亚马逊云科技

用户指南

Amazon Verified Permissions

Amazon Verified Permissions: 用户指南

Amazon Web Services 文档中描述的 Amazon Web Services 服务或功能可能因区域而异。要查看适用于中国区域的差异，请参阅 中国的 Amazon Web Services 服务入门 (PDF)。

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 策略语言来定义精细的权限，以保护应用程序的资源。

有关使用已验证权限设置政策决策点 (PDP) 的指导和示例，请参阅Amazon 规范性指南中的使用亚马逊验证权限实施 PDP。

主题

使用 Verified Permissions 进行授权

Verified Permissions 通过验证是否允许委托人在应用程序的给定上下文中对资源执行操作来提供授权。已验证权限假设委托人事先已通过其他方式进行识别和身份验证，例如使用 OpenID Connect 等协议、托管 Amazon Cognito提供商（例如）或其他身份验证解决方案。已验证的权限与委托人的管理位置和身份验证方式无关。

Verified Permissions 是一项服务，它使客户能够在中创建 Amazon Web Services 管理控制台、维护和测试策略，使用经过验证的权限 APIs以编程方式或通过基础设施即代码解决方案（例如 Amazon CloudFormation）。权限是使用 Cedar 策略语言来表达的。客户端应用程序调用授权 APIs 来评估存储在服务中的 Cedar 策略，并提供是否允许操作的访问决策。

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委托人（人类用户或计算机）才能访问该应用程序，并且只能执行他们有权执行的操作。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 4.7。

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。

Verified Permissions 提供与流行开发框架的集成，因此只需最少的代码更改即可更轻松地在应用程序中实现授权。这些集成使您可以专注于核心业务逻辑，而经过验证的权限则负责处理授权决策。

- Express.js — 基于中间件的集成，使您无需修改现有路由处理程序即可保护 Express 应用程序中的 API 端点。有关更多信息，请参阅 the section called “使用快递”。

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。它是一个用户目录、一个身份验证服务器以及一个用于 OAuth 2.0 访问令牌和 Amazon 凭据的授权服务。创建策略存储时，您可以选择从 Amazon Cognito 用户池中构建委托人和群组。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、 WebSocket APIs HTTP 的 Amazon 服务。创建策略存储库时，您可以选择通过中的 API 构建操作和资源 API Gateway。有关的更多信息 API Gateway，请参阅《API Gateway 开发人员指南》。
- Amazon 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 Amazon Web Services 账户 和应用程序的访问权限。有关更多信息，请参阅 Amazon 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。

Amazon Web Services 管理控制台

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

- Amazon 已验证权限控制台

Amazon 命令行工具

您可以使用 Amazon 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 Amazon 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 Amazon 任务的脚本，命令行工具也会十分有用。

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

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

Amazon SDKs

Amazon 提供 SDKs（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。 SDKs 提供了一种便捷的方法来创建对已验证权限的编程访问权限和 Amazon. 例如，负责处理诸如对请求 SDKs 进行加密签名、管理错误和自动重试请求之类的任务。

要了解更多信息并进行下载 Amazon SDKs，请参阅工具 Amazon Web Services。

以下是各种已验证权限资源的文档链接 Amazon SDKs。

- 适用于 .NET 的 Amazon SDK
- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java 的 Amazon SDK
- 适用于 JavaScript 的 Amazon SDK
- 适用于 PHP 的 Amazon SDK
- Amazon SDK for Python (Boto)
- 适用于 Ruby 的 Amazon SDK
- 适用于 Rust 的 Amazon SDK

Amazon CDK 构造

Amazon Cloud Development Kit (Amazon 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 Amazon CloudFormation。构造或可重复使用的云组件可用于创建 Amazon 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 Amazon CDK，请参阅 C Amazon loud Development Kit。

以下是已验证权限 Amazon 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 Amazon 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 Amazon 账单与成本管理 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 Amazon Web Services 账户 计费的更多信息，请参阅Amazon Billing 用户指南。

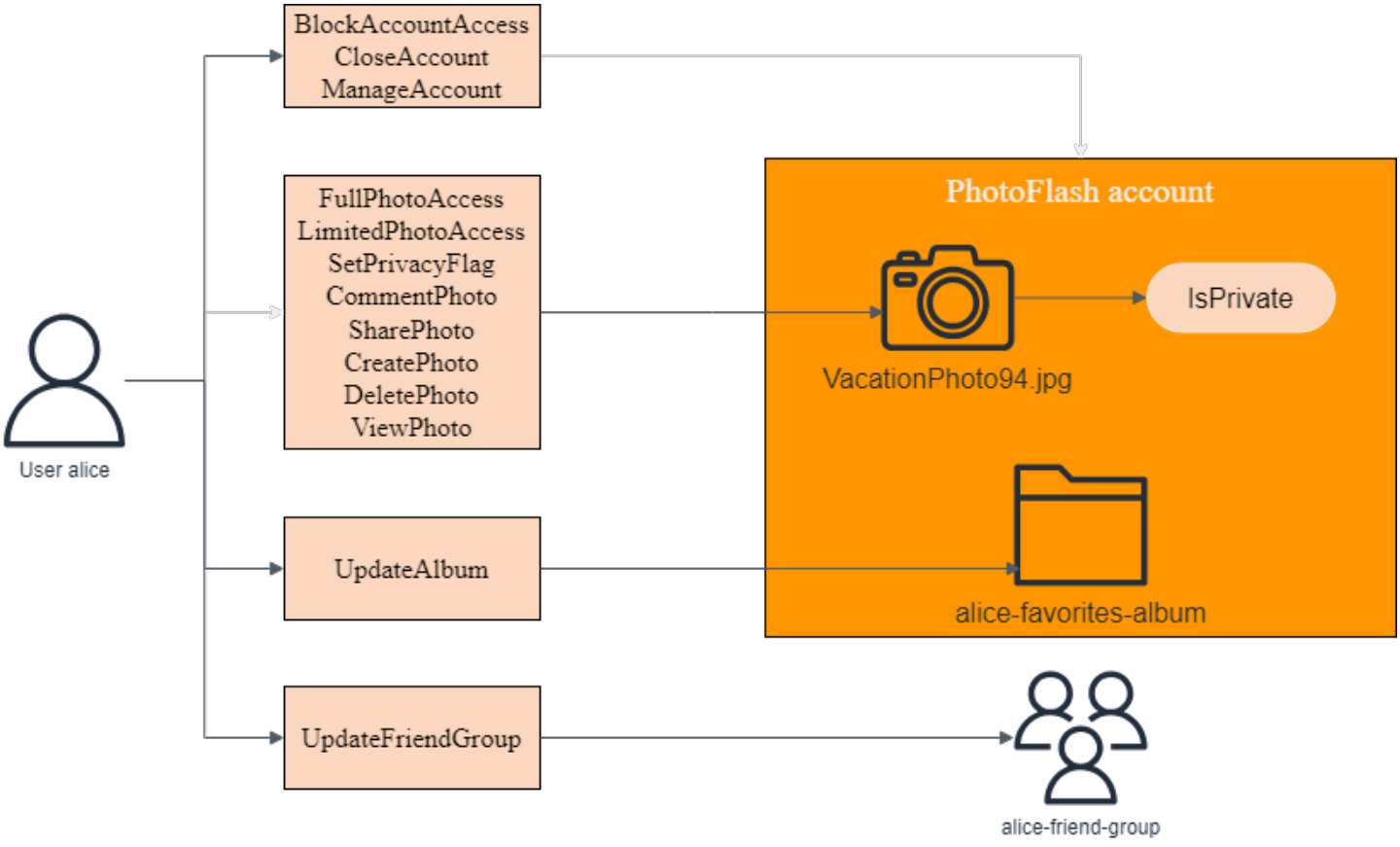
如果您对 Amazon 账单、账号和活动有疑问，请联系 Amazon Web Services 支持。

创建您的第一个 Amazon 认证权限策略商店

在本教程中，假设您是照片共享应用程序的开发人员，并且您正在寻找一种方法来控制该应用程序的用户可以执行的操作。你想控制谁可以添加、删除或查看照片和相册。您还想控制用户可以对其账户执行哪些操作。他们能管理自己的账户吗，朋友的账户怎么样？ 要控制这些操作，您需要根据用户的身份创建允许或禁止这些操作的策略。Verified Permissions 提供策略存储库或容器来存放这些政策。

在本教程中，我们将介绍如何使用 Amazon Verified Permissions 控制台创建示例策略存储。控制台提供了一些示例策略存储选项，我们将创建一个PhotoFlash策略存储。此策略存储允许委托人（例如用户）对照片或相册等资源执行操作，例如共享。

下图说明了委托人与她可以对各种资源（即她的 PhotoFlash 账户、`VactionPhoto94.jpg`文件、相册`alice-favorites-album`和用户组）采取的操作之间的关系`alice-friend-group`。`User::alice`



现在您已经了解了PhotoFlash策略存储，接下来让我们创建策略存储并对其进行探索。

先决条件

注册获取 Amazon Web Services 账户

如果您没有 Amazon Web Services 账户，请完成以下步骤来创建一个。

报名参加 Amazon Web Services 账户

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

   在注册时，将接到电话或收到短信，要求使用电话键盘输入一个验证码。

   当您注册时 Amazon Web Services 账户，就会创建Amazon Web Services 账户根用户一个。根用户有权访问该账户中的所有 Amazon Web Services 服务 和资源。作为最佳安全实践，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。

Amazon 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 My Account (我的账户) 来查看当前的账户活动并管理您的账户。

保护 IAM 用户

注册后 Amazon Web Services 账户，开启多重身份验证 (MFA)，保护您的管理用户。有关说明，请参阅《IAM 用户指南》中的为 IAM 用户启用虚拟 MFA 设备（控制台）。

要允许其他用户访问您的 Amazon Web Services 账户 资源，请创建 IAM 用户。为了保护您的 IAM 用户，请启用 MFA 并仅向 IAM 用户授予执行任务所需的权限。

有关创建和保护 IAM 用户的更多信息，请参阅《IAM 用户指南》中的以下主题：

- 在你的 IAM 用户中创建 Amazon Web Services 账户
- 适用于 Amazon 资源的访问权限管理
- IAM 基于身份的策略示例

步骤 1：创建 PhotoFlash 策略存储

在以下步骤中，您将使用 Amazon 控制台创建PhotoFlash策略存储。

创建 PhotoFlash 策略存储

1. 在已验证的权限控制台中，选择创建新的策略存储。
2. 对于 “启动” 选项，选择 “从示例策略存储开始”。
3. 对于示例项目，请选择PhotoFlash。
4. 选择创建策略存储。

看到 “已创建并配置策略存储” 消息后，选择 “转至概览” 以浏览您的策略存储。

步骤 2：创建策略

创建策略存储时，创建了一个默认策略，允许用户完全控制自己的账户。这是一项有用的政策，但出于我们的目的，让我们创建一个更具限制性的策略来探讨已验证权限的细微差别。如果你还记得我们在本教程前面看过的图表`User::alice`，我们有一个委托人`UpdateAlbum`，他可以对资源执行操作`alice-favorites-album`。让我们添加一项政策，允许爱丽丝（且只有 Alice）管理这张专辑。

创建策略

1. 在已验证权限控制台中，选择您在步骤 1 中创建的策略存储。
2. 在导航栏中，选择策略。
3. 选择创建策略，然后选择创建静态策略。
4. 对于策略效果，请选择 “允许”。
5. 对于 “委托人范围”，选择 “特定委托人”，然后在 “指定实体类型” 中选择PhotoFlash:: 用户，在 “指定实体标识符” 中输入。**`alice`**
6. 对于 “资源范围”，选择 “特定资源”，然后在 “指定实体类型” 中选择PhotoFlash:: Album，在 “指定实体标识符” 中输入**`alice-favorites-album`**。
7. 对于 “操作范围”，选择 “特定操作集”，然后在 “此策略应适用的操作” 中选择UpdateAlbum。
8. 选择下一步。
9. 在 “详细信息” 下，在 “策略描述-可选” 中输入**`Policy allowing alice to update alice-favorites-album.`**。
10. 选择创建策略

现在，您已经创建了策略，可以在已验证的权限控制台中对其进行测试。

步骤 3：测试策略存储

创建策略存储库和策略后，您可以使用已验证权限测试平台运行模拟授权请求来对其进行测试。

要测试策略，请存储策略

1. 打开已验证权限控制台。选择您的保单存储。
2. 在左侧导航窗格中，选择测试平台。
3. 选择可视模式。
4. 对于校长，请执行以下操作：
   a. 对于委托人正在采取行动，请选择PhotoFlash:: User，对于指定实体标识符，请输入**`alice`**。
   b. 在 “属性” 下，对于 “帐户：实体”，确保选择PhotoFlash:: Accoun t 实体，然后在 “指定实体标识符” 中输入**`alice-account`**。
5. 在资源下，对于委托人正在操作的资源，选择PhotoFlash:: Al bum 资源类型，在 “指定实体标识符” 中输入**`alice-favorites-album`**。
6. 对于 “操作”，请从有效操作列表中选择PhotoFlash:: Action:: UpdateAlbum “”。
7. 在页面顶部，选择运行授权请求以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 “决定：允许”，表明我们的政策按预期运行。

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 示例策略存储中包含的静态策略和您在步骤 2 中创建的策略的授权请求决定。

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 | b ob	PhotoFlash:: 账户 | alice-account	PhotoFlash:: 相册 | alice-favorites-Al bum	N/A	PhotoFlas h:: 动作::”” UpdateAlbum	拒绝
PhotoFlash:: 用户 | 爱丽丝	PhotoFlash:: 账户 | alice-account	PhotoFlas h:: Photo | photo.jpeg	PhotoFlash:: 账户 | bob-account	PhotoFlas h:: 动作::”” ViewPhoto	拒绝

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 | 爱丽丝	PhotoFlash::账户 | alice-account	PhotoFlash:: Photo | photo.jpeg	PhotoFlash::账户 | alice-account	PhotoFlash:: 动作::"" ViewPhoto	允许
PhotoFlash::用户 | 爱丽丝	PhotoFlash::账户 | alice-account	PhotoFlash::Photo | bob-photo.jpeg	PhotoFlash::相册 | Bob-Vacation-Album	PhotoFlash:: 动作::"" DeletePhoto	拒绝

步骤 4：清理资源

浏览完保单存储库后，将其删除。

删除策略存储

1. 在已验证权限控制台中，选择您在步骤 1 中创建的策略存储。
2. 在导航栏中，选择 “设置”。
3. 在 “删除策略存储” 下，选择 “删除此策略存储”。
4. 在 “删除此政策” 存储区中？ 对话框中，输入 delete，然后选择删除。

设计授权模型的最佳实践

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，你应该从用户体验开始。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？ 它们是如何组织的？ 例如，文件是否位于一个文件夹中？
- 资源的组织是否在权限模型中起作用？
- 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？
- 角色是全球角色还是作用域角色？ 例如，“操作员” 是限制在单个租户内，还是 “操作员” 是指整个应用程序中的操作员？
- 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。相反，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、逻辑图等工件中，以及对权限如何影响用户在产品中可以做的事情的总体描述。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。

要帮助回答问题并得出最优模型，请执行以下操作：

- 在 Cedar 政策语言参考指南中查看 Cedar 的设计模式。
- 考虑 Cedar 政策语言参考指南中的最佳实践。
- 考虑一下本页中包含的最佳实践。

最佳实践

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。

返回 403 个禁止的错误，而不是 404 未找到错误

对于包含与任何策略都不对应的实体（尤其是资源）的请求，最好返回 403 Forbidden 错误，而不是 404 未找到错误。这提供了最高级别的安全性，因为您不会暴露实体是否存在，而只是该请求不符合策略存储中任何策略中的策略条件。

将注意力集中在 API 操作之外的资源上

在大多数应用程序中，权限都是围绕支持的资源建模的。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，随着您进一步开发模型，这种以 API 为中心的方法可能不适合具有非常精细的授权模型的应用程序，因为这 APIs 只是客户真正想要保护的对象（底层数据和资源）的代名词。如果多人 APIs 控制对相同资源的访问权限，则管理员可能很难推断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。

我们建议您重点关注底层数据和资源及其关联操作。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	Permissions
`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。

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，并将其与 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。有关更多信息，请参阅Amazon 规范性指南中的 Amazon 已验证权限多租户设计注意事项。

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。

这两种策略都不会对您的 Amazon 账单产生重大影响。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

当您的应用程序有多个策略存储时，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

您可能需要对每个租户应用一些全局策略。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。当租户在您的组织中不再活跃并且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。有关限额的更多信息，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。必须在策略和授权请求中包含租户标识符。	低。隔离是默认行为。其他租户无法访问特定于租户的政策。
授权流程	低。所有查询都以一个策略存储为目标。	中等。必须维护每个租户与其策略存储 ID 之间的映射。
模板和架构管理	低。必须使一个架构适用于所有租户。	高。架构和模板可能不那么复杂，但更改需要更多的协调和复杂性。
全球政策管理	低。所有政策都是全球性的，可以集中更新。	高。您必须在入职时向每个策略存储区添加全局政策。在多个策略存储库之间复制全局策略更新。
租户离职	高。必须仅识别和删除租户特定的政策。	低。删除策略存储。
服务资源配额	高。租户共享影响策略存储的资源配额，例如架构大小、每个资源的策略大小以及每个策略存储的身份源。	低。每个租户都有专用资源配额。

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。

Amazon Verified Permissions 策略存储

策略存储是策略和策略模板的容器。在每个策略存储中，您可以创建一个架构，用于验证添加到策略存储中的策略。此外，您还可以开启策略验证。如果在启用策略验证的情况下向策略存储中添加策略，则策略中定义的实体类型、常见类型和操作将根据架构进行验证，无效策略将被拒绝。

删除保护可防止意外删除策略存储。通过创建的所有新策略存储都启用了删除保护 Amazon Web Services 管理控制台。相比之下，通过 API 或 SDK 调用创建的所有策略存储都将禁用该功能。

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。您也可以创建策略存储别名，以便通过友好名称来引用您的策略存储。有关更多信息，请参阅 Amazon 已验证权限策略存储别名。

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。例如 `MyApplicationNamespace::exampleType`。已验证的权限支持每个策略存储区最多 100 个命名空间。当你使用多个相似的应用程序时，这些命名空间有助于保持一切顺畅。例如，在多租户应用程序中，使用命名空间将租户名称附加到架构中定义的类型后，将使它们与其他租户使用的类似类型区分开来。在查看授权请求的日志时，您可以轻松识别处理授权请求的租户。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
- 删除策略存储

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主体实体，以及通过 Amazon API Gateway API 进行操作和资源实体。如果您希望您的应用程序使用用户的群组成员资格或其他属性来授权 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

要使用创建策略存储 引导式设置 配置方法

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，作为对当前策略库功能的友好参考，例如天气更新 Web 应用程序。
4. 在详细信息部分中，输入架构的命名空间。有关命名空间的更多信息，请参阅 Cedar 文档中的命名空间。
5. 选择下一步。
6. 在资源类型窗口中，输入资源类型的名称。例如，`currentTemperature`可以是天气更新 Web 应用程序的资源。
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。例如，`temperatureFormat`可以是`currentTemperature`资源的属性，可以是华氏度或摄氏度。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。例如，`viewTemperature`可能是天气更新 Web 应用程序中的一个操作。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。默认情况下，用户已添加到此字段，但可以替换。
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。根据架构验证策略时，Verified Permissions 会使用指定的属性值。要移除已为主体类型添加的属性，请选择该属性旁边的移除。

- 如果委托人的 ID 和属性将由生成的 ID 或访问令牌提供，请选择 Cognito 用户池。 Amazon Cognito选择连接用户群体。选择Amazon Web Services 区域并键入要连接的 Amazon Cognito 用户池的用户池 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。
- 如果委托人的 ID 和属性将从外部 OIDC 提供商生成的 ID and/or 访问令牌中提取，请选择外部 OIDC 提供商，然后添加提供商和令牌详细信息。

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。例如，`user-id`可以是天气更新 Web 应用程序中的实体标识符。

      > **Note**
      >
      > 如果您使用的是 Amazon Cognito，则实体标识符的格式必须为`<userpool-id>|<sub>`。

    - 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。例如，`temperature-id`可以是天气更新 Web 应用程序中的实体标识符。
    - 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

要使用创建策略存储 设置为 API Gateway 和身份来源 配置方法

该 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 “开始选项” 部分中，选择设置方式 API Gateway 和身份源，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，请在 API Gateway 控制台中修改您的 API 路径或方法，然后选择导入 API 以查看更新。
    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 Amazon Web Services 区域 与 Amazon Web Services 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，这是该 API-linked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. 如果您选择外部 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，请参阅将 Amazon Cognito 令牌映射到架构和将 OIDC 令牌映射到架构。

   c. （可选）在令牌声明-可选中，选择添加令牌声明，输入令牌的名称，然后选择值类型。

   d. 在用户和群组令牌声明中，执行以下操作：

      i. 在身份源的令牌中输入用户声明名称。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。

      ii. 在身份源的令牌中输入群组声明名称。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。

   e. 在受众验证中，选择`Add value`并添加您希望策略商店在授权请求中接受的值。

   f. 选择下一步。

7. 如果您选择 Amazon Cognito，“已验证权限” 会查询您的用户池中的群组。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 Depl oy 应用程序集成中，选择是要稍后手动附加 Lambda 授权者，还是想让已验证的权限立即为您执行此操作，并查看已验证权限将为创建策略存储和 Lambda 授权者而采取的步骤。

9. 准备好创建新资源时，选择创建策略存储。

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，如果您选择手动连接授权方，请配置您的授权方。

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 A mazon 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

要使用创建策略存储 示例政策存储 配置方法

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，允许用户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示例策略存储区不包括策略模板。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Amazon 已验证权限示例模板关联政策。

Empty policy store

要使用创建策略存储 空的策略存储 配置方法

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 静态策略和创建与 Amazon 验证权限模板关联的政策。

Amazon CLI

要创建空的策略存储，请使用 Amazon CLI.

