aws

开发人员指南

Amazon Cloud Directory

Amazon Cloud Directory: 开发人员指南

Table of Contents

什么是 Amazon Cloud Directory？

Amazon Cloud Directory 是 AWS 中的一个高度可用、基于目录的多租户存储。这些目录可根据应用程序的需要自动扩展到数亿个对象。这样，操作员工便可专注于开发和部署能够推动业务发展的应用程序，而不必管理目录基础设施。与传统目录系统不同，Cloud Directory 对目录对象的组织并不局限于单个固定的层次结构中。

借助 Cloud Directory，您可以将目录对象组织为多个层次结构，以便在目录信息中支持许多组织中心和关系。例如，用户目录可以提供基于报告结构、位置和项目附属关系的分层视图。同样，设备目录可以具有基于其制造商、当前拥有者和物理位置的多个分层视图。

Cloud Directory 的核心是一个基于图形的专用目录存储，可为开发人员提供基本构建块。通过 Cloud Directory，开发人员可以执行以下操作：

- 轻松创建基于目录的应用程序，无需担心部署、全球扩展、可用性和性能
- 构建可提供用户和组管理、权限或策略管理、设备注册、客户管理、地址簿和应用程序或产品目录的应用程序
- 定义新目录对象或扩展现有类型，以满足其应用程序需求，同时减少需要编写的代码
- 降低基于 Cloud Directory 的分层应用程序的复杂性
- 管理架构信息随着时间的发展，确保未来对用户的兼容性

Cloud Directory 包含一组 API 操作，用于访问基于 Cloud Directory 的目录中存储的各种对象和策略。有关可用操作的列表，请参阅Amazon Cloud Directory API 操作。有关操作以及执行每项 API 操作所需权限的列表，请参阅 Amazon Cloud Directory API 权限：API 权限：操作、资源和条件参考。

有关受支持的 Cloud Directory 区域的列表，请参阅AWS 区域和终端节点文档中）。有关其他资源，请参阅 Cloud Directory Service。

不是什么 Cloud Directory

Cloud Directory 不是为希望管理或迁移目录基础设施的 IT 管理员提供的目录服务。

入门

在此入门练习中，您将创建一个架构。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. 尽管没有要求，但我们建议您在开始使用控制台之前参阅了解关键 Cloud Directory 概念，以便您熟悉核心功能和术语。

主题

- 创建架构
- 创建 Amazon Cloud Directory
- 使用 Cloud Directory 接口 VPC 终端节点

创建架构

Amazon Cloud Directory 支持上传符合 JSON 标准的文件以进行架构创建。要创建新架构，可以从头开始创建自己的 JSON 文件，或下载控制台中列出的现有架构之一。然后将它作为自定义架构进行上传。有关更多信息，请参阅 自定义架构。

您还可以使用 Cloud Directory API 创建、删除、下载、列出、发布、更新和升级架构。有关架构 API 操作的更多信息，请参阅 Amazon Cloud Directory API 参考指南。

根据首选方法，选择以下任一过程。

创建自定义架构

1. 在AWS Directory Service 控制台导航窗格中，在Cloud Directory中，选择Schemas。
2. 创建包含所有新架构定义的 JSON 文件。有关如何对 JSON 文件设置格式的更多信息，请参阅 JSON 架构格式。
3. 在控制台中，选择上传新架构。
4. 在上传新架构对话框中，为架构键入名称。
5. Select选择文件，选择您刚刚创建的新 JSON 文件，然后选择打开。
6. 选择 Upload。这会将新架构添加到架构库中，并将它置于开发状态。有关架构状态的更多信息，请参阅架构生命周期。

在控制台中基于现有架构创建自定义架构

1. 在AWS Directory Service 控制台导航窗格中，在Cloud Directory中，选择Schemas。
2. 在列出架构的表中，选择要复制的架构附近的选项。
3. 选择 Actions。
4. 选择Download schema。
5. 重命名 JSON 文件，根据需要进行编辑，然后保存文件。有关如何对 JSON 文件设置格式的更多信息，请参阅 JSON 架构格式。
6. 在控制台中，选择上传新架构，选择您刚编辑的 JSON 文件，然后选择打开。

   这会将新架构添加到架构库中，并将它置于开发状态。有关架构状态的更多信息，请参阅架构生命周期。

创建 Amazon Cloud Directory

AWS Directory Service 要求首先应用架构，然后才能在 Amazon Cloud Directory 中创建目录。若不使用架构，则无法创建目录，通常目录会应用一个架构。但是，可以使用 Cloud Directory API 操作将其他架构应用于目录。有关更多信息，请参阅 Amazon Cloud Directory API 参考指南 中的 ApplySchema。

创建 Cloud Directory

1. 在AWS Directory Service导航窗格中，在Cloud Directory中，选择目录。
2. 选择设置 Cloud Directory。
3. 在选择要应用于新目录的架构，键入目录的友好名称，如`User Repository`，然后选择下列选项之一：

   - 托管架构
   - 示例架构
   - 自定义架构

   示例架构和自定义架构放置在开发状态（默认情况下）。有关架构状态的更多信息，请参阅架构生命周期。架构必须先转换为已发布状态，然后才能应用于目录。要使用控制台成功发布示例架构，您必须有权执行以下操作：

   - `clouddirectory:Get*`

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

因为示例架构是 AWS 提供的只读模板，所以无法直接发布它们。相反，当选择基于示例架构创建目录时，控制台会创建所选示例架构的临时副本，并将它置于开发状态。它随后会创建该开发架构的副本，并将它置于已发布状态。发布之后，开发架构即被删除，这就是在发布示例架构时需要 `DeleteSchema` 操作的原因。

4. 选择 Next。
5. 查看目录信息并进行必要的更改。如果信息正确，请选择 Create (创建)。

使用 Cloud Directory 接口 VPC 终端节点

如果使用 Amazon Virtual Private Cloud (Amazon VPC) 托管 AWS 资源，则可以在 VPC 和 Cloud Directory 之间建立私有连接。您可以使用此连接实现 Cloud Directory 与 VPC 上的资源的通信而不用访问公共 Internet。

Amazon VPC 是一项 AWS 服务，可用来启动在虚拟网络中定义的 AWS 资源。借助 VPC，您可以控制您的网络设置，如 IP 地址范围、子网、路由表和网络网关。要将 VPC 连接到 Cloud Directory，请定义一个接口 VPC 终端节点（用于 Cloud Directory）。该终端节点提供了到 Cloud Directory 的可靠、可扩展的连接，无需 Internet 网关、网络地址转换 (NAT) 实例或 VPN 连接。有关更多信息，请参阅 。什么是 Amazon VPC？中的Amazon VPC 用户指南。

接口 VPC 终端节点由 AWS PrivateLink 提供支持，后者是一种 AWS 技术，可将 elastic network interface 与私有 IP 地址结合使用来支持 AWS 服务之间的私有通信。有关更多信息，请参阅 。适用于 AWS 服务的 AWS PrivateLink。

以下步骤适用于 Amazon VPC 的用户。有关更多信息，请参阅 。Amazon VPC 入门中的Amazon VPC 用户指南。

Availability

Cloud Directory 当前在以下区域中支持 VPC 终端节点：

- US East (Ohio)
- US East (N. Virginia)
- US West (Oregon)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)
- Canada (Central)
- Europe (Frankfurt)
- Europe (Ireland)
- Europe (London)
- AWS GovCloud（美国西部）

为 Cloud Directory 创建 VPC

要开始将 Cloud Directory 与您的 VPC 一起使用，请使用 Amazon VPC 控制台为 Cloud Directory 创建接口 VPC 终端节点。有关更多信息，请参阅创建接口终端节点。

- 适用于服务类别中，选择AWS 服务。
- 对于 Service Name (服务名称)，选择 **`com.amazonaws.region.clouddirectory`**。这将为 Cloud Directory 操作创建 VPC 终端节点。

有关一般信息，请参阅Amazon VPC 是什么？中的Amazon VPC 用户指南。

控制对 Cloud Directory VPC 终端节点的访问

VPC 终端节点策略是一种 IAM 资源策略，您在创建或修改终端节点时可将它附加到终端节点。如果在创建终端节点时未附加策略，我们将为您附加默认策略以允许对服务进行完全访问。终端节点策略不会覆盖或替换 IAM 用户策略或服务特定的策略。这是一个单独的策略，用于控制从终端节点中对指定服务进行的访问。

终端节点策略必须采用 JSON 格式编写。有关更多信息，请参阅 。使用 VPC 终端节点控制对服务的访问中的Amazon VPC 用户指南。

下面是 Cloud Directory 的终端节点策略示例。该策略允许通过 VPC 连接到 Cloud Directory 的用户列出目录，并禁止他们执行其他 Cloud Directory 操作。

```
{
```

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

修改 Cloud Directory 的 VPC 终端节点策略

1. 通过以下网址打开 Amazon VPC 控制台：https://console.aws.amazon.com/vpc/。
2. 在导航窗格中，选择终端节点。
3. 如果还没有为 Cloud Directory 创建终端节点，请选择创建终端节点。然后选择**com.amazonaws.region.clouddirectory**，然后选择创建终端节点。
4. 选择**com.amazonaws.region.clouddirectory**端点，然后选择策略选项卡。
5. 选择编辑策略并对策略进行更改。

有关更多信息，请参阅 。使用 VPC 终端节点控制对服务的访问中的Amazon VPC 用户指南。

了解关键 Cloud Directory 概念

Amazon Cloud Directory 是基于目录的数据存储，可以采用面向架构的方式创建各种类型的对象。

主题

Schema

架构是定义可以在目录中创建的对象及其组织方式的分面的集合。架构还会强制实施数据完整性和互操作性。一次可将一个架构应用于多个目录。有关更多信息，请参阅 Schemas。

Facets

分面是在架构中定义的属性、约束和链接的集合。分面组合在一起定义目录中的对象。例如，人员和设备可以是用于定义与多个设备关联的公司员工的分面。有关更多信息，请参阅 Facets。

托管架构

提供的一种模式，使您能够更轻松地快速开发和维护您的应用程序。有关更多信息，请参阅 托管架构。

示例架构

AWS Directory Service 控制台在默认情况下提供的示例架构集。例如，人员、组织和设备都是示例架构。有关更多信息，请参阅 示例架构。

自定义架构

用户定义的一个或多个架构，可以从架构部分或是在 AWS Directory Service 控制台的 Cloud Directory 创建过程中进行上传，或是通过 API 调用进行创建。

Directory

目录是基于架构的数据存储，包含在多层次结构中进行组织的特定类型的对象 (有关更多详细信息，请参阅目录结构)。例如，用户目录可以提供基于报告结构、位置和项目附属关系的分层视图。同样，设备目录可以具有基于其制造商、当前拥有者和物理位置的多个分层视图。

目录为数据存储定义逻辑边界，从而将它与服务中的所有其他目录完全隔离。它还为单个请求定义边界。单个事务或查询在单个目录的上下文中执行。若不使用架构，则无法创建目录，通常目录会应用一个架构。但是，可以使用 Cloud Directory API 操作将其他架构应用于目录。有关更多信息，请参阅 。`ApplySchema`中的Amazon Cloud Directory API 参考指南。

Objects

对象是目录中的结构化数据实体。目录中的对象旨在捕获有关物理或逻辑实体的元数据 (或属性)，通常用于信息发现和强制实施策略。例如，用户、设备、应用程序、AWS 账户、EC2 实例和 Amazon S3 存储桶全都可以表示为目录中不同类型的对象。

对象的结构和类型信息表示为分面的集合。可以使用 `Path` 或 `ObjectIdentifier` 访问对象。对象还可以具有属性 (用户定义的元数据单元)。例如，用户对象可以具有一个名为 email-address 的属性。属性始终与对象相关联。

Policies

策略是专用类型的对象，可用于存储权限或功能。策略提供了 `LookupPolicy` API 操作。查找策略操作采用对任何对象的引用作为其起始输入。随后它从目录一直查找到根。该操作收集它在指向根的每个路径上遇到的所有策略对象。Cloud Directory 不以任何方式解释其中任何策略。而是由 Cloud Directory 用户使用自己的专用业务逻辑来解释策略。

例如，假设一个系统存储员工信息。员工按工作职能分组在一起。我们需要为人力资源组和会计组的成员建立不同权限。人力资源组的成员有权访问工资单信息，而会计组有权访问分类账信息。为了建立这些权限，我们向每个组附加策略对象。需要评估某个用户的权限时，我们可以对该用户的对象使用 `LookupPolicy` API 操作。这些区域有：`LookupPolicy`API 操作沿树从指定策略的对象查找到根。它在每个节点处停止，检查任何附加的策略并返回这些策略。

策略附加

策略可以通过两种方式附加到其他对象：正常的父子附加和特殊的策略附加。使用正常父子附加可以将策略附加到父节点。这通常用于提供一种在数据目录中查找策略的便利机制。策略不能有子级。在 `LookupPolicy` 调用过程中不会返回通过父子附加进行附加的策略。

策略对象还可以通过策略附加来附加到其他对象。可以使用 AttachPolicy 和 DetachPolicy API 操作管理这些策略附加。通过策略附加可以在使用 LookupPolicy API 时找到策略节点。

策略架构规范

要开始使用策略，必须先向架构添加支持创建策略的分面。要实现此目的，请创建将分面的 objectType 设置为 POLICY 的分面。使用具有类型 POLICY 的分面创建对象可确保对象具有策略功能。

在您添加到定义的任意属性之外，Policy 分面还继承两个属性：

- policy_type (字符串，必需) - 这是您提供用于区分不同策略用途的标识符。如果您的策略在逻辑上归入清除类别，我们建议相应设置策略类型属性。LookupPolicy API 返回已附加策略的策略类型 (请参阅 PolicyAttachment)。这样可以方便地筛选所查找的特定策略类型. 它还允许您使用 policy_type 来确定应该如何处理或解释文档。
- policy_document (二进制，必需) – 您可以将应用程序特定数据存储在此属性中，例如与策略关联的权限授予。如果您偏好，还可以在分面上的正常属性中存储应用程序相关数据。

策略 API 概述

有各种专用 API 操作可用于操作策略。有关可用操作的列表，请参阅 Amazon Cloud Directory 操作。

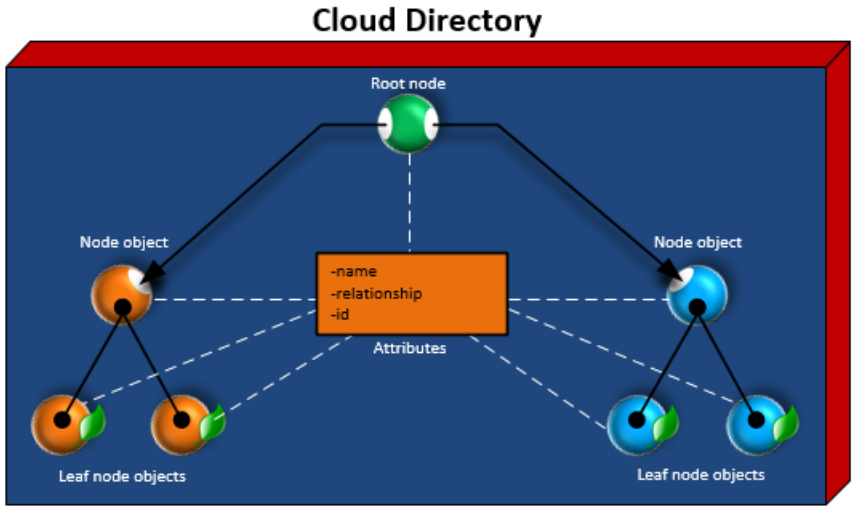
要创建策略对象，请结合使用 CreateObject API 操作及相应的分面：

