aws

ユーザーガイド

Amazon Verified Permissions

Amazon Verified Permissions: ユーザーガイド

Table of Contents

Amazon Verified Permissions とは何でしょうか？

Amazon Verified Permissions は、お客様が作成したカスタムアプリケーション向けの、スケーラブルできめ細かな権限管理および認可サービスです。Verified Permissions を利用すると、認可を外部化し、ポリシーの管理と管理を一元化することで、開発者は安全なアプリケーションをより迅速に構築できます。Verified Permissions は Cedar ポリシー言語を使用して、アプリケーションのリソースを保護するためのきめ細かなアクセス許可を定義します。

Verified Permissions を使用してポリシー決定ポイント (PDP) を設定するガイダンスと例については、AWS 「規範ガイダンス」の「Amazon Verified Permissions を使用した PDP の実装」を参照してください。

トピック

Amazon Verified Permissions による認可

Verified Permissions は、プリンシパルがアプリケーションの特定のコンテキストでリソースに対してアクションを実行できるかどうかを検証することで、認可を提供します。Verified Permissions は、OpenID Connect などのプロトコル、 などのホストプロバイダー、または別の認証ソリューションを使用するなど Amazon Cognito、他の方法でプリンシパルが以前に識別および認証されていることを前提としています。Verified Permissions は、プリンシパルが管理されている場所と認証方法に依存しません。

Verified Permissions は、お客様が Verified Permissions API を使用してプログラムで AWS マネジメントコンソール、または のようなコードソリューションとしてのインフラストラクチャを通じて、 でポリシーを作成、保守、テストできるようにするサービスです CloudFormation。 APIs 権限は Cedar ポリシー言語を使用して表現されます。クライアントアプリケーションは認可 API を呼び出して、サービスに保存されている Cedar ポリシーを評価し、アクションが許可されるかどうかのアクセス判断を行います。

Cedar ポリシー言語

Verified Permissions の認可ポリシーは Cedar ポリシー言語を使用して記述されます。Cedar は、認可ポリシーを記述し、そのポリシーに基づいて認可決定を行うためのオープンソース言語です。アプリケーションを作成するときは、承認されたプリンシパル、人間のユーザー、またはマシンのみがアプリケーションにアクセスし、許可された操作のみを実行できるようにする必要があります。Cedar を使えば、ビジネスロジックを認可ロジックから切り離すことができます。アプリケーションのコードでは、オペレーションに対するリクエストの前に Cedar 認可エンジンを呼び出し、「このリクエストは認可されていますか?」と尋ねます。次に、アプリケーションは、決定が「許可」の場合は要求された操作を実行し、決定が「拒否」の場合はエラーメッセージを返すことができます。

Verified Permissions は現在 Cedar バージョン 4.7 を使用しています。

Cedar の詳細については、以下を参照してください。

- Cedar ポリシー言語リファレンスガイド
- Cedar GitHub リポジトリ

Verified Permissions の利点

アプリケーションの開発を加速

認可をビジネスロジックから切り離すことで、アプリケーション開発を加速します。

Verified Permissions は一般的な開発フレームワークとの統合を提供するため、最小限のコード変更でアプリケーションに認可を簡単に実装できます。これらの統合により、Verified Permissions が認可の決定を処理しながら、コアビジネスロジックに集中できます。

- Express.js – 既存のルートハンドラーを変更せずに Express アプリケーションの API エンドポイントを保護できるミドルウェアベースの統合。詳細については、「the section called “Express の使用”」を参照してください。

