aws

用户指南

# Amazon Verified Permissions

# Amazon Verified Permissions: 用户指南

# Table of Contents

# 什么是 Amazon Verified Permissions？

Amazon Verified Permissions 是一项可扩展、精细的权限管理和授权服务，适用于您构建的自定义应用程序。Verified Permissions 通过将授权外部化和集中策略管理，使开发人员能够更快地构建安全的应用程序。Verified Permissions 使用 Cedar 策略语言为应用程序用户定义精细权限。

主题



## 使用 Verified Permissions 进行授权

Verified Permissions 通过验证在自定义应用程序的给定上下文中是否允许主体对资源执行操作，来提供授权。Verified Permissions 假设该主体事先已通过其他方式进行了身份识别和身份验证，例如使用 OpenID Connect 等协议、托管提供商（如 Amazon Cognito）或其他身份验证解决方案。Verified Permissions 不受限于用户的托管位置和身份验证方式。

Verified Permissions 是一项服务，支持客户在 AWS Management Console中创建、维护和测试策略。权限是使用 Cedar 策略语言来表达的。客户端应用程序会调用授权 API 来评估存储在服务中的 Cedar 策略，并提供是否允许某个操作的访问决策。

## Cedar 策略语言

Verified Permissions 中的授权策略是使用 Cedar 策略语言编写的。Cedar 是一种开源语言，用于编写授权策略并根据这些策略做出授权决策。创建应用程序时，需要确保只有经过授权的用户才能访问该应用程序，并且只能执行每个用户有权执行的操作。使用 Cedar，您可以将业务逻辑与授权逻辑解耦。在应用程序的代码中，您在向操作发出的请求之前，先调用 Cedar 授权引擎，询问“此请求是否获得了授权？”。如果决策为“允许”，则该应用程序可以执行请求的操作；如果决策为“拒绝”，则会返回错误消息。

已验证的权限目前使用 Cedar 版本 2.4。

有关 Cedar 的更多信息，请参阅下文：

- Cedar 策略语言参考指南
- 雪松 GitHub存储库

# Verified Permissions 的优势

## 加快应用程序开发

通过将授权与业务逻辑解耦，加快应用程序开发。

## 应用程序更安全

Verified Permissions 使开发人员能够构建更安全的应用程序。

## 最终用户功能

Verified Permissions 支持您为权限管理提供更丰富的最终用户功能。

# 相关服务

- Amazon Cognito - Amazon Cognito 是 Web 和移动应用程序的身份平台。它是用户目录、身份验证服务器以及 OAuth 2.0 访问令牌和 AWS 凭证的授权服务。创建策略存储库时，您可以选择从 Amazon Cognito 用户池中创建委托人和群组。有关更多信息，请参阅《Amazon Cognito 开发人员指南》。
- Amazon API Gat eway — Amazon API Gateway 是一项用于创建、发布、维护、监控和保护任何规模的 REST、HTTP 和 WebSocket API 的 AWS 服务。创建策略存储库时，您可以选择通过 API Gateway 中的 API 构建操作和资源。有关 API Gateway 的更多信息，请参阅 API Gateway 开发者指南。
- AWS IAM Identity Center - 借助 IAM Identity Center，您可以管理员工身份（也称为员工用户）的登录安全性。IAM Identity Center 提供了一个地方，您可以在其中创建或连接员工用户，并集中管理他们对所有用户 AWS 账户 和应用程序的访问权限。有关更多信息，请参阅 AWS IAM Identity Center《用户指南》。

# 访问 Verified Permissions

您可以通过以下任何方式使用 Amazon Verified Permissions。

### AWS Management Console

该控制台是一个基于浏览器的界面，用于管理 Verified Permissions 和 AWS 资源。有关通过控制台访问 Verified Permissions 的更多信息，请参阅《AWS 登录 用户指南》中的如何登录 AWS。

- Amazon 已验证权限控制台

### AWS 命令行工具

您可以使用 AWS 命令行工具在系统的命令行中发出命令以执行已验证的权限和 AWS 任务。与控制台相比，使用命令行更快、更方便。如果要构建执行 AWS 任务的脚本，命令行工具也会十分有用。

AWS 提供了两组命令行工具：AWS Command Line Interface(AWS CLI) 和AWS Tools for Windows PowerShell。有关安装和使用的信息 AWS CLI，请参阅《AWS Command Line Interface 用户指南》。有关安装和使用适用于 Windows 的工具的信息 PowerShell，请参阅《AWS Tools for Windows PowerShell 用户指南》。

- 已验证命令参考中的 AWS CLI 权限
- 在 Amazon 中验证了权限 AWS Tools for Windows PowerShell

### AWS 软件开发工具包

AWS 提供 SDK（软件开发套件），其中包括适用于各种编程语言和平台（Java、Python、Ruby、.NET、iOS、Android 等）的库和示例代码。软件开发工具包为创建对 Verified Permissions 和 AWS的编程访问提供了便捷的方式。例如，软件开发工具包执行以下类似任务：加密签署请求、管理错误以及自动重试请求。

要了解更多信息并下载 AWS SDK，请参阅工具。 Amazon Web Services

以下是各种 AWS SDK 中已验证权限资源的文档链接。

- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP
- AWS SDK for Python (Boto)
- AWS SDK for Ruby

AWS CDK 构造

AWS Cloud Development Kit (AWS CDK) 是一个开源软件开发框架，用于在代码中定义云基础架构并通过它进行配置 AWS CloudFormation。构造或可重复使用的云组件可用于创建 AWS CloudFormation 模板。然后，这些模板可用于部署您的云基础架构。

要了解更多信息并下载 AWS CDK，请参阅 C AWS loud Development Kit。

以下是已验证权限 AWS CDK 资源（例如构造）的文档链接。

- Amazon 已验证权限 L2 CDK 构造

Verified Permissions API

您可以使用已验证的权限 API AWS 以编程方式访问已验证的权限，该API允许您直接向服务发出 HTTPS 请求。使用该 API 时，必须添加代码，才能使用您的凭证对请求进行数字化签名。

- Amazon 已验证权限 API 参考指南

# Verified Permissions 定价

Verified Permissions 根据您的应用程序每月向 Verified Permissions 发出的授权请求数量提供分层定价。策略管理操作的定价也取决于您的应用程序每月向 Verified Permissions 发出的 cURL（客户端 URL）策略 API 请求的数量。

有关 Verified Permissions 费用和价格的完整列表，请参阅 Amazon Verified Permissions 定价。

若要查看您的账单，请转到 AWS Billing and Cost Management 控制台中的账单和成本管理控制面板。您的账单中包含了提供您的账单详情的使用情况报告的链接。要了解有关 AWS 账户 计费的更多信息，请参阅AWS Billing 用户指南。

如果您对 AWS 账单、账户和活动有疑问，请联系 AWS Support。

# Amazon Verified Permissions 术语和概念

要使用 Amazon Verified Permissions，您应该了解以下概念。

Verified Permissions 概念

- 授权模型
- 授权请求
- 授权响应
- 考虑的策略
- 上下文数据
- 决定性策略
- 实体数据
- 权限、授权和主体
- 策略执行
- 策略存储
- 满足条件的策略
- Verified Permissions 和 Cedar 之间的区别

Cedar 策略语言概念

- 授权
- 实体
- 组和层次结构
- 命名空间
- Policy
- 策略模板
- 架构

# 授权模型

授权模型描述了应用程序发出的授权请求的范围，以及评估这些请求的依据。它是根据不同类型的资源、对这些资源执行的操作以及执行这些操作的主体类型来定义的，同时还考虑了执行这些操作的上下文。

基于角色的访问权限控制（RBAC）是一种评估基础，其会定义各种角色并将这些角色与一组权限相关联。之后，这些角色就可以分配给一个或多个身份。分配到该角色的身份会获得与该角色关联的权限。如果修改了与该角色关联的权限，则此项修改会自动影响分配到该角色的所有身份。Cedar 通过使用主体组来支持 RBAC 决策。

基于属性的访问权限控制（ABAC）是一种评估基础，其中，与身份关联的权限由该身份的属性决定。Cedar 通过使用引用主体属性的策略条件来支持 ABAC 决策。

Cedar 策略语言允许为具有基于属性的条件的一组用户定义权限，从而将 RBAC 和 ABAC 组合到一个策略中。

# 授权请求

授权请求是应用程序对 Verified Permissions 发出的请求，用于评估一组策略，以确定主体是否可以在给定上下文中对资源执行操作。

# 授权响应

授权响应是对授权请求的响应，包括允许或拒绝决策，以及决定性策略的 ID 等其他信息。

# 考虑的策略

考虑的策略是指在评估授权请求时，由 Verified Permissions 选择包含的完整策略集。

# 上下文数据

上下文数据是提供要评估的额外信息的属性值。

# 决定性策略

决定性策略是决定授权响应的策略。例如，如果有两个满足条件的策略，其中一个是拒绝，另一个是允许，那么，拒绝策略将成为决定性策略。如果存在多个满足条件的允许策略且没有满足条件的禁止策

略，那么就会存在多个决定性策略。如果没有匹配的策略，并且响应是拒绝，那么就不存在决定性策略。

## 实体数据

实体数据是有关主体、操作和资源的数据。与策略评估相关的实体数据包括实体层次结构中的组成员关系，以及主体和资源的属性值。

## 权限、授权和主体

Verified Permissions 管理您构建的自定义应用程序中的精细权限和授权。

主体是指将身份绑定到用户名或机器 ID 等标识符的应用程序的用户（人类或机器）。身份验证过程会确定主体是否确实是其所声称的身份。

与该身份关联的是一组应用程序权限，这些权限决定了允许该主体在该应用程序中执行哪些操作。授权是评估这些权限以确定是否允许主体在应用程序中执行特定操作的过程。这些权限可以表示为策略。

## 策略执行

策略执行是在 Verified Permissions 之外的应用程序中执行评估决策的过程。如果 Verified Permissions 评估返回的结果为拒绝，则执行将确保禁止主体访问资源。

## 策略存储

策略存储是策略和模板的容器。每个存储都包含一个架构，用于验证添加到存储中的策略。默认情况下，每个应用程序都有自己的策略存储，但多个应用程序可以共享一个策略存储。当应用程序发出授权请求时，它会识别用于评估该请求的策略存储。策略存储提供了一种隔离策略集的方式，因此可以在多租户应用程序中使用，以包含每个租户的架构和策略。单个应用程序可以为每个租户提供单独的策略存储。

在评估授权请求时，Verified Permissions 仅会考虑策略存储中与该请求相关的策略子集。相关性是根据策略的范围确定的。范围确定了策略适用的特定主体和资源，以及主体可以对资源执行的操作。定义范围有助于缩小考虑的策略集的范围，从而提高性能。

## 满足条件的策略

满足条件的策略是指与授权请求的参数相匹配的策略。

# Verified Permissions 和 Cedar 之间的区别

Amazon Verified Permissions 使用 Cedar 策略语言引擎来执行其授权任务。但是，原生 Cedar 实现与 Verified Permissions 中的 Cedar 实现之间存在一些差异。本主题揭示了二者之间存在的差异。

## 命名空间定义

Verified Permissions 的 Cedar 实现与原生 Cedar 实现存在以下区别：

- Verified Permissions 仅支持在策略存储中定义的架构中的一个命名空间。
- Verified Permissions 不允许您创建具有以下值的命名空间：`aws`、`amazon` 或 `cedar`。

## 策略模板支持

Verified Permissions 和 Cedar 都只允许 `principal` 和 `resource` 范围内的占位符。但是，Verified Permissions 还要求 `principal` 和 `resource` 均不能不受限制。

以下策略在 Cedar 中有效，但由于 `principal` 不受限制，会被 Verified Permissions 拒绝。

```
permit(principal, action == Action::"view", resource == ?resource);
```

以下两个示例在 Cedar 和 Verified Permissions 中均有效，因为 `principal` 和 `resource` 都存在限制条件。

```
permit(principal == User::"alice", action == Action::"view", resource == ?resource);
```

```
permit(principal == ?principal, action == Action::"a", resource in ?resource);
```

## 架构支持

Verified Permissions 要求所有架构 JSON 键名称均为非空字符串。在某些情况下，Cedar 允许使用空字符串，例如用于属性。

## 扩展类型支持

Verified Permissions 支持策略中的 Cedar 扩展类型，但目前不支持将其包含在架构定义中或作为 `IsAuthorized` 和 `IsAuthorizedWithToken` 操作的 `entities` 参数的一部分。

扩展类型包括定点（`decimal`）和 IP 地址（`ipaddr`）数据类型。

## 实体的 Cedar JSON 格式

目前，Verified Permissions 要求您使用为 EntitiesDefinition（EntityItem 元素的数组）定义的结构传递要在授权请求考虑的实体列表。Verified Permissions 目前不支持以 Cedar JSON 格式传递要在授权请求中考虑的实体列表。有关格式化实体以用于 Verified Permissions 的具体要求，请参阅 Amazon Verified Permissions 中的实体格式设置。

## 操作组定义

Cedar 授权方法要求提供实体列表，在对策略进行授权请求评估时需要考虑这些实体。

您可以在架构中定义应用程序使用的操作和操作组。但是，Cedar 不将架构作为评估请求的一部分，仅使用架构来验证您提交的策略和策略模板。由于 Cedar 在评估请求期间不会引用架构，因此，即使在架构中定义了操作组，您也需要将所有操作组的列表包含在必须传递给授权 API 操作的实体列表中。

Verified Permissions 可以为您执行此操作。您在架构中定义的所有操作组都会自动附加到您传递至的实体列表中，作为 `IsAuthorized` 或 `IsAuthorizedWithToken` 操作的参数。

## 长度和大小限制

Verified Permissions 支持以策略存储的形式存储您的架构、策略和策略模板。这种存储方式会导致 Verified Permissions 施加一些与 Cedar 无关的长度和大小限制。

| 对象 | Verified Permissions 限制（以字节为单位） | Cedar 极限 |
| --- | --- | --- |
| 策略大小[1] | 10000 | 无 |
| 内联策略描述 | 150 | 不适用于 Cedar |
| 策略模板大小 | 10000 | 无 |
| 架构大小 | 10000 | 无 |
| 实体类型 | 200 | 无 |
| 策略 ID | 64 | 无 |

| 对象 | Verified Permissions 限制（以字节为单位） | Cedar 极限 |
| --- | --- | --- |
| 策略模板 ID | 64 | 无 |
| 实体 ID | 200 | 无 |
| 策略存储 ID | 64 | 不适用于 Cedar |

[1] Verified Permissions 中的每个策略存储都有策略限制，具体取决于策略存储中创建的策略的主体、操作和资源的总组合大小。与单个资源相关的所有策略的总大小不能超过 200000 字节。在计算模板链接策略的大小时，策略模板的大小仅计算一次，再加上用于实例化每个模板链接策略的每组参数的大小。

# Verified Permissions 入门

利用本教程，开始使用 Amazon Verified Permissions。

主题



## 注册获取 AWS 账户

如果您没有 AWS 账户，请完成以下步骤来创建一个。

要注册 AWS 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

   当您注册时 AWS 账户，就会创建AWS 账户根用户一个。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

AWS 注册过程完成后会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择我的账户来查看当前的账户活动并管理您的账户。

## 创建具有管理访问权限的用户

注册后，请保护您的安全 AWS 账户 AWS 账户根用户 AWS IAM Identity Center，启用并创建管理用户，这样您就不会使用 root 用户执行日常任务。

保护你的 AWS 账户根用户

1. 选择 Root 用户并输入您的 AWS 账户 电子邮件地址，以账户所有者的身份登录。AWS Management Console在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅《AWS 登录 用户指南》中的以根用户身份登录。

2. 为您的根用户启用多重身份验证 (MFA)。

   有关说明，请参阅《用户指南》中的为 AWS 账户 根用户（控制台）启用虚拟 MFA 设备。IAM

创建具有管理访问权限的用户

1. 启用 IAM Identity Center

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的启用 AWS IAM Identity Center。

2. 在 IAM Identity Center 中，为用户授予管理访问权限。

   有关使用 IAM Identity Center 目录 作为身份源的教程，请参阅《用户指南》 IAM Identity Center 目录中的使用默认设置配置AWS IAM Identity Center 用户访问权限。

以具有管理访问权限的用户身份登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录网址。

  有关使用 IAM Identity Center 用户登录的帮助，请参阅AWS 登录 用户指南中的登录 AWS 访问门户。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个权限集，该权限集遵循应用最低权限的最佳做法。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的创建权限集。

2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的添加组。

# IAM 已验证权限的策略

Verified Permissions 负责管理您的应用程序中用户的权限。为了让您的应用程序调用已验证权限 API 或允许 AWS Management Console 用户在已验证权限策略存储中管理 Cedar 策略，您必须添加必要的 IAM 权限。

基于身份的策略是可以附加到身份（例如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅 IAM 用户指南中的创建 IAM 策略。

使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源，以及允许或拒绝操作的条件（如下所列）。您无法在基于身份的策略中指定主体，因为它适用于其附加的用户或角色。要了解您可以在 JSON 策略中使用的所有元素，请参阅 IAM 用户指南中的 IAM JSON 策略元素参考。

| 操作 | 描述 |
| --- | --- |
| CreatePolicyStore | 创建新策略存储的操作。 |
| DeletePolicyStore | 删除策略存储的操作。 |
| ListPolicyStores | 列出中所有策略存储库的操作 AWS 账户。 |
| CreatePolicy | 在策略存储中创建 Cedar 策略的操作。您可以创建静态策略或链接到策略模板的策略。 |
| DeletePolicy | 从策略存储中删除策略的操作。 |
| GetPolicy | 检索有关指定策略信息的操作。 |
| ListPolicies | 列出策略存储中所有策略的操作。 |
| IsAuthorized | 根据授权请求中描述的参数获取授权响应的操作。 |

CreatePolicy 操作权限 IAM 策略示例：

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

```
        {
            "Effect": "Allow",
            "Action": [
                "verifiedpermissions:CreatePolicy"
            ],
            "Resource": "*"
        }
    ]
}
```

# 创建您的第一个 Verified Permissions 策略存储

首次登录 Verified Permissions 控制台时，您可以选择如何创建您的第一个策略存储和 Cedar 策略。按照《AWS 登录用户指南》中的如何登录 AWS 所述，根据用户类型选择相应的登录过程。在控制台主页上，选择 Amazon Verified Permissions 服务。选择开始。

## 创建示例策略存储

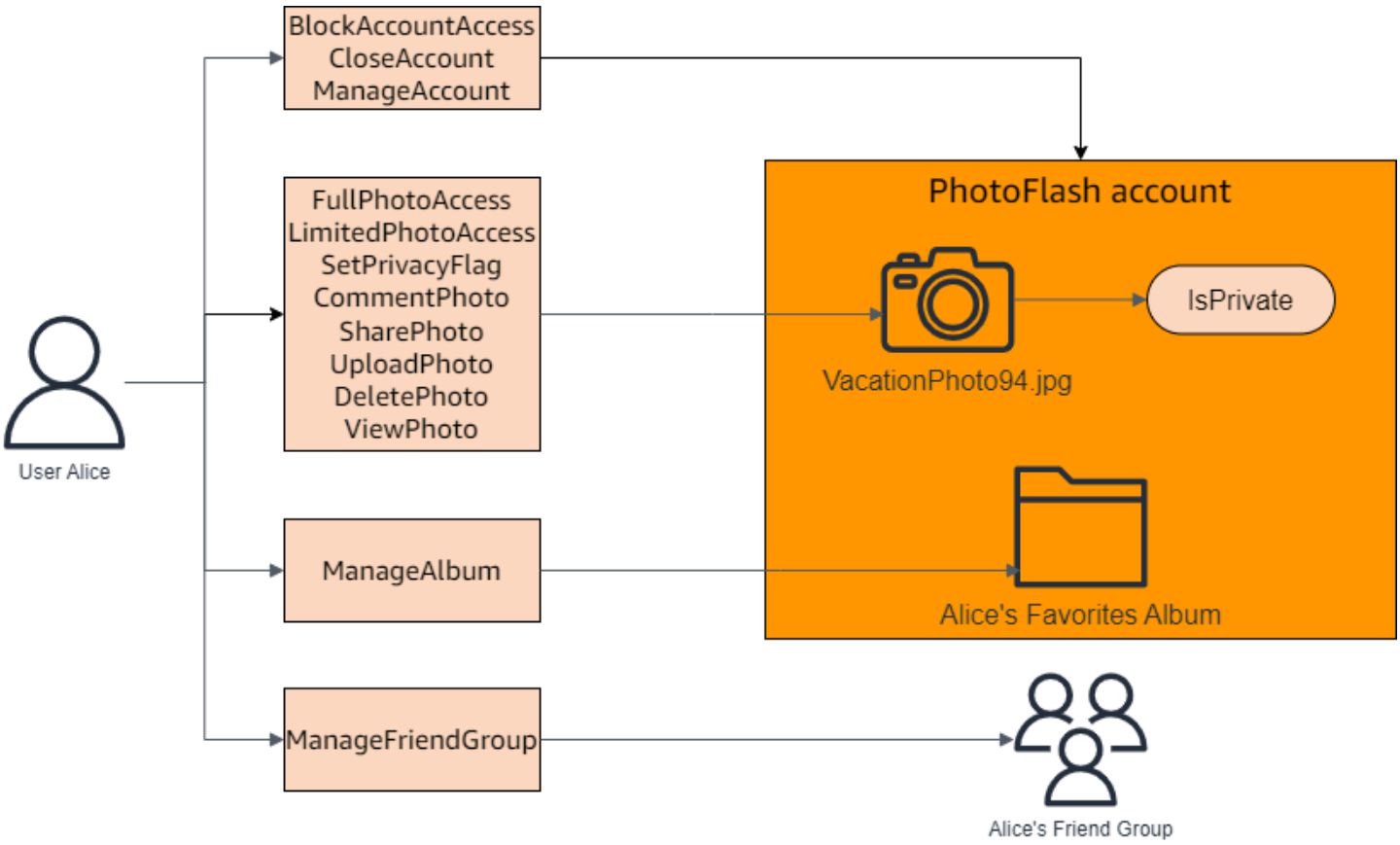
如果这是您第一次使用 Verified Permissions，我们建议您使用其中一个示例策略存储来熟悉 Verified Permissions 的工作原理。示例策略存储提供了预定义的策略和架构。

要使用示例策略存储配置方法创建策略存储，请按以下步骤操作：

1. 在已验证的权限控制台中，选择创建新的策略存储。
2. 在 “起始选项” 部分中，选择示例策略存储。
3. 在示例项目部分中，选择要使用的示例 Verified Permissions 应用程序的类型。在本教程中，选择PhotoFlash策略存储。
4. 系统会根据您选择的示例项目，自动为示例策略存储的架构生成一个命名空间。
5. 选择创建策略存储。

   您的策略存储是使用策略、策略模板和示例策略存储的架构创建的。

下图说明了 PhotoFlash 示例策略存储操作与它们适用的资源类型之间的关系。




## 为示例策略存储创建模板链接策略

PhotoFlash 示例策略存储包括策略、策略模板和架构。您可以根据示例策略存储中包含的策略模板创建模板链接策略。

要为示例策略存储创建模板链接策略，请按以下步骤操作：

1. 打开 Verified Permissions 控制台，网址为 https://console.aws.amazon.com/verifiedpermissions/。选择您的保单商店。
2. 在左侧的导航窗格中，选择策略。
3. 选择创建策略，然后选择创建模板链接策略。
4. 选择策略模板旁边的单选按钮，上面写着 “授予对非私人共享照片的完全访问权限”，然后选择 “下一步”。
5. 在 “校长” 栏中输入`PhotoFlash::User::"Alice"`。在 “资源” 中输入`PhotoFlash::Album::"Bob-Vacation-Album"`。
6. 选择创建模板链接策略。

   新的模板链接策略显示在策略下。

7. 为 PhotoFlash 示例策略存储创建另一个与模板关联的策略。选择创建策略，然后选择创建模板链接策略。
8. 选择策略模板旁边的单选按钮，上面写着 “授予对非私人共享照片的有限访问权限”，然后选择 “下一步”。
9. 在 “校长” 栏中输入`PhotoFlash::FriendGroup::"MySchoolFriends"`。在 “资源” 中输入`PhotoFlash::Album::"Alice’s favorite album"`。
10. 选择创建模板链接策略。

    新的模板链接策略显示在策略下。

我们将在本教程的下一部分中测试新的模板链接策略。有关可用于为其创建模板关联策略的值的更多示例，请参阅。 PhotoFlash PhotoFlash与模板关联的策略示例

## 测试示例策略存储

创建示例策略存储和模板链接策略后，您可以使用 Verified Permissions 测试平台运行模拟授权请求，从而测试示例 Verified Permissions 静态策略和新的模板链接策略。

根据您创建示例策略存储的时间，您的策略模板可能与本过程中的参考有所不同。在开始本教程的这一部分之前，请检查您的 PhotoFlash 示例策略存储中是否有遵循的每个策略模板。如果您的策略与这些政策不一致，请编辑现有策略或通过示例项目选项创建新的策略存储PhotoFlash。

授予对非私人共享照片的完全访问权限

```
permit (
    principal in ?principal,
    action in PhotoFlash::Action::"FullPhotoAccess",
    resource in ?resource
)
when { resource.IsPrivate == false };
```

授予对非私人共享照片的有限访问权限

```
permit (
    principal in ?principal,
    action in PhotoFlash::Action::"LimitedPhotoAccess",
    resource in ?resource
)
when { resource.IsPrivate == false };
```

要测试示例策略存储策略，请按以下步骤操作：

1. 打开 Verified Permissions 控制台，网址为 https://console.aws.amazon.com/verifiedpermissions/。选择您的保单商店。
2. 在左侧导航窗格中，选择测试平台。
3. 选择可视模式。
4. 在主体部分，从架构中的主体类型中选择PhotoFlash:: User。在文本框中为该用户输入一个标识符。例如，`Alice`。
5. 不要为主体选择添加父级。
6. 在 “账户:实体” 属性中，确保选中PhotoFlash:: 账户实体。为该账户输入一个标识符。例如，`Alice-account`。
7. 在资源部分中，选择PhotoFlash:: Photo 资源类型。在文本框中为该照片输入一个标识符。例如，`photo.jpeg`。
8. 选择 “添加父项”，然后选择PhotoFlash:: Acco unt 作为实体类型。为照片的父账户输入您在账户：实体字段中为该用户指定的同一标识符。例如，`Alice-account`。
9. 在 “操作” 部分中，从有效操作列表中选择PhotoFlash:: Action:: ViewPhoto “”。
10. 在其他实体部分中，选择添加此实体，添加建议的账户实体。
11. 选择页面顶部的运行授权请求，在示例策略存储中模拟 Cedar 策略的授权请求。测试平台应显示允许请求的决策。

下表提供了您可以使用 Verified Permissions 测试平台测试的主体、资源和操作的其他值。该表包括基于 PhotoFlash 示例策略存储库中包含的静态策略以及您在上一节中创建的模板关联策略所做的授权请求决定。

| 主体值 | 主体账户：实体值 | 资源值 | 资源父级值 | 操作 | 授权决策 |
|---|---|---|---|---|---|
| PhotoFlash:: 用户 \| Alic e | PhotoFlash:: 账户 \| Alice-Account | PhotoFlash:: Photo \| photo.jpeg | PhotoFlash:: 账户 \| BOB 账户 | PhotoFlash:: 动作::”” ViewPhoto | 拒绝 |
| PhotoFlash:: 用户 \| Alic e | PhotoFlash:: 账户 \| Alice-Account | PhotoFlash:: Photo \| photo.jpeg | PhotoFlash:: 账户 \| Alice-Account | PhotoFlash:: 动作::”” ViewPhoto | 允许 |

| 主体值 | 主体账户：实体值 | 资源值 | 资源父级值 | 操作 | 授权决策 |
|---|---|---|---|---|---|
| PhotoFlash::用户 \| Alic e | PhotoFlash::账户 \| Alice-Account | PhotoFlash::Photo \| Bob-photo.jpeg | PhotoFlash::专辑 \| Bob-Vacation-Album | PhotoFlash:: 动作::"" ViewPhoto | 允许 |
| PhotoFlash::用户 \| Alic e | PhotoFlash::账户 \| Alice-Account | PhotoFlash::Photo \| Bob-photo.jpeg | PhotoFlash::专辑 \| Bob-Vacation-Album | PhotoFlash:: 动作::"" DeletePhoto | 拒绝 |
| PhotoFlash::用户 \| Alic e | PhotoFlash::账户 \| Alice-Account | PhotoFlash::Photo \| Bob-photo.jpeg, IsPrivate: Boolean \| true | PhotoFlash::专辑 \| Bob-Vacation-Album | PhotoFlash:: 动作::"" ViewPhoto | 拒绝 |
| PhotoFlash::用户 \| Jane, PhotoFlash::FriendGroup \| MySchoolFriends | PhotoFlash::账户 \| Jane-Account | PhotoFlash:: Photo \| photo.jpeg | PhotoFlash::Album \| Alice 最喜欢的专辑 | PhotoFlash:: 动作::"" ViewPhoto | 允许 |
| PhotoFlash::用户 \| Jane, PhotoFlash::FriendGroup \| MySchoolFriends | PhotoFlash::账户 \| Jane-Account | PhotoFlash:: Photo \| photo.jpeg | PhotoFlash::Album \| Alice 最喜欢的专辑 | PhotoFlash:: 动作::"" DeletePhoto | 拒绝 |

# 使用关联的 API 和身份提供商创建策略存储

Amazon 验证权限的一个常见用例是授权从应用程序客户端向后端 API 发出的请求。 AWS 有一项用于应用程序用户身份验证的服务：Amazon Cognito。 AWS 还有一项用于安全托管 API 的服

务：Amazon API Gateway。将已验证权限策略存储与这两者结合使用时 AWS 服务，您可以将应用程序中的用户池身份验证和 API 授权与一组一致的集中式策略连接起来。经过验证的权限策略存储内置了对 Amazon Cognito 用户池身份源和 API Gateway API 的支持。

要创建链接到现有用户池和 API 的策略存储，请在创建新的策略存储时选择 “使用 Cognito 和 API Gatewa y 设置”。

与 API 关联的策略存储会自动为授权请求配置您的授权模型和资源。使用 Cognito 和 API Gateway 进行设置创建过程会生成一个策略存储，其中包含用户池身份源，以及将 API Gateway 连接到已验证权限的 Lambda 授权方。最初，您可以根据用户的群组成员资格授权 API 请求。例如，“已验证权限” 只能向`Directors`群组成员授予访问权限。

随着应用程序的发展，您可以使用用户属性和 OAuth 2.0 范围实现精细授权。例如，“已验证权限” 只能向在域中拥有`email`属性的用户授予访问权限`mycompany.co.uk`。

在您自动化 API 的授权模型后，您剩下的责任就是对用户进行身份验证并在您的应用程序中生成 API 请求，以及维护您的策略存储。

要了解更多信息，请参阅与 API 关联的策略存储。

# Amazon Verified Permissions 策略存储

策略存储是策略和策略模板的容器。每个策略存储都包含一个架构，用于验证添加到策略存储中的策略。我们建议为每个应用程序创建一个策略存储，或者针对多租户应用程序为每个租户创建一个策略存储。在发出授权请求时，必须指定一个策略存储。

我们建议将命名空间用于策略存储中的 Cedar 实体，以防止产生歧义。命名空间是类型的字符串前缀，由一对冒号（::）作为分隔符进行分隔。Verified Permissions 支持每个策略存储库使用一个命名空间。有关更多信息，请参阅《Cedar 策略语言参考指南》中的命名空间。

主题

- 创建 Verified Permissions 策略存储
- 与 API 关联的策略存储
- 切换 Verified Permissions 策略存储
- 删除 Verified Permissions 策略存储

## 创建 Verified Permissions 策略存储

您可以使用以下方法创建策略存储：

- 按照指导设置进行操作-在创建第一个策略之前，您将定义具有有效操作的资源类型和委托人类型。
- 使用 API Gateway 和身份源进行设置 — 使用身份提供商 (IdP) 登录的用户以及通过 Amazon API Gateway API 登录的用户定义您的委托人实体。如果您希望您的应用程序以用户的群组成员身份授权 API 请求，我们建议您使用此选项。
- 从示例策略存储区开始-选择预定义的示例项目策略存储库。如果您正在学习 Verified Permissions 并想要查看和测试示例策略，我们建议您使用此选项。
- 创建空策略存储库-您将自己定义架构和所有访问策略。如果您已经熟悉如何配置策略存储，我们建议您使用此选项。

### Guided setup

要使用引导式设置配置方法创建策略存储，请按以下步骤操作：

引导式设置向导将引导您完成创建策略存储第一次迭代的过程。您将为第一个资源类型创建架构，描述适用于该资源类型的操作以及您为其授予权限的主体类型。然后，您将创建第一个策略。完成

此向导后，您将能够向策略存储中添加内容，扩展架构以描述其他资源和主体类型，以及创建其他策略和模板。

1. 在已验证的权限控制台中，选择创建新的策略存储。
2. 在 “开始选项” 部分中，选择引导式设置。
3. 输入策略存储描述。此文本可以是任何适合您组织的文本，作为对当前政策存储功能（例如天气更新）的友好参考。
4. 在详细信息部分中，输入架构的命名空间。
5. 选择下一步。
6. 在资源类型窗口中，输入资源类型的名称。
7. （可选）选择添加属性，添加资源属性。输入属性名称，然后为资源的每个属性选择一个属性类型。选择每个属性是否为必填项。根据架构验证策略时，Verified Permissions 会使用指定的属性值。要删除已为该资源类型添加的属性，请选择该属性旁边的删除。
8. 在操作字段中，输入要为指定的资源类型授权的操作。要为该资源类型添加其他操作，请选择添加操作。要删除已为该资源类型添加的操作，请选择该操作旁边的删除。
9. 在主体类型的名称字段中，输入将对您的资源类型使用指定操作的主体类型的名称。
10. 选择下一步。
11. 在主体类型窗口中，为您的主体类型选择身份来源。

    - 如果主体的 ID 和属性将由您的 Verified Permissions 应用程序直接提供，请选择自定义。要添加主体属性，请选择添加属性。输入属性名称，然后为主体的每个属性选择一个属性类型。根据架构验证策略时，Verified Permissions 会使用指定的属性值。要删除已为该主体类型添加的属性，请选择该属性旁边的删除。
    - 如果主体的 ID 和属性将通过 Amazon Cognito 生成的 ID 或访问令牌提供，请选择 Cognito 用户群体。选择连接用户群体。选择 AWS 区域并输入要连接的 Amazon Cognito 用户群体的用户群体 ID。选择连接。有关更多信息，请参阅《Amazon Cognito 开发人员指南》中的使用 Amazon Verified Permissions 进行授权。
12. 选择下一步。
13. 在策略详细信息部分中，为您的第一个 Cedar 策略输入可选的策略描述。
14. 在主体范围字段中，选择将从策略中获得权限的主体。