- 要向对象附加或分离策略，请分别使用操作 AttachPolicy 和 DetachPolicy。
- 要沿树查找附加到对象的策略，请使用 LookupPolicy API 操作。
- 要列出附加到特定对象的策略，请使用 ListObjectPolicies API 操作。

有关操作以及执行每项 API 操作所需权限的列表，请参阅 Amazon Cloud Directory API 权限：API 权限：操作、资源和条件参考。

目录结构

目录中的数据采用由节点、叶节点以及节点之间的链接组成的树模式形成分层结构，如下图所示。这可在应用程序开发中用于对分层数据进行建模、存储和快速遍历。



根节点

根是目录中的顶层节点，用于在层次结构中组织父节点和子节点。这类似于文件系统中的文件夹包含子文件夹和文件的方式。

Node

节点表示可以具有子对象的对象。例如，节点可以在逻辑上表示一组经理，由此各个用户对象是子节点(或叶节点)。节点对象只能有一个父级。

叶节点

叶节点表示没有子级的对象，可能会也可能不会直接连接到父节点。例如，用户或设备对象。叶节点对象可以有多个父级。虽然叶节点对象无需连接到父节点，不过强烈建议这样做，因为如果没有从根开始的路径，则只能通过 `NodeId` 访问对象。如果将这类对象的 ID 放错了位置，则无法再次找到它。

节点链接

一个节点与另一个节点之间的连接。Cloud Directory 在节点之间支持各种链接类型，包括父子链接、策略链接和索引属性链接。

Schemas

借助 Amazon Cloud Directory，架构能够定义可以在目录中创建的对象类型 (用户、设备和组织)、强制对每个对象类实施数据验证以及处理随时间推移对架构进行的更改。更具体地说，架构定义以下内容：

- 可以映射到目录中的对象的一种或多种分面类型 (如 Person、Organization_Person)
- 可以映射到目录中的对象的属性 (如 Name、Description)。属性在各种类型的分面上可以是必需的，也可以设置为可选，在分面的上下文中进行定义。
- 可以对对象属性强制实施的约束 (如 Required、Integer、String)

某个目录应用架构后，该目录中的所有数据都必须符合所应用的架构。通过这种方式，架构定义实质上是一个蓝图，可以用于通过所应用架构构建多个目录。构建之后，应用的这些架构可能各自在不同的方面与原始蓝图有所不同。

应用的架构可以在以后使用版本控制更新，然后重新应用到使用它的所有目录。有关更多信息，请参阅就地架构升级。

Cloud Directory 提供了 API 操作来创建、读取、更新和删除架构。这使编程代理可以轻松地使用架构的内容。这类代理会访问目录以发现应用于目录中数据的分面、属性和约束的完整集合。有关架构 API 的更多信息，请参阅 Amazon Cloud Directory API 参考指南。

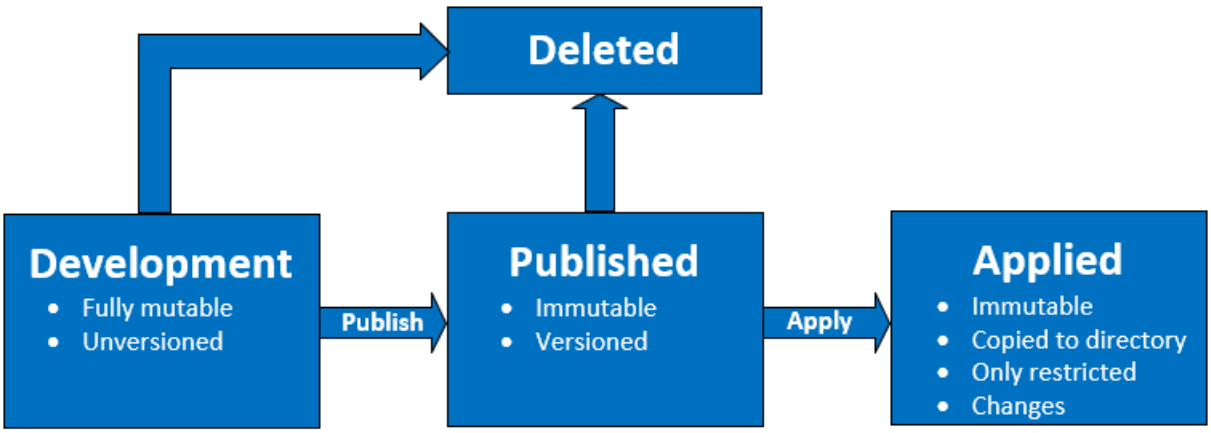
Cloud Directory 支持上传符合 JSON 标准的文件以进行架构创建。也可以使用 AWS Directory Services 控制台创建和管理架构。有关更多信息，请参阅 创建 Amazon Cloud Directory。

主题

架构生命周期

Cloud Directory 提供架构生命周期来帮助进行架构开发。此生命周期由三个状态组成：开发、已发布和已应用。这些状态旨在促进架构的构建和分发。每个状态都具有不同功能来帮助实现此目标。

下图描述了可能的转换和用语。所有架构转换都在写入时复制。例如，发布开发架构不会更改或删除开发架构。



您可以删除处于开发或已发布状态的架构。删除架构操作无法撤消，在删除后也无法还原。

处于开发、已发布和已应用状态的架构具有用于表示架构的 ARN。这些 ARN 在 API 操作中用于描述 API 所操作的架构。通过查看架构 ARN，可方便地识别架构的状态。

- 开发：`arn:aws:clouddirectory:us-east-1:1234567890:schema/development/`*`SchemaName`*
- 已发布：`arn:aws:clouddirectory:us-east-1:1234567890:schema/published/`*`SchemaName`*`/`*`Version`*
- 已应用：`arn:aws:clouddirectory:us-east-1:1234567890:directory/directoryid/schema/`*`SchemaName`*`/`*`Version`*

开发状态

架构最初在开发状态下进行创建。此状态下的架构是完全可变的。可以随意添加或删除分面和属性。大部分架构设计在此状态下进行。此状态下的架构具有名称，但是没有版本。

已发布状态

已发布架构状态存储准备好应用于数据目录的架构。架构是从开发状态发布到已发布状态。无法更改处于已发布状态的架构。可以将已发布架构应用于任意数量的数据目录。

已发布和已应用架构必须具有与之关联的版本。有关版本的更多信息，请参阅架构版本控制。

已应用状态

已发布架构可以应用于数据目录。已应用于数据目录的架构称为已应用。将架构应用于数据目录之后，可以在创建对象时使用架构的分面。可以将多个架构应用于相同数据目录。只允许对已应用架构进行以下更改。

- 向已应用架构添加分面
- 向已应用架构添加非必需属性

Facets

分面是架构中最基本的抽象。它们表示可以与目录中的对象关联的属性集，在概念方面与 LDAP 对象类相似。每个目录对象可以与特定数量的分面关联。有关更多信息，请参阅 Amazon Cloud Directory 限制。

每个分面维护自己的独立属性集。每个分面由基本元数据组成，如分面名称、版本信息和行为。架构 ARN、分面和属性的组合定义了对象的唯一性。

对象分面集、其约束以及它们之间的关系构成抽象架构定义。架构分面用于对以下内容定义约束：

1. 对象中允许的属性
2. 允许应用于对象的策略类型

一旦您将所需的分面添加到架构，就可以将架构应用到目录并创建适用的对象。例如，您可以通过添加诸如计算机、电话和平板电脑等分面来定义设备架构。然后，您可以使用这些分面，在该架构应用到的目录中创建计算机对象、电话对象以及平板电脑对象。

通过 Cloud Directory 的架构支持可以轻松地添加或修改分面和属性，不必担心破坏应用程序。有关更多信息，请参阅 就地架构升级。

就地架构升级

Cloud Directory 提供了更新现有架构属性和分面的功能，用于帮助将您的应用程序集成到 AWS 提供的服务中。处于已发布或已应用状态的架构具有多个版本且无法更改。有关更多信息，请参阅 架构生命周期。

架构版本控制

架构版本指示架构的唯一标识符，开发人员可在为应用程序编程时指定版本，以遵循特定规则和数据格式规定。开发人员务必了解版本控制与 Cloud Directory 配合使用时有两种重要方式。这二者 (主要版本和次要版本) 可以决定未来的架构升级如何影响应用程序。

主要版本

主要版本 是用于跟踪架构的主要版本更改的版本标识符。其长度最多为 10 个字符。不同版本的相同架构是完全独立的。例如，具有相同名称和不同版本的两个架构将被视为完全不同的架构，它们有自己的命名空间。

不向后兼容的更改

我们建议仅在架构不兼容时对主要版本进行更改。例如，在更改现有属性的数据类型 (例如从 `string` 更改为 `integer`) 时或者从架构中删除必需属性时。向后兼容的更改需要将目录数据从以前的架构版本迁移到新架构版本。

次要版本

次要版本 是用于架构就地升级或要进行向后兼容升级 (例如添加额外的属性或添加分面) 时使用的版本标识符。使用次要版本的升级架构可以在使用它的所有目录中就地应用，不会中断任何正在运行的应用程序。这包括在生产环境中使用的目录。有关示例使用案例，请参阅“如何通过就地架构升级轻松应用 Amazon Cloud Directory 架构更改”在云 Directory 博客中。

次要版本信息和历史记录与其他架构信息一起保存在架构元数据存储库中。对象中不保留次要版本信息。引入次要版本的优势在于，只要未更改主要版本，客户端代码就可以无缝工作。

次要版本限制

Cloud Directory 保留最多五个次要版本，因此限制最多五个次要版本。但是，对已发布和应用的架构实施次要版本限制的方式有所不同：

- 已应用架构： 超过次要版本限制后，Cloud Directory 会自动删除最旧的次要版本。
- 已发布架构： 一旦超过次要版本限制，Cloud Directory 不会删除任何次要版本，但它会通过`LimitExceededException`已超出限制。超出次要版本限制后，您可以使用DeleteSchemaAPI 或请求提高限制。

使用架构升级 API 操作

您可以使用 UpgradePublishedSchema API 调用来升级已发布的架构。架构升级使用 UpgradeAppliedSchema API 调用就地应用到依赖于它的目录。您也可以通过调用 GetAppliedSchemaVersion 来获取已应用架构的主要版本和次要版本。或者通过调用查看目录的关联架构 ARN 和架构修订历史记录ListAppliedSchemaArns。Cloud Directory 维护已应用架构更改的最近五个版本。

有关说明性示例，请参阅"如何通过就地架构升级轻松应用 Amazon Cloud Directory 架构更改"在云 Directory 博客中。该博客文章将演示如何在 Cloud Directory 中执行就地架构升级和使用架构版本。它介绍了如何将额外的属性添加到现有分面，将新分面添加到架构，发布新架构并将其应用到正在运行的目录以完成架构就地升级。它还演示了如何查看目录架构的版本历史记录，这用助于确保目录队列运行相同的架构版本并对其应用了正确的架构更改历史记录。

托管架构

Cloud Directory 使您可以使用托管架构轻松快速地开发应用程序。借助托管架构，您可以创建一个目录并从该目录以更快的速度开始创建和检索对象。有关更多信息，请参阅 创建您的目录。

目前，有一个称为 QuickStartSchema 的托管架构。您可以使用 类型化链接 等构造来构建丰富的分层数据模型并建立对象之间的关系。然后，您可以通过遍历该层次结构来查询数据中的任何信息。

QuickStartSchema 托管架构由以下 JSON 表示：

```
QuickStartSchema: {
    "facets": {
        "DynamicObjectFacet": {
            "facetStyle": "DYNAMIC"
        },
        "DynamicTypedLinkFacet": {
            "facetAttributes": {
                "DynamicTypedLinkAttribute": {
                    "attributeDefinition": {
                        "attributeRules": {},
                        "attributeType": "VARIANT",
                        "isImmutable": false
                    },
                    "requiredBehavior": "REQUIRED_ALWAYS"
                }
            },
```

```
            "identityAttributeOrder": [
                "DynamicAttribute"
            ]
        }
    }
}
```

QuickStartSchema ARN

`QuickStartSchema` 托管架构使用以下 ARN：

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

例如，您可以使用此 ARN 创建一个名为 `ExampleDirectory` 的目录，如下所示：

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

分面样式

您可以在任何给定分面定义两种不同的样式：`Static` 和 `Dynamic`。

静态分面

如果您拥有目录的数据模型的所有详细信息，如包含其数据类型的属性列表，并且您还希望定义必填字段或唯一字段等属性的约束条件，静态分面是最佳选择。Cloud Directory 将在创建或更改对象期间强制执行数据约束和规则检查。

动态分面

如果您需要灵活地更改属性数或更改要存储在属性内的数据值，您可以使用动态分面。在创建或更改对象期间，Cloud Directory 不会强制执行任何数据约束和规则检查。

在使用动态分面创建架构后，您可以定义在创建对象时所需的任何属性。Cloud Directory 将接受键/值对形式的属性并将其存储在提供的对象上。

您可以将动态分面添加到新架构或现有架构。也可以在单个架构中组合静态和动态分面，以便在目录内获得每个分面样式的优势。

在使用动态分面创建任何属性时，它们将创建为 Variant 数据类型。要存储定义为Variant数据类型，您可以使用 Cloud Directory 支持的任何基元数据类型的值，如String或者Binary。随着时间的推移，您还可以将该属性的值更改为其他数据类型。不实施数据验证。

您可以使用动态分面定义以下类型的对象：

- NODE
- LEAF_NODE
- POLICY

有关托管架构、动态方面或变体数据类型的其他详细信息，以及查看示例使用案例，请参阅[如何使用 AWS 托管架构在 Amazon Cloud Directory 上快速开发应用程序]在 Amazon Cloud Directory 博客中。

示例架构

Cloud Directory 带有适用于组织、人员和设备的示例架构。下一节列出了各种示例架构并列出了每种架构的差异。

Organizations

下表列出了组织 示例架构中包含的分面。

“Organization”分面	数据类型	Length	是否必需行为？	描述
account_id	字符串	1024	否	组织的唯一 ID
account_name	字符串	1024	否	组织的名称
organization_status	字符串	1024	否	状态，如“active”、“suspended”、“inactive”、“closed”
mailing_address (street1)	字符串	1024	否	此公司/实体的实际邮寄地址

“Organization”分面	数据类型	Length	是否必需行为？	描述
mailing_address (street2)	字符串	1024	否	此公司/实体的实际邮寄地址
mailing_address (city)	字符串	1024	否	此公司/实体的实际邮寄地址
mailing_address (state)	字符串	1024	否	此公司/实体的实际邮寄地址
mailing_address (country)	字符串	1024	否	此公司/实体的实际邮寄地址
mailing_address (postal_code)	字符串	1024	否	此公司/实体的实际邮寄地址
email	字符串	1024	否	组织的电子邮件 ID
web_site	字符串	1024	否	网站 URL
telephone_number	字符串	1024	否	组织的电话号码
description	字符串	1024	否	组织的说明