より安全なアプリケーション

Verified Permissions により、デベロッパーはより安全なアプリケーションを構築できます。

エンドユーザー機能

Verified Permissions により、権限管理のためのより充実したエンドユーザー機能を提供できます。

関連サービス

- Amazon Cognito はウェブアプリとモバイルアプリ用のアイデンティティプラットフォームです。これは、OAuth 2.0 アクセストークンと AWS 認証情報のための、ユーザーディレクトリであり、認証サーバーであり、認可サービスです。ポリシーストアを作成するときに、 Amazon Cognito ユーザープールからプリンシパルとグループを構築するオプションがあります。詳細については、「Amazon Cognito デベロッパーガイド」をご覧ください。
- Amazon API Gateway - Amazon API Gatewayは、あらゆる規模の REST、HTTP、および WebSocket API を作成、公開、維持、モニターリング、およびセキュア化するための AWS サービスです。ポリシーストアを作成するときは、 API からアクションとリソースを構築するオプションがあります API Gateway。詳細については API Gateway、「 API Gateway デベロッパーガイド」を参照してください。
- AWS IAM アイデンティティセンター— IAM アイデンティティセンター を使用すると、ワークフォースユーザーとも呼ばれるワークフォース ID のサインインセキュリティを管理できます。IAM Identity Center は、ワークフォースユーザーを作成または接続し、すべての AWS アカウント およびアプリケーションへのアクセスを一元管理できる 1 つの場所を提供します。詳細については、「AWS IAM アイデンティティセンター ユーザーガイド」を参照してください。

Verified Permissions へのアクセス

Amazon Verified Permissions は次のいずれかの方法で使用できます。

AWS マネジメントコンソール

コンソールは Verified Permissions および AWS リソースを管理するためのブラウザーベースのインターフェイスです。コンソールから IAM にアクセスする方法の詳細については、AWS サインイン ユーザーガイドの「AWSにサインインする方法」を参照してください。

- Amazon Verified Permissions コンソール

AWS コマンドラインツール

AWS コマンドラインツールを使用して、システムのコマンドラインでコマンドを発行し、Verified Permissions と AWS タスクを実行できます。コマンドラインを使用すると、コン

ソールよりも高速で便利になります。コマンドラインツールは、 AWS のタスクを実行するスクリプトを作成する場合にも便利です。

AWS には、 AWS Command Line Interface (AWS CLI) と の 2 セットのコマンドラインツールが用意されていますAWS Tools for Windows PowerShell。のインストールと使用の詳細については AWS CLI、 AWS Command Line Interface ユーザーガイドを参照してください。Tools for Windows PowerShell のインストールおよび使用の方法については、AWS Tools for PowerShell ユーザーガイドを参照してください。

- AWS CLI コマンドリファレンスの「 verifiedpermissions」
- での Amazon Verified アクセス許可 AWS Tools for Windows PowerShell

AWS SDKs

AWS にはSDKs (ソフトウェア開発キット) が用意されています。SDK は、Verified Permissions や AWSへのプログラムによるアクセス許可を作成するのに役立ちます。例えば、SDK は要求への暗号を使用した署名、エラーの管理、要求の自動的な再試行などのタスクを処理します。

詳細と AWS SDKs「Tools for Amazon Web Services」を参照してください。

以下は、さまざまな AWS SDKs。

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

AWS CDK コンストラクト

AWS Cloud Development Kit (AWS CDK) は、コードでクラウドインフラストラクチャを定義し、それをプロビジョニングするためのオープンソースのソフトウェア開発フレームワークです CloudFormation。コンストラクトまたは再利用可能なクラウドコンポーネントを使用してテンプレートを作成できます CloudFormation 。その後、これらのテンプレートを使用してクラウドインフラストラクチャをデプロイできます。

詳細と AWS CDK のダウンロードについては、AWS 「 クラウド開発キット」を参照してください。

以下は、 コンストラクトなどの Verified Permissions AWS CDK リソースのドキュメントへのリンクです。

- Amazon Verified Permissions L2 CDK コンストラクト

Verified Permissions API

Verified Permissions API を使用すると、Verified Permissions および に AWS プログラムでアクセスできます。これにより、HTTPS リクエストを サービスに直接発行できます。 API を使用する場合は、認証情報を使用してリクエストにデジタル署名するコードを含める必要があります。

- Amazon Verified Permissions API リファレンスガイド

Verified Permissions の料金

Verified Permissions は、アプリケーションが検証済み権限に対して行う 1 か月あたりの認可リクエスト数に基づいて段階的に課金されます。また、ポリシー管理アクションには、アプリケーションが Verified Permissions に対して毎月行うcURL (クライアントURL) ポリシーAPIリクエストの量に基づく料金設定もあります。

Verified Permissions の課金および料金の詳細な一覧については、「Amazon Verified Permissions の料金表」を参照してください。

請求を表示するにはAWS Billing and Cost Management コンソールで請求およびコスト管理ダッシュボードに移動します。請求書には料金の明細が記載された使用状況レポートへのリンクが記載されています。 AWS アカウント 請求の詳細については、AWS Billing 「 ユーザーガイド」を参照してください。

