWS サービス。
- [サービス名] には [**com.amazonaws.region.clouddirectory**] を選択します。これにより、Cloud Directory 操作用の VPC エンドポイントが作成されます。

一般情報については、」Amazon VPC とは?()Amazon VPC ユーザーガイド。

Cloud Directory VPC エンドポイントへのアクセスの制御

VPC エンドポイントポリシーは、エンドポイントの作成時または変更時にエンドポイントにアタッチする IAM リソースポリシーです。エンドポイントの作成時にポリシーをアタッチしない場合、サービスへのフルアクセスを許可するデフォルトのポリシーがアタッチされます。エンドポイントポリシーは、IAM ユーザーポリシーやサービス固有のポリシーを上書き、または置き換えません。これは、エンドポイントから指定されたサービスへのアクセスを制御するための別のポリシーです。

エンドポイントのポリシーは、JSON 形式で記述される必要があります。詳細については、「」を参照してください。VPC エンドポイントによるサービスのアクセスコントロール()Amazon VPC ユーザーガイド。

Cloud Directory のエンドポイントポリシーの例を次に示します。このポリシーでは、VPC を介して Cloud Directory に接続するユーザーは、ディレクトリをリストできますが、他の Cloud Directory アクションを実行することはできません。

```
{
  "Statement": [
    {
      "Sid": "ReadOnly",
      "Principal": "*",
      "Action": [
        "clouddirectory:ListDirectories"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Cloud Directory の VPC エンドポイントポリシーを変更するには

1. Amazon VPC コンソール (https://console.aws.amazon.com/vpc/) を開きます。
2. ナビゲーションペインで、[エンドポイント] を選択します。

3. Cloud Directory のエンドポイントをまだ作成していない場合は、エンドポイントの作成。次に、**com.amazonaws.region.clouddirectory**を選択し、エンドポイントの作成。
4. 選択**com.amazonaws.region.clouddirectory**エンドポイントを選択し、ポリシータブを画面の下部で.
5. [ポリシーの編集] を選択してポリシーを変更します。

詳細については、「」を参照してください。VPC エンドポイントによるサービスのアクセスコントロール()Amazon VPC ユーザーガイド。

クラウドディレクトリの主要概念の理解

Amazon Cloud Directory は、スキーマ指向のさまざまな種類のオブジェクトを作成できるディレクトリベースのデータストアです。

トピック

Schema

スキーマとは、ディレクトリに作成できるオブジェクトとその構成方法を定義するファセットの集合です。スキーマはまた、データの完全性と相互運用性を強制します。1 つのスキーマを複数のディレクトリに同時に適用できます。詳細については、「Schemas」を参照してください。

Facets

ファセットは、スキーマ内で定義された属性、制約、およびリンクの集合です。組み合わされたファセットは、ディレクトリ内のオブジェクトを定義します。たとえば、Person と Device は、複数のデバイスの関連付けを使用して従業員を定義するファセットにすることができます。詳細については、「Facets」を参照してください。

マネージド型スキーマ

アプリケーションを迅速に開発および保守しやすくするために提供されるスキーマ。詳細については、「マネージド型スキーマ」を参照してください。

サンプルスキーマ

サンプルスキーマのセットは、AWS Directory Service コンソールでデフォルトで提供されます。たとえば、Person、Organization、および Device はすべてサンプルスキーマです。詳細については、「サンプルスキーマ」を参照してください。

カスタムスキーマ

スキーマセクションからアップロードできる、または AWS ディレクトリサービスコンソールのクラウドディレクトリ作成プロセス中にアップロードできる、または API 呼び出しによって作成された、ユーザーが定義した 1 つ以上のスキーマ。

Directory

ディレクトリは、複数階層構造で編成された特定のタイプのオブジェクトを含むスキーマベースのデータストアです (詳細については、「ディレクトリ構造」を参照)。たとえば、ユーザーのディレクトリは、構造、場所、およびプロジェクト所属の報告に基づいて階層表示を提供することができます。同様に、デバイスのディレクトリは、その製造元、現在の所有者、および物理的な場所に基づいて、複数の階層表示を持つことができます。

ディレクトリは、データストアの論理境界を定義し、サービス内の他のすべてのディレクトリと完全に分離します。また、個別リクエストの境界を定義します。単一のトランザクションまたはクエリは、単一のディレクトリのコンテキスト内で実行されます。スキーマなしでディレクトリを作成することはできず、通常はスキーマが 1 つ適用されます。ただし、Cloud Directory API 操作を使用して、ディレクトリに追加のスキーマを適用します。詳細については、「」を参照してください。`ApplySchema`()Amazon Cloud Directory API リファレンスガイド。

Objects

オブジェクトは、ディレクトリ内の構造化されたデータエンティティです。ディレクトリ内のオブジェクトは、通常、情報の発見やポリシーの実施を目的として、物理エンティティまたは論理エンティティに関するメタデータ (または属性) を取得することを目的としています。たとえば、ユーザー、デバイス、アプリケーション、AWS アカウント、EC2 インスタンス、Amazon S3 バケットはすべて、ディレクトリ内のさまざまなタイプのオブジェクトとして表現できます。

オブジェクトの構造と型情報は、ファセットの集合として表現されます。`Path` または `ObjectIdentifier` を使ってオブジェクトにアクセスできます。オブジェクトには、メタデータのユーザー定義の単位である属性も含めることができます。たとえば、ユーザーオブジェクトは email-address と呼ばれる属性を持つことができます。属性は常にオブジェクトに関連付けられます。

Policies

ポリシーは、アクセス権限や機能を格納するのに便利な特別な種類のオブジェクトです。ポリシーは、`LookupPolicy` API アクションを提供します。参照ポリシーアクションは、任意のオブジェク

トを開始入力として参照します。次に、ディレクトリからルートに移動します。このアクションは、ルートへの各パスで遭遇する すべてのポリシーオブジェクトを収集します。クラウドディレクトリは、これらのポリシーのいずれも解釈しません。代わりに、クラウドディレクトリユーザーは、独自の特殊なビジネスロジックを使用してポリシーを解釈します。

たとえば、従業員の情報を格納するシステムを想像してみてください。従業員は職務機能によってグループ化されます。人事部門と経理部門のメンバーに対して異なる権限を設定したいと考えています。人事部門のメンバーは給与情報にアクセスでき、経理部門は元帳情報にアクセスできます。これらの権限を確立するには、これらの各グループにポリシーをアタッチします。ユーザーのアクセス権限を評価するときは、そのユーザーのオブジェクトに対して LookupPolicy API アクションを使用できます。-LookupPolicyAPI アクションは、ツリー上を、指定されたポリシーオブジェクトからルートまで移動します。ノードごとに停止して、アタッチされているポリシーがあるか確認し、あれば返します。

ポリシーのアタッチメント

ポリシーは、通常の親子アタッチメントと特別なポリシーアタッチメントの 2 つの方法で他のオブジェクトにアタッチできます。通常の親子アタッチメントを使用して、親ノードにポリシーをアタッチすることができます。これは、データディレクトリ内のポリシーを見つけるための簡単なメカニズムを提供するのに便利です。ポリシーは子を持つことはできません。LookupPolicy API コール中に、親子アタッチメントを介してアタッチされたポリシーは返されません。

ポリシーオブジェクトは、ポリシーのアタッチメントを介して他のオブジェクトにもアタッチできます。これらのポリシーのアタッチメントは、AttachPolicy API アクションおよび DetachPolicy API アクションを使用して管理できます。ポリシーのアタッチメントを使用すると、LookupPolicy API を使用するときにポリシーノードを配置できます。

ポリシースキーマの仕様

ポリシーの使用を開始するには、まずポリシーの作成をサポートするファセットをスキーマに追加する必要があります。これを実現するには、ファセットを作成し、ファセットの objectType を POLICY に設定します。POLICY タイプのファセットを使用してオブジェクトを作成すると、オブジェクトにポリシー機能が確実に適用されます。

ポリシーファセットは、ユーザーが定義に追加した属性に加えて、以下の 2 つの属性を継承します。

- policy_type (文字列、必須) – これは、複数の異なるポリシーの使用を区別するために指定できる識別子です。ポリシーが論理的に明確なカテゴリに該当する場合は、ポリシータイプ属性を適切に設

定することをお勧めします。LookupPolicy API は、アタッチされたポリシーのポリシータイプを返します (「PolicyAttachment」を参照)。これにより、探している特定のポリシータイプを簡単にフィルタリングできます. policy_type を使用して、ドキュメントを処理または解釈する方法も判断できます。

- policy_document (バイナリ、必須) – この属性には、アプリケーション固有のデータ (ポリシーに関連付けられたアクセス権限の付与など) を保存できます。必要に応じて、アプリケーション関連のデータをファセットの通常の属性に保存することもできます。

Policy API の概要

ポリシーの操作には、さまざまな特殊な API アクションを使用できます。使用可能なオペレーションのリストについては、[Amazon クラウドディレクトリのアクション] を参照してください。

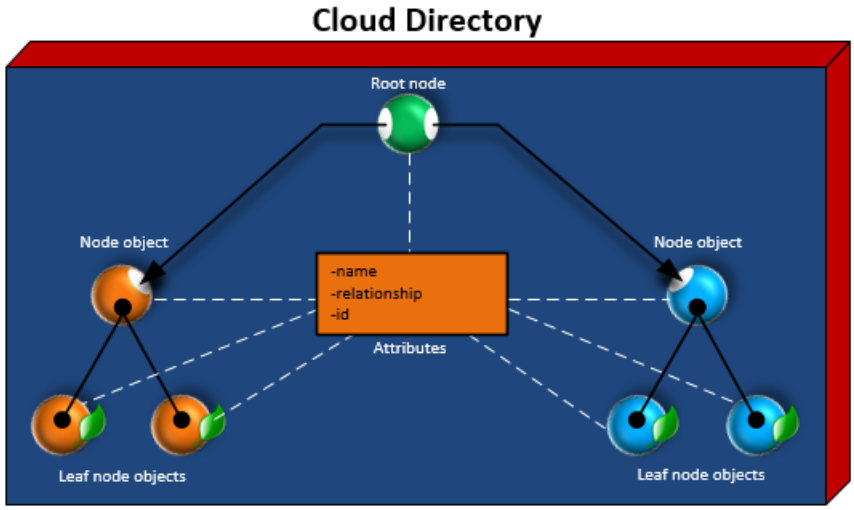
ポリシーオブジェクトを作成するには、CreateObject API アクションを適切なファセットと共に使用します。

- オブジェクトのポリシーをアタッチまたはデタッチするには、AttachPolicy アクションと DetachPolicy アクションをそれぞれ使用します。
- ツリー上のオブジェクトにアタッチされているポリシーを検索するには、LookupPolicy API アクションを使用します。
- 特定のオブジェクトにアタッチされているポリシーを表示するには、ListObjectPolicies API アクションを使用します。

操作のリストと、各 API アクションを実行するための必要な権限については、「Amazon Cloud Directory API のアクセス許可: アクション、リソース、条件リファレンス」 を参照してください。

ディレクトリ構造

ディレクトリ内のデータは、次の図に示すように、ノード、リーフノード、ノード間のリンクで構成されるツリーパターンで階層構造になっています。これは、階層データをモデル化、格納、およびすばやくトラバースするアプリケーション開発に役立ちます。



ルートノード

ルートは、階層内の親ノードと子ノードを編成するために使用されるディレクトリ内の最上位ノードです。これは、ファイルシステム内のフォルダにサブフォルダとファイルを含める方法と類似しています。

Node

ノードは、子オブジェクトを持つことができるオブジェクトを表します。たとえば、ノードは、種々のユーザーオブジェクトが子またはリーフノードであるマネージャーのグループを論理的に表すことができます。ノードオブジェクトは親を 1 つだけ持つことができます。

リーフノード

リーフノードは、親ノードに子が直接接続されているかいないかにかかわらず、子を持たないオブジェクトを表します。たとえば、ユーザーまたはデバイスオブジェクトです。リーフノードオブジェクトは、複数の親を持つことができます。リーフノードオブジェクトを親ノードに接続することは必須ではありませんが、ルートからのパスがなければオブジェクトにアクセスできるのは `NodeId` に限られるため、親ノードに接続するよう強くお勧めします。そのようなオブジェクトの ID を間違えた場合は、再度その ID を見つける方法がありません。

ノードリンク

ノード間の接続。クラウドディレクトリは、親子リンク、ポリシーリンク、およびインデックス属性リンクなど、ノード間のさまざまなリンクタイプをサポートしています。

Schemas

Amazon Cloud Directory では、ディレクトリ内で作成できるオブジェクトのタイプ (ユーザー、デバイス、組織) を定義し、オブジェクトクラスごとにデータの検証を実施して、経過とともにスキーマへの変更を処理します。より具体的には、次のスキーマを定義します。

- ディレクトリ内のオブジェクト (Person、Organization_Person など) にマップできる 1 つ以上のタイプのファセット
- ディレクトリ内のオブジェクトにマップされる可能性のある属性 (名前、説明など)。属性は、さまざまなタイプのファセットで有効またはオプションにすることができ、ファセットのコンテキスト内で定義されます。
- オブジェクト属性 (必須、整数、文字列など) に適用される制約。

スキーマがディレクトリに適用されると、そのディレクトリ内のすべてのデータは適用されたスキーマに準拠する必要があります。このように、スキーマ定義は本質的に、適用されたスキーマを持つ複数のディレクトリを構築するために使用できる設計図です。構築されると、適用されるスキーマは元の設計図とは異なる場合があります。

適用したスキーマは、後でバージョニングを使用して更新され、それを使用するすべてのディレクトリに再適用される場合があります。詳細については、「スキーマのインプレースアップグレード」を参照してください。

クラウドディレクトリでは、API オペレーションがスキーマの作成、読み込み、更新、削除を行います。これにより、スキーマの内容がプログラムエージェントによって簡単に消費されるようになります。そのようなエージェントはディレクトリにアクセスして、ディレクトリ内のデータに適用されるファセット、属性、および制約のフルセットを検出します。スキーマ API オペレーションの詳細については、「Amazon Cloud Directory API リファレンスガイド」を参照してください。

クラウドディレクトリでは、スキーマ作成に準拠した JSON ファイルのアップロードをサポートしています。また、AWS Directory Services コンソールを使用してスキーマの作成や管理をすることもできます。詳細については、「Amazon クラウドディレクトリを作成する」を参照してください。

トピック

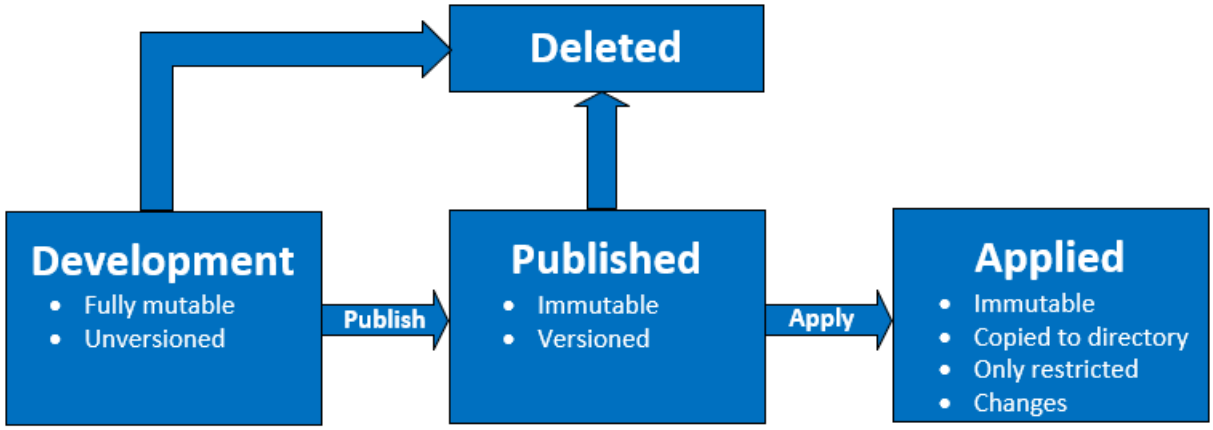
- スキーマのライフサイクル
- Facets
- スキーマのインプレースアップグレード

- マネージド型スキーマ
- サンプルスキーマ
- カスタムスキーマ
- 属性リファレンス
- 属性ルール
- 仕様の形式

スキーマのライフサイクル

クラウドディレクトリは、スキーマの開発に役立つスキーマライフサイクルを提供します。このライフサイクルは、次の 3 つの状態で構成されています。開発、発行済み、適用済みです。これらの状態は、スキーマの構築とディストリビューションを容易にするように設計されています。これらの各状態には、この取り組みを支援するさまざまな機能があります。

次の図は、可能な移行と用語を示しています。すべてのスキーマ移行はコピーオンライトです。たとえば、開発スキーマを発行しても、開発スキーマは変更または削除されません。



スキーマは、開発または発行済みの状態のときに削除できます。スキーマの削除は元に戻すことができません。また、削除したスキーマは復元できません。

開発、発行済み、適用済みの状態のスキーマには、対応する ARN があります。これらの ARN は、API オペレーションで使用され、API が操作するスキーマを記述します。スキーマ ARN を調べることによって、スキーマの状態を識別することは簡単です。

- 開発: `arn:aws:clouddirectory:us-east-1:1234567890:schema/development/SchemaName`

- 発行済み: `arn:aws:clouddirectory:us-east-1:1234567890:schema/published/SchemaName/Version`
- 申請済み: `arn:aws:clouddirectory:us-east-1:1234567890:directory/directoryid/schema/SchemaName/Version`

開発状態

スキーマは、最初に開発状態で作成されます。この状態のスキーマは完全に変更可能です。ファセットと属性を自由に追加または削除することができます。スキーマ設計の大部分はこの状態で行われます。この状態のスキーマには名前はありますが、バージョンはありません。

発行済み状態

発行済みスキーマの状態には、データディレクトリに適用する準備ができているスキーマが格納されます。スキーマは開発状態から発行済み状態に発行されます。発行済み状態のスキーマは変更できません。発行済みスキーマは、任意の数のデータディレクトリに適用できます。

発行済みスキーマおよび適用済みスキーマには、関連付けられたバージョンが必要です。バージョンの詳細については、「スキーマのバージョニング」を参照してください。

適用済み状態

発行済みスキーマをデータディレクトリに適用できます。データディレクトリに適用されたスキーマを適用済みと言います。スキーマをデータディレクトリに適用すると、オブジェクトを作成するときにスキーマのファセットを使用できます。同じデータディレクトリに複数のスキーマを適用できます。適用されたスキーマでは、次の変更のみが許可されます。

- 適用されたスキーマにファセットを追加する
- 適用されたスキーマに必須でない属性を追加する

Facets

ファセットは、スキーマ内で最も基本的な抽象化です。これらの属性は、ディレクトリ内のオブジェクトに関連付けることができる一連の属性を表し、概念が LDAP オブジェクトクラスと類似しています。各ディレクトリオブジェクトには、それに関連するファセットが一定数まで存在することがあります。詳細については、「Amazon Cloud Directory の制限」を参照してください。

各ファセットは、独自の属性セットを保持します。各ファセットは、ファセット名、バージョン情報、動作などの基本的なメタデータで構成されています。スキーマ ARN、ファセット、および属性の組み合わせは、オブジェクトの一意性を定義します。

オブジェクトファセットのセット、ファセットの制約、およびファセット間の関係は、抽象スキーマ定義を構成します。スキーマファセットは、以下のついての制約を定義するために使用されます。

1. オブジェクトで許可された属性
2. オブジェクトに適用できるポリシータイプ

スキーマに必要なファセットを追加したら、そのスキーマをディレクトリに追加し、該当するオブジェクトを作成できます。たとえば、コンピュータ、スマートフォン、タブレットなどのファセットを追加して、デバイススキーマを定義できます。次に、これらのファセットを使用してコンピュータオブジェクト、スマートフォンオブジェクト、タブレットオブジェクトを、スキーマの適用先のディレクトリに作成できます。

クラウドディレクトリのスキーマサポートにより、アプリケーションの破損を心配することなく、ファセットや属性の追加や変更が容易になります。詳細については、「スキーマのインプレースアップグレード」を参照してください。

スキーマのインプレースアップグレード

Cloud Directory では、既存のスキーマ属性とファセットを更新し、アプリケーションを AWS 提供のサービスと統合できます。発行済みまたは適用済みの状態のスキーマにはバージョンがあり、変更することはできません。詳細については、「スキーマのライフサイクル」を参照してください。

スキーマのバージョニング

スキーマのバージョンは、データの特定のルールや形式に準拠するようにアプリケーションをプログラミングするときに、開発者が指定できるスキーマの一意の識別子を示します。開発者は、Cloud Directory のバージョニングの主に 2 つの異なる方法から成り、のバージョニングが重要であることを理解することが重要です。この 2 つとは、メジャーバージョンとマイナーバージョンであり、今後のスキーマのアップグレードとアプリケーションの関係を決定します。

メジャーバージョン

メジャーバージョンは、スキーマの主要なバージョン変更を追跡するためのバージョン識別子です。最大 10 文字を使用できます。同じスキーマの異なるバージョンは完全に独立しています。たとえ

ば、同じ名前でも異なるバージョンの 2 つのスキーマは、それぞれ独自の名前空間を持ち、完全に異なるスキーマとして扱われます。

下位互換性のない変更

メジャーバージョンを変更するのは、スキーマ間に互換性がない場合に限ります。たとえば、既存の属性のデータ型を変更する場合 (`string` を `integer` に変更するなど) や、スキーマから必須属性を削除する場合などが該当します。下位互換性のない変更を行うには、前のスキーマバージョンから新しいスキーマバージョンにディレクトリのデータを移行する必要があります。

マイナーバージョン

マイナーバージョンは、スキーマのインプレースアップグレード、または下位互換性のあるアップグレード (属性やファセットを追加するなど) を行う場合に使用するバージョン識別子です。マイナーバージョンを使用してアップグレードしたスキーマは、実行中のアプリケーションを一切破損することなく、それを使用するすべてのディレクトリにわたってインプレースで適用できます。これには、本番稼働用環境で使用されているディレクトリも含まれます。ユースケース例については、インプレーススキーマアップグレードにより Amazon Cloud Directory スキーマの変更を簡単に適用する方法Cloud Directory Directory ブログの「」を参照してください。

マイナーバージョンの情報と履歴は、他のスキーマ情報と共に、スキーマメタデータリポジトリに保存されます。オブジェクトには一切のマイナーバージョン情報が保存されません。マイナーバージョンを導入する利点は、メジャーバージョンが変更されない限り、クライアントコードがシームレスに動作することです。

マイナーバージョン制限

Cloud Directory はマイナーバージョンを保持するため、最大5つまで制限されます。ただし、マイナーバージョンの制限は、公開スキーマと適用スキーマに対して次のように異なる方法で適用されます。

- 適用済みのスキーマ: マイナーバージョンの制限を超えると、Cloud Directory は古いマイナーバージョンを自動的に削除します。
- 発行済みのスキーマ: マイナーバージョンの制限を超えると、Cloud Directory はマイナーバージョンを削除しませんが、`LimitExceededException`制限を超過しました。マイナーバージョンの制限を超えると、スキーマを削除できます。DeleteSchemaAPI を使用するか、制限引き上げをリクエストします。

スキーマアップグレード API オペレーションの使用

発行済みのスキーマをアップグレードするには、UpgradePublishedSchema API コールを使用できます。スキーマのアップグレードは、UpgradeAppliedSchema API コールにより、スキーマに依存するディレクトリにインプレースで適用されます。また、適用済みのスキーマのメジャーバージョンとマイナーバージョンを取得するには、GetAppliedSchemaVersion を呼び出します。ディレクトリに関連付けられたスキーマ ARN とスキーマ改訂履歴を表示するには、を呼び出します。ListAppliedSchemaArns。Cloud Directory は、適用済みスキーマの最新バージョンを 5 つ保持します。

例については、インプレーススキーマアップグレードにより Amazon Cloud Directory スキーマの変更を簡単に適用する方法Cloud Directory Directory ブログの「」を参照してください。このブログ投稿では、Cloud Directory でスキーマのインプレースアップグレードを実行し、スキーマバージョンを使用する方法を示しています。既存のファセットへの属性の追加、スキーマへの新しいファセットの追加、新しいスキーマの発行、実行中のディレクトリに対するスキーマの適用を通じて、スキーマのアップグレードをインプレースで完了する方法について説明されています。ディレクトリスキーマのバージョン履歴を表示する方法も示されています。これにより、ディレクトリフリートが同じバージョンのスキーマを実行していること、およびスキーマ変更の正しい履歴が適用されていることを確認できます。

マネージド型スキーマ

Cloud Directory は、マネージド型スキーマを使用して、迅速なアプリケーション開発を容易にします。マネージド型スキーマにより、ディレクトリを作成し、そのディレクトリからより速いペースでオブジェクトの作成と取得を開始できます。詳細については、「ディレクトリを作成する」を参照してください。

現在、QuickStartSchema と呼ばれる 1 つのマネージド型スキーマがあります。型付きリンク などの構築を使用して、豊富な階層データモデルを構築し、オブジェクト間の関係を確立できます。次に、階層をトラバースしてデータ内の情報に対してクエリを実行できます。

QuickStartSchema マネージド型スキーマは、次の JSON によって表されます。