“Legal_Entity”分面	数据类型	Length	是否必需行为？	描述
registered_company_name	字符串	1024	否	法律实体名称
mailing_address (street1)	字符串	1024	否	此公司/实体的实际注册地址

“Legal_Entity”分面	数据类型	Length	是否必需行为？	描述
mailing_address (street2)	字符串	1024	否	此公司/实体的实际注册地址
mailing_address (city)	字符串	1024	否	此公司/实体的实际注册地址
mailing_address (state)	字符串	1024	否	此公司/实体的实际注册地址
mailing_address (country)	字符串	1024	否	此公司/实体的实际注册地址
mailing_address (postal_code)	字符串	1024	否	此公司/实体的实际注册地址
industry_vertical	字符串	1024	否	行业分部
billing_currency	字符串	1024	否	账单货币
tax_id	字符串	1024	否	税务识别号

Person

下表列出了人员 示例架构中包含的分面。

“Person”分面	数据类型	Length	是否必需行为？	描述
display_name	字符串	1024	否	用户的名称，适合于向最终用户显示。
first_name	字符串	1024	否	用户的教名，或大多数西方语言中的名字

“Person”分面	数据类型	Length	是否必需行为？	描述
last_name	字符串	1024	否	用户的家族名，或大多数西方语言中的姓氏
middle_name	字符串	1024	否	用户的中间名
nickname	字符串	1024	否	在现实生活中用于称呼用户的非正式方式，例如用“Bob”或“Bobby”代替“Robert”
email	字符串	1024	否	用户的电子邮件地址
mobile_phone_number	字符串	1024	否	用户的电话号码
home_phone_number	字符串	1024	否	用户的电话号码
username	字符串	1024	Y	用户的唯一标识符
profile	字符串	1024	否	作为统一资源定位符的 URI，指向表示用户在线配置文件的位置 (例如网页)。
picture	字符串	1024	否	作为统一资源定位符的 URI，指向表示用户图像的资源位置。
网站	字符串	1024	否	URL
timezone	字符串	1024	否	用户的时区
locale	字符串	1024	否	用于指示用户的默认位置，以便用于本地化币种、日期时间格式或数字表示等项目。
address (street1)	字符串	1024	否	此用户的实际邮寄地址。
address (street2)	字符串	1024	否	此用户的实际邮寄地址。

“Person”分面	数据类型	Length	是否必需行为？	描述
address (city)	字符串	1024	否	此用户的实际邮寄地址。
address (state)	字符串	1024	否	此用户的实际邮寄地址。
address (country)	字符串	1024	否	此用户的实际邮寄地址。
address (postal_code)	字符串	1024	否	此用户的实际邮寄地址。
user_status	字符串	1024	否	指示用户的管理状态的值

“Organization_Person”分面	数据类型	Length	是否必需行为？	描述
标题	字符串	1024	否	组织中的职务
preferred_language	字符串	1024	否	指示用户的首选书面或口语语言，通常用于选择本地化用户界面。
employee_id	字符串	1024	否	分配给人员的字符串标识符，通常是数字或字母数字
cost_center	Integer	1024	否	标识成本中心
department	字符串	1024	否	标识部门的名称
manager	字符串	1024	否	用户的经理
company_name	字符串	1024	否	标识组织的名称
company_address (street1)	字符串	1024	否	组织的实际邮寄地址
company_address (street2)	字符串	1024	否	组织的实际邮寄地址

“Organization_Person”分面	数据类型	Length	是否必需行为？	描述
company_address (city)	字符串	1024	否	组织的实际邮寄地址
company_address (state)	字符串	1024	否	组织的实际邮寄地址
company_address (country)	字符串	1024	否	组织的实际邮寄地址
company_address (postalCode)	字符串	1024	否	组织的实际邮寄地址

Device

下表列出了设备 示例架构中包含的分面。

“Device”分面	数据类型	Length	是否必需行为？	描述
device_id	字符串	1024	否	由字母数字组成的唯一设备 ID
name	字符串	1024	否	设备的友好名称
description	字符串	1024	否	设备的说明
X.509_certificates	字符串	1024	否	X.509 证书
device_version	字符串	1024	否	设备版本
device_os_type	字符串	1024	否	设备上的操作系统
device_os_version	字符串	1024	否	设备上的操作系统版本号
serial_number	字符串	1024	否	设备的序列号

“Device”分面	数据类型	Length	是否必需行为？	描述
device_status	字符串	1024	否	状态的设备 (如 active、not_active、suspended、shutdown、off)

自定义架构

创建自定义架构的第一步是精确定义必须进行索引的字段。这些必需字段组成架构骨架元素，供您添加自己的字段。将每个字段的名称和类型 (如 string、integer、Boolean) 映射到对象的结构。您可以定义具有类型和约束的架构，然后将它们应用于目录。Once defined, Cloud Directory performs validation for attributes.

有关更多信息，请参阅 创建架构。

属性引用

Amazon Cloud Directory 分面包含属性。属性可以是属性定义或属性引用。属性定义是声明其名称和基元类型 (字符串、二进制、布尔值、日期时间或数字) 的属性。它们还可以选择性地声明必需行为、默认值、不可变标志和属性规则 (如最小/最大长度)。

属性引用是从其他预先存在的属性定义派生其基元类型、默认值、不可变标志和属性规则的属性。属性 (attribute) 引用没有自己的基元类型、默认值、不可变标志或规则，因为这些属性 (property) 来自目标属性 (attribute) 定义。

属性引用可以覆盖目标定义的必需行为 (这方面的更多详细信息请参阅下文)。

创建属性引用时，只需提供属性名称和目标属性定义 (其中包括目标属性定义的分面名称和属性名称)。属性引用不能引用其他属性引用。此外，属性引用当前不能以不同架构中的属性定义为目标。

如果对象的两个或更多属性需要引用相同的存储位置，可以使用属性引用。例如，假设一个对象应用了 User 分面和 EnterpriseUser 分面。User 分面具有 FirstName 属性定义，而 EnterpriseUser 分面具有指向 User.FirstName 的属性引用。两个 FirstName 属性都引用对象上的相同存储位置，因此对 User.FirstName 或 EnterpriseUser.FirstName 进行的任何更改都具有相同效果。

API 示例

以下示例演示如何通过 Cloud Directory API 使用属性引用。在此示例中，基本分面包含属性定义，另一个分面包含引用基本分面中的属性的属性。请注意，引用属性可以标记为 Required，而基本分面为 Not Required。

```
   // 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 示例:

以下示例演示如何在 JSON 模型中使用属性引用。此模型表示的架构与上面的模型相同。

```
{
 "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"
    }
}
```

属性引用注意事项

属性引用必须以相同架构中预先存在的属性定义为目标。

- 属性引用可以将相同分面或不同分面中预先存在的属性定义作为目标。
- 属性引用不能以其他属性引用为目标。
- 删除所有引用之前，无法删除包含的属性定义是其他分面的属性引用目标的分面。

可以通过创建对象或向现有对象应用分面，采用与使用传统属性定义相同的方式来使用属性引用。

> **Note**
>
> 可以应用引用了其他分面的分面，但是无需直接应用目标分面。未应用目标分面时，不会对属性引用的行为进行更改。(仅当希望目标分面上的其他属性在对象上存在时，才需要应用该分面。)

设置属性引用值

要更改属性的值时，您可以调用 UpdateObjectAttributes API 操作。更新 (或删除) 对象上的定义或任何引用了该定义的其他引用具有相同效果。

获取属性引用值

您可以调用 ListObjectAttributes API 操作来检索存储别名。此调用返回元组的列表，其中每个元组都包含一个属性键及其关联值。属性键对应于对象上存在的存储别名的列表。

> **Note**
>
> 可以为未显式应用于对象的分面返回属性键。当属性引用以未应用于对象的分面为目标时，可能发生这种情况。

例如，假设有 User 分面和 EnterpriseUser 分面。EnterpriseUser.FirstName 属性引用 User.FirstName。然后将 User 和 EnterpriseUser 分面都应用于一个对象，将 User.FirstName 设置为 Robert，然后将 EnterpriseUser.FirstName 设置为 Bob。调用 ListObjectAttributes 时，只会看到“User.FirstName = Bob”，因为两个 FirstName 属性只有一个存储别名。

将索引与属性引用结合使用

只能使用属性定义 (而不是引用) 创建索引。列出索引不会为属性引用返回属性键。但是，它会为作为索引对象上存在的引用的目标的任何属性定义返回属性键。换句话说，在索引层，仅将属性引用视为属性的备选标识符 (在运行时解析为正确的属性定义标识符)。

例如，假设分面 User 属性 FirstName 具有索引。您附加一个只应用了 EnterpriseUser 分面的对象。随后将该对象的 EnterpriseUser.FirstName 属性值设置为 Bob。最后调用 ListIndex 操作。结果仅包含“User.FirstName = Bob”。

属性引用的必需行为

属性引用可以具有与其目标属性定义不同的必需行为。因此，可以是基本定义可选，而对该定义的引用是必需的。如果对象有一个基本定义以及一个或多个对该基本定义的引用，基本定义和所有引用都必须遵循所有相关属性间存在的最强必需行为。

- 与属性定义一样，在创建对象或向现有对象添加分面时，必须为任何必需属性定义提供值。
- 为方便起见，如果对象的多个属性引用相同存储位置时，您只需为该存储位置的属性之一提供值。
- 同样，如果为相同存储位置提供多个值，则这些值必须相等。

属性规则

规则描述属性类型的允许值并约束对任何特定属性所允许的值。创建分面时，必须在属性定义中指定规则。Cloud Directory 支持以下规则类型：

- 字符串长度
- 二进制长度
- 集的字符串
- 数字比较

字符串长度

约束字符串属性值的长度。

允许的规则参数键：min、max

允许的规则参数值：数字

二进制长度

约束二进制属性值的字节数组长度。

允许的规则参数键：min、max

允许的规则参数值：数字

集的字符串

将字符串属性的值约束为允许的指定字符串集。

允许的规则参数键：allowedValues

允许的规则参数值：字符串集，其中每个字符串都进行 UTF-8 编码

允许的值使用逗号分隔，可以包含在引号内。当允许的值包含逗号时，这很有用。例如：

- One,two,three = 匹配 One、two 或 three
- "with,comma","withoutcomma" = 匹配“with,comma”或“withoutcomma”
- with"quote,withoutquote 匹配“with"quote”或“withoutquote”

数字比较

约束对数字属性允许的数值。

允许的规则参数键：min、max

允许的规则参数值：数字

格式规范

Cloud Directory 架构会向数据目录中的数据添加结构。Cloud Directory 提供了两种机制，供您定义架构。开发人员可以使用特定 API 操作构建架构，也可以使用架构上传功能完全上传架构。架构文档可以通过 API 调用或通过控制台进行上传。本节介绍上传整个架构文档时要使用的格式。

JSON 架构格式

架构文档是采用以下整体格式的 JSON 文档。

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

架构文档包含分面名称到分面的映射。每个分面进而包含一个映射，其中包含属性。架构中的所有分面名称都必须唯一。分面中的所有属性名称都必须唯一。

属性 JSON 子部分

分面包含属性。每个属性都定义可以存储在属性中的值的类型。以下 JSON 格式描述一个属性。

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

必须提供属性定义或属性引用。有关每个内容的更多信息，请参阅相关子部分。

必需行为字段指示此属性是否为必需的。必须提供此字段。可能值如下所示：

- REQUIRED_ALWAYS：创建对象或是向对象添加分面时必须提供此属性。无法删除此属性。
- NOT_REQUIRED：此属性可以存在，也可以不存在。

属性定义子部分

属性定义与属性值关联的类型和规则。下面的 JSON 布局对格式进行说明。

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

默认值子部分

精确指定以下默认值之一。长数值和布尔值应在引号外部提供 (作为其相应的 Javascript 类型而不是字符串)。二进制值使用 URL 安全 Base64 编码字符串提供 (如 RFC 4648 中所述)。日期时间采用自纪元 (1970 年 1 月 1 日 00:00:00 UTC) 以来的毫秒数提供。

```
{
 "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
}
```

属性规则子部分

属性规则对属性值定义约束。可以为每个属性定义多个规则。属性规则包含规则的规则类型和参数集。可以在属性规则 一节中找到更多详细信息。

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

```
}
```

属性引用子部分

属性引用是高级功能。它们使多个分面可以共享属性定义和存储的值。有关更多信息，请参阅属性引用部分。可以使用以下模板在 JSON 架构中定义属性引用。