    - 选择特定主体，将策略应用于特定主体。在允许执行操作的主体字段中选择该主体，然后为该主体输入一个实体标识符。
    - 选择所有主体，将该策略应用于策略存储中的所有主体。

15. 在资源范围字段中，选择授权指定主体对哪些资源执行操作。
    - 选择特定资源，将该策略应用于特定资源。在此策略适用的资源字段中选择该资源，然后为该资源输入一个实体标识符。
    - 选择所有资源，将该策略应用于策略存储中的所有资源。
16. 在操作范围字段中，选择授权指定主体执行的操作。
    - 选择特定操作集合，将该策略应用于特定操作。在此策略适用的操作字段中，选中操作旁边的复选框。
    - 选择所有操作，将该策略应用于策略存储中的所有操作。
17. 在策略预览部分中查看该策略。选择创建策略存储。

## Set up with API Gateway and an identity source

使用 API Gateway 设置和身份源配置方法创建策略存储

API Gateway 选项使用经过验证的权限策略来保护 API，这些策略旨在通过用户的群组或角色做出授权决定。此选项构建一个策略存储库，用于测试身份源组的授权，以及使用 Lambda 授权方的 API。

IdP 中的用户及其群组要么成为您的委托人（ID 令牌），要么成为您的上下文（访问令牌）。API Gateway API 中的方法和路径将成为您的策略授权的操作。您的应用程序将成为资源。根据此工作流程，已验证权限将创建策略存储、Lambda 函数和 API Lambda 授权方。完成此工作流程后，您必须为您的 API 分配 Lambda 授权方。

1. 在已验证的权限控制台中，选择创建新的策略存储。
2. 在 “起始选项” 部分，选择 “使用 API Gateway 和身份源进行设置”，然后选择 “下一步”。
3. 在 “导入资源和操作” 步骤的 API 下，选择一个 API，该API将用作策略存储资源和操作的模型。
    a. 从 API 中配置的阶段中选择部署阶段，然后选择导入 API。有关 API 阶段的更多信息，请参阅 Amazon API Gateway 开发者指南中的为 REST API 设置阶段。
    b. 预览导入的资源和操作地图。
    c. 要更新资源或操作，请修改您的 API 路径或方法，然后选择导入 API。
    d. 如果您对自己的选择感到满意，请选择 “下一步”。
4. 在身份来源中，选择身份提供者类型。你可以选择 Amazon Cognito 用户池或 OpenID Connect (OIDC) IdP 类型。

5. 如果你选择 Amazon Cognito：

   a. 选择 AWS 区域 与 AWS 账户 您的策略存储区相同的用户池。

   b. 选择要传递给要提交授权的 API 的令牌类型。两种令牌类型都包含用户组，这是这种与 API 关联的授权模型的基础。

   c. 在应用程序客户端验证下，您可以将策略存储的范围限制为多租户用户池中的一部分 Amazon Cognito 应用程序客户端。要要求该用户使用用户池中的一个或多个指定应用程序客户端进行身份验证，请选择 “仅接受具有预期应用程序客户端 ID 的令牌”。要接受任何通过用户池进行身份验证的用户，请选择不验证应用程序客户端 ID。

   d. 选择下一步。

6. 如果您选择了 OIDC 提供商：

   a. 在发卡机构 URL 中，输入您的 OIDC 发行人的 URL。例如，这是提供授权服务器、签名密钥以及有关您的提供商的其他信息的服务端点`https://auth.example.com`。您的发卡机构 URL 必须托管 OIDC 发现文档，网址为。`/.well-known/openid-configuration`

   b. 在令牌类型中，选择您希望您的应用程序提交以进行授权的 OIDC JWT 类型。有关更多信息，请参阅 在架构和策略中使用身份源。

   c. 在 Token 索赔中，选择您想要在保单存储中设置用户属性的方式。这些属性定义了您的保单可以参考的声明。

      i. 选择索赔来源。

         A. 要提供示例令牌，请选择 “从 JWT 有效负载中提取”，然后粘贴所选令牌类型的 JWT 的有效负载。JWT 包含标头、有效载荷和签名。您的示例 JWT 必须经过解码且仅限有效负载。要解析有效负载，请选择提取。

         B. 要输入您自己的属性集，请选择 “手动输入声明”。

      ii. 输入或确认要添加到架构中用户主体或操作上下文属性的每个令牌声明名称和声明值类型。

   d. 在用户和群组声明中，为身份来源选择用户声明。通常`sub`，这是来自您的身份证或访问令牌的声明，该令牌包含待评估实体的唯一标识符。来自已连接的 OIDC IdP 的身份将映射到您的策略存储中的用户类型。

   e. 在 “用户和群组声明” 中，为身份来源选择群组声明。通常`groups`，这是来自您的身份证或访问令牌的声明，其中包含用户的群组列表。您的策略存储将根据群组成员资格对请求进行授权。

f. 在受众验证或客户端 ID 中，输入您希望您的政策商店在授权请求中接受的客户端 ID 或受众网址（如果有）。对于访问令牌，请输入受众声明值，例如`https://myapp.example.com`。对于 ID 令牌，请输入客户端 ID，例如`1example23456789`。

g. 选择下一步。

7. 如果您选择了 Amazon Cognito，则经过验证的权限会在您的用户池中查询群组。对于 OIDC 提供商，请手动输入组名。“将操作分配给群组” 步骤可为您的策略存储创建允许群组成员执行操作的策略。

   a. 选择或添加要包含在策略中的群组。

   b. 为您选择的每个群组分配操作。

   c. 选择下一步。

8. 在 Deploy 应用程序集成中，查看已验证的权限将用于创建您的策略存储和 Lambda 授权方的步骤。

9. 准备好创建新资源时，选择创建并部署。

10. 在浏览器中保持 “策略存储状态” 步骤处于打开状态，以通过已验证的权限监控资源创建的进度。

11. 一段时间后（通常为大约一个小时），或者当 D eploy Lambda 授权方步骤显示成功时，配置您的授权方。

    经过验证的权限将在您的 API 中创建一个 Lambda 函数和一个 Lambda 授权者。选择 “打开 API” 以导航到您的 API。

    要了解如何分配 Lambda 授权方，请参阅亚马逊 API Gateway 开发者指南中的使用 API Gateway Lambda 授权方。

    a. 导航到您的 API 的授权方，并记下已验证权限创建的授权方的名称。

    b. 导航到 “资源”，然后在 API 中选择一种顶级方法。

    c. 在 “方法请求设置” 下选择 “编辑”。

    d. 将授权者设置为您之前记下的授权者名称。

    e. 展开 HTTP 请求标头，输入名称或`AUTHORIZATION`，然后选择必需。

    f. 部署 API 阶段。

    g. 保存您的更改。

12. 使用您在选择身份来源步骤中选择的令牌类型的用户池令牌来测试您的授权方。有关用户池登录和检索令牌的更多信息，请参阅 Amazon Cognito 开发者指南中的用户池身份验证流程。

13. 在 API 请求的AUTHORIZATION标题中使用用户池令牌再次测试身份验证。
14. 检查您的新保单存储。添加和完善政策。

### Sample policy store

要使用示例策略存储配置方法创建策略存储，请按以下步骤操作：

1. 在 “起始选项” 部分中，选择示例策略存储。
2. 在示例项目部分中，选择要使用的示例 Verified Permissions 应用程序的类型。

   - PhotoFlash是一个面向客户的 Web 应用程序示例，它使用户能够与朋友共享个人照片和相册。用户可以对允许谁查看、评论和重新共享照片设置精细权限。账户所有者还可以创建好友组，并将照片整理到相册中。
   - DigitalPetStore 是一个示例应用程序，任何人都可以在其中注册并成为客户。客户可以添加待售宠物、搜索宠物和下单。添加宠物的客户将被记录为宠物主人。宠物主人可以更新宠物的详细信息、上传宠物图片或删除宠物清单。已下单的客户将被记录为订单所有者。订单所有者可以获取订单的详细信息或取消订单。宠物商店经理拥有管理权限。

     > **Note**
     >
     > DigitalPetStore 示例政策存储区不包含策略模板。PhotoFlash和TinyTodo示例策略存储包括策略模板。

   - TinyTodo是一个允许用户创建任务和任务列表的示例应用程序。列表所有者可以管理和共享自己的列表，并指定谁可以查看或编辑他们的列表。
3. 系统会根据您选择的示例项目，自动为示例策略存储的架构生成一个命名空间。
4. 选择创建策略存储。

   您的策略存储是使用您选择的示例策略存储的策略和架构创建的。有关您可以为示例策略存储创建的模板链接策略的更多信息，请参阅Verified Permissions 示例策略存储的模板链接策略示例。

### Empty policy store

要使用清空策略存储配置方法创建策略存储，请按以下步骤操作：

1. 在 “起始选项” 部分中，选择清空策略存储。

2. 选择创建策略存储。

创建的空策略存储没有架构，这意味着策略未经过验证。有关更新策略存储架构的更多信息，请参阅 Amazon Verified Permissions 策略存储架构。

有关为策略存储创建策略的更多信息，请参阅创建 Amazon Verified Permissions 静态策略和创建模板链接策略。

AWS CLI

要使用 AWS CLI创建空策略存储，请按以下步骤操作：

您可以使用 `create-policy-store` 操作创建策略存储。

> **Note**
>
> 使用创建的策略存储 AWS CLI 为空。
>
> - 要添加架构，请参阅 Amazon Verified Permissions 策略存储架构。
> - 要添加策略，请参阅创建 Amazon Verified Permissions 静态策略。
> - 要添加策略模板，请参阅创建策略模板。

```
$ aws verifiedpermissions create-policy-store \
    --validation-settings "mode=STRICT"
{
    "arn": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111",
    "createdDate": "2023-05-16T17:41:29.103459+00:00",
    "lastUpdatedDate": "2023-05-16T17:41:29.103459+00:00",
    "policyStoreId": "PSEXAMPLEabcdefg111111"
}
```

AWS SDKs

您可以使用 `CreatePolicyStore` API 创建策略存储。有关更多信息，请参阅《Amazon 已验证权限 API 参考指南》中的 CreatePolicyStor e。

# 与 API 关联的策略存储

在 Amazon Verified Permissions 控制台中创建新的策略存储时，可以选择 “使用 API Gateway 和身份源进行设置” 选项。使用此选项，您可以构建与 API 关联的策略存储，这是一种用于向 Amazon Cognito 用户池或 OIDC 身份提供商 (IdP) 进行身份验证的应用程序的授权模型，并从 Amazon API Gateway API API 获取数据。要开始使用，请参阅 使用关联的 API 和身份提供商创建策略存储。

主题



> **Important**
>
> 您使用 “使用 API Gateway 设置” 和 “已验证权限” 控制台中的身份源选项创建的策略存储不适用于立即部署到生产环境。使用初始策略存储，完成授权模型并将策略存储资源导出到 CloudFormation。使用 AWS Cloud Development Kit (CDK) 以编程方式将经过验证的权限部署到生产环境中。有关更多信息，请参阅 通过以下方式进入生产阶段 AWS CloudFormation。

在关联到 API 和身份源的策略存储中，您的应用程序在向 API 发出请求时会在授权标头中显示用户池令牌。您的策略存储库的身份源为已验证的权限提供令牌验证。令牌与 IsAuthorizedWithTokenAPI 形成授权请求。`principal`Verified Permissions 围绕用户的群组成员资格制定策略，如身份 (ID) 和访问令牌（例如`cognito:groups`用户池）中的群组声明中所示。您的 API 在 Lambda 授权机构中处理来自您的应用程序的令牌，并将其提交给已验证权限以做出授权决定。当您的 API 收到来自 Lambda 授权方的授权决定时，它会将请求传递给您的数据源或拒绝该请求。

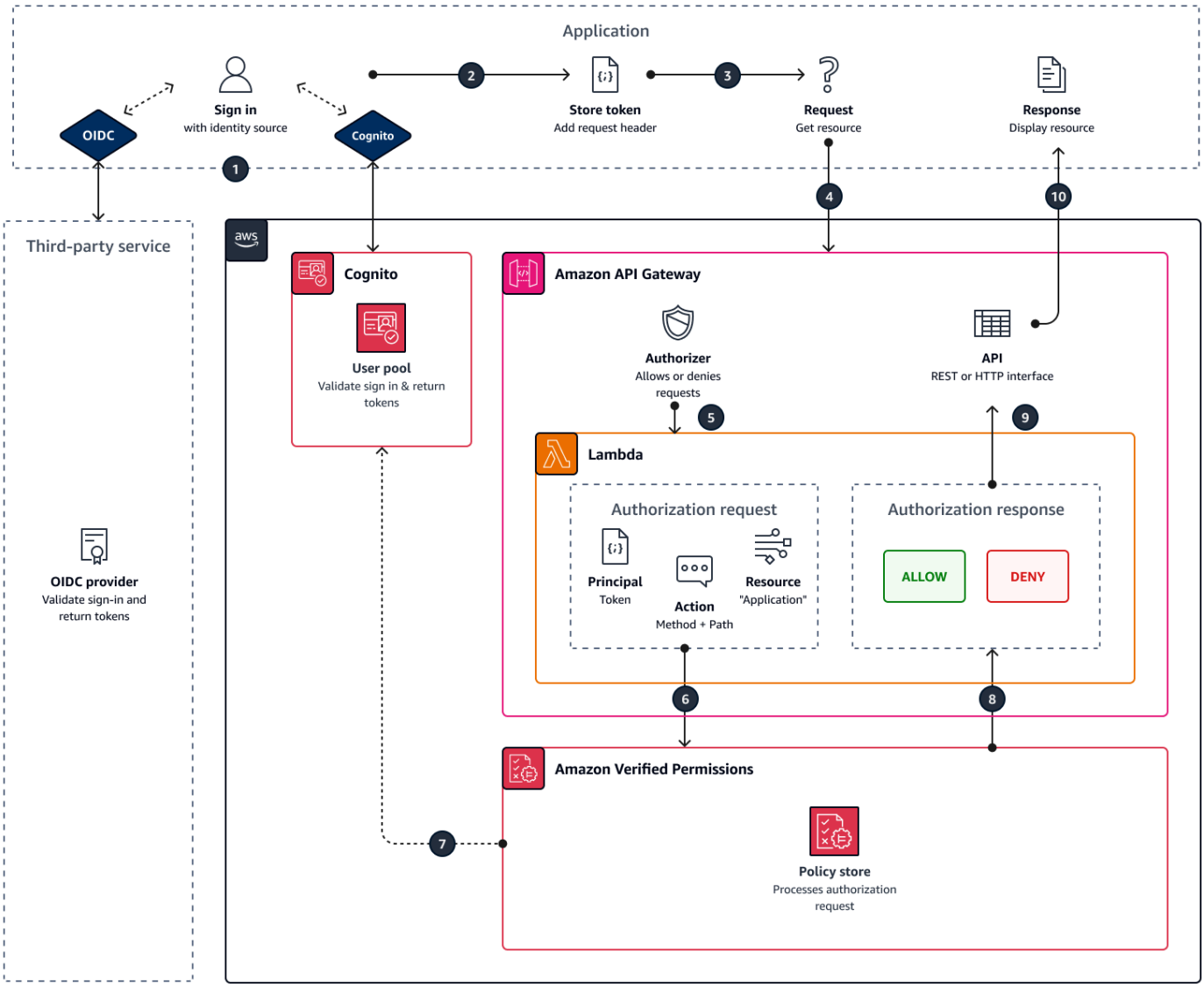
身份源和带有已验证权限的 API Gateway 授权的组成部分

- 用于对用户进行身份验证和分组的 Amazon Cognito 用户池或 OIDC IdP。用户的令牌会填充群组成员资格以及已验证权限在您的策略存储中评估的委托人或上下文。
- 一个 API Gateway REST API。例如，“已验证权限” 定义来自 API 路径和 API 方法的操作`MyAPI::Action::get /photo`。

- 一个 Lambda 函数和一个 API 的 Lam bda 授权者。Lambda 函数从您的用户池中获取不记名令牌，请求已验证权限的授权，然后将决策返回给 API Gateway。使用 Cognito 和 API Gateway 进行设置工作流程会自动为您创建此 Lambda 授权者。
- 已验证权限策略存储。策略存储标识源是您的用户池。策略存储架构反映了您的 API 的配置，策略将用户组与允许的 API 操作关联起来。
- 一种通过您的 IdP 对用户进行身份验证并将令牌附加到 API 请求的应用程序。

## 已验证权限如何授权 API 请求

当您创建新的策略存储并选择 “使用 Cognito 和 API Gateway 进行设置” 选项时，“已验证的权限” 会创建策略存储架构和策略。架构和策略反映了 API 操作以及您想要授权执行这些操作的用户池组。经过验证的权限还会创建 Lambda 函数和授权者。您必须在 API 中的方法上配置新的授权方。




1. 您的用户通过 Amazon Cognito 或其他 OIDC IdP 使用您的应用程序登录。IdP 颁发包含用户信息的 ID 和访问令牌。
2. 您的应用程序存储 JWT。有关更多信息，请参阅 Amazon Cognito 开发者指南中的在用户池中使用令牌。
3. 您的用户请求您的应用程序必须从外部 API 检索的数据。
4. 您的应用程序从 API Gateway 中的 REST API 请求数据。它会附加 ID 或访问令牌作为请求标头。
5. 如果您的 API 有用于授权决策的缓存，则它会返回之前的响应。如果缓存已禁用或 API 没有当前缓存，则 API Gateway 会将请求参数传递给基于令牌的 Lambda 授权机构。
6. Lambda 函数通过 API 向已验证权限策略存储发送授权请求。IsAuthorizedWithTokenLambda 函数传递授权决策的要素：

a. 用户的代币作为委托人。

b. 例如，API 方法与 API 路径相结合`GetPhoto`，作为操作。

c. `Application`作为资源的术语。

7. 已验证的权限会验证令牌。有关如何验证亚马逊 Cognito 令牌的更多信息，请参阅 Amazon Cognito 开发者指南中的使用亚马逊验证权限进行授权。
8. Verified Permissions 会根据您的策略存储中的策略评估授权请求并返回授权决定。
9. Lambda 授权方向 API Gateway 返回`Allow`或`Deny`响应。
10. API 会向您的应用程序返回数据或`ACCESS_DENIED`响应。您的应用程序处理并显示 API 请求的结果。

## 添加基于属性的访问控制 (ABAC)

与 IdP 的典型身份验证会话返回 ID 和访问令牌。您可以将这两种令牌类型中的任何一种作为不记名令牌传递给您的 API 的应用程序请求。根据您在创建策略存储时所做的选择，“验证权限” 需要两种类型的令牌中的一种。这两种类型都包含有关用户群组成员资格的信息。有关 Amazon Cognito 中令牌类型的更多信息，请参阅 Amazon Cognit o 开发者指南中的在用户池中使用令牌。

创建策略存储后，您可以添加和扩展策略。例如，您可以在将新群组添加到用户池时向策略中添加新群组。由于您的策略存储已经知道您的用户池以令牌形式呈现群组的方式，因此您可以使用新策略允许任何新群组执行一系列操作。

您可能还想将基于群组的策略评估模型扩展为基于用户属性的更精确的模型。用户池令牌包含其他用户信息，这些信息可能有助于做出授权决策。

身份令牌

ID 令牌代表用户的属性，具有最高级别的精细访问控制。要评估电子邮件地址、电话号码或部门和经理等自定义属性，请评估 ID 令牌。

访问令牌

访问令牌代表用户在 OAuth 2.0 范围内的权限。要添加授权层或设置对额外资源的请求，请评估访问令牌。例如，您可以验证用户是否属于相应的群组，并且其范围通常用于授权 API 的访问权限。`PetStore.read`用户池可以使用资源服务器向令牌添加自定义范围，也可以在运行时自定义令牌。

在架构和策略中使用身份源有关处理 ID 和访问令牌声明的政策示例。

# API 关联策略存储的注意事项

在已验证权限控制台中构建与 API 关联的策略存储库时，您正在为最终的生产部署创建测试。在进入生产环境之前，请为 API 和用户池建立固定配置。请考虑以下因素：

API Gateway 会缓存响应

在与 API 关联的策略存储中，已验证权限会创建授权缓存 TTL 为 120 秒的 Lambda 授权方。您可以调整此值或在授权者中关闭缓存。在启用了缓存的授权方中，您的授权方每次都会返回相同的响应，直到 TTL 到期。这可以将用户池令牌的有效寿命延长一段时间，该持续时间等于请求阶段的缓存 TTL。

亚马逊 Cognito 群组可以重复使用

Amazon Verified Permissions 根据用户 ID 或访问令牌中的`cognito:groups`声明来确定用户池用户的群组成员资格。此声明的值是该用户所属的用户池组的友好名称数组。您无法将用户池组与唯一标识符相关联。

您删除并重新创建的用户池组与策略存储中的同名用户池组与同一个组相同。从用户池中删除组时，请从策略存储中删除对该组的所有引用。

API 派生的命名空间和架构是 point-in-time

Verified Permissions 会在某个时间点捕获您的 API：它仅在您创建策略存储时查询您的 API。当 API 的架构或名称发生变化时，您必须更新您的策略存储和 Lambda 授权机构，或者创建一个新的 API 关联策略存储。Verified Permissions 从您的 API 名称派生出策略存储命名空间。

Lambda 函数没有 VPC 配置

已验证权限为您的 API 授权方创建的 Lambda 函数未连接到 VPC。默认情况下。网络访问权限仅限于私有 VPC 的 API 无法与 Lambda 函数通信，该函数使用已验证的权限授权访问请求。

“已验证权限” 在中部署授权方资源 CloudFormation

要创建与 API 关联的策略存储，您必须使用高权限 AWS 委托人登录已验证权限控制台。该用户部署了一个 AWS CloudFormation 堆栈，该堆栈可以跨多个 AWS 服务堆栈创建资源。该委托人必须有权在已验证的权限、 IAM、Lambda 和 API Gateway 中添加和修改资源。作为最佳实践，请勿与组织中的其他管理员共享这些证书。

通过以下方式进入生产阶段 AWS CloudFormation有关已验证权限创建的资源的概述，请参阅。

## 通过以下方式进入生产阶段 AWS CloudFormation

与 API 关联的策略存储库是一种快速为 API Gateway API 构建授权模型的方法。它们旨在用作应用程序授权组件的测试环境。创建测试策略存储后，请花时间完善策略、架构和 Lambda 授权方。

您可能会调整 API 的架构，要求对策略存储架构和策略进行同等调整。与 API 关联的策略存储不会自动从 API 架构更新其架构，经过验证的权限仅在您创建策略存储时对 API 进行轮询。如果您的 API 更改得足够多，则可能需要使用新的策略存储库重复此过程。

当您的应用程序和授权模型准备好部署到生产环境时，请将您开发的 API 关联策略存储与自动化流程集成。作为最佳实践，我们建议您将策略存储架构和策略导出到可以部署到其他 AWS 账户 和的 AWS CloudFormation 模板中 AWS 区域。

与 API 关联的策略存储过程的结果是初始策略存储和 Lambda 授权者。Lambda 授权方有多个依赖资源。已验证权限将这些资源部署到自动 CloudFormation 生成的堆栈中。要部署到生产环境，您必须将策略存储和 Lambda 授权方资源收集到模板中。与 API 关联的策略存储由以下资源组成：

1. AWS::VerifiedPermissions::PolicyStore：将架构复制到`SchemaDefinition`对象。将"角色转义为\".
2. AWS::VerifiedPermissions::IdentitySource：从测试策略存储库的输出GetIdentitySource 中复制值，并根据需要进行修改。
3. 其中一项或多AWS::VerifiedPermissions::Policy项：将您的策略声明复制到`Definition`对象。将"角色转义为\".
4. AWS:: Lambda:: Func tion, AWS::IAM: Role, AWS:: Role,IAM:: Policy，AWS::ApiGateway: Authorizer，AWS::Lambda::Permission:: 从创建策略存储时部署的已验证权限的堆栈的 “模板” 选项卡中复制模板。

以下模板是示例策略存储。您可以将现有堆栈中的 Lambda 授权方资源附加到此模板中。

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
        "MyExamplePolicyStore": {
            "Type": "AWS::VerifiedPermissions::PolicyStore",
            "Properties": {
                "ValidationSettings": {
                    "Mode": "STRICT"
                },
                "Description": "ApiGateway: PetStore/test",
```

```
                "Schema": {
                    "CedarJson": "{\"PetStore\":{\"actions\":{\"get /pets\":
{\"appliesTo\":{\"principalTypes\":[\"User\"],\"resourceTypes\":[\"Application\"],
\"context\":{\"type\":\"Record\",\"attributes\":{}}}},\"get /\":{\"appliesTo\":
{\"principalTypes\":[\"User\"],\"resourceTypes\":[\"Application\"],\"context\":{\"type
\":\"Record\",\"attributes\":{}}}},\"get /pets/{petId}\":{\"appliesTo\":{\"context
\":{\"type\":\"Record\",\"attributes\":{}},\"resourceTypes\":[\"Application\"],
\"principalTypes\":[\"User\"]}},\"post /pets\":{\"appliesTo\":{\"principalTypes\":
[\"User\"],\"resourceTypes\":[\"Application\"],\"context\":{\"type\":\"Record\",
\"attributes\":{}}}}},\"entityTypes\":{\"Application\":{\"shape\":{\"type\":\"Record\",
\"attributes\":{}}},\"User\":{\"memberOfTypes\":[\"UserGroup\"],\"shape\":{\"attributes
\":{},\"type\":\"Record\"}},\"UserGroup\":{\"shape\":{\"type\":\"Record\",\"attributes
\":{}}}}}}"
                }
            }
        },
        "MyExamplePolicy": {
            "Type": "AWS::VerifiedPermissions::Policy",
            "Properties": {
                "Definition": {
                    "Static": {
                        "Description": "Policy defining permissions for testgroup
 cognito group",
                        "Statement": "permit(\nprincipal in PetStore::UserGroup::
\"us-east-1_EXAMPLE|testgroup\",\naction in [\n  PetStore::Action::\"get /\",
\n  PetStore::Action::\"post /pets\",\n  PetStore::Action::\"get /pets\",\n
 PetStore::Action::\"get /pets/{petId}\"\n],\nresource);"
                    }
                },
                "PolicyStoreId": {
                    "Ref": "MyExamplePolicyStore"
                }
            },
            "DependsOn": [
                "MyExamplePolicyStore"
            ]
        },
        "MyExampleIdentitySource": {
            "Type": "AWS::VerifiedPermissions::IdentitySource",
            "Properties": {
                "Configuration": {
                    "CognitoUserPoolConfiguration": {
                        "ClientIds": [
                            "1example23456789"
```

```
                        ],
                        "GroupConfiguration": {
                            "GroupEntityType": "PetStore::UserGroup"
                        },
                        "UserPoolArn": "arn:aws:cognito-idp:us-
east-1:123456789012:userpool/us-east-1_EXAMPLE"
                    }
                },
                "PolicyStoreId": {
                    "Ref": "MyExamplePolicyStore"
                },
                "PrincipalEntityType": "PetStore::User"
            },
            "DependsOn": [
                "MyExamplePolicyStore"
            ]
        }
    }
}
```

## 对 API 关联策略存储进行故障排除

使用此处的信息来帮助您诊断和修复构建与 Amazon Verified Permissions API 关联的策略存储库时的常见问题。

主题

- 我更新了政策，但授权决定没有改变
- 我将 Lambda 授权器附加到我的 API 中，但它没有生成授权请求
- 我收到了意想不到的授权决定，想查看授权逻辑
- 我想从我的 Lambda 授权机构那里查找日志
- 我的 Lambda 授权机构不存在
- 我的 API 位于私有 VPC 中，无法调用授权方
- 我想在我的授权模型中处理其他用户属性
- 我想添加新的操作、操作上下文属性或资源属性

## 我更新了政策，但授权决定没有改变

默认情况下，已验证权限将 Lambda 授权机构配置为将授权决策缓存 120 秒。两分钟后重试，或者在授权方上禁用缓存。有关更多信息，请参阅 Amazon API Gateway 开发者指南中的启用 API 缓存以增强响应能力。

## 我将 Lambda 授权器附加到我的 API 中，但它没有生成授权请求

要开始处理请求，您必须部署授权者所连接的 API 阶段。有关更多信息，请参阅 Amazon API Gateway 开发者指南中的部署 RES T API。

## 我收到了意想不到的授权决定，想查看授权逻辑

与 API 关联的策略存储流程会为您的授权方创建一个 Lambda 函数。已验证的权限会自动将您的授权决策逻辑构建到授权者函数中。创建策略存储后，您可以返回以查看和更新函数中的逻辑。

要从 AWS CloudFormation 控制台中找到您的 Lambda 函数，请在新策略存储的概述页面上选择检查部署按钮。

您也可以在 AWS Lambda 控制台中找到您的函数。导航到策略存储 AWS 区域 中的控制台，搜索前缀为的函数名称AVPAuthorizerLambda。如果您创建了多个与 API 关联的策略存储，请使用函数的上次修改时间将其与策略存储的创建相关联。

## 我想从我的 Lambda 授权机构那里查找日志

Lambda 函数收集指标并将其调用结果记录在亚马逊中。 CloudWatch要查看您的日志，请在 Lambda 控制台中找到您的函数，然后选择监控选项卡。选择查看 CloudWatch 日志并查看日志组中的条目。

有关 Lambda 函数日志的更多信息，请参阅AWS Lambda 开发者指南 AWS Lambda中的将 Amazon CloudWatch 日志与一起使用。

## 我的 Lambda 授权机构不存在

完成 API 关联策略存储的设置后，您必须将 Lambda 授权方附加到您的 API。如果您在 API Gateway 控制台中找不到授权方，则您的策略存储的其他资源可能已失效或尚未部署。与 API 关联的策略存储将这些资源部署在堆栈中。 AWS CloudFormation

在创建过程结束时，已验证权限会显示一个标有 “检查部署” 标签的链接。如果您已经离开了此屏幕，请前往 CloudFormation 控制台，在最近的堆栈中搜索前缀为的名称。AVPAuthorizer-<policy store ID> CloudFormation 在堆栈部署的输出中提供了宝贵的故障排除信息。

有关 CloudFormation 堆栈故障排除的帮助，请参阅《AWS CloudFormation 用户指南》CloudFormation中的故障排除。

### 我的 API 位于私有 VPC 中，无法调用授权方

已验证权限不支持通过 VPC 终端节点访问 Lambda 授权方。您必须在您的 API 和用作授权方的 Lambda 函数之间打开一条网络路径。

### 我想在我的授权模型中处理其他用户属性

与 API 关联的策略存储流程从用户令牌中的群组声明中派生出经过验证的权限策略。要更新您的授权模型以考虑其他用户属性，请将这些属性集成到您的策略中。

您可以将 Amazon Cognito 用户池中的 ID 和访问令牌中的许多声明映射到已验证的权限政策声明。例如，大多数用户的 ID 令牌中都有`email`声明。有关将来自您的身份来源的索赔添加到政策的更多信息，请参阅在架构和策略中使用身份源。

### 我想添加新的操作、操作上下文属性或资源属性

与 API 关联的策略存储库及其创建的 Lambda 授权方是一种资源。 point-in-time它们反映了创建 API 时的状态。策略存储架构不会为操作分配任何上下文属性，也不会为默认`Application`资源分配任何属性或父项。

向 API 添加操作（路径和方法）时，必须更新策略存储库以了解新操作。您还必须更新您的 Lambda 授权机构以处理新操作的授权请求。您可以重新开始使用新的保单存储，也可以更新现有的保单存储。

要更新现有的政策存储，请找到您的函数。检查自动生成的函数中的逻辑，并对其进行更新以处理新的操作、属性或上下文。然后编辑您的架构，使其包含新的操作和属性。

## 切换 Verified Permissions 策略存储

AWS Management Console

要切换策略存储或创建其他策略存储，请按以下步骤操作：

1. 打开 Verified Permissions 控制台，网址为 https://console.aws.amazon.com/verifiedpermissions/。选择您的保单存储。
2. 在左侧的导航窗格中，选择当前策略存储旁边的切换。
3. 您可以在现有策略存储之间进行切换，或者创建其他策略存储。

- 要切换策略存储，请选择要切换到的策略存储的策略存储 ID。
- 要创建新的策略存储，请选择创建新策略存储。按照创建 Verified Permissions 策略存储中的说明进行操作。

AWS CLI

要切换策略存储或创建其他策略存储，请按以下步骤操作：

AWS CLI 不维护“默认”策略存储。相反，大多数 AWS CLI 命令使用 `--policy-store-id` 来指定每个命令使用哪个策略存储。

要创建新的策略存储，请使用create-policy-store命令。

# 删除 Verified Permissions 策略存储

AWS Management Console

要删除策略存储，请按以下步骤操作：

1. 打开 Verified Permissions 控制台，网址为 https://console.aws.amazon.com/verifiedpermissions/。选择您的保单存储。
2. 在左侧的导航窗格中，选择设置。
3. 选择删除此策略存储。
4. 在文本框中输入 `delete`，然后选择删除。

AWS CLI

要删除策略存储，请按以下步骤操作：

您可以使用 `delete-policy-store` 操作删除策略存储。

```
$ aws verifiedpermissions delete-policy-store \
    --policy-store-id PSEXAMPLEabcdefg111111
```

如果成功，此命令不会产生任何输出。

# Amazon Verified Permissions 策略存储架构

架构是对应用程序支持的实体类型结构以及应用程序在授权请求中可能提供的操作的声明。

有关更多信息，请参阅《Cedar 策略语言参考指南》中的 Cedar 架构格式。

> **Note**
>
> 您可以自由选择是否在 Verified Permissions 中使用架构，但我们强烈建议您在生产软件中使用架构。创建新策略时，Verified Permissions 可以使用架构来验证范围和条件中引用的实体和属性，以避免策略中出现可能导致系统行为混乱的错别字和错误。如果您激活策略验证，则所有新策略都必须符合该架构。

### AWS Management Console

要创建架构，请按以下步骤操作：

1. 打开 Verified Permissions 控制台，网址为 https://console.aws.amazon.com/verifiedpermissions/。选择您的保单商店。
2. 在左侧导航窗格中，选择架构。
3. 选择创建架构。

### AWS CLI

要提交新架构，或使用 AWS CLI覆盖现有架构，请按以下步骤操作：

您可以通过运行类似于以下示例的 AWS CLI 命令来创建策略存储。

考虑使用包含以下 Cedar 内容的架构：