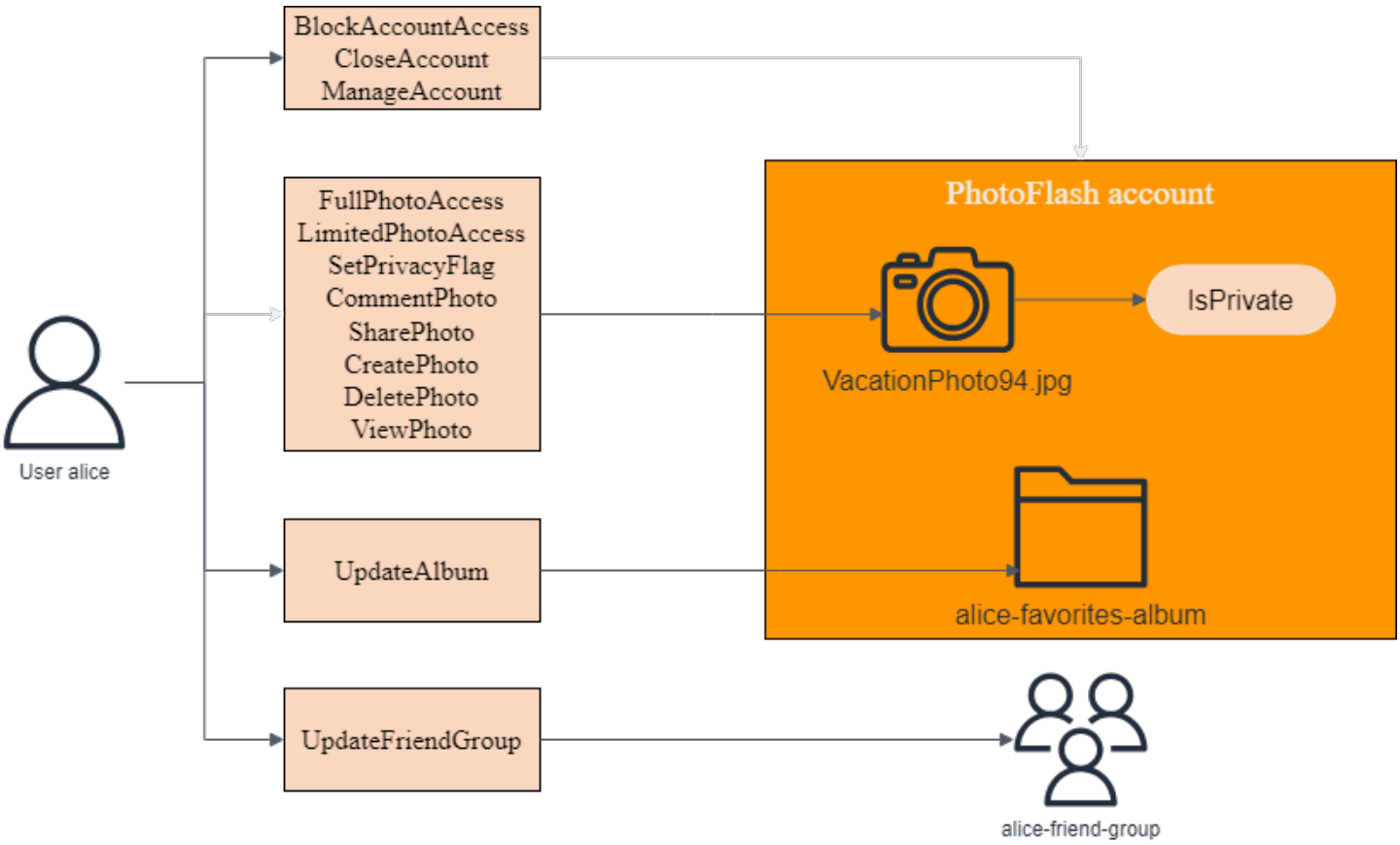
AWS 請求、アカウント、イベントについてご質問がある場合は、 にお問い合わせください サポート。

最初の Amazon Verified Permissions ポリシーストアを作成する

このチュートリアルでは、自分が写真共有アプリケーションの開発者であり、アプリケーションのユーザーが実行できるアクションを制御する方法を探しているとします。写真やフォトアルバムを追加、削除、または表示できるユーザーを制御します。また、ユーザーが自分のアカウントに対して実行できるアクションを制御することもできます。アカウントを管理できますか? 友達のアカウントについてはどうですか? これらのアクションを制御するには、ユーザーの ID に基づいてこれらのアクションを許可または禁止するポリシーを作成します。Verified Permissions は、これらのポリシーを格納するためのポリシーストアまたはコンテナを提供します。