```
{
 "targetSchemaArn": "schema ARN"
 "targetFacetName": "facet name"
 "targetAttributeName": "attribute name"
}
```

架构文档示例

以下为显示有效 JSON 格式设置的架构文档示例。

> **Note**
>
> 在 allowedValues 字符串中表示的所有值必须以逗号分隔并且没有空格。例如："SENSITIVE,CONFIDENTIAL,PUBLIC"。

基本架构文档

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

具有类型化链接的架构文档

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

目录对象

开发人员使用可扩展架构对目录对象进行建模以自动强制实施数据正确性约束，从而可以更简单地进行编程。Amazon Cloud Directory 提供基于已定义索引属性的丰富信息查找，从而可在目录树中实现快速树遍历和搜索。Cloud Directory 数据在静态和传输态下会进行加密。

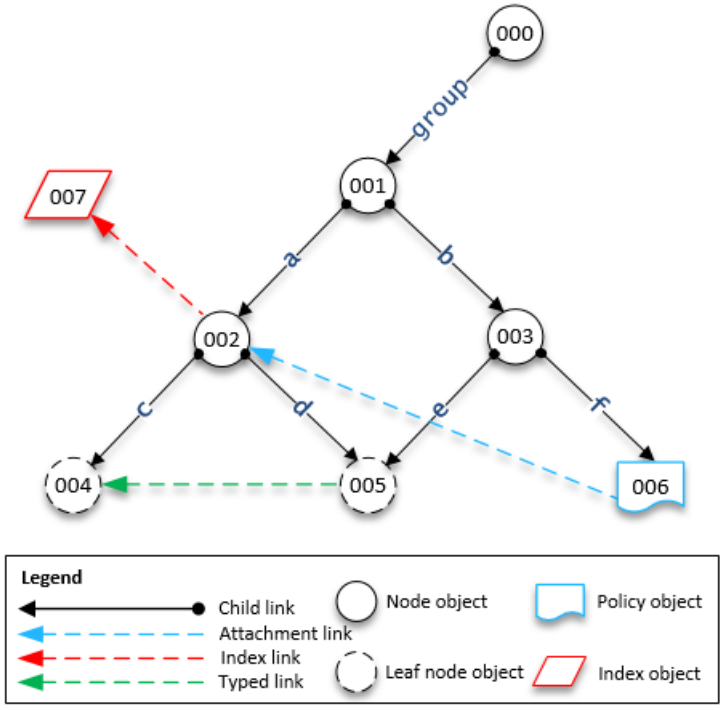
对象是 Cloud Directory 的基本元素。每个对象都具有通过对象标识符指定的全局唯一标识符。对象是零个或更多分面及其属性键和值的集合。对象可以通过单个已应用架构中的一个或多个分面或是通过多个已应用架构的分面进行创建。在对象创建过程中，必须指定所有必需属性值。对象可以拥有有限数量的分面。有关更多信息，请参阅 Amazon Cloud Directory 限制。

对象可以是常规对象、策略对象或索引对象。对象还可以是节点对象或叶节点对象。对象的类型从附加到它的分面的对象类型进行推断。

主题

Links

链接是定义关系的两个对象之间的指向性边缘。Cloud Directory 当前支持以下链接类型。



子链接

子链接可在它连接的对象之间创建父子关系。例如，在上图中，子链接 b 连接对象 001 和 003。子链接在 Cloud Directory 中定义层次结构。在参与定义链接指向的对象的路径时，子链接会获得名称。

附加链接

附加链接将叶节点策略对象应用于另一个叶节点或节点对象。附加链接不定义 Cloud Directory 的层次结构。例如，在上图中，附加链接对节点对象 006 应用存储在策略叶节点对象 002 中的策略。每个对象都可以附加多个策略，但是不能附加任何给定策略类型的多个策略。

索引链接

索引链接提供基于索引对象和已定义索引属性的丰富信息查找，从而可在目录树中实现快速树遍历和搜索。从概念上讲，索引类似于具有子节点：会根据索引属性来标记指向索引节点的链接，而不是在附加子级时给定标签。但是，索引链接不是父–子关系的边缘，有自己的一组枚举 API 操作。有关更多信息，请参阅 索引和搜索。

类型化链接

通过类型化链接，您可以在 Cloud Directory 中的层次结构内或跨层次结构的对象之间建立关系。然后，您可以使用这些关系来查询信息，如哪些用户拥有“xyz”或哪些设备归用户“abc”所有。

您可以使用类型化链接，为目录中不同对象之间的关系建模。例如，在上图中，请考虑对象 004 (表示用户) 与对象 005 (表示设备) 之间的关系。

我们可以使用类型化链接为两个对象之间的所有权关系建模。我们可以将属性添加到类型化链接，用于表示采购成本或者设备是租赁还是采购的。与类型化链接关联的属性有两种类型：

- 基于身份的属性 – 类型化链接的一个属性，可将类型化链接与其他链接（例如，子链接、附加链接、索引链接）区分开来。每个类型化链接分面定义一组有序的身份属性。类型化链接的身份是源对象 ID、分面标识符（类型）、其身份属性的值（由其分面定义）以及目标对象 ID。标识符在单个目录中必须是唯一的。
- 可选属性 - 存储有关类型化链接的跟踪特性的属性，这些跟踪特性与链接的身份无关。例如，可选属性可能标识第一次建立类型化链接的日期或其上次修改时间。

对于对象，您必须使用 `CreateTypedLinkFacet` API 创建类型化链接分面，用于定义类型化链接结构及其属性。类型化链接分面需要唯一的分面名称以及一组与链接关联的属性。在设计类型化链接结构时，您可以对类型化链接分面定义一组有序的属性。要查看类型化示例架构，请参阅具有类型化链接的架构文档。

当您需要执行以下任何操作时，可以使用类型化链接属性：

- 允许筛选传入或传出的类型化链接。有关更多信息，请参阅 类型化链接列表。
- 表示两个对象之间的关系。
- 跟踪有关类型化链接的管理数据，例如创建此链接的日期。

在确定类型化链接是否适合您的使用案例时，请考虑以下情况：

- 类型化链接不能用于基于路径的对象规范。相反，您必须使用 `ListOutgoingTypedLinks` 或 `ListIncomingTypedLinks` API 操作选择类型化链接。
- 类型化链接不参与到 `LookupPolicy` 或 `ListObjectParentPaths` API 操作中。
- 相同两个对象之间具有相同方向的类型化链接不能具有相同属性值。这可以帮助避免在相同对象之间出现重复的类型化链接。
- 当您要添加可选信息时，可以使用其他属性。
- 所有身份属性值的组合大小限制为 64 个字节。有关更多信息，请参阅 Amazon Cloud Directory 限制。

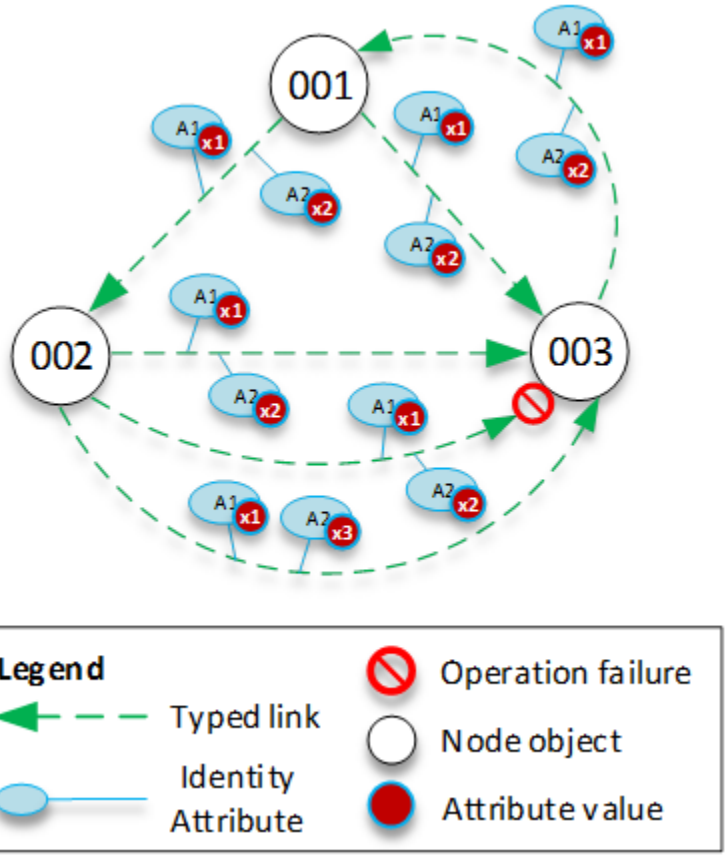
相关 Cloud Directory 博客文章

- 使用 Amazon Cloud Directory 类型化链接跨层次结构创建和搜索关系

类型化链接身份

身份唯一地定义了两个对象之间是否可以存在类型化链接。例外情况是当您使用完全相同的属性值在一个方向上连接两个对象时。属性必须配置为 REQUIRED_ALWAYS。

从不同类型化连接分面创建的类型化链接从不彼此冲突。例如，考虑以下图表：



- 对象 001 具有指向不同对象 (002 和 003) 类型化链接和属性 (A1 和 A2)，属性具有相同的属性值 (x1 和 x2)。此操作将会成功。
- 对象 002 和 003 之间有类型化链接。此操作将失败，因为对象之间不能存在两个具有相同方向和相同属性的类型化链接。
- 对象 001 和 003 之间存在具有相同属性的类型化链接。但是，由于链接指向不同方向，此操作将成功。

- 对象 002 和 003 之间具有类型化链接，其中 A1 具有相同的值，但 A2 的值不同。类型化链接身份会考虑所有属性，所以此操作将成功。

类型化链接规则

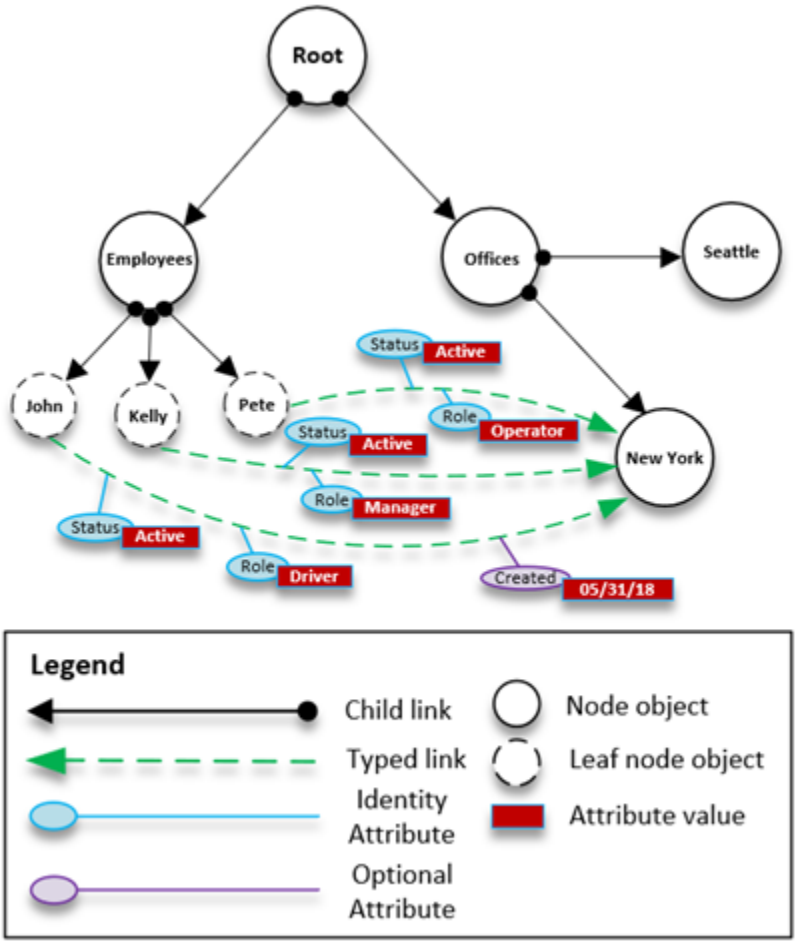
在您希望向链接属性添加限制时，可以将规则添加到类型化链接属性。这些规则等同于对象属性上的规则。有关更多信息，请参阅 属性规则。

类型化链接列表

Cloud Directory 提供了 API 操作，可用于从对象选择传入或传出类型化链接。可以选择类型化链接的特定子集，而不是在每个类型化链接上迭代。您还可以指定特定类型化链接分面，用于仅筛选具有该类型的类型化链接。

您可以根据在类型化链接分面上定义属性的顺序来筛选类型化链接。您可以为多个属性提供范围筛选器。为类型化链接选择提供范围时，任何不精确的范围都必须在末尾指明。系统假定任何未指定范围的属性将匹配整个范围。筛选器按照在类型化链接分面上定义的属性顺序来解释，而不是按提供给任意 API 调用的顺序。

例如，在下图中，考虑用于存储有关员工及其能力信息的 Cloud Directory。



假设我们使用名为 EmployeeCapability 的类型化链接对员工能力进行建模，该链接配置有三个字符串属性：Status、Role 和 Created。ListIncomingTypedLinks 和 ListOutgoingTypedLinks API 操作支持以下筛选器。

- 分面 = EmployeeCapability，状态 = Active，角色 = Driver
  - 选择角色是驱动者的活跃员工。此筛选器包含两个完全匹配。
- 分面 = EmployeeCapability，状态 = Active，角色 = Driver，创建日期 = 05/31/18
  - 选择其角色是司机且其分面在 2018 年 5 月 31 日或之后创建的活跃员工。
- 分面 = EmployeeCapability，状态 = Active
  - 选择所有活跃员工。
- 分面 = EmployeeCapability，状态 = Active，角色 = A 到 M
  - 选择角色以 A 到 M 开头的所有活跃员工。
- 分面 = EmployeeCapability
  - 这会选择所有 EmployeeCapability 类型的类型化链接。

不支持以下筛选器：

- 分面 = EmployeeCapability，状态介于 A 到 C，角色 = Driver
  - 由于任何范围都必须显示在筛选器的末尾，因此不允许此筛选器。
- 分面 = EmployeeCapability，角色 = Driver
  - 由于隐式状态范围不是精确匹配，并且没有显示在范围列表的末尾，因此不允许此筛选器。
- 状态 = Active
  - 由于未指定类型化链接分面，因此不允许此筛选器。

类型化链接架构

您可以通过两种方法创建类型化链接分面。您可以从单独的 API 调用管理类型化链接分面，包括 CreateTypedLinkFacet、DeleteTypedLinkFacet 和 UpdateTypedLinkFacet。您也可以在单个 PutSchemaFromJson API 调用中上传表示架构的 JSON 文档。有关更多信息，请参阅 JSON 架构格式。要查看类型化示例架构，请参阅具有类型化链接的架构文档。

在架构开发声明周期的不同阶段允许的更改类型，类似于对象分面操作所允许的更改。开发状态中的架构支持任意更改。已发布状态的架构不可变，不支持任何更改。已应用到数据目录的架构只允许特定更改。在已应用的类型化链接分面上设置了顺序和属性之后，不能更改该属性。

另外两个 API 操作列表分面及其属性：

- ListTypedLinkFacetAttributes
- ListTypedLinkFacetNames

类型化链接交互

一旦创建了类型化链接分面，您就可以使用类型化链接开始进行创建和交互。要附加和分离类型化链接，请使用 AttachTypedLink 和 DetachTypedLink API 操作。

TypedLinkSpecifier 是一个结构，包含用于唯一标识类型化链接的所有信息。在该结构中，您可以找到 TypedLinkFacet、SourceObjectID、DestinationObjectID 和 IdentityAttributeValues。这些内容用于唯一指定所要操作的类型化链接。AttachTypedLink API 操作返回类型化链接说明符，而 DetachTypedLink API 操作接受说明符作为输入。与此类似，ListIncomingTypedLinks 和 ListOutgoingTypedLinks API 操作提供类型化链接说明符作为输出。您也可以从头开始构造类型化链接说明符。类型化链接相关 API 操作的完整列表包括以下内容：

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

> **Note**
>
> 不支持属性引用和更新类型化链接。要更新类型化链接，您必须删除它并添加更新后的版本。

范围筛选器

多种 Cloud Directory 列表 API 允许以范围的形式来指定筛选器。这些筛选器使您能够高效地选择附加到指定节点的链接子集。

范围通常作为映射 (键-值对数组) 提供，其键是属性标识符，其值是对应的范围。这可以筛选其身份由一个或多个属性组成的链接。例如，一个 TypedLink 设置为对 Role 关系进行建模，用于确定同时具有 RoleType 和 Authorizer 属性的权限。然后，ListOutgoingTypedLinks 调用可以指定范围，将结果筛选为 RoleType:"Admin" 和 Authorizer:"Julia"。用于筛选单个列表请求的范围的映射必须只包含定义链接身份的属性 (索引的 OrderedIndexedAttributeList 或 TypedLink 的 IdentityAttributeOrder)，但不必包含全部属性的范围。缺少的范围将自动使用跨所有可能值 (从 FIRST 到 LAST) 的范围填充。

如果您将每个属性视为定义独立的平面值域，则范围结构定义该域中的两个逻辑点 (起点和终点)，并且范围匹配这些点之间所有可能的点。范围结构的 StartValue 和 EndValue 定义这两个点的基础，“modes”进一步将其细化为指示在范围中包括还是排除每个点自身。在以上 RoleType:"Admin" 示例中，RoleType 属性的值为“Admin”，模式均为“INCLUSIVE”(编写为 ["Admin" to "Admin"])。对于 ListIndex 调用，其索引在用户分面的 LastName 上定义，其筛选器可以使用

StartValue="D"，StartMode=INCLUSIVE，EndValue:"G"，EndMode:EXCLUSIVE 用于将列表缩小到以 D、E 或 F 开头的名称。

范围的起点必须早于或等于终点。如果 EndValue 在 StartValue 之前，将返回错误。该值还必须具有与所筛选的属性相同的基元类型，字符串值对应于字符串属性，整数值对应于整数属性，以此类推。例如，StartValue="D"，StartMode=EXCLUSIVE，EndValue="D"，EndMode=INCLUSIVE 无效，因为终点包括终点值，而起点是在该值之后。

起点或终点可以使用三种特殊模式。以下模式无需指定对应值字段，因为它们自身暗示了位置。

- FIRST - 位于域中所有可能值之前。在用于起点时，此项匹配从域起点直到终点的所有可能值。在用于终点时，域中的任何值都不与范围匹配。
- LAST - 位于域中所有可能值之后。在用于终点时，此项匹配起点之后的所有可能值，包括缺少的值。在用于起点时，域中的任何值都不与范围匹配。
- LAST_BEFORE_MISSING_VALUES - 此模式仅对可选属性有用，此时可以忽略值 (请参阅 缺失值)。它对应于缺失值和实际域值之间的点。在用于终点时，此项匹配起点之后的所有非缺失域值。当用于起点时，它排除所有非缺失域值。如果该属性必需，则此模式等同于 LAST，因为这里可能没有缺失值。

多个范围限制

Cloud Directory 对有多个属性的模式作出限制，以便确保高效、低延迟的请求处理。每个链接具有多个用于标识身份的属性，按照明确定义的顺序指定它们。例如，以上 Role 示例定义 RoleType 属性为最重要，Authorizer 属性最不重要。List 请求只能指定一个“符合条件”范围，可以为 1) 单个值或 2) 跨所有可能值 (可以有多个范围与这两个要求匹配)。任何范围，其属性比符合条件范围属性更重要时，必须指定单个值，任何较不重要的范围必须跨所有可能值。在 Role 示例中，筛选器集 (RoleType:"Admin", Authorizer:["J" to "L"]) (单值 + 符合条件范围)，(RoleType:["Admin" to "User"]) (符合条件范围 + 隐式跨越范围)，以及 (RoleType:[FIRST to LAST]) (两个跨越范围，一个隐式) 均为有效筛选器集的示例。(RoleType:[FIRST to LAST], Authorizer:"Julia") 是无效的集合，因为跨越范围重要性高于单值范围。

填充范围结构时有一些有用的模式，包括：

匹配单个值

为 StartValue 和 EndValue 均指定值，并将模式均设置为“INCLUSIVE”。

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

匹配一个前缀

指定前缀为 INCLUSIVE 模式的 StartValue，前缀后面的第一个值是 EXCLUSIVE 模式的 EndValue。

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

大于某个值的筛选

指定 EXCLUSIVE 模式的 StartValue 的值，以及 EndMode 的 LAST 值 (或者，在适用时指定 LAST_BEFORE_MISSING_VALUES 以排除缺少的值)。

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

等于或小于某个值的筛选

指定 INCLUSIVE 模式的 EndValue 的值，指定 FIRST 作为 StartMode。

缺失值

当某个属性在架构中标记为可选时，可以"缺少"该值，因为在附加分面时不必提供该值，或者接下来可能会删除属性。如果具有此类缺少值的对象附加到了索引上，则索引链接仍然存在，但移动到链接集的末尾。`ListIndex` 调用首先返回索引属性均存在的任意链接，然后返回缺少一个或多个属性的链接。这大致类似于关系数据库的 NULL 值，这些值在非 NULL 值之后排序。您可以通过选择 LAST 或 LAST_BEFORE_MISSING_VALUES 模式来指定某个范围中是否包含这些缺少的值。例如，您向 ListIndex 调用提供了筛选器，通过范围 [LAST_BEFORE_MISSING_VALUES to LAST] 进行筛选，从而只返回某个索引中缺少的值。

访问对象

可以通过路径或通过 `objectIdentifier` 访问目录中的对象。

Path (路径)— Cloud Directory 树中的每个对象都可以通过描述如何访问它的路径名称进行标识和查找。路径从目录的根 (上图中的节点 `000`) 开始。路径表示法以使用斜杠 (/) 标记的链接开头，后跟通过路径分隔符 (也是斜杠) 分隔的子链接，直到到达路径的最后一部分。例如，上图中的对象 `005` 可以使用路径 `/group/a/d` 进行标识。多个路径可能会标识一个对象，因为作为叶节点的对象可以具有多个父级。以下路径也可以用于标识对象 `005`：/group/b/e

对象标识符— 目录中的每个对象都具有唯一全局标识符，即。`ObjectIdentifier`。`ObjectIdentifier`作为`CreateObject`API 调用。还可以使用 `ObjectIdentifier` API 调用提取 `GetObjectInformation`。例如，要提取对象 `005`

的对象标识符，可以通过将对象引用作为指向对象的路径 (即 group/b/e 或 group/a/d) 来调用 GetObjectInformation。

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

填充对象

可以使用 AddFacetToObject API 调用将新分面添加到对象。对象的类型基于附加到对象的分面来确定。目录中的对象附加基于对象的类型进行工作。附加对象时，请注意以下规则：

- 叶节点对象不能有子级。
- 节点对象可以有多个子级。
- 策略类型的对象不能有子级，可以有零个或一个父级。

更新对象

可以通过多种方式更新对象：

1. 使用 UpdateObjectAttributes 操作可更新对象上的各个分面属性。
2. 使用 AddFacetToObject 操作可将新分面添加到对象。
3. 使用 RemoveFacetFromObject 操作可从对象删除现有分面。

删除对象

附加的对象必须满足特定条件，您才能从目录中将其删除：

1. 必须将对象从树分离。仅当对象没有任何子级时才能分离对象。如果对象有子级，则必须先分离所有子级。
2. 仅当删除了分离的对象上的所有属性时，才能删除该对象。通过删除附加到某个对象的每个分面，可以删除该对象上的属性。可以通过调用 GetObjectInformation 来提取附加到对象的分面的列表。
3. 对象还必须没有父级、没有策略附加并且没有索引附加。

因为对象必须从树中完全分离才能进行删除，所以必须使用对象标识符才能将其删除。

查询对象

本部分讨论用于在目录中查询对象的各种相关元素。

目录遍历

由于 Cloud Directory 是一种树，因此可以使用ListObjectChildrenAPI 操作或从下而上使用ListObjectParentsAPI 操作

策略查找

如果提供对象引用，则 LookupPolicy API 操作可采用从上向下的方式，沿其指向根的路径返回附加的所有策略。任何不指向根的路径都会被忽略。返回所有策略类型对象。

如果对象是叶节点，则它可以具有多个指向根的路径。此调用对于每次调用仅返回一个路径。要提取其他路径，请使用分页标记。

索引查询

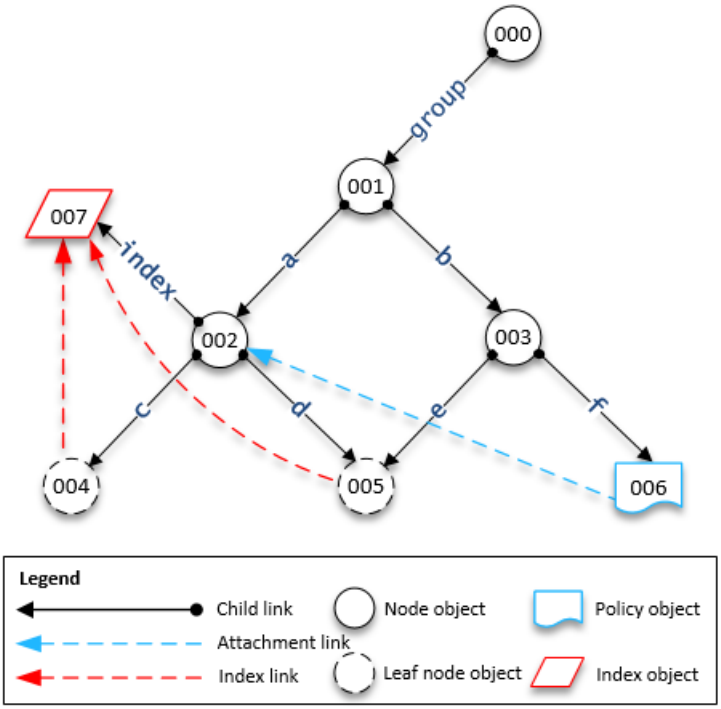
Cloud Directory 通过使用以下范围，支持丰富的索引查询功能：

- FIRST - 从第一个索引属性值开始。开始属性值是可选的。
- LAST - 返回直到索引结束的属性值，包括缺失值。结束属性值是可选的。
- LAST_BEFORE_MISSING_VALUES - 返回直到索引结束的属性值，不包括缺失值。
- INCLUSIVE - 包括所指定的属性值。
- EXCLUSIVE - 不包括所指定的属性值。

父路径列表

使用 ListObjectParentPaths API 调用可以为任何类型的对象 (节点、叶节点、策略节点、索引节点) 检索所有可用父路径。如果需要计算对象的所有父级，此 API 操作十分有用。该调用从目录根返回所有对象，直到请求的对象。它还基于用户定义的 MaxResults 返回路径数 (如果存在多个指向父级的路径)。返回的路径和节点的顺序在多个 API 调用间是一致的，除非删除或移动了对象。会从目标对象中忽略不指向目录根的路径。

举例来说明此工作原理，假设一个目录的对象层次结构类似于下面的示意图。



编号形状表示不同对象。该对象与目录根 (000) 之间的箭头数表示完整路径，会在输出中进行表示。下表显示对层次结构中特定叶节点对象进行的查询的请求和响应。

针对对象的示例查询

请求	响应
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 **Note** 在此示例中，对象 005 将节点 002 和 003 作为父级。此外，因为 MaxResults 为 2，所以两个路径都会在列表中显示对象。

请求	响应
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** 在此示例中，对象 005 将节点 002 和 003 作为父级。此外，因为 MaxResults 为 1，所以进行多个包含页面标记的分页调用来获取所有路径及对象列表。
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

一致性级别

Amazon Cloud Directory 是分布式目录存储。数据分布到不同可用区中的多个服务器。成功的写入请求会更新所有服务器上的数据。数据最终在所有服务器上可用 (通常在一秒内)。为了帮助用户使用服务，Cloud Directory 为读取操作提供了两个一致性级别。本部分介绍 Cloud Directory 的不同一致性级别和最终一致性质。

读取隔离级别

从 Cloud Directory 读取数据时，必须指定要从中进行读取的隔离级别。不同隔离级别在延迟与数据新鲜度之间进行权衡。

- 最终— 快照隔离级别读取立即可用的任何数据。它提供任何隔离级别的最低延迟。 它还提供目录中数据的可能较旧视图。EVENTUAL 隔离不提供先写后读一致性。这意味着不保证在写入数据之后立即能够读取数据。
- 可序列化— 可序列化隔离级别提供 Cloud Directory 所提供的最高一致性级别。在 SERIALIZABLE 隔离级别进行的读取可确保从任何成功写入接收数据。如果对请求的数据以及更改尚不可用的数据进行了更改，则系统使用 `RetryableConflictException` 拒绝请求。我们建议重试这些异常 (请参阅以下部分)。成功重试时，SERIALIZABLE 读取可提供先写后读一致性。

写入请求

Cloud Directory 可确保多个写入请求不会同时更新相同对象。如果发现在对相同对象执行两个写入请求，则其中一个操作会失败，并出现 `RetryableConflictExceptionx`。我们建议重试这些异常 (请参阅以下部分)。

> **Note**
>
> 在写入操作期间收到的 `RetryableConflictException` 响应无法用于检测争用条件。对于被证明存在这种情况的使用案例，不保证始终会发生异常。是否发生异常取决于在内部处理每个请求的顺序。

RetryableConflictExceptions

在对相同对象执行写入之后使用 SERIAVE 隔离级别执行写入操作或读取操作时，Cloud Directory 使用进行响应。`RetryableConflictException`。此异常指示 Cloud Directory 服务器尚未处理前一次写入的内容。这些情况是临时的，会快速地自行补救。请务必注意，`RetryableConflictException` 不能用于检测任何类型的先写后读一致性。并非特定使用案例才会导致此异常。

我们建议配置 Cloud Directory 客户端以重试`RetryableConflictException`。此配置可在操作过程中提供无错误的行为。以下示例代码演示如何在 Java 中进行此配置。