```
{
    "MySampleNamespace": {
        "actions": {
            "remoteAccess": {
                "appliesTo": {
                    "principalTypes": [ "Employee" ]
                }
            }
```

```
        },
        "entityTypes": {
            "Employee": {
                "shape": {
                    "type": "Record",
                    "attributes": {
                        "jobLevel": {"type": "Long"},
                        "name": {"type": "String"}
                    }
                }
            }
        }
    }
}
```

您必须先将 JSON 转义为单行字符串，并在其前面加上其数据类型的声明：cedarJson。以下示例使用 schema.json 文件中的以下内容，该文件包含 JSON 架构的转义版本。

> **Note**
>
> 为了便于阅读，此处的示例采用的是换行格式。您必须将整个文件放在一行上，这样命令才能接受。

```
{"cedarJson": "{\"MySampleNamespace\": {\"actions\": {\"remoteAccess\": {\"appliesTo
\":
{\"principalTypes\": [\"Employee\"]}}},\"entityTypes\": {\"Employee\": {\"shape\":
{\"attributes\": {\"jobLevel\": {\"type\": \"Long\"},\"name\": {\"type\": \"String
\"}},
\"type\": \"Record\"}}}}}"}
```

```
$ aws verifiedpermissions put-schema \
    --definition file://schema.json \
    --policy-store PSEXAMPLEabcdefg111111
{
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "namespaces": [
        "MySampleNamespace"
    ],
    "createdDate": "2023-07-17T21:07:43.659196+00:00",
    "lastUpdatedDate": "2023-08-16T17:03:53.081839+00:00"
```

```
}
```

AWS SDKs

您可以使用 PutSchema API 创建策略存储。有关更多信息，请参阅PutSchema《Amazon 已验证权限 API 参考指南》。

# 在可视模式下编辑架构

在 “已验证权限” 控制台中选择 “架构” 时，可视模式会显示构成架构的实体类型和操作。在此顶级视图或任何实体的详细信息中，您可以选择 “编辑架构” 以开始更新架构。可视模式不适用于某些架构格式，例如嵌套记录。

可视架构编辑器以一系列图表开头，这些图表说明架构中各实体之间的关系。选择 “展开” 可最大限度地了解架构的实体关系。

操作图

操作图表视图列出了您在策略存储中配置的委托人类型、他们有资格执行的操作以及他们有资格对其执行操作的资源。实体之间的界限表明您有能力创建允许委托人对资源采取操作的策略。如果您的操作图未显示两个实体之间的关系，则必须先在它们之间创建这种关系，然后才能在策略中允许或拒绝这种关系。选择一个实体以查看属性概述，然后向下钻取以查看全部详细信息。选择 “按此 [操作 | 资源类型 | 主体类型] 筛选”，即可在视图中查看只有其自身连接的实体。

实体类型图

实体类型图侧重于委托人与资源之间的关系。如果您想了解架构中复杂的嵌套父关系，请查看此图。将鼠标悬停在实体上方可深入了解该实体拥有的父关系。

图表下方是架构中实体类型和操作的列表视图。当您想要立即查看特定操作或实体类型的详细信息时，列表视图非常有用。选择任何实体以查看详细信息。

要在可视模式下编辑 Verified Permissions 架构，请按以下步骤操作：

1. 打开 Verified Permissions 控制台，网址为 https://console.aws.amazon.com/verifiedpermissions/。选择您的保单商店。
2. 在左侧导航窗格中，选择架构。
3. 选择可视模式。查看实体关系图并计划要对架构进行的更改。您可以选择按一个实体进行筛选，以检查其与其他实体的各个连接。

4. 选择 Edit schema。
5. 在详细信息部分中，为您的架构输入命名空间。
6. 在实体类型部分中，选择添加新实体类型。
7. 输入实体的名称。
8. （可选）选择添加父级，以添加新实体所属的父实体。要删除已添加到该实体的父实体，请选择该父实体名称旁边的删除。
9. 选择添加属性，为该实体添加属性。输入属性名称，然后为该实体的每个属性选择属性类型。根据架构验证策略时，Verified Permissions 会使用指定的属性值。选择每个属性是否为必填项。要删除已为该实体添加的属性，请选择该属性旁边的删除。
10. 选择添加实体类型，将该实体添加到架构中。
11. 在操作部分中，选择添加新操作。
12. 输入操作的名称。
13. （可选）选择添加资源，添加该操作适用的资源类型。要删除已为该操作添加的资源类型，请选择该资源类型名称旁边的删除。
14. （可选）选择添加主体，添加该操作适用的主体类型。要删除已为该操作添加的主体类型，请选择该主体类型名称旁边的删除。
15. 选择添加属性以添加可添加到授权请求中操作上下文的属性。输入属性名称并为每个属性选择属性类型。根据架构验证策略时，Verified Permissions 会使用指定的属性值。选择每个属性是否为必填项。要删除已为该操作添加的属性，请选择该属性旁边的删除。
16. 选择添加操作。
17. 为该架构添加完所有实体类型和操作后，选择保存更改。

# 在 JSON 模式下编辑架构

要在 JSON 模式下编辑 Verified Permissions 架构，请按以下步骤操作：

1. 打开 Verified Permissions 控制台，网址为 https://console.aws.amazon.com/verifiedpermissions/。选择您的保单商店。
2. 在左侧导航窗格中，选择架构。
3. 选择 JSON 模式，然后选择编辑架构。
4. 在内容字段中输入 JSON 架构的内容。只有解决完所有语法错误，您才能保存架构更新。您可以选择格式 JSON，使用建议的间距和缩进来为架构的 JSON 语法设置格式。
5. 选择保存更改。

# 删除架构

## AWS Management Console

要删除 Verified Permissions 架构，请按以下步骤操作：

1. 打开 Verified Permissions 控制台，网址为 https://console.aws.amazon.com/verifiedpermissions/。选择您的保单商店。
2. 在左侧导航窗格中，选择架构。
3. 选择删除架构。

## AWS CLI

要删除 Verified Permissions 架构，请按以下步骤操作：

没有用于删除架构的命令。您可以使用 `cedarJson` 字段中包含空架构的 `put-schema` 命令来删除策略存储中的架构。空架构用一对大括号“{}”表示。

```
$ aws verifiedpermissions put-schema \
    --policy-store-id PSEXAMPLEabcdefg111111 \
    --definition cedarJson='{}'{
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "namespaces": [],
    "createdDate": "2023-06-14T21:55:27.347581Z",
    "lastUpdatedDate": "2023-06-19T17:55:04.95944Z"
}
```

# Amazon Verified Permissions 策略验证模式

您可以在 Verified Permissions 中设置策略验证模式，以控制是否根据策略存储中的架构验证策略更改。

> **Important**
>
> 启用策略验证后，所有创建或更新策略或策略模板的尝试都将根据策略存储中的架构进行验证。如果验证失败，Verified Permissions 会拒绝该请求。

### AWS Management Console

要为策略存储设置策略验证模式，请按以下步骤操作：

1. 打开 Verified Permissions 控制台，网址为 https://console.aws.amazon.com/verifiedpermissions/。选择您的保单商店。
2. 选择设置。
3. 在策略验证模式部分中，选择修改。
4. 请执行以下操作之一：
   - 要激活策略验证并强制所有策略更改都必须根据架构进行验证，请选择严格（推荐）单选按钮。
   - 要关闭策略更改的策略验证，请选择关闭单选按钮。输入 `confirm`，确认策略更新将不再根据您的架构进行验证。
5. 选择保存更改。

### AWS CLI

要为策略存储设置验证模式，请按以下步骤操作：

您可以通过使用UpdatePolicyStore操作并为ValidationSettings参数指定不同的值来更改策略存储的验证模式。

```
$ aws verifiedpermissions update-policy-store \
    --validation-settings "mode=OFF",
    --policy-store-id PSEXAMPLEabcdefg111111
```

```
{
    "createdDate": "2023-05-17T18:36:10.134448+00:00",
    "lastUpdatedDate": "2023-05-17T18:36:10.134448+00:00",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "validationSettings": {
        "Mode": "OFF"
    }
}
```

有关更多信息，请参阅《Cedar 策略语言参考指南》中的策略验证。

# Amazon Verified Permissions 策略

策略是一种允许或禁止主体对资源采取一项或多项操作的语句。每个策略的评估都独立于任何其他策略。有关 Cedar 策略的结构和评估方式的更多信息，请参阅《Cedar 策略语言参考指南》中的根据架构验证 Cedar 策略。

**Important**

在编写引用主体、资源和操作的 Cedar 策略时，您可以定义用于每个元素的唯一标识符。我们强烈建议您遵循以下最佳实践：

- 为所有主体和资源标识符使用诸如通用唯一标识符（UUID）之类的值。

  例如，如果用户 `jane` 离开公司，而您后来让其他人使用 `jane` 这个名称，那么，该新用户将自动获得仍引用 `User::"jane"` 的策略所授予的所有内容的访问权限。Cedar 无法区分新用户和旧用户。这同时适用于主体标识符和资源标识符。请务必使用保证唯一且永远不可重复使用的标识符，确保您不会因为策略中存在旧标识符而在无意中授予他人访问权限。

  在为实体使用 UUID 时，我们建议您在它后面加上 // 注释说明符和实体的“友好”名称。这有助于使您的策略更易于理解。例如：principal == User::"a1b2c3d4-e5f6-a1b2-c3d4-EXAMPLE11111", // alice

- 请勿在主体或资源的唯一标识符中包含个人识别信息、机密信息或敏感信息。这些标识符包含在 AWS CloudTrail 跟踪中共享的日志条目中。

# Amazon Verified Permissions 中的实体格式设置

Amazon Verified Permissions 使用 Cedar 策略语言来创建策略。策略的语法和支持的数据类型与《Cedar 策略语言参考指南》Cedar 中的基本策略构造和 Cedar 支持的数据类型中概述的语法和数据类型相匹配。但是，在发出授权请求时，Verified Permissions 和 Cedar 在实体格式上存在差异。

Verified Permissions 中实体的 JSON 格式与 Cedar 有以下不同：

- 在 Verified Permissions 中，JSON 对象必须将其所有键值对包装在名为 `Record` 的 JSON 对象中。
- Verified Permissions 中的 JSON 列表必须包装在 JSON 键值对中，其中，键名称为 `Set`，值为来自 Cedar 的原始 JSON 列表。

- 对于 String、Long 和 Boolean 类型名称，在 Verified Permissions 中，Cedar 中的每个键值对都会被替换为 JSON 对象。该对象的名称是原始键名称。在 JSON 对象中，有一个键值对，其中键名称是标量值（String、Long 或 Boolean）的类型名称，值是来自 Cedar 实体的值。
- Cedar 实体和 Verified Permissions 实体的语法格式存在以下不同：

| Cedar 格式 | Verified Permissions 格式 |
|---|---|
| uid | Identifier |
| type | EntityType |
| id | EntityId |
| attrs | Attributes |
| parents | Parents |

以下示例显示了如何使用 Cedar 设置列表中实体的格式。

```
[
  {
    "number": 1
  },
  {
    "sentence": "Here is an example sentence"
  },
  {
    "Question": false
  }
]
```

以下示例显示了前面 Cedar 列表示例中的相同实体在 Verified Permissions 中的格式。

```
{
  "Set": [
    {
      "Record": {
        "number": {
          "Long": 1
        }
```

```
      }
    },
    {
      "Record": {
        "sentence": {
          "String": "Here is an example sentence"
        }
      }
    },
    {
      "Record": {
        "question": {
          "Boolean": false
        }
      }
    }
  ]
}
```

以下示例显示了在授权请求中评估策略时，如何设置 Cedar 实体的格式。

```
[
    {
        "uid": {
            "type": "PhotoApp::User",
            "id": "alice"
        },
        "attrs": {
            "age": 25,
            "name": "alice",
            "userId": "123456789012"
        },
        "parents": [
            {
                "type": "PhotoApp::UserGroup",
                "id": "alice_friends"
            },
            {
                "type": "PhotoApp::UserGroup",
                "id": "AVTeam"
            }
        ]
    },
```

```
    {
        "uid": {
            "type": "PhotoApp::Photo",
            "id": "vacationPhoto.jpg"
        },
        "attrs": {
            "private": false,
            "account": {
                "__entity": {
                    "type": "PhotoApp::Account",
                    "id": "ahmad"
                }
            }
        },
        "parents": []
    },
    {
        "uid": {
            "type": "PhotoApp::UserGroup",
            "id": "alice_friends"
        },
        "attrs": {},
        "parents": []
    },
    {
        "uid": {
            "type": "PhotoApp::UserGroup",
            "id": "AVTeam"
        },
        "attrs": {},
        "parents": []
    }
]
```

以下示例显示了前面 Cedar 示例中的相同实体在 Verified Permissions 中的格式。

```
[
    {
        "Identifier": {
            "EntityType": "PhotoApp::User",
            "EntityId": "alice"
        },
        "Attributes": {
```

```
            "age": {
                "Long": 25
            },
            "name": {
                "String": "alice"
            },
            "userId": {
                "String": "123456789012"
            }
        },
        "Parents": [
            {
                "EntityType": "PhotoApp::UserGroup",
                "EntityId": "alice_friends"
            },
            {
                "EntityType": "PhotoApp::UserGroup",
                "EntityId": "AVTeam"
            }
        ]
    },
    {
        "Identifier": {
            "EntityType": "PhotoApp::Photo",
            "EntityId": "vacationPhoto.jpg"
        },
        "Attributes": {
            "private": {
                "Boolean": false
            },
            "account": {
                "EntityIdentifier": {
                    "EntityType": "PhotoApp::Account",
                    "EntityId": "ahmad"
                }
            }
        },
        "Parents": []
    },
    {
        "Identifier": {
            "EntityType": "PhotoApp::UserGroup",
            "EntityId": "alice_friends"
        },
```

```
        "Parents": []
    },
    {
        "Identifier": {
            "EntityType": "PhotoApp::UserGroup",
            "EntityId": "AVTeam"
        },
        "Parents": []
    }
]
```

# 创建 Amazon Verified Permissions 静态策略

您可以创建 Cedar 静态策略，以允许或拒绝主体对应用程序的指定资源执行指定操作。

AWS Management Console

要创建静态策略，请按以下步骤操作：

1. 打开 Verified Permissions 控制台，网址为 https://console.aws.amazon.com/verifiedpermissions/。选择您的保单商店。
2. 在左侧的导航窗格中，选择策略。
3. 选择创建策略，然后选择创建静态策略。
4. 在策略效果部分，选择当请求与策略匹配时，策略是允许还是禁止。
5. 在主体范围字段中，选择策略将适用的主体范围。
   - 选择特定主体，将策略应用于特定主体。为将被允许或禁止执行策略中指定操作的主体指定实体类型和标识符。
   - 选择主体群组，将策略应用于一组主体。在主体群组字段中输入主体群组名称。
   - 选择所有主体，将该策略应用于策略存储中的所有主体。
6. 在资源范围字段中，选择策略将适用的资源范围。
   - 选择特定资源，将该策略应用于特定资源。为策略应适用的资源指定实体类型和标识符。
   - 选择资源群组，将该策略应用于一组资源。在资源群组字段中输入资源群组名称。
   - 选择所有资源，将该策略应用于策略存储中的所有资源。
7. 在操作范围部分中，选择策略将适用的资源范围。
   - 选择特定操作集合，将该策略应用于一组操作。选中操作旁边的复选框，应用该策略。

- 选择所有操作，将该策略应用于策略存储中的所有操作。

8. 选择下一步。
9. 在策略部分中，查看您的 Cedar 策略。您可以选择格式，使用建议的间距和缩进来设置策略语法的格式。有关更多信息，请参阅《Cedar 策略语言参考指南》中的 Cedar 中的基本策略构造。
10. 在详细信息部分中，输入策略的可选描述。
11. 选择创建策略。

AWS CLI

要创建静态策略，请按以下步骤操作：

您可以使用CreatePolicy操作创建静态策略。以下示例创建了一个简单的静态策略。

```
$ aws verifiedpermissions create-policy \
    --definition "{ \"static\": { \"Description\": \"MyTestPolicy\", \"Statement\":
 \"permit(principal,action,resource) when {principal.owner == resource.owner};\"}}"
 \
    --policy-store-id PSEXAMPLEabcdefg111111
{
"Arn": "arn:aws:verifiedpermissions::123456789012:policy/PSEXAMPLEabcdefg111111/
SPEXAMPLEabcdefg111111",
    "createdDate": "2023-05-16T20:33:01.730817+00:00",
    "lastUpdatedDate": "2023-05-16T20:33:01.730817+00:00",
    "policyId": "SPEXAMPLEabcdefg111111",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "policyType": "STATIC"
}
```

# 编辑 Amazon Verified Permissions 静态策略

您可以编辑策略存储中现有的 Cedar 静态策略。您只能直接更新静态策略。您只能更改静态策略的某些元素：

- 策略所引用的 `action`。
- 条件子句，例如 `when` 和 `unless`。

您无法更改静态策略的以下元素：

- 将策略从静态策略更改为模板链接策略。
- 将静态策略的效果从 `permit` 更改为 `forbid`。
- 静态策略所引用的 `principal`。
- 静态策略所引用的 `resource`。

要更改模板链接策略，您必须更新该模板。有关更多信息，请参阅 编辑策略模板。

AWS Management Console

要编辑静态策略，请按以下步骤操作：

1. 打开 Verified Permissions 控制台，网址为 https://console.aws.amazon.com/verifiedpermissions/。选择您的保单商店。
2. 在左侧的导航窗格中，选择策略。
3. 选中要编辑的静态策略旁边的单选按钮，然后选择编辑。
4. 在策略正文部分中，更新静态策略的 `action` 或条件子句。您无法更新策略效果，以及策略的 `principal` 或 `resource`。
5. 选择更新策略。

> **Note**
>
> 如果在策略存储中启用了策略验证，则更新静态策略会导致 Verified Permissions 针对策略存储中的架构验证策略。如果更新后的静态策略未通过验证，则操作将失败，并且不会保存更新。

AWS CLI

要编辑静态策略，请按以下步骤操作：

您可以使用UpdatePolicy操作编辑静态策略。以下示例编辑了一个简单的静态策略。

该示例使用 `definition.txt` 文件来包含策略定义。

```
{
```

```
    "static": {
        "description":  "Grant everyone of janeFriends UserGroup access to the
 vacationFolder Album",
        "statement": "permit(principal in UserGroup::\"janeFriends\", action,
 resource in Album::\"vacationFolder\" );"
    }
}
```

以下命令引用了该文件。

```
$ aws verifiedpermissions create-policy \
    --definition file://definition.txt \
    --policy-store-id PSEXAMPLEabcdefg111111

{
    "createdDate": "2023-06-12T20:33:37.382907+00:00",
    "lastUpdatedDate": "2023-06-12T20:33:37.382907+00:00",
    "policyId": "SPEXAMPLEabcdefg111111",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "policyType": "STATIC",
    "principal": {
        "entityId": "janeFriends",
        "entityType": "UserGroup"
    },
    "resource": {
        "entityId": "vacationFolder",
        "entityType": "Album"
    }
}
```

# 查看策略

## AWS Management Console

要查看您的 Verified Permissions 策略，请按以下步骤操作：

1. 打开 Verified Permissions 控制台，网址为 https://console.aws.amazon.com/verifiedpermissions/。选择您的保单商店。
2. 在左侧的导航窗格中，选择策略。此时，系统将显示您创建的所有策略。
3. 选择搜索文本框，按主体或资源筛选策略。

4. 选择策略旁边的单选按钮，以显示有关该策略的详细信息，例如策略的创建和更新时间，以及策略内容。
5. 您可以通过选择策略旁边的单选按钮，然后选择删除来删除策略。选择删除策略，确认删除该策略。

### AWS CLI

要列出策略存储中的所有可用策略，请按以下步骤操作：

您可以使用GetPolicy操作查看策略列表。以下示例会检索包含静态策略和模板链接策略的列表。

```
$ aws verifiedpermissions list-policies \
    --policy-store-id PSEXAMPLEabcdefg111111
{
    "Policies": [
        {
            "createdDate": "2023-05-17T18:38:31.359864+00:00",
            "definition": {
                "static": {
                    "Description": "Grant everyone of janeFriends UserGroup access
 to the vacationFolder Album"
                }
            },
            "lastUpdatedDate": "2023-05-18T16:15:04.366237+00:00",
            "policyId": "SPEXAMPLEabcdefg111111",
            "policyStoreId": "PSEXAMPLEabcdefg111111",
            "policyType": "STATIC",
            "resource": {
                "entityId": "publicFolder",
                "entityType": "Album"
            }
        },
        {
            "createdDate": "2023-05-22T18:57:53.298278+00:00",
            "definition": {
                "templateLinked": {
                    "policyTemplateId": "PTEXAMPLEabcdefg111111"
                }
            },
            "lastUpdatedDate": "2023-05-22T18:57:53.298278+00:00",
            "policyId": "TPEXAMPLEabcdefg111111",
            "policyStoreId": "PSEXAMPLEabcdefg111111",
```

```
            "policyType": "TEMPLATELINKED",
            "principal": {
                "entityId": "alice",
                "entityType": "User"
            },
            "resource": {
                "entityId": "VacationPhoto94.jpg",
                "entityType": "Photo"
            }
        }
    ]
}
```

要查看单个策略的详细信息，请按以下步骤操作：

您可以使用GetPolicy操作检索策略的详细信息。以下示例会检索模板链接策略的详细信息。

```
$ aws verifiedpermissions get-policy \
    --policy-id TPEXAMPLEabcdefg111111
    --policy-store-id PSEXAMPLEabcdefg111111

{
    "arn": "arn:aws:verifiedpermissions::123456789012:policy/PSEXAMPLEabcdefg111111/
TPEXAMPLEabcdefg111111",
    "createdDate": "2023-03-15T16:03:07.620867Z",
    "lastUpdatedDate": "2023-03-15T16:03:07.620867Z",
    "policyDefinition": {
        "templatedPolicy": {
            "policyTemplateId": "PTEXAMPLEabcdefg111111",
            "principal": {
                "entityId": "alice",
                "entityType": "User"
            },
            "resource": {
                "entityId": "Vacation94.jpg",
                "entityType": "Photo"
            }
        }
    },
    "policyId": "TPEXAMPLEabcdefg111111",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "policyType": "TEMPLATELINKED",
    "principal": {
        "entityId": "alice",
```

```
        "entityType": "User"
    },
    "resource": {
        "entityId": "Vacation94.jpg",
        "entityType": "Photo"
    }
}
```

# Amazon Verified Permissions 示例策略

以下 “已验证权限” 策略示例基于为 Cedar 策略语言参考指南的 “示例架构” 部分中 PhotoFlash 描述的假设应用程序定义的架构。有关 Cedar 策略语法的更多信息，请参阅《Cedar 策略语言参考指南》中的 Cedar 中的基本策略构造。

策略示例

- 允许单个实体访问
- 允许实体组访问
- 允许任何实体访问
- 允许访问实体的属性（ABAC）
- 拒绝访问

## 允许单个实体访问

此示例说明了如何创建一个策略，来允许用户 `alice` 查看照片 `VacationPhoto94.jpg`。

```
permit(
  principal == User::"alice",
  action == Action::"view",
  resource == Photo::"VacationPhoto94.jpg"
);
```

## 允许实体组访问

此示例说明了如何创建一个策略，来允许 `alice_friends` 组中的任何人查看照片 `VacationPhoto94.jpg`。

```
permit(
```

```
  principal in Group::"alice_friends",
  action == Action::"view",
  resource == Photo::"VacationPhoto94.jpg"
);
```

此示例说明了如何创建一个策略，来允许用户 alice 查看相册 alice_vacation 中的任何照片。

```
permit(
  principal == User::"alice",
  action == Action::"view",
  resource in Album::"alice_vacation"
);
```

此示例说明了如何创建一个策略，来允许用户 alice 查看、编辑或删除相册 alice_vacation 中的任何照片。

```
permit(
  principal == User::"alice",
  action in [Action::"view", Action::"edit", Action::"delete"],
  resource in Album::"alice_vacation"
);
```

此示例说明了如何创建一个策略，来允许用户 alice 获得相册 alice_vacation 的权限，其中，admin 是在架构层次结构中定义的群组，包含查看、编辑和删除照片的权限。

```
permit(
  principal == User::"alice",
  action in PhotoflashRole::"admin",
  resource in Album::"alice_vacation"
);
```

此示例说明了如何创建一个策略，来允许用户 alice 获得相册 alice_vacation 的权限，其中，viewer 是在架构层次结构中定义的群组，包含查看和评论照片的权限。策略中列出的第二个操作还会向用户 alice 授予 edit 权限。

```
permit(
  principal == User::"alice",
  action in [PhotoflashRole::"viewer", Action::"edit"],
  resource in Album::"alice_vacation"
)
```

## 允许任何实体访问

此示例说明了如何创建一个策略，来允许任何经过身份验证的主体查看相册 alice_vacation。

```
permit(
  principal,
  action == Action::"view",
  resource in Album::"alice_vacation"
);
```

此示例说明了如何创建一个策略，来允许用户 alice 列出 jane 账户中的所有相册、列出每个相册中的照片以及查看账户中的照片。

```
permit(
  principal == User::"alice",
  action in [Action::"listAlbums", Action::"listPhotos", Action::"view"],
  resource in Account::"jane"
);
```

此示例说明了如何创建一个策略，来允许用户 alice 对相册 jane_vaction 中的资源执行任何操作。

```
permit(
  principal == User::"alice",
  action,
  resource in Album::"jane_vacation"
);
```

## 允许访问实体的属性（ABAC）

基于属性的访问控制 (ABAC) 是一种授权策略，该策略基于属性来定义权限。Verified Permissions 允许将属性附加到主体、操作和资源。然后，可以在策略的 when 和 unless 子句中引用这些属性，这些策略会评估构成请求上下文的主体、操作和资源的属性。

以下示例使用了在 Cedar 策略语言参考指南的 “示例架构” 部分中 PhotoFlash 描述的假设应用程序中定义的属性。

此示例说明了如何创建一个策略，来允许 HardwareEngineering 部门中任何职级高于或等于 5 的主体查看和列出相册 device_prototypes 中的照片。

```
permit(
```

```
  principal,
  action in [Action::"listPhotos", Action::"view"],
  resource in Album::"device_prototypes"
)
when {
  principal.department == "HardwareEngineering" &&
  principal.jobLevel >= 5
};
```

此示例说明了如何创建一个策略，来允许用户 alice 查看文件类型 JPEG 的任何资源。

```
permit(
  principal == User::"alice",
  action == Action::"view",
  resource
)
when {
  resource.fileType == "JPEG"
};
```

操作具有上下文属性。您必须在授权请求context中传递这些属性。此示例说明如何创建允许用户执行任何readOnly操作alice的策略。您也可以为架构中的操作设置appliesTo属性。例如，当您要确保用户只能尝试为该类型的资源进行授权时，这会ViewPhoto为资源指定有效的操作PhotoFlash::Photo。

```
permit(
  principal == PhotoFlash::User::"alice",
  action,
  resource
) when {
    context has readOnly &&
    context.readOnly == true
};
```

但是，在架构中设置操作属性的更好方法是将它们排列到功能操作组中。例如，您可以创建一个名为ReadOnlyPhotoAccess并设置PhotoFlash::Action::"ViewPhoto"为操作组成员的ReadOnlyPhotoAccess动作。此示例说明如何创建策略，授予 Alice 访问该组中只读操作的权限。

```
permit(
  principal == PhotoFlash::User::"alice",
```

```
  action,
  resource
) when {
    action in PhotoFlash::Action::"ReadOnlyPhotoAccess"
};
```

此示例说明了如何创建一个策略，来允许所有主体对拥有 owner 属性的资源执行任何操作。

```
permit(
  principal,
  action,
  resource
)
when {
  principal == resource.owner
};
```

此示例说明了如何创建一个策略，来允许任何主体查看任何资源，前提是该主体的 department 属性与该资源的 department 属性相匹配。

> **Note**
>
> 如果某个实体没有在策略条件中提及的属性，则在做出授权决策时，该策略将被忽略，而对该实体的策略评估将会失败。例如，此策略不能向任何没有 department 属性的主体授予对任何资源的访问权限。

```
permit(
  principal,
  action == Action::"view",
  resource
)
when {
  principal.department == resource.owner.department
};
```

此示例说明了如何创建一个策略，来允许任何主体对某个资源执行任何操作，前提是该主体是该资源的 owner 或者该主体是该资源 admins 组的一员。

```
permit(
```

```
  principal,
  action,
  resource,
)
when {
  principal == resource.owner |
  resource.admins.contains(principal)
};
```

## 拒绝访问

如果某个策略的效果是 `forbid`，则它会限制权限，而不是授予权限。

> **Important**
>
> 在授权期间，如果同时执行 `permit` 和 `forbid` 策略，则 `forbid` 优先。

以下示例使用了在 Cedar 策略语言参考指南的 “示例架构” 部分中 PhotoFlash 描述的假设应用程序中定义的属性。

此示例说明了如何创建一个策略，以拒绝用户 `alice` 对任何资源执行除 `readOnly` 以外的所有操作。

```
forbid (
  principal == User::"alice",
  action,
  resource
)
unless {
  action.readOnly
};
```

此示例说明了如何创建一个策略，以拒绝访问具有 `private` 属性的所有资源，主体具有该资源 `owner` 属性的情况除外。

```
forbid (
  principal,
  action,
  resource
)
when {
```

```
  resource.private
}
unless {
  principal == resource.owner
};
```

# Amazon Verified Permissions 策略模板

您可以在 Verified Permissions 中创建 Cedar 策略模板，为您的系统定义访问控制规则。策略模板是带 `principal` 和/或 `resource` 占位符的 Cedar 策略。策略模板允许定义一次策略，然后将其附加到多个主体和资源。策略模板更新会反映在使用该模板的所有主体和资源中。有关更多信息，请参阅《Cedar 策略语言参考指南》中的 Cedar 策略模板。

我们建议使用策略模板来创建可在整个应用程序中共享的策略。例如，您可以为编辑器创建策略模板，该模板为使用该策略模板的主体和资源提供读取、编辑和评论权限。

```
permit(
  principal == ?principal,
  action in [Action::"Read", Action::"Edit", Action::"Comment"],
  resource == ?resource
);
```

当主体被指定为资源的编辑者时，您的应用程序可以使用该模板实例化策略，为主体提供对资源执行读取、编辑和评论操作的权限。

## 创建策略模板

AWS Management Console

要创建策略模板，请按以下步骤操作：

1. 打开 Verified Permissions 控制台，网址为 https://console.aws.amazon.com/verifiedpermissions/。选择您的保单商店。
2. 在左侧的导航窗格中，选择策略模板。
3. 选择创建策略模板。
4. 在详细信息部分，输入策略模板描述。
5. 在策略模板正文部分，使用占位符 `?principal` 和 `?resource` 以允许基于此模板创建的策略自定义其授予的权限。您可以选择格式，使用建议的间距和缩进来设置策略模板语法的格式。
6. 选择创建策略模板。

AWS CLI

要创建策略模板，请按以下步骤操作：

您可以使用CreatePolicyTemplate操作创建策略模板。以下示例创建了一个带主体占位符的策略模板。

template1.txt 文件包含以下内容。

```
"VacationAccess"
permit(
    principal in ?principal,
    action == Action::"view",
    resource == Photo::"VacationPhoto94.jpg"
);
```

```
$ aws verifiedpermissions create-policy-template \
    --description "Template for vacation picture access"
    --statement file://template1.txt
    --policy-store-id PSEXAMPLEabcdefg111111
{
    "createdDate": "2023-05-18T21:17:47.284268+00:00",
    "lastUpdatedDate": "2023-05-18T21:17:47.284268+00:00",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "policyTemplateId": "PTEXAMPLEabcdefg111111"
}
```

# 创建模板链接策略

您可以创建模板链接策略以链接到策略模板。模板链接策略与其策略模板保持关联。如果您更改策略模板中的策略声明，则任何链接到该模板的策略将自动使用新声明来做出从那一刻起做出的所有授权决定。

AWS Management Console

要通过实例化策略模板来创建模板链接策略，请按以下步骤操作：

1. 打开 Verified Permissions 控制台，网址为 https://console.aws.amazon.com/verifiedpermissions/。选择您的保单存储。
2. 在左侧的导航窗格中，选择策略。

3. 选择创建策略，然后选择创建模板链接策略。
4. 选中要使用的策略模板旁边的单选按钮，然后选择下一步。
5. 输入要用于此模板链接策略特定实例的主体和资源。指定的值显示在预览策略语句字段中。

> **Note**
>
> 主体和资源值的格式必须与静态策略相同。例如，要为主体指定 AdminUsers 组，请输入 Group::"AdminUsers"。如果您输入 AdminUsers，会显示验证错误。

6. 选择创建模板链接策略。

   新的模板链接策略显示在策略下。

### AWS CLI

要通过实例化策略模板来创建模板链接策略，请按以下步骤操作：

您可以创建一个模板链接策略，该策略引用现有策略模板，并为该模板使用的任何占位符指定值。

下方示例创建了一个模板链接策略，该策略使用包含以下语句的模板：

```
permit(
    principal in ?principal,
    action == Action::"view",
    resource == Photo::"VacationPhoto94.jpg"
);
```

它还使用下方的 definition.txt 文件来提供 definition 参数的值：

```
{
    "templateLinked": {
        "policyTemplateId": "pt-4651be67-c128-4d22-8e67-9b068980c631",
        "principal": {
            "entityType": "User",
            "entityId": "alice"
        }
    }
}
```

输出显示的是从模板中获得的资源和从定义参数中获得的主体

```
$ aws verifiedpermissions create-policy \
    --definition file://definition.txt
    --policy-store-id PSEXAMPLEabcdefg111111
{
    "createdDate": "2023-05-22T18:57:53.298278+00:00",
    "lastUpdatedDate": "2023-05-22T18:57:53.298278+00:00",
    "policyId": "TPEXAMPLEabcdefg111111",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "policyType": "TEMPLATELINKED",
    "principal": {
        "entityId": "alice",
        "entityType": "User"
    },
    "resource": {
        "entityId": "VacationPhoto94.jpg",
        "entityType": "Photo"
    }
}
```

# 编辑策略模板

AWS Management Console

要编辑您的策略模板，请按以下步骤操作：

1. 打开 Verified Permissions 控制台，网址为 https://console.aws.amazon.com/verifiedpermissions/。选择您的保单商店。
2. 在左侧的导航窗格中，选择策略模板。控制台会显示您在当前策略存储中创建的所有策略模板。
3. 选择策略模板旁边的单选按钮，以显示有关该策略模板的详细信息，例如策略模板的创建和更新时间，以及策略模板的内容。
4. 选择编辑，以编辑您的策略模板。根据需要更新策略描述和策略正文，然后选择更新策略模板。
5. 您可以通过选择策略模板旁边的单选按钮，然后选择删除来删除策略模板。选择确定，确认删除策略模板。

AWS CLI

要更新策略模板，请按以下步骤操作：

您可以使用UpdatePolicy操作创建静态策略。以下示例利用文件中定义的新策略替换指定策略模板的策略主体来更新该模板。

template1.txt 文件的内容：