```
QuickStartSchema: {
    "facets": {
        "DynamicObjectFacet": {
            "facetStyle": "DYNAMIC"
        },
        "DynamicTypedLinkFacet": {
```

```
            "facetAttributes": {
                "DynamicTypedLinkAttribute": {
                    "attributeDefinition": {
                        "attributeRules": {},
                        "attributeType": "VARIANT",
                        "isImmutable": false
                    },
                    "requiredBehavior": "REQUIRED_ALWAYS"
                }
            },
            "identityAttributeOrder": [
                "DynamicAttribute"
            ]
        }
    }
}
```

QuickStartSchema ARN

QuickStartSchema マネージド型スキーマでは、以下の ARN が使用されます。

```
String QUICK_START_SCHEMA_ARN = "arn:aws:clouddirectory:::schema/managed/
quick_start/1.0/001" ;
```

たとえば、この ARN を使用して、以下に示す ExampleDirectory という名前のディレクトリを作成できます。

```
CreateDirectoryRequest createDirectoryRequest = new CreateDirectoryRequest()
    .withName("ExampleDirectory") // Directory name
    .withSchemaArn(QUICK_START_SCHEMA_ARN);
```

ファセットスタイル

指定された任意のファセット Static および Dynamic で、異なる 2 つのスタイルがあります。

静的なファセット

静的なファセットは、データ型を含む属性のリストなど、ディレクトリのデータモデルのすべての詳細があり、必須のフィールドや一意のフィールドなど、属性の制約を定義する場合に最適な選択となります。Cloud Directory では、オブジェクトの作成や変更時にデータ制約とルールチェックが強制されます。

動的なファセット

属性の数を変更したり、属性内に保存されているデータ値を変更したりする柔軟性が必要なときに、動的なファセットを使用できます。Cloud Directory では、オブジェクトの作成や変更時にデータ制約やルールチェックは行われません。

動的なファセットでスキーマを作成した後、オブジェクトの作成中に、任意の属性を定義できます。Cloud Directory は属性をキーと値のペアとして受け取り、提供されたオブジェクトに保存します。

新規または既存のスキーマに動的なファセットを追加できます。また、単一のスキーマ内で静的および動的なファセットを組み合わせ、ディレクトリ内でファセットの各スタイルの利点を得ることができます。

動的なファセットを使用して属性を作成すると、`Variant` データ型として作成されます。として定義された属性の値を格納するには`Variant`データ型は、Cloud Directory でサポートされている任意のプリミティブデータ型の値を使用できます。`String`または`Binary`。時間の経過とともに、属性の値を別のデータ型に変更することもできます。データ検証の強制はありません。

動的なファセットを使用して、次のタイプのオブジェクトを定義できます。

- `NODE`
- `LEAF_NODE`
- `POLICY`

管理スキーマ、動的ファセット、バリアントデータ型の詳細、およびユースケースの例については、AWS マネージド型スキーマを使用してアプリケーションを迅速に Amazon Cloud Directory で開発する方法を Amazon Cloud Directory ブログでご覧ください。

サンプルスキーマ

クラウドディレクトリには、組織、個人、およびデバイスのサンプルスキーマが用意されています。次のセクションでは、さまざまなサンプルスキーマと、それぞれの相違点を表示しています。

Organizations

次の表に、Organizations サンプルスキーマに含まれるファセットを示します。

「Organization」ファセット	データ型	長さ。	必須動作か否か	説明
account_id	文字列	1024	N	組織の一意の ID
account_name	文字列	1024	N	組織名
organization_status	文字列	1024	N	「アクティブ」、「非アクティブ」、「停止」、「クローズ済み」などのステータス
mailing_address (street1)	文字列	1024	N	この会社/団体の物理的な郵送先住所
mailing_address (street2)	文字列	1024	N	この会社/団体の物理的な郵送先住所
mailing_address (city)	文字列	1024	N	この会社/団体の物理的な郵送先住所
mailing_address (state)	文字列	1024	N	この会社/団体の物理的な郵送先住所
mailing_address (country)	文字列	1024	N	この会社/団体の物理的な郵送先住所
mailing_address (postal_code)	文字列	1024	N	この会社/団体の物理的な郵送先住所
email	文字列	1024	N	組織の E メール ID
web_site	文字列	1024	N	ウェブサイトの URL
telephone_number	文字列	1024	N	組織の電話番号
description	文字列	1024	N	組織の説明

「Legal_Entity」ファセット	データ型	長さ。	必須動作か否か	説明
registered_company_name	文字列	1024	N	法人名
mailing_address (street1)	文字列	1024	N	この会社/団体の物理的な登録済み郵送先住所
mailing_address (street2)	文字列	1024	N	この会社/団体の物理的な登録済み郵送先住所
mailing_address (city)	文字列	1024	N	この会社/団体の物理的な登録済み郵送先住所
mailing_address (state)	文字列	1024	N	この会社/団体の物理的な登録済み郵送先住所
mailing_address (country)	文字列	1024	N	この会社/団体の物理的な登録済み郵送先住所
mailing_address (postal_code)	文字列	1024	N	この会社/団体の物理的な登録済み郵送先住所
industry_vertical	文字列	1024	N	Industry Segment
billing_currency	文字列	1024	N	Billing currency
tax_id	文字列	1024	N	税金識別番号

Person

次の表に、Person サンプルスキーマに含まれるファセットを示します。

「Person」ファセット	データ型	長さ。	必須動作か否か	説明
display_name	文字列	1024	N	エンドユーザーに表示するのに適したユーザーの名前。
first_name	文字列	1024	N	指定されたユーザーの名前、またはほとんどの欧米言語名
last_name	文字列	1024	N	ユーザーの家族名、またはほとんどの欧米言語の姓
middle_name	文字列	1024	N	ユーザーのミドルネーム
nickname	文字列	1024	N	「Robert」の代わりに「Bob」や「Bobby」など、実生活でユーザーに対処するカジュアルな方法。
email	文字列	1024	N	ユーザーの E メールアドレス
mobile_phone_number	文字列	1024	N	ユーザーの電話番号
home_phone_number	文字列	1024	N	ユーザーの電話番号
username	文字列	1024	Y	ユーザー用の一意の識別子
profile	文字列	1024	N	ユニフォームリソースロケータであり、ユーザーのオンラインプロファイル (ウェブページなど) を示す場所を指す URI。
picture	文字列	1024	N	ユニフォームリソースロケータであり、ユーザーの

「Person」ファセット	データ型	長さ。	必須動作か否か	説明
				イメージを表すリソースの場所を指す URI。
website	文字列	1024	N	URL
timezone	文字列	1024	N	ユーザーのタイムゾーン
サイト	文字列	1024	N	通貨、日付時刻形式、数値表現などの項目をローカライズする目的で、ユーザーのデフォルトの場所を示すために使用される。
address (street1)	文字列	1024	N	このユーザーの物理的なメールアドレス。
address (street2)	文字列	1024	N	このユーザーの物理的なメールアドレス。
address (city)	文字列	1024	N	このユーザーの物理的なメールアドレス。
address (state)	文字列	1024	N	このユーザーの物理的なメールアドレス。
address (country)	文字列	1024	N	このユーザーの物理的なメールアドレス。
address (postal_code)	文字列	1024	N	このユーザーの物理的なメールアドレス。
user_status	文字列	1024	N	ユーザーの管理ステータスを示す値

「Organization_Person」ファセット	データ型	長さ。	必須動作か否か	説明
タイトル	文字列	1024	N	組織のタイトル
preferred_language	文字列	1024	N	ユーザーが望んで書いた言語や話した言語を示し、ローカライズされたユーザーインターフェイスの選択に一般的に使用される。
employee_id	文字列	1024	N	個人に割り当てられた文字列識別子 (通常は数値または英数字)。
cost_center	整数	1024	N	コストセンターを識別
department	文字列	1024	N	部門の名前を指定する
manager	文字列	1024	N	ユーザーのマネージャー
company_name	文字列	1024	N	組織の名前を特定する
company_address (street1)	文字列	1024	N	組織の物理的郵送先住所
company_address (street2)	文字列	1024	N	組織の物理的郵送先住所
company_address (city)	文字列	1024	N	組織の物理的郵送先住所
company_address (state)	文字列	1024	N	組織の物理的郵送先住所
company_address (country)	文字列	1024	N	組織の物理的郵送先住所
company_address (postalCode)	文字列	1024	N	組織の物理的郵送先住所

Device

次の表は、デバイスのサンプルスキーマに含まれるファセットを示しています。

「Device」ファセット	データ型	長さ。	必須動作か否か	説明
device_id	文字列	1024	N	英数字の一意のデバイス ID
name	文字列	1024	N	デバイスのわかりやすい名前
description	文字列	1024	N	デバイスの説明
X.509_certificates	文字列	1024	N	X.509 証明書
device_version	文字列	1024	N	デバイスのバージョン
device_os_type	文字列	1024	N	デバイスのオペレーティングシステム
device_os_version	文字列	1024	N	デバイスのオペレーティングシステムのバージョン番号
serial_number	文字列	1024	N	デバイスのシリアル番号
device_status	文字列	1024	N	デバイスのステータス (アクティブ、非アクティブ、中断、シャットダウン、オフなど)

カスタムスキーマ

カスタムスキーマを作成するための最初のステップは、どのフィールドをインデックスする必要があるかを正確に定義することです。これらの必須フィールドは、独自のフィールドを追加するスキーマ

のスケルトン要素を構成します。各フィールドの名前と型 (文字列、整数、ブールなど) をオブジェクトの構造にマップします。タイプと制約を使用してスキーマを定義し、それをディレクトリに適用することができます。Once defined, Cloud Directory performs validation for attributes.

詳細については、「スキーマの作成」を参照してください。

属性リファレンス

Amazon Cloud Directory ファセットには属性が含まれています。属性は、属性定義または属性参照のいずれかです。属性定義は、その名前とを宣言する属性とプリミティブ型 (文字列、バイナリ、ブール、DateTime、または数字) を宣言する属性です。また、オプションで必要な動作、デフォルト値、変更不可能なフラグ、および属性ルール (最小 / 最大長さなど) を宣言できます。

属性参照は、プリミティブ型、デフォルト値、変更不可能なフラグおよび別の既存の属性定義の属性ルールから取得される属性です。属性参照には、そのプロパティがターゲット属性定義から取得されるため、独自のプリミティブ型、デフォルト値、変更不可能なフラグまたはルールがありません。

属性参照は必要なターゲットの定義の動作を上書きする場合があります (詳細については以下を参照)。

属性リファレンスを作成する場合は、属性名と (ターゲットの属性定義のファセット名と属性名を含む) ターゲットの属性定義のみを指定します。属性リファレンスは、他の属性を参照しない場合があります。また、現時点では、属性リファレンスは別のスキーマからターゲットの属性定義は参照しない場合があります。

オブジェクトの 2 つ以上の属性に同じストレージの場所を参照させる場合に属性リファレンスを使用できます。たとえば、ユーザーのファセットと EnterpriseUser のファセットが適用されたオブジェクトがあるとします。ユーザーのファセットには、FirstName 属性定義があり、EnterpriseUser のファセットには User.FirstName をポイントする属性リファレンスがあります。両方の FirstName 属性が、同じストレージの場所を参照しているため、User.FirstName または EnterpriseUser.FirstName のいずれかに変更を加えると同じ結果が得られます。

API の例

次の例では、クラウドディレクトリ API を使用した属性リファレンスの使用方法について説明します。この例では、ベースのファセットには属性定義が含まれ、別のファセットには、ベースのファセットの属性を参照する属性が含まれています。ベースの属性が不要の場合は、リファレンス属性は必須とマークされることに注意してください。

```
   // create base facet
   CreateFacetRequest req1 = new CreateFacetRequest()
     .withSchemaArn(devSchemaArn)
     .withName("baseFacet")
     .withAttributes(List(
       new FacetAttribute()
         .withName("baseAttr")
         .withRequiredBehavior(RequiredAttributeBehavior.NOT_REQUIRED)
         .withAttributeDefinition(new
FacetAttributeDefinition().withType(FacetAttributeType.STRING))))
     .withObjectType(ObjectType.DIRECTORY)
   cloudDirectoryClient.createFacet(req1)

   // create another facet that refers to the base facet
   CreateFacetRequest req2 = new CreateFacetRequest()
     .withSchemaArn(devSchemaArn)
     .withName("facetA")
     .withAttributes(List(
       new FacetAttribute()
         .withName("ref")
         .withRequiredBehavior(RequiredAttributeBehavior.REQUIRED_ALWAYS)
         .withAttributeReference(new FacetAttributeReference()
           .withTargetFacetName("baseFacet")
           .withTargetAttributeName("baseAttr"))))
     .withObjectType(ObjectType.DIRECTORY)
   cloudDirectoryClient.createFacet(req2)
```

JSON 例:

次の例では、JSON モデルでの属性リファレンスの使用方法を示します。このモデルで表されるスキーマは上記のモデルと同じです。

```
{
 "facets" : {
   "baseFacet" : {
     "facetAttributes" : {
       "baseAttr" : {
         "attributeDefinition" : {
           "attributeType" : "STRING"
         },
         "requiredBehavior" : "NOT_REQUIRED"
       }
```