```
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);
 }
}
```

索引和搜索

Amazon Cloud Directory 支持两种索引方法：基于值和基于类型。基于值的索引是最常见的形式。使用这种索引，您可以根据对象属性的值，在目录中索引和搜索对象。使用基于类型的索引，您可以根据对象类型，在目录中索引和搜索对象。分面有助于定义对象类型。有关架构和分面的更多信息，请参阅Schemas和Facets。

通过 Cloud Directory 中的索引可根据对象的属性值或分面值简单列出其他对象。每个索引在创建时都定义为与特定命名属性或分面结合使用。例如，可在“Person”分面的“email”属性上定义索引。索引是一类对象，这意味着客户端可以根据应用程序逻辑的需要灵活地创建、修改、列出和删除它们。

从概念上说，索引类似于带有子级的节点，会根据索引属性来标记指向索引节点的链接，而不是在附加子级时给定标签。但是，索引链接不是父–子关系的边缘，有自己的一组枚举 API 操作。

务必知道的是，Cloud Directory 中的索引不像其他系统中那样自动填充，而是使用 API 调用直接将对象附加到索引或从索引分离对象。虽然这样做会增加一些工作量，但允许您灵活地定义不同的索引范围。例如，您可以定义一个只跟踪特定节点的直接子级的索引。或者，您可以定义索引来跟踪本地根目录中给定分支内的所有对象，如某一部门的所有节点，您可以同时执行这两个任务。

主题

- 索引生命周期
- 基于分面的索引
- 唯一索引与不唯一索引

索引生命周期

您可以使用以下 API 调用帮助开发索引的生命周期。

1. 您可以用 `CreateIndex` API 调用创建索引。您提供一个索引定义结构，用于描述该索引要跟踪的附加对象的属性。该定义还指明索引是否应强制唯一性。结果是新索引的对象 ID，应其像其他任何对象一样立即附加到层次结构中。例如，这可以是专用于保存索引的分支。
2. 用 `AttachToIndex` API 调用将对象手动附加到索引。然后，该索引会自动跟踪每个附加对象的已定义属性的值。
3. 要使用索引以更具效率地枚举搜索对象，请调用 `ListIndex` 并指定您感兴趣的值范围。
4. 使用 `ListAttachedIndices` API 调用来枚举附加到给定对象的索引。
5. 使用 `DetachFromIndex` API 调用手动从索引中删除对象。

6. 从索引中分离所有对象之后，可以使用 DeleteObject API 调用删除该索引。

除了对所有对象使用的空间进行限制外，对目录中的索引数没有任何限制。索引及其附件确实会占用空间，但与节点和父-子链接所占用的空间相似。对于可以附加到给定对象的索引数有限制。有关更多信息，请参阅 Amazon Cloud Directory 限制。

基于分面的索引

通过基于分面的索引和搜索，您可以只搜索目录的子集，从而优化目录搜索。为此，您可以使用架构分面。例如，您不必在目录中的所有用户对象上进行搜索，而是可以只搜索包含员工分面的用户对象。这可以有效地帮助减少查询延迟时间以及所检索的数据量。

使用基于分面的索引，您可以使用 Cloud Directory 索引 API 操作来创建并将索引附加到对象的分面。您还可以列出索引结果，然后根据特定分面筛选这些结果。这可以将搜索范围缩小到只包含特定类型分面的对象，从而有效地缩短查询时间和数据量。

用于 “facets” 和 CreateIndex API 调用的 ListIndex 属性，覆盖应用到对象的分面集合。此属性仅可用于 CreateIndex 和 ListIndex API 调用。如以下示例代码中所示，架构 ARN 使用目录的区域、所有者账户和目录 ID 来引用 Cloud Directory 架构。此服务提供的架构不显示在列表中。

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

例如，以下示例代码创建基于分面的索引，特定于您的 AWS 账户和目录，您可以枚举使用分面 SalesDepartmentFacet 创建的所有对象。

> **Note**
>
> 请确保在以下所示参数中使用“facets”值。示例代码所示的 “facets” 实例引用了由 Cloud Directory 服务提供和控制的值。您可以将这些用于索引，但只能具有只读访问权限。