このチュートリアルでは、Amazon Verified Permissions コンソールを使用してサンプルポリシーストアを作成する方法について説明します。コンソールにはいくつかのサンプルポリシーストアオプションが用意されており、PhotoFlash ポリシーストアを作成します。このポリシーストアでは、ユーザーなどのプリンシパルが写真やアルバムなどのリソースで共有などのアクションを実行できます。

次の図は、プリンシパル、`User::alice`、および PhotoFlash アカウント、`VactionPhoto94.jpg` ファイル、フォトアルバム、`alice-favorites-album`ユーザーグループなどのさまざまなリソースに対して実行できるアクションの関係を示していますalice-friend-group。



PhotoFlash ポリシーストアについて理解できたところで、ポリシーストアを作成して調べてみましょう。

前提条件

にサインアップする AWS アカウント

の使用を開始するには AWS、 が必要です AWS アカウント。の作成の詳細については AWS アカウント、「 AWS アカウント管理 リファレンスガイド」の「 の開始方法 AWS アカウント」を参照してください。

ステップ 1: PhotoFlash ポリシーストアを作成する

次の手順では、 AWS コンソールを使用して PhotoFlash ポリシーストアを作成します。

PhotoFlash ポリシーストアを作成するには

1. Verified Permissions コンソールで、新しいポリシーストアの作成を選択します。

2. 開始オプションで、サンプルポリシーストアから開始を選択します。
3. サンプルプロジェクトで、PhotoFlash を選択します。
4. [ポリシーストアを作成]を選択します。

「作成および設定されたポリシーストア」というメッセージが表示されたら、概要に移動を選択してポリシーストアを確認します。

ステップ 2: ポリシーを作成する

ポリシーストアを作成すると、ユーザーが自分のアカウントを完全に制御できるようにするデフォルトのポリシーが作成されました。これは便利なポリシーですが、ここでは、Verified Permissions のニュアンスを調べるために、より制限の厳しいポリシーを作成しましょう。チュートリアルの前半で説明した図を覚えている場合は、プリンシパル がありUser::alice、リソース UpdateAlbumに対してアクションを実行できますalice-favorites-album。Alice と Alice のみがこのアルバムを管理できるようにするポリシーを追加しましょう。

ポリシーを作成する方法

1. Verified Permissions コンソールで、ステップ 1 で作成したポリシーストアを選択します。
2. ナビゲーションで、ポリシーを選択します。
3. [ポリシーを作成] を選択し、次に [静的ポリシーを作成] を選択します。
4. ポリシー効果 で、許可 を選択します。
5. プリンシパルスコープで特定のプリンシパルを選択し、エンティティタイプの指定で PhotoFlash::User を選択し、エンティティ識別子の指定で を入力しますalice。
6. リソーススコープで特定のリソースを選択し、エンティティタイプの指定で PhotoFlash::Album を選択し、エンティティ識別子の指定で を入力しますalice-favorites-album。
7. アクションスコープで特定のアクションセットを選択し、このポリシーを適用するアクション (複数可) で UpdateAlbum を選択します。
8. [次へ] を選択します。
9. 詳細のポリシーの説明 - オプションで を入力しますPolicy allowing alice to update alice-favorites-album.。
10. [ポリシーの作成] を選択します。

ポリシーを作成したら、Verified Permissions コンソールでテストできます。

ステップ 3: ポリシーストアをテストする

ポリシーストアとポリシーを作成したら、Verified Permissions テストベンチを使用してシミュレートされた認可リクエストを実行して、それらをテストできます。

ポリシーストアポリシーをテストするには

1. Verified Permissions コンソールを開きます。ポリシーストアを選択します。
2. 左側のナビゲーションペインで、[テストベンチ] を選択します。
3. [ビジュアルモード] を選択します。
4. プリンシパルの場合は、次の操作を行います。
    a. プリンシパルがアクションを実行するには、PhotoFlash::User を選択し、エンティティ識別子を指定するには、 を入力します**alice**。
    b. 属性の アカウント: エンティティ で、PhotoFlash::Account エンティティが選択されていることを確認します。エンティティ識別子を指定する で、 と入力します**alice-account**。
5. 「リソース」で、プリンシパルが動作しているリソースに PhotoFlash::Album リソースタイプを選択し、「エンティティ識別子を指定」に「」と入力します**alice-favorites-album**。
6. Action では、有効なアクションのリストから PhotoFlash::Action::"UpdateAlbum" を選択します。
7. ページの上部で、認可リクエストの実行を選択して、サンプルポリシーストアの Cedar ポリシーの認可リクエストをシミュレートします。テストベンチには、ポリシーが期待どおりに動作していることを示す決定: 許可が表示されます。