```
      },
      "objectType" : "DIRECTORY"
    },
    "facetA" : {
      "facetAttributes" : {
        "ref" : {
          "attributeReference" : {
            "targetFacetName" : "baseFacet",
            "targetAttributeName" : "baseAttr"
          },
          "requiredBehavior" : "REQUIRED_ALWAYS"
        }
      },
      " objectType" : "DIRECTORY"
    }
}
```

属性リファレンスに関する考慮事項

属性リファレンスは、同じスキーマでは既存の属性定義をターゲットとしている必要があります。

- 属性リファレンスは、同じファセットまたは別のファセットで既存の属性定義をターゲットとする場合があります。
- 属性リファレンスは、他の属性をターゲットとしない場合があります。
- 別のファセットの属性リファレンスのターゲットである属性定義を含むファセットは、すべての参照が削除されるまで削除できません。

オブジェクトを作成するか、またはファセットを既存のオブジェクトに適用することによって、従来の属性定義を使用する場合と同じ方法で属性リファレンスを使用できます。

> **Note**
>
> リファレンスを使用してファセットを他のファセットへ適用できますが、ターゲットファセットを直接適用する必要はありません。ターゲットファセットが適用されない場合は、属性リファレンスの動作に変更はありません。(ファセットの他の属性をオブジェクトに存在させる場合は、ターゲットファセットのみを適用する必要があります)。

属性リファレンス値の設定

属性の値を変更する場合は、UpdateObjectAttributes API アクションを呼び出すことができます。定義またはそのオブジェクトの同じ定義に対するその他のリファレンスのいずれかを更新 (削除) すると同じ結果が得られます。

属性リファレンス値の取得

ListObjectAttributes API アクションを呼び出して、ストレージエイリアスを取得できます。この呼び出しは、それぞれが属性キーとそれに関連付けられている値が含まれる、タプルのリストを返します。属性キーは、そのオブジェクトに存在するストレージエイリアスのリストに対応しています。

> **Note**
>
> 属性キーはオブジェクトに明示的に適用されなかったファセットに返すことができます。これは属性リファレンスがオブジェクトに適用されていないファセットをターゲットにする場合に発生します。

たとえば、ユーザーファセットと EnterpriseUser ファセットがあるとします。この EnterpriseUser.FirstName 属性は、User.FirstName を参照します。ユーザーおよび EnterpriseUser ファセットの両方をオブジェクトに適用して、User.FirstName を Robert に設定し、EnterpriseUser.FirstName を Bob に設定します。ListObjectAttributes を呼び出すと、両方の FirstName 属性に 1 つのストレージエイリアスしかないため、「User.FirstName = Bob」のみが表示されます。

属性リファレンスを含むインデックスの使用

属性リファレンスではなく、属性定義を含むインデックスのみを作成できます。インデックスのリストは属性リファレンスの属性キーを返しません。ただし、インデックス化されたオブジェクトの既存のリファレンスが対象としている任意の属性定義には属性キーを返します。つまり、インデックスレイヤーで、属性リファレンスは、実行時の正しい属性定義識別子に解決される、属性の代替の識別子としてのみ扱われます。

たとえば、ファセットユーザー属性 FirstName のインデックスがあるとしましょう。EnterpriseUser ファセットのみが適用されたオブジェクトをアタッチします。次に、オブジェクトの EnterpriseUser.FirstName 属性の値を Bob に設定します。最後に、ListIndex アクションを呼び出します。結果には、「User.FirstName=Bob」のみが含まれます。

属性リファレンスの必須動作

属性リファレンスには、そのターゲット属性定義とは異なる必須動作があります。これにより、基本定義をオプションにし、同じ定義への参照を必須にすることができます。オブジェクトに基本定義と同じ基本定義に 1 つ以上の参照がある場合は、基本定義とすべての参照は関連するすべての属性に存在する最強の必須動作に準拠する必要があります。

- 属性定義と同様に、オブジェクトを作成、またはファセットを既存のオブジェクトにファセットを追加する際は、任意の必須属性定義の値を指定する必要があります。
- 利便性を高めるため、オブジェクトの複数の属性が、同じストレージの場所を参照している場合は、そのストレージの場所の属性のいずれかの値を指定します。
- 同様に、同じストレージの場所に複数の値を指定する場合は、値は同じである必要があります。

属性ルール

ルールは、属性タイプの許容値を記述し、特定の属性に許容される値を制約します。ファセットを作成するときは、属性定義の一部としてルールを指定する必要があります。Cloud Directory は、次のルールタイプをサポートしています。

- 文字列の長さ
- バイナリ長
- セットからの文字列
- 数値比較

文字列の長さ

文字列属性値の長さを制限します。

許可されたルールパラメータキー: min、max

許可されたルールパラメータ値: 数値

バイナリ長

バイナリ属性値のバイト配列の長さを制限します。

許可されたルールパラメータキー: min、max

許可されたルールパラメータ値: 数値

セットからの文字列

文字列属性の値を、指定された文字列のセットに制限します。

許可されたルールパラメータキー: 許可された値

許可されたルールパラメータ値: 各文字列が UTF-8 でエンコードされる文字列のセット

指定できる値はカンマ区切りで、引用符で囲むことができます。これは、許可された値にカンマが含まれている場合に便利です。例:

- One,two,three は、1、2、または 3 と一致
- “with,comma”,”withoutcomma” は、「カンマあり」または「カンマンなし」と一致
- with”quote,withoutquote は、「見積もりあり」または「見積もりなし」と一致

数値比較

数値属性に許容される数値を制約します。

許可されたルールパラメータキー: min、max

許可されたルールパラメータ値: 数値

仕様の形式

クラウドディレクトリスキーマは、データディレクトリ内のデータに構造を追加します。クラウドディレクトリには、スキーマを定義するための 2 つのメカニズムが用意されています。開発者は、特定の API 操作を使用してスキーマを構築したり、スキーマアップロード機能を使用してスキーマ全体をアップロードすることができます。スキーマドキュメントは、API コールまたはコンソール経由でアップロードできます。このセクションでは、スキーマドキュメント全体をアップロードするときに使用する形式について説明します。

JSON スキーマ形式

スキーマドキュメントは、次の JSON ドキュメントの全体的な形式です。

```
{
    "facets": {
        "facet name": {
            "facetAttributes": {
                "attribute name": Attribute JSON Subsection
```

```
            }
        }
    }
}
```

スキーマドキュメントには、ファセット名とファセットのマップが含まれています。各ファセットには、それぞれ属性を含むマップが含まれています。スキーマ内のすべてのファセット名は一意でなければなりません。ファセット内のすべての属性名は一意でなければなりません。

属性 JSON サブセクション

ファセットには属性が含まれています。各属性は、属性に格納できる値のタイプを定義します。次の JSON 形式で属性を記述します。

```
{
    "attributeDefinition": Attribute Definition Subsection,
    "attributeReference": Attribute Reference Subsection,
    "requiredBehavior": "REQUIRED_ALWAYS" or "NOT_REQUIRED"
}
```

属性定義または属性リファレンスのいずれかを指定する必要があります。それぞれの詳細については、関連するサブセクションを参照してください。

必須の動作フィールドは、この属性が必須かどうかを示します。このフィールドを指定する必要があります。指定できる値は次のとおりです。

- REQUIRED_ALWAYS: この属性は、オブジェクトの作成時またはファセットの追加時に指定する必要があります。この属性は削除できません。
- NOT_REQUIRED: この属性は、存在する場合もありますが、存在しない場合もあります。

属性定義サブセクション

属性は、属性値に関連する型とルールを定義します。次の JSON レイアウトでその形式について説明します。

```
{
    "attributeType": One of "STRING", "NUMBER", "BINARY", "BOOLEAN" or "DATETIME",
    "defaultValue": Default Value Subsection,
    "isImmutable": true or false,
    "attributeRules": "Attribute Rules Subsection"
```

```
}
```

デフォルト値サブセクション

次のデフォルト値のいずれかを正確に指定します。ロング値とブール値は、引用符の外側に (文字列ではなくそれぞれの Javascript 型として) 提供する必要があります。バイナリ値は、URL セーフ Base64 でエンコードされた文字列 (RFC 4648 で説明) を使用して提供されます。日時データ型は、エポック (1970 年 1 月 1 日 00:00:00 UTC) からの時間が秒単位で指定されます。

```
{
 "stringValue": "a string value",
 "longValue": an integer value,
 "booleanValue": true or false,
 "binaryValue": a URL-safe Base64 encoded string,
 "datetimeValue": an integer value representing milliseconds since epoch
}
```

属性ルールサブセクション

属性ルールは、属性値の制約を定義します。各属性に複数のルールを定義できます。属性ルールには、ルールタイプとそのルールのパラメータセットが含まれています。詳細については、「属性ルール」セクションを参照してください。

```
{
    "rule name": {
        "parameters": {
            "rule parameter key 1": "value",
            "rule parameter key 2": "value"
        },
        "ruleType": "rule type value"
    }
}
```

属性リファレンスサブセクション

属性リファレンスは、高度な機能です。属性リファレンスによって、複数のファセットは属性定義と格納された値を共有することができます。詳細については、「属性リファレンス」セクションを参照してください。次のテンプレートを使用して、JSON スキーマで属性参照を定義できます。

```
{
 "targetSchemaArn": "schema ARN"
```

```
 "targetFacetName": "facet name"
 "targetAttributeName": "attribute name"
}
```

スキーマドキュメントの例

以下は、有効な JSON 形式を示すスキーマドキュメントの例です。

> **Note**
>
> allowedValues 文字列で表すすべての値は、カンマで区切り、スペースなしにする必要があります。たとえば、"SENSITIVE,CONFIDENTIAL,PUBLIC" と指定します。

基本的なスキーマドキュメント

```
{
    "facets": {
        "Employee": {
            "facetAttributes": {
                "Name": {
                    "attributeDefinition": {
                        "attributeType": "STRING",
                        "isImmutable": false,
                        "attributeRules": {
                            "NameLengthRule": {
                                "parameters": {
                                    "min": "3",
                                    "max": "100"
                                },
                                "ruleType": "STRING_LENGTH"
                            }
                        }
                    },
                    "requiredBehavior": "REQUIRED_ALWAYS"
                },
                "EmailAddress": {
                    "attributeDefinition": {
                        "attributeType": "STRING",
                        "isImmutable": true,
                        "attributeRules": {
                            "EmailAddressLengthRule": {
```

```
                            "parameters": {
                                "min": "3",
                                "max": "100"
                            },
                            "ruleType": "STRING_LENGTH"
                        }
                    }
                },
                "requiredBehavior": "REQUIRED_ALWAYS"
            },
            "Status": {
                "attributeDefinition": {
                    "attributeType": "STRING",
                    "isImmutable": false,
                    "attributeRules": {
                        "rule1": {
                            "parameters": {
                                "allowedValues": "ACTIVE,INACTIVE,TERMINATED"
                            },
                            "ruleType": "STRING_FROM_SET"
                        }
                    }
                },
                "requiredBehavior": "REQUIRED_ALWAYS"
            }
        },
        "objectType": "LEAF_NODE"
    },
    "DataAccessPolicy": {
        "facetAttributes": {
            "AccessLevel": {
                "attributeDefinition": {
                    "attributeType": "STRING",
                    "isImmutable": true,
                    "attributeRules": {
                        "rule1": {
                            "parameters": {
                                "allowedValues": "SENSITIVE,CONFIDENTIAL,PUBLIC"
                            },
                            "ruleType": "STRING_FROM_SET"
                        }
                    }
                },
                "requiredBehavior": "REQUIRED_ALWAYS"
```

```
                }
            },
            "objectType": "POLICY"
        },
        "Group": {
            "facetAttributes": {
                "Name": {
                    "attributeDefinition": {
                        "attributeType": "STRING",
                        "isImmutable": true
                    },
                    "requiredBehavior": "REQUIRED_ALWAYS"
                }
            },
            "objectType": "NODE"
        }
    }
}
```

スキーマドキュメントと型付きリンク

```
{
    "sourceSchemaArn": "",
    "facets": {
        "employee_facet": {
            "facetAttributes": {
                "employee_login": {
                    "attributeDefinition": {
                        "attributeType": "STRING",
                        "isImmutable": true,
                        "attributeRules": {}
                    },
                    "requiredBehavior": "REQUIRED_ALWAYS"
                },
                "employee_id": {
                    "attributeDefinition": {
                        "attributeType": "STRING",
                        "isImmutable": true,
                        "attributeRules": {}
                    },
                    "requiredBehavior": "REQUIRED_ALWAYS"
                },
                "employee_name": {
```

```
                    "attributeDefinition": {
                        "attributeType": "STRING",
                        "isImmutable": true,
                        "attributeRules": {}
                    },
                    "requiredBehavior": "REQUIRED_ALWAYS"
                },
                "employee_role": {
                    "attributeDefinition": {
                        "attributeType": "STRING",
                        "isImmutable": true,
                        "attributeRules": {}
                    },
                    "requiredBehavior": "REQUIRED_ALWAYS"
                }
            },
            "objectType": "LEAF_NODE"
        },
        "device_facet": {
            "facetAttributes": {
                "device_id": {
                    "attributeDefinition": {
                        "attributeType": "STRING",
                        "isImmutable": true,
                        "attributeRules": {}
                    },
                    "requiredBehavior": "REQUIRED_ALWAYS"
                },
                "device_type": {
                    "attributeDefinition": {
                        "attributeType": "STRING",
                        "isImmutable": true,
                        "attributeRules": {}
                    },
                    "requiredBehavior": "REQUIRED_ALWAYS"
                }
            },
            "objectType": "NODE"
        },
        "region_facet": {
            "facetAttributes": {},
            "objectType": "NODE"
        },
        "group_facet": {
```

```
            "facetAttributes": {
                "group_type": {
                    "attributeDefinition": {
                        "attributeType": "STRING",
                        "isImmutable": true,
                        "attributeRules": {}
                    },
                    "requiredBehavior": "REQUIRED_ALWAYS"
                }
            },
            "objectType": "NODE"
        },
        "office_facet": {
            "facetAttributes": {
                "office_id": {
                    "attributeDefinition": {
                        "attributeType": "STRING",
                        "isImmutable": true,
                        "attributeRules": {}
                    },
                    "requiredBehavior": "REQUIRED_ALWAYS"
                },
                "office_type": {
                    "attributeDefinition": {
                        "attributeType": "STRING",
                        "isImmutable": true,
                        "attributeRules": {}
                    },
                    "requiredBehavior": "REQUIRED_ALWAYS"
                },
                "office_location": {
                    "attributeDefinition": {
                        "attributeType": "STRING",
                        "isImmutable": true,
                        "attributeRules": {}
                    },
                    "requiredBehavior": "REQUIRED_ALWAYS"
                }
            },
            "objectType": "NODE"
        }
    },
    "typedLinkFacets": {
        "device_association": {
```