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 Amazon 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 Amazon 已验证权限策略模板。

```
$ 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"
}
```

Amazon SDKs

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

使用 Amazon 软件开发工具包在 Rust 中实现亚马逊验证权限

本主题提供了使用 Amazon 软件开发工具包在 Rust 中实现 Amazon 验证权限的实际示例。此示例说明如何开发一种可以测试用户是否能够查看照片的授权模型。示例代码使用来自 aws-sdk-verifiedpermissionscrate 适用于 Rust 的 Amazon SDK，它提供了一组用于与 Amazon Web Services 服务之交互的强大工具。

先决条件

在开始之前，请确保在系统上配置了 Amazon CLI，并且熟悉 Rust。

- 有关安装的说明 Amazon CLI，请参阅 Amazon CLI 安装指南。
- 有关配置的说明 Amazon CLI，请参阅中的配置设置 Amazon CLI和配置和凭据文件设置。 Amazon CLI
- 有关 Rust 的更多信息，请参阅 rust-lang.org 和 Rust Amazon 开发工具包开发人员指南。

准备好环境后，让我们来探讨如何在 Rust 中实现已验证的权限。

测试示例代码

示例代码执行以下操作：

- 设置要与之通信的 SDK 客户端 Amazon
- 创建策略存储
- 通过添加架构来定义策略存储的结构
- 添加用于检查授权请求的策略
- 发送测试授权请求以验证所有设置是否正确

测试示例代码

1. 创建一个 Rust 项目。
2. 将中的`main.rs`任何现有代码替换为以下代码：

```
use std::time::Duration;
use std::thread::sleep;
use aws_config::BehaviorVersion;
use aws_sdk_verifiedpermissions::Client;
use aws_sdk_verifiedpermissions::{
    operation::{
        create_policy::CreatePolicyOutput,
        create_policy_store::CreatePolicyStoreOutput,
        is_authorized::IsAuthorizedOutput,
        put_schema::PutSchemaOutput,
    },
    types::{
        ActionIdentifier, EntityIdentifier, PolicyDefinition, SchemaDefinition,
 StaticPolicyDefinition, ValidationSettings
    },
};

//Function that creates a policy store in the client that's passed
async fn create_policy_store(client: &Client, valid_settings: &ValidationSettings)-
> CreatePolicyStoreOutput {
    let policy_store =
 client.create_policy_store().validation_settings(valid_settings.clone()).send().await;
    return policy_store.unwrap();
}

//Function that adds a schema to the policy store in the client
async fn put_schema(client: &Client, ps_id: &str, schema: &str) -> PutSchemaOutput
 {
```

```
    let schema =
 client.put_schema().definition(SchemaDefinition::CedarJson(schema.to_string())).policy_sto
    return schema.unwrap();
}

//Function that creates a policy in the policy store in the client
async fn create_policy(client: &Client, ps_id: &str,
 policy_definition:&PolicyDefinition) -> CreatePolicyOutput {
    let create_policy =
 client.create_policy().definition(policy_definition.clone()).policy_store_id(ps_id).send()
    return create_policy.unwrap();
}

//Function that tests the authorization request to the policy store in the client
async fn authorize(client: &Client, ps_id: &str, principal: &EntityIdentifier,
 action: &ActionIdentifier, resource: &EntityIdentifier) -> IsAuthorizedOutput {
    let is_auth =
 client.is_authorized().principal(principal.to_owned()).action(action.to_owned()).resource(
    return is_auth.unwrap();
}

#[::tokio::main]
async fn main() -> Result<(), aws_sdk_verifiedpermissions::Error> {

//Set up SDK client
    let config = aws_config::load_defaults(BehaviorVersion::latest()).await;
    let client = aws_sdk_verifiedpermissions::Client::new(&config);

//Create a policy store
    let valid_settings = ValidationSettings::builder()
    .mode({aws_sdk_verifiedpermissions::types::ValidationMode::Strict
    })
    .build()
    .unwrap();
    let policy_store = create_policy_store(&client, &valid_settings).await;
    println!(
    "Created Policy store with ID: {:?}",
    policy_store.policy_store_id
    );

//Add schema to policy store
    let schema= r#"{
        "PhotoFlash": {
            "actions": {
```

```
                "ViewPhoto": {
                    "appliesTo": {
                        "context": {
                            "type": "Record",
                            "attributes": {}
                        },
                        "principalTypes": [
                            "User"
                        ],
                        "resourceTypes": [
                            "Photo"
                        ]
                    },
                    "memberOf": []
                }
            },
            "entityTypes": {
                "Photo": {
                    "memberOfTypes": [],
                    "shape": {
                        "type": "Record",
                        "attributes": {
                            "IsPrivate": {
                                "type": "Boolean"
                            }
                        }
                    }
                },
                "User": {
                    "memberOfTypes": [],
                    "shape": {
                        "attributes": {},
                        "type": "Record"
                    }
                }
            }
        }
    }"#;
    let put_schema = put_schema(&client, &policy_store.policy_store_id,
schema).await;
    println!(
        "Created Schema with Namespace: {:?}",
        put_schema.namespaces
    );
```

```
//Create policy
    let policy_text = r#"
        permit (
            principal in PhotoFlash::User::"alice",
            action == PhotoFlash::Action::"ViewPhoto",
            resource == PhotoFlash::Photo::"VacationPhoto94.jpg"
        );
        "#;
    let policy_definition =
 PolicyDefinition::Static(StaticPolicyDefinition::builder().statement(policy_text).build().
    let policy = create_policy(&client, &policy_store.policy_store_id,
 &policy_definition).await;
    println!(
        "Created Policy with ID: {:?}",
        policy.policy_id
    );

//Break to make sure the resources are created before testing authorization
    sleep(Duration::new(2, 0));

//Test authorization
    let principal=
 EntityIdentifier::builder().entity_id("alice").entity_type("PhotoFlash::User").build().unw
    let action =
 ActionIdentifier::builder().action_type("PhotoFlash::Action").action_id("ViewPhoto").build
    let resource =
 EntityIdentifier::builder().entity_id("VacationPhoto94.jpg").entity_type("PhotoFlash::Phot
    let auth = authorize(&client, &policy_store.policy_store_id, &principal,
 &action, &resource).await;
    println!(
        "Decision: {:?}",
        auth.decision
        );
        println!(
        "Policy ID: {:?}",
        auth.determining_policies
        );
     Ok(())
}
```

3. 通过在终端`cargo run`中输入来运行代码。

如果代码运行正确，则终端将显示决定策略的策略 ID，Decision: Allow后面是确定策略的策略 ID。这意味着您已成功创建策略存储并使用适用于 Rust 的 Amazon SDK 对其进行了测试。

清理 资源

浏览完保单存储后，将其删除。

删除策略存储

您可以使用delete-policy-store操作删除策略存储，*PSEXAMPLEabcdefg111111*替换为要删除的策略存储 ID。

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

如果成功，此命令不会产生任何输出。

与 API 关联的策略存储

一个常见的用例是使用亚马逊验证权限来授权用户访问 APIs 托管在 Amazon API Gateway 上。使用 Amazon 控制台中的向导，您可以为在 Amazon Cognito 中管理的用户或任何 OIDC 身份提供商 (IdP) 创建基于角色的访问策略，并部署调 Amazon Lambda 用已验证权限的授权方来评估这些策略。

要完成向导，请在创建新的策略存储时选择设置方式 API Gateway 和身份提供商，然后按照步骤进行操作。

已创建与 API 关联的策略存储，它会为授权请求预置您的授权模型和资源。策略存储库有一个身份源和一个连接到已验证权限的 Lambda 授权者 API Gateway 。创建策略存储后，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” 只能向属于该Directors群组成员的用户授予访问权限。

随着应用程序的增长，您可以使用 C edar 策略语言实现具有用户属性和范围 OAuth 2.0 的精细授权。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。

为 API 设置授权模型后，剩下的责任是对用户进行身份验证并在应用程序中生成 API 请求，以及维护您的策略存储。

要观看演示，请查看Amazon Web Services YouTube 频道上的 Amazon 已验证权限-快速入门概述和演示。

主题

> **Important**
>
> 您在已验证权限控制台中使用设置 API Gateway 方式和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。使用以编程方式将已验证的权限部署到生产环境中。Amazon Cloud Development Kit (Amazon CDK)有关更多信息，请参阅 通过以下方式进入生产阶段 Amazon CloudFormation。

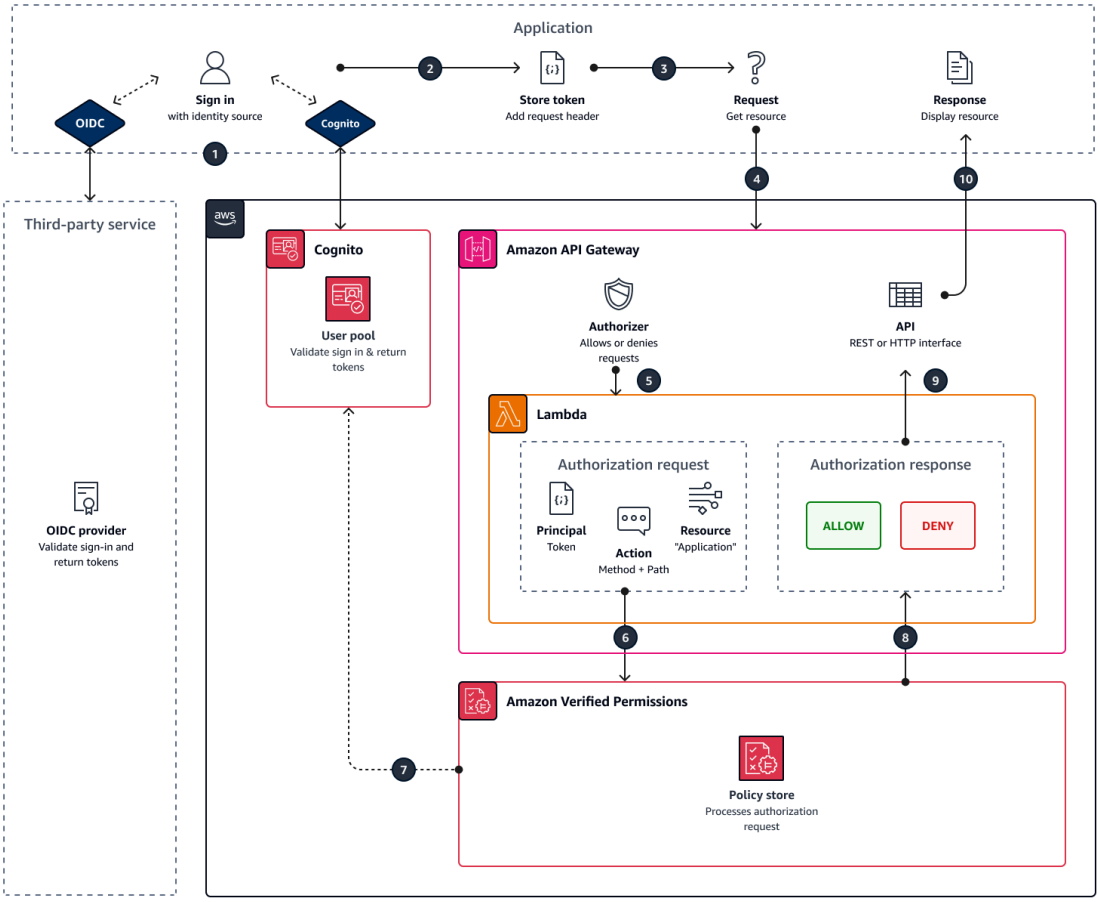
在关联到 API 和身份源的策略存储中，您的应用程序在向 API 发出请求时会在授权标头中显示用户池令牌。您的策略存储库的身份源为已验证的权限提供令牌验证。令牌与 IsAuthorizedWithTokenAPI 形成授权请求。`principal`Verified Permissions 围绕用户的群组成员资格制定策略，如身份 (ID) 和访问令牌（例如`cognito:groups`用户池）中的群组声明中所示。您的 API 在 Lambda 授权机构中处理来自您的应用程序的令牌，并将其提交给已验证权限以做出授权决定。当您的 API 收到来自 Lambda 授权方的授权决定时，它会将请求传递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

身份来源和带有已验证权限的 API Gateway 授权的组成部分

- 用于对Amazon Cognito用户进行身份验证和分组的用户池或 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 GatewayREST 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设置方式 API Gateway 和身份源工作流程会自动为您创建此 Lambda 授权方。
- 已验证权限策略存储。策略存储标识源是您的 Amazon Cognito 用户池或 OIDC 提供商组。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。

经过验证的权限如何授权 API 请求

当您创建新的策略存储并选择设置方式和身份源选项时，Ver API Gateway i fied Permissions 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组。经过验证的权限还会创建 Lambda 函数和授权者。



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 JWTs. 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. 您的应用程序从中的 REST API 请求数据 API Gateway。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 授权者。
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 Amazon 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 授权方会向返回`Allow`或`Deny`响应。 API Gateway
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。

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 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。

Amazon 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

经过验证的权限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 函数将在默认 VPC 中启动。默认情况下。 APIs 将网络访问权限限制为私有的 Lambda 函数 VPCs 无法与授权具有已验证权限的访问请求的 Lambda 函数通信。

“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 Amazon 委托人登录已验证权限控制台。该用户部署了一个 Amazon CloudFormation 堆栈，该堆栈可以跨多个 Amazon Web Services 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 和中添加和修改资源。 API Gateway 最佳做法是，不要与组织中的其他管理员共享这些凭证。

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

添加基于属性的访问控制 (ABAC)

与 IdP 的典型身份验证会话会返回 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之一。这两种类型都包含有关用户群组成员资格的信息。有关令牌类型的更多信息 Amazon Cognito，请参阅《Amazon Cognito 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。

ID 令牌

ID 令牌代表用户的属性，具有高度的细粒度访问控制。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`用户池可以通过资源服务器向令牌添加自定义范围，并在运行时自定义令牌。

有关处理 ID 和访问 Amazon Cognito 令牌声明的策略示例，请参阅将令牌映射到架构和将 OIDC 令牌映射到架构。

通过以下方式进入生产阶段 Amazon CloudFormation

与 API 关联的策略存储库是一种快速构建 API 授权模型的方法。 API Gateway 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 Amazon Web Services 账户 和的 Amazon CloudFormation 模板中 Amazon Web Services 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 关联的策略存储库由以下资源组成：

1. AWS::VerifiedPermissions::PolicyStore：将您的架构复制到`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. Amazon:: Lambda:: Function，::: Role，Amazon::: Policy，IAM::: Authori Amazon zer，:IAM:: Authoriz er，Amazon ApiGatewayAWS::Lambda::Permission

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

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 Amazon 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 Amazon Lambda 控制台中找到您的函数。导航到策略存储 Amazon Web Services 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Amazon Lambda 开发者指南 Amazon 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者，则您的策略存储的其他资源可能已失败或尚未部署。与 API 关联的策略存储将这些资源部署在堆栈中。 Amazon 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《Amazon 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。有关将身份来源的声明添加到策略的更多信息，请参阅将Amazon Cognito 令牌映射到架构和将 OIDC 令牌映射到架构。

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。

删除策略存储

您可以使用 Amazon Web Services 管理控制台 或删除 Amazon 已验证的权限策略商店 Amazon CLI。删除策略存储会永久删除架构以及策略存储中的所有策略和策略模板。与已删除的策略存储关联的所有策略存储别名也将被删除。

删除保护可防止意外删除策略存储。通过创建的所有新策略存储都启用了删除保护 Amazon Web Services 管理控制台。相比之下，通过 API 或 SDK 调用创建的所有策略存储都将禁用该功能。

出于以下原因，您可能需要删除策略存储：

- 您已达到给定区域中可用策略存储的配额。有关更多信息，请参阅 资源配额。

- 您不再支持多租户应用程序中的租户，因此不再需要该策略存储。

Amazon Web Services 管理控制台

删除策略存储

1. 打开已验证权限控制台。选择您的保单商店。
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。

> **Note**
>
> 如果启用了删除保护，则需要先将其禁用，然后才能选择 “删除”。要将其禁用，请选择 “禁用删除保护”。

Amazon CLI

删除策略存储

您可以使用`delete-policy-store`操作删除策略存储，*`PSEXAMPLEabcdefg111111`*替换为要删除的策略存储 ID。

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

如果成功，此命令不会产生任何输出。

> **Note**
>
> 如果为此策略存储启用了删除保护，则必须先运行该`update-policy-store`操作并禁用删除保护。
>
> ```
> aws verifiedpermissions update-policy-store \
>     --deletion-protection "DISABLED" \
>     --policy-store-id PSEXAMPLEabcdefg111111
> ```

Amazon 已验证权限策略存储别名

策略存储别名是策略存储的友好名称。例如，策略存储别名允许您使用policy-store-alias/example-policy-store而不是来引用策略存储。PSEXAMPLEabcdefg111111策略存储别名可用于任何接受policyStoreId输入参数的已验证权限操作。

您可以使用 CreatePolicyStoreAlias API 或使用AWS::VerifiedPermissions::PolicyStoreAlias CloudFormation 资源为策略存储创建策略存储别名。

Amazon Verified Permissions API 可以完全控制每个 Amazon Web Services 账户 和地区的策略存储别名。API 包括创建策略存储别名 (CreatePolicyStoreAlias)、查看策略存储别名和策略存储别名 ARN (GetPolicyStoreAlias,ListPolicyStoreAliases) 以及删除策略存储别名 (DeletePolicyStoreAlias) 的操作。

主题

策略存储别名的属性

亚马逊验证权限中的策略存储别名是如何运作的。

策略存储别名是独立的 Amazon 资源

策略存储别名不是策略存储的属性。您对策略存储别名采取的操作不会影响其关联的策略存储。您可以删除策略存储别名，而不会对关联的策略存储产生任何影响。如果删除策略存储，则与该策略存储关联的所有策略存储别名也会被删除。

每个策略存储别名都有一个 Amazon 资源名称 (ARN)，用于唯一标识策略存储别名。如果您在 IAM 策略中将策略存储别名指定为资源，则该策略指的是策略存储别名，而不是关联的策略存储。

每个策略存储区别名都有两种格式

创建策略存储别名时，需要指定策略存储别名。Amazon 验证权限会为您创建策略存储别名 ARN。

- 策略存储别名 ARN 是唯一标识策略存储别名的亚马逊资源名称 (ARN)。

```
# Alias ARN
arn:aws:verifiedpermissions:us-east-1:123456789012:policy-store-alias/example-policy-store
```

- 策略存储别名，在 Amazon Web Services 账户 和区域中是唯一的。在 Amazon 已验证权限 API 中，策略存储区别名始终以`policy-store-alias/`为前缀。

```
# Alias name
policy-store-alias/example-policy-store
```

策略存储区别名不是秘密

策略存储别名可能会以纯文本形式显示在 CloudTrail 日志和其他输出中。不要在策略存储别名中包含机密或敏感信息。

每个策略存储别名一次都与一个策略存储关联

策略存储别名及其关联的策略存储必须属于相同 Amazon Web Services 账户 和区域。您可以将策略存储别名与相同 Amazon Web Services 账户 和区域中的任何策略存储相关联。

例如，此`ListPolicyStoreAliases`输出显示`example-policy-store`策略存储别名恰好与一个目标策略存储相关联，该目标存储由`policyStoreId`属性表示。

```
{
    "aliasName": "policy-store-alias/example-policy-store",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "aliasArn": "arn:aws:verifiedpermissions:us-west-2:123456789012:policy-store-alias/example-policy-store",
    "createdAt": "2024-01-15T12:30:00.000000+00:00",
    "state": "Active"
}
```

多个别名可以与同一个策略存储关联

例如，您可以将`example-policy-store`和`example-policy-store-2`别名与同一个策略存储相关联。

```
[
    {
        "aliasName": "policy-store-alias/example-policy-store",
        "policyStoreId": "PSEXAMPLEabcdefg111111",
        "aliasArn": "arn:aws:verifiedpermissions:us-west-2:123456789012:policy-store-
alias/example-policy-store",
        "createdAt": "2024-01-15T12:30:00.000000+00:00",
        "state": "Active"
    },
    {
        "aliasName": "policy-store-alias/example-policy-store-2",
        "policyStoreId": "PSEXAMPLEabcdefg111111",
        "aliasArn": "arn:aws:verifiedpermissions:us-west-2:123456789012:policy-store-
alias/example-policy-store-2",
        "createdAt": "2024-01-16T09:15:00.000000+00:00",
        "state": "Active"
    }
]
```

策略存储区别名在中必须是唯一的 Amazon Web Services 账户 和区域

例如，您只能有一个策略存储别名，每个别名分别为 `example-policy-store` “区域” Amazon Web Services 账户 和 “区域”。策略存储区别名区分大小写。您无法更改策略存储的别名。但是，在 24 小时预留期到期后，您可以删除策略存储别名并使用所需名称创建新的策略存储别名。

您可以在不同的区域中创建具有相同名称的策略存储别名。每个策略存储区别名都将有一个唯一的 ARN。如果您的代码引用策略存储别名（例如）`policy-store-alias/example-policy-store`，则可以在多个区域中运行它。在每个区域，它使用不同的策略存储。

策略存储别名支持软删除和硬删除

默认情况下，当策略存储别名被删除时，该别名会被软删除。策略存储别名的保留期限为 24 小时。如果您在此期间尝试创建具有相同名称的策略存储别名，则该请求将被拒绝。在此期间，`GetPolicyStoreAlias`返回带有`PendingDeletion`状态的策略存储别名。您也可以通过指定`HardDelete`为来硬删除策略存储别名`deletionMode`。硬删除的策略存储别名会立即删除，策略存储别名将立即可供重复使用。有关更多信息，请参阅 删除 Amazon 已验证权限策略存储别名。

您可以使用别名来标识策略存储

在所有接受`policyStoreId`（例如`IsAuthorized`）的操作中，您可以使用策略存储别名来标识策略存储。在这种情况下，策略存储别名必须以为前缀`policy-store-alias/`。策略存储别名不能用于为`DeletePolicyStore`操作标识策略存储。

您不能使用策略存储别名或策略存储别名 ARN 来标识 IAM 策略Resource元素中的策略存储。要在通过策略存储别名引用策略存储时控制对策略存储的访问权限，请参阅控制对策略存储别名的访问权限。

创建 Amazon 已验证权限策略存储别名

您可以使用友好名称创建策略存储别名来引用策略存储。每个 Amazon Web Services 账户 区域的策略存储别名必须是唯一的。策略存储别名只能与策略存储别名属于相同且处于相同区域 Amazon Web Services 账户 且处于活动状态的策略存储相关联。策略存储别名是独立的资源，拥有自己的权限 ARNs 和 IAM 授权。

默认情况下，只有 10 个策略存储别名可以与同一个策略存储关联。

> **Note**
>
> CreatePolicyStoreAlias是等性的。如果您使用与现有策略存储别名匹配的策略存储别名和策略存储 ID 调用该CreatePolicyStoreAlias操作，则操作成功并返回现有的策略存储别名。CreatePolicyStoreAlias但是，如果您使用现有的策略存储别名但使用不同的策略存储 ID 调用该CreatePolicyStoreAlias操作，则该操作将返回ConflictException。

Amazon CLI

创建策略存储别名

您可以使用CreatePolicyStoreAlias操作创建策略存储别名。以下示例使用名称创建策略存储别名example-policy-store。

```
$ aws verifiedpermissions create-policy-store-alias \
    --alias-name policy-store-alias/example-policy-store \
    --policy-store-id PSEXAMPLEabcdefg111111
{
    "aliasName": "policy-store-alias/example-policy-store",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "aliasArn": "arn:aws:verifiedpermissions:us-west-2:123456789012:policy-store-
alias/example-policy-store",
    "createdAt": "2024-01-15T12:30:00.000000+00:00"
}
```

正在检索 Amazon 已验证权限策略存储别名

您可以使用GetPolicyStoreAlias操作来获取有关特定策略存储别名的详细信息，或者使用列出您 Amazon Web Services 账户 和区域中所有策略存储别名的ListPolicyStoreAliases操作，来检索有关策略存储别名的信息。