以下の表は、Verified Permissions テストベンチでテストできるプリンシパル、リソース、アクションの追加値を示しています。この表には、PhotoFlash サンプルポリシーストアに含まれる静的ポリシーと、ステップ 2 で作成したポリシーに基づく認可リクエストの決定が含まれています。

プリンシパル値	プリンシパル アカウント: エンティティ値	リソース値	リソース親値	[アクション]	認可決定
PhotoFlash::User | bob	PhotoFlash::Account | alice-account	PhotoFlash::Album	該当なし	PhotoFlash::Action	拒否

プリンシパル値	プリンシパルアカウント:エンティティ値	リソース値	リソース親値	[アクション]	認可決定
		| alice-favorites-album		::"UpdateAlbum"	
PhotoFlash::User | alice	PhotoFlash::Account | alice-account	PhotoFlash::Photo | photo.jpeg	PhotoFlash::Account | bob-account	PhotoFlash::Action::"ViewPhoto"	拒否
PhotoFlash::User | alice	PhotoFlash::Account | alice-account	PhotoFlash::Photo | photo.jpeg	PhotoFlash::Account | alice-account	PhotoFlash::Action::"ViewPhoto"	許可
PhotoFlash::User | alice	PhotoFlash::Account | alice-account	PhotoFlash::Photo | bob-photo.jpeg	PhotoFlash::Album | Bob-Vacation-Album	PhotoFlash::Action::"DeletePhoto"	拒否

ステップ 4: リソースをクリーンアップする

ポリシーストアの探索が完了したら、削除します。

ポリシーストアを削除するには

1. Verified Permissions コンソールで、ステップ 1 で作成したポリシーストアを選択します。
2. ナビゲーションで、設定 を選択します。
3. 「ポリシーストアの削除」で、「このポリシーストアの削除」を選択します。
4. このポリシーストアを削除しますか? ダイアログボックスに「削除」と入力し、「削除」を選択します。

認可モデルを設計するためのベストプラクティス

ソフトウェアアプリケーション内で Amazon Verified Permissions サービスを使用する準備をしていると、最初のステップとしてポリシーステートメントをすぐに作成するのが難しい場合があります。これは、アプリケーションが何をすべきかを完全に決定する前に、SQL ステートメントや API 仕様を記述して、アプリケーションの他の部分の開発を開始するのと似ています。代わりに、ユーザーエクスペリエンスから始める必要があります。次に、その経験から逆算して実装アプローチにたどり着きます。

この作業を進めていくと、次のような質問をすることになります。

- 私のリソースは何でしょうか？ どのように整理されていますか? たとえば、ファイルはフォルダ内に存在しますか?
- リソースの組織はアクセス許可モデルに関与していますか?
- プリンシパルは各リソースに対してどのようなアクションを実行できますか?
- プリンシパルはどのようにしてこれらの権限を取得するのでしょうか?
- エンドユーザーに「管理者」、「オペレーター」、「ReadOnly」などの定義済みの権限から選択してもらいたいですか、それともアドホックなポリシーステートメントを作成すべきですか？ それとも両方一緒ですか？
- ロールはグローバルですか、それともスコープ内ですか? たとえば、「オペレータ」は単一のテナント内で制限されていますか、「オペレータ」はアプリケーション全体でオペレータを意味しますか?
- ユーザーエクスペリエンスを実現するためにはどのような種類のクエリが必要ですか? たとえば、プリンシパルがそのユーザーのホームページを表示するためにアクセスできるすべてのリソースを一覧表示する必要があるでしょうか？
- ユーザーが誤って自分のリソースからロックアウトされてしまうことはありませんか? それを回避する必要がありますか？