```
permit(
    principal in ?principal,
    action == Action::"view",
    resource in ?resource)
when {
    principal has department && principal.department == "research"
};
```

```
$ aws verifiedpermissions update-policy-template \
    --policy-template-id PTEXAMPLEabcdefg111111 \
    --description "My updated template description" \
    --statement file://template1.txt \
    --policy-store-id PSEXAMPLEabcdefg111111
{
    "createdDate": "2023-05-17T18:58:48.795411+00:00",
    "lastUpdatedDate": "2023-05-17T19:18:48.870209+00:00",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "policyTemplateId": "PTEXAMPLEabcdefg111111"
}
```

# Verified Permissions 示例策略存储的模板链接策略示例

使用示例策略存储方法在 Verified Permissions 中创建策略存储时，您的策略存储是使用预定义的策略、策略模板和所选示例项目的架构创建的。以下 Verified Permissions 模板链接策略示例可用于示例策略存储及其各自的策略、策略模板和架构。

## PhotoFlash与模板关联的策略示例

此示例说明如何创建模板关联策略，该策略使用策略模板授予对与个人用户共享的非私密照片和照片的有限访问权限。

> **Note**
>
> Cedar 策略语言将实体视为 in 自身。因此，principal in User::"Alice" 等同于 principal == User::"Alice"。

```
permit (
 principal in PhotoFlash::User::"Alice",
 action in PhotoFlash::Action::"SharePhotoLimitedAccess",
 resource in PhotoFlash::Photo::"VacationPhoto94.jpg"
 );
```

此示例说明如何创建与模板关联的策略，该策略使用策略模板授予对与个人用户和相册共享的非私密照片的有限访问权限。

```
permit (
 principal in PhotoFlash::User::"Alice",
 action in PhotoFlash::Action::"SharePhotoLimitedAccess",
 resource in PhotoFlash::Album::"Italy2023"
 );
```

此示例说明如何创建模板关联策略，该策略使用策略模板授予对与朋友群组和个人照片的非私密共享照片的有限访问权限。

```
permit (
 principal in PhotoFlash::FriendGroup::"Jane::MySchoolFriends",
 action in PhotoFlash::Action::"SharePhotoLimitedAccess",
 resource in PhotoFlash::Photo::"VacationPhoto94.jpg"
 );
```

此示例说明如何创建模板关联策略，该策略使用策略模板授予对与好友群组和相册共享的非私密照片的有限访问权限。

```
permit (
 principal in PhotoFlash::FriendGroup::"Jane::MySchoolFriends",
 action in PhotoFlash::Action::"SharePhotoLimitedAccess",
 resource in PhotoFlash::Album::"Italy2023"
 );
```

此示例说明如何创建模板关联策略，该策略使用策略模板授予对与朋友群组和个人照片的非私人共享照片的完全访问权限。

```
permit (
 principal in PhotoFlash::UserGroup::"Jane::MySchoolFriends",
 action in PhotoFlash::Action::"SharePhotoFullAccess",
 resource in PhotoFlash::Photo::"VacationPhoto94.jpg"
 );
```

此示例说明了如何使用策略模板来创建模板关联策略，该策略模板禁止用户访问账户。

```
forbid(
 principal == PhotoFlash::User::"Bob",
 action,
 resource in PhotoFlash::Account::"Alice-account"
 );
```

## DigitalPetStore

DigitalPetStore 示例策略存储库不包含任何策略模板。创建DigitalPetStore示例策略存储后，您可以通过在左侧导航窗格中选择策略来查看策略存储中包含的策略。

## TinyToDo 与模板关联的策略示例

此示例说明了如何创建使用提供个人用户和任务列表查看者访问权限的策略模板的模板链接策略。

```
permit (
    principal == TinyTodo::User::"https://cognito-idp.us-east-1.amazonaws.com/us-
east-1_h2aKCU1ts|5ae0c4b1-6de8-4dff-b52e-158188686f31|bob",
    action in [TinyTodo::Action::"ReadList", TinyTodo::Action::"ListTasks"],
    resource == TinyTodo::List::"1"
);
```

此示例说明了如何创建使用提供个人用户和任务列表编辑者访问权限的策略模板的模板链接策略。

```
permit (
    principal == TinyTodo::User::"https://cognito-idp.us-east-1.amazonaws.com/us-
east-1_h2aKCU1ts|5ae0c4b1-6de8-4dff-b52e-158188686f31|bob",
    action in [
        TinyTodo::Action::"ReadList",
```

```
        TinyTodo::Action::"UpdateList",
        TinyTodo::Action::"ListTasks",
        TinyTodo::Action::"CreateTask",
        TinyTodo::Action::"UpdateTask",
        TinyTodo::Action::"DeleteTask"
    ],
    resource == TinyTodo::List::"1"
);
```

# 结合使用 Amazon Verified Permissions 与身份提供者

在 Amazon 已验证的权限中，身份源代表外部身份提供商 (IdP)。身份源提供的信息来自使用与您的策略存储有信任关系的 IdP 进行身份验证的用户。当您的应用程序使用来自身份源的令牌发出授权请求时，您的策略存储可以根据用户属性和访问权限做出授权决策。Verified Permissions 身份源可直接连接到您的中央身份存储和身份验证服务，从而改善授权。

你可以使用具有已验证权限的 OpenID Connect (OIDC) 身份提供商 (IdPs)。您的应用程序可以使用 OIDC 身份 (ID) 生成授权请求或访问 JSON 网络令牌 (JWT)。使用 ID 令牌，已验证权限将用户 ID 和属性声明作为基于属性的访问控制 (ABAC) 的主体读取。使用访问令牌，已验证权限将用户 ID 作为委托人读取，将其他声明作为上下文读取。使用这两种令牌类型，您可以`groups`将类似的声明映射到委托人组，并构建评估基于角色的访问控制 (RBAC) 的策略。

您可以添加 Amazon Cognito 用户池或自定义 OpenID Connect (OIDC) IdP 作为身份来源。

主题



# 使用 Amazon Cognito 身份来源

经过验证的权限与 Amazon Cognito 用户池密切配合。Amazon Cognito JWT 的结构是可预测的。经过验证的权限可以识别这种结构，并从其中包含的信息中获得最大收益。例如，您可以使用 ID 令牌或访问令牌实现基于角色的访问控制 (RBAC) 授权模型。

新的 Amazon Cognito 用户池身份源需要以下信息：

- 的 AWS 区域。
- 用户池 ID。
- 例如，您要与身份源关联的用户实体类型`MyCorp::User`。

- 例如，您要与身份源关联的群组实体类型`MyCorp::UserGroup`。
- （可选）您想要授权向策略存储库发出请求的用户池中的客户端 ID。

由于已验证权限仅适用于相同的 Amazon Cognito 用户池 AWS 账户，因此您无法在其他账户中指定身份来源。例如，Verified Permissions 会将实体前缀（您在作用于用户池主体的策略中必须引用的身份源标识符）设置为用户池的 ID。`us-west-2_EXAMPLE`

用户池令牌声明可以包含属性、范围、群组、客户端 ID 和自定义数据。Amazon Cognito JWT 能够在已验证的权限中包含可能有助于做出授权决策的各种信息。其中包括：

1. 带有`cognito:`前缀的用户名和群组声明
2. 使用自定义用户属性 `custom: prefix`
3. 在运行时添加了自定义声明
4. OIDC 的标准声明，比如和 `sub email`

我们将在中的已验证权限政策中详细介绍这些声明以及如何管理这些声明在架构和策略中使用身份源。

> **Important**
>
> 尽管您可以在 Amazon Cognito 令牌到期之前将其撤销，但 JWT 被视为具有签名和有效性的独立无状态资源。符合 JSON Web Token RFC 7519 的服务需要远程验证令牌，无需向发布者进行验证。这意味着，Verified Permissions 可以根据已被撤销的令牌或向后来被删除的用户颁发的令牌来授予访问权限。为了降低这种风险，我们建议您创建有效期尽可能较短的令牌，并在想要删除授权以终止用户会话时撤消刷新令牌。

针对已验证权限中的用户池身份源的 Cedar 策略对包含字母数字和下划线 (_) 以外的字符的声明名称使用特殊语法。这包括包含字`:`符（如`cognito:username`和）的用户池前缀声明`custom:department`。要编写引用`cognito:username`或`custom:department`声明的保单条件，请分别将其`principal["cognito:username"]`写成和。`principal["custom:department"]`

> **Note**
>
> 如果令牌包含带`cognito:`或`custom:`前缀的声明和带有字面值`cognito`或的声明名称`custom`，则带有的授权请求IsAuthorizedWithToken将失败，并显示为`ValidationException`。

此示例说明如何创建引用与委托人关联的某些 Amazon Cognito 用户池索赔的策略。

```
permit(
     principal == ExampleCo::User::"us-east-1_example|4fe90f4a-ref8d9-4033-
a750-4c8622d62fb6",
     action,
     resource == ExampleCo::Photo::"VacationPhoto94.jpg"
)
when {
     principal["cognito:username"]) == "alice" &&
     principal["custom:department"]) == "Finance"
};
```

有关映射声明的更多信息，请参阅将 ID 令牌映射到架构。有关亚马逊 Cognito 用户授权的更多信息，请参阅 Amazon Cognit o 开发者指南中的使用亚马逊验证权限进行授权。

# 使用 OIDC 身份来源

您也可以将任何合规的 OpenID Connect (OIDC) IdP 配置为策略存储的身份源。OIDC 提供商与 Amazon Cognito 用户池类似：他们生成 JWT 作为身份验证的产物。要添加 OIDC 提供商，您必须提供颁发机构 URL

新的 OIDC 身份源需要以下信息：

- 发行人网址。经过验证的权限必须能够在此 URL 上发现`.well-known/openid-configuration`终端节点。
- 您要在授权请求中使用的令牌类型。在本例中，您选择了身份令牌。
- 例如，您要与身份源关联的用户实体类型`MyCorp::User`。
- 例如，您要与身份源关联的群组实体类型`MyCorp::UserGroup`。
- ID 令牌示例，或 ID 令牌中声明的定义。

- 要应用于用户和群组实体 ID 的前缀。在 CLI 和 API 中，您可以选择此前缀。例如`MyCorp::User::"auth.example.com|a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"`，在您使用 “使用 API Gateway 进行设置” 和 “身份源” 或 “引导式设置” 选项创建的策略存储中，Verified Permissions 会分配颁发者名称减去`https://`的前缀。

使用 OIDC 身份源进行授权使用与用户池身份源相同的 API 操作：IsAuthorizedWithToken和BatchIsAuthorizedWith令牌。

此示例说明如何创建允许会计部门员工访问年终报告的策略，该策略具有机密分类且不在卫星办公室的员工。已验证权限从委托人 ID 令牌中的声明中派生这些属性。

```
permit(
     principal in MyCorp::UserGroup::"MyOIDCProvider|Accounting",
     action,
     resource in MyCorp::Folder::"YearEnd2024"
) when {
     principal.jobClassification == "Confidential" &&
     !(principal.location like "SatelliteOffice*")
};
```

# 客户和受众验证

向策略存储中添加身份源时，Verified Permissions 具有用于验证 ID 和访问令牌是否按预期使用的配置选项。这种验证发生在 `BatchIsAuthorizedWithToken` API 请求`IsAuthorizedWithToken`的处理过程中。身份令牌和访问令牌以及 Amazon Cognito 和 OIDC 身份源的行为有所不同。通过 Amazon Cognito 用户池提供商，经过验证的权限可以验证身份和访问令牌中的客户端 ID。通过 OIDC 提供商，经过验证的权限可以验证 ID 令牌中的客户端 ID 和访问令牌中的受众。

例如，客户端 ID 是与通过提供商配置的 OAuth 或 OIDC 应用程序关联的标识符。`1example23456789`例如，受众是与目标应用程序的预期依赖方或目标相关联的 URL 路径`https://myapplication.example.com`。这种`aud`说法并不总是与观众有关。

已验证权限按如下方式执行身份来源受众和客户端验证：

Amazon Cognito

亚马逊 Cognito ID 令牌的`aud`声明包含应用程序客户端 ID。访问令牌的`client_id`声明也包含应用程序客户端 ID。

当您在身份源中为客户端应用程序验证输入一个或多个值时，Verified Permissions 会将此应用程序客户端 ID 列表与 ID 令牌aud声明或访问令牌client_id声明进行比较。已验证权限不会验证 Amazon Cognito 身份源的中继方受众网址。

OIDC

OIDC ID 令牌的aud声明包含客户端 ID 列表。访问令牌的aud声明包含该令牌的受众网址。访问令牌也有包含预期客户端 ID 的client_id声明。

您可以为 OIDC 提供商的受众验证输入一个或多个值。当您选择令牌类型的 ID 令牌时，Verified Permissions 会验证客户端 ID，检查aud声明中的客户端 ID 中是否至少有一个成员与受众验证值匹配。

Verified Permissions 通过检查aud声明是否与受众验证值相匹配来验证受众的访问令牌。此访问令牌值主要来自aud声明，但如果不存在声明，则可能来自cid或client_idaud声明。请咨询您的 IdP，了解正确的受众群体声明和格式。

ID 令牌受众验证值示例如下1example23456789。

访问令牌受众验证值的示例是https://myapplication.example.com。

# JWT 的客户端授权

您可能需要在应用程序中处理 JSON Web 令牌并将其声明传递给已验证权限，而无需使用策略存储标识源。您可以从 JSON Web 令牌 (JWT) 中提取实体属性并将其解析为已验证的权限。

此示例说明了如何从 OIDC IDP 调用已验证权限。[1]

```
async function authorizeUsingJwtToken(jwtToken) {

    const payload = await verifier.verify(jwtToken);

    var principalEntity = {
        entityType: "PhotoFlash::User", // the application needs to fill in the
 relevant user type
        entityId: payload["sub"], // the application need to use the claim that
 represents the user-id
    };
    var resourceEntity = {
        entityType: "PhotoFlash::Photo", //the application needs to fill in the
 relevant resource type
```

```
        entityId: "jane_photo_123.jpg", // the application needs to fill in the
 relevant resource id
    };
    var action = {
        actionType: "PhotoFlash::Action", //the application needs to fill in the
 relevant action id
        actionId: "GetPhoto", //the application needs to fill in the relevant action
 type
    };
    var entities = {
        entityList: [],
    };
    entities.entityList.push(...getUserEntitiesFromToken(payload));
    var policyStoreId = "PSEXAMPLEabcdefg111111"; // set your own policy store id

    const authResult = await client
        .isAuthorized({
        policyStoreId: policyStoreId,
        principal: principalEntity,
        resource: resourceEntity,
        action: action,
        entities,
        })
        .promise();

    return authResult;

}

function getUserEntitiesFromToken(payload) {
  let attributes = {};
  let claimsNotPassedInEntities = ['aud', 'sub', 'exp', 'jti', 'iss'];
  Object.entries(payload).forEach(([key, value]) => {
    if (claimsNotPassedInEntities.includes(key)) {
        return;
    }
    if (Array.isArray(value)) {
      var attibuteItem = [];
      value.forEach((item) => {
        attibuteItem.push({
          string: item,
        });
      });
      attributes[key] = {
```

```
        set: attibuteItem,
      };
    } else if (typeof value === 'string') {
      attributes[key] = {
        string: value,
      }
    } else if (typeof value === 'bigint' || typeof value ==='number') {
        attributes[key] = {
            long: value,
          }
    } else if (typeof value === 'bigint' || typeof value ==='number') {
        attributes[key] = {
            long: value,
          }
    } else if (typeof value === 'boolean') {
        attributes[key] = {
            boolean: value,
       }
    }

  });

  let entityItem = {
    attributes: attributes,
    identifier: {
      entityType: "PhotoFlash::User",
      entityId: payload["sub"], // the application need to use the claim that
 represents the user-id
    }
  };
  return [entityItem];
}
```

[1] 此代码示例使用 aws-j wt-verify 库来验证由 OIDC 兼容签名的 JWT。 IdPs

# 创建 Amazon Verified Permissions 身份来源

以下过程将身份源添加到现有策略存储中。添加身份源后，必须向架构中添加属性。

在已验证权限控制台中创建新的策略存储时，您也可以创建身份源。在此过程中，您可以自动将身份源令牌中的声明导入实体属性中。选择 “引导式设置” 或 “使用 API Gateway 和身份提供商进行设置” 选项。这些选项还会创建初始策略。

> **Note**
>
> 在您创建策略存储之前，左侧的导航窗格中不会显示身份来源。您创建的身份来源与当前的策略存储相关联。

在已验证权限 API 中使用 create-identity -source AWS CLI 或 CreateIdentity Source 创建身份源时，可以省略委托人实体类型。但是，空白实体类型会创建实体类型为的身份源`AWS::Cognito`。此实体名称与策略存储架构不兼容。要将 Amazon Cognito 身份与您的策略存储架构集成，您必须将委托人实体类型设置为支持的策略存储实体。

主题



## 亚马逊 Cognito 身份来源

AWS Management Console

要创建 Amazon Cognito 用户群体身份来源，请按以下步骤操作：

1. 打开 Verified Permissions 控制台，网址为 https://console.aws.amazon.com/verifiedpermissions/。选择您的保单商店。
2. 在左侧的导航窗格中，选择身份来源。
3. 选择创建身份来源。
4. 在 Cognito 用户池详细信息中，选择 AWS 区域 并输入您的身份源的用户池 ID。
5. 在主体配置中，为身份源选择主体类型。连接的 Amazon Cognito 用户群体中的身份将映射到所选的主体类型。
6. 如果要映射用户池`cognito:groups`声明，请在 “群组配置” 中选择 “使用 Cognito 群组”。选择作为主体类型的父项的实体类型。
7. 在客户端应用程序验证中，选择是否验证客户端应用程序 ID。
   - 如要验证客户端应用程序 ID，请选择仅接受具有匹配客户端应用程序 ID 的令牌。为每个要验证的客户端应用程序 ID 选择添加新客户端应用程序 ID。要删除已添加的客户端应用程序 ID，请选择该客户端应用程序 ID 旁边的删除。
   - 如果您不想验证客户端应用程序 ID，请选择不要验证客户端应用程序 ID。

8. 选择创建身份来源。
9. 在引用从 Cedar 策略中的身份或访问令牌中提取的属性之前，必须更新架构，让 Cedar 知道您的身份来源创建的主体类型。架构中的新增内容必须包含您要在 Cedar 策略中引用的属性。有关将 Amazon Cognito 令牌属性映射到 Cedar 主体属性的更多信息，请参阅[在架构和策略中使用身份源]。

   当您创建与 API 关联的策略存储时，Verified Permissions 会查询您的用户池以获取用户属性，并创建一个架构，其中您的委托人类型将填充用户池属性。

AWS CLI

要创建 Amazon Cognito 用户群体身份来源，请按以下步骤操作：

您可以使用 Source 操作创建身份CreateIdentity源。以下示例创建了一个身份来源，可以从 Amazon Cognito 用户群体访问经过身份验证的身份。

以下 `config.txt` 文件包含 Amazon Cognito 用户群体的详细信息，供 `create-identity-source` 命令中的 --configuration 参数使用。

```
{
    "cognitoUserPoolConfiguration": {
        "userPoolArn": "arn:aws:cognito-idp:us-west-2:123456789012:userpool/us-
west-2_1a2b3c4d5",
        "clientIds":["a1b2c3d4e5f6g7h8i9j0kalbmc"],
        "groupConfiguration": {
              "groupEntityType": "MyCorp::UserGroup"
        }
    }
}
```

命令:

```
$ aws verifiedpermissions create-identity-source \
    --configuration file://config.txt \
    --principal-entity-type "User" \
    --policy-store-id 123456789012
{
    "createdDate": "2023-05-19T20:30:28.214829+00:00",
    "identitySourceId": "ISEXAMPLEabcdefg111111",
    "lastUpdatedDate": "2023-05-19T20:30:28.214829+00:00",
    "policyStoreId": "PSEXAMPLEabcdefg111111"
```

```
}
```

在引用从 Cedar 策略中的身份或访问令牌中提取的属性之前，必须更新架构，让 Cedar 知道您的身份来源创建的主体类型。架构中的新增内容必须包含您要在 Cedar 策略中引用的属性。有关将 Amazon Cognito 令牌属性映射到 Cedar 主体属性的更多信息，请参阅在架构和策略中使用身份源。

当您创建与 API 关联的策略存储时，Verified Permissions 会查询您的用户池以获取用户属性，并创建一个架构，其中您的委托人类型将填充用户池属性。

有关在 Verified Permissions 中为经过身份验证的用户使用 Amazon Cognito 访问令牌和身份令牌的更多信息，请参阅《Amazon Cognito 开发人员指南》中的使用 Amazon Verified Permissions 进行授权。

## OIDC 身份来源

AWS Management Console

创建 OpenID Connect (OIDC) 身份源

1. 打开 Verified Permissions 控制台，网址为 https://console.aws.amazon.com/verifiedpermissions/。选择您的保单商店。
2. 在左侧的导航窗格中，选择身份来源。
3. 选择创建身份来源。
4. 选择外部 OIDC 提供商。
5. 在发卡机构 URL 中，输入您的 OIDC 发行人的 URL。例如，这是提供授权服务器、签名密钥以及有关您的提供商的其他信息的服务端点`https://auth.example.com`。您的发卡机构 URL 必须托管 OIDC 发现文档，网址为。`/.well-known/openid-configuration`
6. 在令牌类型中，选择您希望您的应用程序提交以进行授权的 OIDC JWT 类型。有关更多信息，请参阅 在架构和策略中使用身份源。
7. 在用户和群组声明中，为身份源选择用户实体和用户声明。用户实体是您的策略存储中的一个实体，您想要引用来自 OIDC 提供商的用户。用户声明通常`sub`是来自您的身份证或访问令牌的索赔，该令牌包含待评估实体的唯一标识符。来自连接的 OIDC IdP 的身份将映射到选定的主体类型。
8. 在 “用户和群组声明” 中，为身份来源选择群组实体和群组声明。组实体是用户实体的父实体。团体索赔将映射到该实体。群组声明通常是来自您的 ID 或访问令牌的声明`groups`，其中包含

要评估的实体的字符串、JSON 或以空格分隔的用户组名称字符串。来自连接的 OIDC IdP 的身份将映射到选定的主体类型。

9. 在受众验证中，输入您希望您的政策商店在授权请求中接受的客户端 ID 或受众网址（如果有）。
10. 选择创建身份来源。
11. 更新您的架构，让 Cedar 知道您的身份源创建的主体类型。架构中的新增内容必须包含您要在 Cedar 策略中引用的属性。有关将 Amazon Cognito 令牌属性映射到 Cedar 主体属性的更多信息，请参阅在架构和策略中使用身份源。

    当您创建与 API 关联的策略存储时，Verified Permissions 会查询您的用户池以获取用户属性，并创建一个架构，其中您的委托人类型将填充用户池属性。

AWS CLI

创建 OIDC 身份源

您可以使用 Source 操作创建身份CreateIdentity源。以下示例创建了一个身份来源，可以从 Amazon Cognito 用户群体访问经过身份验证的身份。

以下`config.txt`文件包含供命令`--configuration`参数使用的 OIDC IdP 的详细信息。`create-identity-source`此示例为 ID 令牌创建 OIDC 身份源。

```
{
    "openIdConnectConfiguration": {
        "issuer": "https://auth.example.com",
        "tokenSelection": {
                "identityTokenOnly": {
                        "clientIds":["1example23456789"],
                        "principalIdClaim": "sub"
                },
        },
        "entityIdPrefix": "MyOIDCProvider",
        "groupConfiguration": {
              "groupClaim": "groups",
              "groupEntityType": "MyCorp::UserGroup"
        }
    }
}
```

以下config.txt文件包含供命令--configuration参数使用的 OIDC IdP 的详细信息。create-identity-source此示例为访问令牌创建 OIDC 身份源。

```
{
    "openIdConnectConfiguration": {
        "issuer": "https://auth.example.com",
        "tokenSelection": {
                "accessTokenOnly": {
                        "audiences":["https://auth.example.com"],
                        "principalIdClaim": "sub"
                },
        },
        "entityIdPrefix": "MyOIDCProvider",
        "groupConfiguration": {
              "groupClaim": "groups",
              "groupEntityType": "MyCorp::UserGroup"
        }
    }
}
```

命令:

```
$ aws verifiedpermissions create-identity-source \
    --configuration file://config.txt \
    --principal-entity-type "User" \
    --policy-store-id 123456789012
{
    "createdDate": "2023-05-19T20:30:28.214829+00:00",
    "identitySourceId": "ISEXAMPLEabcdefg111111",
    "lastUpdatedDate": "2023-05-19T20:30:28.214829+00:00",
    "policyStoreId": "PSEXAMPLEabcdefg111111"
}
```

在引用从 Cedar 策略中的身份或访问令牌中提取的属性之前，必须更新架构，让 Cedar 知道您的身份来源创建的主体类型。架构中的新增内容必须包含您要在 Cedar 策略中引用的属性。有关将 Amazon Cognito 令牌属性映射到 Cedar 主体属性的更多信息，请参阅在架构和策略中使用身份源。

当您创建与 API 关联的策略存储时，Verified Permissions 会查询您的用户池以获取用户属性，并创建一个架构，其中您的委托人类型将填充用户池属性。

# 编辑 Amazon Verified Permissions 身份来源

创建身份源后，您可以编辑身份源的某些参数。如果您的策略存储架构与您的身份源属性相匹配，则请注意，您必须单独更新架构以反映您对身份源所做的更改。

主题



## Amazon Cognito 用户池身份来源

AWS Management Console

要更新 Amazon Cognito 用户群体身份来源，请按以下步骤操作：

1. 打开 Verified Permissions 控制台，网址为 https://console.aws.amazon.com/verifiedpermissions/。选择您的保单商店。
2. 在左侧的导航窗格中，选择身份来源。
3. 选择要编辑的身份来源的 ID。
4. 选择编辑。
5. 在 Cognito 用户池详细信息中，选择 AWS 区域 并键入您的身份源的用户池 ID。
6. 在委托人详细信息中，您可以更新身份源的委托人类型。连接的 Amazon Cognito 用户群体中的身份将映射到所选的主体类型。
7. 如果要映射用户池`cognito:groups`声明，请在 “群组配置” 中选择 “使用 Cognito 群组”。选择作为主体类型的父项的实体类型。
8. 在客户端应用程序验证中，选择是否验证客户端应用程序 ID。
   - 如要验证客户端应用程序 ID，请选择仅接受具有匹配客户端应用程序 ID 的令牌。为每个要验证的客户端应用程序 ID 选择添加新客户端应用程序 ID。要删除已添加的客户端应用程序 ID，请选择该客户端应用程序 ID 旁边的删除。
   - 如果您不想验证客户端应用程序 ID，请选择不要验证客户端应用程序 ID。
9. 选择保存更改。
10. 如果您更改了该身份来源的主体类型，则必须更新架构，以正确反映更新后的主体类型。

如要删除身份来源，您可以选择身份来源旁边的单选按钮，然后选择删除身份来源。在文本框中输入 delete，然后选择删除身份来源，确认删除该身份来源。

AWS CLI

要更新 Amazon Cognito 用户群体身份来源，请按以下步骤操作：

您可以使用 Source 操作更新身份UpdateIdentity源。以下示例更新了指定的身份来源，以使用不同的 Amazon Cognito 用户群体。

以下 config.txt 文件包含 Amazon Cognito 用户群体的详细信息，供 create-identity-source 命令中的 --configuration 参数使用。

```
{
    "cognitoUserPoolConfiguration": {
        "userPoolArn": "arn:aws:cognito-idp:us-west-2:123456789012:userpool/us-
west-2_1a2b3c4d5",
        "clientIds":["a1b2c3d4e5f6g7h8i9j0kalbmc"],
        "groupConfiguration": {
              "groupEntityType": "MyCorp::UserGroup"
        }
    }
}
```

命令:

```
$ aws verifiedpermissions update-identity-source \
    --update-configuration file://config.txt \
    --policy-store-id 123456789012
{
    "createdDate": "2023-05-19T20:30:28.214829+00:00",
    "identitySourceId": "ISEXAMPLEabcdefg111111",
    "lastUpdatedDate": "2023-05-19T20:30:28.214829+00:00",
    "policyStoreId": "PSEXAMPLEabcdefg111111"
}
```

如果要更改该身份来源的主体类型，必须更新架构，以正确反映更新后的主体类型。

# OpenID Connect (OIDC) 身份来源

AWS Management Console

更新 OIDC 身份源

1. 打开 Verified Permissions 控制台，网址为 https://console.aws.amazon.com/verifiedpermissions/。选择您的保单商店。
2. 在左侧的导航窗格中，选择身份来源。
3. 选择要编辑的身份来源的 ID。
4. 选择编辑。
5. 在 OIDC 提供商详细信息中，根据需要更改发行者 URL。
6. 在将令牌声明映射到架构属性中，根据需要更改用户和组声明与策略存储实体类型之间的关联。更改实体类型后，必须更新策略和架构属性以应用于新的实体类型。
7. 在受众验证中，添加或删除要强制执行的受众群体值。
8. 选择保存更改。

如要删除身份来源，您可以选择身份来源旁边的单选按钮，然后选择删除身份来源。在文本框中输入 `delete`，然后选择删除身份来源，确认删除该身份来源。

AWS CLI

更新 OIDC 身份源

您可以使用 Source 操作更新身份UpdateIdentity源。以下示例将指定的身份源更新为使用其他 OIDC 提供商。

以下 `config.txt` 文件包含 Amazon Cognito 用户群体的详细信息，供 `create-identity-source` 命令中的 --configuration 参数使用。

```
{
    "openIdConnectConfiguration": {
        "issuer": "https://auth2.example.com",
        "tokenSelection": {
                "identityTokenOnly": {
                        "clientIds":["2example10111213"],
                        "principalIdClaim": "sub"
                },
```

```
        },
        "entityIdPrefix": "MyOIDCProvider",
        "groupConfiguration": {
              "groupClaim": "groups",
              "groupEntityType": "MyCorp::UserGroup"
        }
    }
}
```

命令:

```
$ aws verifiedpermissions update-identity-source \
    --update-configuration file://config.txt \
    --policy-store-id 123456789012
{
    "createdDate": "2023-05-19T20:30:28.214829+00:00",
    "identitySourceId": "ISEXAMPLEabcdefg111111",
    "lastUpdatedDate": "2023-05-19T20:30:28.214829+00:00",
    "policyStoreId": "PSEXAMPLEabcdefg111111"
}
```

如果要更改该身份来源的主体类型，必须更新架构，以正确反映更新后的主体类型。

# 在架构和策略中使用身份源

您可能会发现要将身份源添加到策略存储中，并将提供商声明映射到您的策略存储架构中。您可以自动执行此过程或手动更新架构。用户指南的这一部分包含以下信息：

- 何时可以自动填充策略存储架构的属性
- 如何在您的已验证权限政策中使用 Amazon Cognito 和 OIDC 令牌声明
- 如何为身份源手动构建架构

通过引导式设置与 API 关联的策略存储和带有身份源的策略存储不需要将身份 (ID) 令牌属性手动映射到架构。您可以为已验证权限提供用户池或 OIDC 令牌中的属性，并创建填充用户属性的架构。在 ID 令牌授权中，已验证权限将声明映射到委托人实体的属性。在以下情况下，您可能需要手动将 Amazon Cognito 令牌映射到您的架构：

- 您根据示例创建了空白的策略存储区或策略存储区。
- 您希望将访问令牌的使用范围扩展到基于角色的访问控制 (RBAC) 之外。

- 您可以使用已验证的权限 REST API、 AWS 软件开发工具包或创建策略存储 AWS CDK。

要使用 Amazon Cognito 或 OIDC 身份提供商 (IdP) 作为已验证权限策略存储中的身份源，您的架构中必须包含提供商属性。如果您创建策略存储的方式是自动从 ID 令牌中的提供者信息填充架构，那么您就可以编写策略了。如果您创建的策略存储没有身份源架构，则必须向架构中添加提供者属性。您的架构必须与提供者令牌创建的实体IsAuthorizedWithToken或令BatchIsAuthorizedWith牌 API 请求相对应。然后，您可以使用提供者令牌中的属性来编写策略。

有关在已验证权限中为经过身份验证的用户使用 Amazon Cognito ID 和访问令牌的更多信息，请参阅 Amazon Co gnito 开发者指南中的使用亚马逊验证权限进行授权。

主题

- 关于架构映射的注意事项
- 将 ID 令牌映射到架构
- 映射访问令牌
- Amazon Cognito 以冒号分隔的声明的替代符号

## 关于架构映射的注意事项

不同标记类型的属性映射不同

在访问令牌授权中，已验证权限将声明映射到上下文。在 ID 令牌授权中，已验证权限将声明映射到委托人属性。对于您在已验证权限控制台中创建的策略存储，只有空策略存储和示例策略存储才会使您没有身份来源，并要求您在架构中填充用户池属性以进行 ID 令牌授权。访问令牌授权基于基于角色的访问控制 (RBAC) 和群组成员资格声明，不会自动将其他声明映射到策略存储架构。

身份源属性不是必需的

在已验证权限控制台中创建身份源时，不会将任何属性标记为必填属性。这样可以防止丢失的索赔导致授权请求中的验证错误。您可以根据需要将属性设置为 required，但它们必须存在于所有授权请求中。

RBAC 不需要架构中的属性

身份源的架构取决于您在添加身份源时所建立的实体关联。身份源将一个声明映射到用户实体类型，将一个声明映射到群组实体类型。这些实体映射是身份源配置的核心。有了这些最少的信息，您就可以在基于角色的访问控制 (RBAC) 模型中编写对特定用户和用户可能属于的特定组执行授权操作的策略。向架构中添加令牌声明可扩展策略存储的授权范围。来自 ID 令牌的用户属性包含有关用户的信息，这些

信息可以为基于属性的访问控制 (ABAC) 授权做出贡献。访问令牌中的上下文属性包含诸如 OAuth 2.0 范围之类的信息，这些信息可以提供来自提供商的额外访问控制信息，但需要进行额外的架构修改。

已验证权限控制台中的 “使用 API Gateway 和身份源进行设置” 和 “引导式设置” 选项为架构分配 ID 令牌声明。访问令牌声明的情况并非如此。要向架构中添加非组访问令牌声明，必须在 JSON 模式下编辑架构并添加 CommonTypes 属性。有关更多信息，请参阅 映射访问令牌。

OIDC 团体声称支持多种格式

添加 OIDC 提供商时，您可以在 ID 或访问令牌中选择要映射到策略存储中用户的群组成员资格的群组名称。经过验证的权限可以识别以下格式的群组声明：

1. 不带空格的字符串：`"groups": "MyGroup"`
2. 以空格分隔的列表:. `"groups": "MyGroup1 MyGroup2 MyGroup3"` 每个字符串都是一个组。
3. JSON（以逗号分隔）列表：`"groups": ["MyGroup1", "MyGroup2", "MyGroup3"]`

> **Note**
>
> Verified Permissions 将以空格分隔的群组声明中的每个字符串解释为一个单独的群组。要将带有空格字符的组名解释为单个组，请替换或删除声明中的空格。例如，将名为的群组设置为My Group的格式MyGroup。

选择代币类型

您的策略存储与身份源配合的方式取决于身份源配置中的一个关键决定：是处理 ID 还是访问令牌。使用 Amazon Cognito 身份提供商，您可以在创建与 API 关联的策略存储时选择令牌类型。创建与 API 关联的策略存储时，必须选择是要为 ID 还是访问令牌设置授权。此信息会影响已验证权限应用于您的策略存储的架构属性，以及您的 API Gateway API 的 Lambda 授权方的语法。对于 OIDC 提供商，您必须在添加身份源时选择令牌类型。您可以选择 ID 或访问令牌，并且您的选择会将未选择的令牌类型排除在保单存储库中的处理范围之外。特别是如果您希望从身份令牌声明自动映射到已验证权限控制台中的属性中受益，请在创建身份源之前尽早决定要处理的令牌类型。更改令牌类型需要花费大量精力来重构您的策略和架构。以下主题介绍如何在策略存储中使用 ID 和访问令牌。

Cedar 解析器要求某些字符使用方括号

策略通常以类似的格式引用架构属性`principal.username`。对于令牌声明名称中可能出现的大多数非字母数字字符`:`，例如`.`、或`/`，Verified Permissions 无法解析像或这样的条件值。`principal.cognito:groups` `context.ip-address`您必须改为使

用principal["cognito:username"]或context["ip-address"]格式的方括号表示法来格式化这些条件。下划线字符_是声明名称中的有效字符，也是该要求的唯一非字母数字例外。

此类型主属性的部分示例架构如下所示：