获取策略存储别名

使用该GetPolicyStoreAlias操作检索有关特定策略存储别名的详细信息，包括关联的策略存储 ID。

Amazon CLI

检索有关策略存储别名的详细信息

您可以使用GetPolicyStoreAlias操作检索策略存储别名。以下示例检索名称为的策略存储别名的example-policy-store详细信息。

```
$ aws verifiedpermissions get-policy-store-alias \
    --alias-name policy-store-alias/example-policy-store
{
    "aliasName": "policy-store-alias/example-policy-store",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "aliasArn": "arn:aws:verifiedpermissions:us-west-2:123456789012:policy-store-
alias/example-policy-store",
    "createdAt": "2024-01-15T12:30:00.000000+00:00",
    "state": "Active"
}
```

商品政策商店别名

使用ListPolicyStoreAliases操作列出您 Amazon Web Services 账户 和区域中的所有策略存储别名。您可以使用filter参数仅列出与特定策略存储关联的策略存储别名。

Amazon CLI

列出所有策略存储别名

您可以使用ListPolicyStoreAliases操作列出策略存储别名。以下示例列出了 us-west-2 区域中 1234567890 Amazon Web Services 账户 12 拥有的所有策略存储别名。

```
$ aws verifiedpermissions list-policy-store-aliases
{
    "policyStoreAliases": [
        {
            "aliasName": "policy-store-alias/example-policy-store",
            "policyStoreId": "PSEXAMPLEabcdefg111111",
            "aliasArn": "arn:aws:verifiedpermissions:us-west-2:123456789012:policy-
store-alias/example-policy-store",
            "createdAt": "2024-01-15T12:30:00.000000+00:00",
            "state": "Active"
        },
        {
            "aliasName": "policy-store-alias/example-policy-store-2",
            "policyStoreId": "PSEXAMPLEabcdefg111111",
            "aliasArn": "arn:aws:verifiedpermissions:us-west-2:123456789012:policy-
store-alias/example-policy-store-2",
            "createdAt": "2024-01-16T09:15:00.000000+00:00",
            "state": "Active"
        },
        {
            "aliasName": "policy-store-alias/example-policy-store-3",
            "policyStoreId": "PSEXAMPLEabcdefg222222",
            "aliasArn": "arn:aws:verifiedpermissions:us-west-2:123456789012:policy-
store-alias/example-policy-store-3",
            "createdAt": "2024-01-17T14:45:00.000000+00:00",
            "state": "Active"
        }
    ]
}
```

列出特定策略存储的策略存储别名

使用filter参数仅列出与特定策略存储关联的别名。

```
$ aws verifiedpermissions list-policy-store-aliases \
    --filter '{"policyStoreId": "PSEXAMPLEabcdefg111111"}'
{
    "policyStoreAliases": [
        {
            "aliasName": "policy-store-alias/example-policy-store",
            "policyStoreId": "PSEXAMPLEabcdefg111111",
            "aliasArn": "arn:aws:verifiedpermissions:us-west-2:123456789012:policy-
store-alias/example-policy-store",
```

```
            "createdAt": "2024-01-15T12:30:00.000000+00:00",
            "state": "Active"
        },
        {
            "aliasName": "policy-store-alias/example-policy-store-2",
            "policyStoreId": "PSEXAMPLEabcdefg111111",
            "aliasArn": "arn:aws:verifiedpermissions:us-west-2:123456789012:policy-
store-alias/example-policy-store-2",
            "createdAt": "2024-01-16T09:15:00.000000+00:00",
            "state": "Active"
        }
    ]
}
```

删除 Amazon 已验证权限策略存储别名

当不再需要策略存储别名时，可以将其删除。删除策略存储别名不会影响关联的策略存储。删除策略存储会删除与该策略存储关联的所有策略存储别名。

Amazon Verified Permissions 支持两种策略存储别名的删除模式：

- 软删除（默认）：策略存储别名进入`PendingDeletion`状态。策略存储别名保留 24 小时，在此期间不能重复使用。在此期间，`GetPolicyStoreAlias`返回带有`PendingDeletion`状态的策略存储别名。当您未指定或指定时`deletionMode`，这是默认行为`SoftDelete`。
- 硬删除：策略存储别名会立即删除。策略存储的别名可以立即重新使用。要执行硬删除，请指定`HardDelete`为`deletionMode`。

软删除策略存储别名

默认情况下，删除策略存储别名会执行软删除。策略存储别名进入`PendingDeletion`状态，策略存储别名保留 24 小时。

Amazon CLI

软删除策略存储别名

您可以使用DeletePolicyStoreAlias操作软删除策略存储别名。以下示例软删除了名称为的策略存储别名`example-policy-store`。

```
$ aws verifiedpermissions delete-policy-store-alias \
```

```
    --alias-name policy-store-alias/example-policy-store
```

您也可以明确指定软删除模式。

```
$ aws verifiedpermissions delete-policy-store-alias \
    --alias-name policy-store-alias/example-policy-store \
    --deletion-mode SoftDelete
```

硬删除策略存储别名

要立即删除策略存储别名，请指定HardDelete为deletionMode。硬删除的策略存储别名不会进入PendingDeletion状态，策略存储别名将立即可供重复使用。

如果您硬删除之前被软删除的策略存储别名，则策略存储别名会立即删除。

> **Important**
>
> Amazon 已验证的权限最终是一致的。如果您硬删除策略存储别名并立即重新创建它以指向其他策略存储，则引用该策略存储别名的请求可能会在短时间内继续解析到先前关联的策略存储。为避免出现意外的授权结果，请在重新创建具有相同名称的策略存储别名之前，留出一段时间让删除内容传播。

Amazon CLI

硬删除策略存储别名

您可以通过使用deletion-mode参数设置为的DeletePolicyStoreAlias操作来硬删除策略存储别名HardDelete。以下示例立即删除名称为的策略存储别名example-policy-store。

```
$ aws verifiedpermissions delete-policy-store-alias \
    --alias-name policy-store-alias/example-policy-store \
    --deletion-mode HardDelete
```

使用 Amazon 已验证权限策略在 API 操作中存储别名

任何接受policyStoreId参数（例如、和 GetPolicyStore）的 Amazon Verified Permissions 操作都可以接受策略存储别名来代替策略存储 ID。IsAuthorizedIsAuthorizedWithToken

> **Important**
>
> 使用策略存储别名作为policyStoreId参数值时，必须包含前policy-store-alias/缀。例如，使用policy-store-alias/example-policy-store，不是example-policy-store。

在操作中使用策略存储别名

以下IsAuthorized命令使用名为的策略存储别名example-policy-store来标识策略存储。

Amazon CLI

```
$ aws verifiedpermissions is-authorized \
    --policy-store-id policy-store-alias/example-policy-store \
    --principal entityType=User,entityId=alice \
    --action actionType=Action,actionId=view \
    --resource entityType=Photo,entityId=photo123
```

> **Note**
>
> 您不能使用策略存储别名来代替该DeletePolicyStore操作的policyStoreId字段。

跨使用策略存储别名 Amazon Web Services 区域

别名的最强大用途之一是在多个 Amazon Web Services 区域中运行的应用程序中。例如，您可能有一个全球应用程序，该应用程序在每个区域中使用不同的策略存储。

- 在 us-east-1 中，你想使用。PSEXAMPLEabcdefg111111
- 在 eu-west-1 中，你想使用。PSEXAMPLEabcdefg222222

您可以在每个区域创建不同版本的应用程序，也可以使用字典或 switch 语句为每个区域选择正确的策略存储。但是，在每个区域中创建具有相同策略存储别名的策略存储别名要容易得多。请记住，策略存储别名区分大小写。

Amazon CLI

```
$ aws --region us-east-1 verifiedpermissions create-policy-store-alias \
    --alias-name policy-store-alias/my-app \
    --policy-store-id PSEXAMPLEabcdefg111111

$ aws --region eu-west-1 verifiedpermissions create-policy-store-alias \
    --alias-name policy-store-alias/my-app \
    --policy-store-id PSEXAMPLEabcdefg222222
```

然后，在代码中使用策略存储别名。当您的代码在每个区域运行时，策略存储别名将引用其在该区域中的关联策略存储。

Amazon CLI

```
$ aws verifiedpermissions is-authorized \
    --policy-store-id policy-store-alias/my-app \
    --principal entityType=User,entityId=alice \
    --action actionType=Action,actionId=view \
    --resource entityType=Photo,entityId=photo123
```

但是，存在策略存储别名被删除的风险。在这种情况下，应用程序尝试使用策略存储别名将失败，您可能需要重新创建或更新策略存储别名。要降低这种风险，请谨慎地授予委托人管理您在应用程序中使用的策略存储别名的权限。

控制对策略存储别名的访问权限

管理策略存储别名的委托人必须有权与这些策略存储别名进行交互，对于某些操作，还必须有权与策略存储别名关联的策略存储进行交互。您可以使用 IAM 策略提供这些权限。

以下各节描述了创建和管理策略存储别名所需的权限。

已验证权限：CreatePolicyStoreAlias

要创建策略存储别名，委托人需要对策略存储别名和关联的策略存储都具有以下权限。

- `verifiedpermissions:CreatePolicyStoreAlias`用于策略存储别名。在附加到允许创建 IAM 策略存储别名的委托人的策略中提供此权限。

以下示例策略语句在Resource元素中指定了特定的策略存储别名。但是您可以列出多个策略存储别名 ARNs 或指定策略存储别名模式，例如"sample*"。您也可以将Resource值指定为"*"以允许委托人在 Amazon Web Services 账户 和区域中创建任何策略存储别名。

```
{
  "Sid": "IAMPolicyForCreateAlias",
  "Effect": "Allow",
  "Action": "verifiedpermissions:CreatePolicyStoreAlias",
  "Resource": "arn:aws:verifiedpermissions:us-east-1:123456789012:policy-store-alias/
example-policy-store"
}
```

- verifiedpermissions:CreatePolicyStoreAlias用于关联的策略存储。此权限必须在 IAM 策略中提供。

```
{
  "Sid": "PolicyStorePermissionForAlias",
  "Effect": "Allow",
  "Action": "verifiedpermissions:CreatePolicyStoreAlias",
  "Resource": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111"
}
```

已验证权限：GetPolicyStoreAlias

要获取有关特定策略存储别名的详细信息，委托人必须拥有策略中策略存储别名的verifiedpermissions:GetPolicyStoreAlias权限。 IAM

以下示例策略声明向委托人授予获取特定策略存储别名的权限。

```
{
  "Sid": "IAMPolicyForGetAlias",
  "Effect": "Allow",
  "Action": "verifiedpermissions:GetPolicyStoreAlias",
  "Resource": "arn:aws:verifiedpermissions:us-east-1:123456789012:policy-store-alias/
example-policy-store"
}
```

已验证权限：ListPolicyStoreAliases

要在 Amazon Web Services 账户 和区域中列出策略存储别名，委托人必须拥有 IAM 策略中的verifiedpermissions:ListPolicyStoreAliases权限。由于此策略与任何特定的策略存储或策略存储别名资源无关，因此策略中资源元素的值必须为"*"。

例如，以下 IAM 策略声明授予委托人列出中所有策略存储别名的权限。 Amazon Web Services 账户

```
{
  "Sid": "IAMPolicyForListingAliases",
  "Effect": "Allow",
  "Action": "verifiedpermissions:ListPolicyStoreAliases",
  "Resource": "*"
}
```

已验证权限：DeletePolicyStoreAlias

要删除策略存储别名，委托人只需要拥有策略存储别名的权限。

> **Note**
>
> 删除策略存储别名不会对关联的策略存储产生任何影响，但引用该策略存储别名的应用程序会收到错误。如果您错误地删除了策略存储别名，则可以在 24 小时预留期之后重新创建该别名。

委托人需要verifiedpermissions:DeletePolicyStoreAlias获得策略存储别名的权限。在附加到允许删除 IAM 策略存储别名的委托人的策略中提供此权限。

以下示例策略语句在Resource元素中指定了策略存储别名。但是您可以列出多个策略存储别名 ARNs 或指定策略存储别名模式，例如"sample*"。您也可以将Resource值指定为"*"以允许委托人删除 Amazon Web Services 账户 和区域中的任何策略存储别名。

```
{
  "Sid": "IAMPolicyForDeleteAlias",
  "Effect": "Allow",
  "Action": "verifiedpermissions:DeletePolicyStoreAlias",
  "Resource": "arn:aws:verifiedpermissions:us-east-1:123456789012:policy-store-alias/
example-policy-store"
}
```

限制策略存储别名权限

在任何接受policyStoreId字段作为输入的操作中，您可以使用策略存储别名来引用策略存储。当您这样做时，Amazon Verified Permissions 会verifiedpermissions:GetPolicyStoreAlias针对策略存储区别名和对关联的策略存储进行请求的操作进行授权。

例如，如果使用策略存储别名执行IsAuthorized操作，则委托人需要两者：

- verifiedpermissions:GetPolicyStoreAlias策略存储别名的权限
- verifiedpermissions:IsAuthorized关联策略存储的权限

以下示例策略授予IsAuthorized使用特定策略存储别名进行呼叫的权限。

```
{
  "Sid": "IAMPolicyForAliasUsage",
  "Effect": "Allow",
  "Action": "verifiedpermissions:GetPolicyStoreAlias",
  "Resource": "arn:aws:verifiedpermissions:us-east-1:123456789012:policy-store-alias/
example-policy-store"
},
{
  "Sid": "IAMPolicyForPolicyStoreOperation",
  "Effect": "Allow",
  "Action": "verifiedpermissions:IsAuthorized",
  "Resource": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111"
}
```

要限制委托人可以使用的策略存储别名，请限制verifiedpermissions:GetPolicyStoreAlias权限。例如，以下策略允许委托人使用任何策略存储别名，但以开头的别名除外Restricted。

```
{
  "Sid": "IAMPolicyForAliasAllow",
  "Effect": "Allow",
  "Action": "verifiedpermissions:GetPolicyStoreAlias",
  "Resource": "arn:aws:verifiedpermissions:us-east-1:123456789012:policy-store-alias/*"
},
{
  "Sid": "IAMPolicyForAliasDeny",
  "Effect": "Deny",
```

```
  "Action": "verifiedpermissions:GetPolicyStoreAlias",
  "Resource": "arn:aws:verifiedpermissions:us-east-1:123456789012:policy-store-alias/
Restricted*"
}
```

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。要了解已验证权限和 Cedar 处理架构的方式之间的区别，请参阅架构支持。

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。

Amazon Web Services 管理控制台

要创建架构，请按以下步骤操作：

1. 打开已验证权限控制台。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。

Amazon CLI

要提交新架构或覆盖现有架构，请使用 Amazon CLI.

您可以通过运行类似于以下示例的 Amazon 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"
```

```
}
```

Amazon 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》。

编辑策略存储架构

当您在 Amazon Verified Permissions 控制台中选择架构时，将显示构成架构的实体类型和操作。您可以在可视模式或 JSON 模式下查看和编辑您的架构。可视模式允许您使用各种向导添加新的类型和操作，从而更新架构。使用 JSON 模式，您可以直接在 JSON 编辑器中开始更新架构的 JSON 代码。

Visual Mode

可视化架构编辑器以一系列图表开头，这些图表说明了架构中各实体之间的关系。选择 “展开” 以最大化图表视图。有两个图表可用：

- 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 [action | 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. 打开已验证权限控制台。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。

JSON mode

在进行更新时，您会注意到 JSON 编辑器会根据 JSON 语法验证您的代码，并且会在您编辑时识别错误和警告，从而更轻松地快速发现问题。此外，您无需担心 JSON 的格式，只需在更新后选择“格式化 JSON”，格式就会更新以匹配预期的 JSON 格式。

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

1. 打开已验证权限控制台。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。

启用 Amazon 已验证权限策略验证模式

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，则已验证权限会拒绝请求尝试。因此，我们建议您在开发应用程序时不进行验证，将其打开以进行测试，而在应用程序处于生产状态时将其保持开启状态。

Amazon Web Services 管理控制台

要为策略存储设置策略验证模式，请按以下步骤操作：

1. 打开已验证权限控制台。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 Save changes（保存更改）。

Amazon 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。每项策略的评估都独立于其他所有策略。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。

您可以选择为策略分配策略名称。策略存储区内所有策略的策略名称必须是唯一的，并以name/此为前缀。在接受policyId参数的控制平面操作中，您可以使用策略名称代替策略 ID。以下示例使用策略名称来检索策略GetPolicy。

```
$ aws verifiedpermissions get-policy \
    --policy-id name/example-policy \
    --policy-store-id PSEXAMPLEabcdefg111111
```

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：

- 对所有主体和资源标识符使用通用唯一标识符 (UUID)。

  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：校长 == 角色:: “a1b2c3d4-e5f6-a1b2-c3d4-example11111"，//管理员
- 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 Amazon CloudTrail 跟踪中共享的日志条目中。

主题

- 创建 Amazon Verified Permissions 静态策略
- 编辑 Amazon Verified Permissions 静态策略
- 添加上下文

- 使用 Amazon 已验证权限测试平台
- Amazon Verified Permissions 示例策略

创建 Amazon Verified Permissions 静态策略

您可以为委托人创建静态策略，以允许或禁止他们对应用程序的指定资源执行指定操作。静态策略包含`principal`和`resource`的特定值，可以随时用于授权决策。

Amazon Web Services 管理控制台

要创建静态策略，请按以下步骤操作：

1. 打开已验证权限控制台。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。

> **Note**
>
> 如果您想使用政策声明，请跳至步骤 8 并将该政策粘贴到下一页的 “政策” 部分。

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。如果您选择 Per m it，则该策略允许委托人对资源执行操作。相反，如果您选择 “禁止”，则该策略不允许委托人对资源执行操作。
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。为被允许或禁止采取策略中指定的操作的委托人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。

Amazon 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"
}
```

使用策略名称创建策略

创建策略时，您可以选择指定策略名称。对于策略存储区内的所有策略，该名称必须是唯一的，并以name/此为前缀。您可以使用名称来代替策略 ID。

```
$ aws verifiedpermissions create-policy \
    --definition "{ \"static\": { \"Statement\": \"permit(principal, action,
 resource in Album::\\\"public_folder\\\");\"}}" \
    --policy-store-id PSEXAMPLEabcdefg111111 \
    --name name/example-policy
```

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

> **Note**
>
> 如果您指定的名称已与策略存储中的其他策略相关联，则会收到ConflictException错误消息。

编辑 Amazon Verified Permissions 静态策略

您可以编辑策略存储库中的现有静态策略。您只能直接更新静态策略。要更改与模板关联的策略，必须更新策略模板。有关更多信息，请参阅 编辑 Amazon 已验证权限策略模板。

您可以更改静态策略的以下元素：

- 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。要更改这些元素中的任何一个，您需要删除并重新创建策略。

- 从静态策略到模板关联策略的策略。
- 来自permit或的静态策略的效果forbid。
- 静态策略所引用的 principal。
- 静态策略所引用的 resource。

Amazon Web Services 管理控制台

要编辑静态策略，请按以下步骤操作：

1. 打开已验证权限控制台。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。

Amazon 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"
    }
}
```

更新策略的名称

更新策略时，您可以设置或更新策略名称。对于策略存储区内的所有策略，该名称必须是唯一的，并以name/此为前缀。如果您未在更新请求中包含名称字段，则现有名称将保持不变。要删除名称，请将其设置为空字符串。

```
$ aws verifiedpermissions update-policy \
    --policy-id SPEXAMPLEabcdefg111111 \
    --policy-store-id PSEXAMPLEabcdefg111111 \
    --definition file://definition.txt \
    --name name/example-policy
{
    "createdDate": "2023-06-12T20:33:37.382907+00:00",
    "lastUpdatedDate": "2023-06-12T20:47:42.804511+00:00",
    "policyId": "SPEXAMPLEabcdefg111111",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "policyType": "STATIC",
    "principal": {
        "entityId": "janeFriends",
        "entityType": "UserGroup"
    },
    "resource": {
        "entityId": "vacationFolder",
        "entityType": "Album"
    }
}
```

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。访问令牌声明是上下文。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必须 JSON-formatted 位于`contextMap`元素中。

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": {
    "AccountCodes": {
      "set": [
        {
          "long": 111122223333
        },
        {
          "long": 444455556666
        },
        {
          "long": 123456789012
        }
      ]
    },
    "approvedBy": {
    "entityIdentifier": {
      "entityId": "Bob",
      "entityType": "DigitalPetStore::User"
    }
    },
    "MfaAuthorized": {
      "boolean": true
    },
    "NetworkInfo": {
      "record": {
        "IPAddress": {
          "string": "192.0.2.178"
        },
        "Country": {
          "string": "United States of America"
```

```
        },
        "SSL": {
          "boolean": true
        }
    }
    },
    "RequestedOrderCount": {
      "long": 4
    },
    "UserAgent": {
      "string": "My UserAgent 1.12"
    }
  }
}
```

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。

字符串

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。有关以访问令牌信息为上下文的上下文示例，请参阅映射 Amazon Cognito 访问令牌和映射 OIDC 访问令牌。

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": {
        "AccountCodes": {
          "type": "Set",
          "required": true,
          "element": {
            "type": "Long"
          }
        },
        "approvedBy": {
          "name": "User",
          "required": true,
          "type": "Entity"
        },
        "MfaAuthorized": {
          "type": "Boolean",
          "required": true
        },
        "NetworkInfo": {
          "type": "NetworkInfo",
          "required": true
        },
```

```
        "RequestedOrderCount": {
          "type": "Long",
          "required": true
        },
        "UserAgent": {
          "required": true,
          "type": "String"
        }
      }
    },
    "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 CommonType 结构，方法是在actions架构中添加以下内容。commonType构造是一组共享的属性，您可以将其应用于不同的实体。

```
"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”。该策略最初只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.AccountCodes.contains(111122223333) &&
    context.approvedBy in DigitalPetStore::Role::"Employee" &&
    context.MfaAuthorized == true &&
    context.NetworkInfo.Country like "*United States*" &&
    context.NetworkInfo.IPAddress like "192.0.2.*" &&
    context.NetworkInfo.SSL == true &&
    context.RequestedOrderCount <= 4 &&
    context.UserAgent like "*My UserAgent*"
};
```

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": {
      "AccountCodes": {
        "set": [
          {"long": 111122223333},
          {"long": 444455556666},
          {"long": 123456789012}
        ]
      },
      "approvedBy": {
        "entityIdentifier": {
          "entityId": "Bob",
          "entityType": "DigitalPetStore::User"
        }
      },
      "MfaAuthorized": {
        "boolean": true
      },
      "NetworkInfo": {
        "record": {
          "Country": {"string": "United States of America"},
          "IPAddress": {"string": "192.0.2.178"},
          "SSL": {"boolean": true}
        }
      },
      "RequestedOrderCount":{
        "long": 4
      },
      "UserAgent": {
        "string": "My UserAgent 1.12"
      }
    }
  },
  "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"
}
```

使用 Amazon 已验证权限测试平台

使用已验证权限测试平台，通过对已验证的权限策略运行授权请求，对这些策略进行测试和故障排除。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。

有关测试平台的直观概述和演示，请参阅 Amazon YouTube 频道上的 Amazon 已验证权限——策略创建和测试（Primer 系列 #3）。

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. 打开已验证权限控制台。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. 打开已验证权限控制台。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。

Amazon Verified Permissions 示例策略

此处包含的一些政策示例是基本的 Cedar 政策示例，有些已通过验证 Permissions-specific。基本内容链接到Cedar政策语言参考指南，并包含在那里。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

- 允许访问单个实体
- 允许访问实体组
- 允许任何实体进行访问
- 允许访问实体的属性 (ABAC)
- 拒绝访问
- 使用方括号表示法来引用代币属性
- 使用点符号来引用属性
- 反映 Amazon Cognito ID 令牌属性
- 反映 OIDC ID 令牌的属性
- 反映 Amazon Cognito 访问令牌属性
- 反映 OIDC 访问令牌属性

使用方括号表示法来引用代币属性

以下示例说明如何创建使用方括号表示法来引用令牌属性的策略。

有关在已验证权限的策略中使用令牌属性的更多信息，请参阅将 Amazon Cognito 令牌映射到架构和将 OIDC 令牌映射到架构。