```
            "facetAttributes": {
                "device_type": {
                    "attributeDefinition": {
                        "attributeType": "STRING",
                        "isImmutable": false,
                        "attributeRules": {}
                    },
                    "requiredBehavior": "REQUIRED_ALWAYS"
                },
                "device_label": {
                    "attributeDefinition": {
                        "attributeType": "STRING",
                        "isImmutable": false,
                        "attributeRules": {}
                    },
                    "requiredBehavior": "REQUIRED_ALWAYS"
                }
            },
            "identityAttributeOrder": [
                "device_label",
                "device_type"
            ]
        }
    }
}
```

ディレクトリオブジェクト

開発者は、拡張可能なスキーマを使用してディレクトリオブジェクトを設計し、データの正確性に関する制約を強化するため、プログラミングしやすくなります。Amazon Cloud Directory では、定義済みのインデックス付き属性に基づく豊富な情報ルックアップを使用しているため、ディレクトリツリー内の高速なツリートラバーサルおよび検索が可能になります。クラウドディレクトリのデータは、不使用時と転送中のいずれも暗号化されます。

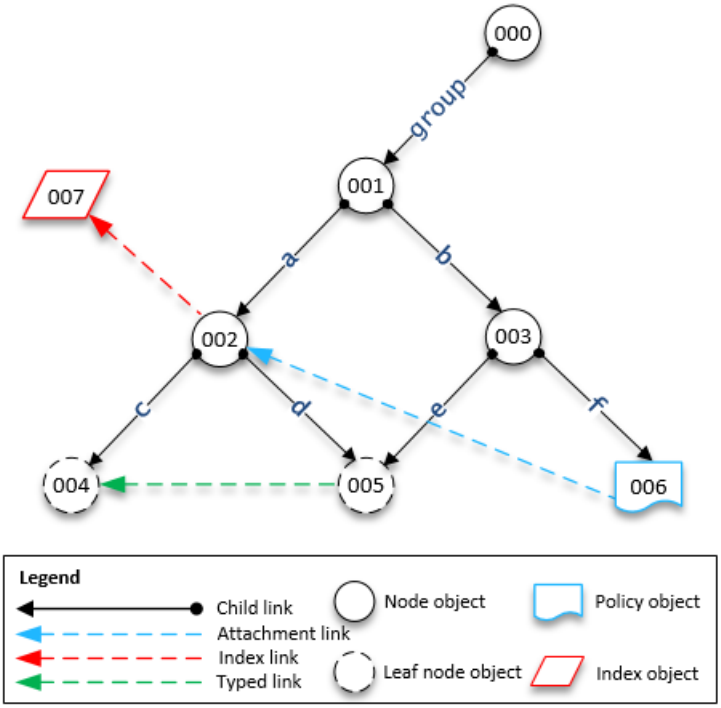
オブジェクトは、クラウドディレクトリの基本的な要素です。各オブジェクトには、グローバルに一意な識別子が割り当てられているため、このオブジェクト識別子を使用して指定することができます。オブジェクトは、属性キーや属性値を含むゼロ以上のファセットの集合です。また、オブジェクトは、適用済みの単一スキーマ内の 1 つ以上のファセット、または適用済みの複数のスキーマのファセットから作成することができます。オブジェクト作成時、必要な属性値をすべて指定する必要があります。オブジェクトは限定数のファセットを持つことができます。詳細については、「Amazon Cloud Directory の制限」を参照してください。

オブジェクトは、通常のオブジェクト、ポリシーオブジェクト、またはインデックスオブジェクトです。また、ノードオブジェクトまたはリーフノードオブジェクトのいずれかになる場合もあります。オブジェクトタイプは、オブジェクトにアタッチされているファセットのオブジェクトタイプから推定されます。

トピック

Links

リンクは、関係を定義する 2 つのオブジェクト間の有向辺を表します。現在、Cloud Directory をサポートしています。



子リンク

子リンクは、接続するオブジェクト間の親子関係を作成します。たとえば、上の図で子リンク b は、オブジェクト 001 とオブジェクト 003 を接続しています。子リンクは、クラウドディレクトリの階層を定義します。子リンクには、リンクで指定するオブジェクトのパスを定義する際の名前があります。

アタッチメントリンク

アタッチメントリンクは、リーフノードポリシーオブジェクトを別のリーフノードまたはノードオブジェクトに適用します。アタッチメントリンクでは、Cloud Directory 階層構造を定義することはできません。たとえば、上の図のアタッチメントリンクでは、ポリシーリーフノードオブジェクト 006 に保存されているポリシーを、ノードオブジェクト 002 に適用します。各オブジェクトには、複数のポリシーをアタッチできますが、指定されたポリシータイプのポリシーを複数アタッチすることはできません。

インデックスリンク

インデックスリンクでは、インデックスオブジェクトおよび定義済みのインデックス付き属性に基づく豊富な情報ルックアップを採用しているため、ディレクトリツリー内の高速なツリートラバーサルおよび検索が可能になります。概念的には、インデックスは子を持つノードに似ています。インデックス付きノードへのリンクは、子をアタッチしたときにラベル付けされるのではなく、インデックス

付き属性に従ってラベル付けされます。ただし、インデックスリンクは親子エッジではなく、独自の列挙 API オペレーションのセットを持っています。詳細については、「インデックス作成と検索」を参照してください。

型付きリンク

型付きリンクを使用すると、Cloud Directory の同じ階層のオブジェクト間、または異なる階層のオブジェクト間の関係を確立できます。これらの関係を使用して、どのユーザーがデバイス "xyz" を所有しているか、どのデバイスがユーザー "abc" によって所有されているかなどの情報を照会できます。

型付きリンクを使用して、ディレクトリ内の異なるオブジェクト間の関係をモデル化できます。たとえば、上の図で、ユーザーを表すオブジェクト `004` と、デバイスを表すオブジェクト `005` との関係を考えます。

この場合、型付きリンクを使用して両オブジェクト間の所有関係をモデル化できます。型付きリンクに属性を追加して、購入コストを表したり、デバイスをレンタルしたか購入したかを表したりできます。型付きリンクに関連付けられる属性には、以下の 2 種類があります。

- アイデンティティベースの属性 - 型付きリンクを他のリンク (子、アタッチメント、インデックスリンクなど) と区別するための属性。型付きリンクの各ファセットで、順序付けられたアイデンティティ属性のセットを定義します。型付きリンクのアイデンティティは、ソースオブジェクト ID、ファセット識別子 (型)、そのリンクのアイデンティティ属性の値 (ファセットで定義)、ターゲットオブジェクト ID です。識別子は 1 つのディレクトリ内で一意であることが必要です。
- オプションの属性 - 型付きリンクのアイデンティティとは関係のない追跡特性を保存する属性。たとえば、オプションの属性により、型付きリンクが最初に確立された日付や最後に変更された日付を識別します。

オブジェクトと同じように、型付きリンクのファセットを作成するには、`CreateTypedLinkFacet` API を使用して型付きリンクの構造とその属性を定義する必要があります。型付きリンクのファセットでは、一意のファセット名と属性セットをリンクに関連付ける必要があります。型付きリンクの構造を設計するときに、型付きリンクのファセットで順序付けられた属性のセットを定義できます。型付きリンクのサンプルスキーマについては、「スキーマドキュメントと型付きリンク」を参照してください。

型付きリンクの属性は、以下のいずれかの操作が必要なときに使用できます。

- 受信または送信方向の型付きリンクのフィルタ処理を行う。詳細については、「型付きリンクのリスティング」を参照してください。

- 2 つのオブジェクト間の関係を表す。
- 型付きリンクに関する管理データ (リンクの作成日など) を追跡する。

型付きリンクがユースケースに適切かどうかは、以下の点を考慮して判断します。

- 型付きリンクは、パスベースのオブジェクト指定では使用できません。代わりに、型付きリンクは `ListOutgoingTypedLinks` API オペレーションまたは `ListIncomingTypedLinks` API オペレーションを使用して選択する必要があります。
- 型付きリンクは、`LookupPolicy` API オペレーションまたは `ListObjectParentPaths` API オペレーションには関与しません。
- 同じ 2 つのオブジェクト間の同じ方向の型付きリンクは、同じ属性値を持つことはできません。これにより、同じオブジェクト間の重複する型付きリンクを回避できます。
- オプションの情報を追加する場合は、追加の属性を使用できます。
- すべての ID 属性値の合計サイズは 64 バイトに制限されます。詳細については、「Amazon Cloud Directory の制限」を参照してください。

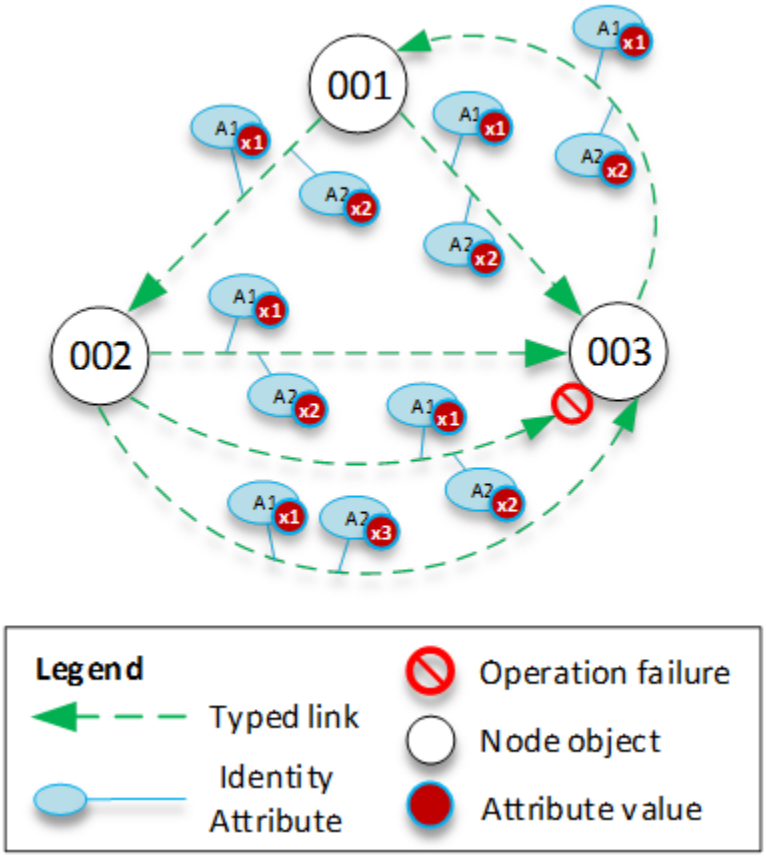
Cloud Directory 関連ブログ記事

- Amazon Cloud Directory 型付きリンクを使用して階層間の関係を作成および検索する

型付きリンクのアイデンティティ

アイデンティティは、2 つのオブジェクト間で型付きリンクが存在できるかどうかを一意に定義します。例外は、2 つのオブジェクトを同じ属性値で同じ方向に接続する場合です。属性は `REQUIRED_ALWAYS` として設定する必要があります。

異なる型付きリンクのファセットから作成された型付きリンクが相互に競合することはありません。たとえば、次の図について考えます。



- オブジェクト 001 の 2 つの型付きリンクは、それぞれに属性 (A1 と A2) があり、同じ属性値 (x1 と x2) で、異なるオブジェクト (002 と 003) に向かっています。このオペレーションは成功します。
- オブジェクト 002 とオブジェクト 003 の間には型付きリンクがあります。同じ属性値で同じ方向の 2 つの型付きリンクはオブジェクト間に存在できないため、このオペレーションは失敗します。
- オブジェクト 001 とオブジェクト 003 の間には、同じ属性を持つ 2 つの型付きリンクがあります。ただし、2 つのリンクは方向が異なるため、このオペレーションは成功します。
- オブジェクト 002 とオブジェクト 003 の間には、A1 の値が同じでも A2 の値が異なる型付きリンクがあります。型付きリンクのアイデンティティでは、すべての属性が考慮されるため、このオペレーションは成功します。

型付きリンクのルール

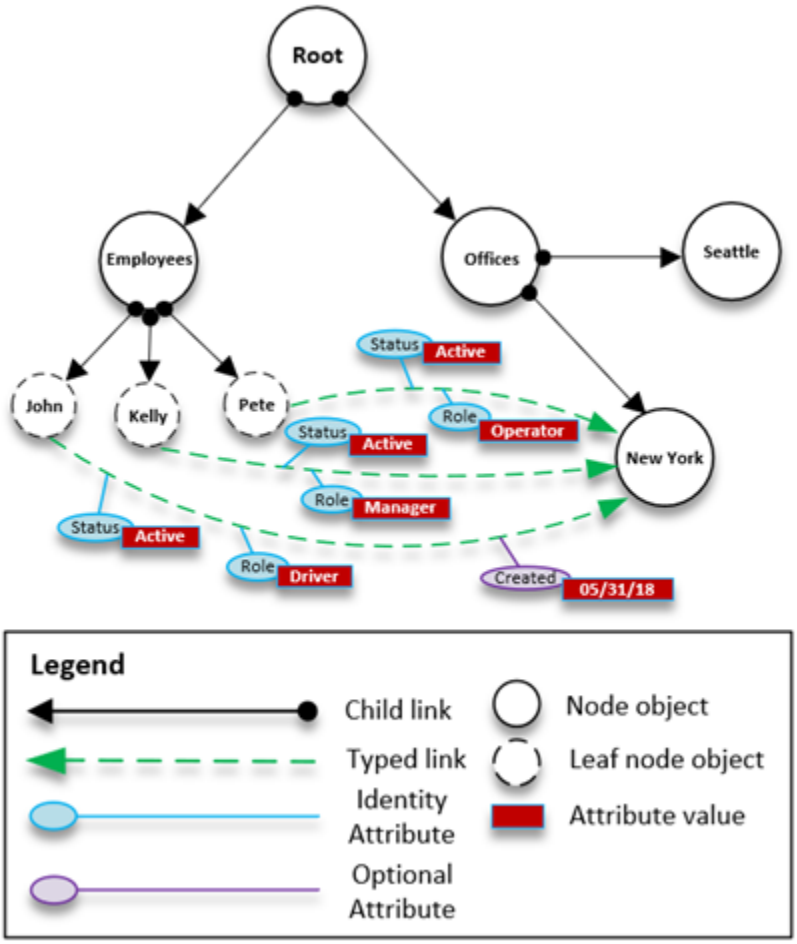
リンクの属性に制限を追加する場合、型付きリンクの属性にルールを追加できます。これらのルールは、オブジェクトの属性に対するルールに相当します。詳細については、「属性ルール」を参照してください。

型付きリンクのリスティング

Cloud Directory の API オペレーションを使用して、オブジェクトの発着信する型付きリンクを選択できます。すべての型付きリンクを反復処理する代わりに、型付きリンクの特定のサブセットを選択できます。特定の型付きリンクのファセットを指定することで、そのタイプの型付きリンクのみをフィルタすることもできます。

型付きリンクのファセットに定義された属性の順序に従って型付きリンクをフィルタできます。複数の属性に対して範囲フィルタを指定できます。選択した型付きリンクに範囲を指定する場合、不正確な範囲は最後に指定する必要があります。範囲が指定されていない属性は、範囲全体にマッチするとみなされます。フィルタは、API コールに提供された順序ではなく、型付きリンクのファセットに定義された属性の順序で解釈されます。

たとえば、次の図で、従業員とその能力に関する情報を保存するための Cloud Directory を考えます。



たとえば、従業員の能力を EmployeeCapability という型付きリンクでモデル化し、Status、Role、Created という 3 つの文字列属性を設定するとします。以下のフィルタは、ListIncomingTypedLinks API オペレーションと ListOutgoingTypedLinks API オペレーションでサポートされます。

- ファセット =EmployeeCapability、ステータス =Active、ロール =Driver
  - 運転手であるアクティブな従業員を選択します。このフィルタには 2 つの完全一致が含まれます。
- Facet (ファセット) = EmployeeCapability、Status (ステータス) = Active、Role (ロール) = Driver、Created (作成日) = 05/31/18
  - 運転手であり、ファセットが 2018 年 5 月 31 日以降に作成された、アクティブな従業員を選択します。
- ファセット = EmployeeCapability、ステータス = Active
  - すべてのアクティブな従業員を選択します。
- ファセット = EmployeeCapability、ステータス = Active、ロール = A〜M

  - ロールが A〜M で始まるアクティブな従業員を選択します。
- ファセット = EmployeeCapability
  - これは、EmployeeCapability タイプのすべての型付きリンクを選択します。

以下のフィルタはサポート されません。

- ファセット = EmployeeCapability、A〜C のステータス、ロール = Driver
  - 範囲はフィルタの最後に表示される必要があるため、このフィルタは許可されません。
- ファセット =EmployeeCapability、Role=Driver
  - 暗黙のステータス範囲は完全一致ではなく、範囲のリストの最後に表示されることがないため、このフィルタは許可されません。
- Status = Active
  - 型付きリンクのファセットが指定されていないため、このフィルタは許可されません。

型付きリンクのスキーマ

型付きリンクのファセットは、2 つの方法で作成できます。型付きリンクのファセットは、個別の API コール (CreateTypedLinkFacet、DeleteTypedLinkFacet、UpdateTypedLinkFacet など) から管理できます。スキーマを JSON ドキュメントとして 1 回の PutSchemaFromJson API コールでアップロードすることもできます。詳細については、「JSON スキーマ形式」を参照してください。型付きリンクのサンプルスキーマについては、「スキーマドキュメントと型付きリンク」を参照してください。

スキーマ開発ライフサイクルの各段階で許可される変更の種類は、オブジェクトのファセットの操作で許可される変更に似ています。開発状態のスキーマでは、あらゆる変更がサポートされます。発行済み状態のスキーマは変更不可であり、一切の変更がサポートされません。データディレクトリに適用されるスキーマに対する特定の変更のみ許可されます。適用済みの型付きリンクのファセットに順序と属性を設定すると、その順序は変更できません。

ファセットとその属性をリストする他の 2 つの API オペレーションは以下のとおりです。

- ListTypedLinkFacetAttributes
- ListTypedLinkFacetNames

型付きリンクの操作

型付きリンクのファセットの作成が完了すると、型付きリンクの作成と操作を開始できます。型付きリンクをアタッチまたはデタッチするには、AttachTypedLink API オペレーションと DetachTypedLink API オペレーションを使用します。

TypedLinkSpecifier は、型付きリンクを一意に識別するすべての情報を含む構造です。この構造内に TypedLinkFacet、SourceObjectID、DestinationObjectID、および IdentityAttributeValues があります。これらを使用して、操作対象の型付きリンクを一意に指定します。AttachTypedLink API オペレーションは、型付きリンク指定子を返し、DetachTypedLink API オペレーションは型付きリンク指定子を入力として受け取ります。同様に、ListIncomingTypedLinks API オペレーションと ListOutgoingTypedLinks API オペレーションは、型付きリンク指定子を出力として渡します。型付きリンク指定子は、最初から作成することもできます。型付きリンクに関する API オペレーションの完全な一覧は以下のとおりです。

- AttachTypedLink
- CreateTypedLinkFacet
- DeleteTypedLinkFacet
- DetachTypedLink
- GetLinkAttributes
- GetTypedLinkFacetInformation
- ListIncomingTypedLinks
- ListOutgoingTypedLinks
- ListTypedLinkFacetNames
- ListTypedLinkFacetAttributes
- UpdateLinkAttributes
- UpdateTypedLinkFacet

> **Note**
>
> 属性の参照と型付きリンクの更新はサポートされていません。型付きリンクを更新するには、それを削除した上で更新したバージョンを追加する必要があります。

範囲フィルタ

のいくつかのCloud Directory リスト API では、フィルタを範囲形式で指定できます。これらのフィルタを使うと、指定したノードにアタッチされたリンクのサブセットを効率的に選択できます。

通常、範囲はマップ (キー/値のペアの配列) として指定します。キーは属性識別子、値は対応する範囲です。これにより、アイデンティティが 1 つ以上の属性で構成されているリンクをフィルタリングできます。たとえば、ロール関係をモデル化してアクセス権限を決定する TypedLink セットアップは、RoleType 属性と Authorizer 属性の両方を持つ場合があります。この場合、`ListOutgoingTypedLinks` 呼び出しは、範囲を指定することで、結果を RoleType:”Admin” と Authorizer:”Julia” でフィルタリングできます。1 つのリストリクエストのフィルタリングに使用する範囲のマップには、リンクのアイデンティティ (インデックスの OrderedIndexedAttributeList または TypedLink の IdentityAttributeOrder) を定義する属性のみを含める必要があり、すべての属性の範囲を含める必要はありません。欠落した範囲は、FIRST から LAST までのすべての該当する値にまたがる範囲で自動的に満たされます。

各属性は独立したフラットな値のドメインを定義するものと考えると、範囲構造は、そのドメインの 2 つの論理ポイント (スタートポイントとエンドポイント) を定義し、範囲は 2 つの論理ポイント間のすべての該当するポイントにマッチします。範囲構造の StartValue と EndValue では、これら 2 つのポイントの基礎を定義し、さらに「モード」により適用範囲を絞り込むとで、各ポイント自体を範囲に含めるか範囲から除外するかを指定します。上の例の RoleType:”Admin” の場合、RoleType 属性の値は両方とも “Admin” で、モードは両方とも “INCLUSIVE” です ([“Admin” to “Admin”] と表記されます)。ListIndex 呼び出しのフィルタ (インデックスを User ファセットの LastName で定義する場合) では、「StartValue=”D”, StartMode=INCLUSIVE, EndValue:”G”, EndMode:EXCLUSIVE」を使用してリストを絞り込み、D、E、または F で始まる名前を抽出できます。

範囲のスタートポイントは、常にエンドポイント以前にする必要があります。EndValue が StartValue に先行している場合は、Cloud Directory からエラーが返されます。また、値はフィルタリングする属性と同じプリミティブ型にする必要があります。つまり、文字列属性には文字列値、整数属性には整数などを使います。たとえば、「StartValue=“D”, StartMode=EXCLUSIVE, EndValue=“D”, EndMode=INCLUSIVE」は無効です。エンドポイントに含まれる値がスタートポイントより先行しているためです。

スタートポイントまたはエンドポイントで使用できる 3 つの特殊なモードがあります。以下のモードでは、位置が明白であるため、対応する値フィールドを指定する必要がありません。

- FIRST - ドメインのすべての該当する値に先行します。これをスタートポイントに使用すると、ドメインの先頭からエンドポイントまでのすべての該当する値にマッチします。これをエンドポイントに使用すると、ドメインのいずれの値も範囲にマッチしません。

- LAST - ドメインのすべての該当する値の後になります。これをエンドポイントに使用すると、スタートポイントに続くすべての該当する値にマッチします (欠落した値も含まれます)。これをスタートポイントに使用すると、ドメインのいずれの値も範囲にマッチしません。
- LAST_BEFORE_MISSING_VALUES - このモードは、オプションの属性で値が欠落している場合にのみ役立ちます (「欠落した値」を参照してください)。これは欠落した値と実際のドメイン値の間のポイントに対応します。これをエンドポイントに使用すると、スタートポイントに続くすべての欠落していないドメイン値にマッチします。これをスタートポイントに使用すると、すべての欠落していないドメイン値が除外されます。属性が必須である場合、欠落した値は存在しないため、このモードは LAST と同じになります。

複数の範囲の制限

複数の属性がある場合は、低レイテンシーで効率的なリクエスト処理を保証するために、Cloud Directory でパターンが制限されます。複数の識別属性を持つ各リンクは、明確に定義された順序で属性を指定します。たとえば、上の例の Role では、RoleType 属性を重要性が最上位、Authorizer 属性を最下位として定義します。リストリクエストで指定できるのは単一の "識別" 範囲であり、その条件として 1) 単一の値でないこと、2) すべての該当する値にまたがる範囲でないことの両方が求められます (この 2 つの条件に該当する範囲は複数存在する場合があります)。識別範囲属性より重要性が高い属性は、範囲として単一の値を指定し、重要性がより低い属性の範囲はすべての該当する値にまたがる必要があります。Role の例の場合、フィルタセット (RoleType:”Admin”, Authorizer:[“J” to “L”]) (単一の値 + 識別範囲)、(RoleType:[”Admin” to “User”]) (識別範囲 + 暗黙のまたがる範囲)、および (RoleType:[FIRST to LAST]) (2 つのまたがる範囲、1 つの暗黙の範囲) のすべてが有効なフィルタセットの例です。(RoleType:[FIRST to LAST], Authorizer:”Julia”) は有効なセットではありません。またがる範囲が単一の値の範囲より重要性が高くなっているためです。

範囲構造を満たす場合に役立つパターンは以下のとおりです。

単一の値のマッチング

StartValue と EndValue の両方の値を指定し、両方のモードを「INCLUSIVE」に設定します。

例: `StartValue=“Admin”, StartMode=INCLUSIVE, EndValue=“Admin”, EndMode=INCLUSIVE`

プレフィックスのマッチング

プレフィックスを StartValue (INCLUSIVE モード) として指定し、プレフィックスに続く最初の値を EndValue (EXCLUSIVE モード) として指定します。

例: `StartValue="Jo", StartMode=INCLUSIVE, EndValue="Jp", EndMode=EXCLUSIVE ("p" is the next character value after "o")`

値を超える範囲のフィルタリング

StartValue (EXCLUSIVE モード) の値を指定し、LAST を EndMode (または、欠落した値を除外する場合は LAST_BEFORE_MISSING_VALUES) として指定します。

例: `StartValue=127, StartMode=EXCLUSIVE, EndValue=null, EndMode=LAST`

値以下の範囲のフィルタリング

EndValue (INCLUSIVE モード) の値を指定し、FIRST を StartMode として指定します。

欠落した値

(オプション) スキーマで属性を「Optional」としてマークすると、ファセットをアタッチする際に値を指定する必要がないため、または属性が後で削除されている場合があるため、値が「欠落」していることがあります。値が欠落しているオブジェクトをインデックスにアタッチすると、インデックスリンクは、まだ存在している場合でも、リンクセットの最後に移動されます。`ListIndex` 呼び出しは、最初にインデックス付き属性がすべて存在するリンクを返してから、1 つ以上の属性が欠落しているリンクを返します。これは、リレーショナルデータベースの NULL 値とほぼ同じです (これらの値の順番は、NULL 以外の値の後になります)。これらの欠落した値を範囲に含めるかどうかは、LAST モードまたは LAST_BEFORE_MISSING_VALUES モードを選択して指定できます。たとえば、ListIndex 呼び出しにフィルタを指定し、範囲 [LAST_BEFORE_MISSING_VALUES to LAST] でフィルタリングすることで、インデックス内の欠落した値のみを返させることができます。

オブジェクトへのアクセス

ディレクトリのオブジェクトは、パス、または `objectIdentifier` を使用してアクセスできます。

パス— Cloud Directory ツリー内の各オブジェクトは、到達する方法を示すパス名を使用して特定し、表示することができます。パスの開始地点は、ディレクトリのルート (前述の図のノード `000`) です。パス表記は、スラッシュ (/) でラベル付けしたリンクから開始し、パスの末尾まで、パスの区切り文字 (スラッシュを含む) で区切られた子リンクが続きます。たとえば、前に示した図のオブジェクト `005` は、パス `/group/a/d` を使用して特定できます。オブジェクトがリーフノードの場合は複数の親が存在するため、複数のパスでオブジェクトを識別する場合があります。また、以下のパスを使用して、オブジェクト `005` : /group/b/e を識別することもできます。

オブジェクト識別子— ディレクトリの各オブジェクトには、一意のグローバル識別子 () が割り当てられています。ObjectIdentifier。ObjectIdentifier「」の一部として、が返ります。CreateObjectAPI コールを使用します。また、「ObjectIdentifier」の API コールを使用して、GetObjectInformation を取得することもできます。たとえば、オブジェクト 005 のオブジェクト識別子を取得するには、オブジェクトに続くパスとしてオブジェクト参照 (group/b/e または group/a/d) を使用して、GetObjectInformation を呼び出します。