```
"User": {
   "shape": {
      "type": "Record",
      "attributes": {
         "cognito:username": {
            "type": "String",
            "required": true
         },
         "custom:employmentStoreCode": {
            "type": "String",
            "required": true,
         },
         "email": {
            "type": "String",
            "required": false
         }
      }
   }
}
```

这种类型的上下文属性的部分示例架构如下所示：

```
"GetOrder": {
   "memberOf": [],
   "appliesTo": {
      "resourceTypes": [
         "Order"
      ],
      "context": {
         "type": "Record",
         "attributes": {
            "ip-address": {
               "required": false,
               "type": "String"
            }
   }
   },
      "principalTypes": [
```

```
            "User"
        ]
    }
}
```

将针对此架构进行验证的属性的示例策略如下所示：

```
permit (
    principal in MyCorp::UserGroup::"us-west-2_EXAMPLE|MyUserGroup",
    action,
    resource
) when {
    principal["cognito:username"] == "alice" &&
    principal["custom:employmentStoreCode"] == "petstore-dallas" &&
    principal has email && principal.email == "alice@example.com" &&
    context["ip-address"] like "192.0.2.*"
};
```

## 将 ID 令牌映射到架构

Verified Permissions 将身份令牌声明作为用户的属性进行处理：他们的姓名和头衔、群组成员资格、联系信息。ID 令牌在基于属性的访问控制 (ABAC) 授权模型中最有用。如果您想让经过验证的权限根据谁提出请求来分析对资源的访问权限，请为您的身份来源选择 ID 令牌。

### 亚马逊 Cognito ID 令牌

Amazon Cognito ID 令牌适用于大多数 OIDC 依赖方库。它们通过其他索赔扩展了OIDC的功能。您的应用程序可以通过 Amazon Cognito 用户池身份验证 API 操作或用户池托管用户界面对用户进行身份验证。有关更多信息，请参阅 Amazon Cognito 开发者指南中的使用 API 和终端节点。

亚马逊 Cognito ID 代币中的有用声明

*cognito:username* 和 *preferred_username*

用户名的变体。

*sub*

用户的唯一用户标识符 (UUID)

带*custom:*前缀的索赔

自定义用户池属性的前缀，例如custom:employmentStoreCode。

标准索赔

标准 OIDC 声称类似email和。phone_number有关更多信息，请参阅 OpenID Connect Core 1.0 中包含勘误集 2 的标准声明。

*cognito:groups*

用户的群组成员资格。在基于角色的访问控制 (RBAC) 的授权模型中，此声明提供了您可以在策略中评估的角色。

临时索赔

声明不是用户的财产，但在运行时由用户池生成令牌前 Lambda 触发器添加。临时索赔与标准索赔类似，但不在标准范围内，例如tenant或department。

在引用带有:分隔符的 Amazon Cognito 属性的策略中，引用格式中的属性。principal["*cognito:username*"]角色声明cognito:groups是此规则的例外。已验证权限将此声明的内容映射到用户实体的父实体。

有关来自 Amazon Cognito 用户池的 ID 令牌结构的更多信息，请参阅 Amazon Cog nito 开发者指南中的使用 ID 令牌。

以下示例 ID 令牌具有四种类型的属性。分别是 Amazon Cognito 特定声明 cognito:username、自定义声明 custom:employmentStoreCode、标准声明 email 和临时声明 tenant。

```
{
    "sub": "91eb4550-XXX",
    "cognito:groups": [
        "Store-Owner-Role",
        "Customer"
    ],
    "email_verified": true,
    "clearance": "confidential",
    "iss": "https://cognito-idp.us-east-2.amazonaws.com/us-east-2_EXAMPLE",
    "cognito:username": "alice",
    "custom:employmentStoreCode": "petstore-dallas",
    "origin_jti": "5b9f50a3-05da-454a-8b99-b79c2349de77",
    "aud": "1example23456789",
    "event_id": "0ed5ad5c-7182-4ecf-XXX",
    "token_use": "id",
    "auth_time": 1687885407,
    "department": "engineering",
```

```
    "exp": 1687889006,
    "iat": 1687885407,
    "tenant": "x11app-tenant-1",
    "jti": "a1b2c3d4-e5f6-a1b2-c3d4-TOKEN1111111",
    "email": "alice@example.com"
}
```

当您使用 Amazon Cognito 用户池创建身份源时，您需要指定授权请求中验证权限生成的委托人实体的类型。`IsAuthorizedWithToken`然后，作为评估该请求的一部分，您的策略会测试该主体的属性。您的架构定义身份源的委托人类型和属性，然后您可以在 Cedar 策略中引用它们。

您还可以指定要从 ID 令牌组声明中派生的群组实体的类型。在授权请求中，Verified Permissions 会将每个群组成员的声明映射到该群组实体类型。在策略中，您可以将该组实体引用为委托人。

以下示例说明了如何在 Verified Permissions 架构中反映示例身份令牌中的属性。有关编辑架构的更多信息，请参阅在 JSON 模式下编辑架构。如果您的身份来源配置指定了主体类型 `User`，那么，您可以添加类似于以下示例的内容，以使这些属性可用于 Cedar。

```
"User": {
   "shape": {
      "type": "Record",
      "attributes": {
         "cognito:username": {
            "type": "String",
            "required": false
         },
         "custom:employmentStoreCode": {
            "type": "String",
            "required": false
         },
         "email": {
            "type": "String"
         },
         "tenant": {
            "type": "String",
            "required": true
         }
      }
   }
}
```

在更新架构以反映 Amazon Cognito 属性后，您可以创建引用这些属性的策略。

```
permit (
    principal in MyCorp::UserGroup::"us-west-2_EXAMPLE|MyUserGroup",
    action,
    resource
) when {
    principal["cognito:username"] == "alice" &&
    principal["custom:employmentStoreCode"] == "petstore-dallas" &&
    principal.tenant == "x11app-tenant-1" &&
    principal has email && principal.email == "alice@example.com"
};
```

## OIDC 身份令牌

使用 OIDC 提供商提供的身份令牌与使用亚马逊 Cognito ID 令牌大致相同。区别在于索赔。您的 IdP 可能呈现标准的 OIDC 属性，或者具有自定义架构。在已验证权限控制台中创建新的策略存储时，可以添加带有示例 ID 令牌的 OIDC 身份源，也可以手动将令牌声明映射到用户属性。由于已验证权限不知道您的 IdP 的属性架构，因此您必须提供此信息。

有关更多信息，请参阅 创建 Verified Permissions 策略存储。

以下是具有 OIDC 身份源的策略存储的示例架构。

```
"User": {
   "shape": {
      "type": "Record",
      "attributes": {
         "email": {
            "type": "String"
         },
         "email_verified": {
            "type": "Boolean"
         },
         "name": {
            "type": "String",
            "required": true
         },
         "phone_number": {
            "type": "String"
         },
         "phone_number_verified": {
            "type": "Boolean"
         }
```

```
      }
   }
}
```

以下政策适用于您的 OIDC 提供商中群组的成员。

```
permit (
    principal in MyCorp::UserGroup::"MyOIDCProvider|MyUserGroup",
    action,
    resource
) when {
    principal.email_verified == true && principal.email == "alice@example.com" &&
    principal.phone_number_verified == true && principal.phone_number like "+1206*"
};
```

## 映射访问令牌

Verified Permissions 处理访问令牌声明，而不是作为操作属性或上下文属性的群组声明。除了群组成员资格外，来自您的 IdP 的访问令牌可能还包含有关 API 访问的信息。访问令牌在使用基于角色的访问控制 (RBAC) 的授权模型中很有用。依赖组成员资格以外的访问令牌声明的授权模型需要在架构配置方面付出额外的努力。

### 映射 Amazon Cognito 访问令牌

Amazon Cognito 访问令牌具有可用于授权的声明：

亚马逊 Cognito 访问令牌中的有用声明

*client_id*

OIDC 依赖方的客户端应用程序的 ID。使用客户端 ID，已验证权限可以验证授权请求是否来自策略存储的允许客户端。在 machine-to-machine （M2M）授权中，请求系统使用客户端密钥对请求进行授权，并提供客户端 ID 和范围作为授权证据。

*scope*

代表令牌持有者的访问权限的 OAuth 2.0 范围。

*cognito:groups*

用户的群组成员资格。在基于角色的访问控制 (RBAC) 的授权模型中，此声明提供了您可以在策略中评估的角色。

临时索赔

不是访问权限但是在运行时由用户池生成令牌前 Lambda 触发器添加的声明。临时索赔与标准索赔类似，但不在标准范围内，例如tenant或department。自定义访问令牌会增加您的 AWS 账单成本。

有关来自 Amazon Cognito 用户池的访问令牌结构的更多信息，请参阅 Amazon Cog nito 开发者指南中的使用访问令牌。

Amazon Cognito 访问令牌在传递到 Verified Permissions 时，会映射到上下文对象。可以使用 `context.token.attribute_name` 来引用访问令牌的属性。以下示例访问令牌同时包含 `client_id` 和 `scope` 声明。

```
{
    "sub": "91eb4550-9091-708c-a7a6-9758ef8b6b1e",
    "cognito:groups": [
        "Store-Owner-Role",
        "Customer"
    ],
    "iss": "https://cognito-idp.us-east-2.amazonaws.com/us-east-2_EXAMPLE",
    "client_id": "1example23456789",
    "origin_jti": "a1b2c3d4-e5f6-a1b2-c3d4-TOKEN1111111",
    "event_id": "bda909cb-3e29-4bb8-83e3-ce6808f49011",
    "token_use": "access",
    "scope": "MyAPI/mydata.write",
    "auth_time": 1688092966,
    "exp": 1688096566,
    "iat": 1688092966,
    "jti": "a1b2c3d4-e5f6-a1b2-c3d4-TOKEN2222222",
    "username": "alice"
}
```

以下示例说明了如何在 Verified Permissions 架构中反映示例访问令牌中的属性。有关编辑架构的更多信息，请参阅在 JSON 模式下编辑架构。

```
{
   "MyApplication": {
      "actions": {
         "Read": {
            "appliesTo": {
               "context": {
```

```
                  "type": "ReusedContext"
               },
               "resourceTypes": [
                  "Application"
               ],
               "principalTypes": [
                  "User"
               ]
            }
         }
      },
      ...
      ...
      "commonTypes": {
         "ReusedContext": {
            "attributes": {
               "token": {
                  "type": "Record",
                  "attributes": {
                     "scope": {
                        "type": "Set",
                        "element": {
                           "type": "String"
                        }
                     },
                     "client_id": {
                        "type": "String"
                     }
                  }
               }
            },
            "type": "Record"
         }
      }
   }
}
```

在更新架构以反映 Amazon Cognito 属性后，您可以创建引用这些属性的策略。

```
permit(principal, action in [MyApplication::Action::"Read",
 MyApplication::Action::"GetStoreInventory"], resource)
when {
    context.token.client_id == "52n97d5afhfiu1c4di1k5m8f60" &&
```

```
    context.token.scope.contains("MyAPI/mydata.write")
};
```

## 映射 OIDC 访问令牌

来自外部 OIDC 提供商的大多数访问令牌都与 Amazon Cognito 访问令牌非常一致。传递给 “已验证权限” 时，OIDC 访问令牌会映射到上下文对象。可以使用 `context.token.`*`attribute_name`* 来引用访问令牌的属性。以下示例 OIDC 访问令牌包括基本声明示例。

```
{
    "sub": "91eb4550-9091-708c-a7a6-9758ef8b6b1e",
    "groups": [
        "Store-Owner-Role",
        "Customer"
    ],
    "iss": "https://auth.example.com",
    "client_id": "1example23456789",
    "aud": "https://myapplication.example.com"
    "scope": "MyAPI-Read",
    "exp": 1688096566,
    "iat": 1688092966,
    "jti": "a1b2c3d4-e5f6-a1b2-c3d4-TOKEN2222222",
    "username": "alice"
}
```

以下示例说明了如何在 Verified Permissions 架构中反映示例访问令牌中的属性。有关编辑架构的更多信息，请参阅在 JSON 模式下编辑架构。

```
{
   "MyApplication": {
      "actions": {
         "Read": {
            "appliesTo": {
               "context": {
                  "type": "ReusedContext"
               },
               "resourceTypes": [
                  "Application"
               ],
               "principalTypes": [
                  "User"
               ]
```

```
            }
         }
      },
      ...
      ...
      "commonTypes": {
         "ReusedContext": {
            "attributes": {
               "token": {
                  "type": "Record",
                  "attributes": {
                     "scope": {
                        "type": "Set",
                        "element": {
                           "type": "String"
                        }
                     },
                     "client_id": {
                        "type": "String"
                     }
                  }
               }
            },
            "type": "Record"
         }
      }
   }
}
```

更新架构以反映 IdP 属性后，您可以创建引用这些属性的策略。

```
permit(
    principal,
    action in [MyApplication::Action::"Read",
 MyApplication::Action::"GetStoreInventory"],
    resource
)
when {
    context.token.client_id == "52n97d5afhfiu1c4di1k5m8f60" &&
    context.token.scope.contains("MyAPI-read")
};
```

# Amazon Cognito 以冒号分隔的声明的替代符号

在启动已验证权限时，Amazon Cognito 令牌的推荐架构声称喜欢这些冒号分隔的字符串，`cognito:groups`并将其`custom:store`转换为使用该字符作为层次结构分隔`.`符。这种格式称为点符号。例如，在您的政策`principal.cognito.groups`中提及`cognito:groups`变为了。尽管您可以继续使用这种格式，但我们建议您使用括号表示法来构建架构和策略。在这种格式中，对的引用将`cognito:groups`变`principal["cognito:groups"]`为您的政策。从已验证权限控制台自动生成的用户池 ID 令牌架构使用方括号表示法。

您可以继续在手动构建的 Amazon Cognito 身份源的架构和策略中使用点符号。对于任何其他类型的 OIDC IdP，您不能在架构`:`或策略中使用带点符号或任何其他非字母数字字符。

点符号架构将`:`字符的每个实例嵌套为`cognito`或`custom`初始短语的子项，如以下示例所示：

```
"CognitoUser": {
   "shape": {
      "type": "Record",
      "attributes": {
         "cognito": {
            "type": "Record",
            "required": true,
            "attributes": {
               "username": {
                  "type": "String",
                  "required": true
               }
            }
         },
         "custom": {
            "type": "Record",
            "required": true,
            "attributes": {
               "employmentStoreCode": {
                  "type": "String",
                  "required": true
               }
            }
         },
         "email": {
            "type": "String"
         },
         "tenant": {
```

```
            "type": "String",
            "required": true
         }
      }
   }
}
```

使用这种格式的架构，您可以创建带有点符号的策略，如以下示例所示：

```
permit(principal, action, resource)
when {
    principal.cognito.username == "alice" &&
    principal.custom.employmentStoreCode == "petstore-dallas" &&
    principal.tenant == "x11app-tenant-1" &&
    principal has email && principal.email == "alice@example.com"
};
```

# 为您的应用程序设计授权模型

当您准备在软件应用程序中使用 Amazon Verified Permissions 服务时，立即编写策略语句可能会很困难。这就类似于在完全确定应用程序应该实现什么功能之前，通过编写 SQL 语句或 API 规范来开始开发应用程序的其他部分。相反，您应该从用户体验开始，清楚地了解在应用程序界面中管理权限时，最终用户应该看到哪些内容。然后，从用户经验开始倒推，找到一种实现方法。

在做这项工作时，您会发现自己存在几个疑问，比如：

- 我的资源有哪些？ 它们彼此之间有关系吗？ 例如，文件是否位于一个文件夹中？
- 主体可以对每种资源执行哪些操作？
- 主体如何获得这些权限？
- 您是希望最终用户从 “管理员”、“操作员” 或 “ReadOnly” 等预定义权限中进行选择，还是应该创建临时策略声明？ 或者二者结合？
- 权限是否应该跨资源继承，例如文件是否应该从父文件夹继承权限？
- 为了呈现用户体验，需要哪些类型的查询？ 例如，为了呈现用户的主页，是否需要列出主体可以访问的所有资源？
- 用户有没有可能会不小心将自己排除在自己的资源之外？ 是否需要避免这种情况发生？

此练习的最终结果被称为授权模型；它定义了主体、资源、操作以及它们之间的相互关系。生成此模型不需要对 Cedar 或 Verified Permissions 服务有独特的了解。它首先是一项用户体验设计练习，就像其他任何练习一样，可以体现在界面模型、逻辑图等构件中，以及对权限如何影响用户在产品中看到的内容的总体描述中。Cedar 的设计具有足够的灵活性，可以满足客户的模型需求，而不会为了符合 Cedar 的实现要求而强迫模型做出不自然的弯曲。因此，对期望的用户体验有清晰的了解是获得最佳模型的最佳方式。

这一部分提供了与如何进行设计练习、需要注意的事项以及成功使用 Verified Permissions 的最佳实践有关的一般指导。

除了此处提供的指导之外，请记得参考《Cedar 策略语言参考指南》中的最佳实践。

主题

- 多租户注意事项
- 尽可能填充策略范围
- 每个资源都位于容器中
- 将主体与资源容器分离
- 请勿在属性中嵌入权限
- 在模型中更倾向精细权限，在用户界面中更倾向聚合权限
- 考虑查询授权的其他原因

## 没有规范的“正确”模型

在设计授权模型时，没有一个唯一的正确答案。不同的应用程序可以对相似的概念有效地使用不同的授权模型，这是可以接受的。例如，思考一下计算机文件系统的表示形式。在类似 Unix 的操作系统中创建文件时，它不会自动继承父文件夹的权限。相比之下，在许多其他操作系统和大多数在线文件共享服务中，文件确实会继承其父文件夹的权限。这两个选项都有效，具体取决于应用程序进行优化的情况。

授权解决方案的正确性并非绝对，取决于它如何提供客户期望的体验，以及它是否能按照客户期望的方式保护他们的资源。如果您的授权模型能够做到这些，那么它就是一个成功的模型。

这就是为什么说，从所期望的用户体验开始设计是对创建有效授权模型最有帮助的先决条件。

## 将注意力集中在 API 操作之外的资源上

在大多数面向消费者的应用程序中，权限都是围绕应用程序支持的资源建模的。例如，一个文件共享应用程序可能会将权限表示为可以对文件或文件夹执行的操作。这是一个良好的简单模型，它将底层实现和后端 API 操作抽象了出来。

相比之下，其他类型的应用程序，尤其是 Web 服务，通常会围绕 API 操作本身设计权限。例如，如果 Web 服务提供了一个名为 `createThing()` 的 API，则授权模型可能会定义相应的权限，或者在 Cedar 中定义名为 `createThing` 的 `action`。这适用于许多情况，并且使理解权限变得很简单。要调用 `createThing` 操作，您需要 `createThing` 操作权限。看起来很简单，对吧？

您会发现，“已验证权限” 控制台中的入门流程包括直接通过 API 构建资源和操作的选项。这是一个有用的基准：您的策略存储库与其授权的 API 之间的直接映射。

但是，这种以 API 为中心的方法可能不是最优方法，因为 API 只是客户真正想要保护的东西（底层数据和资源）的代名词。如果有多个 API 控制对相同资源的访问权限，则管理员可能很难理清这些资源的路径并相应地管理访问权限。

例如，假设有一个包含组织成员的用户目录。可以将用户划分为组，其中一个安全目标是禁止未经授权的各方查看组成员资格。管理此用户目录的服务提供了两个 API 操作：

- `listMembersOfGroup`
- `listGroupMembershipsForUser`

客户可以使用其中任何一个操作来查看群组成员资格。因此，权限管理员必须记得协调对这两个操作的访问权限。如果您随后选择添加新的 API 操作来解决其他使用案例，情况会变得更加复杂，比如下面的情况：

- `isUserInGroups`（一个新 API，用于快速测试用户是否属于一个或多个组）

从安全角度来看，此 API 为查看组成员资格开辟了第三条途径，这就破坏了管理员精心设计的权限。

我们建议您忽略 API 语义，而将注意力集中在底层数据和资源及其关联操作上。将这种方法应用于组成员资格示例将获得抽象权限，例如 `viewGroupMembership`，三个 API 操作都必须查询该权限。

| API 名称 | 权限 |
| --- | --- |
| `listMembersOfGroup` | 需要具有针对该组的 `viewGroupMembership` 权限 |
| `listGroupMembershipsForUser` | 需要具有针对该用户的 `viewGroupMembership` 权限 |
| `isUserInGroups` | 需要具有针对该用户的 `viewGroupMembership` 权限 |

通过定义这一权限，管理员可以在现在和未来成功地控制查看组成员资格的权限。代价是，现在，每个 API 操作必须记录它可能需要的几种权限，并且管理员在制定权限时必须查阅此文档。如果是为满足安全要求，这可能是一个有效的折衷方案。

## 复合授权是正常情况

当单个用户活动（例如，单击应用程序界面的某个按钮）需要多个单独的授权查询来确定该活动是否被允许时，就会发生复合授权。例如，将文件移动到文件系统中的新目录可能需要三种不同的权限：能够从源目录中删除文件，能够将文件添加到目标目录，可能还需要对文件本身进行修改（取决于应用程序）。

如果您不熟悉如何设计授权模型，您可能会认为每个授权决策都必须在单个授权查询中进行解析。但这可能会导致模型过于复杂，策略语句极度混乱。实际上，使用复合授权可以帮助您生成更简单的授权模型。衡量授权模型的设计是否精良的一个标准是，当您充分分解了单个操作时，您的复合操作（例如，移动文件）可以通过直观的基元聚合来表示。

当授予权限的过程涉及到多个参与方时，也会发生复合授权。考虑有这样一个组织目录，用户可以是组的成员。一种简单的方法是向组所有者授予添加任何人的权限。但是，如果您希望您的用户首先同意被添加，那该怎么办？ 这就引入了握手协议，其中，用户和组都必须同意这一成员资格。为此，您可以引入另一种与该用户绑定的权限，并指定是否可以将该用户添加到任何组或特定组。当调用方随后尝试向组中添加成员时，该应用程序必须执行两方面的权限：调用方有权向指定组添加成员，并且被添加的单个用户具有被添加的权限。当存在 N 次握手时，通常会观察到 N 个复合授权查询来执行协议的各个部分。

如果您发现自己面临涉及多个资源的设计挑战，并且不清楚如何对权限进行建模，这可能表明您面临的是复合授权场景。在这种情况下，通过将操作分解为多个单独的授权检查，您可能会找到解决方案。

# 多租户注意事项

您可能需要开发供多个客户（使用您的应用程序的企业或租户）使用的应用程序，并将它们与 Amazon Verified Permissions 集成。在开发授权模型之前，请制定多租户策略。您可以在一个共享策略存储区中管理客户的策略，也可以为每个租户分配一个策略存储。

1. 一个共享策略存储库

   所有租户共享一个保单存储库。应用程序将所有授权请求发送到共享策略存储区。

2. 每租户策略存储

   每个租户都有专门的保单存储。应用程序将查询不同的策略存储以获得授权决定，具体取决于提出请求的租户。

这两种策略都会产生相对较高的授权请求量，这可能会对您的账单产生影响。AWS那么，你应该如何设计自己的方法呢？ 以下是可能影响您的已验证权限多租户授权策略的常见条件。

租户策略隔离

将每个租户的策略与其他租户的策略隔离开来对于保护租户数据非常重要。当每个租户都有自己的策略存储库时，他们每个人都有自己的一组独立的策略。

授权流程

您可以在请求中使用策略存储库 ID 和每个租户的策略存储来识别发出授权请求的租户。对于共享策略存储，所有请求都使用相同的策略存储 ID。

模板和架构管理

您的策略模板和策略存储架构会在每个策略存储中增加一定程度的设计和维护开销。

全球政策管理

您可能需要将一些全局策略应用于每个租户。管理全局策略的开销水平因共享策略存储模式和按租户策略存储模式而异。

租户离职

一些租户会为您的架构和策略提供特定于其案例的元素。当租户不再在您的组织中处于活动状态，而您想要删除他们的数据时，工作量会因他们与其他租户的隔离程度而异。

服务资源配额

已验证权限的资源和请求速率配额可能会影响您的多租户决策。有关配额的更多信息，请参阅资源配额。

## 比较共享策略存储库和每租户策略存储库

在共享和按租户策略商店模型中，每种考虑因素都需要自己的时间和资源投入水平。

| 考虑因素 | 共享策略存储库中的工作级别 | 每租户策略存储库中的工作量级别 |
| --- | --- | --- |
| 租户策略隔离 | 中等。Must include tenant identifiers in policies and authorization requests. | 低。 Isolation is default behavior. Tenant-specific policies are inaccessible to other tenants. |
| 授权流程 | 低。 All queries target one policy store. | 中等。 Must maintain mappings between each tenant and their policy store ID. |
| 模板和架构管理 | 低。 Must make one schema work for all tenants. | 高。 Schemas and templates might be less complex |

| | | |
|---|---|---|
| | | individually, but changes require more coordination and complexity. |
| 全球政策管理 | 低。 All policies are global and can be centrally updated. | 高。 You must add global policies to each policy store in onboarding. Replicate global policy updates between many policy stores. |
| 租户离职 | 中等。 Must identify and delete only tenant-specific policies. | 低。 Delete the policy store. |
| 服务资源配额 | 高。 Tenants share resource quotas that affect policy stores like schema size, policy size per resource, and identity sources per policy store. | 低。 Each tenant has dedicated resource quotas. |

## 如何选择

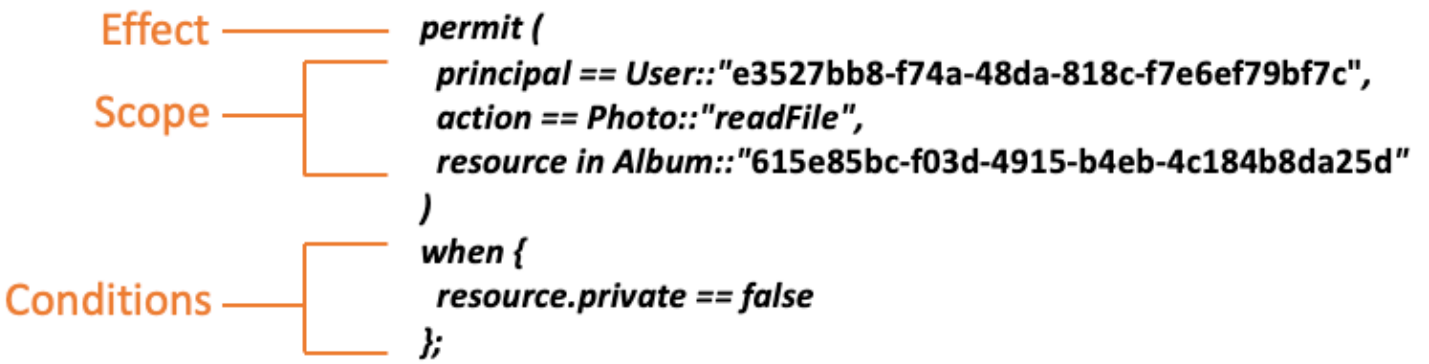
每个多租户应用程序都不一样。在做出架构决策之前，请仔细比较这两种方法及其注意事项。

如果您的应用程序不需要租户特定的策略并且使用单一身份源，那么为所有租户使用一个共享策略存储可能是最有效的解决方案。这样可以简化授权流程和全局策略管理。使用一个共享策略存储区退出租户所需的精力更少，因为应用程序不需要删除租户特定的策略。

但是，如果您的应用程序需要许多租户特定的策略，或者使用多个身份源，则每个租户的策略存储可能是最有效的。您可以使用向每个租户授予每个策略存储的权限的IAM策略来控制对租户策略的访问权限。退出租户涉及删除其策略存储；在 shared-policy-store 环境中，您必须查找并删除租户特定的策略。

# 尽可能填充策略范围

策略范围是 Cedar 策略语句中 `permit` 或 `forbid` 关键字之后和左括号之间的部分。




我们建议您尽可能填充 `principal` 和 `resource` 的值。这样，Verified Permissions 就可以为策略编制索引，提高检索效率，进而提高性能。如果您需要向许多不同的主体或资源授予相同的权限，我们建议您使用策略模板并将其附加到每个主体和资源对。

避免创建一个 `when` 子句中包含主体和资源列表的大型策略。因为，创建一个这样的大型策略可能会导致您遇到可扩展性限制或操作难题。例如，要在策略的大型列表中添加或删除单个用户，必须读取整个策略，编辑列表，完整编写新策略，并在一个管理员覆盖另一个管理员的更改时处理并发错误。相比之下，通过使用许多精细权限，添加或删除用户就像添加或删除适用于他们的单个策略一样简单。

## 每个资源都位于容器中

在设计授权模型时，每个操作都必须与特定资源相关联。通过诸如 `viewFile` 之类的操作，您可以直观地看到可以将其应用到的资源，可能是单个文件，也可能是文件夹中的文件集合。但是，如果是 `createFile` 之类的操作，显示的结果就不会那么直观。在对创建文件的能力进行建模时，它适用于什么资源？ 不可能是文件本身，因为该文件尚不存在。

这是资源创建常见问题的一个示例。资源创建涉及到自我引导。即使尚不存在任何资源，也必须存在一种方式，能够使某个实体获得创建资源的权限。解决方案是认识到每个资源都必须存在于某个容器内，而容器本身充当权限的锚点。例如，如果系统中已经存在一个文件夹，则可以将创建文件的能力建模为对该文件夹的权限，因为在该文件夹中，权限是实例化新资源所必需的。

```
permit (
    principal == User::"6688f676-1aa9-456a-acf4-228340b54e9d",
    action == Action::"createFile",
    resource == Folder::"c863f89b-461f-4fc2-b638-e5fa5f79a48b"
);
```

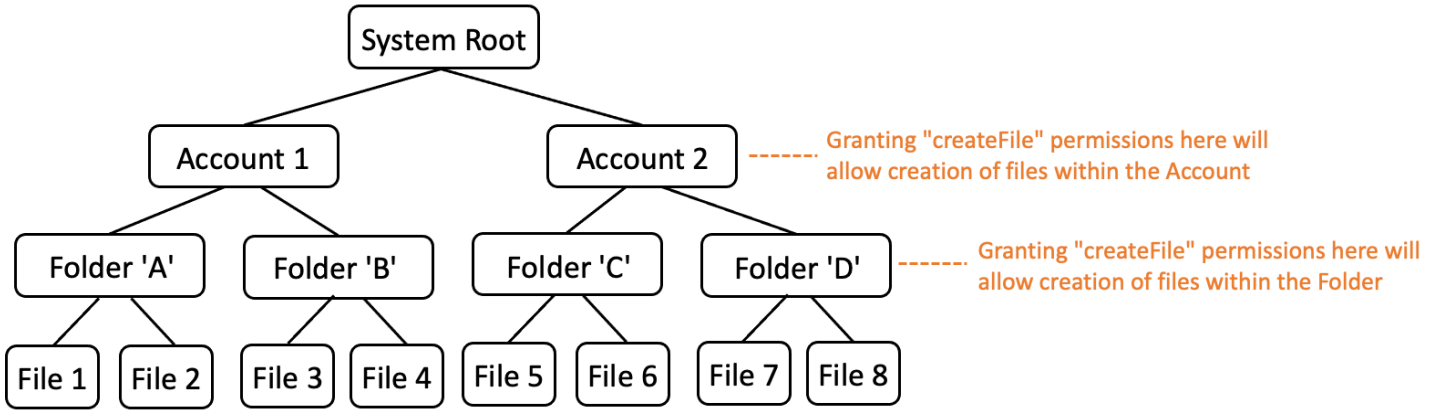
但是，如果不存在文件夹，那该怎么办？ 也许这是应用程序中的一个全新客户账户，尚不存在任何资源。在这种情况下，仍然存在一种上下文，通过一个问题就可以清晰地理解，即“客户可以在哪里创建

新文件？”。您不希望他们能够在任意客户账户中创建文件。这就存在一个隐含的上下文：客户自己的账户边界。因此，账户本身代表资源创建的容器，可以使用类似以下示例的策略对其进行显式建模。