```
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.*"
};
```

使用点符号来引用属性

以下示例说明如何创建使用点符号来引用属性的策略。

有关在已验证权限的策略中使用令牌属性的更多信息，请参阅将 Amazon Cognito 令牌映射到架构和将 OIDC 令牌映射到架构。

```
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"
};
```

反映 Amazon Cognito ID 令牌属性

以下示例说明如何创建引用 ID 令牌属性的策略 Amazon Cognito。

有关在已验证权限的策略中使用令牌属性的更多信息，请参阅将 Amazon Cognito 令牌映射到架构和将 OIDC 令牌映射到架构。

```
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"
};
```

反映 OIDC ID 令牌的属性

以下示例说明如何创建引用 OIDC 提供商的 ID 令牌属性的策略。

有关在已验证权限的策略中使用令牌属性的更多信息，请参阅将 Amazon Cognito 令牌映射到架构和将 OIDC 令牌映射到架构。

```
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*"
};
```

反映 Amazon Cognito 访问令牌属性

以下示例说明如何创建引用访问令牌属性的策略 Amazon Cognito。

有关在已验证权限的策略中使用令牌属性的更多信息，请参阅将 Amazon Cognito 令牌映射到架构和将 OIDC 令牌映射到架构。

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

反映 OIDC 访问令牌属性

以下示例说明如何创建引用 OIDC 提供商的访问令牌属性的策略。

有关在已验证权限的策略中使用令牌属性的更多信息，请参阅将 Amazon Cognito 令牌映射到架构和将 OIDC 令牌映射到架构。

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

Amazon 已验证权限策略模板和模板关联政策

在已验证权限中，策略模板是带有principalresource、或两者占位符的策略。不能仅使用策略模板来处理授权请求。要处理授权请求，必须基于策略模板创建与模板关联的策略。策略模板允许只定义一次策略，然后与多个委托人和资源一起使用。对策略模板的更新会反映在使用该模板的所有策略中。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。

您可以选择为策略模板分配策略模板名称。策略模板名称在策略存储中必须是唯一的，并以name/此为前缀。在接受policyTemplateId参数的控制平面操作中，您可以使用策略模板名称代替策略模板ID。只GetPolicyTemplate在输出中ListPolicyTemplates返回名称。以下示例使用策略模板名称来检索策略模板GetPolicyTemplate。

```
$ aws verifiedpermissions get-policy-template \
    --policy-template-id name/example-policy-template \
    --policy-store-id PSEXAMPLEabcdefg111111
```

例如，以下策略模板为使用该策略模板的委托人和资源提供ReadEdit、和Comment权限。

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

如果您要创建Editor基于此模板命名的策略，则当委托人被指定为特定资源的编辑者时，您的应用程序将创建一个策略，为委托人提供读取、编辑和评论该资源的权限。

与静态策略不同，模板关联策略是动态的。以前面的示例为例，如果您要从策略模板中删除Comment操作，则任何链接到该模板或基于该模板的策略都将相应更新，并且策略中指定的委托人将无法再对相应的资源发表评论。

有关更多与模板关联的策略示例，请参阅。Amazon 已验证权限示例模板关联政策

创建 Amazon 已验证权限策略模板

您可以使用 Amazon Web Services 管理控制台、或，在 “已验证权限” 中 Amazon CLI创建策略模板Amazon SDKs。策略模板允许只定义一次策略，然后与多个委托人和资源一起使用。创建策略模板后，您可以创建与模板关联的策略，以使用包含特定委托人和资源的策略模板。有关更多信息，请参阅创建与 Amazon 验证权限模板关联的政策。

Amazon Web Services 管理控制台

创建策略模板

1. 打开已验证权限控制台。选择您的保单存储。
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。

Amazon 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"
}
```

使用策略模板名称创建策略模板

在创建策略模板时，您可以选择指定策略模板名称。对于策略存储区内的所有策略模板，该名称必须是唯一的，并以name/此为前缀。您可以使用名称来代替策略模板 ID。

```
$ aws verifiedpermissions create-policy-template \
    --description "Template for vacation picture access" \
    --statement file://template1.txt \
    --policy-store-id PSEXAMPLEabcdefg111111 \
    --name name/example-policy-template
{
    "createdDate": "2023-06-12T20:47:42.804511+00:00",
    "lastUpdatedDate": "2023-06-12T20:47:42.804511+00:00",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "policyTemplateId": "PTEXAMPLEabcdefg111111"
}
```

> **Note**
>
> 如果您指定的名称已与策略存储中的其他策略模板相关联，则会收到ConflictException错误消息。

创建与 Amazon 验证权限模板关联的政策

您可以使用、或创建与模板关联的策略或基于策略模板的 Amazon Web Services 管理控制台策略。Amazon CLI Amazon SDKs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。

有关与模板关联的策略示例，请参阅。Amazon 已验证权限示例模板关联政策

Amazon Web Services 管理控制台

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. 打开已验证权限控制台。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。

Amazon 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 == PhotoFlash::Action::"view",
    resource == PhotoFlash::Photo::"VacationPhoto94.jpg"
);
```

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

```
{
    "templateLinked": {
        "policyTemplateId": "PTEXAMPLEabcdefg111111",
        "principal": {
            "entityType": "PhotoFlash::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": "PhotoFlash::User"
    },
    "resource": {
        "entityId": "VacationPhoto94.jpg",
        "entityType": "PhotoFlash::Photo"
    }
}
```

编辑 Amazon 已验证权限策略模板

您可以使用、或，在 “已验证权限” 中 Amazon Web Services 管理控制台编辑或更新策略模板 Amazon SDKs。 Amazon CLI编辑策略模板将自动更新与该模板关联或基于该模板的策略，因此在编辑策略模板时要小心，并确保不会意外引入会破坏应用程序的更改。

您可以更改策略模板的以下元素：

- 策略模板所`action`引用的
- 条件子句，例如`when`和 `unless`

您无法更改策略模板的以下元素。要更改这些元素中的任何一个，都需要删除并重新创建策略模板。

- 来自`permit`或的策略模板的效果 `forbid`
- 策略模板所`principal`引用的
- 策略模板所`resource`引用的

Amazon Web Services 管理控制台

要编辑您的策略模板，请按以下步骤操作：

1. 打开已验证权限控制台。选择您的保单存储。

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。

Amazon CLI

编辑策略模板

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"
}
```

更新策略模板的名称

更新策略模板时，您可以设置或更新策略模板名称。对于策略存储区内的所有策略模板，该名称必须是唯一的，并以name/此为前缀。如果您未在更新请求中包含姓名字段，则现有名称将保持不变。要删除名称，请将其设置为空字符串。

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

Amazon 已验证权限示例模板关联政策

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 例子

以下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"
 );
```

以下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"
 );
```

以下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"
 );
```

以下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"
 );
```

以下示例显示了如何使用策略模板创建与模板关联的策略，该策略模板授予对与好友群组共享的非私密照片和个人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"
 );
```

以下示例显示了如何使用策略模板 “阻止用户访问账户” 来创建与模板关联的策略。

```
forbid(
 principal == PhotoFlash::User::"Bob",
```

```
 action,
 resource in PhotoFlash::Account::"Alice-account"
 );
```

DigitalPetStore 例子

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 例子

以下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，为单个用户和任务列表提供查看者访问权限。

```
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"
);
```

以下示例显示了如何创建与模板关联的策略，该策略使用策略模板为单个用户和任务列表提供编辑访问权限。

```
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"
);
```

使用身份源和令牌保护您的应用程序

在 Amazon Verified Perm issions 中创建代表外部身份提供商 (IdP) 的身份源，从而快速保护您的应用程序。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。您可以添加 Amazon Cognito 用户池或自定义 OpenID Connect (OIDC) IdP 作为身份来源。

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符合 OIDC 的身份提供商生成的 JSON Web 令牌 (JWTs) 生成授权请求。令牌中的用户身份映射到委托人 ID。使用 ID 令牌，“已验证权限” 会将属性声明映射到委托人属性。使用访问令牌，这些声明会映射到上下文。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。

> **Note**
>
> 已验证权限根据来自 IdP 令牌的信息做出授权决定，但不会以任何方式直接与 IdP 交互。

有关 APIs使用 Amazon Cognito 用户池或 OIDC 身份提供商为 Amazon API Gateway REST 构建授权逻辑的 step-by-step演练，请参阅安全博客上的 “使用 API Gateway APIs 亚马逊验证权限授权” Amazon Cognito 或自带身份提供商进行授权。Amazon

主题

- 选择合适的身份提供商
- 使用 Amazon Cognito 身份来源
- 使用 OIDC 身份来源

选择合适的身份提供商

虽然经过验证的权限适用于各种各样的 IdPs，但在决定在应用程序中使用哪个权限时，请考虑以下几点：

Amazon Cognito 在以下情况下使用：

- 你在没有现有身份基础架构的情况下构建新的应用程序
- 你想要具有内置安全功能的 Amazon托管用户池

- 您需要整合社交身份提供商
- 你想要简化的代币管理

在以下情况下使用 OIDC 提供商：

- 你有现有的身份基础架构（Auth0、Okta、Azure AD）
- 您需要保持集中的用户管理
- 您有具体的合规要求 IdPs

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 JWTs 具有可预测的结构。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 Amazon Web Services 区域。
- 用户池 ID。
- 例如，您要与身份源关联的委托人实体类型`MyCorp::User`。
- 例如，您要与身份源关联的委托人组实体类型`MyCorp::UserGroup`。
- 您想要授权的用户池 IDs 中的客户端，以便向您的策略存储库发出请求。

由于已验证权限仅适用于相同的 Amazon Cognito 用户池 Amazon Web Services 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`在这种情况下，您可以将该用户池中的用户引用`a1b2c3d4-5678-90ab-cdef-EXAMPLE22222`为 `us-west-2_EXAMPLE|a1b2c3d4-5678-90ab-cdef-EXAMPLE22222`

用户池令牌声明可以包含属性、范围、群组 IDs、客户和自定义数据。 Amazon Cognito JWTs能够在“已验证权限” 中包含可能有助于做出授权决策的各种信息。这些方法包括：

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将 Amazon Cognito 令牌映射到架构。

> **Important**
>
> 尽管您可以在 Amazon Cognito 令牌到期之前将其撤销，但 JWTs 它们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 可以根据已撤销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。有关更多信息，请参阅通过撤消令牌结束用户会话

以下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,
     action,
     resource == ExampleCo::Photo::"VacationPhoto94.jpg"
)
when {
     principal["cognito:username"]) == "alice" &&
     principal["custom:department"]) == "Finance"
};
```

以下示例说明如何创建引用 Cognito 用户池中用户的委托人的策略。请注意，委托人 ID 的形式为"<userpool-id>|<sub>"。

```
permit(
     principal == ExampleCo::User::"us-east-1_example|a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111",
     action,
     resource == ExampleCo::Photo::"VacationPhoto94.jpg"
);
```

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:字符的用户池前缀声明，例如cognito:username和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`。

有关映射声明的更多信息，请参阅将 Amazon Cognito 令牌映射到架构。有关 Amazon Cognito 用户授权的更多信息，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。

主题

- 创建 Amazon 认证权限 Amazon Cognito 身份源
- 编辑 Amazon 已验证权限的 Amazon Cognito 身份来源
- 将 Amazon Cognito 令牌映射到架构
- 客户和受众验证 Amazon Cognito

创建 Amazon 认证权限 Amazon Cognito 身份源

以下过程将身份源添加到现有策略存储中。

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。选择引导式设置或设置方式 API Gateway 和身份提供者选项。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。

使用 Amazon CLI 或CreateIdentitySource在已验证权限 API create-identity-source中创建身份源时，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`Amazon::Cognito`。此实体名称与策略存储架构不兼容。要将 Amazon Cognito 身份与策略存储架构集成，必须将委托人实体类型设置为支持的策略存储实体。

Amazon Web Services 管理控制台

要创建 Amazon Cognito 用户群体身份来源，请按以下步骤操作：

1. 打开已验证权限控制台。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 Amazon Web Services 区域 并输入您的身份源的用户池 ID。
5. 在承担者配置中，对于 “委托人类型”，为来自该来源的委托人选择实体类型。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 IDs。
   - 要验证客户端应用程序 IDs，请选择 “仅接受具有匹配客户端应用程序的令牌” IDs。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。
   - IDs如果您不想验证客户端应用程序，请选择不验证客户端应用程序 IDs。
8. 选择创建身份来源。
9. （可选）如果您的策略存储具有架构，则必须先更新架构，让 Cedar 知道您的身份源创建的主体类型，然后才能引用从 Cedar 策略中的身份或访问令牌中提取的属性。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，请参阅将 Amazon Cognito 令牌映射到架构。

   > **Note**
   >
   > 当您创建与 API 关联的策略存储或在创建策略存储时使用设置 API Gateway 和身份提供商时，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。

10. 创建使用令牌中的信息做出授权决策的策略。有关更多信息，请参阅 创建 Amazon Verified Permissions 静态策略。

现在，您已经创建了身份源、更新了架构并创建了策略，请使用已验证的权限`IsAuthorizedWithToken`来做出授权决定。有关更多信息，请参阅IsAuthorizedWithTokenAmazon 已验证权限 API 参考指南。

Amazon 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CreateIdentitySource操作创建身份源。以下示例创建了一个可以从 Amazon Cognito 用户池中访问经过身份验证的身份的身份源。

1. 创建一个包含以下 Amazon Cognito 用户池详细信息的`config.txt`文件，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"
        }
    }
}
```

2. 运行以下命令创建 Amazon Cognito 身份源。

```
$ 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"
}
```

3. （可选）如果您的策略存储具有架构，则必须先更新架构，让 Cedar 知道您的身份源创建的主体类型，然后才能引用从 Cedar 策略中的身份或访问令牌中提取的属性。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，请参阅将 Amazon Cognito 令牌映射到架构。

> **Note**
>
> 当您创建与 API 关联的策略存储或在创建策略存储时使用设置 API Gateway 和身份提供商时，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。

4. 创建使用令牌中的信息做出授权决策的策略。有关更多信息，请参阅 创建 Amazon Verified Permissions 静态策略。

现在，您已经创建了身份源、更新了架构并创建了策略，请使用已验证的权限`IsAuthorizedWithToken`来做出授权决定。有关更多信息，请参阅 IsAuthorizedWithTokenAmazon 已验证权限 API 参考指南。

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。

编辑 Amazon 已验证权限的 Amazon Cognito 身份来源

创建身份源后，您可以编辑身份源的某些参数。您无法更改身份源的类型，必须删除身份源并创建一个新的身份源以从 OIDC 或 OIDC 切换 Amazon Cognito 到。 Amazon Cognito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。

Amazon Web Services 管理控制台

更新 Amazon Cognito 身份源

1. 打开已验证权限控制台。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 Amazon Web Services 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 IDs。
   - 要验证客户端应用程序 IDs，请选择 “仅接受具有匹配客户端应用程序的令牌” IDs。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。
   - IDs如果您不想验证客户端应用程序，请选择不验证客户端应用程序 IDs。
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。

Amazon CLI

更新 Amazon Cognito 身份源

您可以使用UpdateIdentitySource操作更新身份源。以下示例将指定的身份源更新为使用其他 Amazon Cognito 用户池。

1. 创建一个包含以下 Amazon Cognito 用户池详细信息的`config.txt`文件，供`update-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"
        }
    }
}
```

2. 运行以下命令以更新 Amazon Cognito 身份源。

```
$ 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"
}
```

> **Note**
>
> 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。

将 Amazon Cognito 令牌映射到架构

您可能会发现想要将身份源添加到策略存储中，并将地图提供商声明或令牌添加到策略存储架构中。您可以自动执行此过程，方法是使用引导式设置创建带有身份源的策略存储，或者在创建策略存储后手动更新架构。将令牌映射到架构后，您可以创建引用它们的策略。

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策略中使用 Amazon Cognito 令牌声明
- 如何为身份源手动构建架构

通过引导式设置创建的 API 关联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。您可以为用户池中的属性提供经过验证的权限，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：

- 您根据示例创建了一个空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、 Amazon 软件开发工具包或创建策略存储 Amazon CDK。

要在您的已验证权限策略存储中 Amazon Cognito 用作身份源，您的架构中必须包含提供者属性。架构是固定的，必须与提供者令牌在其中创建的实体IsAuthorizedWithToken或BatchIsAuthorizedWithTokenAPI 请求相对应。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与使用 API 请求创建的实体相匹配的提供者属性。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

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。

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。

Amazon Cognito 身份令牌中的有用声明

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`。

标准声明

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 Cognito 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。它包括 Amazon Cognito特定索赔cognito:username、自定义索赔custom:employmentStoreCode、标准索赔email和临时索赔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"
```

```
}
```

使用 Amazon Cognito 用户池创建身份源时，您可以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，请参阅编辑策略存储架构。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
         }
      }
   }
}
```

有关将针对此架构进行验证的策略示例，请参阅反映 Amazon Cognito ID 令牌属性。

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。

Amazon Cognito 访问令牌具有可用于授权的声明：

Amazon 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 Amazon 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 Cognito 开发者指南中的使用访问令牌。

当 Amazon Cognito 访问令牌传递给 “已验证权限” 时，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，请参阅编辑策略存储架构。

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

有关将针对此架构进行验证的策略示例，请参阅反映 Amazon Cognito 访问令牌属性。

Amazon Cognito 冒号分隔的索赔的替代表示法

在 Verified Permissi `cognito:groups` ons 启动时， Amazon Cognito 令牌声明的推荐架构会`custom:store`将这些冒号分隔的字符串转换为使用该`.`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架构和 Amazon Cognito 身份源的策略中使用点符号。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
        }
     }
  }
}
```

有关将针对此架构进行验证并使用点符号的策略示例，请参阅使用点符号来引用属性。

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。访问令牌的上下文属性包含诸如 OAuth 2.0 作用域之类的信息，这些信息可以提供来自提供者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 和身份提供者进行设置” 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。

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，以及适用于您的 API 的 Lambda 授权者的语法。 API Gateway 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:username` `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"
      ]
   }
}
```

有关将针对此架构进行验证的策略示例，请参阅使用方括号表示法来引用代币属性。

客户和受众验证 Amazon Cognito

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。ID 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 是与您的应用程序使用的身份提供商实例关联的标识符`1example23456789`。例如，受众是与访问令牌的预期依赖方或目标相关联的 URL 路径`https://mytoken.example.com`。使用访问令牌时，`aud`声明始终与受众相关联。

Amazon 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 IDs 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 身份来源的依赖方受众 URL。

的客户端授权 JWTs

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。

此示例说明如何使用 JWT 从应用程序调用 “已验证权限”。[1]

```
async function authorizeUsingJwtToken(jwtToken) {

    const payload = await verifier.verify(jwtToken);

    let 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
    };
    let 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
    };
    let 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
    };
    let entities = {
        entityList: [],
    };
    entities.entityList.push(...getUserEntitiesFromToken(payload));
    let 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 === 'boolean') {
        attributes[key] = {
            boolean: value,
       }
    }
```

```
  });

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

[1] 此代码示例使用该aws-jwt-verify库来验证由 OID IdPs C JWTs 兼容的签名。

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：它们是作为身份验证的 JWTs 产物生成的。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。
- 不包含通配符的 CNAME 记录。例如，a.example.com无法映射到*.example.net。相反，*.example.com无法映射到。a.example.net
- 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的前缀 IDs。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 和身份提供者进行设置” 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。

有关使用 API 操作授权来自 OIDC 来源的请求的更多信息，请参阅。可用于授权的 API 操作

以下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可以访问年终报告。已验证权限从委托人 ID 令牌中的声明中派生这些属性。

请注意，在委托人中引用组时，必须使用in运算符才能正确评估策略。

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

主题

- 创建 Amazon 已验证权限 OIDC 身份源
- 编辑 Amazon 已验证权限 OIDC 身份源
- 将 OIDC 令牌映射到架构
- OIDC 提供商的客户和受众验证

创建 Amazon 已验证权限 OIDC 身份源

以下过程将身份源添加到现有策略存储中。

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 “设置方式” API Gateway 和 “身份提供商” 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。

使用 Amazon CLI 或CreateIdentitySource在已验证权限 API create-identity-source中创建身份源时，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Amazon::Cognito。此实体名称与策略存储架构不兼容。要将 Amazon Cognito 身份与策略存储架构集成，必须将委托人实体类型设置为支持的策略存储实体。

Amazon Web Services 管理控制台

创建 OpenID Connect (OIDC) 身份源

1. 打开已验证权限控制台。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 将 OIDC 令牌映射到架构。
7. 在将令牌声明映射到架构实体中，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. （可选）在 “将令牌声明映射到架构实体” 中，为身份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 “验证-可选” 中，输入您 URLs 希望您的策略商店在授权请求中接受的客户 IDs 或受众（如果有）。
10. 选择创建身份来源。
11. （可选）如果您的策略存储具有架构，则必须先更新架构，让 Cedar 知道您的身份源创建的主体类型，然后才能引用从 Cedar 策略中的身份或访问令牌中提取的属性。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 OIDC 代币属性映射到 Cedar 主体属性的更多信息，请参阅。将 OIDC 令牌映射到架构
12. 创建使用令牌中的信息做出授权决策的策略。有关更多信息，请参阅 创建 Amazon Verified Permissions 静态策略。

现在，您已经创建了身份源、更新了架构并创建了策略，请使用已验证的权限`IsAuthorizedWithToken`来做出授权决定。有关更多信息，请参阅 IsAuthorizedWithTokenAmazon 已验证权限 API 参考指南。

Amazon CLI

创建 OIDC 身份源

您可以使用CreateIdentitySource操作创建身份源。以下示例创建了可以从 OIDC 身份提供商 (IdP) 访问经过身份验证的身份的身份源。

1. 创建一个包含 OIDC IdP 以下详细信息的config.txt文件，供命令的--configuration参数使用。create-identity-source

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

2. 运行以下命令创建 OIDC 身份源。