```
// 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")
    )))
```

唯一索引与不唯一索引

唯一索引不同于不唯一索引，其差别在于对附加到索引的对象实施索引属性值的唯一性。例如，您可能将 Person 对象填充到两个索引中，唯一索引是“email”属性，不唯一索引是“lastname”属性。lastname 索引允许多个 Person 对象附加到相同的姓氏。另一方面，`AttachToIndex` 调用目标，如果某个 Person 已经附加了相同的 email 属性，在 email 索引返回 `LinkNameAlreadyInUseException` 错

误。请注意，错误不会删除 Person 对象本身。因此，应用程序可以创建 Person，并在一个批处理请求中将其附加到层次结构和其附加到索引。这可以确保如果任何索引违反了唯一性，对象及其全部附件将自动回滚。

如何管理 Cloud Directory

此节列出了运行和维护 Cloud Directory 环境的所有过程。

主题

管理您的目录

此节介绍如何为您的 Cloud Directory 环境维护常见目录任务。

主题

创建您的目录

AWS Directory Service 要求首先应用架构，然后才能在 Amazon Cloud Directory 中创建目录。若不使用架构，则无法创建目录，通常目录会应用一个架构。但是，可以使用 Cloud Directory API 操作将其他架构应用于目录。有关更多信息，请参阅 Amazon Cloud Directory API 参考指南 中的 ApplySchema。

创建 Cloud Directory

1. 在AWS Directory Service 控制台导航窗格中，在Cloud Directory中，选择目录。
2. 选择设置 Cloud Directory。
3. UDER选择要应用于新目录的架构，键入目录的友好名称，如`User Repository`，然后选择下列选项之一：

    - 托管架构
    - 示例架构
    - 自定义架构

示例架构和自定义架构放置在开发状态（默认情况下）。有关架构状态的更多信息，请参阅架构生命周期。架构必须先转换为已发布状态，然后才能应用于目录。要使用控制台成功发布示例架构，您必须有权执行以下操作：

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

因为示例架构是 AWS 提供的只读模板，所以无法直接发布它们。相反，当选择基于示例架构创建目录时，控制台会创建所选示例架构的临时副本，并将它置于开发状态。它随后会创建该开发架构的副本，并将它置于已发布状态。发布之后，开发架构即被删除，这就是在发布示例架构时需要 `DeleteSchema` 操作的原因。

4. 选择 Next。
5. 查看目录信息并进行必要的更改。如果信息正确，请选择 Create (创建)。

删除目录

使用以下过程删除 Cloud Directory 中的目录。

> **Note**
>
> 您必须先禁用目录，然后才能删除目录。有关说明，请参阅禁用您的目录。

要删除目录

1. 在AWS Directory Service 控制台导航窗格中，在Cloud Directory中，选择目录。
2. 选择要删除的目录 ID 旁边的表格中的选项。
3. 选择 Actions。
4. 选择Delete

5. 在删除目录对话框中，通过键入目录的名称来确认操作，然后选择Delete。

禁用您的目录

使用以下过程禁用 Cloud Directory 中的目录。

禁用目录

1. 在AWS Directory Service 控制台导航窗格中，在Cloud Directory中，选择目录。
2. 选择要禁用的目录 ID 旁边的表格中的选项。
3. 选择 Actions。
4. 选择禁用

启用您的目录

使用以下过程在 Cloud Directory 中启用先前禁用的目录。

启用目录

1. 在AWS Directory Service 控制台导航窗格中，在Cloud Directory中，选择目录。
2. 选择要启用的目录 ID 旁边的表格中的选项。
3. 选择 Actions。
4. 选择Enable (启用 Gem)

管理您的架构

此节介绍如何为您的 Cloud Directory 环境维护常见架构任务。

主题

- 创建您的架构
- 删除架构
- 下载架构
- 发布架构
- 更新架构

- 升级架构

创建您的架构

Amazon Cloud Directory 支持上传符合 JSON 标准的文件以进行架构创建。要创建新架构，可以从头开始创建自己的 JSON 文件，或下载控制台中列出的现有架构之一。然后将它作为自定义架构进行上传。有关更多信息，请参阅 自定义架构。

还可以使用 Cloud Directory API 创建、删除、下载、列出、发布、更新和升级架构。有关架构 API 操作的更多信息，请参阅 Amazon Cloud Directory API 参考指南。

根据首选方法，选择以下任一过程。

创建自定义架构

1. 在AWS Directory Service 控制台导航窗格中，在Cloud Directory中，选择Schemas。
2. 创建包含所有新架构定义的 JSON 文件。有关如何对 JSON 文件设置格式的更多信息，请参阅 JSON 架构格式。
3. 在控制台中，选择上传新架构。
4. 在上传新架构对话框中，为架构键入一个名称。
5. SelectSelect file，选择您刚刚创建的新 JSON 文件，然后选择打开。
6. 选择 Upload。这会将新架构添加到架构库中，并将它置于开发状态。有关架构状态的更多信息，请参阅架构生命周期。

在控制台中基于现有架构创建自定义架构

1. 在AWS Directory Service 控制台导航窗格中，在Cloud Directory中，选择Schemas。
2. 在列出架构的表中，选择要复制的架构附近的选项。
3. 选择 Actions。
4. 选择下载架构。
5. 重命名 JSON 文件，根据需要进行编辑，然后保存文件。有关如何对 JSON 文件设置格式的更多信息，请参阅 JSON 架构格式。
6. 在控制台中，选择上传新架构，选择您刚编辑的 JSON 文件，然后选择打开。

   这会将新架构添加到架构库中，并将它置于开发状态。有关架构状态的更多信息，请参阅架构生命周期。

删除架构

可以使用以下过程删除 Cloud Directory 中的架构。

删除架构

1. 在AWS Directory Service 控制台导航窗格中，在Cloud Directory中，选择Schemas。
2. 在表中选择要删除的架构名称旁边的选项。
3. 选择 Actions。
4. 选择Delete
5. 在删除架构对话框中，通过选择Delete。

下载架构

使用以下过程下载架构。

下载架构

1. 在AWS Directory Service 控制台导航窗格中，在Cloud Directory中，选择Schemas。
2. 在表中选择要下载的架构名称旁边的选项。
3. 选择 Actions。
4. 选择下载架构

发布架构

使用以下过程在 Cloud Directory 中发布架构。

发布架构

1. 在AWS Directory Service 控制台导航窗格中，在Cloud Directory中，选择Schemas。
2. 在表中选择要发布的架构名称旁边的选项。
3. 选择 Actions。
4. 选择发布
5. 在发布架构对话框中，提供以下信息：

    a. 架构名称

b. 主要版本
c. 次要版本

6. 选择 Publish。

更新架构

使用以下过程更新 Cloud Directory 中的架构。

更新架构

1. 在AWS Directory Service 控制台导航窗格中，在Cloud Directory中，选择Schemas。
2. 在表中选择要更新的方案名称旁边的选项。
3. 选择 Actions。
4. 选择 Update (更新)。
5. 在更新架构对话框中，可以选择修改架构名称，或选择Select file以应用或删除小面和属性。
6. 选择 Update (更新)。

升级架构

升级架构会将您选择的小面和属性添加到您选择的已发布架构中。使用以下过程升级已发布的架构。

升级架构

1. 在AWS Directory Service 控制台导航窗格中，在Cloud Directory中，选择Schemas。
2. 在表中选择要升级的架构名称旁边的选项。
3. 选择 Actions。
4. 选择升级
5. 在升级已发布的架构对话框中，选择以下任一选项，然后选择升级：
   - 从您当前的开发架构列表中进行选择
   - 上传新架构文件 (JSON)
6. 选择升级。

Amazon Cloud Directory 中的安全性

AWS 的云安全性的优先级最高。作为 AWS 客户，您将从专为满足大多数安全敏感型组织的要求而打造的数据中心和网络架构中受益。

安全性是 AWS 和您的共同责任。责任共担模型将其描述为云的 安全性和云中 的安全性：

- 云的安全性 – AWS 负责保护在 AWS 云中运行 AWS 服务的基础设施。AWS 还向您提供可安全使用的服务。作为 AWS 合规性计划的一部分，第三方审核人员将定期测试和验证安全性的有效性。要了解适用于 Amazon Cloud Directory 的合规性计划，请参阅合规性计划范围内的 AWS 服务。
- 云中的安全性 – 您的责任是由使用的 AWS 服务决定的。您还需要对其他因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

此文档将帮助您了解如何在使用 Cloud Directory 时应用责任共担模式。以下主题说明如何配置 Cloud Directory 以满足您的安全性和合规性目标。您还会了解如何使用其他 AWS 服务以帮助您监控和保护您的 Cloud Directory 资源。

主题

Amazon Cloud Directory 中的 Identity and Access Management

访问 Amazon Cloud Directory 时需要有 AWS 可以用来验证您的请求的凭证。这些凭证必须有权访问 AWS 资源。以下部分提供了有关如何使用的详细信息。AWS Identity and Access Management (IAM)和 Cloud Directory，通过控制可以访问您的资源的对象来保护这些资源。

Authentication

您可以下面任一类型的身份访问 AWS：

- AWS 账户根用户 – 首次创建 AWS 账户时，您将从一个对账户中的所有 AWS 服务和资源具有完全访问权限的单点登录身份开始。此身份称为 AWS 账户根用户，可以通过使用您用于创建账户的电子邮件地址和密码进行登录来访问该身份。强烈建议您不使用根用户执行日常任务，即使是管理任务。相反，请遵循仅使用根用户创建您的第一个 IAM 用户的最佳实践。然后请妥善保存根用户凭证，仅用它们执行少数账户和服务管理任务。
- IAM 用户— 一个IAM 用户是您的 AWS 账户中的一种身份，它具有特定的自定义权限（例如，在 Cloud Directory 中创建目录的权限）。您可以使用 IAM 用户名和密码登录安全 AWS 网页，例如 AWS 管理控制台、AWS 开发论坛或 AWS 支持中心。

  除了用户名和密码之外，您还可以为每个用户生成访问密钥。在通过多个开发工具包之一或使用 AWS 命令行界面 (CLI) 以编程方式访问 AWS 服务时，可以使用这些密钥。SDK 和 CLI 工具使用访问密钥对您的请求进行加密签名。如果您不使用 AWS 工具，则必须自行对请求进行签名。Cloud Directory 支持签名版本 4，这是用于对入站 API 请求进行身份验证的协议。有关对请求进行身份验证的更多信息，请参阅签名版本 4 签名流程中的AWS 一般参考。

- IAM 角色 – IAM 角色是可在账户中创建的一种具有特定权限的 IAM 身份。IAM 角色类似于 IAM 用户，因为它是一个 AWS 身份，具有确定其在 AWS 中可执行和不可执行的操作的权限策略。但是，角色旨在让需要它的任何人代入，而不是唯一地与某个人员关联。此外，角色没有关联的标准长期凭证（如密码或访问密钥）。相反，当您代入角色时，它会为您提供角色会话的临时安全凭证。具有临时凭证的 IAM 角色在以下情况下很有用：

  - 联合身份用户访问 – 您可以使用来自 AWS Directory Service、您的企业用户目录或 Web 身份提供商的现有用户身份，而不创建 IAM 用户。他们被称为联合身份用户。在通过身份提供商请求访问权限时，AWS 将为联合身份用户分配角色。有关联合身份用户的更多信息，请参阅联合身份用户和角色中的IAM 用户指南。

  - AWS 服务访问 – 服务角色是一个 IAM 角色，服务担任该角色以代表您在您的账户中执行操作。服务角色只在您的账户内提供访问权限，不能用于为访问其他账户中的服务授权。IAM 管理员可以

在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅 。创建向 AWS 服务委托权限的角色中的IAM 用户指南。

- 在 Amazon EC2 上运行的应用程序 – 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 AWS CLI 或 AWS API 请求的应用程序的临时凭证。这优先于在 EC2 实例中存储访问密钥。要将 AWS 角色分配给 EC2 实例并使其对该实例的所有应用程序可用，您可以创建一个附加到该实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅 。使用 IAM 角色向在 Amazon EC2 实例上运行的应用程序授予权限中的IAM 用户指南。

访问控制

您可以使用有效的凭证来对自己的请求进行身份验证，但您还必须拥有权限才能创建或访问 Cloud Directory 资源。例如，您必须拥有创建 Amazon Cloud Directory 的权限。

以下部分介绍如何管理 Cloud Directory 的权限。我们建议您先阅读概述。

- 管理您的 Cloud Directory 资源的访问权限概述
- 为 Cloud Directory 使用基于身份的策略（IAM 策略）
- Amazon Cloud Directory API 权限：API 权限：操作、资源和条件参考

管理您的 Cloud Directory 资源的访问权限概述

每个 AWS 资源都归某个 AWS 账户所有，创建和访问资源的权限由权限策略进行管理。账户管理员可以向 IAM 身份 (即：用户、组和角色) 挂载权限策略，某些服务 (如 AWS Lambda) 也支持向资源挂载权限策略。

> **Note**
>
> 账户管理员（或管理员用户）是具有管理员权限的用户。有关更多信息，请参阅IAM 用户指南中的 IAM 最佳实践。

在授予权限时，您要决定谁获得权限，获得对哪些资源的权限，以及您允许对这些资源执行的具体操作。

主题

Cloud Directory 资源和操作

在 Cloud Directory 中，主要资源是目录和架构。这些资源具有关联的唯一 Amazon 资源名称 (ARN)，如下表所示。

资源类型	ARN 格式
目录	`arn:aws:clouddirectory: region:account-id :directory/ directory-id`
Schema	`arn:aws:clouddirectory: region:account-id :schema/schema-state /schema-name`

有关架构状态和 ARN 的更多信息，请参阅ARN 示例中的Amazon Cloud Directory API 参考。

Cloud Directory 提供一组操作来处理相应资源。有关可用操作的列表，请参阅 Amazon Cloud Directory 操作或 Directory Service 操作。

了解资源所有权

资源所有者 是创建资源的 AWS 账户。也就是说，资源所有者是委托人实体（根账户、IAM 用户或 IAM 角色）的 AWS 账户。以下示例说明了它的工作原理：

- 如果使用您的 AWS 账户的根账户凭证来创建 Cloud Directory 资源 (如目录)，则您的 AWS 账户就是该资源的拥有者。
- 如果您在 AWS 账户中创建 IAM 用户并对该用户授予创建 Cloud Directory 资源的权限，则该用户也可以创建 Cloud Directory 资源。但是，该用户所属的 AWS 账户拥有这些 资源。

- 如果您在 AWS 账户中创建具有创建 Cloud Directory 资源权限的 IAM 角色，则任何能够担任该角色的人都可以创建 Cloud Directory 资源。该角色所属的 AWS 账户拥有此 Cloud Directory 资源。

管理对资源的访问

权限策略 规定谁可以访问哪些内容。下一节介绍创建权限策略时的可用选项。

> **Note**
>
> 本节讨论如何在 Cloud Directory 范围内使用 IAM。这里不提供有关 IAM 服务的详细信息。有关完整的 IAM 文档，请参阅什么是 IAM？中的IAM 用户指南。有关 IAM 策略语法和说明的信息，请参阅AWS IAM 策略参考中的IAM 用户指南。

附加到 IAM 身份的策略称作基于身份的功能策略（IAM 策略）和附加到资源的策略称作基于资源的策略。Cloud Directory 只支持基于身份的策略（IAM 策略）。

主题

- 基于身份的策略（IAM 策略）
- 基于资源的策略

基于身份的策略（IAM 策略）

您可以向 IAM 身份附加策略。例如，您可以执行以下操作：

- 将权限策略附加到账户中的用户或组— 账户管理员可以使用与特定用户关联的权限策略授予该用户创建 Cloud Directory 资源 (如新目录) 的权限。
- 将权限策略附加到角色（授予跨账户权限）— 您可以将基于身份的权限策略挂载到 IAM 角色，以授予跨账户的权限。例如，账户 A 中的管理员可以创建一个角色，以向其他 AWS 账户（如账户 B）或某项 AWS 服务授予跨账户权限，如下所述：
  1. 账户 A 管理员可以创建一个 IAM 角色，然后向该角色附加授予其访问账户 A 中资源的权限策略。
  2. 账户 A 管理员可以向将账户 B 标识为能够代入该角色的委托人的角色附加信任策略。
  3. 之后，账户 B 管理员可以委托权限，指派账户 B 中的任何用户代入该角色。这样，账户 B 中的用户就可以创建或访问账户 A 中的资源了。如果您需要授予 AWS 服务权限来代入该角色，则信任策略中的委托人也可以是 AWS 服务委托人。

  有关使用 IAM 委派权限的更多信息，请参阅访问控制中的IAM 用户指南。

以下权限策略对用户授予权限以运行以 `Create` 开头的所有操作。这些操作显示有关 Cloud Directory 资源 (如目录或架构) 的信息。请注意，中的通配符 (*)`Resource`元素表示可对该账户拥有的所有 Cloud Directory 资源执行操作。

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

有关将基于身份的策略用于 Cloud Directory 的更多信息，请参阅为 Cloud Directory 使用基于身份的策略（IAM 策略）。有关用户、组、角色和权限的更多信息，请参阅《IAM 用户指南》 中的身份 (用户、组和角色)。

基于资源的策略

其他服务 (如 Amazon S3) 还支持基于资源的权限策略。例如，您可以将策略附加到 S3 存储桶以管理对该存储桶的访问权限。Cloud Directory 不支持基于资源的策略。

指定策略元素：操作、效果、资源和委托人

对于每个 Cloud Directory 资源（请参阅Cloud Directory 资源和操作），该服务定义了一组 API 操作。有关可用 API 操作的列表，请参阅Amazon Cloud Directory 操作或者Directory Service 操作。为授予这些 API 操作的权限，Cloud Directory 定义了一组您可以在策略中指定的操作。请注意，执行某项 API 操作可能需要执行多个操作的权限。

以下是基本的策略元素：

- 资源— 在策略中，您可以使用 Amazon 资源名称 (ARN) 标识策略应用到的资源。对于 Cloud Directory 资源，您随时可以在 IAM 策略中使用通配符 (*)。有关更多信息，请参阅 Cloud Directory 资源和操作。
- 操作 – 您可以使用操作关键字标识要允许或拒绝的资源操作。例如，`clouddirectory:GetDirectory`权限允许执行 Cloud Directory 的用户权限`GetDirectory`operation.

- Effect— 您可以指定当用户请求特定操作（可以是允许或拒绝）时的效果。如果没有显式授予（允许）对资源的访问权限，则隐式拒绝访问。您也可显式拒绝对资源的访问，这样可确保用户无法访问该资源，即使有其他策略授予了访问权限的情况下也是如此。
- 委托人 – 在基于身份的策略（IAM 策略）中，附加了策略的用户是隐式委托人。对于基于资源的策略，您可以指定要接收权限的用户、账户、服务或其他实体 (仅适用于基于资源的策略)。Cloud Directory 不支持基于资源的策略。

有关 IAM 策略语法和介绍的更多信息，请参阅AWS IAM 策略参考中的IAM 用户指南。

有关显示所有 Amazon Cloud Directory API 操作及其适用资源的表，请参阅Amazon Cloud Directory API 权限：API 权限：操作、资源和条件参考。

在策略中指定条件

当您授予权限时，可使用访问策略语言来指定规定策略何时生效的条件。例如，您可能希望策略仅在特定日期后应用。有关使用策略语言指定条件的更多信息，请参阅Condition中的IAM 用户指南。

要表示条件，您可以使用预定义的条件键。没有特定于 Cloud Directory 的条件键。但有 AWS 范围内的条件密钥，您可以根据需要使用。有关 AWS 范围内的键的完整列表，请参阅可用的全局条件键中的IAM 用户指南。

为 Cloud Directory 使用基于身份的策略（IAM 策略）

本主题提供了基于身份的策略的示例，在这些策略中，账户管理员可以向 IAM 身份（即：用户、组和角色）附加权限策略。

> **Important**
>
> 我们建议您首先阅读以下介绍性主题，这些主题讲解了管理对 Cloud Directory 资源的访问权限的基本概念和选项。有关更多信息，请参阅 管理您的 Cloud Directory 资源的访问权限概述。

本主题的各个部分涵盖以下内容：

- 使用 AWS Directory Service 控制台所需的权限
- Amazon Cloud Directory 的 AWS 托管（预定义）策略

使用 AWS Directory Service 控制台所需的权限

要使用户可以使用 AWS Directory Service 控制台，该用户必须拥有上面策略中列出的权限或是 Directory Service 完全访问角色或 Directory Service 只读角色所授予的权限，如Amazon Cloud Directory 的 AWS 托管（预定义）策略。

如果创建比必需的最低权限更为严格的 IAM 策略，对于附加了该 IAM 策略的用户，控制台将无法按预期正常运行。

Amazon Cloud Directory 的 AWS 托管（预定义）策略

AWS 通过提供由 AWS 创建和管理的独立 IAM 策略来解决许多常用案例。托管策略可授予常用案例的必要权限，因此，您可以免去调查都需要哪些权限的工作。有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管策略。

以下 AWS 托管策略 (可以将它们附加到您账户中的用户) 特定于 Amazon Cloud Directory：

- 亚马逊云目录只读访问— 向用户或组授予对所有 Amazon Cloud Directory 资源的只读访问权限。有关更多信息，请参阅 AWS 管理控制台中的策略页面。
- 卓越亚马逊云目录完全访问— 向用户或组授予对 Amazon Cloud Directory 的完全访问权限。有关更多信息，请参阅 AWS 管理控制台中的策略页面。

此外，还有其他 AWS 托管策略适用于其他 IAM 角色。这些策略会分配给与 Amazon Cloud Directory 中的用户关联的角色，要使这些用户有权访问其他 AWS 资源 (如 Amazon EC2)，需要这些策略。

还可创建自定义 IAM 策略来允许用户访问必需的 操作和资源。您可以将这些自定义策略附加到需要这些权限的 IAM 用户或组。

Amazon Cloud Directory API 权限：API 权限：操作、资源和条件参考

在设置访问控制和编写您可挂载到 IAM 身份的权限策略 (基于身份的策略) 时，可以使用下表作为参考。该列表包含每个 Amazon Cloud Directory API 操作、您可授予操作执行权限的对应操作以及您可为其授予权限的 AWS 资源。您在策略的 `Action` 字段中指定操作，并在策略的 `Resource` 字段中指定资源值。

您可以在 Amazon Cloud Directory 策略中使用 AWS 范围的条件键来表达条件。有关 AWS 范围内的键的完整列表，请参阅可用的全局条件键中的IAM 用户指南。

> **Note**
>
> 要指定操作，请在 API 操作名称之前使用 `clouddirectory:` 前缀 (例如，`clouddirectory:CreateDirectory`)。

Amazon Cloud Directory 中的日志记录和监控

您应对目录进行监控，确保对所做的更改进行记录，这是最佳实践。这有助于确保能够调查任何意外的更改，并回滚不需要的更改。Amazon Cloud Directory 当前支持 AWS CloudTrail，您可以使用它来监控您的目录和任何相关活动。

有关更多信息，请参阅 。使用 CloudTrail 记录 Cloud Directory API 调用。

Amazon Cloud Directory 的合规性验证

作为多个 AWS 合规性计划的一部分，第三方审计员将评估 Amazon Cloud Directory 的安全性和合规性。其中包括 ISO、SOC、PCI、FedRAMP、HIPAA 及其他计划。

有关特定合规性计划范围内的 AWS 服务列表，请参阅合规性计划范围内的 AWS 服务。有关一般信息，请参阅 AWS 合规性计划。

您可以使用 AWS Artifact 下载第三方审计报告。有关更多信息，请参阅下载 AWS Artifact 中的报告。

您在使用 Cloud Directory 时的合规性责任由您的数据的敏感性、您公司的合规性目标以及适用的法律法规决定。AWS 提供以下资源来帮助实现合规性：

- 安全性与合规性快速入门指南— 这些部署指南讨论了架构注意事项，并提供了在 AWS 上部署基于安全性和合规性的基准环境的步骤。
- 《设计符合 HIPAA 安全性和合规性要求的架构》白皮书 – 此白皮书介绍公司如何使用 AWS 创建符合 HIPAA 标准的应用程序。
- AWS 合规性资源 – 这一系列的操作手册和指南可能适用于您所在的行业和地区。
- AWS Config – 此 AWS 服务可评估您的资源配置符合内部实践、行业指引和法规的程度。
- AWS Security Hub – 此 AWS 服务提供了 AWS 中安全状态的全面视图，可帮助您检查是否符合安全行业标准和最佳实践。

Amazon Cloud Directory 中的弹性能

AWS 全球基础设施围绕 AWS 区域和可用区构建。AWS 区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。Cloud Directory 是基于这些原则构建的，并可在多个 AWS 区域中使用，这些区域在物理上彼此隔离。在每个区域内，至少通过三个可用区进一步支持该服务，从而最大限度地减少由于任何单个可用区域不可用而导致的服务停机时间。

有关 AWS 区域和可用区的更多信息，请参阅 AWS 全球基础设施。

Amazon Cloud Directory 的基础设施安全性

作为一项托管服务，Amazon Cloud Directory 由 AWS 全球网络安全程序提供保护，如Amazon Web Services Services：安全过程概述白皮书。

您可以使用 AWS 发布的 API 调用通过网络访问 Cloud Directory。客户端必须支持传输层安全性 (TLS)。建议使用 TLS 1.2 或更高版本。客户端还必须支持具有完全向前保密 (PFS) 的密码套件，例如 Ephemeral Diffie-Hellman (DHE) 或 Elliptic Curve Ephemeral Diffie-Hellman (ECDHE)。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

如果在通过命令行界面或 API 访问 AWS 时需要经过 FIPS 140-2 验证的加密模块，请使用 FIPS 终端节点。有关可用的 FIPS 终端节点的更多信息，请参阅美国联邦信息处理标准 (FIPS) 第 140-2 版。

此外，必须使用访问密钥 ID 和与 IAM 委托人关联的秘密访问密钥来对请求进行签名。或者，您可以使用 AWS Security Token Service (AWS STS) 生成临时安全凭证来对请求进行签名。

事务支持

使用 Amazon Cloud Directory，通常需要添加新对象或者添加新对象与现有对象之间的关系，用于体现真实层次结构的更改。批处理操作提供了以下优势，可以使得类似于这样的目录任务更易于管理：

- 批处理操作可以最大程度地减少在目录中写入和读取对象所需的往返次数，从而提高整体应用程序的性能。
- 批处理写入提供了 SQL 数据库等效事务语义。所有操作均成功完成；如果有任何操作失败，则不会应用任何一个操作。
- 使用批处理引用，您可以创建对象并使用对新对象的引用执行进一步操作，例如将其添加到关系中，减少在写入操作前使用读取操作的开销。

BatchWrite

使用 BatchWrite 操作可对目录执行多个写入操作。批处理写入中的所有操作按顺序执行。其工作方式类似于 SQL 数据库事务。如果批处理写入中的一个操作失败，则整个批处理写入都不会影响目录。如果批处理写入失败，则引发批处理写入异常。异常包含失败操作的索引以及异常类型和消息。这些信息可以帮助确定失败的根本原因。

批处理写入中支持以下 API 操作：

- AddFacetToObject
- AttachObject
- AttachPolicy
- AttachToIndex
- AttachTypedLink
- CreateIndex
- CreateObject
- DeleteObject
- DetachFromIndex
- DetachObject
- DetachTypedLink
- RemoveFacetFromObject

- UpdateObjectAttributes

批处理引用名称

在中间批处理操作需要引用某个对象时，只有批处理写入支持批处理引用名称。例如，假设在给定批处理写入中，要分离 10 个不同的对象并将它们附加到目录的另一个部分。如果不使用批处理引用，则必须读取所有 10 个对象引用，并在批处理写入中的重新附加过程中提供它们作为输入。您可以使用批处理引用在附加过程中标识分离的资源。批处理引用可以是用数字符号/井号 (#) 作为前缀的任何常规字符串。

例如，在以下代码示例中，链接名称为 "this-is-a-typo" 的对象使用批处理引用名称 "ref" 从根分离。随后该对象使用链接名称 "correct-link-name" 附加到根。该对象使用设置为批处理引用的子引用进行标识。如果不使用批处理引用，则需要先获取所分离的 objectIdentifier，并在附加过程中在子引用中提供它。可以使用批处理引用名称避免这种额外的读取。