```
GetObjectInformationRequest request = new GetObjectInformationRequest()
    .withDirectoryArn(directoryArn)
    .withObjectReference("/group/b/e")
    .withConsistencyLevel(level)
GetObjectInformationResult result = cdClient.getObjectInformation(request)
String objectIdentifier = result.getObjectIdentifier()
```

オブジェクトの追加

新しいファセットをオブジェクトに追加するには、「AddFacetToObject」の API コールを使用します。オブジェクトタイプは、オブジェクトにアタッチされたファセットに基づいて判断されます。ディレクトリのオブジェクトアタッチメントは、オブジェクトタイプに基づいて使用できます。オブジェクトをアタッチするときは、次のルールに注意してください。

- リーフノードに子オブジェクトを含めることはできません。
- ノードオブジェクトは、子ノードを複数持つことができます。
- ポリシータイプのオブジェクトは、子オブジェクトを持つことができないため、親オブジェクトは 0 または 1 になります。

オブジェクトの更新

オブジェクトは、複数の方法で更新できます。

1. 「UpdateObjectAttributes」オペレーションを使用して、オブジェクトの各ファセット属性を更新します。
2. 「AddFacetToObject」オペレーションを使用して、新しいファセットをオブジェクトに追加します。
3. 「RemoveFacetFromObject」オペレーションを使用して、既存のファセットをオブジェクトから削除します。

オブジェクトの削除

アタッチされたオブジェクトをディレクトリから削除するには、一定の条件を満たしている必要があります。

1. ツリーからオブジェクトをデタッチします。オブジェクトは、子オブジェクトが含まれていない場合のみ、デタッチすることができます。オブジェクトに子オブジェクトが含まれている場合は、まず子オブジェクトをすべてデタッチする必要があります。
2. デタッチされたオブジェクトは、そのオブジェクトの属性がすべて削除されている場合のみ削除できます。オブジェクトの属性を削除するには、そのオブジェクトにアタッチされた各ファセットを削除します。オブジェクトにアタッチされたファセットの一覧を取得するには、`GetObjectInformation` を呼び出します。
3. オブジェクトに親、ポリシーアタッチメント、インデックスアタッチメントがないことを確認します。

オブジェクトは削除するツリーから完全にデタッチされているため、オブジェクト識別子を使用して削除する必要があります。

オブジェクトのクエリ

このセクションでは、ディレクトリ内のオブジェクトのクエリに関連するさまざまな要素について説明します。

ディレクトリのトラバーサル

Cloud Directory はツリー構造になっているため、上部から、オブジェクトのクエリを行うことができます。`ListObjectChildren`API 操作を使用するか、`ListObjectParents`API オペレーション。

ポリシーの検索

オブジェクトリファレンスを指定すると、「`LookupPolicy`」 API オペレーションでルートへのパスに沿ってアタッチされているすべてのポリシーが上位から返ります。ルートに到達しないパスはすべて無視されます。ポリシータイプのオブジェクトがすべて返ります。

オブジェクトがリーフノードの場合は、複数のパスをルートに含めることができます。この呼び出しでは、各呼び出しのパスが 1 つのみ返ります。追加のパスを取得するには、ページ分割トークンを使用します。

インデックスのクエリ実行

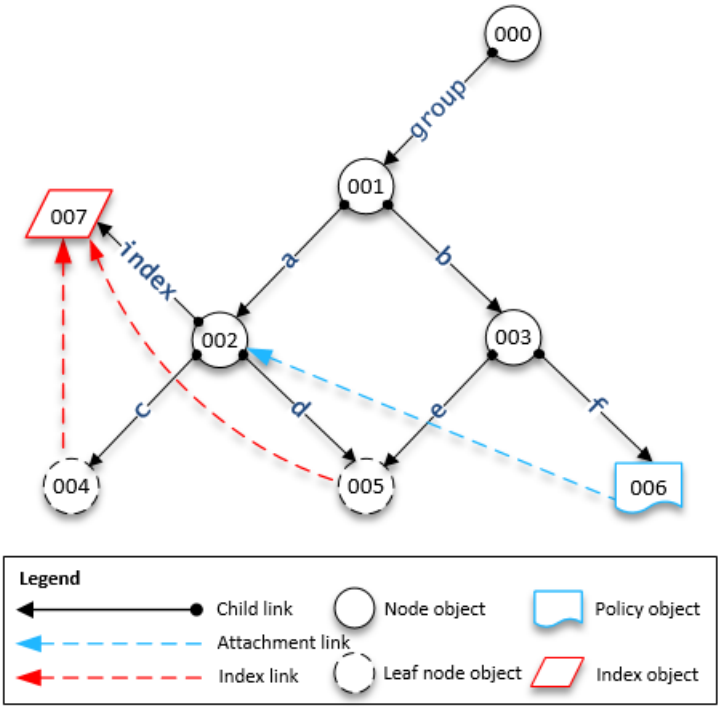
クラウドディレクトリでは、次の範囲を使用してインデックスのクエリを行う豊富な機能を使用できます。

- FIRST - 最初にインデックスが付けられた属性値から開始します。開始属性値はオプションです。
- LAST - 欠落した値を含め、インデックスの末尾まで属性値を返します。終了属性値はオプションです。
- LAST_BEFORE_MISSING_VALUES - 欠落した値を除いて、インデックスの末尾まで属性値を返します。
- INCLUSIVE - 指定されている属性値を含みます。
- EXCLUSIVE - 指定されている属性値を除きます。

親パスの表示

「ListObjectParentPaths」の API コールを使用して、すべてのオブジェクトタイプ (ノード、リーフノード、ポリシーノード、インデックスノード) で利用できる親パスをすべて取得できます。この API オペレーションは、オブジェクトのすべての親を評価する必要がある場合に役立ちます。この呼び出しでは、リクエストされたオブジェクトまで、ディレクトリルートからすべてのオブジェクトを返します。また、親までに複数のパスが存在する場合は、ユーザーが定義した MaxResults に基づき、パス数を返します。返されるパスおよびノードの順序は、オブジェクトが削除または移動されている場合を除き、複数の API コール間で一貫しています。ディレクトリルートにつながらないパスは、ターゲットオブジェクトから無視されます。

この仕組みに関する一例として、ディレクトリには、以下の図のようなオブジェクト階層があります。



番号付きの形状は、オブジェクトが異なることを意味します。このオブジェクトとディレクトリルート (000) の間にある矢印数は、完全なパスを表しており、出力で表現されます。階層内の特定のリーフノードオブジェクトへのクエリのリクエストおよびレスポンスを以下のテーブルに示します。

オブジェクトに対するクエリの例

リクエスト	レスポンス
`004, PageToken : null, MaxResults: 1`	`[{/group/a/c], [000, 001, 002, 004]}], PageToken: null`
`005, PageToken : null, MaxResults: 2`	`[{/group/a/d, [000, 001, 002, 005]}, { /group/b/e, [000, 001, 003, 005]}], PageToken: null`

リクエスト	レスポンス
	Note この例では、オブジェクト 005 には、親として 002 および 003 のノードの両方が含まれます。また、MaxResults が 2 であるため、どちらのパスのリストにもオブジェクトが表示されます。
005, PageToken : null, MaxResults: 1	[{/group/a/d, [000, 001, 002, 005]}], PageToken: <encrypted_next_token>
005, PageToken : <encrypted_next_token>, MaxResults: 1	[{/group/b/e, [000, 001, 003, 005]}], PageToken: null Note この例では、オブジェクト 005 には、親として 002 および 003 のノードの両方が含まれます。また、MaxResults が 1 であるため、ページトークンでページ分割された呼び出しを複数回行うことで、オブジェクトのリストを含むすべてのパスが取得されます。
006, PageToken : null, MaxResults: 1	[{/group/b/f, [000, 001, 003, 006]}], PageToken: null
007, PageToken : null, MaxResults: 1	[{/group/a/index, [000, 001, 002, 007]}], PageToken: null

整合性のレベル

Amazon クラウドディレクトリは、分散型のディレクトリストアです。データは、さまざまなアベイラビリティーゾーンの複数のサーバーに分散されます。書き込みリクエストが正常に行われると、すべてのサーバー上のデータが更新されます。データは、通常 1 秒以内にすべてのサーバー上で利用できるようになります。このサービスのユーザーを支援するために、クラウドディレクトリでは、読み書きオペレーションに関して 2 つの整合性レベルを用意しています。このセクションでは、その整合性レベルに加え、クラウドディレクトリの結果整合性について説明します。

分離レベルの読み込み

クラウドディレクトリからデータを読み込む場合は、読み込む分離レベルを指定する必要があります。各分離レベルのレイテンシーとデータ鮮度の間にはトレードオフがあります。

- 結果整合性— スナップショットの分離レベルは、データがすぐに利用できるかどうかに関係なく読み込みます。。すべての分離レベルのうち、レイテンシーは最も低くなります。 また、表示されるディレクトリ内のデータが古い場合があります。EVENTUAL の分離では、書き込み後の読み取りの整合性は保証されません。つまり、書き込み後すぐにデータを読み取れるとは限りません。
- 直列化可能— 直列化分離レベルは、Cloud Directory で最も高い整合性を提供します。SERIALIZABLE の分離レベルで読み込みを行うと、正常な書き込みからデータを受け取ることを保証します。リクエストしたものの変更が反映されてないデータに変更を加えると、システムは `RetryableConflictException` でそのリクエストを拒否します。これらの例外を再試行することをお勧めします (次のセクションを参照してください)。正常に再試行されると、SERIALIZABLE は、書き込み後の読み取り整合性を実現します。

書き込みリクエスト

クラウドディレクトリでは、複数の書き込みリクエストを行っても、同一のオブジェクトを同時にアップデートすることはできません。同一のオブジェクトに対して 2 つの書き込みリクエストが実行されると、いずれかのオペレーションは、`RetryableConflictException` で拒否されます。これらの例外を再試行することをお勧めします (以下のセクションを参照)。

> **Note**
>
> `RetryableConflictException` 書き込み操作時に受信したレスポンスで、競合状態を検出することはできません。この状況を早めるユースケースが提示された場合にこの例外が常

に発生することを保証するものではありません。例外の発生有無は、内部で処理される各リクエストの順序によって異なります。

RetryableConflictExceptions

同一オブジェクトでの書き込み後に、SERIALIzABLE 分離レベルで書き込み操作または読み取り操作を実行すると、Cloud Directory は、で応答する場合があります。`RetryableConflictException`。この例外は、クラウドディレクトリサーバーで、以前の書き込みの内容が処理されていないことを表します。この状況は一時的なもので、内部で迅速に対処されます。`RetryableConflictException` を使用して、書き込み後の読み込み整合性を検出できないことを把握することが重要です。特定のユースケースによって、この例外が発生することを保証するものではありません。

Cloud Directory クライアントを設定してを再試行することをお勧めします。`RetryableConflictException`。この設定を行うことで、オペレーション時にエラーが発生しなくなります。Java でこの設定を行う手順を以下のサンプルコードで示します。

```
     RetryPolicy retryPolicy = new RetryPolicy(new CloudDirectoryRetryCondition(),
         PredefinedRetryPolicies.DEFAULT_BACKOFF_STRATEGY,
         PredefinedRetryPolicies.DEFAULT_MAX_ERROR_RETRY,
               true);

       ClientConfiguration clientConfiguration = new
ClientConfiguration().withRetryPolicy(retryPolicy);


 AmazonCloudDirectory client = new AmazonCloudDirectory (
   new BasicAWSCredentials(…), clientConfiguration);


public static class CloudDirectoryRetryCondition extends SDKDefaultRetryCondition {

 @Override
 public boolean shouldRetry(AmazonWebServiceRequest originalRequest,
AmazonClientException exception,
   int retriesAttempted) {

  if (exception.getCause() instanceof RetryableConflictException) {
   return true;
  }
```

```
    return super.shouldRetry(originalRequest, exception, retriesAttempted);
  }
}
```

インデックス作成と検索

Amazon Cloud Directory は、インデックス作成の方法として次の2つのをサポートしています。値ベースとタイプベースの。値ベースのインデックス作成が最も一般的な形式です。これにより、オブジェクトの属性値に基づいてディレクトリ内のオブジェクトのインデックスを作成して検索できます。タイプベースのインデックス作成では、オブジェクトタイプに基づいてディレクトリ内のオブジェクトのインデックスを作成して検索できます。オブジェクトタイプは、ファセットで定義します。スキーマとファセットの詳細については、「Schemas」と「Facets」を参照してください。

Cloud Directory インデックスにより、オブジェクトを属性値またはファセット値に従って簡単にリスト表示できます。各インデックスは、特定の名前付き属性またはファセットを使用するように作成時に定義されます。たとえば、「Person」ファセットの「E メール」属性にインデックスを定義することができます。インデックスは第 1 クラスのオブジェクトであり、アプリケーションロジックのニーズに応じてクライアントが柔軟に作成、変更、リストおよび削除できることを意味します。

概念的に、インデックスは子を持つノードと似ています。インデックス付きノードへのリンクは、子をアタッチしたときにラベル付けされるのではなく、インデックス付き属性に従ってラベル付けされます。ただし、インデックスリンクは親子エッジではなく、独自の列挙 API オペレーションのセットを持っています。

クラウドディレクトリのインデックスは、他のシステムと同じように自動的に入力されないことを理解することが重要です。代わりに、API コールを使用してオブジェクトをインデックスに対して直接アタッチまたはデタッチします。これは少し作業量が多くなりますが、さまざまなインデックススコープを柔軟に定義できます。たとえば、特定のノードの直接の子ノードのみを追跡するインデックスを定義できます。または、部門のすべてのノードなど、ローカルルートにある特定のブランチのすべてのオブジェクトを追跡するインデックスを定義することもできます。両方を同時に行うこともできます。

トピック

- インデックスのライフサイクル
- ファセットベースのインデックス作成
- 一意なインデックスと一意でないインデックスの相違点

インデックスのライフサイクル

以下の API コールをインデックスの開発ライフサイクルに役立てることができます。

1. CreateIndex API コールでインデックスを作成します。インデックスで追跡するアタッチ済みオブジェクトの属性を記述するインデックス定義構造を指定します。また、この定義はインデックスが一意性を強制するかどうかを示します。結果は新しいインデックスのオブジェクト ID であり、他のオブジェクトと同様に、この ID は即座に階層にアタッチする必要があります。たとえば、これはインデックスを保持する専用のブランチである場合があります。
2. AttachToIndex API 呼び出しを使用して、手動でインデックスにオブジェクトをアタッチします。次に、インデックスは、アタッチされている各オブジェクトの定義済み属性の値を自動的に追跡します。
3. インデックスを使用してオブジェクトを検索し、より効率的に列挙するには、ListIndex を呼び出して、対象とする値の範囲を指定します。
4. ListAttachedIndices API コールを使用して、特定のオブジェクトにアタッチされているインデックスを列挙します。
5. DetachFromIndex API コールを使用して、手動でインデックスからオブジェクトを削除します。
6. すべてのオブジェクトをインデックスからデタッチしたら、そのインデックスを DeleteObject API コールで削除できます。

ディレクトリ内のインデックスの数に制限はありませんが、すべてのオブジェクトが使用する領域の制限を除きます。インデックスとそのアタッチメントはスペースを消費しますが、ノードや親子リンクでの消費と類似しています。特定のオブジェクトにアタッチできるインデックスの数には制限があります。詳細については、「Amazon Cloud Directory の制限」を参照してください。

ファセットベースのインデックス作成

ファセットベースのインデックス作成と検索では、ディレクトリのサブセットのみを検索することで、ディレクトリの検索を最適化できます。これを行うには、スキーマファセットを使用します。たとえば、ディレクトリ内のすべてのユーザーオブジェクトを検索する代わりに、従業員ファセットを含むユーザーオブジェクトのみを検索できます。この効率化により、クエリで取得するデータのレイテンシー時間と量を低減できます。

ファセットベースのインデックス作成では、Cloud Directory インデックス API オペレーションを使用してインデックスを作成し、オブジェクトのファセットにアタッチできます。また、インデックス結果を表示し、これらの結果を特定のファセットに基づいてフィルタすることもできます。これにより、特定タイプのファセットを含むオブジェクトのみを検索範囲にして、クエリ時間を短縮し、データ量を削減できます。

“facets” API コールと CreateIndex API コールで使用される ListIndex 属性は、オブジェクトに適用されるファセットのコレクションを表面化させます。この属性は、CreateIndex API コールと ListIndex API コールでのみ使用できます。次のサンプルコードに示すように、スキーマ ARN はディレクトリのリージョン、所有者アカウント、およびディレクトリ ID を使用して Cloud Directory スキーマを参照します。このサービスが提供するスキーマは一覧に表示されません。

```
String cloudDirectorySchemaArn = String.format(“arn:aws:clouddirectory:%s:%s:directory/
%s/schema/CloudDirectory/1.0", region, ownerAccount, directoryId);
```

たとえば、次のサンプルコードで作成される AWS アカウントおよびディレクトリに固有のファセットベースのインデックスでは、ファセット SalesDepartmentFacet で作成されるすべてのオブジェクトを列挙できます。

> **Note**
>
> 以下に示すように、パラメータ内で「facets」値を必ず使用します。サンプルコードに示されている「facets」のインスタンスは、Cloud Directory サービスによって提供および制御される値を参照します。これらはインデックス作成に利用できますが、読み取り専用アクセスに限られます。

```
// Create a facet-based index
String cloudDirectorySchemaArn = String.format(“arn:aws:clouddirectory:%s:%s:directory/
%s/schema/CloudDirectory/1.0",
    region, ownerAccount, directoryId);

facetIndexResult = clouddirectoryClient.createIndex(new CreateIndexRequest()
  .withDirectoryArn(directoryArn)
  .withOrderedIndexedAttributeList(List(new AttributeKey()
        .withSchemaArn(cloudDirectorySchemaArn)
        .withFacetName("facets")
        .withName("facets")))
        .withIsUnique(false)
        .withParentReference("/")
        .withLinkName("MyFirstFacetIndex"))
facetIndex = facetIndexResult.getObjectIdentifier()

// Attach objects to the facet-based index
clouddirectoryClient.attachToIndex(new
 AttachToIndexRequest().withDirectoryArn(directoryArn)
```

```
  .withIndexReference(facetIndex).withTargetReference(userObj))