```
// Grants permission to create files within an account,
// or within any sub-folder inside the account.
permit (
    principal == User::"6688f676-1aa9-456a-acf4-228340b54e9d",
    action == Action::"createFile",
    resource in Account::"c863f89b-461f-4fc2-b638-e5fa5f79a48b"
);
```

但是，如果连账户也不存在，那该怎么办？ 您可以选择设计客户注册工作流程，以使其在系统中创建新账户。如果是这样，您将需要一个容器来表示该流程可以创建账户的最外层边界。这个根级容器代表整个系统，名称可能类似于“system root”。但是，是否需要这个容器以及如何命名由您（应用程序所有者）来决定。

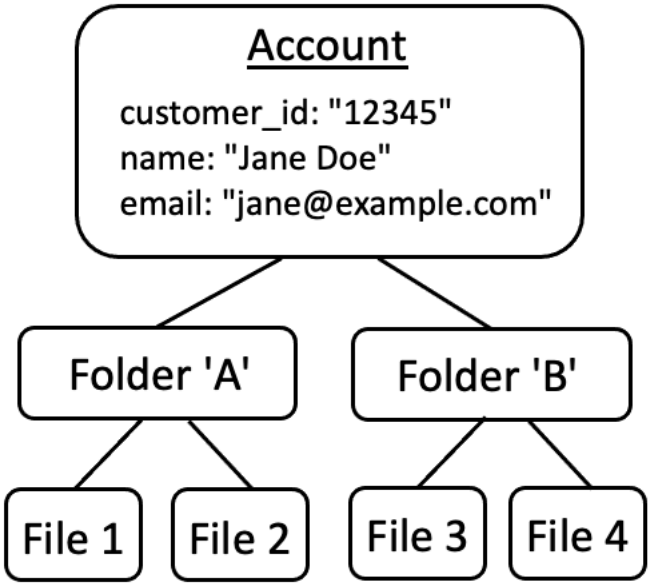
因此，对于此示例应用程序，生成的容器层次结构如下所示：




这是一个示例层次结构，其他层次结构也是有效的。需要记住的是，资源创建总是在资源容器的上下文中进行的。这些容器可能是隐式的，例如账户边界，很容易被忽视。在设计授权模型时，请务必注意这些隐式假设，以便在授权模型中对其进行正式记录和表示。

## 将主体与资源容器分离

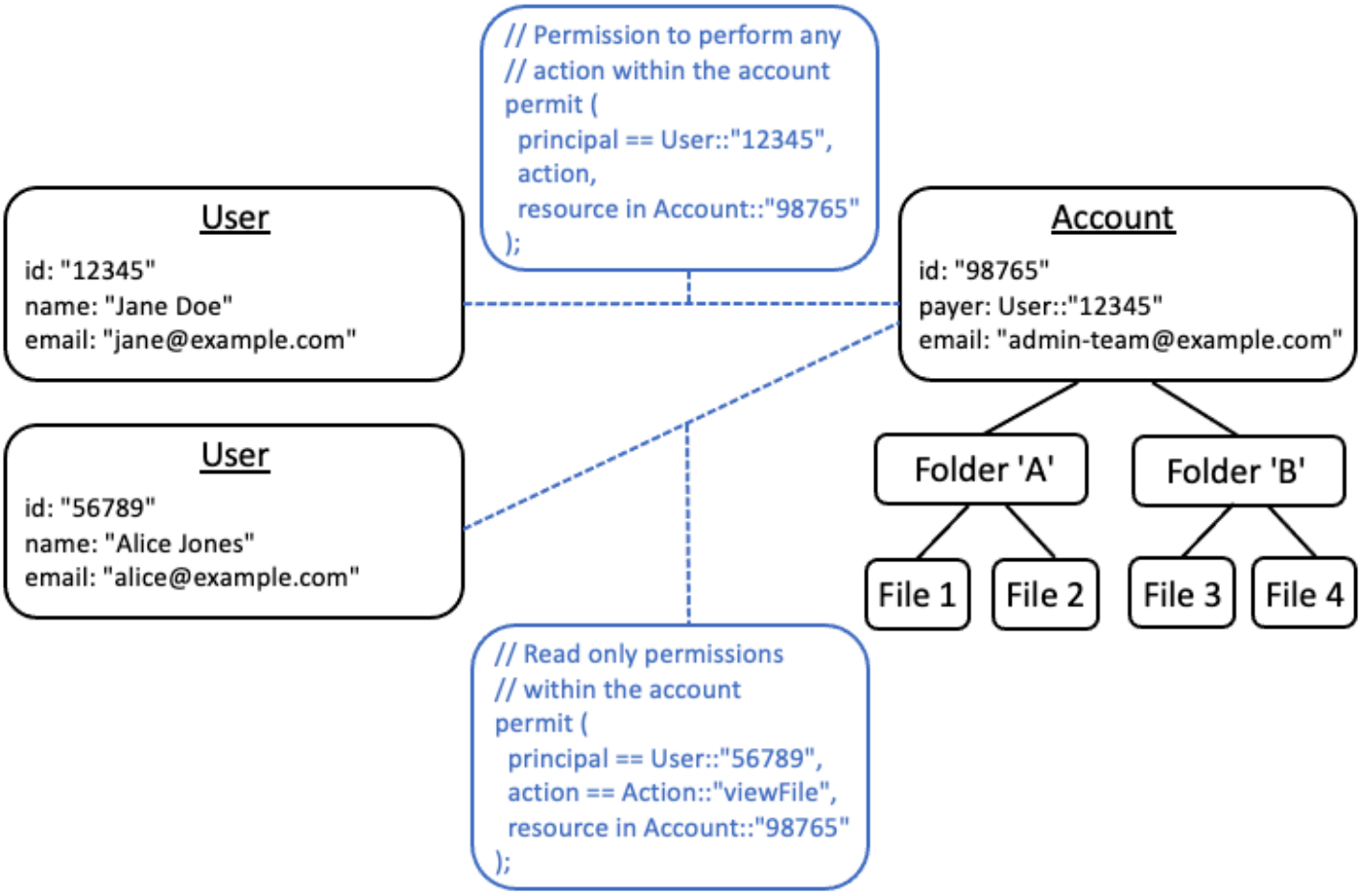
在设计资源层次结构时，一种常见的倾向是使用客户的用户身份作为客户账户中资源的容器，尤其是在面向消费者的应用程序中。



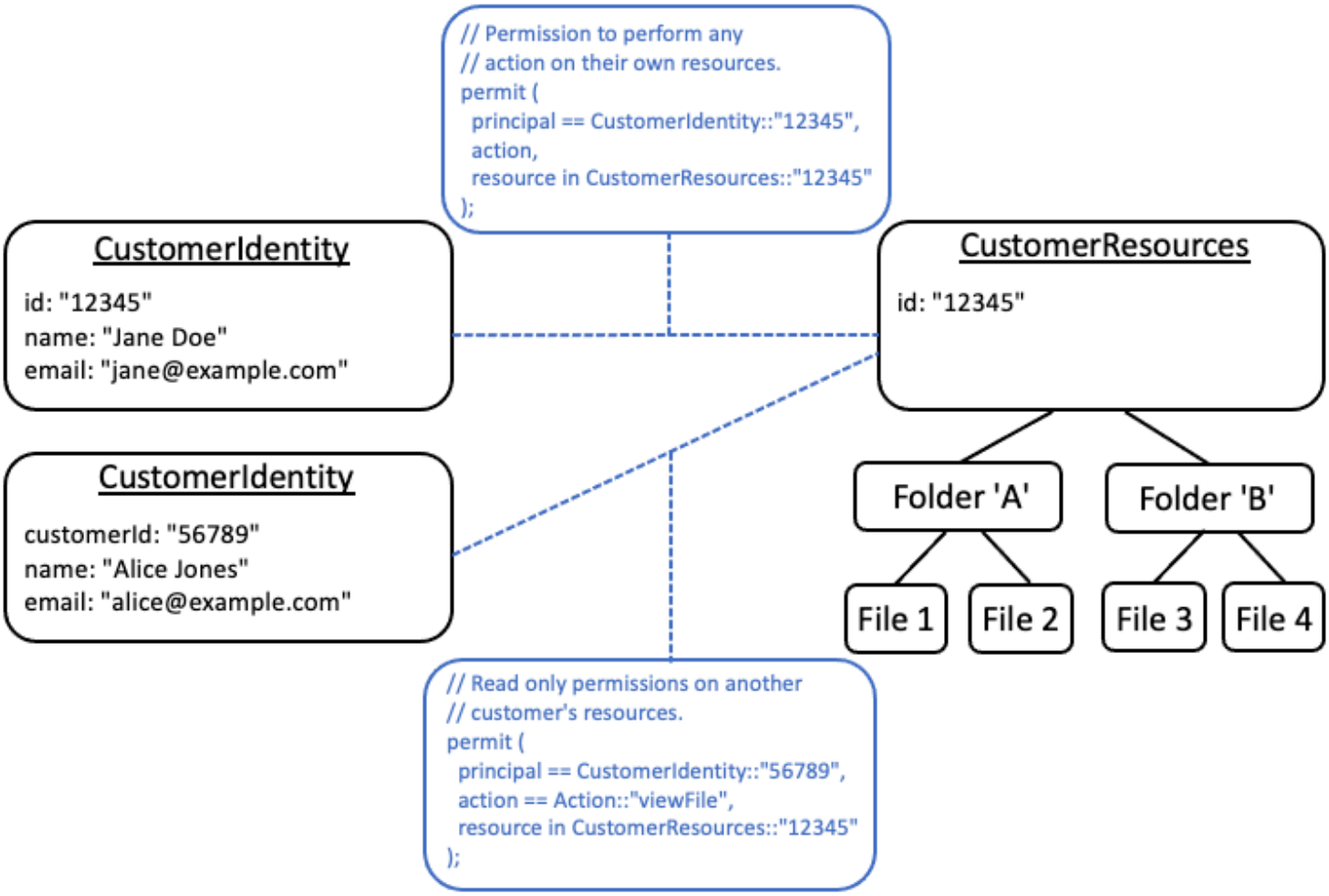

我们建议您将此策略视为一种反模式。这是因为，在更丰富的应用程序中，自然会倾向于将访问权限委托给其他用户。例如，您可以选择引入“家庭”账户，让其他用户可以在其中共享账户资源。同样，企业客户有时会希望将多名员工指定为部分账户的操作员。您可能还需要将一个账户的所有权转让给另一位用户，或者需要合并多个账户的资源。

当使用用户身份作为账户的资源容器时，前面的场景就更难以实现。更令人担忧的是，如果以这种方式授予其他人访问账户容器的权限，他们可能会被意外授予修改用户身份的权限，例如更改 Jane 的电子邮件或登录凭证。

因此，在可能的情况下，更具弹性的方法是将主体与资源容器分离，并使用“管理员权限”或“所有权”之类的概念对二者之间的连接进行建模。




如果您的现有应用程序无法使用这种分离模型，建议您在设计授权模型时，考虑尽可能地模仿这种模型。例如，一个应用程序只拥有一个名为 Customer 的概念，该概念概况了用户身份、登录凭证和他们拥有的资源，可以将其映射到一个授权模型，该模型包含 Customer Identity 的一个逻辑实体（包含姓名、电子邮件等）和 Customer Resources 或 Customer Account 的一个单独逻辑实体，作为它们所拥有的所有资源的父节点。这两个实体可以共享相同的 Id，但具有不同的 Type。




# 请勿在属性中嵌入权限

最好将属性用作授权决策的输入。请勿使用属性来表示权限本身，例如在用户上声明名为“permittedFolders”的属性：

```
// ANTI-PATTERN: comingling permissions into user attributes
{
    "id": "df82e4ad-949e-44cb-8acf-2d1acda71798",
    "name": "alice",
    "email": "alice@example.com",
    "permittedFolders": [
        "Folder::\"c943927f-d803-4f40-9a53-7740272cb969\"",
        "Folder::\"661817a9-d478-4096-943d-4ef1e082d19a\"",
        "Folder::\"b8ee140c-fa09-46c3-992e-099438930894\""
    ]
}
```

并且，随后在策略中使用该属性：

```
// ANTI-PATTERN
permit (
    principal,
    action == Action::"readFile",
    resource
)
when {
    resource in principal.permittedFolders
};
```

这种方法将原本简单的授权模型（即特定的主体可以访问特定文件夹）转变为基于属性的访问权限控制（ABAC）模型，并伴随着一些不利后果。其中一个不利后果是，要快速确定谁有权访问某个资源变得更加困难。在前面的示例中，为了确定谁有权访问某个特定文件夹，必须遍历每个用户，以检查该文件夹是否在其属性中列出，并且要特别注意，这样做时，有一个策略会授予访问权限。

这种方法的另一个风险是，当权限被整合到单个 User 记录中时，可能会出现扩展方面的问题。如果用户可以访问许多内容，则其 User 记录的累计大小将增加，并且可能接近存储数据的任何系统的最大限制。

相反，我们建议您使用多个单独的策略来表示这种情况，可以使用策略模板来最大限度地减少重复次数。

```
//BETTER PATTERN
permit (
    principal == User::"df82e4ad-949e-44cb-8acf-2d1acda71798",
    action == Action::"readFile",
    resource in Folder::"c943927f-d803-4f40-9a53-7740272cb969"
);

permit (
    principal == User::"df82e4ad-949e-44cb-8acf-2d1acda71798",
    action == Action::"readFile",
    resource in Folder::"661817a9-d478-4096-943d-4ef1e082d19a"
);

permit (
    principal == User::"df82e4ad-949e-44cb-8acf-2d1acda71798",
    action == Action::"readFile",
    resource in Folder::"b8ee140c-fa09-46c3-992e-099438930894"
);
```

在授权评估期间，Verified Permissions 可以有效地处理许多单独的精细策略。随着时间的推移，以这种方式对事物进行建模更易于管理和审计。

## 在模型中更倾向精细权限，在用户界面中更倾向聚合权限

设计师未来经常会后悔的一种策略，是设计一个具有非常广泛的操作的授权模型，例如 Read 和 Write，而后意识到需要的是更精细的操作。对更精细粒度的需求可能来自于客户对更精细访问控制的反馈，或者来自于鼓励最低权限许可的合规性和安全性审计员。

如果未预先定义精细权限，那么可能需要进行复杂的转换，才能修改应用程序代码和策略语句，以使用更精细的权限。例如，需要修改先前针对粗粒度操作进行授权的应用程序代码，才能使用精细操作。此外，还需要更新策略，以反映迁移情况：

```
permit (
    principal == User::"6688f676-1aa9-456a-acf4-228340b54e9d",
    // action  ==  Action::"read",              -- coarse-grained permission --
 commented out
    action in [                         //        -- finer grained permissions
        Action::"listFolderContents",
        Action::"viewFile"
    ],
    resource in Account::"c863f89b-461f-4fc2-b638-e5fa5f79a48b"
);
```

为了避免这种代价高昂的迁移，最好预先定义精细权限。尽管预先定义精细权限可以避免代价高昂的迁移，但如果最终用户随后被迫理解更多的精细权限，尤其是在大多数客户对 Read 和 Write 等粗粒度控制感到满意的情况下，这就可能需要作出取舍。为了实现两全其美的效果，您可以使用策略模板或操作组等机制，将精细权限分组到预定义的集合中，例如 Read 和 Write。通过这种方法，客户只能看到粗粒度的权限。但是在后台，通过将粗粒度的权限建模为一系列精细操作，您的应用程序可以适应未来的需求。您可以应客户或审计员的要求，公开精细权限。

## 考虑查询授权的其他原因

我们通常会将授权检查与用户请求相关联。此检查是确定用户是否有权执行该请求的一种方式。不过，您还可以使用授权数据来影响应用程序界面的设计。例如，您可能希望显示一个主屏幕，其中仅显示最终用户可以访问的资源列表。在查看资源的详细信息时，您可能希望该界面仅显示用户可以对该资源执行的操作。

这些情况可能会给授权模型带来一些难题。例如，严重依赖基于属性的访问权限控制（ABAC）策略，会使快速回答“谁有权访问什么？”变得更加困难。这是因为回答该问题需要检查每个规则是否与每个主体和资源相匹配。因此，对于需要进行优化以仅列出用户可访问的资源的产品，可能会选择使用基于角色的访问权限控制（RBAC）模型。通过使用 RBAC，可以更轻松地迭代附加到用户的所有策略，以确定资源访问权限。

# 测试平台

Verified Permissions 测试平台允许您通过对 Verified Permissions 策略运行授权请求来测试和排除故障。该测试平台使用您指定的参数来确定您的策略存储中的 Cedar 策略是否会授权该请求。测试授权请求时，您可以在可视模式和 JSON 模式之间进行切换。有关 Cedar 策略的结构和评估方式的更多信息，请参阅《Cedar 策略语言参考指南》中的 Cedar 中的基本策略构造。

> **Note**
>
> 使用 Verified Permissions 发出授权请求时，您可以在其他实体部分，在请求中提供主体和资源列表。不过，您无法包含有关操作的详细信息。这些信息必须在架构中指定，或者从请求中推断得出。您无法将操作置于其他实体部分。

有关测试台架的直观概述和演示，请观看此视频。

Visual mode

> **Note**
>
> 要使用测试平台的可视模式，您必须在策略存储中定义一个架构。

要在可视模式下测试策略，请按以下步骤操作：

1. 打开 Verified Permissions 控制台，网址为 https://console.aws.amazon.com/verifiedpermissions/。选择您的保单商店。
2. 在左侧导航窗格中，选择测试平台。
3. 选择可视模式。
4. 在主体部分，从架构的主体类型中选择执行操作的主体。在文本框中为该主体输入一个标识符。
5. （可选）选择添加父级，为指定主体添加父实体。要移除已添加到该主体的父实体，请选择父实体名称旁边的删除。
6. 为指定主体的每个属性指定属性值。该测试平台使用模拟授权请求中指定的属性值。
7. 在资源部分中，选择主体正在执行的资源。在文本框中为该资源输入一个标识符。

8. （可选）选择添加父级，为指定资源添加父实体。要删除已添加到该资源的父实体，请选择该父实体名称旁边的删除。
9. 为指定资源的每个属性指定属性值。该测试平台使用模拟授权请求中指定的属性值。
10. 在操作部分中，从指定主体和资源的有效操作列表中选择主体正在执行的操作。
11. 为指定操作的每个属性指定属性值。该测试平台使用模拟授权请求中指定的属性值。
12. （可选）在其他实体部分中，选择添加实体，以添加要在授权决策中评估的实体。
13. 从下拉列表中选择实体标识符并输入该实体标识符。
14. （可选）选择添加父级，为指定实体添加父实体。要删除已添加到该实体的父实体，请选择该父实体名称旁边的删除。
15. 为指定实体的每个属性指定属性值。该测试平台使用模拟授权请求中指定的属性值。
16. 选择确认，将该实体添加到测试平台。
17. 选择运行授权请求，模拟策略存储中 Cedar 策略的授权请求。测试平台会显示允许或拒绝该请求的决策，以及与满足条件的策略或评估期间遇到的错误有关的信息。

### JSON mode

要在 JSON 模式下测试策略，请按以下步骤操作：

1. 打开 Verified Permissions 控制台，网址为 https://console.aws.amazon.com/verifiedpermissions/。选择您的保单商店。
2. 在左侧导航窗格中，选择测试平台。
3. 选择 JSON 模式。
4. 在请求详细信息部分中，如果您定义了一个架构，请从该架构的主体类型中选择执行操作的主体。在文本框中为该主体输入一个标识符。

   如果未定义架构，请在执行操作的主体文本框中输入该主体。
5. 如果您定义了一个架构，请从该架构的资源类型中选择资源。在文本框中为该资源输入一个标识符。

   如果未定义架构，请在资源文本框中输入该资源。
6. 如果您定义了一个架构，请从指定主体和资源的有效操作列表中选择操作。

   如果未定义架构，请在操作文本框中输入该操作。
7. 在上下文字段中，输入要模拟的请求的上下文。该请求上下文是可用于授权决策的附加信息。
8. 在实体字段中，输入授权决策中要评估的实体及其属性的层次结构。

9. 选择运行授权请求，模拟策略存储中 Cedar 策略的授权请求。测试平台会显示允许或拒绝该请求的决策，以及与满足条件的策略或评估期间遇到的错误有关的信息。

# 在 Amazon 验证权限中实现授权

创建策略存储、策略、模板、架构和授权模型后，就可以开始使用 Amazon Verified Permissions 授权请求了。要实现已验证的权限授权，您必须将中的策略配置 AWS 与应用程序中的集成结合起来。要将已验证权限与您的应用程序集成，请添加一个 AWS SDK 并实现调用已验证权限 API 并针对您的策略存储生成授权决策的方法。

使用经过验证的权限进行授权对于应用程序中的 UX 权限和 API 权限非常有用。

用户体验权限

控制用户对您的应用程序 UX 的访问权限。您可以只允许用户查看他们需要访问的确切表单、按钮、图形和其他资源。例如，当用户登录时，您可能需要确定其账户中是否有 “转账” 按钮。您还可以控制用户可以采取的操作。例如，在同一个银行应用程序中，您可能需要确定是否允许您的用户更改交易类别。

API 权限

控制用户对数据的访问权限。应用程序通常是分布式系统的一部分，它们从外部 API 中引入信息。在银行应用程序的示例中，Verified Permissions 允许显示 “转账” 按钮，当您的用户发起转账时，必须做出更复杂的授权决定。Verified Permissions 可以授权列出符合条件的转账目标账户的 API 请求，然后授权将转账推送到其他账户的请求。

说明此内容的示例来自策略存储库示例。要继续操作，请在您的测试环境中DigitalPet创建 Store 示例政策存储。

有关使用批量授权实现 UX 权限的端到端示例应用程序，请参阅AWS 安全博客上的 “使用 Amazon 已验证的权限进行大规模细粒度授权”。

## 用于授权的 API 操作

已验证权限 API 具有以下授权操作。

### IsAuthorized

`IsAuthorized`API 操作是使用已验证权限的授权请求的入口点。您必须提交承担者、操作、资源、上下文和实体元素。已验证权限会根据您的策略存储架构验证您请求中的实体。然后，Verified Permissions 会根据请求的策略存储区中适用于请求中实体的所有策略来评估您的请求。

IsAuthorizedWithToken

该IsAuthorizedWithToken操作根据亚马逊 Cognito JSON 网络令牌 (JWT) 中的用户数据生成授权请求。经过验证的权限可直接使用 Amazon Cognito 作为您的策略存储中的身份来源。Verified Permissions 会根据用户 ID 或访问令牌中的声明填充您的请求中的委托人的所有属性。您可以通过 Amazon Cognito 用户池中的用户属性或群组成员资格来授权操作和资源。

您不能在IsAuthorizedWithToken请求中包含有关群组或用户主体类型的信息。您必须将所有委托人数据填充到您提供的 JWT 中。

BatchIs已授权

该BatchIsAuthorized操作在单个 API 请求中处理针对单个委托人或资源的多个授权决策。此操作将请求分组为单个批处理操作，从而最大限度地减少配额使用量，并返回最多 30 个复杂嵌套操作中每个操作的授权决策。通过对单个资源的批量授权，您可以筛选用户可以对资源执行的操作。通过对单个委托人的批量授权，您可以筛选用户可以对其采取操作的资源。

BatchIsAuthorizedWith代币

该BatchIsAuthorizedWithToken操作在一个 API 请求中为单个委托人处理多个授权决策。委托人由您的策略存储身份源以 ID 或访问令牌形式提供。此操作将请求分组为单个批处理操作，以最大限度地减少配额使用量，并返回最多 30 个操作和资源请求中每个请求的授权决策。在您的策略中，您可以通过其属性或其在 Amazon Cognito 用户池中的群组成员资格来授权其访问权限。

与一样IsAuthorizedWithToken，您不能在BatchIsAuthorizedWithToken请求中包含有关群组或用户委托人类型的信息。您必须将所有委托人数据填充到您提供的 JWT 中。

# 测试您的授权模型

要在部署应用程序时了解已验证权限授权决定的影响，您可以在开发策略时使用测试平台和通过 HTTPS REST API 请求对已验证权限进行评估。测试平台是用于评估策略存储库中的 AWS Management Console 授权请求和响应的工具。

随着您从概念理解转向应用程序设计，经过验证的权限 REST API 是您开发的下一步。经过验证的权限 API 接受带有IsAuthorizedIsAuthorizedWithToken、的授权请求以及BatchIs向区域服务终端节点发出的已签名 AWS API 请求。要测试您的授权模型，您可以使用任何 API 客户端生成请求，并验证您的策略是否按预期返回授权决策。

例如，您可以按照以下步骤IsAuthorized在示例策略存储中进行测试。

Test bench

1. 打开 Verified Permissions 控制台，网址为 https://console.aws.amazon.com/verifiedpermissions/。从示例策略存储中创建名为 Store 的策略DigitalPet存储。
2. 在新保单库中选择 “测试台”。
3. 在已验证的权限 API 参考IsAuthorized中填充您的测试平台请求。以下详细信息复制了示例 4 中引用 DigitalPetStore 示例的条件。

   a. 将爱丽丝设为校长。要让校长采取行动，请选择`DigitalPetStore::User`并输入`Alice`。

   b. 将爱丽丝的角色设定为客户。选择 “添加父母”，选择`DigitalPetStore::Role`，然后输入 “客户”。

   c. 将资源设置为顺序 “1234”。对于委托人正在操作的资源，选择`DigitalPetStore::Order`并输入`1234`。

   d. 该`DigitalPetStore::Order`资源需要一个`owner`属性。将 Alice 设置为订单的所有者。选择`DigitalPetStore::User`并输入 `Alice`

   e. Alice 请求查看订单。对于委托人正在采取的行动，请选择`DigitalPetStore::Action::"GetOrder"`。
4. 选择 “运行授权请求”。在未经修改的策略存储中，此请求会导致`ALLOW`决策。请注意返回决策的 “满意” 政策。
5. 从左侧导航菜单中，选择策略。查看静态政策，描述为 “客户角色-获取订单”。
6. 请注意，由于委托人是客户角色并且是资源的所有者，因此已验证权限允许该请求。

REST API

1. 打开 Verified Permissions 控制台，网址为 https://console.aws.amazon.com/verifiedpermissions/。从示例策略存储中创建名为 Store 的策略DigitalPet存储。
2. 记下您的新保单存储区的保单存储 ID。
3. IsAuthorized在已验证权限 API 参考中，复制示例 4 中引用 DigitalPetStore 示例的请求正文。
4. 打开您的 API 客户端，为您的政策存储创建对区域服务端点的请求。如示例所示，填充标题。
5. 粘贴示例请求正文，然后将的`policyStoreId`值更改为您之前记下的策略存储 ID。
6. 提交请求并查看结果。在默认DigitalPet商店策略存储中，此请求会返回一个`ALLOW`决定。

您可以更改测试环境中的策略、架构和请求，以更改结果并做出更复杂的决策。

1. 更改请求的方式将决定从已验证的权限更改为已验证的权限。例如，将爱丽丝的角色更改为`Employee`或将命令 1234 的`owner`属性更改为。Bob
2. 以影响授权决策的方式更改策略。例如，修改描述为 “Cust omer Role-Get Order” 的政策，以删除`User`必须是所有者的条件，`Resource`然后修改请求以使其Bob想要查看订单。
3. 更改架构以允许策略做出更复杂的决策。更新请求实体，以便 Alice 可以满足新的要求。例如，编辑架构`User`以允许其成为`ActiveUsers`或的成员`InactiveUsers`。更新政策，以便只有活跃用户才能查看自己的订单。更新请求实体，使 Alice 成为活跃用户或非活动用户。

# 与应用程序和 AWS SDK 集成

要在您的应用程序中实施 Amazon Verified Permissions，您必须定义您希望应用程序强制执行的策略和架构。在授权模型到位并经过测试后，下一步是从强制执行开始生成 API 请求。为此，您必须设置应用程序逻辑以收集用户数据并将其填充到授权请求中。

应用程序如何使用已验证的权限授权请求

1. 收集有关当前用户的信息。通常，用户的详细信息在经过身份验证的会话的详细信息中提供，例如 JWT 或 Web 会话 Cookie。这些用户数据可能来自链接到您的策略存储库的 Amazon Cognito 身份源，也可能来自其他 OpenID Connect (OIDC) 提供商。
2. 收集有关用户想要访问的资源的信息。通常，当用户做出要求您的应用程序加载新资产的选择时，您的应用程序将收到有关资源的信息。
3. 确定您的用户想要采取的操作。
4. 生成对已验证权限的授权请求，其中包含用户尝试操作的委托人、操作、资源和实体。verified Permissions 会根据您的策略存储中的策略评估请求并返回授权决定。
5. 您的应用程序读取来自已验证权限的允许或拒绝响应，并强制执行对用户请求的决定。

已验证权限 API 操作内置在 AWS SDK 中。要在应用程序中加入经过验证的权限，请将适用于您所选语言的 AWS SDK 集成到应用程序包中。

要了解更多信息并下载 AWS SDK，请参阅工具。 Amazon Web Services

以下是各种 AWS SDK 中已验证权限资源的文档链接。

- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go

- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP
- AWS SDK for Python (Boto)
- AWS SDK for Ruby

以下 AWS SDK for JavaScript 示例`IsAuthorized`源自使用亚马逊验证权限和 Amazon Cognito 简化细粒度授权。

```
const authResult = await avp.isAuthorized({
    principal: 'User::"alice"',
    action: 'Action::"view"',
    resource: 'Photo::"VacationPhoto94.jpg"',
    // whenever our policy references attributes of the entity,
    // isAuthorized needs an entity argument that provides
    // those attributes
    entities: {
       entityList: [
         {
            "identifier": {
                "entityType": "User",
                "entityId": "alice"
            },
            "attributes": {
                "location": {
                    "String": "USA"
                }
            }
         }
       ]
    }
});
```

更多开发者资源

- Amazon 已验证权限研讨会
- Amazon 已验证权限-资源
- 使用亚马逊验证权限为 ASP.NET Core 应用程序实施自定义授权策略提供程序
- 使用 Amazon 验证权限为业务应用程序构建授权服务

- 使用亚马逊认证权限和 Amazon Cognito 简化精细授权

# 添加上下文

情境是指与政策决策相关的信息，但不是您的委托人、行为或资源身份的一部分。您可能只想允许来自一组源 IP 地址的操作，或者仅当您的用户已使用 MFA 登录时才允许执行操作。您的应用程序可以访问此上下文会话数据，并且必须将其填充到授权请求中。已验证权限授权请求中的上下文数据在元素中必须采用 JSON 格式。`contextMap`

说明此内容的示例来自策略存储库示例。要继续操作，请在您的测试环境中创建DigitalPetStore示例策略存储。

以下上下文对象根据示例DigitalPetStore策略存储为应用程序声明每种 Cedar 数据类型之一。

```
"context": {
   "contextMap": {
      "MfaAuthorized": {
         "boolean": true
      },
      "AccountCodes": {
         "set": [
            {
               "long": 111122223333
            },
            {
               "long": 444455556666
            },
            {
               "long": 123456789012
            }
         ]
      },
      "UserAgent": {
         "string": "My UserAgent 1.12"
      },
      "RequestedOrderCount": {
         "long": 4
      },
      "NetworkInfo": {
         "record": {
            "IPAddress": {
               "string": "192.0.2.178"
            },
```

```
            "Country": {
               "string": "United States of America"
            },
            "SSL": {
               "boolean": true
            }
         }
      },
      "approvedBy": {
         "entityIdentifier": {
            "entityId": "Bob",
            "entityType": "DigitalPetStore::User"
         }
      }
   }
}
```

授权上下文中的数据类型

布尔值

二进制`true`或`false`值。在示例中，布尔值 f `true` or `MfaAuthenticated` 表示客户在请求查看订单之前已执行多因素身份验证。

设置

上下文元素的集合。集合成员可以是完全相同的类型（如本例所示），也可以是不同的类型，包括嵌套的集合。在示例中，客户与 3 个不同的账户相关联。

String

由字母、数字或符号组成的序列，用`"`字符括起来。在示例中，`UserAgent`字符串表示客户用来请求查看其订单的浏览器。

长整型

一个整数。在示例中，`RequestedOrderCount`表示此请求是由于客户要求查看其过去的四个订单而产生的批次的一部分。

记录

属性的集合。您必须在请求上下文中声明这些属性。带有架构的策略存储区必须在架构中包含该实体和该实体的属性。在示例中，`NetworkInfo`记录包含有关用户的原始 IP、由客户端确定的该 IP 的地理位置以及传输中的加密的信息。

EntityIdentifier

对请求`entities`元素中声明的实体和属性的引用。在示例中，用户的订单已由员工批准Bob。

要在示例DigitalPetStore应用程序中测试此示例上下文，您必须更新您的请求**`entities`**、策略存储架构和静态策略，描述为 Customer Role-Get Order。

# 正在修改 DigitalPetStore 以接受授权上下文

最初，DigitalPetStore不是一个非常复杂的策略存储。它不包含任何预配置的策略或上下文属性来支持我们所呈现的上下文。要使用此上下文信息评估授权请求示例，请对您的策略存储和授权请求进行以下修改。

Schema

对您的策略存储架构应用以下更新以支持新的上下文属性。更新`GetOrder`actions如下。

```
"GetOrder": {
   "memberOf": [],
   "appliesTo": {
      "resourceTypes": [
         "Order"
      ],
      "context": {
         "type": "Record",
         "attributes": {
            "UserAgent": {
               "required": true,
               "type": "String"
            },
            "approvedBy": {
               "name": "User",
               "required": true,
               "type": "Entity"
            },
            "AccountCodes": {
               "type": "Set",
               "required": true,
               "element": {
                  "type": "Long"
               }
            },
```

```
            "RequestedOrderCount": {
               "type": "Long",
               "required": true
            },
            "MfaAuthorized": {
               "type": "Boolean",
               "required": true
            }
         }
      },
      "principalTypes": [
         "User"
      ]
   }
}
```

要引用在请求上下文NetworkInfo中命名的record数据类型，请在架构中创建一个 C ommonTyp e 结构，如下所示。commonType构造是一组共享的属性，您可以将其应用于不同的实体。

> **Note**
>
> “已验证权限” 可视架构编辑器目前不支持commonType构造。当您将它们添加到架构中时，您将无法再在可视模式下查看架构。

```
"commonTypes": {
   "NetworkInfo": {
      "attributes": {
         "IPAddress": {
            "type": "String",
            "required": true
         },
         "SSL": {
            "required": true,
            "type": "Boolean"
         },
         "Country": {
            "required": true,
            "type": "String"
         }
      },
      "type": "Record"
```