```
$ 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"
}
```

3. （可选）如果您的策略存储具有架构，则必须先更新架构，让 Cedar 知道您的身份源创建的主体类型，然后才能引用从 Cedar 策略中的身份或访问令牌中提取的属性。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 OIDC 代币属性映射到 Cedar 主体属性的更多信息，请参阅。将 OIDC 令牌映射到架构

4. 创建使用令牌中的信息做出授权决策的策略。有关更多信息，请参阅 创建 Amazon Verified Permissions 静态策略。

现在，您已经创建了身份源、更新了架构并创建了策略，请使用已验证的权限`IsAuthorizedWithToken`来做出授权决定。有关更多信息，请参阅 IsAuthorizedWithTokenAmazon 已验证权限 API 参考指南。

编辑 Amazon 已验证权限 OIDC 身份源

创建身份源后，您可以编辑身份源的某些参数。您无法更改身份源的类型，必须删除身份源并创建一个新的身份源以从 OIDC 或 OIDC 切换 Amazon Cognito 到。 Amazon Cognito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。

Amazon Web Services 管理控制台

更新 OIDC 身份源

1. 打开已验证权限控制台。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。

Amazon CLI

更新 OIDC 身份源

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

1. 创建一个包含 OIDC IdP 以下详细信息的config.txt文件，供命令的--configuration参数使用。update-identity-source

```
{
    "openIdConnectConfiguration": {
        "issuer": "https://auth2.example.com",
        "tokenSelection": {
                "identityTokenOnly": {
                        "clientIds":["2example10111213"],
                        "principalIdClaim": "sub"
                },
        },
        "entityIdPrefix": "MyOIDCProvider",
        "groupConfiguration": {
              "groupClaim": "groups",
              "groupEntityType": "MyCorp::UserGroup"
        }
    }
}
```

2. 运行以下命令更新 OIDC 身份源。

```
$ 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"
}
```

> **Note**
>
> 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。

将 OIDC 令牌映射到架构

您可能会发现想要将身份源添加到策略存储中，并将地图提供商声明或令牌添加到策略存储架构中。您可以自动执行此过程，方法是使用引导式设置创建带有身份源的策略存储，或者在创建策略存储后手动更新架构。将令牌映射到架构后，您可以创建引用它们的策略。

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
- 如何为身份源手动构建架构

通过引导式设置创建的 API 关联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。您可以为用户池中的属性提供经过验证的权限，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。

要使用 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者属性。架构是固定的，必须与提供者令牌创建的实体IsAuthorizedWithToken或BatchIsAuthorizedWithTokenAPI 请求相对应。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与使用 API 请求创建的实体相匹配的提供者属性。然后，您可以使用提供者令牌中的属性来编写策略。

主题

- 将 ID 令牌映射到架构
- 映射访问令牌
- 关于架构映射的注意事项

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。

使用 OIDC 提供商提供的 ID 令牌与使用 Amazon 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"
         }
      }
   }
}
```

有关将针对此架构进行验证的策略示例，请参阅反映 OIDC ID 令牌的属性。

映射访问令牌

Verified Permissions 处理除组声明之外的访问令牌声明作为操作属性或上下文属性。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。

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，请参阅编辑策略存储架构。

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

有关将针对此架构进行验证的策略示例，请参阅反映 OIDC 访问令牌属性。

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。

不需要身份源属性

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可以为基于属性的访问控制 (ABAC) 授权做出贡献的用户信息。访问令牌的上下文属性包含诸如 OAuth 2.0 作用域之类的信息，这些信息可以提供来自提供者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 和身份提供者进行设置” 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，设置名为的群组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。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:username 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"
      ]
   }
}
```

有关将针对此架构进行验证的策略示例，请参阅使用方括号表示法来引用代币属性。

OIDC 提供商的客户和受众验证

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。ID 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 是与您的应用程序使用的身份提供商实例关联的标识符`1example23456789`。例如，受众是与访问令牌的预期依赖方或目标相关联的 URL 路径`https://mytoken.example.com`。使用访问令牌时，`aud`声明始终与受众相关联。

OIDC ID 令牌的`aud`声明包含客户端 IDs，例如。`1example23456789`

OIDC 访问令牌的`aud`声明包含令牌的受众网址（例如`https://myapplication.example.com`）和包含客户端 IDs（例如）的`client_id`声明。`1example23456789`

设置策略存储时，请输入一个或多个受众验证值，您的政策存储使用该值来验证令牌的受众。

- ID 令牌 — 已验证权限通过检查`aud`声明 IDs 中至少有一名客户成员与受众验证值匹配来验证客户端 ID。
- 访问令牌 — 已验证权限通过检查`aud`声明中的网址是否与受众验证值相匹配来验证受众。如果不存在任何`aud`声明，则可以使用`cid`或`client_id`声明来验证受众。请咨询您的身份提供商，了解正确的受众主张和格式。

的客户端授权 JWTs

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。

此示例说明如何使用 JWT 从应用程序调用 “已验证权限”。[1]

```
async function authorizeUsingJwtToken(jwtToken) {

    const payload = await verifier.verify(jwtToken);

    let 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
```

```
    };
    let 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
    };
    let 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
    };
    let entities = {
        entityList: [],
    };
    entities.entityList.push(...getUserEntitiesFromToken(payload));
    let 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 === 'boolean') {
        attributes[key] = {
            boolean: value,
       }
    }

  });

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

[1] 此代码示例使用该aws-jwt-verify库来验证由 OID IdPs C JWTs 兼容的签名。

Amazon 已验证权限的集成

Amazon Verified Permissions 集成可帮助您在应用程序中实现精细授权，同时最大限度地减少代码并遵循特定于框架的最佳实践。这些集成提供了中间件组件和实用工具，可将您的应用程序与经过验证的权限无缝连接。

通过集成，您可以：

- 在几分钟内实现授权
- 遵循特定于框架的模式和惯例
- 减少维护开销
- 最大限度地减少潜在的安全实施错误
- 专注于业务逻辑而不是授权码

添加到您的应用程序后，集成会执行以下操作：

1. 通过特定于框架的中间件拦截传入的请求
2. 从请求中提取相关的授权上下文
3. 使用已验证的权限确定授权决策
4. 根据授权结果实施访问控制

已验证的权限目前支持以下框架：

- Express.js 适用于 Node.js 应用程序

将 Express 与亚马逊验证权限集成

Verified Permissions Express 集成提供了一种基于中间件的方法来在 Express.js 应用程序中实现授权。通过这种集成，您可以使用精细的授权策略保护您的 API 端点，而无需修改现有的路由处理程序。该集成通过拦截请求、根据您定义的策略对其进行评估并确保只有授权用户才能访问受保护的资源来自动处理授权检查。

本主题将引导您完成设置 Express 集成，从创建策略存储到实施和测试授权中间件。通过执行这些步骤，您只需最少的代码更改即可向 Express 应用程序添加强大的授权控件。

本主题中引用了以下GitHub存储库：

- cedar-policy/ authorization-for-expressjs-Express.js 的 Cedar 授权中间件
- veri@@ fiedpermissions/ authorization-clients-js-的已验证权限授权客户端 JavaScript
- verifiedpermissions/examples/express-petstore-使用 Express.js 中间件的示例实现

先决条件

在实施 Express 集成之前，请确保：

- 有权访问已验证权限的Amazon 账户
- Node.js 和 npm 安装了
- 一个 Express.js 应用程序
- OpenID Connect (OIDC) 身份提供商（例如）Amazon Cognito
- Amazon CLI配置了适当的权限

设置集成

步骤 1：创建策略存储

使用以下方法创建策略存储 Amazon CLI：

```
aws verifiedpermissions create-policy-store --validation-settings "mode=STRICT"
```

> **Note**
>
> 保存响应中返回的策略存储 ID，以便在后续步骤中使用。

步骤 2：安装依赖关系

在您的 Express 应用程序中安装所需的软件包：

```
npm i --save @verifiedpermissions/authorization-clients-js
npm i --save @cedar-policy/authorization-for-expressjs
```

配置授权

步骤 1：生成并上传 Cedar 架构

架构定义了应用程序的授权模型，包括应用程序中的实体类型和允许用户执行的操作。我们建议为架构定义命名空间。在此示例中，我们使用的是 YourNamespace。您将架构附加到已验证权限策略存储中，当添加或修改策略时，该服务会自动根据架构验证策略。

该@cedar-policy/authorization-for-expressjs软件包可以分析您的应用程序的 OpenAPI 规格并生成 Cedar 架构。具体而言，路径对象在您的规范中是必需的。

如果您没有 OpenAPI 规范，则可以按照express-openapi-generator软件包的快速说明生成 OpenAPI 规范。

根据你的 OpenAPI 规范生成一个架构：

```
npx @cedar-policy/authorization-for-expressjs generate-schema --api-spec schemas/
openapi.json --namespace YourNamespace --mapping-type SimpleRest
```

接下来，格式化 Cedar 架构以便与一起使用 Amazon CLI。有关所需特定格式的更多信息，请参阅策略存储架构。如果你需要格式化架构方面的帮助，可以在已验证的权限 GitHub /示例存储库prepare-cedar-schema.sh中调用一个名为的脚本。以下是对该脚本的调用示例，该脚本在v2.cedarschema.forAVP.json文件中输出已验证权限格式的架构。

```
./scripts/prepare-cedar-schema.sh v2.cedarschema.json v2.cedarschema.forAVP.json
```

将格式化的架构上传到您的策略存储区，policy-store-id替换为您的策略存储 ID：

```
aws verifiedpermissions put-schema \
  --definition file://v2.cedarschema.forAVP.json \
  --policy-store-id policy-store-id
```

步骤 2：创建授权策略

如果未配置任何策略，Cedar 将拒绝所有授权请求。Express 框架集成通过根据先前生成的架构生成示例策略来帮助启动此过程。

在生产应用程序中使用此集成时，我们建议使用基础设施即代码 (IaaC) 工具创建新策略。有关更多信息，请参阅 与 Amazon CloudFormation。

生成 Cedar 策略示例：

```
npx @cedar-policy/authorization-for-expressjs generate-policies --schema
 v2.cedarschema.json
```

这将在/policies目录中生成示例策略。然后，您可以根据自己的用例自定义这些策略。例如：

```
// Defines permitted administrator user group actions
permit (
    principal in YourNamespace::UserGroup::"<userPoolId>|administrator",
    action,
    resource
);

// Defines permitted employee user group actions
permit (
    principal in YourNamespace::UserGroup::"<userPoolId>|employee",
    action in
        [YourNamespace::Action::"GET /resources",
         YourNamespace::Action::"POST /resources",
         YourNamespace::Action::"GET /resources/{resourceId}",
         YourNamespace::Action::"PUT /resources/{resourceId}"],
    resource
);
```

格式化策略以便与一起使用 Amazon CLI。有关所需格式的更多信息，请参阅参考资料中的 create-pol icy。Amazon CLI 如果你在格式化策略时需要帮助，可以在已验证的权限 GitHub /示例存储库convert_cedar_policies.sh中有一个名为的脚本。以下是对该脚本的调用：

```
./scripts/convert_cedar_policies.sh
```

将格式化的策略上传到 Verified Per policy_1.json missions，替换为策略文件的路径和policy-store-id名称以及您的策略存储库 ID：

```
aws verifiedpermissions create-policy \
  --definition file://policies/json/policy_1.json \
  --policy-store-id policy-store-id
```

步骤 3：连接身份提供商

默认情况下，经过验证的权限授权中间件会读取 API 请求的授权标头中提供的 JSON Web 令牌 (JWT)，以获取用户信息。除了执行授权策略评估外，已验证权限还可以验证令牌。

使用您的userPoolArn和创建一个名为的身份源配置文件identity-source-configuration.txt，该文件如下所示clientId：

```
{
    "cognitoUserPoolConfiguration": {
        "userPoolArn": "arn:aws:cognito-idp:region:account:userpool/pool-id",
        "clientIds": ["client-id"],
        "groupConfiguration": {
            "groupEntityType": "YourNamespace::UserGroup"
        }
    }
}
```

通过运行以下 Amazon CLI 命令创建身份源，policy-store-id替换为您的策略存储 ID：

```
aws verifiedpermissions create-identity-source \
  --configuration file://identity-source-configuration.txt \
  --policy-store-id policy-store-id \
  --principal-entity-type YourNamespace::User
```

实现授权中间件

更新您的 Express 应用程序以包含授权中间件。在此示例中，我们使用的是身份令牌，但您也可以使用访问令牌。有关更多信息，请参阅authorization-for-expressjs上的GitHub。

```
const { ExpressAuthorizationMiddleware } = require('@cedar-policy/authorization-for-
expressjs');

const { AVPAuthorizationEngine } = require('@verifiedpermissions/authorization-
clients');

const avpAuthorizationEngine = new AVPAuthorizationEngine({
    policyStoreId: 'policy-store-id',
    callType: 'identityToken'
});

const expressAuthorization = new ExpressAuthorizationMiddleware({
    schema: {
        type: 'jsonString',
        schema: fs.readFileSync(path.join(__dirname, '../v4.cedarschema.json'),
 'utf8'),
    },
```

```
    authorizationEngine: avpAuthorizationEngine,
    principalConfiguration: { type: 'identityToken' },
    skippedEndpoints: [],
    logger: {
        debug: (s) => console.log(s),
        log: (s) => console.log(s),
    }
});

// Add the middleware to your Express application
app.use(expressAuthorization.middleware);
```

测试集成

您可以使用不同的用户令牌向 API 端点发出请求，从而测试您的授权实现。授权中间件将根据您定义的策略自动评估每个请求。

例如，如果您设置了具有不同权限的不同用户组：

- 管理员：对所有资源和管理功能的完全访问权限
- 员工：可以查看、创建和更新资源
- 客户：只能查看资源

您可以通过使用不同的用户登录并尝试各种操作来验证权限策略是否按预期运行。在 Express 应用程序的终端中，您可以看到日志输出，其中提供了有关授权决策的更多详细信息。

问题排查

如果授权失败，请尝试以下方法：

- 验证您的保单商店 ID 是否正确
- 确保您的身份源配置正确
- 检查您的政策格式是否正确
- 验证您的 JWT 令牌是否有效

后续步骤

实现基本集成后，请考虑：

- 为特定的授权场景实现自定义映射器
- 为授权决策设置监控和日志记录
- 为不同的用户角色创建其他策略

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，必须将中授权策略的配置 Amazon 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 Amazon 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，并从外部引入信息 APIs。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创建DigitalPetStore示例策略存储。

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Amazon 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”。

主题

可用于授权的 API 操作

已验证权限 API 具有以下授权操作。

IsAuthorized

IsAuthorizedAPI 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。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 JSON Web 令牌 (JWTs) 中的用户数据生成授权请求。Verified Permissions 可直接与 OIDC 提供商配合使用，例如 Amazon Cognito 作为策略存储中的身份源。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。您可以通过身份源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。

BatchIsAuthorized

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。

BatchIsAuthorizedWithToken

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用户目录中的群组成员资格来授权他们的访问权限。

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

要了解部署应用程序时 Amazon 已验证权限授权决定的影响，您可以在制定策略时使用使用 Amazon 已验证权限测试平台和通过 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 Amazon Web Services 管理控制台 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、和BatchIsAuthorized作为已签名的 Amazon API

向区域服务终端节点发出的授权请求。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. 在已验证权限控制台上打开已验证权限控制台。使用名称从示例策略存储中创建策略存储DigitalPetStore。
2. 在新保单库中选择 “测试台”。
3. 在已验证权限 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 中引用该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，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. 在已验证权限控制台上打开已验证权限控制台。使用名称从示例策略存储中创建策略存储DigitalPetStore。
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 中引用该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DigitalPetStore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. 更改请求的方式会更改已验证权限的决定。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 “客户角色-获取订单” 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。

将您的授权模型与应用程序集成

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 身份来源，也可能来自其他 Open ID Connect (O IDC) 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 操作内置于中 Amazon SDKs。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 Amazon SDK 集成到应用程序包中。

要了解更多信息并进行下载 Amazon SDKs，请参阅工具 Amazon Web Services。

以下是各种已验证权限资源的文档链接 Amazon SDKs。

- 适用于 .NET 的 Amazon SDK
- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Java 的 Amazon SDK
- 适用于 JavaScript 的 Amazon SDK
- 适用于 PHP 的 Amazon SDK
- Amazon SDK for Python (Boto)
- 适用于 Ruby 的 Amazon SDK
- 适用于 Rust 的 Amazon SDK

以下 适用于 JavaScript 的 Amazon SDK 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 简化细粒度授权

提出 API 请求

对亚马逊认证权限的查询请求是使用该POST方法的 HTTP 或 HTTPS 请求。

已验证权限终端节点

端点是用作 Web 服务入口点的 URL。在请求减少延迟时，您可以选择适当的 Amazon Web Services 区域 终端节点。有关已验证权限使用的终端节点的信息，请参阅中的 Amazon 已验证权限Amazon Web Services 一般参考。

请求参数

Amazon 已验证的权限使用 JSON-based 协议。您可以在 HTTP 请求的正文中以 JSON 的形式传递请求参数。列表以 JSON 数组表示。有关更多信息，请参阅《Amazon 已验证权限 API 参考》中的常用参数。

请求标识符

在每个响应中，都通过 `x-amzn-RequestId` HTTP 响应标头 Amazon 返回一个请求 ID。此字符串是 Amazon 分配以提供跟踪信息的唯一标识符。尽管请求 ID 包含在每个响应中，但为了提高可读性并减少冗余，它并未在各个 API 文档页面上列出。

查询 API 身份验证

您可以通过 HTTPS 发送查询请求。您必须在每个查询请求中包含签名。有关创建和添加签名的更多信息，请参阅中的签署 Amazon API 请求Amazon Web Services 一般参考。

可用的库

Amazon 为喜欢使用特定语言的 API 而不是命令行工具和 Query API 来构建应用程序的软件开发人员提供了库、示例代码、教程和其他资源。这些库提供了基本功能（未包含在 API 中），例如请求身份验证、请求重试和错误处理，因此可以更轻松地入门。已验证权限库和资源适用于以下语言和平台：

- 适用于 Go 的 Amazon SDK
- Amazon 适用于 Java 2.x 的 SDK
- 适用于 Java 的 Amazon SDK 1.x

- 适用于 JavaScript 的 Amazon SDK
- 适用于 .NET 的 Amazon SDK
- 适用于 PHP 的 Amazon SDK
- Amazon SDK for Python (Boto)
- 适用于 Ruby 的 Amazon SDK

有关所有语言的库和示例代码的更多信息，请参阅示例代码和库。

使用 P OST ###出 API 请求

如果您不使用其中一个 Amazon SDK，则可以使用请求方法通过 HTTPS 发出已验证权限POST请求。该POST方法要求您在请求标头中指定操作，并在请求正文中以 JSON 格式提供操作数据。

标头名称	标头值
Host	Amazon 已验证权限终端节点。例如：verifiedpermissions.us-east-1.amazonaws.com
X-Amz-Date	您必须在 HTTP 日期标头或 Amazon x-am z-date 标头中提供时间戳。某些 HTTP 客户端库不允许您设置日期标头。如果存在 x-amz-date 标头，则系统会在请求身份验证期间忽略任何 Date 标头。 x-amz-date 标头必须以 ISO 8601 基本格式指定。例如：20130315T092054Z
Authorization	Amazon 用于确保请求有效性和真实性的一组授权参数。有关构造此标头的更多信息，请参阅中的签名版本 4 签名流程Amazon Web Services 一般参考。
X-Amz-Target	指定要执行的 “已验证权限” 操作。 VerifiedPermissions. *API_Name* 例如，要调用该CreatePolicy 操作，请使用以下目标值。 VerifiedPermissions.CreatePolicy
Content-Type	指定输入格式。使用以下值。

标头名称	标头值
	application/x-amz-json-1.0
Accept	指定响应格式。使用以下值。 application/x-amz-json-1.0
Content-Length	有效负载的大小（以字节为单位）。
Content-Encoding	指定输入和输出的编码格式。使用以下值。 amz-1.0

以下是 HTTP 请求的标头示例，该请求返回指定策略存储区中所有策略的列表， Amazon Web Services 账户 其中Principal引用了 name User d alice。在此示例中，为了便于Authorization阅读，此处用文字换行。不要在你的实际请求中用文字换行。

```
POST / HTTP/1.1
Host: verifiedpermissions.us-east-1.amazonaws.com
X-Amz-Date: 20230101T200059Z
Accept-Encoding: identity
Content-Type: application/x-amz-json-1.0
X-Amz-Target: VerifiedPermissions.ListPolicies
User-Agent: <UserAgentString>
Authorization: AWS4-HMAC-SHA256 Credential=<Credential>, SignedHeaders=<Headers>,
 Signature=<Signature>
Content-Length: <PayloadSizeBytes>

{
    "PolicyStoreId": "PSExAmPLE222222222",
    "Filter": {
        "Principal": {
            "Id": {
                "EntityType": "User",
                "EntityId": "alice"
            }
        }
    }
}
```

Amazon Verified Permissions 的安全性

云安全 Amazon 是重中之重。作为 Amazon 客户，您可以受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

安全是双方共同承担 Amazon 的责任。责任共担模式将其描述为云的安全性和云中的安全性：

- 云安全 — Amazon 负责保护在云中运行 Amazon 服务的基础架构 Amazon Web Services 云。Amazon 还为您提供可以安全使用的服务。作为的一部分，第三方审计师定期测试和验证我们安全的有效性。要了解适用于 Amazon Verified 权限的合规计划，请参阅合规计划划分的范围内的服务。
- 云端安全-您的责任由您使用的 Amazon 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。您还将学习如何使用其他 Amazon 服务来帮助您监控和保护您的已验证权限资源。

主题

Amazon Verified Permissions 中的数据保护

分 Amazon 分担责任模型适用于亚马逊验证权限中的数据保护。如该模式中所述， Amazon 负责保护运行所有 Amazon Web Services 云的全球基础结构。您负责维护对托管在此基础结构上的内容的控制。此内容包括您 Amazon Web Services 服务 使用的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。

- 出于数据保护目的，我们建议您保护 Amazon Web Services 账户 凭据并使用 Amazon IAM Identity Center 或 Amazon Identity and Access Management (IAM) 设置个人用户。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）。

  - 用于 SSL/TLS 与 Amazon 资源通信。我们要求使用 TLS 1.2。
  - 使用设置 API 和用户活动日志 Amazon CloudTrail。
  - 使用 Amazon 加密解决方案以及其中的所有默认安全控件 Amazon Web Services 服务。
  - 使用高级托管安全服务 Amazon Macie，例如，它有助于发现和保护存储在中的敏感数据 Amazon S3。
  - 如果在通过命令行界面或 API 访问 Amazon 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（如名称字段）。这包括当您使用已验证权限或其他 Amazon Web Services 服务 方式使用控制台 Amazon CLI、API 或时 Amazon SDKs。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，强烈建议您不要在网址中包含凭证信息来验证对该服务器的请求。
- 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。我们建议您使用诸如通用唯一标识符 (UUIDs) 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 会自动加密所有客户数据，例如使用政策。 Amazon 托管式密钥 Amazon 验证权限还允许客户使用 a 客户托管式密钥 来加密其数据。

有关使用客户托管密钥进行加密的详细信息，请参阅the section called “客户自主管理型密钥”。

在 Amazon 已验证权限中加密资源

Amazon Verified Permissions 默认提供加密，以使用 Amazon 自有的加密密钥保护敏感的静态客户数据。作为额外的保护层，Amazon Verified Permissions 允许您使用 Amazon Key Management Service (Amazon KMS) 客户托管式密钥 s (CMK) 加密您的策略存储。此功能可确保通过静态加密保护敏感数据，从而帮助您：