この作業の最終成果は承認モデルと呼ばれ、プリンシパル、リソース、アクションを定義し、それらがどのように相互に関連しているかを定義します。このモデルを作成するのに、Cedar や Verified Permissions サービスに関する独自の知識は必要ありません。代わりに、何よりもまず、他のものと同様に、ユーザーエクスペリエンス設計の演習であり、インターフェイスモックアップ、論理図、アクセス許可が製品でユーザーができることにどのように影響するかの全体的な説明などのアーティファクトにマニフェストできます。Cedar は、Cedar の実装に合わせてモデルを不自然に曲げさせる

のではなく、あるモデルで顧客に対応できる柔軟性を備えるように設計されています。そのため、希望するユーザー体験を的確に把握することが、最適なモデルを導き出す最善の方法です。

質問に答えて最適なモデルにたどり着くには、次の操作を行います。

- Cedar ポリシー言語リファレンスガイドの「Cedar 設計パターン」を参照してください。
- Cedar ポリシー言語リファレンスガイドのベストプラクティスを検討してください。
- このページに含まれるベストプラクティスを検討してください。

ベストプラクティス

- 正規の「正しい」モデルはありません
- 404 個の見つからないエラーではなく、403 個の禁止エラーを返す
- API オペレーション以外のリソースに集中する
- マルチテナンシーに関する考慮事項

正規の「正しい」モデルはありません

認可モデルを設計する場合、単一の一意に正しい回答はありません。アプリケーションが異なれば、似たような概念でも異なる認可モデルを効果的に使用できますが、これは問題ありません。たとえば、コンピューターのファイルシステムの表現について考えます。Unix のようなオペレーティングシステムでファイルを作成する場合、親フォルダからアクセス許可を自動的に継承しません。これとは対照的に、他の多くのオペレーティングシステムやほとんどのオンラインファイル共有サービスでは、ファイルは親フォルダーの権限を継承します。どちらの選択肢も、アプリケーションが最適化する状況に応じて有効です。

認可ソリューションの正しさは絶対的なものではありませんが、顧客が求めるエクスペリエンスをどのように提供するか、また期待どおりにリソースを保護するかどうかという観点から考える必要があります。認可モデルがこれを実現できれば、成功と言えるでしょう。

そのため、望ましいユーザーエクスペリエンスから設計を開始することが、効果的な認可モデルを作成する上で最も役立つ前提条件となります。

404 個の見つからないエラーではなく、403 個の禁止エラーを返す

エンティティ、特に 404 Not found エラーではなくポリシーに対応していないリソースを含むリクエストには、403 Forbidden エラーを返すことをお勧めします。 これにより、リクエストがポリシース

トア内のポリシーのポリシー条件を満たしていないだけで、エンティティが存在するかどうかを公開しないため、最高レベルのセキュリティが提供されます。

API オペレーション以外のリソースに集中する

ほとんどのアプリケーションでは、アクセス許可はサポートされているリソースを中心にモデル化されています。たとえば、ファイル共有アプリケーションでは、権限をファイルまたはフォルダに対して実行できるアクションとして表現する場合があります。これは、基礎となる実装とバックエンド API 操作を抽象化した、優れた、シンプルなモデルです。

これとは対照的に、他の種類のアプリケーション、特にウェブサービスでは、API 操作自体を中心に権限を設計することがよくあります。たとえば、Web サービスが `createThing()` という名前の API を提供する場合、認可モデルは対応する権限、または Cedar の `createThing` という名前の `action` を定義する可能性があります。これは多くの状況に当てはまり、権限を理解しやすくなります。`createThing`オペレーションを呼び出すには、`createThing`アクション権限が必要です。簡単でしょう？

Verified Permissions コンソールの開始プロセスには、API から直接リソースとアクションを構築するオプションが含まれています。これは便利なベースラインです。ポリシーストアとポリシーストアが認可する API 間の直接マッピングです。

ただし、モデルをさらに開発するとき、API はお客様が本当に保護しようとしているもの、つまり基盤となるデータとリソースのプロキシにすぎないため、この APIs に焦点を当てたアプローチは、非常に詳細な認可モデルを持つアプリケーションには適していない可能性があります。複数の API が同じリソースへのアクセスを制御している場合、管理者がそれらのリソースへのパスを判断し、それに応じてアクセスを管理することが難しくなる可能性があります。

たとえば、組織のメンバーを含むユーザーディレクトリを考えてみましょう。ユーザーはグループにまとめられますが、セキュリティ上の目標の 1 つは、権限のない第三者によるグループメンバーの発見を禁止することです。このユーザーディレクトリを管理するサービスには、次の 2 つの API オペレーションがあります。