```
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

使用 BatchRead 操作可对目录执行多个读取操作。例如，在以下代码示例中，在单个批处理读取中读取具有引用“/managers”的对象的子级以及具有引用“/managers/bob”的对象的属性。

```
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);
```

BatchRead 支持以下 API 操作：

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

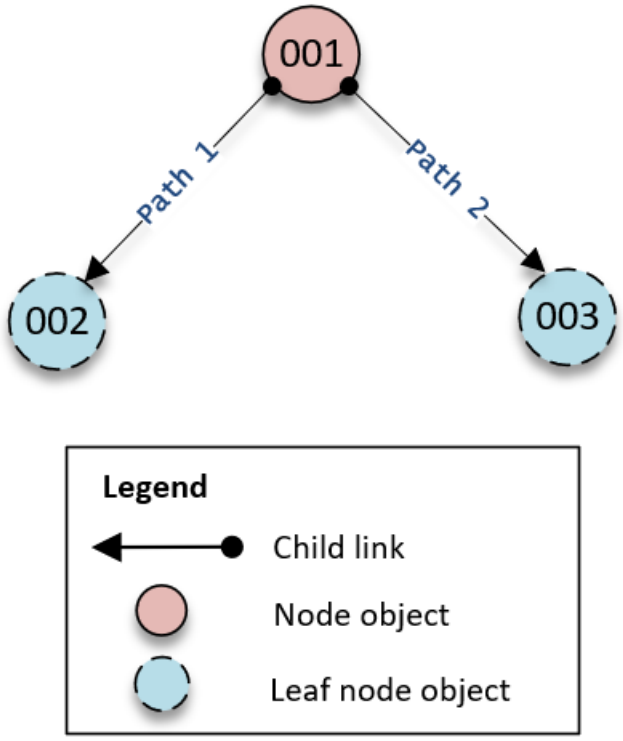
针对批处理操作的限制

对服务器的每个请求 (包括批处理请求) 所能操作的资源存在最大数量限制，不论请求中有多少个操作。这可以在编写批处理请求时带来巨大的灵活性，只要您保持在资源限制内。有关资源最大值的更多信息，请参阅Amazon Cloud Directory 限制。