```
    }
}
```

Policy

以下策略设置了每个提供的上下文元素必须满足的条件。它建立在现有静态政策的基础上，描述为“客户角色-获取订单”。该策略最初仅要求发出请求的委托人是资源的所有者。

```
permit (
    principal in DigitalPetStore::Role::"Customer",
    action in [DigitalPetStore::Action::"GetOrder"],
    resource
) when {
    principal == resource.owner &&
    context.MfaAuthorized == true &&
    context.UserAgent like "*My UserAgent*" &&
    context.RequestedOrderCount <= 4 &&
    context.AccountCodes.contains(111122223333) &&
    context.NetworkInfo.Country like "*United States*" &&
    context.NetworkInfo.SSL == true &&
    context.NetworkInfo.IPAddress like "192.0.2.*" &&
    context.approvedBy in DigitalPetStore::Role::"Employee"
};
```

现在，我们要求检索订单的请求必须满足我们在请求中添加的其他上下文条件。

1. 用户必须使用 MFA 登录。
2. 用户的 Web 浏览器`User-Agent`必须包含字符串`My UserAgent`。
3. 用户必须请求查看 4 个或更少的订单。
4. 用户的账户代码之一必须是`111122223333`。
5. 用户的 IP 地址必须来自美国，他们必须处于加密会话中，并且他们的 IP 地址必须以`192.0.2.`开头。
6. 员工必须已批准他们的订单。在授权请求的`entities`元素中，我们将声明一个角色为的`Bob`用户`Employee`。

Request body

在使用适当的架构和策略配置策略存储后，您可以向 “已验证权限 API” 操作提交此授权请求IsAuthorized。请注意，该`entities`区段包含一个角色为的用户的定义`Employee`。Bob

```
{
   "principal": {
      "entityType": "DigitalPetStore::User",
      "entityId": "Alice"
   },
   "action": {
      "actionType": "DigitalPetStore::Action",
      "actionId": "GetOrder"
   },
   "resource": {
      "entityType": "DigitalPetStore::Order",
      "entityId": "1234"
   },
   "context": {
     "contextMap": {
      "MfaAuthorized": {
       "boolean": true
      },
      "UserAgent": {
       "string": "My UserAgent 1.12"
      },
      "RequestedOrderCount":{
       "long": 4
      },
      "AccountCodes": {
       "set": [
        {"long": 111122223333},
        {"long": 444455556666},
        {"long": 123456789012}
        ]
      },
      "NetworkInfo": {
        "record": {
          "IPAddress": {"string": "192.0.2.178"},
          "Country": {"string": "United States of America"},
          "SSL": {"boolean": true}
        }
      },
      "approvedBy": {
        "entityIdentifier": {
         "entityId": "Bob",
         "entityType": "DigitalPetStore::User"
        }
```

```
      }
     }
   },
   "entities": {
     "entityList": [
       {
         "identifier": {
            "entityType": "DigitalPetStore::User",
            "entityId": "Alice"
         },
         "attributes": {
            "memberId": {
              "string": "801b87f2-1a5c-40b3-b580-eacad506d4e6"
              }
         },
         "parents": [
            {
            "entityType": "DigitalPetStore::Role",
            "entityId": "Customer"
            }
         ]
       },
       {
         "identifier": {
            "entityType": "DigitalPetStore::User",
            "entityId": "Bob"
         },
         "attributes": {
            "memberId": {
              "string": "49d9b81e-735d-429c-989d-93bec0bcfd8b"
              }
         },
         "parents": [
            {
            "entityType": "DigitalPetStore::Role",
            "entityId": "Employee"
            }
         ]
       },
       {
         "identifier": {
            "entityType": "DigitalPetStore::Order",
            "entityId": "1234"
         },
```

```
         "attributes": {
            "owner": {
              "entityIdentifier": {
                "entityType": "DigitalPetStore::User",
                "entityId": "Alice"
              }
            }
         },
         "parents": []
       }
     ]
   },
   "policyStoreId": "PSEXAMPLEabcdefg111111"
}
```

# Amazon Verified Permissions 的安全性

AWS 十分重视云安全性。为了满足对安全性最敏感的组织的需求，我们打造了具有超高安全性的数据中心和网络架构。作为 AWS 客户，您也将从这些数据中心和网络架构受益。

安全性是 AWS 和您的共同责任。责任共担模式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 Cloud 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。第三方审核员定期测试和验证我们的安全性的有效性，作为 AWS 合规性计划的一部分。要了解适用于 Amazon Verified Permissions 的合规性计划，请参阅按合规性计划提供的范围内 AWS 服务。
- 云中的安全性 - 您的责任由您使用的 AWS 服务决定。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 Verified Permissions 时应用责任共担模式。以下主题介绍了如何配置 Verified Permissions 以实现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 Verified Permissions 资源。

主题



# Amazon Verified Permissions 中的数据保护

AWS 责任共担模式适用于 Amazon Verified Permissions 中的数据保护。如该模式中所述，AWS 负责保护运行所有 AWS Cloud 的全球基础设施。您负责维护对托管在此基础设施上的内容的控制。此内容包括您所使用的 AWS 服务 的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题解答。有关欧洲数据保护的信息，请参阅 AWS 安全性博客上的 AWS 责任共担模式和 GDPR 博客文章。

- 出于数据保护目的，我们建议您保护 AWS 账户凭证并使用 AWS IAM Identity Center 或 AWS Identity and Access Management（IAM）设置单个用户。这样，每个用户只会获得履行其工作职责所需的权限。

- 我们建议您通过以下方式保护您的数据：
  - 对每个账户使用多重身份验证（MFA）。
  - 使用 SSL/TLS 与 AWS 资源进行通信。我们要求使用 TLS 1.2。
  - 使用 AWS CloudTrail 设置 API 和用户活动日志记录。
  - 使用 AWS 加密解决方案以及 AWS 服务中的所有默认安全控制。
  - 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
  - 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 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准 (FIPS) 第 140-2 版》。
- 我们强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如 Name（名称）字段）。这包括使用控制台、API、AWS CLI 或 AWS SDK 处理 Verified Permissions 或其他 AWS 服务时。您在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供 URL，我们强烈建议您不要在 URL 中包含凭证信息来验证您对该服务器的请求。
- 您的操作名称不应包含任何敏感信息。
- 我们还强烈建议您始终为您的实体（资源和主体）使用唯一、不可变且不可重复使用的标识符。在测试环境中，您可以选择使用简单的实体标识符，例如使用 `jane` 或 `bob` 作为 `User` 类实体的名称。但是，在生产系统中，出于安全考虑，使用不可重复使用的唯一值至关重要。我们建议您使用通用唯一标识符（UUID）之类的值。例如，思考一下有一个离开公司的用户 `jane`。后来，您让其他人使用了 `jane` 这个名字。该新用户可以自动访问仍引用 `User::"jane"` 的策略所授予的所有内容。Verified Permissions 和 Cedar 无法区分新用户和以前的用户。

  本指南适用于主体和资源标识符。请务必使用保证唯一且永远不可重复使用的标识符，确保您不会因为策略中存在旧标识符而在无意中授予他人访问权限。

- 请确保您提供的用于定义 `Long` 和 `Decimal` 值的字符串在每种类型的有效范围内。此外，请确保您使用任何算术运算符得出的值均不会超出有效范围。如果超出有效范围，则该操作将导致溢出异常。导致错误的策略将被忽略，这意味着许可策略可能会意外无法允许访问，或者禁止策略可能会意外无法阻止访问。

## 数据加密

Amazon Verified Permissions 使用 AWS 托管式密钥自动加密所有客户数据，例如策略，因此既没有必要也不支持使用客户托管密钥。

# 适用于 Amazon Verified Permissions 的身份和访问管理

AWS Identity and Access Management (IAM) AWS 服务 可以帮助管理员安全地控制对 AWS 资源的访问权限。 IAM 管理员控制谁可以通过身份验证（登录）和授权（拥有权限）使用已验证的权限资源。 IAM 无需支付额外费用即可使用。 AWS 服务

主题



## 受众

您使用 AWS Identity and Access Management (IAM) 的方式会有所不同，具体取决于您在已验证权限中所做的工作。

服务用户 – 如果需要使用 Verified Permissions 服务来完成任务，则您的管理员会为您提供所需的凭证和权限。如果需要使用更多 Verified Permissions 功能才能完成工作，那您可能需要额外权限。了解如何管理访问权限有助于您向管理员请求适合的权限。如果您无法访问 Verified Permissions 中的功能，请参阅Amazon Verified Permissions 身份和访问问题排查。

服务管理员 – 如果您在公司负责管理 Verified Permissions 资源，则您可能具有 Verified Permissions 的完全访问权限。您负责确定您的服务用户应访问哪些 Verified Permissions 功能和资源。然后，您必须向 IAM 管理员提交更改服务用户权限的请求。查看此页面上的信息以了解的基本概念 IAM。要详细了解贵公司如何使用经过验证 IAM 的权限，请参阅Amazon 已验证权限的工作原理 IAM。

IAM 管理员-如果您是 IAM 管理员，则可能需要详细了解如何编写策略来管理对已验证权限的访问权限。要查看可在中使用的基于身份的已验证权限策略示例 IAM，请参阅。适用于 Amazon Verified Permissions 的基于身份的策略示例

# 使用身份进行身份验证

身份验证是您 AWS 使用身份凭证登录的方式。您必须以 IAM 用户身份或通过担任 IAM 角色进行身份验证（登录 AWS）。 AWS 账户根用户

您可以使用通过身份源提供的凭据以 AWS 联合身份登录。 AWS IAM Identity Center （ IAM Identity Center）用户、贵公司的单点登录身份验证以及您的 Google 或 Facebook 凭据就是联合身份的示例。当您以联合身份登录时，您的管理员之前使用 IAM 角色设置了联合身份。当你使用联合访问 AWS 时，你就是在间接扮演一个角色。

根据您的用户类型，您可以登录 AWS Management Console 或 AWS 访问门户。有关登录的更多信息 AWS，请参阅《AWS 登录 用户指南》中的如何登录到您 AWS 账户的。

如果您 AWS 以编程方式访问，则会 AWS 提供软件开发套件 (SDK) 和命令行接口 (CLI)，以便使用您的凭据对请求进行加密签名。如果您不使用 AWS 工具，则必须自己签署请求。有关使用推荐的方法自行签署请求的更多信息，请参阅IAM 用户指南中的签署 AWS API 请求。

无论使用何种身份验证方法，您可能需要提供其他安全信息。例如， AWS 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅《AWS IAM Identity Center 用户指南》中的多重身份验证和《IAM 用户指南》中的在 AWS中使用多重身份验证（MFA）。

## AWS 账户 root 用户

创建时 AWS 账户，首先要有一个登录身份，该身份可以完全访问账户中的所有资源 AWS 服务 和资源。此身份被称为 AWS 账户 root 用户，使用您创建账户时使用的电子邮件地址和密码登录即可访问该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关需要您以根用户身份登录的任务的完整列表，请参阅《IAM 用户指南》中的需要根用户凭证的任务。

## 联合身份

作为最佳实践，要求人类用户（包括需要管理员访问权限的用户）使用与身份提供商的联合身份验证 AWS 服务 通过临时证书进行访问。

联合身份是指您的企业用户目录、Web 身份提供商、Identity C enter 目录中的用户，或者任何使用 AWS 服务 通过身份源提供的凭据进行访问的用户。 AWS Directory Service当联合身份访问时 AWS 账户，他们将扮演角色，角色提供临时证书。

要集中管理访问权限，建议您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中创建用户和群组，也可以连接并同步到您自己的身份源中的一组用户和群组，以便在您的所有 AWS 账户 和

应用程序中使用。有关 IAM Identity Center 的信息，请参阅《AWS IAM Identity Center 用户指南》中的什么是 IAM Identity Center？

## IAM 用户和群组

I AM 用户是您 AWS 账户 内部对个人或应用程序具有特定权限的身份。在可能的情况下，我们建议使用临时凭证，而不是创建具有长期凭证（如密码和访问密钥）的 IAM 用户。但是，如果您有一些特定的使用场景需要长期凭证以及 IAM 用户，我们建议您轮换访问密钥。有关更多信息，请参阅《IAM 用户指南》中的对于需要长期凭证的使用场景定期轮换访问密钥。

IAM 组 是一个指定一组 IAM 用户的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您有一个名为 IAM Admins 的组并为该组授予管理 IAM 资源的权限。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅《IAM 用户指南》中的何时创建 IAM 用户（而不是角色）。

## IAM 角色

IAM 角色是您内部具有特定权限 AWS 账户 的身份。它类似于 IAM 用户，但与特定人员不关联。您可以 AWS Management Console 通过切换 IAM 角色在中临时扮演角色。您可以通过调用 AWS CLI 或 AWS API 操作或使用自定义 URL 来代入角色。有关使用角色的方法的更多信息，请参阅《IAM 用户指南》中的使用 IAM 角色。

IAM 具有临时证书的角色在以下情况下很有用：

- 联合用户访问 – 要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关联合身份验证的角色的信息，请参阅《IAM 用户指南》中的为第三方身份提供商创建角色。如果您使用 IAM Identity Center，则需要配置权限集。为控制您的身份在进行身份验证后可以访问的内容，IAM Identity Center 会将权限集与 IAM 中的角色相关联。有关权限集的信息，请参阅《AWS IAM Identity Center 用户指南》中的 权限集。
- 临时 IAM 用户权限 — IAM 用户或角色可以代入一个 IAM 角色，以临时获得特定任务的不同权限。
- 跨账户存取 - 您可以使用 IAM 角色允许其他账户中的某个人（可信任主体）访问您账户中的资源。角色是授予跨账户存取权限的主要方式。但是，对于某些资源 AWS 服务，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解角色和基于资源的跨账户访问策略之间的区别，请参阅用户指南中的IAM 角色与基于资源的策略有何不同。IAM
- 上运行的应用程序 Amazon EC2-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证书。这优先于在 EC2 实例中存储访问密钥。要向 EC2 实例分配

AWS 角色并使其可供其所有应用程序使用，您需要创建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IAM 用户指南中的使用 IAM 角色向在 Amazon EC2 实例上运行的应用程序授予权限。

要了解是使用 IAM 角色还是 IAM 用户，请参阅用户指南中的何时创建 IAM 角色（而不是IAM 用户）。

## 使用策略管理访问

您可以 AWS 通过创建策略并将其附加到 AWS 身份或资源来控制中的访问权限。策略是其中的一个对象 AWS ，当与身份或资源关联时，它会定义其权限。 AWS 在委托人（用户、root 用户或角色会话）发出请求时评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略都以 JSON 文档的 AWS 形式存储在中。有关 JSON 策略文档的结构和内容的更多信息，请参阅《IAM 用户指南》中的 JSON 策略概览。

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

默认情况下，用户和角色没有权限。要授予用户对其所需资源执行操作的权限， IAM 管理员可以创建 IAM 策略。然后，管理员可以将 IAM 策略添加到角色中，用户可以代入这些角色。

IAM 无论您使用何种方法执行操作，策略都会定义该操作的权限。例如，假设您有一个允许 `iam:GetRole` 操作的策略。拥有该策略的用户可以从 AWS Management Console AWS CLI、或 AWS API 获取角色信息。

### 基于身份的策略

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅IAM 用户指南中的创建 IAM 策略。

基于身份的策略可以进一步归类为内联策略或托管策略。内联策略直接嵌入单个用户、组或角色中。托管策略是独立的策略，您可以将其附加到中的多个用户、群组和角色 AWS 账户。托管策略包括 AWS 托管策略和客户托管策略。要了解如何在托管式策略和内联策略之间进行选择，请参阅《IAM 用户指南》中的在托管式策略与内联策略之间进行选择。

### 基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资

源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 AWS 服务。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略 IAM 中使用 AWS 托管策略。

### 访问控制列表 (ACL)

访问控制列表（ACL）控制哪些主体（账户成员、用户或角色）有权访问资源。ACL 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

Amazon S3 和 Amazon VPC 就是支持 ACL 的服务示例。 AWS WAF要了解有关 ACL 的更多信息，请参阅《Amazon Simple Storage Service 开发人员指南》中的访问控制列表（ACL）概览。

### 其他策略类型

AWS 支持其他不太常见的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界 — 权限边界是一项高级功能，您可以在其中设置基于身份的策略可以向 IAM 实体（IAM 用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅《IAM 用户指南》中的 IAM 实体的权限边界。
- 服务控制策略 (SCP)-SCP 是 JSON 策略，用于指定组织或组织单位 (OU) 的最大权限。 AWS Organizations AWS Organizations 是一项用于对您的企业拥有的多 AWS 账户 项进行分组和集中管理的服务。如果在组织内启用了所有功能，则可对任意或全部账户应用服务控制策略 (SCP)。SCP 限制成员账户中的实体（包括每个 AWS 账户根用户实体）的权限。有关 Organizations 和 SCP 的更多信息，请参阅《AWS Organizations 用户指南》中的 SCP 的工作原理。
- 会话策略 – 会话策略是当您以编程方式为角色或联合用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅《IAM 用户指南》中的会话策略。

### 多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解在涉及多种策略类型时如何 AWS 确定是否允许请求，请参阅IAM 用户指南中的策略评估逻辑。

# Amazon 已验证权限的工作原理 IAM

在使用管理 IAM 对已验证权限的访问权限之前，请先了解哪些 IAM 功能可用于已验证权限。

IAM 您可以通过 Amazon 验证权限使用的功能

| IAM 功能 | 支持 Verified Permissions |
| --- | --- |
| 基于身份的策略 | 是 |
| 基于资源的策略 | 否 |
| 策略操作 | 是 |
| 策略资源 | 是 |
| 策略条件密钥 | 否 |
| ACL | 否 |
| ABAC（策略中的标签） | 否 |
| 临时凭证 | 是 |
| 主体权限 | 是 |
| 服务角色 | 否 |
| 服务相关角色 | 否 |

要全面了解已验证的权限和其他 AWS 服务如何与大多数 IAM 功能配合使用，请参阅IAM 用户指南IAM中与之配合使用的AWS 服务。

## 适用于 Verified Permissions 的基于身份的策略

| | |
| --- | --- |
| 支持基于身份的策略 | 是 |

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅IAM 用户指南中的创建 IAM 策略。

使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源，以及允许或拒绝操作的条件。您无法在基于身份的策略中指定主体，因为它适用于其附加的用户或角色。要了解可在 JSON 策略中使用的所有元素，请参阅《IAM 用户指南》中的 IAM JSON 策略元素引用。

适用于 Verified Permissions 的基于身份的策略示例

要查看 Verified Permissions 基于身份的策略的示例，请参阅 适用于 Amazon Verified Permissions 的基于身份的策略示例。

## Verified Permissions 中基于资源的策略

| | |
|---|---|
| 支持基于资源的策略 | 否 |

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 AWS 服务。

要启用跨账户访问权限，您可以将整个账户或另一个账户中的 IAM 实体指定为基于资源的策略中的委托人。将跨账户主体添加到基于资源的策略只是建立信任关系工作的一半而已。当委托人和资源处于不同位置时 AWS 账户，可信账户中的 IAM 管理员还必须向委托人实体（用户或角色）授予访问资源的权限。他们通过将基于身份的策略附加到实体以授予权限。但是，如果基于资源的策略向同一个账户中的主体授予访问权限，则不需要额外的基于身份的策略。有关更多信息，请参阅《IAM 用户指南》IAM中的跨账户资源访问权限。

## Verified Permissions 的策略操作

| | |
|---|---|
| 支持策略操作 | 是 |

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

JSON 策略的 Action 元素描述可用于在策略中允许或拒绝访问的操作。策略操作通常与关联的 AWS API 操作同名。有一些例外情况，例如没有匹配 API 操作的仅限权限 操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

在策略中包含操作以授予执行关联操作的权限。

有关 Verified Permissions 操作的列表，请参阅《服务授权参考》中的 Amazon Verified Permissions 定义的操作。

Verified Permissions 中的策略操作在操作前使用以下前缀：

```
verifiedpermissions
```

要在单个语句中指定多项操作，请使用逗号将它们隔开。

```
"Action": [
      "verifiedpermissions:action1",
      "verifiedpermissions:action2"
         ]
```

您也可以使用通配符（*）指定多个操作。例如，要指定以单词 Get 开头的所有操作，包括以下操作：

```
"Action": "verifiedpermissions:Get*"
```

要查看 Verified Permissions 基于身份的策略的示例，请参阅 适用于 Amazon Verified Permissions 的基于身份的策略示例。

## Verified Permissions 的策略资源

| | |
|---|---|
| 支持策略资源 | 是 |

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体 可以对什么资源执行操作，以及在什么条件下执行。

Resource JSON 策略元素指定要向其应用操作的一个或多个对象。语句必须包含 Resource 或 NotResource 元素。作为最佳实践，请使用其 Amazon 资源名称 (ARN) 指定资源。对于支持特定资源类型（称为资源级权限）的操作，您可以执行此操作。

对于不支持资源级权限的操作（如列出操作），请使用通配符 (*) 指示语句应用于所有资源。

```
"Resource": "*"
```

有关 Verified Permissions 资源类型及其 ARN 的列表，请参阅《服务授权参考》中的 Amazon Verified Permissions 定义的资源类型。要了解您可以在哪些操作中指定每个资源的 ARN，请参阅 Amazon Verified Permissions 定义的操作。

## Verified Permissions 的策略条件键

| | |
|---|---|
| 支持特定于服务的策略条件密钥 | 否 |

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

在 Condition 元素（或 Condition 块）中，可以指定语句生效的条件。Condition 元素是可选的。您可以创建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。

如果您在一个语句中指定多个 Condition 元素，或在单个 Condition 元素中指定多个键，则 AWS 使用逻辑 AND 运算评估它们。如果您为单个条件键指定多个值，则使用逻辑OR运算来 AWS 评估条件。在授予语句的权限之前必须满足所有的条件。

在指定条件时，您也可以使用占位符变量。例如，只有在使用 IAM 用户名标记 IAM 用户时，您才能为其授予访问资源的权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM 策略元素：变量和标签。

AWS 支持全局条件密钥和特定于服务的条件密钥。要查看所有 AWS 全局条件键，请参阅《IAM 用户指南》中的AWS 全局条件上下文密钥。

## Verified Permissions 中的 ACL

| | |
|---|---|
| 支持 ACL | 否 |

访问控制列表 (ACL) 控制哪些主体（账户成员、用户或角色）有权访问资源。ACL 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

## ABAC 与 Verified Permissions 结合使用

| | |
|---|---|
| 支持 ABAC（策略中的标签） | 否 |

基于属性的访问权限控制 (ABAC) 是一种授权策略，该策略基于属性来定义权限。在中 AWS，这些属性称为标签。您可以将标签附加到 IAM 实体（用户或角色）和许多 AWS 资源。标记实体和资源是 ABAC 的第一步。然后设计 ABAC 策略，以在主体的标签与他们尝试访问的资源标签匹配时允许操作。

ABAC 在快速增长的环境中非常有用，并在策略管理变得繁琐的情况下可以提供帮助。

要基于标签控制访问，您需要使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 条件键在策略的条件元素中提供标签信息。

如果某个服务对于每种资源类型都支持所有这三个条件键，则对于该服务，该值为是。如果某个服务仅对于部分资源类型支持所有这三个条件键，则该值为部分。

有关 ABAC 的更多信息，请参阅《IAM 用户指南》中的什么是 ABAC？。要查看设置 ABAC 步骤的教程，请参阅《IAM 用户指南》中的使用基于属性的访问权限控制（ABAC）。

## 将临时凭证用于 Verified Permissions

| | |
|---|---|
| 支持临时凭证 | 是 |

当你使用临时证书登录时，有些 AWS 服务 不起作用。有关其他信息，包括哪些 AWS 服务 适用于临时证书 AWS 服务 ，请参阅《IAM 用户指南》 IAM中的 “适用于临时证书”。

如果您使用除用户名和密码之外的任何方法登录，则 AWS Management Console 使用的是临时证书。例如，当您 AWS 使用公司的单点登录 (SSO) 链接进行访问时，该过程会自动创建临时证书。当您以用户身份登录控制台，然后切换角色时，您还会自动创建临时凭证。有关切换角色的更多信息，请参阅《IAM 用户指南》中的切换到角色（控制台）。

您可以使用 AWS CLI 或 AWS API 手动创建临时证书。然后，您可以使用这些临时证书进行访问 AWS。 AWS 建议您动态生成临时证书，而不是使用长期访问密钥。有关更多信息，请参阅 IAM 中的临时安全凭证。

## Verified Permissions 的跨服务主体权限

| | |
|---|---|
| 支持主体权限 | 是 |

当您使用 IAM 用户或角色在中执行操作时 AWS，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用调用委托人的权限以及 AWS 服务 向下

游服务发出请求的请求。 AWS 服务只有当服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两个操作的权限。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。

### Verified Permissions 的服务角色

| | |
|---|---|
| 支持服务角色 | 否 |

服务角色是由一项服务代入、代表您执行操作的 IAM 角色。 IAM 管理员可以在内部创建、修改和删除服务角色 IAM。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务委派权限的角色。

### Verified Permissions 的服务相关角色

| | |
|---|---|
| 支持服务相关角色 | 否 |

服务相关角色是一种与服务相关联的 AWS 服务服务角色。服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。 IAM 管理员可以查看但不能编辑服务相关角色的权限。

有关创建或管理服务相关角色的详细信息，请参阅与之配合 IAM使用的AWS 服务。在表中查找服务相关角色列中包含 Yes 的表。选择是链接以查看该服务的服务相关角色文档。

## 适用于 Amazon Verified Permissions 的基于身份的策略示例

默认情况下，用户和角色没有创建或修改 Verified Permissions 资源的权限。他们也无法使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或 AWS API 执行任务。 IAM 管理员必须创建 IAM 策略，授予用户和角色对其所需资源执行操作的权限。然后，管理员必须为需要这些策略的用户附加这些策略。

要了解如何使用这些示例 JSON 策略文档创建 IAM 基于身份的策略，请参阅IAM 用户指南中的创建 IAM 策略。

有关 Verified Permissions 定义的操作和资源类型的详细信息，包括每种资源类型的 ARN 格式，请参阅《服务授权参考》中的 Amazon Verified Permissions 的操作、资源和条件键。

主题

- 策略最佳实践

- 使用 Verified Permissions 控制台
- 允许用户查看他们自己的权限

## 策略最佳实践

基于身份的策略确定某个人是否可以创建、访问或删除您账户中的 Verified Permissions 资源。这些操作可能会使 AWS 账户产生成本。创建或编辑基于身份的策略时，请遵循以下指南和建议：

- 开始使用 AWS 托管策略并转向最低权限权限 — 要开始向用户和工作负载授予权限，请使用为许多常见用例授予权限的AWS 托管策略。它们在你的版本中可用 AWS 账户。我们建议您通过定义针对您的用例的 AWS 客户托管策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管式策略 或 工作职能的AWS 托管式策略。
- 应用最低权限权限-使用 IAM 策略设置权限时，仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用应用权限 IAM 的更多信息，请参阅《IAM 用户指南》 IAM中的策略和权限。
- 使用 IAM 策略中的条件进一步限制访问权限-您可以在策略中添加条件以限制对操作和资源的访问权限。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果服务操作是通过特定的方式使用的，则也可以使用条件来授予对服务操作的访问权限 AWS 服务，例如 AWS CloudFormation。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。
- 使用 IAM Access Analyzer 验证您的 IAM 策略，以确保权限的安全性和功能性 — IAM Access Analyzer 会验证新的和现有的策略，以便这些策略符合 IAM 策略语言 (JSON) 和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的 IAM Access Analyzer 策略验证。
- 需要多重身份验证 (MFA)-如果 AWS 账户您的场景需要 IAM 用户或根用户，请启用 MFA 以提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的配置受 MFA 保护的 API 访问。

有关中最佳做法的更多信息 IAM，请参阅《IAM 用户指南》 IAM中的安全最佳实践。

## 使用 Verified Permissions 控制台

要访问 Amazon Verified Permissions 控制台，您必须具有一组最低的权限。这些权限必须允许您列出和查看有关中已验证权限资源的详细信息 AWS 账户。如果创建比必需的最低权限更为严格的基于身份的策略，对于附加了该策略的实体（用户或角色），控制台将无法按预期正常运行。

对于仅调用 AWS CLI 或 AWS API 的用户，您无需为其设置最低控制台权限。相反，只允许访问与其尝试执行的 API 操作相匹配的操作。

为确保用户和角色仍然可以使用已验证权限控制台，还需要将已验证的权限*ConsoleAccess*或*ReadOnly* AWS 托管策略附加到实体。有关更多信息，请参阅《IAM 用户指南》中的为用户添加权限。

## 允许用户查看他们自己的权限

该示例说明了您如何创建策略，以允许 IAM 用户查看附加到其用户身份的内联和托管策略。此策略包括在控制台上或使用 AWS CLI 或 AWS API 以编程方式完成此操作的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

# Amazon Verified Permissions 身份和访问问题排查

使用以下信息可帮助您诊断和修复在使用 Verified Permissions 和 IAM时可能遇到的常见问题。

主题



## 我无权在 Verified Permissions 中执行操作

如果您收到错误提示，表明您无权执行某个操作，则您必须更新策略以允许执行该操作。

当 `mateojackson` IAM 用户尝试使用控制台查看有关虚构 *`my-example-widget`* 资源的详细信息，但不拥有虚构 `verifiedpermissions:`*`GetWidget`* 权限时，会发生以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 verifiedpermissions:GetWidget on resource: my-example-widget
```

在此情况下，必须更新 `mateojackson` 用户的策略，以允许使用 `verifiedpermissions:`*`GetWidget`* 操作访问 *`my-example-widget`* 资源。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

## 我无权执行 iam：PassRole

如果您收到一个错误，表明您无权执行 `iam:PassRole` 操作，则必须更新策略以允许您将角色传递给 Verified Permissions。

有些 AWS 服务 允许您将现有角色传递给该服务，而不是创建新的服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为 `marymajor` 的 IAM 用户尝试使用控制台在 Verified Permissions 中执行操作时，会发生以下示例错误。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在这种情况下，必须更新 Mary 的策略以允许她执行 `iam:PassRole` 操作。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

### 我想允许我以外的人访问我的 AWS 账户 “已验证权限” 资源

您可以创建一个角色，以便其他账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以担任角色。对于支持基于资源的策略或访问控制列表（ACL）的服务，您可以使用这些策略向人员授予对您的资源的访问权。

要了解更多信息，请参阅以下内容：

- 要了解 Verified Permissions 是否支持这些功能，请参阅 Amazon 已验证权限的工作原理 IAM。
- 要了解如何提供对您拥有的资源的访问权限，请参阅用户指南中的向您拥有的另一 AWS 账户 个 IAM IAM 用户提供访问权限。 AWS 账户
- 要了解如何向第三方提供对您的资源的访问权限 AWS 账户，请参阅IAM 用户指南中的向第三方提供访问权限。 AWS 账户
- 要了解如何通过身份联合验证提供访问权限，请参阅《IAM 用户指南》中的为经过外部身份验证的用户（身份联合验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户访问的区别，请参阅IAM 用户指南IAM中的跨账户资源访问权限。

## Amazon Verified Permissions 合规性验证

要了解是否属于特定合规计划的范围，请参阅AWS 服务 “按合规计划划分的范围” ”，然后选择您感兴趣的合规计划。 AWS 服务 有关一般信息，请参阅AWS 合规计划AWS。

您可以使用下载第三方审计报告 AWS Artifact。有关更多信息，请参阅中的 “下载报告” 中的 “ AWS Artifact。

您在使用 AWS 服务 时的合规责任取决于您的数据的敏感性、贵公司的合规目标以及适用的法律和法规。 AWS 提供了以下资源来帮助实现合规性：

- 安全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，并提供了部署以安全性和合规性为重点 AWS 的基准环境的步骤。
- 构建 HIPAA 安全与合规性 Amazon Web Services — 本白皮书描述了公司如何使用 AWS 来创建符合 HIPAA 资格的应用程序。

> **Note**
>
> 并非所有 AWS 服务 人都符合 HIPAA 资格。有关更多信息，请参阅符合 HIPAA 要求的服务参考。

- AWS 合规资源AWS — 此工作簿和指南集可能适用于您所在的行业和所在地区。
- AWS 客户合规指南 — 从合规角度了解责任共担模式。这些指南总结了保护的最佳实践， AWS 服务并将指南映射到跨多个框架（包括美国国家标准与技术研究院 (NIST)、支付卡行业安全标准委员会 (PCI) 和国际标准化组织 (ISO)）的安全控制。
- 使用AWS Config 开发人员指南中的规则评估资源 — 该 AWS Config 服务评估您的资源配置在多大程度上符合内部实践、行业准则和法规。
- AWS Security Hub— 这 AWS 服务 可以全面了解您的安全状态 AWS。Security Hub 通过安全控件评估您的 AWS 资源并检查其是否符合安全行业标准和最佳实践。有关受支持服务及控件的列表，请参阅 Security Hub 控件参考。
- Amazon GuardDuty — 它通过监控您的 AWS 账户环境中是否存在可疑和恶意活动，来 AWS 服务检测您的工作负载、容器和数据面临的潜在威胁。 GuardDuty 通过满足某些合规性框架规定的入侵检测要求，可以帮助您满足各种合规性要求，例如 PCI DSS。
- AWS Audit Manager— 这 AWS 服务 可以帮助您持续审计 AWS 使用情况，从而简化风险管理以及对法规和行业标准的合规性。

## Amazon Verified Permissions 的顺应力

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，这些可用区与延迟率低、吞吐量高且冗余性高的网络连接在一起。利用可用区，您可以设计和操作在可用区之间无中断地自动实现故障转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

当您创建 Verified Permissions 策略存储时，它是在单个 AWS 区域中创建的，并会自动复制到构成该区域可用区的数据中心。目前，Verified Permissions 不支持任何跨区域复制。

有关 AWS 区域和可用区的更多信息，请参阅 AWS 全球基础设施。

# 监控 Amazon Verified Permissions

监控是保持 Amazon Verified Permissions 和您的其他 AWS 解决方案的可靠性、可用性和性能的重要环节。AWS 提供了以下监控工具来监控 Verified Permissions、在出现错误时进行报告并适时自动采取措施：

- AWS CloudTrail 捕获由您的 AWS 账户或代表该账户发出的 API 调用和相关事件，并将日志文件传输到您指定的 Amazon S3 桶。您可以标识哪些用户和账户调用了 AWS、从中发出调用的源 IP 地址以及调用的发生时间。有关更多信息，请参阅 AWS CloudTrail 用户指南。