- `listMembersOfGroup`
- `listGroupMembershipsForUser`

顧客は、これらの操作のいずれかを使用してグループメンバーシップを確認できます。そのため、権限管理者は両方の操作へのアクセスを調整することを忘れないようにする必要があります。次のよう

な他のユースケースに対応するために後から新しい API オペレーションを追加することになった場合、これはさらに複雑になります。

- isUserInGroups(ユーザーが 1 つ以上のグループに属しているかどうかをすばやくテストするための新しい API)

セキュリティの観点から見ると、この API はグループメンバーシップを発見するための 3 つ目の方法となり、管理者が巧妙に作り上げた権限を妨害することになります。

基盤となるデータとリソース、およびそれらの関連付けオペレーションに集中することをお勧めします。この方法をグループメンバーシップの例に適用すると、3 つの API オペレーションがそれぞれ参照しなければならないようなviewGroupMembership、抽象的な権限になってしまいます。

API 名	アクセス許可
listMembersOfGroup	グループにviewGroupMembership 権限が必要です。
listGroupMembershipsForUser	ユーザーにviewGroupMembership 権限が必要です。
isUserInGroups	ユーザーにviewGroupMembership 権限が必要です。

この 1 つの権限を定義することで、管理者はグループメンバーシップを検索するためのアクセスを現在および永久に正常に制御できます。トレードオフとして、各 API 操作で必要になる可能性のある複数の権限を文書化する必要があります。管理者は権限を作成する際にこのドキュメントを参照する必要があります。セキュリティ要件を満たすために必要がある場合、これは有効なトレードオフになります。

マルチテナンシーに関する考慮事項

アプリケーションを使用する企業やテナントなど、複数の顧客が使用するアプリケーションを開発し、Amazon Verified Permissions と統合することもできます。認可モデルを開発する前に、マルチテナント戦略を開発してください。顧客のポリシーは、1 つの共有ポリシーストアで管理することも、テナントごとのポリシーストアを割り当てることもできます。詳細については、「 規範ガイダンス」の「Amazon Verified Permissions マルチテナント設計上の考慮事項」を参照してください。AWS

1. 1 つの共有ポリシーストア

   すべてのテナントは 1 つのポリシーストアを共有します。アプリケーションは、すべての認可リクエストを共有ポリシーストアに送信します。

2. テナントごとのポリシーストア

   各テナントには専用のポリシーストアがあります。アプリケーションは、リクエストを行うテナントに応じて、異なるポリシーストアに承認の決定をクエリします。

どちらの戦略も AWS 請求書に大きな影響を与えません。では、どのようにアプローチを設計すべきですか? 以下は、Verified Permissions マルチテナンシー認可戦略に寄与する可能性のある一般的な条件です。

テナントポリシーの分離

テナントデータを保護するためには、各テナントのポリシーを他のテナントから分離することが重要です。各テナントに独自のポリシーストアがある場合、それぞれに独自のポリシーセットがあります。

認可フロー

承認リクエストを行うテナントは、リクエスト内のポリシーストア ID とテナントごとのポリシーストアで識別できます。共有ポリシーストアでは、すべてのリクエストが同じポリシーストア ID を使用します。

テンプレートとスキーマの管理

アプリケーションに複数のポリシーストアがある場合、ポリシーテンプレートとポリシーストアスキーマは、各ポリシーストアに設計とメンテナンスのオーバーヘッドのレベルを追加します。

グローバルポリシー管理

一部のグローバルポリシーをすべてのテナントに適用できます。グローバルポリシーの管理のオーバーヘッドレベルは、共有ポリシーストアモデルとテナントごとのポリシーストアモデルによって異なります。

テナントのオフボーディング

一部のテナントは、そのケースに固有のスキーマとポリシーに要素を提供します。テナントが組織でアクティブでなくなり、データを削除する場合、労力のレベルは他のテナントからの分離のレベルによって異なります。

サービスリソースクォータ

Verified Permissions には、マルチテナンシーの決定に影響を与える可能性のあるリソースとリクエストレートのクォータがあります。クォータの詳細については、「リソースのクォータ」を参照してください。