// List all objects
val listResults = clouddirectoryClient.listIndex(new ListIndexRequest()
  .withDirectoryArn(directoryArn)
  .withIndexReference(facetIndex)
  .getIndexAttachments()

// List the index results filtering for a certain facet
val filteredResults = clouddirectoryClient.listIndex(new ListIndexRequest()
  .withDirectoryArn(directoryArn)
  .withIndexReference(facetIndex)
  .withRangesOnIndexedValues(new ObjectAttributeRange()
    .withAttributeKey(new AttributeKey()
      .withFacetName("facets")
      .withName("facets")
      .withSchemaArn(cloudDirectorySchemaArn))
    .withRange(new TypedAttributeValueRange()
      .withStartMode(RangeMode.INCLUSIVE)
      .withStartValue("MySchema/1.0/SalesDepartmentFacet")
      .withEndMode(RangeMode.INCLUSIVE)
      .withEndValue("MySchema/1.0/SalesDepartmentFacet")
    )))
```

一意なインデックスと一意でないインデックスの相違点

一意なインデックスは、インデックスにアタッチされているオブジェクトに対してインデックス付き属性値の一意性を強制するという点で、一意でないインデックスとは異なります。たとえば、Person オブジェクトを 2 つのインデックス内に事前設定する場合、一意なインデックスは「email」属性とし、一意でないインデックスは「lastname」属性とします。lastname インデックスでは、同じラストネームを持つ多数の Person オブジェクトをアタッチできます。一方、email インデックスをターゲットとする `AttachToIndex` 呼び出しでは、同じ email 属性の Person がアタッチ済みである場合、`LinkNameAlreadyInUseException` エラーが返されます。Person オブジェクト自体は、エラーに伴って削除されないことに注意してください。したがって、アプリケーションは Person を作成して階層にアタッチし、これをインデックスにアタッチするという操作のすべてを、単一のバッチリクエストで処理する場合があります。これにより、いずれかのインデックスで一意性の違反が発生した場合、オブジェクトとそのすべてのアタッチメントは自動的にロールバックされます。

Cloud Directory 管理

このセクションでは、Cloud Directory 環境を運用およびメンテナンスするすべての手順を示します。

トピック

ディレクトリの管理

このセクションでは、Cloud Directory 環境の一般的なディレクトリタスクをメンテナンスする方法について説明します。

トピック

ディレクトリを作成する

Amazon クラウドディレクトリにディレクトリを作成する前に、まず AWS Directory Service にスキーマを適用する必要があります。スキーマなしでディレクトリを作成することはできず、通常はスキーマが 1 つ適用されます。ただし、クラウドディレクトリ API 操作を使用して、ディレクトリに追加のスキーマを適用します。詳細については、『Amazon Cloud Directory API リファレンスガイド』の「ApplySchema」を参照してください。

Cloud Directory を作成するには

1. 左AWS Directory Service コンソールナビゲーションペインのクラウドのディレクトリ] で、ディレクトリ。
2. 選択Cloud Directory セットアップ。
3. []新しいディレクトリに適用するスキーマの選択[] に、ディレクトリのわかりやすい名前 (`User Repository`[] をクリックし、以下のいずれかのオプションを選択します。

- マネージ型スキーマ
- サンプルスキーマ
- カスタムスキーマ

サンプルスキーマとカスタムスキーマは、開発状態 (デフォルトで) です。スキーマの状態に関する詳細については、「スキーマのライフサイクル」を参照してください。スキーマをディレクトリに適用するには、スキーマを発行済み状態に変換する必要があります。コンソールを使用してサンプルスキーマを正常に発行するには、次の操作に対する権限が必要です。

- `clouddirectory:Get*`
- `clouddirectory:List*`
- `clouddirectory:CreateSchema`
- `clouddirectory:CreateDirectory`
- `clouddirectory:PutSchemaFromJson`
- `clouddirectory:PublishSchema`
- `clouddirectory:DeleteSchema`

サンプルスキーマは AWS によって提供される読み取り専用テンプレートであるため、直接発行することはできません。代わりに、サンプルスキーマに基づいてディレクトリを作成することを選択すると、コンソールは選択したサンプルスキーマの一時コピーを作成し、それを作成します。開発状態。次に、その開発スキーマのコピーが作成され、発行済み状態にされます。発行されると、開発スキーマが削除されるため、サンプルスキーマを発行するときに、`DeleteSchema` アクションが必要になります。

4. [Next] を選択します。
5. ディレクトリ情報を確認し、必要な変更を加えます。情報が正しい場合は、[作成] を選択します。

ディレクトリの削除

Cloud Directory のディレクトリを削除するには、次の手順を実行します。

> **Note**
>
> ディレクトリを削除する前に、まずそのディレクトリを無効化する必要があります。手順については、「ディレクトリの無効化」を参照してください。

ディレクトリを削除するには

1. 左AWS Directory Service コンソールナビゲーションペインのクラウドのディレクトリ[]で、ディレクトリ。
2. テーブルで、削除するディレクトリ ID の横にあるオプションを選択します。
3. [Actions] を選択します。
4. 選択削除
5. 左ディレクトリの削除ダイアログで、ディレクトリの名前を入力して操作を確認し、削除。

ディレクトリの無効化

Cloud Directory のディレクトリを無効化するには、次の手順を実行します。

ディレクトリを無効化するには

1. 左AWS Directory Service コンソールナビゲーションペインのクラウドのディレクトリ[]で、ディレクトリ。
2. テーブルで、無効にするディレクトリ ID の横にあるオプションを選択します。
3. [Actions] を選択します。
4. 選択Disable (無効)

ディレクトリの有効化

次の手順に従って、Cloud Directory で以前に無効にしたディレクトリを有効にします。

ディレクトリを有効にするには

1. 左AWS Directory Service コンソールナビゲーションペインのクラウドのディレクトリ[]で、ディレクトリ。
2. テーブルで、有効にするディレクトリ ID の横にあるオプションを選択します。

3. [Actions] を選択します。
4. 選択Enable (Gems の有効化)

スキーマを管理する

このセクションでは、Cloud Directory 環境の一般的なスキーマのタスクをメンテナンスする方法について説明します。

トピック

- スキーマを作成します。
- スキーマを削除します。
- スキーマをダウンロードする
- スキーマの発行
- スキーマを更新する
- スキーマをアップグレードする

スキーマを作成します。

Amazon Cloud Directory Directory では、スキーマの作成に準拠した JSON ファイルをアップロードすることができます。新しいスキーマを作成するには、一から独自の JSON ファイルを作成するか、コンソールに表示されている既存のスキーマのいずれかをダウンロードします。次に、カスタムのスキーマとしてアップロードします。詳細については、「カスタムスキーマ」を参照してください。

また、Cloud Directory API を使用して、スキーマの作成、削除、ダウンロード、リスト、公開、更新、アップグレードを行うことができます。スキーマ API オペレーションの詳細については、「Amazon クラウドディレクトリの API リファレンスガイド」を参照してください。

任意の方法に応じて、以下のいずれかの手順を選択します。

カスタムスキーマを作成するには

1. 左AWS Directory Service コンソールナビゲーションペインのクラウドのディレクトリ] で、スキーマ。
2. すべての新しいスキーマを定義する JSON ファイルを作成します。JSON ファイルを整形する方法の詳細については、「JSON スキーマ形式」を参照してください。

3. コンソールで、新しいスキーマをアップロード。
4. 左新しいスキーマをアップロードダイアログで、スキーマの名前を入力します。
5. Selectファイルを選択] で、作成したばかりの新しい JSON ファイルを選択し、[オープン。
6. [Upload] を選択します。これにより、新しいスキーマがスキーマライブラリに追加され、開発状態。スキーマの状態に関する詳細については、「スキーマのライフサイクル」を参照してください。

コンソールに表示されている既存のスキーマに基づいてカスタムスキーマを作成するには

1. 左AWS Directory Service コンソールナビゲーションペインのクラウドのディレクトリ] で、スキーマ。
2. スキーマが表示されているテーブルで、コピーするスキーマの近くのオプションを選択します。
3. [Actions] を選択します。
4. 選択スキーマをダウンロード。
5. JSON ファイルの名前を変更し、必要に応じて編集してから、ファイルを保存します。JSON ファイルを整形する方法の詳細については、「JSON スキーマ形式」を参照してください。
6. コンソールで、新しいスキーマをアップロード] で、編集した JSON ファイルを選択し、[]オープン。

   これにより、新しいスキーマがスキーマライブラリに追加され、開発状態。スキーマの状態に関する詳細については、「スキーマのライフサイクル」を参照してください。

スキーマを削除します。

以下の手順に従って、Cloud Directory でスキーマを削除します。

スキーマを削除するには

1. 左AWS Directory Service コンソールナビゲーションペインのクラウドのディレクトリ] で、スキーマ。
2. テーブルで、削除するスキーマ名の横にあるオプションを選択します。
3. [Actions] を選択します。
4. 選択削除
5. 左スキーマを削除します。ダイアログで、操作を確定するには削除。

スキーマをダウンロードする

以下の手順に従って、スキーマをダウンロードします。

スキーマをダウンロードするには

1. 左AWS Directory Service コンソールナビゲーションペインのクラウドのディレクトリ] で、スキーマ。
2. テーブルで、ダウンロードするスキーマ名の横にあるオプションを選択します。
3. [Actions] を選択します。
4. 選択スキーマをダウンロード

スキーマの発行

以下の手順に従って、Cloud Directory でスキーマを発行します。

スキーマを公開するには

1. 左AWS Directory Service コンソールナビゲーションペインのクラウドのディレクトリ] で、スキーマ。
2. パブリッシュするスキーマ名の横にあるテーブルで、オプションを選択します。
3. [Actions] を選択します。
4. 選択発行
5. 左スキーマの発行ダイアログで、次の情報を入力します。
    a. スキーマ名
    b. メジャーバージョン
    c. マイナーバージョン
6. [Publish] を選択します。

スキーマを更新する

Cloud Directory でスキーマを更新するには、次の手順に従います。

スキーマを更新するには

1. 左AWS Directory Service コンソールナビゲーションペインのクラウドのディレクトリ] で、スキーマ。
2. 表で、更新するスキーマ名の横にあるオプションを選択します。
3. [Actions] を選択します。
4. [Update (更新)] を選択する
5. 左スキーマを更新します。ダイアログで、オプションでスキーマ名] を選択するか、ファイルを選択ファセットと属性を適用または削除するには、次の手順に従います。
6. [更新] を選択します。

スキーマをアップグレードする

スキーマをアップグレードすると、選択したファセットと属性が、パブリッシュされたスキーマに追加されます。以下の手順に従って、パブリッシュされたスキーマをアップグレードします。

スキーマをアップグレードするには

1. 左AWS Directory Service コンソールナビゲーションペインのクラウドのディレクトリ] で、スキーマ。
2. アップグレードするスキーマ名の横にある表で、オプションを選択します。
3. [Actions] を選択します。
4. 選択アップグレード
5. 左パブリッシュされたスキーマをアップグレードするダイアログで、次のいずれかのオプションを選択し、アップグレード:

   - 現在の開発スキーマのリストから選択
   - 新しいスキーマファイル (JSON) をアップロードする
6. 選択アップグレード。

Amazon Cloud Directory でのセキュリティ

クラウドセキュリティは AWS の最優先事項です。AWS のお客様は、セキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャから利点を得られます。

セキュリティは、AWS とお客様の間の共有責任です。共有責任モデルでは、これをクラウドのセキュリティおよびクラウド内のセキュリティとして説明しています。

- クラウドのセキュリティ – AWS は、AWS クラウドで AWS のサービスを実行するインフラストラクチャを保護する責任を担います。また、AWS では安全に使用できるサービスも用意されています。AWS コンプライアンスプログラムの一環として、サードパーティーの監査人によって定期的にセキュリティの有効性がテストおよび検証されています。Amazon Cloud Directory に適用されるコンプライアンスプログラムの詳細については、コンプライアンスプログラムによる AWS 対象範囲内のサービス。
- クラウド内のセキュリティ – お客様の責任はお客様が使用する AWS のサービスによって決まります。また、お客様は、お客様のデータの機密性、企業の要件、および適用可能な法律および規制などの他の要因についても責任を担います。

このドキュメントは、Cloud Directory を使用する際の責任共有モデルの適用方法を理解するのに役立ちます。以下のトピックでは、セキュリティおよびコンプライアンスの目的を達成するために Cloud Directory を設定する方法を示します。また、Cloud Directory リソースのモニタリングや保護に AWS の他のサービスを利用する方法についても説明します。

トピック

Amazon Cloud Directory での Identity and Access Management

Amazon Cloud Directory へのアクセスには、AWS によってリクエストの認証に使用される認証情報が必要です。これらの認証情報を取得したユーザーに、AWS リソースにアクセスするためのア

クセス権限を付与する必要があります。以下のセクションでは、使用方法について詳しく説明します。AWS Identity and Access Management (IAM)およびCloud Directory を使用して、リソースにアクセスできるユーザーを制御することで、リソースをセキュリティで保護できます。

- Authentication
- アクセスコントロール

Authentication

AWS にアクセスできるアイデンティティのタイプは以下のとおりです。

- AWS アカウントのルートユーザー - AWS アカウントを初めて作成する場合は、このアカウントのすべての AWS のサービスとリソースに対して完全なアクセス許可を持つシングルサインインアイデンティティで始めます。このアイデンティティは root ユーザーと呼ばれ、AWS アカウントの作成に使用したメールアドレスとパスワードでのサインインによりアクセスされます。強くお勧めしているのは、日常的なタスクには、それが管理者タスクであっても、root ユーザーを使用しないことです。代わりに、最初の IAM ユーザーを作成するためにのみ、ルートユーザーを使用するというベストプラクティスに従います。その後、ルートユーザーの認証情報を安全な場所に保管し、それらを使用して少数のアカウントおよびサービス管理タスクのみを実行します。
- IAM ユーザー—IAM ユーザーは、特定のカスタム権限 (たとえば、Cloud Directory にディレクトリを作成するアクセス権限) を持つ AWS アカウント内のアイデンティティです。IAM のユーザー名とパスワードを使用して、AWS マネジメントコンソール、AWS ディスカッションフォーラム、AWS サポートセンターなどのセキュリティ保護された AWS ウェブページにサインインできます。

  ユーザー名とパスワードに加えて、各ユーザーのアクセスキーを生成することもできます。複数の SDK の 1 つを通してまたは AWS コマンドラインインターフェイス (CLI) を使用して、プログラムで AWS のサービスにアクセスするときに、これらのキーを使用します。SDK と CLI ツールでは、アクセスキーを使用してリクエストが暗号で署名されます。AWS ツールを使用しない場合は、リクエストに自分で署名する必要があります。Cloud Directory署名バージョン 4は、インバウンド API リクエストを認証するためのプロトコルです。リクエストの認証の詳細については、」署名バージョン 4 の署名プロセス()AWS 全般のリファレンス。

- IAM ロール - IAM ロールは、アカウントで作成して特定のアクセス許可を付与できる IAM アイデンティティです。IAM ロールは、AWS でできることとできないことを決定するのが、アクセス許可ポリシーを伴う AWS アイデンティティであるという点で IAM ユーザーと似ています。ただし、ユーザーは 1 人の特定の人に一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。また、ロールには標準の長期認証情報 (パスワードやアクセスキーなど) も関連付けられません。代わりに、ロールを引き受けると、ロールセッション用の一時的なセキュリティ認証情報が提供されます。IAM ロールと一時的な認証情報は、次の状況で役立ちます。

  - フェデレーティッドユーザーアクセス - ユーザーを作成するのではなく、AWS Directory Service、エンタープライズユーザーディレクトリ、またはウェブ ID プロバイダーの既存のユーザーアイデンティティを使用することもできます。このようなユーザーはフェデレーティッドユーザーと呼ばれます。AWS では、ID プロバイダーを通じてアクセスがリクエストされたとき、フェデレーティッドユーザーにロールを割り当てます。フェデレーティッドユーザーの詳細については、」フェデレーティッドユーザーとロール()IAM ユーザーガイド。

  - AWS のサービスへのアクセス - サービスロールは、サービスがお客様に代わってお客様のアカウントでアクションを実行するために引き受ける IAM ロールです。サービスロールは、お客様のアカウント内のみでアクセスを提供します。他のアカウントのサービスへのアクセス権を付与するためにサービスロールを使用することはできません。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、「」を参照してください。AWS のサービスにアクセス許可を委任するロールの作成()IAM ユーザーガイド。

  - Amazon EC2 で実行されているアプリケーション - EC2 インスタンスで実行され、AWS CLI または AWS API リクエストを作成しているアプリケーションの一時的な認証情報を管理するには、IAM ロールを使用します。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。AWS ロールを EC2 インスタンスに割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスにアタッチされたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時認証情報を取得することができます。詳細については、「」を参照してください。IAM ロールを使用して、Amazon EC2 インスタンスで実行されるアプリケーションにアクセス許可を付与する()IAM ユーザーガイド。

アクセスコントロール

有効な認証情報があればリクエストを認証できますが、許可を持っていないかぎり Cloud Directory リソースの作成やアクセスはできません。たとえば、Amazon Cloud Directory を作成するにはアクセス権限が必要です。

以下のセクションでは、Cloud Directory のアクセス許可を管理する方法について説明します。最初に概要のセクションを読むことをお勧めします。

- Cloud Directory リソースへのアクセス権限の管理の概要
- Cloud Directory でのアイデンティティベースのポリシー (IAM ポリシー) の使用
- Amazon Cloud Directory API のアクセス許可: アクション、リソース、条件リファレンス

Cloud Directory リソースへのアクセス権限の管理の概要

すべての AWS リソースは AWS アカウントによって所有され、となり、リソースの作成またはアクセスは、アクセス権限のポリシーによって管理されます。アカウント管理者は、アクセス権限ポリシーを IAM アイデンティティ (ユーザー、グループ、ロール) にアタッチできます。一部のサービス (AWS Lambda など) もリソースにアクセス権限ポリシーをアタッチすることができます。

> **Note**
>
> アカウント管理者 (または管理者ユーザー) は、管理者権限を持つユーザーです。詳細については、『IAM ユーザーガイド』の「IAM ベストプラクティス」を参照してください。

アクセス権限を付与する場合、アクセス権限を取得するユーザー、取得するアクセス権限の対象となるリソース、およびそれらのリソースに対して許可される特定のアクションを決定します。

トピック

- Cloud Directory リソースと運用
- リソース所有権について
- リソースへのアクセスの管理
- ポリシー要素の指定: アクション、効果、リソース、プリンシパル
- ポリシーでの条件の指定

Cloud Directory リソースと運用

Cloud Directory では、プライマリリソースはディレクトリとスキーマです。これらのリソースには、次の表に示すとおり、一意の Amazon リソースネーム (ARN) が関連付けられています。

リソースタイプ	ARN 形式
ディレクトリ	`arn:aws:clouddirectory:` *`region`*`:`*`account-id`* `:directory/` *`directory-id`*
スキーマ	`arn:aws:clouddirectory:` *`region`*`:`*`account-id`* `:schema/`*`schema-state`* `/`*`schema-name`*

スキーマの状態と ARN の詳細については、「」を参照してください。ARN の例()Amazon Cloud Directory API リファレンス。

Cloud Directory には、適切なリソースを操作するための一連のオペレーションが用意されています。使用可能なオペレーションのリストについては、「Amazon Cloud Directory Actions」または「Directory Service Actions」のいずれかを参照してください。

リソース所有権について

リソース所有者は、リソースを作成した AWS アカウントです。つまり、リソース所有者は、リソースの作成リクエストを認証するプリンシパルエンティティ (ルートアカウント、IAM ユーザー、または IAM ロール) の AWS アカウントです。以下の例では、このしくみを示しています。

- AWS アカウントのルートアカウントの認証情報を使用して、ディレクトリなどの Cloud Directory リソースを作成する場合、AWS アカウントは AWS リソースの所有者です。
- AWS アカウントに IAM ユーザーを作成し、そのユーザーに Cloud Directory リソースを作成するためのアクセス権限を付与する場合、そのユーザーは Cloud Directory リソースも作成できます。ただし、ユーザーが属する AWS アカウントは リソースを所有しています。
- AWS アカウントに Cloud Directory リソースを作成するためのアクセス権限を持つ IAM ロールを作成する場合は、ロールを引き受けることのできるいずれのユーザーも Cloud Directory リソースを作成できます。ロールが属する AWS アカウントは、Cloud Directory リソースを所有します。

リソースへのアクセスの管理

アクセスポリシーでは、誰が何にアクセスできるかを記述します。以下のセクションで、アクセス権限のポリシーを作成するために使用可能なオプションについて説明します。

> **Note**
>
> このセクションでは、Cloud Directory のコンテキストでの IAM の使用について説明します。これは、IAM サービスに関する詳細情報を取得できません。完全な IAM ドキュメントについては、「」を参照してください。IAM とは()IAM ユーザーガイド。IAM ポリシーの構文と記述については、「」を参照してください。AWS IAM ポリシーのリファレンス()IAM ユーザーガイド。

IAM アイデンティティにアタッチされたポリシーは、アイデンティティベースリソースにアタッチされたポリシー (IAM ポリシー) およびポリシーは、リソースベースポリシー Cloud Directory では、アイデンティティベースのポリシー (IAM ポリシー) のみサポートされます。

トピック

- アイデンティティベースのポリシー (IAM ポリシー)
- リソースベースのポリシー

アイデンティティベースのポリシー (IAM ポリシー)

ポリシーを IAM アイデンティティにアタッチできます。たとえば、次の操作を実行できます。

- アカウントのユーザーまたはグループにアクセス権限ポリシーをアタッチする— アカウント管理者は、特定のユーザーに関連付けられるアクセス権限ポリシーを使用して、そのユーザーに Cloud Directory リソース (新しいディレクトリなど) の作成を許可するアクセス権限を付与することができます。
- ロールにアクセス権限ポリシーをアタッチする (クロスアカウントアクセス権限を付与する)— アイデンティティベースのアクセス権限ポリシーを IAM ロールにアタッチして、クロスアカウントアクセス権限を付与できます。たとえば、アカウント A の管理者は、次のように他のまたは AWS にクロスアカウントのアクセス権限を別の AWS アカウント (アカウント B) または AWS サービスに付与するロールを作成することができます。
  1. アカウント A の管理者は、IAM ロールを作成して、アカウント A のリソースに権限を付与するロールに権限ポリシーをアタッチします。

2. アカウント A の管理者は、アカウント B をそのロールを引き受けるプリンシパルとして識別するロールに、信頼ポリシーをアタッチします。
3. アカウント B の管理者は、アカウント B のユーザーにロールを引き受ける権限を委任できるようになります。これにより、アカウント B のユーザーにアカウント A のリソースの作成とアクセスが許可されます。AWS サービスのアクセス権限を付与してロールを引き受けさせたい場合は、信頼ポリシー内のプリンシパルも、AWS サービスのプリンシパルとなることができます。

IAM を使用したアクセス権限の委任の詳細については、「」を参照してください。アクセス管理()IAM ユーザーガイド。

以下のアクセス権限ポリシーは、Create で始まるすべてのアクションを実行するためのアクセス権限をユーザーに付与します。これらのアクションは、ディレクトリまたはスキーマなどの、Cloud Directory リソースに関する情報を表示します。ワイルドカード文字 (*) は、Resource要素は、アカウントによって所有されるすべての Cloud Directory リソースに対してそれらのアクションが許可されることを示します。