# 使用 AWS CloudTrail 记录 Amazon Verified Permissions API 调用

Amazon Verifie AWS CloudTrail d Permissions 与一项服务集成，该服务在 “已验证权限” 中记录用户、角色或AWS服务所采取的操作。 CloudTrail 将所有针对已验证权限的 API 调用捕获为事件。捕获的调用包含来自 Verified Permissions 控制台的调用和对 Verified Permissions API 操作的代码调用。如果您创建跟踪，则可以允许将 CloudTrail 事件持续传输到 Amazon S3 存储桶，包括已验证权限的事件。如果您未配置跟踪，您仍然可以在 CloudTrail 控制台的 “事件历史记录” 中查看最新的事件。使用收集的信息 CloudTrail，您可以确定向已验证权限发出的请求、发出请求的 IP 地址、谁发出了请求、何时发出请求以及其他详细信息。

要了解更多信息 CloudTrail，请参阅《AWS CloudTrail用户指南》。

## 已验证的权限信息位于 CloudTrail

CloudTrail 在您创建账户AWS 账户时已在您的账户上启用。当活动发生在 “已验证权限” 中时，该活动会与其他AWS服务 CloudTrail 事件一起记录在事件历史记录中。您可以在 AWS 账户 中查看、搜索和下载最新事件。有关更多信息，请参阅使用事件历史记录查看 CloudTrail 事件。

要持续记录 AWS 账户中的事件（包括 Verified Permissions 的事件），请创建跟踪记录。跟踪允许 CloudTrail 将日志文件传输到 Amazon S3 存储桶。默认情况下，在使用控制台创建跟踪时，此跟踪应用于所有 AWS 区域。此跟踪记录在 AWS 分区中记录所有区域中的事件，并将日志文件传送至您指定的 Amazon S3 桶。此外，您可以配置其他AWS服务，以进一步分析和处理 CloudTrail 日志中收集的事件数据。有关更多信息，请参阅下列内容：

- Overview for creating a trail
- CloudTrail 支持的服务和集成
- 配置 Amazon SNS 通知 CloudTrail

- 接收来自多个区域的 CloudTrail 日志文件和接收来自多个账户的 CloudTrail 日志文件

所有已验证的权限操作均由《Amazon 已验证权限 API 参考指南》记录 CloudTrail 并记录在案。例如，对`CreateIdentitySourceDeletePolicy`、和`ListPolicyStores`操作的调用会在 CloudTrail 日志文件中生成条目。

每个事件或日记账条目都包含有关生成请求的人员信息。身份信息有助于您确定以下内容：

- 请求是使用根用户凭证还是 AWS Identity and Access Management（IAM）用户凭证发出的。
- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其它 AWS 服务发出。

有关更多信息，请参阅 CloudTrail userIdentity 元素。

创建跟踪或事件数据存储时，默认情况下不会记录IsAuthorized和IsAuthorizedWithToken之类的数据事件。要记录 CloudTrail 数据事件，必须明确添加要为其收集活动的支持的资源或资源类型。有关更多信息，请参阅《AWS CloudTrail 用户指南》中的数据事件。

## 了解 Verified Permissions 日志文件条目

跟踪是一种配置，允许将事件作为日志文件传输到您指定的 Amazon S3 存储桶。 CloudTrail 日志文件包含一个或多个日志条目。事件代表来自任何来源的单个请求，包括有关请求的操作、操作的日期和时间、请求参数等的信息。 CloudTrail 日志文件不是公共 API 调用的有序堆栈跟踪，因此它们不会按任何特定的顺序出现。

主题

- IsAuthorized
- BatchIsAuthorized
- CreatePolicyStore
- ListPolicyStores
- DeletePolicyStore
- PutSchema
- GetSchema
- CreatePolicyTemplate
- DeletePolicyTemplate
- CreatePolicy

- GetPolicy
- CreateIdentitySource
- GetIdentitySource
- ListIdentitySources
- DeleteIdentitySource

> **Note**
>
> 为了保护数据隐私，已从示例中删除了一些字段。

## IsAuthorized


```
{
    "eventVersion": "1.08",
    "userIdentity": {
  "type": "AssumedRole",
  "principalId": "EXAMPLE_PRINCIPAL_ID",
  "arn": "arn:aws:iam::123456789012:role/ExampleRole",
  "accountId": "123456789012",
  "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
    },
    "eventTime": "2023-11-20T22:55:03Z",
    "eventSource": "verifiedpermissions.amazonaws.com",
    "eventName": "IsAuthorized",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "203.0.113.0",
    "userAgent": "aws-cli/2.11.18 Python/3.11.3 Linux/5.4.241-160.348.amzn2int.x86_64
 exe/x86_64.amzn.2 prompt/off command/verifiedpermissions.is-authorized",
    "requestParameters": {
        "principal": {
            "entityType": "PhotoFlash::User",
            "entityId": "alice"
        },
        "action": {
            "actionType": "PhotoFlash::Action",
            "actionId": "ViewPhoto"
        },
        "resource": {
            "entityType": "PhotoFlash::Photo",
```

```
            "entityId": "VacationPhoto94.jpg"
        },
        "policyStoreId": "PSEXAMPLEabcdefg111111"
    },
    "responseElements": null,
    "additionalEventData": {
        "decision": "ALLOW"
    },
    "requestID": "346c4b6a-d12f-46b6-bc06-6c857bd3b28e",
    "eventID": "8a4fed32-9605-45dd-a09a-5ebbf0715bbc",
    "readOnly": true,
    "resources": [
        {
    "accountId": "123456789012",
    "type": "AWS::VerifiedPermissions::PolicyStore",
    "ARN": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "123456789012",
    "eventCategory": "Data"
}
```

## BatchIsAuthorized

```
{
    "eventVersion": "1.08",
    "userIdentity": {
  "type": "AssumedRole",
  "principalId": "EXAMPLE_PRINCIPAL_ID",
  "arn": "arn:aws:iam::123456789012:role/ExampleRole",
  "accountId": "123456789012",
  "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
    },
    "eventTime": "2023-11-20T23:02:33Z",
    "eventSource": "verifiedpermissions.amazonaws.com",
    "eventName": "BatchIsAuthorized",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "203.0.113.0",
    "userAgent": "aws-cli/2.11.18 Python/3.11.3 Linux/5.4.241-160.348.amzn2int.x86_64
 exe/x86_64.amzn.2 prompt/off command/verifiedpermissions.is-authorized",
```

```
    "requestParameters": {
        "requests": [
            {
                "principal": {
                    "entityType": "PhotoFlash::User",
                    "entityId": "alice"
                },
                "action": {
                    "actionType": "PhotoFlash::Action",
                    "actionId": "ViewPhoto"
                },
                "resource": {
                    "entityType": "PhotoFlash::Photo",
                    "entityId": "VacationPhoto94.jpg"
                }
            },
            {
                "principal": {
                    "entityType": "PhotoFlash::User",
                    "entityId": "annalisa"
                },
                "action": {
                    "actionType": "PhotoFlash::Action",
                    "actionId": "DeletePhoto"
                },
                "resource": {
                    "entityType": "PhotoFlash::Photo",
                    "entityId": "VacationPhoto94.jpg"
                }
            }
        ],
        "policyStoreId": "PSEXAMPLEabcdefg111111"
    },
    "responseElements": null,
    "additionalEventData": {
        "results": [
            {
                "request": {
                    "principal": {
                        "entityType": "PhotoFlash::User",
                        "entityId": "alice"
                    },
                    "action": {
                        "actionType": "PhotoFlash::Action",
```

```
                        "actionId": "ViewPhoto"
                    },
                    "resource": {
                        "entityType": "PhotoFlash::Photo",
                        "entityId": "VacationPhoto94.jpg"
                    }
                },
                "decision": "ALLOW"
            },
            {
                "request": {
                    "principal": {
                        "entityType": "PhotoFlash::User",
                        "entityId": "annalisa"
                    },
                    "action": {
                        "actionType": "PhotoFlash::Action",
                        "actionId": "DeletePhoto"
                    },
                    "resource": {
                        "entityType": "PhotoFlash::Photo",
                        "entityId": "VacationPhoto94.jpg"
                    }
                },
                "decision": "DENY"
            }
        ]
    },
    "requestID": "a8a5caf3-78bd-4139-924c-7101a8339c3b",
    "eventID": "7d81232f-f3d1-4102-b9c9-15157c70487b",
    "readOnly": true,
    "resources": [
        {
    "accountId": "123456789012",
    "type": "AWS::VerifiedPermissions::PolicyStore",
    "ARN": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "123456789012",
    "eventCategory": "Data"
```

```
}
```

## CreatePolicyStore

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::123456789012:role/ExampleRole",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  },
  "eventTime": "2023-05-22T07:43:33Z",
  "eventSource": "verifiedpermissions.amazonaws.com",
  "eventName": "CreatePolicyStore",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
  "requestParameters": {
    "clientToken": "a1b2c3d4-e5f6-a1b2-c3d4-TOKEN1111111",
    "validationSettings": {
      "mode": "OFF"
    }
  },
  "responseElements": {
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "arn": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111",
    "createdDate": "2023-05-22T07:43:33.962794Z",
    "lastUpdatedDate": "2023-05-22T07:43:33.962794Z"
  },
  "requestID": "1dd9360e-e2dc-4554-ab65-b46d2cf45c29",
  "eventID": "b6edaeee-3584-4b4e-a48e-311de46d7532",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management"
}
```

## ListPolicyStores


```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::123456789012:role/ExampleRole",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  },
  "eventTime": "2023-05-22T07:43:33Z",
  "eventSource": "verifiedpermissions.amazonaws.com",
  "eventName": "ListPolicyStores",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
  "requestParameters": {
    "maxResults": 10
  },
  "responseElements": null,
  "requestID": "5ef238db-9f87-4f37-ab7b-6cf0ba5df891",
  "eventID": "b0430fb0-12c3-4cca-8d05-84c37f99c51f",
  "readOnly": true,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management"
}
```


## DeletePolicyStore


```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::123456789012:role/ExampleRole",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  },
  "eventTime": "2023-05-22T07:43:32Z",
  "eventSource": "verifiedpermissions.amazonaws.com",
```

```
  "eventName": "DeletePolicyStore",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
  "requestParameters": {
    "policyStoreId": "PSEXAMPLEabcdefg111111"
  },
  "responseElements": null,
  "requestID": "1368e8f9-130d-45a5-b96d-99097ca3077f",
  "eventID": "ac482022-b2f6-4069-879a-dd509123d8d7",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::VerifiedPermissions::PolicyStore",
      "arn": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management"
}
```


## PutSchema


```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::123456789012:role/ExampleRole",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  },
  "eventTime": "2023-05-16T12:58:57Z",
  "eventSource": "verifiedpermissions.amazonaws.com",
  "eventName": "PutSchema",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
  "requestParameters": {
```

```
    "policyStoreId": "PSEXAMPLEabcdefg111111"
  },
  "responseElements": {
    "lastUpdatedDate": "2023-05-16T12:58:57.513442Z",
    "namespaces": "[some_namespace]",
    "createdDate": "2023-05-16T12:58:57.513442Z",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
  },
  "requestID": "631fbfa1-a959-4988-b9f8-f1a43ff5df0d",
  "eventID": "7cd0c677-733f-4602-bc03-248bae581fe5",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::VerifiedPermissions::PolicyStore",
      "ARN": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management"
}
```


## GetSchema


```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::222222222222:role/ExampleRole",
    "accountId": "222222222222",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  },
  "eventTime": "2023-05-25T01:12:07Z",
  "eventSource": "verifiedpermissions.amazonaws.com",
  "eventName": "GetSchema",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
  "requestParameters": {
```

```
    "policyStoreId": "PSEXAMPLEabcdefg111111"
  },
  "responseElements": null,
  "requestID": "a1f4d4cd-6156-480a-a9b8-e85a71dcc7c2",
  "eventID": "0b3b8e3d-155c-46f3-a303-7e9e8b5f606b",
  "readOnly": true,
  "resources": [
    {
      "accountId": "222222222222",
      "type": "AWS::VerifiedPermissions::PolicyStore",
      "ARN": "arn:aws:verifiedpermissions::222222222222:policy-store/
PSEXAMPLEabcdefg111111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "222222222222",
  "eventCategory": "Management"
}
```

## CreatePolicyTemplate

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::123456789012:role/ExampleRole",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  },
  "eventTime": "2023-05-16T13:00:24Z",
  "eventSource": "verifiedpermissions.amazonaws.com",
  "eventName": "CreatePolicyTemplate",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
  "requestParameters": {
    "policyStoreId": "PSEXAMPLEabcdefg111111"
  },
  "responseElements": {
    "lastUpdatedDate": "2023-05-16T13:00:23.444404Z",
    "createdDate": "2023-05-16T13:00:23.444404Z",
```

```
    "policyTemplateId": "PTEXAMPLEabcdefg111111",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
  },
  "requestID": "73953bda-af5e-4854-afe2-7660b492a6d0",
  "eventID": "7425de77-ed84-4f91-a4b9-b669181cc57b",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::VerifiedPermissions::PolicyStore",
      "arn": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management"
}
```

## DeletePolicyTemplate

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::222222222222:role/ExampleRole",
    "accountId": "222222222222",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  },
  "eventTime": "2023-05-25T01:11:48Z",
  "eventSource": "verifiedpermissions.amazonaws.com",
  "eventName": "DeletePolicyTemplate",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
  "requestParameters": {
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "policyTemplateId": "PTEXAMPLEabcdefg111111"
  },
  "responseElements": null,
  "requestID": "5ff0f22e-6bbd-4b85-a400-4fb74aa05dc6",
```

```
  "eventID": "c0e0c689-369e-4e95-a9cd-8de113d47ffa",
  "readOnly": false,
  "resources": [
    {
      "accountId": "222222222222",
      "type": "AWS::VerifiedPermissions::PolicyStore",
      "ARN": "arn:aws:verifiedpermissions::222222222222:policy-store/
PSEXAMPLEabcdefg111111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "222222222222",
  "eventCategory": "Management"
}
```

## CreatePolicy

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::123456789012:role/ExampleRole",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  },
  "eventTime": "2023-05-22T07:42:30Z",
  "eventSource": "verifiedpermissions.amazonaws.com",
  "eventName": "CreatePolicy",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
  "requestParameters": {
    "clientToken": "a1b2c3d4-e5f6-a1b2-c3d4-TOKEN1111111",
    "policyStoreId": "PSEXAMPLEabcdefg111111"
  },
  "responseElements": {
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "policyId": "SPEXAMPLEabcdefg111111",
    "policyType": "STATIC",
    "principal": {
        "entityType": "PhotoApp::Role",
```

```
        "entityId": "PhotoJudge"
    },
    "resource": {
        "entityType": "PhotoApp::Application",
        "entityId": "PhotoApp"
    },
    "lastUpdatedDate": "2023-05-22T07:42:30.70852Z",
    "createdDate": "2023-05-22T07:42:30.70852Z"
  },
  "requestID": "93ffa151-3841-4960-9af6-30a7f817ef93",
  "eventID": "30ab405f-3dff-43ff-8af9-f513829e8bde",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::VerifiedPermissions::PolicyStore",
      "arn": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management"
}
```

## GetPolicy

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::123456789012:role/ExampleRole",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  },
  "eventTime": "2023-05-22T07:43:29Z",
  "eventSource": "verifiedpermissions.amazonaws.com",
  "eventName": "GetPolicy",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
```

```
  "requestParameters": {
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "policyId": "SPEXAMPLEabcdefg111111"
  },
  "responseElements": null,
  "requestID": "23022a9e-2f5c-4dac-b653-59e6987f2fac",
  "eventID": "9b4d5037-bafa-4d57-b197-f46af83fc684",
  "readOnly": true,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::VerifiedPermissions::PolicyStore",
      "arn": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management"
}
```


## CreateIdentitySource


```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::333333333333:role/ExampleRole",
    "accountId": "333333333333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  },
  "eventTime": "2023-05-19T01:27:44Z",
  "eventSource": "verifiedpermissions.amazonaws.com",
  "eventName": "CreateIdentitySource",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
  "requestParameters": {
    "clientToken": "a1b2c3d4-e5f6-a1b2-c3d4-TOKEN1111111",
    "configuration": {
      "cognitoUserPoolConfiguration": {
```

```
        "userPoolArn": "arn:aws:cognito-idp:000011112222:us-east-1:userpool/us-
east-1_aaaaaaaaaa"
      }
    },
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "principalEntityType": "User"
  },
  "responseElements": {
    "createdDate": "2023-07-14T15:05:01.599534Z",
    "identitySourceId": "ISEXAMPLEabcdefg111111",
    "lastUpdatedDate": "2023-07-14T15:05:01.599534Z",
    "policyStoreId": "PSEXAMPLEabcdefg111111"
    },
  "requestID": "afcc1e67-d5a4-4a9b-a74c-cdc2f719391c",
  "eventID": "f13a41dc-4496-4517-aeb8-a389eb379860",
  "readOnly": false,
  "resources": [
    {
      "accountId": "333333333333",
      "type": "AWS::VerifiedPermissions::PolicyStore",
      "arn": "arn:aws:verifiedpermissions::333333333333:policy-store/
PSEXAMPLEabcdefg111111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "333333333333",
  "eventCategory": "Management"
}
```

## GetIdentitySource

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::333333333333:role/ExampleRole",
    "accountId": "333333333333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  },
  "eventTime": "2023-05-24T19:55:31Z",
  "eventSource": "verifiedpermissions.amazonaws.com",
```

```
  "eventName": "GetIdentitySource",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
  "requestParameters": {
    "identitySourceId": "ISEXAMPLEabcdefg111111",
    "policyStoreId": "PSEXAMPLEabcdefg111111"
  },
  "responseElements": null,
  "requestID": "7a6ecf79-c489-4516-bb57-9ded970279c9",
  "eventID": "fa158e6c-f705-4a15-a731-2cdb4bd9a427",
  "readOnly": true,
  "resources": [
    {
      "accountId": "333333333333",
      "type": "AWS::VerifiedPermissions::PolicyStore",
      "arn": "arn:aws:verifiedpermissions::333333333333:policy-store/
PSEXAMPLEabcdefg111111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "333333333333",
  "eventCategory": "Management"
}
```

## ListIdentitySources

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::333333333333:role/ExampleRole",
    "accountId": "333333333333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  },
  "eventTime": "2023-05-24T20:05:32Z",
  "eventSource": "verifiedpermissions.amazonaws.com",
  "eventName": "ListIdentitySources",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
```

```
  "requestParameters": {
    "policyStoreId": "PSEXAMPLEabcdefg111111"
  },
  "responseElements": null,
  "requestID": "95d2a7bc-7e9a-4efe-918e-97e558aacaf7",
  "eventID": "d3dc53f6-1432-40c8-9d1d-b9eeb75c6193",
  "readOnly": true,
  "resources": [
    {
      "accountId": "333333333333",
      "type": "AWS::VerifiedPermissions::PolicyStore",
      "arn": "arn:aws:verifiedpermissions::333333333333:policy-store/
PSEXAMPLEabcdefg111111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "333333333333",
  "eventCategory": "Management"
}
```

## DeleteIdentitySource

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::333333333333:role/ExampleRole",
    "accountId": "333333333333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  },
  "eventTime": "2023-05-24T19:55:32Z",
  "eventSource": "verifiedpermissions.amazonaws.com",
  "eventName": "DeleteIdentitySource",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
  "requestParameters": {
    "identitySourceId": "ISEXAMPLEabcdefg111111",
    "policyStoreId": "PSEXAMPLEabcdefg111111"
  },
  "responseElements": null,
```

```
  "requestID": "d554d964-0957-4834-a421-c417bd293086",
  "eventID": "fe4d867c-88ee-4e5d-8d30-2fbc208c9260",
  "readOnly": false,
  "resources": [
    {
      "accountId": "333333333333",
      "type": "AWS::VerifiedPermissions::PolicyStore",
      "arn": "arn:aws:verifiedpermissions::333333333333:policy-store/
PSEXAMPLEabcdefg111111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "333333333333",
  "eventCategory": "Management"
}
```

# 使用 AWS CloudFormation创建 Amazon Verified Permissions 资源

Amazon Verified Permissions 与 AWS CloudFormation一项服务集成，可帮助您对 AWS 资源进行建模和设置，从而减少创建和管理资源和基础设施所花费的时间。您可以创建一个描述所需的所有 AWS 资源（例如策略存储）的模板，并为您 AWS CloudFormation 预置和配置这些资源。

使用时 AWS CloudFormation，您可以重复使用您的模板来一致且重复地设置您的已验证权限资源。只需描述一次您的资源，然后在多个 AWS 账户 区域中一遍又一遍地配置相同的资源。

> **Important**
>
> Amazon Cognito Identity 的可用性与 AWS 区域 亚马逊验证权限完全不同。如果您收到 AWS CloudFormation 有关亚马逊 Cognito Identity 的错误，例如`Unrecognized resource types: AWS::Cognito::UserPool, AWS::Cognito::UserPoolClient`，我们建议您在地理位置最接近可用 Amazon Cognito 身份的地方创建 Amaz AWS 区域 on Cognito 用户池和客户端。创建 Verified Permissions 身份来源时，请使用这个新创建的用户群体。

## 已验证的权限和 AWS CloudFormation 模板

要为 Verified Permissions 和相关服务预置和配置资源，您必须了解 AWS CloudFormation 模板。模板是 JSON 或 YAML 格式的文本文件。这些模板描述了您要在 AWS CloudFormation 堆栈中配置的资源。如果你不熟悉 JSON 或 YAML，可以使用 D AWS CloudFormation esigner 来帮助你开始使用 AWS CloudFormation 模板。有关更多信息，请参阅什么是 AWS CloudFormation 设计器？ 在《AWS CloudFormation 用户指南》中。

Verified Permissions 支持在中创建身份源、策略、策略存储和策略模板 AWS CloudFormation。有关更多信息（包括 Verified Permissions 资源的 JSON 和 YAML 模板示例），请参阅《AWS CloudFormation 用户指南》中的 Amazon Verified Permissions 资源类型参考。

## AWS CDK 构造

AWS Cloud Development Kit (AWS CDK) 是一个开源软件开发框架，用于在代码中定义云基础架构并通过它进行配置 AWS CloudFormation。构造或可重复使用的云组件可用于创建 AWS CloudFormation 模板。然后，这些模板可用于部署您的云基础架构。

要了解更多信息并下载 AWS CDK，请参阅 C AWS loud Development Kit。

以下是已验证权限 AWS CDK 资源（例如构造）的文档链接。

- Amazon 已验证权限 L2 CDK 构造

# 了解更多关于 AWS CloudFormation

要了解更多信息 AWS CloudFormation，请参阅以下资源：

- AWS CloudFormation
- AWS CloudFormation 用户指南
- AWS CloudFormation API 引用
- AWS CloudFormation 命令行界面用户指南

# 使用接口端点（AWS PrivateLink）访问 Amazon Verified Permissions

您可以使用 AWS PrivateLink 在您的 VPC 和 Amazon Verified Permissions 之间创建私有连接。您可以像在 VPC 中一样访问 Verified Permissions，而无需使用互联网网关、NAT 设备、VPN 连接或 AWS Direct Connect 连接。VPC 中的实例不需要公有 IP 地址即可访问 Verified Permissions。

您可以通过创建由 AWS PrivateLink 提供支持的接口端点来建立此私有连接。我们将在您为接口端点启用的每个子网中创建一个端点网络接口。这些是请求者托管的网络接口，用作发往 Verified Permissions 的流量的入口点。

有关更多信息，请参阅《AWS PrivateLink 指南》中的通过 AWS PrivateLink 访问 AWS 服务。

## Verified Permissions 注意事项

在为 Verified Permissions 设置接口端点之前，请首先查看《AWS PrivateLink 指南》中的注意事项。

Verified Permissions 支持通过接口端点调用其所有 API 操作。

Verified Permissions 不支持 VPC 端点策略。默认情况下，允许通过接口端点对 Verified Permissions 进行完全访问。或者，您可以将安全组与端点网络接口关联，以控制通过接口端点流向 Verified Permissions 的流量。

## 为 Verified Permissions 创建接口端点

您可以使用 Amazon VPC 控制台或 AWS Command Line Interface（AWS CLI）为 Verified Permissions 创建接口端点。有关更多信息，请参阅《AWS PrivateLink 指南》中的创建接口端点。

使用以下服务名称为 Verified Permissions 创建接口端点：

```
com.amazonaws.region.verifiedpermissions
```

如果为接口端点启用私有 DNS，则可使用区域默认 DNS 名称向 Verified Permissions 发出 API 请求。例如，`verifiedpermissions.us-east-1.amazonaws.com`。

# Amazon Verified Permissions 的配额

您的每项 AWS 服务 AWS 账户 都有默认配额，以前称为限制。除非另有说明，否则，每个限额是区域特定的。您可以请求增加某些配额，但其他一些配额无法增加。

要查看 Verified Permissions 的配额，请打开 Service Quotas 控制台。在导航窗格中，选择 AWS 服务，然后选择 Verified Permissions。

要请求提高配额，请参阅《Service Quotas 用户指南》中的请求提高配额。如果配额在服务限额中尚不可用，请使用提高限制表格。

您 AWS 账户 有以下与已验证权限相关的配额。

主题



## 资源配额

| 名称 | 默认值 | 可调整 | 描述 |
| --- | --- | --- | --- |
| 每个账户在每个区域的策略存储数 | 每个受支持的区域：1,000 个 | 是 | 策略存储的最大数量。 |
| 每个策略存储的策略模板 | 每个受支持的区域：40 个 | 是 | 一个策略存储中策略模板的最大数量。 |
| 每个策略存储的身份来源数 | 1 | 否 | 您可以为一个策略存储定义的身份来源最大数量。 |
| 授权请求大小[1] | 1MB | 否 | 授权请求的最大大小。 |
| 保单规模 | 10000 字节 | 否 | 单个策略的最大大小。 |

| 名称 | 默认值 | 可调整 | 描述 |
| --- | --- | --- | --- |
| 架构大小 | 100000 字节 | 否 | 策略存储区架构的最大大小。 |
| 每个资源的策略大小 | 200,000 个字节[2] | 否 | 引用特定资源的所有策略的最大大小。 |

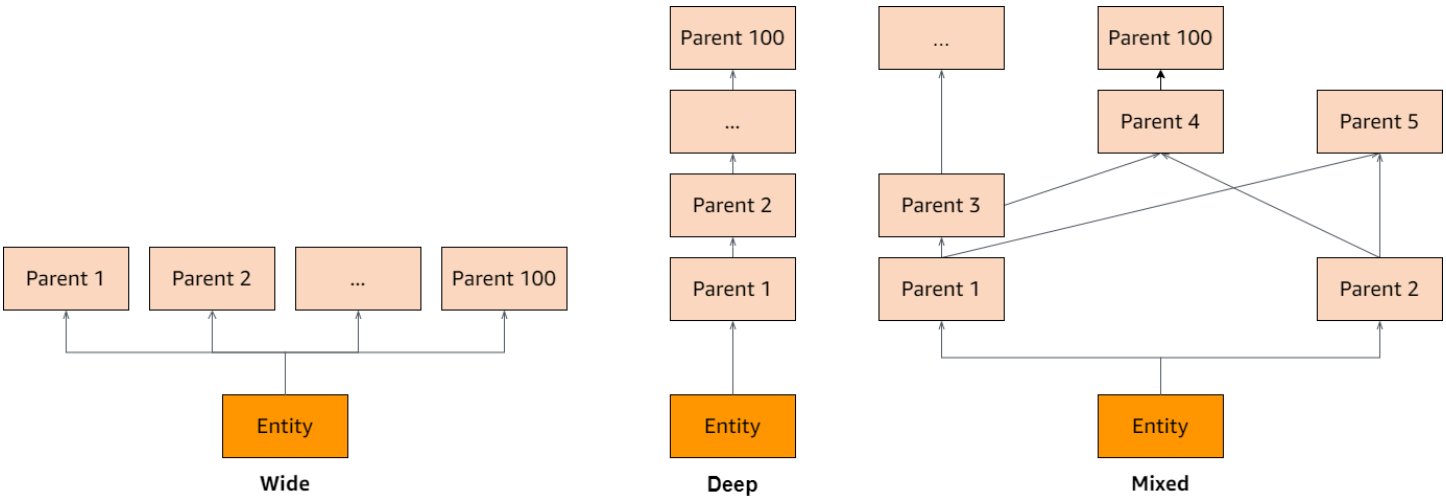
[1] IsAuthorized和的授权请求配额相同IsAuthorizedWithToken。

[2] 与单个资源相关的所有策略的总大小不能超过 200,000 字节。在计算模板链接策略的大小时，策略模板的大小仅计算一次，再加上用于实例化每个模板链接策略的每组参数的大小。

## 层次结构的配额

| 名称 | 默认值 | 可调整 | 描述 |
| --- | --- | --- | --- |
| 每个主体的可传递父项数 | 100 | 否 | 每个主体可传递父项的最大数量。 |
| 每项操作可传递的父项数 | 100 | 否 | 每项操作可传递父项的最大数量。 |
| 每个资源的可传递父项数 | 100 | 否 | 每个资源可传递父项的最大数量。 |

下图说明了如何为实体（主体、操作或资源）定义可传递父项。




# 每秒操作配额

AWS 区域 当应用程序请求超过 API 操作的配额时，经过验证的权限会限制对服务端点的请求。当您超过每秒请求数的配额或尝试同步写入操作时，已验证权限可能会返回异常。您可以在 Service Quotas 中查看您当前的 RPS 配额。要防止应用程序超出某项操作的配额，您必须针对重试和指数级退避对其进行优化。有关更多信息，请参阅使用退避模式重试和管理和监控工作负载中的 API 限制。

| 名称 | 默认值 | 可调整 | 描述 |
| --- | --- | --- | --- |
| BatchIsAuthorized 每个账户每个区域的每秒请求数 | 每个受支持的区域：30 个 | 是 | 每秒的最大 BatchIsAuthorized 请求数。 |
| BatchIsAuthorizedWithToken 每个账户每个区域的每秒请求数 | 每个受支持的区域：30 个 | 是 | 每秒的最大 BatchIsAuthorizedWithToken 请求数。 |
| CreatePolicy 每个账户每个区域的每秒请求数 | 每个受支持的区域：10 个 | 是 | 每秒的最大 CreatePolicy 请求数。 |
| CreatePolicyStore 每个账户每个区域的每秒请求数 | 每个受支持的区域：1 个 | 否 | 每秒的最大 CreatePolicyStore 请求数。 |
| CreatePolicyTemplate 每个账户每个区域的每秒请求数 | 每个受支持的区域：10 个 | 是 | 每秒的最大 CreatePolicyTemplate 请求数。 |

| 名称 | 默认值 | 可调整 | 描述 |
| --- | --- | --- | --- |
| DeletePolicy 每个账户每个区域的每秒请求数 | 每个受支持的区域：10 个 | 是 | 每秒的最大 DeletePolicy 请求数。 |
| DeletePolicyStore 每个账户每个区域的每秒请求数 | 每个受支持的区域：1 个 | 否 | 每秒的最大 DeletePolicyStore 请求数。 |
| DeletePolicyTemplate 每个账户每个区域的每秒请求数 | 每个受支持的区域：10 个 | 是 | 每秒的最大 DeletePolicyTemplate 请求数。 |
| GetPolicy 每个账户每个区域的每秒请求数 | 每个受支持的区域：10 个 | 是 | 每秒的最大 GetPolicy 请求数。 |
| GetPolicyTemplate 每个账户每个区域的每秒请求数 | 每个受支持的区域：10 个 | 是 | 每秒的最大 GetPolicyTemplate 请求数。 |
| GetSchema 每个账户每个区域的每秒请求数 | 每个受支持的区域：10 个 | 是 | 每秒的最大 GetSchema 请求数。 |
| IsAuthorized 每个账户每个区域的每秒请求数 | 每个受支持的区域：200 个 | 是 | 每秒的最大 IsAuthorized 请求数。 |
| IsAuthorizedWithToken 每个账户每个区域的每秒请求数 | 每个受支持的区域：200 个 | 是 | 每秒的最大 IsAuthorizedWithToken 请求数。 |
| ListPolicies 每个账户每个区域的每秒请求数 | 每个受支持的区域：10 个 | 是 | 每秒的最大 ListPolicies 请求数。 |
| ListPolicyStores 每个账户每个区域的每秒请求数 | 每个受支持的区域：10 个 | 是 | 每秒的最大 ListPolicyStores 请求数。 |
| ListPolicyTemplates 每个账户每个区域的每秒请求数 | 每个受支持的区域：10 个 | 是 | 每秒的最大 ListPolicyTemplates 请求数。 |
| PutSchema 每个账户每个区域的每秒请求数 | 每个受支持的区域：10 个 | 是 | 每秒的最大 PutSchema 请求数。 |

| 名称 | 默认值 | 可调整 | 描述 |
| --- | --- | --- | --- |
| UpdatePolicy 每个账户每个区域的每秒请求数 | 每个受支持的区域：10 个 | 是 | 每秒的最大 UpdatePolicy 请求数。 |
| UpdatePolicyTemplate 每个账户每个区域的每秒请求数 | 每个受支持的区域：10 个 | 是 | 每秒的最大 UpdatePolicyTemplate 请求数。 |

# 《Amazon Verified Permissions 用户指南》的文档历史记录

下表介绍了 Verified Permissions 的文档版本。

| 变更 | 说明 | 日期 |
| --- | --- | --- |
| OIDC 身份来源 | 现在，您可以对 OpenID Connect (OIDC) 身份提供商的用户进行授权。 | 2024年6月8日 |
| 使用身份源令牌进行批量授权 | 现在，您可以通过单个 `BatchIsAuthorizedWithToken` API 请求对 Amazon Cognito 用户池中的用户进行授权。 | 2024 年 4 月 5 日 |
| 使用 API Gateway 创建策略存储 | 现在，您可以从现有 API 和 Amazon Cognito 用户池中创建策略存储。 | 2024 年 4 月 1 日 |
| 上下文概念和示例 | 添加了有关使用已验证权限的授权请求中的上下文的信息。 | 2024年2月1日 |
| 授权概念和示例 | 添加了有关使用已验证权限的授权请求的信息。 | 2024年2月1日 |
| AWS CloudFormation 整合 | Verified Permissions 支持在中创建身份源、策略、策略存储和策略模板 AWS CloudFormation。 | 2023 年 6 月 30 日 |
| 初始版本 | 《Amazon Verified Permissions 用户指南》首次发布 | 2023 年 6 月 13 日 |

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。