- 减轻应用程序端的运营负担，以保护敏感数据
- 控制谁可以通过您自己的权限查看您的授权策略 Amazon KMS 客户托管式密钥的详细信息
- 构建符合严格的加密合规性和法规要求的安全敏感型应用程序

以下各节说明如何为新的策略存储配置加密以及如何管理加密密钥。

Amazon KMS Amazon 验证权限的密钥类型

Amazon 已验证权限 Amazon KMS 与集成，用于管理用于 encrypting/decrypting 客户数据的加密密钥。要了解有关密钥类型和状态的更多信息，请参阅《Amazon KMS Developer Guide》中的 Amazon Key Management Service concepts。创建新的策略存储时，可以从以下 Amazon KMS 密钥类型中选择对数据进行加密：

Amazon 自有密钥

默认加密类型。Amazon Verified Permissions 拥有密钥，您无需支付任何额外费用，并且会在创建时对静态资源数据进行加密。使用已验证权限所拥有的密钥，无需在代码或应用程序中对数据进行额外配置。 encrypt/decrypt

客户托管密钥

您在 Amazon 账户中创建、拥有和管理密钥。你可以完全控制 Amazon KMS 钥匙。 Amazon KMS 客户托管式密钥 s 需收费。有关更多信息，请参阅 Amazon KMS 定价页面。有关密钥类型的更多信息，请参阅《Amazon KMS 开发人员指南》中的客户托管密钥。

当您 客户托管式密钥 为顶级资源（即策略存储）指定加密时，Verified Permissions 会使用该密钥对资源及其子资源进行加密。要使用加密策略存储 客户托管式密钥，您需要在密钥策略中授予对已验证权限的访问权限。密钥策略是一种基于资源的策略，您可以将其附加到您的策略上， 客户托管式密钥 以控制对该策略的访问权限。有关更多信息，请参阅the section called “授权使用您的 Amazon KMS 密钥获得 Amazon 验证权限”。

此外，要创建带有的加密策略存储库 客户托管式密钥，或者要对由加密的策略存储进行 API 调用 客户托管式密钥，发出调用的 IAM 用户或角色还必须有权访问密钥。如果已验证权限无法访问密钥，则任

何涉及由该密钥加密的资源的授权决策都可能过时或不准确。当您无法访问密钥时，您将无法访问通过该密钥加密的read/update/delete资源，并且任何使用该密钥进行加密的创建调用都将失败。

> **Note**
>
> 所有提供已验证权限的 Amazon 区域均提供已验证权限的静态加密。

> **Important**
>
> 一旦使用 客户托管式密钥 对策略存储进行加密，就无法更新资源以使用其他密钥进行加密，也无法从该策略存储中移除该密钥。

使用 Amazon KMS 具有 Amazon 验证权限的数据密钥

Amazon Verified Permissions 静态加密功能使用密 Amazon KMS 钥和数据密钥层次结构来保护您的资源数据。

> **Note**
>
> Amazon 验证权限仅支持对称 Amazon KMS 密钥。您不能使用非对称 Amazon KMS 密钥来加密您的 Amazon 验证权限资源。

使用 Amazon 自有密钥

默认情况下，Amazon 已验证权限使用 Amazon 自有密钥加密所有资源。这些密钥可免费使用，并且每年轮换，以保护您的账户资源。您无需查看、管理、使用或审计这些密钥，因此无需采取任何措施来保护数据。有关 Amazon 自有密钥的更多信息，请参阅《Amazon KMS 开发者指南》中的Amazon 自有密钥。

使用客户托管密钥

选择 客户托管式密钥 进行加密有以下好处：

- 您可以创建和管理 Amazon KMS 密钥，包括设置密钥策略和 IAM 策略以控制对 Amazon KMS 密钥的访问。您可以启用和禁用 Amazon KMS 密钥、启用和禁用自动密钥轮换，以及在不再使用 Amazon KMS 密钥时将其删除。

- 您可以在导入 客户托管式密钥 的密钥材料 客户托管式密钥 中使用，也可以在您拥有和管理的自定义密钥存储中使用。
- 您可以通过检查日志 Amazon KMS 中的 Amazon CloudTrail Amazon 已验证权限 API 调用来审核您的已验证权限资源的加密和解密。

让 Amazon 验证权限代表您使用您的 客户托管式密钥 s 获取encryption/decryption, you will need to add specific key policies to allow Amazon Verified Permissions to encrypt/decrypt资源。

授权使用您的 Amazon KMS 密钥获得 Amazon 验证权限

Amazon 验证权限至少需要以下权限 客户托管式密钥：

- `kms:Encrypt`
- `kms:GenerateDataKeyWithoutPlaintext`
- `kms:DescribeKey`
- `kms:ReEncryptTo`
- `kms:ReEncryptFrom`
- `kms:Decrypt`

密钥策略示例如下：

```
{
    "Sid": "Enable AVP to use the KMS key for encrypting project J.A.K. policy
 resources",
    "Effect": "Allow",
    "Principal": {
        "Service": "verifiedpermissions.amazonaws.com"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKeyWithoutPlaintext",
        "kms:Encrypt",
        "kms:ReEncryptFrom",
        "kms:ReEncryptTo",
        "kms:DescribeKey"
    ],
    "Resource": "*"
}
```

了解源上下文

源上下文提供有关试图对给定密钥 Amazon KMS 执行操作的源调用者的信息。这通过将上下文绑定到数据源来防止加密数据的混淆或滥用。

客户可以利用源上下文作为其密钥策略的附加条件，例如以下关键政策声明：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Enable this account full access to this key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "Enable AVP to retrieve this key's metadata",
            "Effect": "Allow",
            "Principal": {
                "Service": "verifiedpermissions.amazonaws.com"
            },
            "Action": "kms:DescribeKey",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "StringLike": {
                    "aws:SourceArn": "arn:aws:verifiedpermissions::111122223333:policy-
store/*"
                }
            }
        },
        {
            "Sid": "Enable AVP to encrypt/decrypt resources utilizing this key",
            "Effect": "Allow",
            "Principal": {
                "Service": "verifiedpermissions.amazonaws.com"
            },
            "Action": [
```

```
                "kms:Decrypt",
                "kms:ReEncryptTo",
                "kms:ReEncryptFrom",
                "kms:GenerateDataKeyWithoutPlaintext",
                "kms:Encrypt"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "StringLike": {
                    "aws:SourceArn": "arn:aws:verifiedpermissions::111122223333:policy-
store/*"
                }
            }
        }
    ]
}
```

如果源账户与该密钥所在的账户相同，则此 Amazon KMS 密钥策略允许已验证权限代表您 Amazon KMS 拨打电话。在检查 CMK 密钥的 Amazon CloudTrail 审计日志时，这些值应该是可验证的。有关全局 Amazon 条件键的更多信息，请参阅使用aws:SourceArn或aws:SourceAccount条件键。

了解加密上下文

加密上下文是一组键值对，其中包含用于加密完整性检查的其他经过身份验证的数据。当您在加密数据的请求中包含加密上下文时，会以加密 Amazon KMS 方式将加密上下文绑定到加密数据。要解密数据，必须传递相同的加密上下文。

Amazon Verified Permissions 在所有 Amazon KMS 加密操作中使用相同的加密上下文，当验证权限代表您 Amazon KMS 调用 encryption/decryption 流程时，可以在 Amazon CloudTrail 日志中进行验证。默认情况下，Verified Permissions 在加密资源时使用以下加密上下文键值对：

```
{
    "aws:verifiedpermissions:policy-store-arn":
 "arn:aws:verifiedpermissions::111122223333:policy-store/PSt123456789012"
}
```

Amazon Verified Permissions 还允许您附加自定义加密上下文，作为您希望在 encryption/decryption 流程中包含的其他元数据的一部分。这意味着您的密钥策略在授予权限时可以更加精细，如下例所示：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Enable this account full access to this key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "Enable AVP to retrieve this key's metadata",
            "Effect": "Allow",
            "Principal": {
                "Service": "verifiedpermissions.amazonaws.com"
            },
            "Action": "kms:DescribeKey",
            "Resource": "*"
        },
        {
            "Sid": "Enable AVP to encrypt/decrypt resources utilizing this key",
            "Effect": "Allow",
            "Principal": {
                "Service": "verifiedpermissions.amazonaws.com"
            },
            "Action": [
                "kms:Decrypt",
                "kms:ReEncryptTo",
                "kms:ReEncryptFrom",
                "kms:GenerateDataKeyWithoutPlaintext",
                "kms:Encrypt"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "kms:EncryptionContext:aws:verifiedpermissions:policy-store-arn":
 "arn:aws:verifiedpermissions::111122223333:policy-store/*",
                    "kms:EncryptionContext:policy_owner": "Tim"
                }
            }
        }
```

```
    ]
}
```

如果加密上下文映射包含密钥，其值遵循的格式`arn:aws:verifiedpermissions::111122223333:policy-store/*`并且还包含键值对`aws:verifiedpermissions:policy-store-arn`，则此密钥策略允许已验证权限代表您 Amazon KMS 拨打电话。`"policy_owner": "Tim"`the section called “创建加密策略存储”有关如何设置自定义加密上下文的信息，请参阅。

> **Note**
>
> 建议将具有基于加密上下文的条件的密钥策略用于加密上下文映射的子集，而不是检查每个密钥值对。上游的服务及其依赖项可能会添加您看不到的其他键值对，如果密钥策略根据加密上下文映射的确切外观有条件地允许，则可能会影响已验证权限的密钥访问权限。

理解 kms: ViaService

`kms:ViaService`条件密钥将密 Amazon KMS 钥的使用限制为来自指定 Amazon 服务的请求。此条件密钥仅适用于转发访问会话 (FAS)。有关的更多信息`kms:ViaService`，请参阅《Amazon KMS 开发人员指南》ViaService中的 kms:。

例如，以下关键政策声明使用`kms:ViaService`条件密钥，仅当请求来自代表的美国东部（弗吉尼亚北部）地区的 Amazon Verified Permissions 时，才允许将用于指定操作`BrentRole`。客户托管式密钥

```
{
    "Sid": "Enable AVP to encrypt/decrypt resources using credentials of BrentRole",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/BrentRole"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKeyWithoutPlaintext",
        "kms:Encrypt",
        "kms:ReEncryptFrom",
        "kms:ReEncryptTo",
        "kms:DescribeKey"
    ],
    "Resource": "*",
```

```
    "Condition": {
        "StringEquals": {
            "kms:ViaService": [
                "verifiedpermissions.us-east-1.amazonaws.com"
            ]
        }
    }
}
```

当Verified Permissions代表您请求加密/解密时，为了使已验证的权限能够传递您的身份、权限和会话属性，这是必要的。 Amazon KMS 有关 FAS 请求的更多信息，请参阅《IAM 用户指南》中的转发访问会话。

完成 Amazon KMS 密钥政策

根据前几节中的概念，这是一个密钥策略示例，它将允许 Amazon Verified Permissions 使用 CMK 进行加密/解密：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Enable this account full access to this key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "Enable AVP to retrieve this key's metadata",
            "Effect": "Allow",
            "Principal": {
                "Service": "verifiedpermissions.amazonaws.com"
            },
            "Action": "kms:DescribeKey",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "StringLike": {
```

```
                    "aws:SourceArn": "arn:aws:verifiedpermissions::111122223333:policy-
store/*"
                }
            }
        },
        {
            "Sid": "Enable AVP to encrypt/decrypt resources utilizing this key",
            "Effect": "Allow",
            "Principal": {
                "Service": "verifiedpermissions.amazonaws.com"
            },
            "Action": [
                "kms:Decrypt",
                "kms:ReEncryptTo",
                "kms:ReEncryptFrom",
                "kms:Encrypt",
                "kms:GenerateDataKeyWithoutPlaintext"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "kms:EncryptionContext:aws:verifiedpermissions:policy-store-arn":
 "arn:aws:verifiedpermissions::111122223333:policy-store/*",
                    "kms:EncryptionContext:policy_owner": "Tim",
                    "aws:SourceArn": "arn:aws:verifiedpermissions::111122223333:policy-
store/*"
                },
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                }
            }
        },
        {
            "Sid": "Enable AVP to encrypt/decrypt resources using credentials of
 BrentRole",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/BrentRole"
            },
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKeyWithoutPlaintext",
                "kms:Encrypt",
                "kms:ReEncryptFrom",
```

```
                "kms:ReEncryptTo",
                "kms:DescribeKey"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": [
                        "verifiedpermissions.us-east-1.amazonaws.com"
                    ]
                },
                "StringLike": {
                    "kms:EncryptionContext:aws:verifiedpermissions:policy-store-arn":
 "arn:aws:verifiedpermissions::111122223333:policy-store/*",
                    "kms:EncryptionContext:policy_owner": "Tim"
                }
            }
        }
    ]
}
```

> **Warning**
>
> 修改 Amazon KMS 已由 Amazon 验证权限使用的密钥的密钥策略时，请谨慎行事。虽然在顶级资源创建期间最初配置密 Amazon KMS 钥时，Verified Permissions 会验证加密和解密权限，但它无法根据需要验证后续的策略更改。无意中删除必要的权限可能会干扰您的授权决策和常规的已验证权限服务流程。有关排除与 Amazon 验证权限中相关的 客户托管式密钥常见错误的指南，请参阅the section called “对 Amazon 验证权限中的客户托管密钥进行故障排除”。

加密资源的必要 IAM 策略

通过账户中的 IAM 角色调用 Verified Permissions 的客户需要确保相应的 IAM 策略具有适当的权限才能使用 客户托管式密钥 对资源进行加密和解密。

为了创建由加密的策略存储 客户托管式密钥，以下 IAM 策略说明了这样做所需的最低限度操作 Amazon KMS 和已验证权限操作：

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

```
        {
            "Action": "verifiedpermissions:CreatePolicyStore",
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "kms:Decrypt",
                "kms:Encrypt",
                "kms:ReEncryptTo",
                "kms:ReEncryptFrom",
                "kms:DescribeKey",
                "kms:GenerateDataKeyWithoutPlaintext"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

> **Note**
>
> 要检索（Get* 和 List* 操作）和删除由加密的策略存储 客户托管式密钥，无需其他权限。

要更新由 a 加密的策略存储 客户托管式密钥、检索（获取* 和 List* 操作）、更新和删除由 a 加密的策略存储的子资源 客户托管式密钥，以下 IAM 策略说明了最低限度的必要操作 Amazon KMS 和 “已验证权限” 操作的执行方式：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "verifiedpermissions:*",
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "kms:Decrypt"
```

```
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

作为单一 IAM 策略，客户只需在其 IAM 角色策略中添加以下内容即可：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "verifiedpermissions:*",
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "kms:Decrypt",
                "kms:Encrypt",
                "kms:ReEncryptTo",
                "kms:ReEncryptFrom",
                "kms:DescribeKey",
                "kms:GenerateDataKeyWithoutPlaintext"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

管理加密策略存储

策略存储是入门级容器，将包含所有相关的策略资源。有关策略存储和子资源层次结构的更多信息，请参阅《亚马逊验证权限用户指南》中的 Amazon 已验证权限策略存储。

在已验证权限中创建策略存储时，您可以使用 Amazon KMS 密钥启用静态加密。这可确保：

- 对策略存储及其子资源进行的所有读取、更新和删除操作都将使用提供的 客户托管式密钥 解密过程

- 任何授权决策调用（即 IsAuthorized BatchIsAuthorized IsAuthorizedWithToken、等）都将使用提供的 客户托管式密钥 解密过程

创建加密策略存储

在创建加密策略存储库之前，请确保 客户托管式密钥 您正在使用的已为 Amazon Verified Permissions 设置了正确的密钥策略声明，以便使用该密钥进行加密/解密。the section called “授权使用您的 Amazon KMS 密钥获得 Amazon 验证权限”有关哪些权限是必需的，请参阅。

使用 Amazon CLI：

```
aws verifiedpermissions create-policy-store --region us-east-1 --encryption-settings
 file://encrypted.json --validation-settings "{\"mode\": \"OFF\"}"
```

哪里`encrypted.json`看起来像：

```
{
    "kmsEncryptionSettings": {
        "key": "arn:aws:kms:us-east-1:111122223333:key/12345678-90ab-cdef-ghij-
klmnopqrstuv",
        "encryptionContext": {
            "<ENCRYPTION_CONTEXT_KEY_1>": "<ENCRYPTION_CONTEXT_VALUE_1>",
            "<ENCRYPTION_CONTEXT_KEY_2>": "<ENCRYPTION_CONTEXT_VALUE_2>",
            ...
        }
    }
}
```

确保`key`用你的 客户托管式密钥 ARN 替换，然后用所需的`encryptionContext`键值对替换`<ENCRYPTION_CONTEXT_KEY>`和`<ENCRYPTION_CONTEXT_VALUE>`配对。`encryptionContext`如果不需要添加键值对，则可以完全省略。

> **Important**
>
> 请勿在自定义加密上下文`aws:verifiedpermissions:policy-store-arn`中包含密钥值对。这是自动添加的，如果它是您传递的自定义加密上下文键值对的一部分，则会导致验证错误。

有关策略库可用 APIs 子资源的更多信息，请参阅 Amazon Verified Permissions API 参考指南中的操作。

> **Note**
>
> 如果您的 Amazon Verified Permissions 资源由于密 Amazon KMS 钥策略不正确而被删除、禁用或无法访问，则资源解密将失败，从而导致授权决策过时。 Amazon KMS 客户托管式密钥 访问丢失可能是暂时的（可以更正密钥策略）或永久的（已删除的密钥无法恢复），具体取决于情况。我们建议您限制对关键操作的访问权限，例如删除或禁用 Amazon KMS 密钥。此外，我们建议您的组织设置Amazon 漏洞访问程序，以确保您的特权用户能够 Amazon 在无法访问 Amazon Verified Permissions 的情况下进行访问。

监控 Amazon 已验证的权限与之互动 Amazon KMS

您可以监控 Amazon Verified Permissions 对您的 客户托管式密钥 直通的使用情况。 Amazon CloudTrail Amazon KMS 通过 Verified Permissions 发出的每个请求都包括加密上下文和请求参数中使用的密钥 ARN（您的 客户托管式密钥）：

以下内容的 Amazon CloudTrail 日志条目示例`GenerateDataKeyWithoutPlaintext`：

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "verifiedpermissions.amazonaws.com"
    },
    "eventTime": "2025-09-28T16:51:04Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKeyWithoutPlaintext",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "verifiedpermissions.amazonaws.com",
    "userAgent": "verifiedpermissions.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-east-1:111122223333:key/abcdefgh-0123-ijkl-4567-
mnopqrstuvwx",
        "encryptionContext": {
            "aws:verifiedpermissions:policy-store-arn":
 "arn:aws:verifiedpermissions::111122223333:policy-store/PSt123456789012",
            "policy_store_editor": "Janus"
        },
```

```
        ...
    },
    ...
}
```

以下内容的 Amazon CloudTrail 日志条目示例Decrypt：

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "verifiedpermissions.amazonaws.com"
    },
    "eventTime": "2025-09-28T16:53:21Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "verifiedpermissions.amazonaws.com",
    "userAgent": "verifiedpermissions.amazonaws.com",
    "requestParameters": {
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "keyId": "arn:aws:kms:us-east-1:111122223333:key/abcdefgh-0123-ijkl-4567-
mnopqrstuvwx",
        "encryptionContext": {
            "aws:verifiedpermissions:policy-store-arn":
 "arn:aws:verifiedpermissions::111122223333:policy-store/PSt123456789012",
            "policy_store_owner": "Elias"
        }
    },
    ...
}
```

以下内容的 Amazon CloudTrail 日志条目示例ReEncrypt：

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "verifiedpermissions.amazonaws.com"
    },
    "eventTime": "2025-09-28T16:51:04Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "ReEncrypt",
```

```
    "awsRegion": "us-east-1",
    "sourceIPAddress": "verifiedpermissions.amazonaws.com",
    "userAgent": "verifiedpermissions.amazonaws.com",
    "requestParameters": {
        "sourceKeyId": "arn:aws:kms:us-east-1:111122223333:key/abcdefgh-0123-ijkl-4567-
mnopqrstuvwx",
        "destinationEncryptionContext": {
            "aws:verifiedpermissions:policy-store-arn":
 "arn:aws:verifiedpermissions::111122223333:policy-store/PSt123456789012"
        },
        "sourceEncryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "destinationKeyId": "arn:aws:kms:us-east-1:111122223333:key/abcdefgh-0123-
ijkl-4567-mnopqrstuvwx",
        "sourceEncryptionContext": {
            "aws:verifiedpermissions:policy_store_arn":
 "arn:aws:verifiedpermissions::111122223333:policy-store/PSt123456789012"
        },
        "destinationEncryptionAlgorithm": "SYMMETRIC_DEFAULT",
        ...
    },
    ...
}
```

请注意，日志条目包括invokedBy引用 Amazon Verified Permissions 的主体，并包含encryptionContext/sourceEncryptionContext/destinationEncryptionContext在地图中。requestParameters

以下内容的 Amazon CloudTrail 日志条目示例DescribeKey：

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "verifiedpermissions.amazonaws.com"
    },
    "eventTime": "2025-09-28T16:51:02Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DescribeKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "verifiedpermissions.amazonaws.com",
    "userAgent": "verifiedpermissions.amazonaws.com",
    "requestParameters": {
```

```
        "keyId": "arn:aws:kms:us-east-1:111122223333:key/abcdefgh-0123-ijkl-4567-
mnopqrstuvwx"
    },
    ...
}
```

请注意，日志条目包括invokedBy引用 Amazon 已验证权限的主体。

有关 Amazon CloudTrail 日志条目的更多信息，请参阅Amazon CloudTrail 用户指南中的了解 Amazon CloudTrail 事件。

限制

本主题介绍已验证权限的当前限制，以及使用 客户托管式密钥对资源进行加密。

- 策略存储一旦启用，就无法禁用加密
- 创建未加密的策略存储后，您无法将策略存储更新为由加密 客户托管式密钥
- 撤销 客户托管式密钥 对现有加密策略存储的已验证权限访问权限后，可能会出现过时的授权决策
- 使用创建策略存储后 客户托管式密钥，您无法修改自定义加密上下文值；这些值是在创建加密策略存储时设置的静态值

对 Amazon 验证权限中的客户托管密钥进行故障排除

本主题描述了您在使用 Amazon Verified 权限时可能遇到的常见 客户托管式密钥 相关错误，并提供了解决这些错误的疑难解答步骤。

访问被拒绝： Amazon KMS 权限问题

错误：“服务或调用方无权使用提供的 Amazon KMS 密钥，因为该区域不存在该资源，没有基于资源的策略允许访问，或者基于资源的策略明确拒绝访问”

这可能意味着服务或调用方在其 IAM 策略/Amazon KMS 密钥策略中缺少所需的kms:*操作权限，或者所引用的密钥不存在或已不存在。

故障排除 Amazon CloudTrail：

- 在中查找kms.amazonaws.com活动 Amazon CloudTrail
- 搜索被确定为不允许的 Amazon KMS 操作的事件名称（即DecryptReEncryptGenerateDataKeyWithoutPlaintext、DescribeKey、等）