```
{
   "Version":"2017-01-11",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":"clouddirectory:Create*",
         "Resource":"*"
      }
   ]
}
```

Cloud Directory でアイデンティティベースのポリシーを使用する方法の詳細については、Cloud Directory でのアイデンティティベースのポリシー (IAM ポリシー) の使用。ユーザー、グループ、ロール、アクセス許可の詳細については、IAM ユーザーガイドの「アイデンティティ (ユーザー、グループ、ロール)」を参照してください。

リソースベースのポリシー

Amazon S3 などの他のサービスでは、リソースベースのアクセス権限ポリシーもサポートされています。たとえば、ポリシーを S3 バケットにアタッチして、そのバケットに対するアクセス許可を管理できます。Cloud Directory では、リソースベースのポリシーはサポートされていません。

ポリシー要素の指定: アクション、効果、リソース、プリンシパル

各Cloud Directory リソースについて (Cloud Directory リソースと運用) では、このサービスは、一連の API オペレーションを定義します。使用可能な API オペレーションのリストについては、Amazon Cloud Directory のアクションまたはDirectory Service アクション。これらの API オペレーションを実行するためのアクセス権限を付与するために、Cloud Directory ではポリシーに一連のアクションを定義できます。API オペレーションを実行する場合に、複数のアクションで権限が必要となる場合があることに注意してください。

以下は、基本的なポリシーの要素です。

- リソース— ポリシーで Amazon Resource Name (ARN) を使用して、ポリシーを適用するリソースを識別します。Cloud Directory リソースの場合、IAM ポリシーでは必ずワイルドカード文字 (*) を使用します。詳細については、「Cloud Directory リソースと運用」を参照してください。
- アクション - アクションのキーワードを使用して、許可または拒否するリソースオペレーションを識別します。たとえば、`clouddirectory:GetDirectory`権限は、ユーザーにCloud Directory を実行するためのアクセス権限を付与します。`GetDirectory`オペレーション.
- 効果— ユーザーが特定のアクションをリクエストする際の効果を指定します。許可または拒否のいずれかになります。リソースへのアクセスを明示的に許可していない場合、アクセスは暗黙的に拒否されます。また、明示的にリソースへのアクセスを拒否すると、別のポリシーによってアクセスが許可されている場合でも、ユーザーはそのリソースにアクセスできなくなります。
- プリンシパル - アイデンティティベースのポリシー (IAM ポリシー) で、ポリシーがアタッチされているユーザーが黙示的なプリンシパルとなります。リソースベースのポリシーでは、権限 (リソースベースのポリシーにのみ適用) を受け取りたいユーザー、アカウント、サービス、またはその他のエンティティを指定します。Cloud Directory では、リソースベースのポリシーはサポートされていません。

IAM ポリシーの構文と記述の詳細については、「」を参照してください。AWS IAM ポリシーのリファレンス()IAM ユーザーガイド。

すべての Amazon Cloud Directory API アクションとそれらが適用されるリソースの表については、Amazon Cloud Directory API のアクセス許可: アクション、リソース、条件リファレンス。

ポリシーでの条件の指定

アクセス権限を付与するとき、アクセスポリシー言語を使用して、ポリシーが有効になる必要がある条件を指定できます。たとえば、特定の日付の後にのみ適用されるポリシーが必要になる場合があります。ポリシー言語での条件の指定の詳細については、」条件()IAM ユーザーガイド。

条件を表すには、あらかじめ定義された条件キーを使用します。Cloud Directory に固有の条件キーはありません。ただし、AWS 全体の条件キーがあり、必要に応じて使用できます。AWS 全体を対象とするすべてのキーのリストについては、「」を参照してください。使用できるグローバル条件キー()IAM ユーザーガイド。

Cloud Directory でのアイデンティティベースのポリシー (IAM ポリシー) の使用

このトピックでは、アカウント管理者が IAM アイデンティティ (ユーザー、グループ、ロール) へのアクセス権限ポリシーをアタッチする、アイデンティティベースのポリシーの例を示します。

> **Important**
>
> 初めに、Cloud Directory リソースへのアクセスを管理するための基本概念と使用可能なオプションについて説明する概要トピックを読むことをお勧めします。詳細については、「Cloud Directory リソースへのアクセス権限の管理の概要」を参照してください。

このセクションでは、次のトピックを対象としています。

- AWS Directory Service コンソールを使用するために必要なアクセス権限
- Amazon Cloud Directory の AWS 管理 (定義済み) ポリシー

AWS Directory Service コンソールを使用するために必要なアクセス権限

AWS Directory Service コンソールを使用して作業するユーザーの場合、そのユーザーは、上記のポリシーに記載されているアクセス許可または、Directory Service Full Access Role または Directory Service 読み取り専用ロールによって付与されたアクセス許可を持っている必要があります。詳細については、」Amazon Cloud Directory の AWS 管理 (定義済み) ポリシー。

これらの最小限必要なアクセス権限よりも制限された IAM ポリシーを作成している場合、その IAM ポリシーを使用するユーザーに対してコンソールは意図したとおりには機能しません。

Amazon Cloud Directory の AWS 管理 (定義済み) ポリシー

AWS は、AWS によって作成され管理されるスタンドアロンの IAM ポリシーが提供する多くの一般的ユースケースに対応します。管理ポリシーは、一般的ユースケースに必要なアクセス権限を付与することで、どの権限が必要なのかをユーザーが調査する必要をなくすることができます。詳細については、IAM ユーザーガイドの「AWS 管理ポリシー」を参照してください。

アカウントのユーザーにアタッチ可能な以下の AWS 管理ポリシーは、Amazon Cloud Directory に固有のものです。

- Amazonクラウドディレクトリ読み取り専用アクセス— ユーザーまたはグループに Amazon Cloud Directory のすべてのリソースに対する読み取り専用アクセスを許可します。詳細については、AWS マネジメントコンソールの「ポリシー」ページを参照してください。
- Amazonクラウドディレクトリフルアクセス— ユーザーまたはグループに Amazon Cloud Directory へのフルアクセスを許可します。詳細については、AWS マネジメントコンソールの「ポリシー」ページを参照してください。

また、他の IAM ロールとの使用に適している AWS 管理ポリシーもあります。これらのポリシーは、Amazon Cloud Directory 内のユーザーに関連付けられたロールに割り当てられます。また、それらのユーザーが Amazon EC2 などの他の AWS リソースにアクセスするために必要です。

また、ユーザーが必要な API アクションおよびリソースにアクセスできるようにするカスタム IAM ポリシーを作成できます。これらのカスタムポリシーは、それらのアクセス権限が必要な IAM ユーザーまたはグループにアタッチできます。

Amazon Cloud Directory API のアクセス許可: アクション、リソース、条件リファレンス

アクセスコントロール をセットアップし、IAM アイデンティティ (アイデンティティベースのポリシー) にアタッチできるアクセス権限ポリシーを作成する際、以下の表をリファレンスとして使用できます。 リストには各 Amazon Cloud Directory API オペレーション、アクションを実行するためのアクセス権限を付与できる対応するアクション、アクセス権限を付与できる AWS リソースが掲載されています。ポリシーの `Action` フィールドでアクションを指定し、ポリシーの `Resource` フィールドでリソースの値を指定します。

Amazon Cloud Directory ポリシーで AWS 全体の条件キーを使用して、条件を表すことができます。AWS 全体を対象とするすべてのキーのリストについては、「」を参照してください。使用できるグローバル条件キー()IAM ユーザーガイド。

> **Note**
>
> アクションを指定するには、API オペレーション名 (`clouddirectory:` など) の前に `clouddirectory:CreateDirectory` プレフィックスを使用します。

Amazon Cloud Directory でのログ記録とモニタリング

ベストプラクティスとして、変更がログに記録されることを確実にするためにディレクトリを監視する必要があります。これにより、予期しない変更を調査することができ、不要な変更をロールバックできます。Amazon Cloud Directory は現在、AWS CloudTrail をサポートしています。これを使用して、ディレクトリおよび関連するアクティビティを監視できます。

詳細については、「」を参照してください。CloudTrail を使用したCloud Directory API コールのログ記録。

Amazon Cloud Directory のコンプライアンス検証

Amazon Cloud Directory のセキュリティとコンプライアンスは、AWS のさまざまなコンプライアンスプログラムの一環として、サードパーティーの監査機関によって評価されます。このプログラムには、ISO、SOC、PCI、FedRAMP、HIPAA などがあります。

特定のコンプライアンスプログラムの対象範囲内の AWS のサービスのリストについては、「コンプライアンスプログラムの対象範囲内の AWS のサービス」を参照してください。一般的な情報については、「AWS コンプライアンスプログラム」を参照してください。

サードパーティーの監査報告書は、AWS Artifact を使用してダウンロードすることができます。詳細については、「AWS Artifact のレポートのダウンロード」を参照してください。

Cloud Directory を使用する際のお客様のコンプライアンス責任は、お客様のデータの機密性や貴社のコンプライアンス目的、適用可能な法律および規制によって決定されます。AWS では、コンプライアンスに役立つ以下のリソースを提供しています。

- セキュリティとコンプライアンスのクイックスタートガイド — これらのデプロイガイドでは、アーキテクチャ上の考慮事項について説明し、セキュリティとコンプライアンスに焦点を当てたベースライン環境を AWS にデプロイするための手順を示します。
- HIPAA のセキュリティとコンプライアンスに関するホワイトペーパーを作成する – このホワイトペーパーでは、企業が AWS を使用して HIPAA 準拠のアプリケーションを作成する方法について説明します。

- AWS コンプライアンスのリソース - このワークブックおよびガイド群には、貴社の所在地や属する業界に適用可能なものが含まれています。
- AWS Config - 貴社のリソース構成が社内実務、業界のガイドライン、規制要件にどこまで準拠できているかが評価される AWS のサービスです。
- AWS セキュリティハブ – この AWS サービスでは、AWS 内のセキュリティ状態を包括的に表示しており、セキュリティ業界の標準およびベストプラクティスへの準拠を確認するのに役立ちます。

Amazon Cloud Directory での耐障害性

AWS のグローバルインフラストラクチャは AWS リージョンとアベイラビリティーゾーンを中心として構築されます。AWS リージョンには、低レイテンシー、高いスループット、そして高度の冗長ネットワークで接続されている複数の物理的に独立· 隔離されたアベイラビリティーゾーンがあります。Cloud Directory は、これらの原則に基づいて構築されており、物理的に互いに隔離された複数の AWS リージョンで利用可能です。各リージョン内では、少なくとも 3 つのアベイラビリティーゾーンを通じてサービスがさらにサポートされ、1 つのアベイラビリティーゾーンが利用できないことによるサービスのダウンタイムを最小限に抑えます。

AWS リージョンとアベイラビリティーゾーンの詳細については、「AWS グローバルインフラストラクチャ」を参照してください。

Amazon Cloud Directory のインフラストラクチャセキュリティ

マネージドサービスである Amazon Cloud Directory は、「」で説明されている AWS グローバルネットワークセキュリティの手順で保護されています。Amazon Web Services: セキュリティプロセスの概要ホワイトペーパー

AWS が公開した API コールを使用して、ネットワーク経由で Cloud Directory にアクセスします。クライアントは Transport Layer Security (TLS) をサポートしている必要があります。TLS 1.2 以降が推奨されています。また、Ephemeral Diffie-Hellman (DHE) や Elliptic Curve Ephemeral Diffie-Hellman (ECDHE) などの Perfect Forward Secrecy (PFS) を使用した暗号スイートもクライアントでサポートされている必要があります。これらのモードは、Java 7 以降など、最近のほとんどのシステムでサポートされています。

コマンドラインインターフェイスまたは API を使用して AWS にアクセスするときに FIPS 140-2 検証済みの暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、連邦情報処理規格 (FIPS) 140-2 を参照してください。

また、リクエストは、アクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、AWS Security Token Service (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

トランザクションのサポート

Amazon Cloud Directory では、実際の階層の変更を反映するために、新しいオブジェクトを追加したり、新しいオブジェクトと既存のオブジェクト間の関係を追加したりする場合があります。バッチ操作では、以下の利点を活用して、このようなディレクトリタスクを容易に管理できます。

- バッチ操作では、ディレクトリに対するオブジェクトの読み書きに必要なラウンドトリップの数を削減し、アプリケーションの全体的なパフォーマンスを改善できます。
- バッチ書き込みでは、SQL データベースと同等のトランザクションセマンティクスを利用できます。すべてのオペレーションは正常に完了します。または、いずれかのオペレーションが失敗すると、他のすべてのオペレーションも適用されません。
- バッチ参照により、新規作成したオブジェクトへの参照を使用して、オブジェクトを関係に追加するなどの他のアクションを行い、書き込み操作の前の読み取り操作の使用に伴うオーバーヘッドを削減できます。

BatchWrite

バッチ書き込み操作では、ディレクトリに対する複数の書き込み操作を実行します。バッチ書き込みのすべてのオペレーションはシーケンシャルに実行されます。これは SQL データベーストランザクションのように機能します。バッチ書き込み内のオペレーションの 1 つに障害が発生しても、バッチ書き込み全体としてはディレクトリに影響しません。バッチ書き込みが失敗すると、バッチ書き込みの例外が発生します。例外には、例外のタイプとメッセージと共に失敗したオペレーションのインデックスが含まれます。この情報は、障害の根本原因を特定するのに役立ちます。

バッチ書き込みの一部として、以下の API オペレーションがサポートされています。

- AddFacetToObject
- AttachObject
- AttachPolicy
- AttachToIndex
- AttachTypedLink
- CreateIndex
- CreateObject
- DeleteObject

- DetachFromIndex
- DetachObject
- DetachTypedLink
- RemoveFacetFromObject
- UpdateObjectAttributes

バッチ参照名

バッチ参照名は、中間バッチ操作の一部としてオブジェクトを参照する必要がある場合に、バッチ書き込みに対してのみサポートされます。たとえば、特定のバッチ書き込みの一環として、デタッチされている 10 個の異なるオブジェクトをディレクトリの別の部分にアタッチするとします。バッチ参照がないと、10 個のオブジェクト参照のすべてを読み取り、これらをバッチ書き込みの一環として再アタッチする際に入力として提供する必要があります。バッチ参照を使用すると、デタッチされているリソースをアタッチする際に識別できます。バッチ参照は、番号記号/ハッシュタグ記号 (#) を接頭辞とする任意の通常の文字列です。

次のコード例では、リンク名 "this-is-a-typo" のオブジェクトが、バッチ参照名 "ref" のルートからデタッチされています。後で、同じオブジェクトがリンク名 "correct-link-name" でルートにアタッチされています。オブジェクトは、バッチ参照に設定された子リファレンスを使用して識別されます。バッチ参照がないと、最初にデタッチされている objectIdentifier を取得して、それをアタッチ中に子リファレンスに提供する必要があります。バッチ参照名を使用してこの余分な読み取りを回避することができます。

```
BatchDetachObject batchDetach = new BatchDetachObject()
        .withBatchReferenceName("ref")
        .withLinkName("this-is-a-typo")
        .withParentReference(new ObjectReference().withSelector("/"));
BatchAttachObject batchAttach = new BatchAttachObject()
        .withParentReference(new ObjectReference().withSelector("/"))
        .withChildReference(new ObjectReference().withSelector("#ref"))
        .withLinkName("correct-link-name");
BatchWriteRequest batchWrite = new BatchWriteRequest()
        .withDirectoryArn(directoryArn)
        .withOperations(new ArrayList(Arrays.asList(batchDetach, batchAttach)));
```

BatchRead

バッチ読み取り操作では、ディレクトリに対して複数の読み取り操作を実行します。次のコード例では、1 回のバッチ読み取りで参照 “/managers” を持つオブジェクトの子が、参照 “/managers/bob” を持つオブジェクトの属性と共に読み取られています。