限制的计算方式是将批处理内每个操作的写入数或读取数相加。例如，当前每个 API 调用的读取操作限制为 200 个对象。假设您编写一个批处理，添加 9 个 ListObjectChildren API 调用，每个调用需要读取 20 个对象。由于读取对象总数 (9 x 20 = 180) 未超过 200，批处理操作将成功。

同样的概念也适用于计算写入操作数。例如，当前写入操作限制为 20。如果您设置批处理添加 2 个 UpdateObjectAttributes API 调用，每个调用有 9 个写入操作，这也会成功。在这两种情况下，如果批处理操作超出限制，操作将会失败并引发 `LimitExceededException`。

计算批处理中包含的对象数量的正确方式是应包含实际节点或 leaf_node 对象；如果使用基于路径的方式迭代您的目录树，您也需要在批处理中包含迭代的每个路径。例如，如下图所示的基本目录树，要读取 `003` 对象的属性值，对象的读取计数总数应该是三。



沿目录树向下遍历读取的工作原理如下：

1. 读取对象 001，来确定对象 003 的路径

2. 沿 Path 2 向下

3. 读取对象 003

同样，对于属性数量，我们需要针对对象 001 和 003 中的属性数量进行计数，以确保不会超出限制。

异常处理

Cloud Directory 中的 Batch 操作有时会失败。在这些情况下，务必要知道如何处理此类故障。对于写入操作和读取操作，解决故障所用的方法不相同。

批处理写入操作失败

如果批处理写入操作失败，Cloud Directory 会使整个批处理操作失败，并返回异常。异常包含失败操作的索引以及异常类型和消息。如果看到 RetryableConflictException，您可以使用指数退避重试。执行此操作的一个简单方法是，每次出现异常或失败时，您等待的时间都加倍。例如，如果第一个批处理写入操作失败，则等待 100 毫秒，然后重试请求。如果第二个请求失败，则等待 200 毫秒，然后重试。如果第三个请求失败，则等待 400 毫秒，然后重试。

批处理读取操作失败

如果批处理读取操作失败，响应包含成功响应或异常响应。单个批处理读取操作失败不会导致整个批处理读取操作失败，Cloud Directory 会为每个操作返回单独的成功或失败响应。

相关 Cloud Directory 博客文章

- 使用 Batch 操作在 Amazon Cloud Directory 中写入和读取多个对象
- 在 Amazon Cloud Directory 中，如何使用 Batch 引用在一个 Batch 请求中引用新对象

Amazon Cloud Directory 合规性

Amazon Cloud Directory 通过了以下标准的审核，可用于需要获取合规性认证的解决方案中。

Amazon Cloud Directory 还满足联邦风险与授权管理项目 (FedRAMP) 安全要求，并且获得了 FedRAMP 联合授权委员会 (JAB) 的 FedRAMP 适度基准临时授权操作 (P-ATO)。有关 FedRAMP 的更多信息，请参阅 FedRAMP 合规性。

Amazon Cloud Directory 已通过支付卡行业 (PCI) 数据安全标准 (DSS) 版本 3.2 一级服务提供商的合规性认证。使用 AWS 产品和服务来存储、处理或传输持卡人数据的客户，在管理自己的 PCI DSS 合规性认证时，可以使用 Cloud Directory。有关 PCI DSS 的更多信息，包括如何请求 AWS PCI Compliance Package 的副本，请参阅 PCI DSS 第 1 级。

AWS 已扩展了其 Health 保险可携性与责任法 (HIPAA) 合规性计划，将 Amazon Cloud Directory 包括作为符合 HIPAA 要求的服务。如果您与 AWS 签订了商业伙伴协议 (BAA)，则可使用 Cloud Directory 帮助构建符合 HIPAA 要求的应用程序。AWS 提供以 HIPA 为中心的白皮书，供想要了解如何利用 AWS 处理和存储医疗信息详情的客户查阅。有关更多信息，请参阅 HIPAA 合规性。

	Amazon Cloud Directory 已成功完成 ISO/IEC 27001、ISO/IEC 27017、ISO/IEC 27018 和 ISO 9001 的合规性认证。有关更多信息，请参阅 ISO 27001、ISO 27017、ISO 27018 和 ISO 9001。
	系统和组织控制 (SOC) 报告是独立的第三方检查报告，用于说明 Amazon Cloud Directory 如何实现关键合规性控制和目标。这些报告的目的是帮助您和您的审计员理解旨在支持运营和合规性的 AWS 控制措施。有关更多信息，请参阅 SOC 合规性。

责任共担

安全 (包括 HIPAA 和 PCI 合规性) 是共同承担的责任。务必明确的是，Cloud Directory 合规性状态不会自动应用到您在 AWS Cloud 中运行的应用程序。您必须确保以符合标准的方式使用 AWS 服务。

使用 Cloud Directory API

Amazon Cloud Directory 包含一组 API 操作，通过这些操作可以对 Cloud Directory 功能进行编程访问。您可以将Amazon Cloud Directory API 参考指南了解如何对 Cloud Directory API 进行请求以便创建和管理各种元素。另外还介绍请求的组成部分、响应的内容以及如何对请求进行身份验证。

Cloud Directory 提供了所有必需的 API 操作供开发人员构建新应用程序。它提供以下类别的 API 调用：

- 架构的创建、读取、更新、删除 (CRUD)
- 分面的 CRUD
- 目录的 CRUD
- 对象 (节点、策略等) 的 CRUD
- 索引定义的 CRUD
- 批处理读取、批处理写入

账单如何与 Cloud Directory API 配合工作

API 调用的账单因发出的 API 调用的具体类型而异。最终一致性读取 API 调用、强一致性读取 API 调用和写入 API 调用都有特定的账单费率。元数据 API 调用是免费的。

强一致性读取操作用于读取值时的先写后读一致性。最终一致性操作用于在运行更新时检索值。使用最终一致性操作时，检索的结果可能不是最准确的，因为您从中读取值的特定主机仍在处理更新。不过，当您检索高性能调用时，这些读取操作的延迟是非常低的。

从 Cloud Directory 读取数据时，您必须指定最终一致性读取类型操作或强一致性读取类型操作。读取类型基于一致性级别。两种一致性级别是，对于最终一致性读取为 EVENTUAL，对于强一致性读取为 SERIALIZABLE。有关更多信息，请参阅 一致性级别。

下表列出了所有云目录 API 以及它们如何影响您的 AWS 账户的账单。

API	最终一致性读取 [1]	强一致性读取 [2]	写入 [3]	元数据 [4]
AddFacetToObject			X	
ApplySchema				X

API	最终一致性读取 [1]	强一致性读取 [2]	写入 [3]	元数据 [4]
AttachObject			X	
AttachPolicy			X	
AttachToIndex			X	
AttachTypedLink			X	
BatchRead	X	X		
BatchWrite			X	
CreateDirectory			X	
CreateFacet				X
CreateIndex			X	
CreateObject			X	
CreateSchema				X
CreateTypedLinkFacet				X
DeleteDirectory				X
DeleteFacet				X
DeleteObject			X	
DeleteSchema				X
DetachFromIndex			X	
DetachObject			X	
DetachPolicy			X	

API	最终一致性读取 [1]	强一致性读取 [2]	写入 [3]	元数据 [4]
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
GetTypedLinkFacetInformation				X
ListAppliedSchemaArns				X
ListAttachedIndices	X	X		

API	最终一致性读取 [1]	强一致性读取 [2]	写入 [3]	元数据 [4]
ListDevelopmentSchemaArns				X
ListDirectories				X
ListFacetAttributes				X
ListFacetNames				X
ListIncomingTypedLinks	X	X		
ListIndex	X	X		
ListManagedSchemaArns				X
ListObjectAttributes	X	X		
ListObjectChildren	X	X		
ListObjectParentPaths	X			
ListObjectParents	X	X		
ListObjectPolicies	X	X		
ListOutgoingTypedLinks	X	X		

API	最终一致性读取 [1]	强一致性读取 [2]	写入 [3]	元数据 [4]
ListPolicyAttachments	X	X		
ListPublishedSchemaArns				X
ListTagsForResource				X
ListTypedLinkFacetAttributes				X
ListTypedLinkFacetNames				X
LookupPolicy	X			
PublishSchema				X
PutSchemaFromJson				X
RemoveFacetFromObject			X	
TagResource				X
UntagResource				X
UpdateFacet				X
UpdateLinkAttributes			X	
UpdateObjectAttributes			X	

API	最终一致性读取 [1]	强一致性读取 [2]	写入 [3]	元数据 [4]
UpdateSchema				X
UpdateTypedLinkFacet				X
UpgradeAppliedSchema				X
UpgradePublishedSchema				X

[1] 最终一致性读取 API 是在 EVENTUAL 一致性级别调用的

[2] 强一致性读取 API 是在 SERIALIZABLE 一致性级别调用的

[3] 写入 API 是按写入 API 调用次数收费的

[4] 元数据 API 不收费，但分类为元数据 API 调用

有关账单的附加信息，请参阅Amazon Cloud Directory 定价。

Amazon Cloud Directory 限制

下面是 Cloud Directory 的默认限制。除非另有说明，否则每个区域具有各自的限制。

Amazon Cloud Directory

架构和目录限制

限制/概念	Quantity
每个分面的属性数 (包括必需属性)	1000
每个对象的分面数	5
对象所附加的唯一索引数	3
每个架构的分面数	30
每个属性的规则数	5
每个分面的具有默认值的属性数	10
每个分面的必需属性数	30
开发架构数	20
发布的架构数	20
应用的架构数	5
目录数	100
最大页面元素	30
最大输入大小 (合并所有输入)	200 KB
最大响应大小 (合并所有输出)	1MB
架构 JSON 文件大小限制	200 KB
分面名称长度	64 个 UTF-8 编码字节

限制/概念	Quantity
目录名称长度	64 个 UTF-8 编码字节
架构名称长度	64 个 UTF-8 编码字节

对象限制

限制/概念	Quantity
写入的对象数	每个 API 调用 20 个
读取的对象数	每个 API 调用 200 个
写入的属性值数	每个 API 调用 1000 个
读取的属性值数	每个 API 调用 1000 个
路径深度	15
最大输入大小 (合并所有输入)	200 KB
最大响应大小 (合并所有输出)	1MB
策略大小限制	10 KB
对象删除期间可删除的属性的数量	30
类型化链接身份属性的聚合值长度	64 个 UTF-8 编码字节
边缘或链路名称长度	64 个 UTF-8 编码字节
索引属性的值长度	512 个 UTF-8 编码字节
非索引属性的值长度	2 KB
附加到对象的策略数	4

针对批处理操作的限制

在批处理中，您可以调用的操作数不受限制。有关更多信息，请参阅 针对批处理操作的限制。

无法修改的限制

无法更改或提高的 Amazon Cloud Directory 限制包括：

- 分面名称长度
- 目录名称长度
- 架构名称长度
- 最大页面元素
- 边缘或链路名称长度
- 索引属性的值长度

Cloud Directory Service

下表列出了在您使用此服务时可为您提供帮助的相关资源。

Cloud Directory 入门	Link
Cloud Directory 网络研讨会	https://www.youtube.com/watch?v=UANm3DC_IxE
Cloud Directory 示例 Java 代码	https://github.com/aws-samples/AmazonCloudDirectory-sample

Cloud Directory 博客文章	描述
如何使用托管架构在 Amazon Cloud Directory 上快速开发应用程序	此博客文章介绍了如何使用托管架构在 Cloud Directory 上进行快速的原型设计和开发。其中还提供了示例 Java 代码。
如何在 Amazon Cloud Directory 中更有效地搜索	此博客文章介绍了如何使用基于分面的索引更有效地搜索。其中还提供了示例 Java 代码。
如何通过就地架构升级轻松应用 Amazon Cloud Directory 架构更改	此博客文章介绍了为任何运行 Cloud Directory 执行就地架构升级。其中还提供了示例 Java 代码。
使用 Batch 操作在 Amazon Cloud Directory 中写入和读取多个对象	介绍如何使用批量读取和写入。其中还提供了示例 Java 代码。
Amazon Cloud Directory 中的 Batch 引用在一个 Batch 请求中引用新对象	介绍如何使用批量引用。其中还提供了示例 Java 代码。
Cloud Directory 更新-对类型化链接的 Support	介绍有关使用类型化链接在 Cloud Directory 中跨层次结构创建和搜索关系。其中还提供了示例 Java 代码。
新的 Cloud Directory API 更便于沿多个维度查询数据	介绍如何使用 `ListObjectParentPaths` API 通过单一调用跨多个维度查询数据。

Cloud Directory 博客文章	描述
如何使用 Amazon Cloud Directory 创建具有不同层次结构的组织结构图	介绍如何使用示例 Java 代码创建架构和目录。
Amazon Cloud Directory – 用于分层数据的云原生目录	介绍 Cloud Directory 作为 AWS 的新服务推出。

Cloud Directory 文档	Link
Cloud Directory Service	https://docs.aws.amazon.com/clouddirectory/latest/developerguide/what_is_cloud_directory.html
Cloud Directory API 参考	https://docs.aws.amazon.com/clouddirectory/latest/APIReference/welcome.html
Cloud Directory 限制	https://docs.aws.amazon.com/clouddirectory/latest/developerguide/limits.html

Cloud Directory	Link
Cloud Directory 产品信息	https://aws.amazon.com/cloud-directory/
Cloud Directory Service	https://aws.amazon.com/cloud-directory/pricing/

文档历史记录

下表介绍了自上次发行以来的文档更改。Amazon Cloud Directory 开发人员指南。

- 最近文档更新时间： 2018 年 6 月 21 日

更新-历史记录-更改	更新-历史记录-描述	更新-历史记录-日期
新的托管架构	添加了有关托管架构选项的内容。	2018 年 6 月 21 日
已将内容迁移到本指南	已将 AWS Directory Service 管理员指南中的所有现有 Cloud Directory 内容传输到此新的 Amazon Cloud Directory 开发人员指南以更直接地反映客户需求。	2018 年 20 月 6 日
就地架构升级	添加了有关通过就地架构升级跨 Amazon Cloud Directory 目录应用架构更改的内容。	2017 年 12 月 6 日
基于分面的索引	新增基于分面的索引部分。	2017 年 8 月 9 日
批处理	更新了 Amazon Cloud Directory 的批处理的相关信息。	2017 年 26 月 7 日
合规性	新增有关 HIPAA 和 PCI 合规性的信息。	2017 年 14 月 7 日
类型化链接	新增 Amazon Cloud Directory 的新类型化链接的内容。	2017 年 5 月 31 日
Amazon Cloud Directory Service	引入了新目录类型。	2017 年 1 月 26 日

AWS 词汇表

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

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。