- 查看 `errorCode` 和 `errorMessage` 字段
- 检查 `userIdentity` 以确认是哪个主体尝试了该操作

要解决此问题，请在其 IAM 策略和 Amazon KMS 密钥策略中向用户或 IAM 委托人授予适当的 Amazon KMS 操作访问权限。有关更多信息，请参阅 the section called “完成 Amazon KMS 密钥政策”。

验证异常： Amazon KMS 密钥配置

错误：“已配置的 Amazon KMS 密钥没有有效的配置”

这意味着，由于其当前配置，服务无法使用所引用的密钥进行 客户托管式密钥 加密。原因可能包括密钥被禁用、密钥不支持 EncryptionAlgorithm或密钥的类型不受支持 KeyUsage 。

限制异常： Amazon KMS 速率限制

错误：“您已超出通话费率 Amazon KMS”

此错误表示您的密钥已超出加密操作的 Amazon KMS 限制：https://docs.aws.amazon.com/kms/latest/developerguide/requests-per-second.html。

相关信息

- 管理经过验证的权限策略存储
- Amazon KMS 最佳实践
- Amazon KMS 加密上下文
- Amazon CloudTrail 集成
- Amazon CloudTrail 日志条目示例

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

Amazon Identity and Access Management (IAM) Amazon Web Services 服务 可以帮助管理员安全地控制对 Amazon 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。 Amazon Web Services 服务

主题

- 受众
- 使用身份进行身份验证

- 使用策略管理访问
- Amazon 已验证权限的工作原理 IAM
- IAM 已验证权限的策略
- 适用于 Amazon Verified Permissions 的基于身份的策略示例
- Amazon Amazon 已验证权限的托管策略
- Amazon Verified Permissions 身份和访问问题排查

受众

你使用 Amazon Identity and Access Management (IAM) 的方式因你的角色而异：

- 服务用户：如果您无法访问功能，请从管理员处请求权限（请参阅Amazon Verified Permissions 身份和访问问题排查）
- 服务管理员：确定用户访问权限并提交权限请求（请参阅Amazon 已验证权限的工作原理 IAM）
- IAM 管理员-编写用于管理访问权限的策略（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 Amazon 使用身份凭证登录的方式。您必须以 IAM 用户身份进行身份验证，或者通过担任 IAM 角色进行身份验证。 Amazon Web Services 账户根用户

对于编程访问， Amazon 提供 SDK 和 CLI 来对请求进行加密签名。有关更多信息，请参阅《IAM 用户指南》中的 API 请求Amazon 签名版本 4。

Amazon Web Services 账户 root 用户

创建时 Amazon Web Services 账户，首先会有一个名为 Amazon Web Services 账户 root 用户的登录身份，该身份可以完全访问所有资源 Amazon Web Services 服务 和资源。我们强烈建议不要使用根用户进行日常任务。有关需要 root 用户凭据的任务，请参阅《用户指南》中的 “需要根用户凭据的IAM 任务”。

联合身份

作为最佳实践，要求人类用户使用与身份提供商的联合身份验证才能 Amazon Web Services 服务 使用临时证书进行访问。

联合身份是指来自您的企业目录、Web 身份提供商的用户 Amazon Directory Service ，或者 Amazon Web Services 服务 使用来自身份源的凭据进行访问的用户。联合身份代入可提供临时凭证的角色。

IAM 用户和群组

IAM 用户是对某个人员或应用程序具有特定权限的一个身份。建议使用临时凭证，而非具有长期凭证的 IAM 用户。有关更多信息，请参阅《用户指南》中的要求人类用户使用与身份提供商的联合身份验证才能 Amazon 使用临时证书进行访问。IAM

IAM 群组指定 IAM 用户的集合，便于管理大量用户的权限。有关更多信息，请参阅《用户指南》中的 IAM IAM 用户用例。

IAM 角色

IAM 角色是您内部具有特定权限 Amazon Web Services 账户 的身份。它类似于 IAM 用户，但与特定人员不关联。您可以 Amazon Web Services 管理控制台 通过切换 IAM 角色在中临时扮演角色。您可以通过调用 Amazon CLI 或 Amazon 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。有关用于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（联合）创建角色。
- 临时 IAM 用户权限 — IAM 用户或 IAM 角色可以代入一个角色，为特定任务临时获得不同的权限。
- 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 Amazon Web Services 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 Amazon CLI 或 Amazon 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 Amazon 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 Amazon 通过创建策略并将其附加到 Amazon 身份或资源来控制中的访问权限。策略定义了与身份或资源关联时的权限。 Amazon 在委托人提出请求时评估这些政策。大多数策略都以 JSON 文档的 Amazon 形式存储在中。有关 JSON 策略文档的更多信息，请参阅IAM 用户指南中的 JSON 策略概述。

管理员使用策略，通过定义哪个主体可以在什么条件下对哪些资源执行哪些操作来指定谁有权访问什么。

默认情况下，用户和角色没有权限。 IAM 管理员创建 IAM 策略并将其添加到角色中，然后用户可以担任这些角色。 IAM 无论使用何种方法执行操作，策略都会定义权限。

基于身份的策略

基于身份的策略是您附加到身份（用户、组或角色）的 JSON 权限策略文档。这些策略控制身份可以执行什么操作、对哪些资源执行以及在什么条件下执行。要了解如何创建基于身份的策略，请参阅IAM 用户指南中的使用客户托管策略定义自定义 IAM 权限。

基于身份的策略可以是内联策略（直接嵌入到单个身份中）或托管策略（附加到多个身份的独立策略）。要了解如何在托管策略和内联策略之间进行选择，请参阅《IAM 用户指南》中的在托管策略和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。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。您必须在基于资源的策略中指定主体。

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 Amazon 托管策略。

访问控制列表 (ACLs)

访问控制列表 (ACLs) 控制哪些委托人（账户成员、用户或角色）有权访问资源。 ACLs 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 Amazon WAF、和 Amazon VPC 是支持的服务示例 ACLs。要了解更多信息 ACLs，请参阅《亚马逊简单存储服务开发者指南》中的访问控制列表 (ACL) 概述。

其他策略类型

Amazon 支持其他策略类型，这些策略类型可以设置更常见的策略类型授予的最大权限：

- 权限边界-设置基于身份的策略可以授予实体的最大权限。 IAM 有关更多信息，请参阅《IAM 用户指南》中的IAM 实体的权限边界。
- 服务控制策略 (SCPs)-在中指定组织或组织单位的最大权限 Amazon Organizations。有关更多信息，请参阅《Amazon Organizations 用户指南》中的服务控制策略。
- 资源控制策略 (RCPs)-设置账户中资源的最大可用权限。有关更多信息，请参阅《Amazon Organizations 用户指南》中的资源控制策略 (RCPs)。
- 会话策略 – 在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。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 Amazon 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
ACLs	否

IAM 功能	支持 Verified Permissions
ABAC（策略中的标签）	是
临时凭证	是
主体权限	是
服务角色	否
服务关联角色	否

要全面了解已验证的权限和其他 Amazon 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Amazon 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 用户指南中的使用客户托管策略定义自定义 IAM 权限。

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。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 Amazon Web Services 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。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 Amazon 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。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 Amazon 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。作为最佳实践，请使用其 Amazon 资源名称（ARN）指定资源。对于不支持资源级权限的操作，请使用通配符 (*) 指示语句应用于所有资源。

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

要查看已验证权限资源类型及其列表 ARNs，请参阅《服务授权参考》中的 Amazon 已验证权限定义的资源类型。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 Amazon 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。

`Condition` 元素根据定义的条件指定语句何时执行。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。要查看所有 Amazon 全局条件键，请参阅《IAM 用户指南》中的Amazon 全局条件上下文密钥。

ACLs 在 “已验证权限” 中

支持 ACLs	否

访问控制列表 (ACLs) 控制哪些委托人（账户成员、用户或角色）有权访问资源。 ACLs 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）	是

基于属性的访问权限控制（ABAC）是一种授权策略，该策略基于称为标签的属性来定义权限。您可以将标签附加到 IAM 实体和 Amazon 资源，然后设计 ABAC 策略以允许在委托人的标签与资源上的标签匹配时进行操作。

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 “使用 ABAC 授权定义权限”。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

临时证书提供对 Amazon 资源的短期访问权限，并且是在您使用联合身份或切换角色时自动创建的。Amazon 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 用户指南》中的 IAM 中的临时安全证书以及与 IAM之配合Amazon Web Services 服务 使用的临时安全证书。

Verified Permissions 的跨服务主体权限

支持主体权限	是

转发访问会话 (FAS) 使用调用主体的权限 Amazon Web Services 服务，再加上 Amazon Web Services 服务 向下游服务发出请求的请求。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》 Amazon Web Services 服务中的创建角色以向委派权限。

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 Amazon Web Services 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 Amazon Web Services 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Amazon 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。

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 APIs 或允许 Amazon Web Services 管理控制台 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。

Action	描述
CreateIdentitySource	创建新身份源的操作。
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。
CreatePolicyStore	创建新策略存储的操作。
CreatePolicyTemplate	创建新策略模板的操作。

Action	描述
DeleteIdentitySource	删除身份源的操作。
DeletePolicy	从策略存储中删除策略的操作。
DeletePolicyStore	删除策略存储的操作。
DeletePolicyTemplate	删除策略模板的操作。
GetIdentitySource	获取身份来源的操作。
GetPolicy	检索有关指定策略信息的操作。
GetPolicyStore	检索有关指定策略存储的信息的操作。
GetPolicyTemplate	获取策略模板的操作。
GetSchema	获取架构的操作。
IsAuthorized	根据授权请求中描述的参数获取授权响应的操作。
IsAuthorizedWithToken	根据授权请求中描述的参数获取授权响应的操作，其中委托人来自身份令牌。
ListIdentitySources	列出中所有身份源的操作 Amazon Web Services 账户。
ListPolicies	列出策略存储中所有策略的操作。
ListPolicyStores	列出中所有策略存储库的操作 Amazon Web Services 账户。
ListPolicyTemplates	列出中所有策略模板的操作 Amazon Web Services 账户。
ListTagsForResource	列出资源所有标签的操作。
PutSchema	向策略存储中添加架构的操作。

Action	描述
TagResource	向资源添加标签的操作。
UpdateIdentitySource	更新身份源的操作。
UpdatePolicy	在策略存储中更新策略的操作。
UpdatePolicyStore	更新策略存储库的操作。
UpdatePolicyTemplate	更新策略模板的操作。
UntagResource	从资源中移除标签的操作。

CreatePolicy 操作权限 IAM 策略示例：

JSON

```
{
"Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "verifiedpermissions:CreatePolicy"
            ],
            "Resource": "*"
        }
    ]
 }
```

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 Amazon Web Services 管理控制台、 Amazon Command Line Interface (Amazon CLI) 或 Amazon 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。

有关由已验证权限定义的操作和资源类型（包括每种资源类型的格式）的详细信息，请参阅《服务授权参考》中的 Amazon 已验证权限的操作、资源和条件密钥。 ARNs

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 Amazon Web Services 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 Amazon 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Amazon 托管策略。它们在你的版本中可用 Amazon Web Services 账户。我们建议您通过定义针对您的用例的 Amazon 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 Amazon 托管式策略 或 工作职能的Amazon 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 Amazon Web Services 服务的（例如）使用的，则也可以使用条件来授予对服务操作的访问权限 Amazon 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 来验证您的 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 用户指南中的使用 IAM 访问分析器验证策略。
- 需要多重身份验证 (MFA)-如果 Amazon Web Services 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 “使用 MFA 保护 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 Amazon Web Services 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 Amazon CLI 或 Amazon 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* Amazon 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式策略。此策略包括在控制台上或使用 Amazon CLI 或 Amazon 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 Amazon 已验证权限的托管策略

要向用户、群组和角色添加权限，使用 Amazon 托管策略比自己编写策略要容易得多。创建 IAM 客户托管策略以仅向您的团队提供他们所需的权限需要时间和专业知识。要快速入门，您可以使用我们的 Amazon 托管策略。这些策略涵盖常见使用案例，可在您的 Amazon Web Services 账户中使用。有关 Amazon 托管策略的更多信息，请参阅《IAM 用户指南》中的Amazon 托管策略。

Amazon 服务维护和更新 Amazon 托管策略。您无法更改 Amazon 托管策略中的权限。服务偶尔会向 Amazon 托管式策略添加额外权限以支持新特征。此类更新会影响附加策略的所有身份（用户、组和角色）。当启动新特征或新操作可用时，服务最有可能会更新 Amazon 托管式策略。服务不会从 Amazon 托管策略中移除权限，因此策略更新不会破坏您的现有权限。

此外，还 Amazon 支持跨多个服务的工作职能的托管策略。例如，ReadOnlyAccess Amazon 托管策略提供对所有 Amazon 服务和资源的只读访问权限。当服务启动一项新功能时， Amazon 会为新操作和资源添加只读权限。有关工作职能策略的列表和说明，请参阅《IAM 用户指南》中的工作职能 Amazon 托管策略。

Amazon 托管策略： AmazonVerifiedPermissionsFullAccess

`AmazonVerifiedPermissionsFullAccess`托管策略授予对已验证权限的完全访问权限。要使用 Amazon Cognito基于身份的身份源，您需要附加单独的策略，例如AmazonCognitoReadOnly策略。

JSON

```
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AccountLevelPermissions",
      "Effect": "Allow",
      "Action": [
        "verifiedpermissions:CreatePolicyStore",
        "verifiedpermissions:ListPolicyStores"
      ],
      "Resource": "*"
    },
    {
      "Sid": "PolicyStoreLevelPermissions",
      "Effect": "Allow",
      "Action": [
        "verifiedpermissions:*"
      ],
      "Resource": [
        "arn:aws:verifiedpermissions::*:policy-store/*"
      ]
    }
  ]
}
```

Amazon 托管策略： AmazonVerifiedPermissionsReadOnlyAccess

`AmazonVerifiedPermissionsReadOnlyAccess`托管策略授予对已验证权限的只读访问权限。

此策略授予对 Amazon Verified 权限的所有读取操作的访问权限，包括授权查询 APIs `IsAuthorized`和`IsAuthorizedWithToken`。

> **Note**
>
> 分别向`BatchIsAuthorized`和授`BatchIsAuthorizedWithToken`予访问权限时，将自动授予对`IsAuthorized`和`IsAuthorizedWithToken`的访问权限。

JSON

```
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AccountLevelPermissions",
      "Effect": "Allow",
      "Action": [
        "verifiedpermissions:ListPolicyStores"
      ],
      "Resource": "*"
    },
    {
      "Sid": "PolicyStoreLevelPermissions",
      "Effect": "Allow",
      "Action": [
        "verifiedpermissions:GetIdentitySource",
        "verifiedpermissions:GetPolicy",
        "verifiedpermissions:GetPolicyStore",
        "verifiedpermissions:GetPolicyTemplate",
        "verifiedpermissions:GetSchema",
        "verifiedpermissions:IsAuthorized",
        "verifiedpermissions:IsAuthorizedWithToken",
        "verifiedpermissions:ListIdentitySources",
        "verifiedpermissions:ListPolicies",
        "verifiedpermissions:ListPolicyTemplates"
      ],
      "Resource": [
        "arn:aws:verifiedpermissions::*:policy-store/*"
      ]
    }
  ]
}
```

Amazon 托管策略的已验证权限更新

查看有关自该服务开始跟踪已验证权限的 Amazon 托管策略更新以来这些变更的详细信息。要获得有关此页面变更的自动提醒，请订阅 “已验证权限文档历史记录” 页面上的 RSS feed。

更改	描述	日期
AmazonVerifiedPermissionsFullAccess - 新策略	已验证权限添加了一项新策略，允许对已验证权限进行完全访问权限。	2024 年 10 月 11 日
AmazonVerifiedPermissionsReadOnlyAccess：新策略	Verified Permissions 添加了一项新策略，允许访问亚马逊验证权限的所有读取操作，包括授权查询 APIs `IsAuthorized` 和`IsAuthorizedWithToken` 。	2024 年 10 月 11 日
已验证权限已开始跟踪更改	已验证权限开始跟踪其 Amazon 托管策略的更改。	2024 年 10 月 11 日

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
- 我无权执行 iam：PassRole
- 我想允许我以外的人访问我的 Amazon Web Services 账户 “已验证权限” 资源

我无权在 Verified Permissions 中执行操作

如果您收到错误提示，指明您无权执行某个操作，则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 Amazon 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 Amazon Web Services 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 Amazon 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 Amazon Web Services 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，可以代入角色。对于支持基于资源的策略或访问控制列表 (ACLs) 的服务，您可以使用这些策略向人们授予访问您的资源的权限。

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 Amazon Web Services 账户 个 IAM IAM 用户提供访问权限。 Amazon Web Services 账户
- 要了解如何向第三方提供对您的资源的访问权限 Amazon Web Services 账户，请参阅IAM 用户指南中的向第三方提供访问权限。 Amazon Web Services 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Amazon Web Services 服务 “” Amazon Web Services 服务 中的 “按合规计划划分的范围”，然后选择您感兴趣的合规计划。 Amazon Web Services 服务 有关一般信息，请参阅合规计划。

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

您在使用 Amazon Web Services 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。有关您在使用时的合规责任的更多信息 Amazon Web Services 服务，请参阅Amazon 安全文档。

Amazon Verified Permissions 的顺应力

Amazon 全球基础设施是围绕 Amazon Web Services 区域 可用区构建的。 Amazon Web Services 区域 提供多个物理分隔和隔离的可用区，这些可用区通过低延迟、高吞吐量和高度冗余的网络连接。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失效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结构相比，可用区具有更高的可用性、容错性和可扩展性。

当您创建经过验证的权限策略存储时，它将在个人内部创建 Amazon Web Services 区域，并自动复制到构成该区域可用区域的数据中心。目前，Verified Permissions 不支持任何跨区域复制。

有关 Amazon Web Services 区域 和可用区的更多信息，请参阅Amazon 全球基础设施。

监控 Amazon 已验证权限 API 调用

监控是维护 Amazon 验证权限和其他 Amazon 解决方案的可靠性、可用性和性能的重要组成部分。Amazon 提供了以下工具来监控已验证的权限，在出现问题时进行报告，并在适当时自动采取措施：

- Amazon CloudTrail捕获由您的账户或代表您的 Amazon 账户进行的 API 调用和相关事件，并将日志文件传输到您指定的 Amazon S3 存储桶。您可以识别哪些用户和帐户拨打了电话 Amazon、发出呼叫的源 IP 地址以及呼叫发生的时间。有关更多信息，请参阅 Amazon CloudTrail 《用户指南》。

有关使用监控已验证权限的更多信息 CloudTrail，请参阅使用记录亚马逊已验证的权限 API 调用 Amazon CloudTrail。

使用记录亚马逊已验证的权限 API 调用 Amazon CloudTrail

Amazon Verifie Amazon 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 Amazon 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，则可以启用向 Amazon S3 存储桶持续交付 CloudTrail 事件，包括已验证权限的事件。如果您未配置跟踪，则仍可以在 CloudTrail 控制台的事件历史记录中查看最新的管理操作事件，但不能查看 API 调用的事件，例如`isAuthorized`。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Amazon CloudTrail 用户指南。

已验证的权限信息位于 CloudTrail

CloudTrail 在您创建账户 Amazon Web Services 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 Amazon 服务 CloudTrail 事件一起记录在事件历史记录中。您可以在 Amazon Web Services 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。

要持续记录您的事件 Amazon Web Services 账户，包括已验证权限的事件，请创建跟踪。跟踪允许 CloudTrail 将日志文件传送到 Amazon S3 存储桶。默认情况下，在控制台中创建跟踪记录时，此跟踪记录应用于所有 Amazon Web Services 区域。跟踪记录 Amazon 分区中所有区域的事件，并将日志文件传送到您指定的 Amazon S3 存储桶。此外，您可以配置其他 Amazon 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：

- 创建跟踪记录概述

- CloudTrail 支持的服务和集成
- 配置 Amazon SNS 通知 CloudTrail
- 接收来自多个地区的 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。

每个事件或日志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 root 还是 Amazon Identity and Access Management (IAM) 用户凭据发出。
- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他 Amazon 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《Amazon 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，它允许将事件作为日志文件传送到您指定的 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顺序出现。

对于授权 API 调用，响应元素（例如决策）包含在`additionalEventData`而不是下`responseElements`。

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"
}
```

使用创建 Amazon 已验证的权限资源 Amazon CloudFormation

Amazon Verified Permissions 与 Amazon CloudFormation一项服务集成，可帮助您对 Amazon 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 Amazon 资源（例如策略存储）的模板，并为您 Amazon CloudFormation 预置和配置这些资源。

使用时 Amazon 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 Amazon Web Services 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 Amazon Web Services 区域 亚马逊验证权限完全不同。如果您收到 Amazon CloudFormation 有关 Amazon Cognito Identity 的错误，例如`Unrecognized resource types: AWS::Cognito::UserPool, AWS::Cognito::UserPoolClient`，我们建议您在最接近可用 Amazon Cognito Ident Amazon Web Services 区域 ity 的地理位置创建 Amazon 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 Amazon CloudFormation 模板

要为 Verified Permissions 和相关服务预置和配置资源，您必须了解 Amazon 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 Amazon CloudFormation 堆栈中配置的资源。如果你不熟悉 JSON 或 YAML，可以使用 D Amazon CloudFormation esigner 来帮助你开始使用 Amazon CloudFormation 模板。有关更多信息，请参阅什么是 Amazon CloudFormation 设计器？ 在《Amazon CloudFormation 用户指南》中。

Verified Permissions 支持在 Amazon CloudFormation中创建身份源、策略、策略存储、策略模板和策略存储别名。有关更多信息（包括 Verified Permissions 资源的 JSON 和 YAML 模板示例），请参阅《Amazon CloudFormation 用户指南》中的 Amazon Verified Permissions 资源类型参考。

Amazon CDK 构造

Amazon Cloud Development Kit (Amazon 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 Amazon CloudFormation。构造或可重复使用的云组件可用于创建 Amazon 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 Amazon CDK，请参阅 C Amazon loud Development Kit。

以下是已验证权限 Amazon 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 Amazon CloudFormation

要了解更多信息 Amazon CloudFormation，请参阅以下资源：

- Amazon CloudFormation
- Amazon CloudFormation 用户指南
- Amazon CloudFormation API 引用
- Amazon CloudFormation 命令行界面用户指南

使用访问亚马逊验证权限 Amazon PrivateLink

您可以使用 Amazon PrivateLink 在您的 VPC 和 Amazon 验证权限之间创建私有连接。无需使用互联网网关、NAT 设备、VPN 连接或 Amazon Direct Connect 连接，即可像访问您的 VPC 一样访问经过验证的权限。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 Amazon 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《Amazon PrivateLink 指南》中的通过 Amazon PrivateLink访问 Amazon Web Services 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《Amazon 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 Amazon Command Line Interface （ Amazon CLI ）为 Verified Permissions 创建接口端点。有关更多信息，请参阅《Amazon 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`。

为 VPC 端点创建端点策略

端点策略是一种 IAM 资源，您可以将其附加到接口端点。默认终端节点策略允许通过接口端点对已验证权限进行完全访问权限。要控制允许从您的 VPC 访问已验证权限，请将自定义终端节点策略附加到接口终端节点。

端点策略指定以下信息：

- 可执行操作的主体（Amazon Web Services 账户、IAM 用户和 IAM 角色）。
- 可执行的操作。
- 可对其执行操作的资源。

有关更多信息，请参阅《Amazon PrivateLink 指南》中的使用端点策略控制对服务的访问权限。

示例：用于已验证权限操作的 VPC 终端节点策略

以下是自定义端点策略的示例。当您将此策略附加到接口终端节点时，它会向所有委托人授予所有资源上列出的已验证权限操作的访问权限。