```
BatchListObjectChildren listObjectChildrenRequest = new BatchListObjectChildren()
        .withObjectReference(new ObjectReference().withSelector("/managers"));
BatchListObjectAttributes listObjectAttributesRequest = new BatchListObjectAttributes()
        .withObjectReference(new ObjectReference().withSelector("/managers/bob"));
BatchReadRequest batchRead = new BatchReadRequest()
        .withConsistencyLevel(ConsistencyLevel.SERIALIZABLE)
        .withDirectoryArn(directoryArn)
        .withOperations(new ArrayList(Arrays.asList(listObjectChildrenRequest,
 listObjectAttributesRequest)));
BatchReadResult result = cloudDirectoryClient.batchRead(batchRead);
```

バッチ読み取りでサポートされる API オペレーションは以下のとおりです。

- GetObjectInformation
- ListAttachedIndices
- ListIncomingTypedLinks
- ListIndex
- ListObjectAttributes
- ListObjectChildren
- ListObjectParentPaths
- ListObjectPolicies
- ListOutgoingTypedLinks
- ListPolicyAttachments
- LookupPolicy

バッチオペレーションの制限

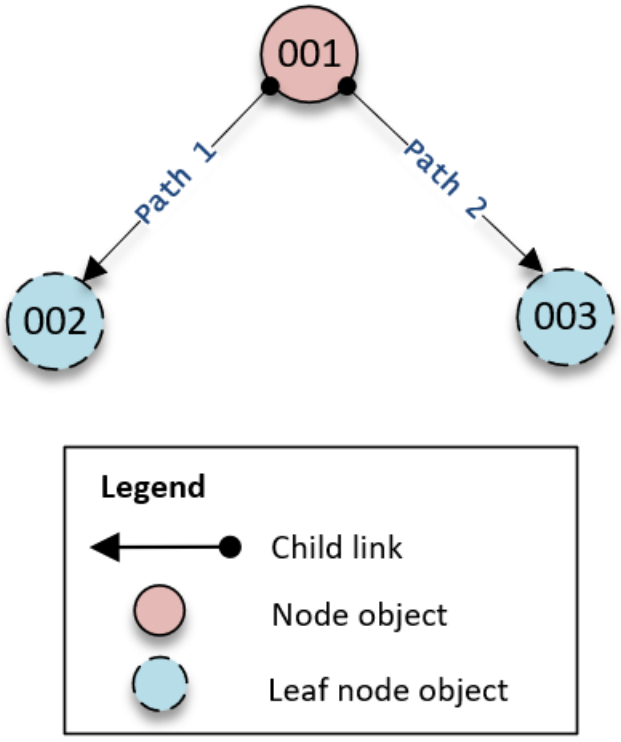
サーバーへのリクエスト (バッチリクエストを含む) ごとに、リクエスト内のオペレーション数に関係なく、操作できるリソースの最大数が決められています。これにより、リソースの最大数を超え

ない限り、柔軟性の高いバッチリクエストを構成できます。リソースの最大数の詳細については、「Amazon Cloud Directory の制限」を参照してください。

制限は、バッチ内の各オペレーションの書き込みまたは読み取りを合計することで計算されます。たとえば、読み取り操作の制限は現在、API コールあたり 200 オブジェクトです。たとえば、バッチを構成して 9 個の ListObjectChildren API コールを追加し、各コールで 20 オブジェクトを読み取る必要があるとします。読み取りオブジェクトの合計数 (9 x 20 =180) は 200 を超えないため、このバッチ操作は成功します。

同じコンセプトが、書き込み操作の計算にも該当します。たとえば、現在の書き込み操作の制限は 20 です。バッチを設定して 2 つの UpdateObjectAttributes API コールを追加し、各コールで 9 個の書き込み操作を行う場合、これも成功します。いずれの場合も、バッチ操作が制限を超えると、オペレーションは失敗し、`LimitExceededException` がスローされます。

バッチ内に含まれているオブジェクトの数を計算するには、実際のノードと leaf_node オブジェクトの両方を含めるのが正しい方法です。パスベースのアプローチを使用してディレクトリツリーを反復処理している場合は、バッチ内で反復処理されている各パスも含める必要があります。たとえば、基本的なディレクトリツリーの次の図に示すように、オブジェクト `003` の属性値を読み込むため、オブジェクトの合計読み込み数は 3 になります。



ツリーを下方向に読み込む際のトラバース動作は次のようになります。

1. 001 オブジェクトを読み込み、オブジェクト 003 へのパスを決定します。

2. Path 2 まで下に移動します。

3. オブジェクトの読み取り 003

同様に、属性の数についても、オブジェクト 001 および 003 の数をカウントして制限に到達しないようにする必要があります。

例外処理

Cloud Directory でのBatch 操作が失敗し、エラーになる場合があります。このような場合、エラーの処理方法を知っておくことが重要です。エラーの解決方法は、読み取り操作と書き込み操作で異なります。

バッチ書き込み操作のエラー

バッチ書き込み操作が失敗すると、Cloud Directory はバッチ操作全体を失敗させて例外を返します。例外には、失敗した操作のインデックスと共に例外のタイプとメッセージが含まれます。RetryableConflictException と表示された場合は、エクスポネンシャルバックオフを使用してもう一度試すことができます。これを行うシンプルな方法としては、例外またはエラーが返されるたびに待機時間を倍化します。たとえば、最初のバッチ書き込み操作が失敗した場合は、100 ミリ秒待ってから、リクエストを再試行します。2 番目のリクエストが失敗したら、200 ミリ秒待ってから再試行します。3 番目のリクエストが失敗したら、400 ミリ秒待ってから再試行します。

バッチ読み取り操作のエラー

バッチ読み取り操作が失敗すると、レスポンスには成功応答または例外応答のいずれかが含まれます。個々のバッチ読み取り操作エラーでバッチ読み取り操作全体がエラーになることはありません。Cloud Directory は、操作ごとに成功応答または失敗応答を個別に返します。

Cloud Directory 関連ブログ記事

- Batch 操作を使用した、Amazon Cloud Directory での複数のオブジェクトの書き込みおよび読み取り
- Amazon Cloud Directory でバッチ参照を使用してBatch リクエストの新規オブジェクトを参照する方法

Amazon Cloud Directory コンプライアンス

Amazon Cloud Directory は、以下の標準について監査済みであり、コンプライアンスの認定を取得する必要がある場合にはソリューションの一部となります。

Amazon Cloud Directory は、FedRAMP (FedRAMP) のセキュリティ要件を満たしており、FedRAMP Joint Authority Board (JAB) Provisional Authority to Operate (P-ATO) の FedRAMP Moderate ベースラインで認定されています。FedRAMP の詳細については、「FedRAMP コンプライアンス」を参照してください。

Amazon Cloud Directory には、クレジットカード業界 (PCI) のデータセキュリティ標準 (DSS) バージョン 3.2、サービスプロバイダーレベル 1 で準拠証明書を取得しています。AWS の製品やサービスを使用してカード所有者のデータを保存、処理、転送するユーザーは、各自の PCI DSS 準拠認定の管理に Cloud Directory を使用できます。PCI DSS の詳細 (AWS PCI Compliance Package をリクエストする方法など) については、「PCI DSS レベル 1」を参照してください。

AWS は、Health 保険の相互運用性と説明責任に関する法令 (HIPAA) コンプライアンスプログラムを拡張し、Amazon Cloud Directory をHIPAA 対応サービス。AWS と事業提携契約 (BAA) を締結している場合は、Cloud Directory を使用して HIPAA 準拠アプリケーションを構築できます。AWS では、HIPAA に焦点を当てたホワイトペーパーは、医療情報の処理や保存に AWS の活用をお考えのお客様向けです。詳細については、「HIPAA への準拠」を参照してください。

	Amazon Cloud Directory は、ISO/IEC 27001、ISO/IEC 27017、ISO/IEC 27018、および ISO 9001 のコンプライアンス認証を正常に取得しました。詳細については、「ISO 27001」、「ISO 27017」、「ISO 27018」、および「ISO 9001」を参照してください。
	システムと Organization Control (SOC) レポートとは、重要な準拠統制および目標を Amazon Cloud Directory がどのように達成したかを実証する、独立した第三者による審査報告書です。このレポートの目的は、オペレーションとコンプライアンスをサポートするよう確立された AWS 統制を、ユーザーおよびユーザーの監査人が容易に把握できるようにすることです。詳細については、「SOC コンプライアンス」を参照してください。

共有責任

セキュリティ (HIPAA と PCI への準拠を含む) は共有責任です。Cloud Directory のコンプライアンス状況は AWS クラウド内で実行するアプリケーションに自動的には適用されない点を理解することが重要です。AWS サービスの使用が標準に準拠していることを確認する必要があります。

Cloud Directory API の使用

Amazon Cloud Directory には、Cloud Directory 機能へのプログラムによるアクセスを可能にする一連の API オペレーションが含まれています。「」を使用できます。Amazon Cloud Directory API リファレンスガイド各種要素の作成および管理を Cloud Directory API にリクエストする方法については、リクエストの構成要素、レスポンスの内容、およびリクエストの認証方法についても説明します。

Cloud Directory では、開発者が新しいアプリケーションを構築するのに必要なすべての API 操作を提供します。次のカテゴリ API 呼び出しを提供します。

- スキーマの作成、読み込み、更新、削除 (CRUD)
- ファセットの CRUD
- ディレクトリの CRUD
- オブジェクトの CRUD (ノード、ポリシーなど)
- インデックス定義の CRUD
- バッチ読み取り、バッチ書き込み

Cloud Directory API に対する請求方法

API コールに対する請求方法は、API コールのタイプ別に異なります。結果的に整合性のある読み込み API コール、強い整合性のある読み込みの API コール、および書き込み API コールごとに異なる請求レートがあります。メタデータ API コールは無料です。

強い整合性のあるオペレーションは、値の読み取り時に、書き込み後の読み取りの整合性のために使用されます。結果整合性のオペレーションは、更新の実行中に値を取得するために使用されます。結果整合性のオペレーションでは、値の読み取り元のホストで更新が現在処理中であるため、最新の正確な結果が取得されない場合があります。ただし、このような読み取りオペレーションのレイテンシーは、パフォーマンス呼び出しの取得時は低くなります。

Cloud Directory からデータを読み取る場合は、オペレーションのタイプとして、結果的に整合性のある読み込み、または強い整合性のある読み込みのいずれかを指定する必要があります。読み取りタイプは、整合性のレベルに基づきます。2 つの整合性レベルは、結果的に整合性のある読み込みの場合は EVENTUAL、強い整合性のある読み込みの場合は SERIALIZABLE です。詳細については、「整合性のレベル」を参照してください。

次の表は、すべての Cloud Directory API の一覧です。また、各 API と AWS アカウントの請求との関係を示しています。

API	結果的に整合性のある読み込み [1]	強力な整合性のある読み込み [2]	書き込み [3]	メタデータ [4]
AddFacetToObject			X	
ApplySchema				X
AttachObject			X	
AttachPolicy			X	
AttachToIndex			X	
AttachTypedLink			X	
バッチ読み取り	X	X		
バッチ書き込み			X	
CreateDirectory			X	
CreateFacet				X
CreateIndex			X	
CreateObject			X	
CreateSchema				X
CreateTypedLinkFacet				X
DeleteDirectory				X
DeleteFacet				X
DeleteObject			X	

API	結果的に整合性のある読み込み [1]	強力な整合性のある読み込み [2]	書き込み [3]	メタデータ [4]
DeleteSchema				X
DetachFromIndex			X	
DetachObject			X	
DetachPolicy			X	
DetachTypedLink			X	
DeleteTypedLinkFacet				X
DisableDirectory				X
EnableDirectory			X	
GetAppliedSchemaVersion				X
GetDirectory				X
GetFacet				X
GetLinkAttributes	X	X		
GetObjectAttributes	X	X		
GetObjectInformation	X	X		
GetSchemaAsJson				X

API	結果的に整合性のある読み込み [1]	強力な整合性のある読み込み [2]	書き込み [3]	メタデータ [4]
GetTypedLinkFacetInformation				X
ListAppliedSchemaArns				X
ListAttachedIndices	X	X		
ListDevelopmentSchemaArns				X
ListDirectories				X
ListFacetAttributes				X
ListFacetNames				X
ListIncomingTypedLinks	X	X		
ListIndex	X	X		
ListManagedSchemaArns				X
ListObjectAttributes	X	X		
ListObjectChildren	X	X		
ListObjectParentPaths	X			

API	結果的に整合性のある読み込み [1]	強力な整合性のある読み込み [2]	書き込み [3]	メタデータ [4]
ListObjectParents	X	X		
ListObjectPolicies	X	X		
ListOutgoingTypedLinks	X	X		
ListPolicyAttachments	X	X		
ListPublishedSchemaArns				X
ListTagsForResource				X
ListTypedLinkFacetAttributes				X
ListTypedLinkFacetNames				X
LookupPolicy	X			
PublishSchema				X
PutSchemaFromJson				X
RemoveFacetFromObject			X	
TagResource				X

API	結果的に整合性のある読み込み [1]	強力な整合性のある読み込み [2]	書き込み [3]	メタデータ [4]
UntagResource				X
UpdateFacet				X
UpdateLinkAttributes			X	
UpdateObjectAttributes			X	
UpdateSchema				X
UpdateTypedLinkFacet				X
UpgradeAppliedSchema				X
UpgradePublishedSchema				X

[1] 結果的に整合性のある読み込み API は、EVENTUAL 整合性レベルで呼び出されます。

[2] 強い整合性のある読み込み API は、SERIALIZABLE 整合性レベルで呼び出されます。

[3] 書き込み API は、書き込み API コールとして請求されます。

[4] メタデータ API は請求対象ではありませんが、メタデータ API コールとして分類されます。

請求の詳細については、」Amazon Cloud Directory 料金表。

Amazon Cloud Directory の制限

Cloud Directory のデフォルトの制限を次に示します。特に明記されていない限り、制限はリージョンごとに存在します。

Amazon Cloud Directory

スキーマおよびディレクトリの制限

制限/概念	数量
ファセットあたりの属性数 (必須の属性を含む)	1,000
オブジェクトあたりのファセット数	5
オブジェクトがアタッチされている一意のインデックスの数	3
スキーマあたりのファセット数	30
属性あたりのルール数	5
ファセットあたりのデフォルト値を持つ属性数	10
ファセットあたりに必要な属性数	30
開発スキーマの数	20
公開済みのスキーマ数	20
適用済みのスキーマ数	5
ディレクトリ数	100
最大ページ要素	30
最大入力サイズ (すべての入力を結合)	200 KB
最大応答サイズ (すべての出力を結合)	1 MB
スキーマの JSON ファイルサイズの上限	200 KB

制限/概念	数量
ファセット名のレングス	64、UTF-8 でエンコードされたバイト
ディレクトリ名のレングス	64、UTF-8 でエンコードされたバイト
スキーマ名のレングス	64、UTF-8 でエンコードされたバイト

オブジェクトの制限

制限/概念	数量
書き込まれたオブジェクトの数	API コールあたり 20
読み取られたオブジェクトの数	API コールあたり 200
書き込まれた属性値の数	API コールあたり 1000
読み取られた属性値の数	API コールあたり 1000
パスの深度	15
最大入力サイズ (すべての入力を結合)	200 KB
最大応答サイズ (すべての出力を結合)	1 MB
ポリシーサイズの制限	10 KB
オブジェクトの削除中に削除できる属性の数	30
型付きリンクの ID 属性の集約された値の長さ	64、UTF-8 でエンコードされたバイト
エッジ名およびリンク名のレングス	64、UTF-8 でエンコードされたバイト
インデックスが付けられた属性値のレングス	512 UTF-8 でエンコードされたバイト
インデックス付けされていない属性値のレングス	2 KB
オブジェクトにアタッチされているポリシーの数	4

バッチ操作の制限

バッチ内で呼び出せる操作の数に制限はありません。詳細については、「バッチオペレーションの制限」を参照してください。

変更できない制限

次の Amazon クラウドディレクトリの制限は、変更または増加することはできません。

- ファセット名のレングス
- ディレクトリ名のレングス
- スキーマ名のレングス
- 最大ページ要素
- エッジ名およびリンク名のレングス
- インデックスが付けられた属性値のレングス

Cloud Directory リソース

以下の表は、本サービスを利用する際に役立つ関連リソースの一覧です。

Cloud Directory の使用開始	リンク
Cloud Directory ウェビナー	https://www.youtube.com/watch?v=UANm3DC_IxE
Cloud Directory サンプル Java コード	https://github.com/aws-samples/AmazonCloudDirectory-sample

Cloud Directory ブログ記事	説明
マネージド型スキーマを使用してアプリケーションを迅速に Amazon Cloud Directory で開発する方法	このブログ記事は、マネージド型スキーマを使用した、Cloud Directory での迅速なプロトタイプ作成と開発について説明しています。サンプルの Java コードも含まれています。
Amazon Cloud Directory でより効率的に検索する方法	このブログ記事では、ファセットベースのインデックスを使用して効率的に検索する方法について説明しています。サンプルの Java コードも含まれています。
インプレーススキーマアップグレードにより Amazon Cloud Directory スキーマの変更を簡単に適用する方法	このブログ記事では、運用 (実行) 中のクラウドディレクトリのインプレーススキーマアップグレードの実行について説明しています。サンプルの Java コードも含まれています。
Batch 操作を使用した、Amazon Cloud Directory での複数のオブジェクトの書き込みおよび読み取り	バッチ読み取り/書き込みの使用方法について説明します。サンプルの Java コードも含まれています。
Amazon Cloud Directory でバッチ参照を使用してBatch リクエストの新規オブジェクトを参照する方法	バッチ参照の使用方法について説明します。サンプルの Java コードも含まれています。

Cloud Directory ブログ記事	説明
Cloud Directory の更新 — Typed Links のSupport	Typed Links を使用して Cloud Directory で階層間の関係を作成および検索する方法について説明します。サンプルの Java コードも含まれています。
新しいクラウドディレクトリ API により、複数の次元に沿ってデータを簡単にクエリできる	`ListObjectParentPaths` API を使用した単一の呼び出しにより、複数のディメンションにわたってデータをクエリする方法について説明します。
Amazon クラウドディレクトリを使用して別個の階層を持つ組織図を作成する方法	サンプルの Java コードを使用してスキーマとディレクトリを作成する方法について説明します。
Amazon クラウドディレクトリ - 階層データ用のクラウドネイティブディレクトリ	AWS の新しいサービスとしてCloud Directory の起動について説明します。

Cloud Directory ドキュメント	リンク
Cloud Directory 開発者ガイド	https://docs.aws.amazon.com/clouddirectory/latest/developerguide/what_is_cloud_directory.html
Cloud Directory API リファレンス	https://docs.aws.amazon.com/clouddirectory/latest/APIReference/welcome.html
Cloud Directory 制限	https://docs.aws.amazon.com/clouddirectory/latest/developerguide/limits.html

Cloud Directory その他	リンク
Cloud Directory 製品情報	https://aws.amazon.com/cloud-directory/
Cloud Directory 価格	https://aws.amazon.com/cloud-directory/pricing/

ドキュメント履歴

次の表は、最終回リリース以降のドキュメントの変更点をまとめるものです。Amazon Cloud Directory 開発者ガイド。

- ドキュメントの最終更新: 2018 年 6 月 21 日

update-history-change	update-history-description	update-history-date
新しいマネージド型スキーマ	マネージド型スキーマオプションのコンテンツを追加しました。	2018 年 6 月 21 日
このガイドにコンテンツを移行しました	お客様のニーズにより直接的に対応するため、のすべての既存のCloud Directoryに関するコンテンツを AWS Directory 開発者ガイドからこの新しい Amazon Cloud Directory 開発者ガイドに移行しました。	2018 年 20 月 6 日
スキーマのインプレースアップグレード	スキーマのインプレースアップグレードによる Amazon Cloud Directory ディレクトリ全体へのスキーマ変更の適用について内容を追加しました。	2017 年 6 月 12 日
ファセットベースのインデックス作成	ファセットベースのインデックスセクションを追加しました。	2017 年 8 月 9 日
バッチ	Amazon Cloud Directory に関する情報を更新しました。	2017 年 7 月 26 日

コンプライアンス	HIPAA および PCI への準拠に関する情報を追加しました。	2017 年 14 月 7 日
型付きリンク	Amazon Cloud Directory の新しい型付きリンクに関するコンテンツを追加しました。	2017 年 5 月 31 日
Amazon Cloud Directory サービスの開始	新しいディレクトリタイプが導入されました。	2017 年 1 月 26 日

AWS の用語集

For the latest AWS terminology, see the AWS glossary in the AWS General Reference.

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。