```
{
   "Statement": [
      {
         "Principal": "*",
         "Effect": "Allow",
         "Action": [
            "verifiedpermissions:IsAuthorized",
            "verifiedpermissions:IsAuthorizedWithToken",
            "verifiedpermissions:GetPolicy"
         ],
         "Resource":"*"
      }
   ]
}
```

Amazon Verified Permissions 的配额

您的每项 Amazon 服务 Amazon Web Services 账户 都有默认配额，以前称为限制。除非另有说明，否则每个配额都是 Region-specific。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 Amazon 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。如果限额在服务限额中尚不可用，请使用提高限制表格。

您 Amazon Web Services 账户 有以下与已验证权限相关的配额。

主题

资源配额

Name	默认值	可调整	说明
每个账户在每个区域的策略存储数	每个支持的区域：30,000	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。
每个策略存储区的策略存储别名	10	是	您可以与单个策略存储关联的最大策略存储别名数。

Name	默认值	可调整	说明
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	是	单个策略的最大大小。
架构大小	100000 字节	是	策略存储区架构的最大大小。
每个策略存储架构的命名空间	100	是	您可以在策略存储架构中定义的最大命名空间数。
每个资源的策略大小	200,000 个字节[2]	是	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] 适用于单个资源的所有策略的总大小默认限制为 200,000 字节。同样，默认情况下，所有策略的总大小限制为 200,000 字节，其中范围未定义资源，因此适用于所有资源。请注意，对于模板关联策略，策略模板的大小仅计算一次，再加上用于实例化每个模板关联策略的每组参数的大小。如果您的策略设计满足某些限制，则可以提高此限制。如果您需要探索此选项，请联系 Amazon Web Services 支持。

Template-linked 策略规模示例

您可以通过计算委托人和资源长度的总和来确定模板关联策略如何影响每个资源配额的策略大小。如果未指定主体或资源，则该部分的长度为 0。如果未指定资源，则其大小将计入"unspecified"资源配额。模板正文本身的大小对策略大小没有影响。

让我们来看看下面的模板：

```
@id("template1")
permit (
  principal in ?principal,
  action in [Action::"view", Action::"comment"],
  resource in ?resource
)
unless {
```

```
  resource.tag =="private"
};
```

让我们根据该模板创建以下策略：

```
TemplateLinkedPolicy {
  policyId: "policy1",
  templateId: "template1",
  principal: User::"alice",
  resource: Photo::"car.jpg"
}

TemplateLinkedPolicy {
  policyId: "policy2",
  templateId: "template1",
  principal: User::"bob",
  resource: Photo::"boat.jpg"
}

TemplateLinkedPolicy {
  policyId: "policy3",
  templateId: "template1",
  principal: User::"jane",
  resource: Photo::"car.jpg"

TemplateLinkedPolicy {
  policyId: "policy4",
  templateId: "template1",
  principal: User::"jane",
  resource
}
```

现在，让我们通过计算principal和resource中每个策略的字符来计算这些策略的大小。每个字符计为 1 个字节。

的大小policy1将是主体的长度 User::"alice" (13) 加上资源的长度 Photo::"car.jpg" (16)。将它们加起来我们有 13 + 16 = 29 字节。

的大小policy2将是主体的长度 User::"bob" (11) 加上资源的长度 Photo::"boat.jpg" (17)。将它们加起来我们有 11 + 17 = 28 个字节。

的大小policy3将是主体的长度 User::"jane" (12) 加上资源的长度 Photo::"car.jpg" (16)。将它们加起来我们有 12 + 16 = 28 个字节。

的大小policy4将是主体的长度 User::"jane" (12) 加上资源的长度 (0)。将它们加起来我们有 12 + 0 = 12 个字节。

由于policy2是引用资源的唯一策略Photo::"boat.jpg"，因此资源总大小为 28 字节。

由于policy1policy3两者都引用资源Photo::"car.jpg"，因此资源总大小为 29 + 28 = 57 字节。

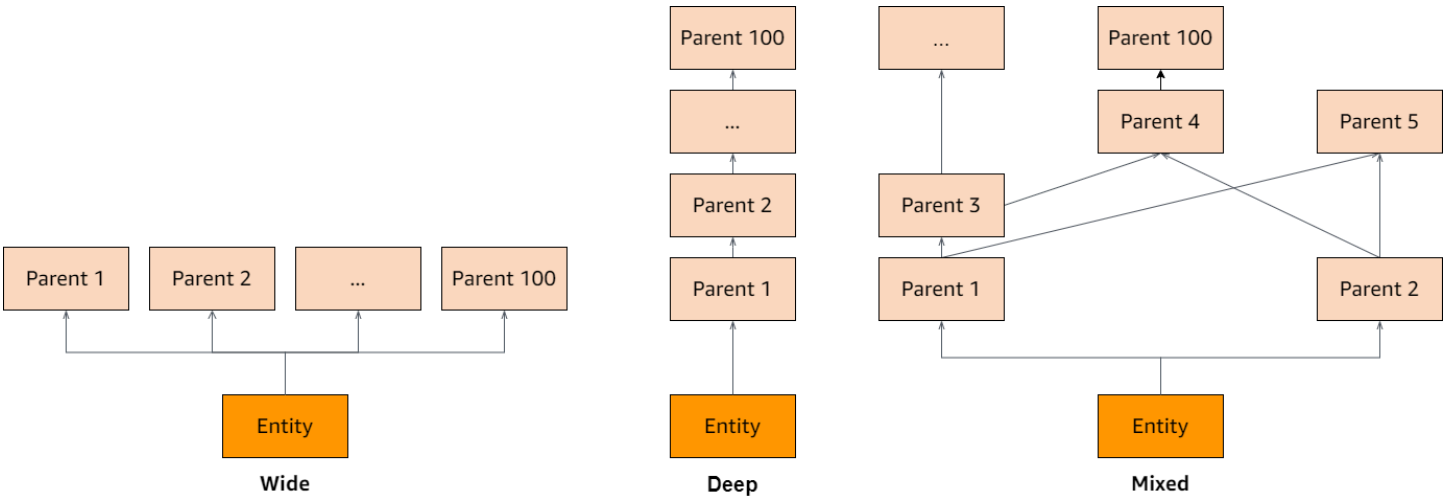
由于policy4是引用资源的唯一策略，因此"unspecified"资源总大小为 12 字节。

层次结构的配额

> **Note**
>
> 以下配额是汇总的，这意味着它们是相加在一起的。该群组中可传递父母的最大数量就是所列数字。例如，如果每位校长的传递父母限制为100，则意味着可能有100名校长的父母，0名家长负责行为和资源，或者任何父母的组合加起来为100名父母。

Name	默认值	可调整	说明
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

Amazon Web Services 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。

Name	默认值	可调整	说明
BatchGetPolicy 每个策略存储区每个区域的每秒请求数	每个受支持的区域：10 个	是	每个策略存储每秒的最大 BatchGetPolicy 请求数。
BatchIsAuthorized 每个策略存储区每个区域的每秒请求数	每个受支持的区域：30 个	是	每个策略存储每秒的最大 BatchIsAuthorized 请求数。
BatchIsAuthorizedWithToken 每个策略存储区每个区域的每秒请求数	每个受支持的区域：30 个	是	每个策略存储每秒的最大 BatchIsAuthorizedWithToken 请求数。
CreateIdentitySource 每个策略存储区每个区域的每秒请求数	每个受支持的区域：1 个	是	每个策略存储每秒的最大 CreateIdentitySource 请求数。

Name	默认值	可调整	说明
CreatePolicy 每个策略存储区每个区域的每秒请求数	每个受支持的区域：10 个	是	每个策略存储每秒的最大 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 每个策略存储区每个区域的每秒请求数	每个受支持的区域：10 个	是	每个策略存储每秒的最大 CreatePolicyTemplate 请求数。
DeleteIdentitySource 每个策略存储区每个区域的每秒请求数	每个受支持的区域：1 个	是	每个策略存储每秒的最大 DeleteIdentitySource 请求数。
DeletePolicy 每个策略存储区每个区域的每秒请求数	每个受支持的区域：10 个	是	每个策略存储每秒的最大 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 每个策略存储区每个区域的每秒请求数	每个受支持的区域：10 个	是	每个策略存储每秒的最大 DeletePolicyTemplate 请求数。
GetIdentitySource 每个策略存储区每个区域的每秒请求数	每个受支持的区域：10 个	是	每个策略存储每秒的最大 GetIdentitySource 请求数。
GetPolicy 每个策略存储区每个区域的每秒请求数	每个受支持的区域：10 个	是	每个策略存储每秒的最大 GetPolicy 请求数。
GetPolicyStore 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 GetPolicyStore 请求数。

Name	默认值	可调整	说明
GetPolicyTemplate 每个策略存储区每个区域的每秒请求数	每个受支持的区域：10 个	是	每个策略存储每秒的最大 GetPolicyTemplate 请求数。
GetSchema 每个策略存储区每个区域的每秒请求数	每个受支持的区域：10 个	是	每个策略存储每秒的最大 GetSchema 请求数。
IsAuthorized 每个策略存储区每个区域的每秒请求数	每个受支持的区域：200 个	是	每个策略存储每秒的最大 IsAuthorized 请求数。
IsAuthorizedWithToken 每个策略存储区每个区域的每秒请求数	每个受支持的区域：200 个	是	每个策略存储每秒的最大 IsAuthorizedWithToken 请求数。
ListIdentitySources 每个策略存储区每个区域的每秒请求数	每个受支持的区域：10 个	是	每个策略存储每秒的最大 ListIdentitySources 请求数。
ListPolicies 每个策略存储区每个区域的每秒请求数	每个受支持的区域：10 个	是	每个策略存储每秒的最大 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 每个策略存储区每个区域的每秒请求数	每个受支持的区域：10 个	是	每个策略存储每秒的最大 ListPolicyTemplates 请求数。
PutSchema 每个策略存储区每个区域的每秒请求数	每个受支持的区域：10 个	是	每个策略存储每秒的最大 PutSchema 请求数。
UpdateIdentitySource 每个策略存储区每个区域的每秒请求数	每个受支持的区域：1 个	是	每个策略存储每秒的最大 UpdateIdentitySource 请求数。

Name	默认值	可调整	说明
UpdatePolicy 每个策略存储区每个区域的每秒请求数	每个受支持的区域：10 个	是	每个策略存储每秒的最大 UpdatePolicy 请求数。
UpdatePolicyStore 每个账户每个区域的每秒请求数	每个受支持的区域：10 个	否	每秒的最大 UpdatePolicyStore 请求数。
UpdatePolicyTemplate 每个策略存储区每个区域的每秒请求数	每个受支持的区域：10 个	是	每个策略存储每秒的最大 UpdatePolicyTemplate 请求数。

Amazon 已验证权限和 Cedar 政策语言术语和概念

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
- 策略存储别名
- 策略名称
- 策略模板名称
- 满足条件的策略
- Amazon 验证权限与 Cedar 政策语言之间的区别

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，它包括允许或拒绝的决定，以及其他信息，例如决定性策略的信息。IDs

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。

策略存储别名

策略存储别名是策略存储的友好名称。在任何接受`policyStoreId`参数的已验证权限操作中，您可以使用策略存储别名来标识策略存储。策略存储别名是独立的 Amazon 资源，有自己的 ARNs别名。每

个别名一次只能与一个策略存储关联，多个别名可以与同一个策略存储关联。有关更多信息，请参阅 Amazon 已验证权限策略存储别名。

策略名称

策略名称是策略的可选友好名称。策略存储区内所有策略的策略名称必须是唯一的，并以name/此为前缀。在接受policyId参数的控制平面操作中，您可以使用策略名称代替策略 ID。可以在创建或更新策略时设置名称。只GetPolicy在输出中ListPolicies返回名称。

策略模板名称

策略模板名称是策略模板的可选友好名称。策略存储区内所有策略模板的策略模板名称必须是唯一的，并以name/此为前缀。在接受policyTemplateId参数的控制平面操作中，您可以使用策略模板名称代替策略模板 ID。可以在创建或更新策略模板时设置名称。只GetPolicyTemplate在输出中ListPolicyTemplates返回名称。

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。

Amazon 验证权限与 Cedar 政策语言之间的区别

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。

> **Note**
>
> 截至 2026 年 5 月，已验证权限已与 Cedar 保持一致，现在支持多个命名空间。

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。Cedar 在少数情况下允许使用空字符串，例如属性或命名空间。

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。

实体格式设置

使用`entityList`参数的 “已验证权限” 中实体的 JSON 格式与 Cedar 在以下方面有所不同：

- 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

Example-清单

以下示例分别显示了如何在 Cedar 和已验证权限中表示实体列表。

Cedar

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

Verified Permissions

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

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

Example-政策评估

以下示例分别显示了 Cedar 和 Verified Permissions 中用于评估授权请求中的策略的实体是如何格式化的。

Cedar

```
[
    {
        "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": []
    }
]
```

Verified Permissions

```
[
    {
```

```
        "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": []
    }
]
```

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	100000	无
实体类型	200	无
策略 ID	64	无
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。

Amazon 已验证权限升级到 Cedar 4 常见问题解答

Amazon Verified Permissions 正在将其使用的 Cedar 版本从版本 2 升级到版本 4。Cedar 是您用来在策略存储库中编写策略、策略模板和架构的开源语言。借助已验证权限中的 Cedar 4 支持，您可以使用`is`操作员和实体标签等新功能来编写更具表现力的策略。

Amazon 已验证权限会自动将策略存储升级到 Cedar 4。但是，为Cedar 2编写的某些策略、架构和授权请求与Cedar 4不兼容。如果您的保单存储是这种情况，那么我们不会自动对其进行升级。在升级到 Cedar 4 之前，您可能需要对策略、策略模板、架构或应用程序代码进行更改。

主题

当我致电 Amazon 验证权限时，为什么会收到一条错误消息，说不再支持 Cedar 2？

从 2026 年 4 月起，使用 Cedar 2 的策略商店将禁用授权 API（`IsAuthorized`、`IsAuthorizedWithToken`和`BatchIsAuthorizedWithT`）。在这种情况下，将返回以下错误消息：

```
Your policy store uses Cedar 2.x, which is no longer supported. See https://
docs.aws.amazon.com/verifiedpermissions/latest/userguide/cedar4-faq.html or contact AWS
 Support for more information.
```

如果您在调用 Verified Permissions 时遇到此错误消息，则表示您的策略存储与 Cedar 4 不兼容，因此无法迁移到新版本。要继续使用亚马逊验证权限，请创建一个新的政策存储库，该存储库将使用 Cedar 4，或者联系 Amazon Web Services 支持。

为什么有些策略、策略模板和架构与 Cedar 4 不兼容？

自Cedar 2以来，Cedar团队已经进行了几项不向后兼容的更改，以修复错误并简化语言。这些更改包括：

- 策略、策略模板和架构的语法更改
- 更精确的策略验证器，可以检测到更多错误
- 更改内置函数的行为，例如 `isInRange`

有关向后不兼容的更改的完整列表，请查找 Cedar 变更日志(*)中标记为的项目。

如何判断我的保单商店使用的是 Cedar 2 还是 Cedar 4？

您可以使用 Amazon Verified Permissions 控制台或使用GetPolicyStore操作来查看您的策略商店使用的 Cedar 版本。

> **Note**
>
> 同一地区的所有保单商店都使用相同 Amazon Web Services 账户 版本的 Cedar。

Console

查看策略存储库的 Cedar 版本（控制台）

1. 登录 Amazon Web Services 管理控制台 并打开 Amazon 验证权限控制台，网址为https://console.aws.amazon.com/verifiedpermissions/。
2. 在导航窗格中，选择策略存储，然后选择要检查的策略存储。
3. 在导航窗格中，选择设置。
4. 在 “详细信息” 框中，找到 Cedar 版本字段。

该字段显示您的策略存储`CEDAR_2`是否使用 Cedar 2，`CEDAR_4`是否使用 Cedar 4。

CLI

要查看策略库的 Cedar 版本 (Amazon CLI)

1. 安装并配置 Amazon Command Line Interface (Amazon CLI)（如果尚未安装）。有关更多信息，请参阅安装或更新 Amazon CLI的最新版本。
2. 使用 `get-policy-store` 命令。在以下示例中，*`policy-store-id`*使用您的策略存储库的标识符替换：

```
aws verifiedpermissions get-policy-store \
  --policy-store-id policy-store-id
```

输出中的`cedarVersion`字段显示策略存储正在使用哪个版本的 Cedar。例如：

```
{
    "policyStoreId": "ABCDEFG12345678abcdefg",
    "arn": "arn:aws:verifiedpermissions::111122223333:policy-store/
ABCDEFG12345678abcdefg",
    "validationSettings": {
        "mode": "STRICT"
    },
    "createdDate": "2025-06-03T13:09:47.752255+00:00",
    "lastUpdatedDate": "2025-06-03T13:09:47.752255+00:00",
    "deletionProtection": "ENABLED",
    "cedarVersion": "CEDAR_2"
}
```

该字段显示您的策略存储`CEDAR_2`是否使用 Cedar 2，`CEDAR_4`是否使用 Cedar 4。

如何升级到 Cedar 4？

亚马逊验证权限已将大多数客户升级到 Cedar 4。如果您从未创建过策略存储，那么您创建的任何新策略存储都将使用 Cedar 4。如果您是现有客户，那么我们可能已经将您升级到 Cedar 4。查看如何判断我的保单商店使用的是 Cedar 2 还是 Cedar 4？您的保单商店使用的是哪个版本的 Cedar。

如果您尚未升级，则验证权限会在您的一个策略存储中检测到与 Cedar 4 不兼容的策略、策略模板、架构或授权请求。我们将在 2025 年晚些时候向您发送一封电子邮件通知，说明哪些资源不兼容。要更快地升级，请使用开箱 Amazon Web Services 支持。

> **Important**
>
> 同一个保单存储库 Amazon Web Services 账户 使用相同版本的 Cedar。如果您账户中的一个保单存储与 Cedar 4 不兼容，则您不能在该账户的任何保单存储区中使用 Cedar 4。

我可以将我的保单商店从 Cedar 4 降级到 Cedar 2 吗？

不是。 如果您在保单商店升级到 Cedar 4 后遇到问题，请使用打开案例 Amazon Web Services 支持。

为什么我收到一条错误消息，说我的策略存储已配置为 Cedar 2？

Amazon 验证权限的某些功能依赖于 Cedar 4 中的新功能。如果您的政策存储库不使用 Cedar 4，则无法使用以下 API 字段：

- 在IsAuthorizedBatchIsAuthorized、IsAuthorizedWithToken和BatchIsAuthorizedWithToken操作中：
  - `datetime`，`decimal`或者`attributes`或`context`字段中的`duration`值

在升级策略存储之前，您不能在 Cedar 2 之后引入的策略、策略模板或架构中使用语法或数据类型。

如何使我的架构与 Cedar 4 兼容？

已验证权限控制台可以自动修复架构中的一些兼容性问题。如果您的架构无法自动修复，控制台将显示错误列表供您手动修复。

> **Important**
>
> 即使您的策略存储使用 Cedar 2，Amazon 验证权限控制台中的代码编辑器也始终显示来自 Cedar 4 的错误和警告。您可以使用保存更改按钮或已验证权限 API 继续进行与 Cedar 4 不兼容的架构更新。

使用控制台修复架构

1. 登录 Amazon Web Services 管理控制台 并通过 verif iedPermissions 打开亚马逊验证权限控制台。

2. 在导航窗格中，选择策略存储，然后选择要检查的策略存储。
3. 在导航窗格中选择 “架构”。
4. 如果你的架构可以自动修复，你会看到一个标语，上面写着 “点击'修复'预览兼容版本”。选择 “修复”。
5. 查看对架构所做的更改，然后单击 “预览更新的架构”。
6. 查看更新的架构，然后单击 “保存更改”。

如果您的架构无法自动修复，则可以在控制台中看到需要自行修复的错误列表。

1. 如上所述，打开 “编辑架构” 页面。
2. 选择 JSON 模式。
3. 将鼠标悬停在代码编辑器左侧边框中的红色错误图标上。错误消息显示在工具提示中。

以下是您可能遇到的一些常见错误以及如何解决这些错误：

无法从 JSON 解析架构:`*field-name*

使用 Cedar 2，您可以在架构的某些部分（例如类型定义）中包含任意字段，即使它们作为 Cedar 架构的一部分没有任何意义。在 Cedar 4 中，不再允许这样做。要解决此错误，请*field-name*从 JSON 架构中移除名为的字段。有关有效架构字段的列表，请参阅 Cedar 文档。

未知的扩展类型 `*extension-name*`

在 Cedar 2 中，当你声明的属性`type`为时`Extension`，你可以为该`name`字段指定任何值，无论该值是否为有效的扩展类型名称。现在，这是 Cedar 4 的错误。要解决此问题，请*extension-name*替换为有效的扩展类型名称。您可以在 C edar 文档中找到有效的扩展类型名称列表。

如果您仍然不确定如何解决架构中的错误，请联系 Amazon Web Services 支持

如何使我的保单和模板与 Cedar 4 兼容？

已验证权限控制台会显示您的策略或模板中存在的任何导致其与 Cedar 4 不兼容的错误。

在控制台中查看策略或模板的错误

1. 登录 Amazon Web Services 管理控制台 并通过 verif iedPermissions 打开亚马逊验证权限控制台。

2. 在导航窗格中，选择策略存储，然后选择要检查的策略存储。
3. 根据需要在导航窗格中选择策略或策略模板。
4. 选择不兼容的策略或模板。
5. 选择编辑。
6. 将鼠标悬停在代码编辑器左侧边框中的红色错误图标上。错误消息显示在工具提示中。

以下是您可能遇到的一些常见错误以及如何解决这些错误：

策略中禁止使用空集文字

在 Cedar 2 中，你可以使用语法`mySet == []`来检查集合是否为空。在 Cedar 4 中，使用此语法的策略不再针对架构进行验证。`mySet == []`在您的保单中替换为`mySet.isEmpty()`。

《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
对策略存储别名进行硬删除	现在，您可以绕过 24 小时的`PendingDeletion` 预留期，硬删除策略存储别名以便立即删除。	2026 年 4 月 17 日
策略名称和策略模板名称	现在，您可以为策略和策略模板分配名称，以便通过友好名称来引用它们。	2025 年 3 月 4 日
策略存储别名	现在，您可以创建策略存储别名，以便通过友好名称引用您的策略存储。	2025 年 2 月 26 日
新的 Amazon 托管策略	现在，您可以将`AmazonVerifiedPermissionsFullAccess` 和`AmazonVerifiedPermissionsReadOnlyAccess` IAM 托管策略与已验证权限一起使用。	2024 年 10 月 11 日
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

使用创建策略存储 API Gateway	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
Amazon 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 Amazon 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。