共有ポリシーストアとテナントごとのポリシーストアの比較

各考慮事項には、共有ポリシーストアモデルとテナントごとのポリシーストアモデルにおける独自の時間とリソースコミットメントのレベルが必要です。

考慮事項	共有ポリシーストアの労力レベル	テナントごとのポリシーストアの労力レベル
テナントポリシーの分離	Medium。Must include tenant identifiers in policies and authorization requests.	低。 Isolation is default behavior. Tenant-specific policies are inaccessible to other tenants.
認可フロー	低。 All queries target one policy store.	Medium。 Must maintain mappings between each tenant and their policy store ID.
テンプレートとスキーマの管理	低。 Must make one schema work for all tenants.	高 Schemas and templates might be less complex individually, but changes require more coordination and complexity.
グローバルポリシー管理	低。 All policies are global and can be centrally updated.	高 You must add global policies to each policy store in onboarding. Replicate global policy updates between many policy stores.
テナントのオフボーディング	高 Must identify and delete only tenant-specific policies.	低。 Delete the policy store.

サービスリソースクォータ	高 Tenants share resource quotas that affect policy stores like schema size, policy size per resource, and identity sources per policy store.	低。 Each tenant has dedicated resource quotas.

選択方法

マルチテナントアプリケーションはそれぞれ異なります。アーキテクチャを決定する前に、2 つのアプローチとその考慮事項を慎重に比較します。

アプリケーションがテナント固有のポリシーを必要とせず、単一の ID ソースを使用する場合、すべてのテナントに対して 1 つの共有ポリシーストアが最も効果的なソリューションである可能性があります。これにより、認可フローとグローバルポリシー管理が簡素化されます。1 つの共有ポリシーストアを使用してテナントをオフボーディングする場合、アプリケーションがテナント固有のポリシーを削除する必要がないため、より少ない労力で済みます。

ただし、アプリケーションが多くのテナント固有のポリシーを必要とする場合、または複数の ID ソースを使用する場合、テナントごとのポリシーストアが最も効果的である可能性があります。テナントごとのアクセス許可を各ポリシーストアに付与する IAM ポリシーを使用して、テナントポリシーへのアクセスを制御できます。テナントのオフボーディングにはポリシーストアの削除が含まれます。shared-policy-store環境では、テナント固有のポリシーを見つけて削除する必要があります。

Amazon Verified Permissions ポリシーストア

ポリシーストアはポリシーとポリシーテンプレートのコンテナです。各ポリシーストアで、ポリシーストアに追加されたポリシーを検証するために使用されるスキーマを作成できます。さらに、ポリシー検証を有効にすることもできます。ポリシー検証を有効にしてポリシーストアにポリシーを追加すると、ポリシーで定義されているエンティティタイプ、共通タイプ、およびアクションがスキーマに対して検証され、無効なポリシーは拒否されます。

削除保護は、ポリシーストアの偶発的な削除を防ぎます。削除保護は、 を通じて作成されたすべての新しいポリシーストアで有効になります AWS マネジメントコンソール。対照的に、API または SDK コールで作成されたすべてのポリシーストアでは無効になっています。

アプリケーションごとに 1 つのポリシーストアを作成するか、マルチテナントアプリケーションの場合はテナントごとに 1 つのポリシーストアを作成することをお勧めします。認可リクエストを行うときは、ポリシーストアを指定する必要があります。ポリシーストアエイリアスを作成して、フレンドリ名でポリシーストアを参照することもできます。詳細については、「Amazon Verified Permissions ポリシーストアのエイリアス」を参照してください。

あいまいさを避けるため、ポリシーストアの Cedar エンティティには名前空間を使用することをお勧めします。名前空間はタイプの文字列プレフィックスで、区切り文字としてコロン (`::`) のペアで区切られています。例: `MyApplicationNamespace::exampleType`。Verified Permissions は、ポリシーストアごとに 1 つの名前空間をサポートします。これらの名前空間は、複数の類似アプリケーションを操作するときに物事をまっすぐに保つのに役立ちます。例えば、マルチテナントアプリケーションでは、名前空間を使用して、スキーマで定義されたタイプにテナントの名前を追加すると、他のテナントが使用する同様のものとは異なります。認可リクエストのログを確認すると、認可リクエストを処理したテナントを簡単に識別できます。詳細については、Cedar policy language Reference Guideの「Namespaces」を参照してください。

トピック

- Verified Permissions ポリシーストアの作成
- API リンクポリシーストア
- ポリシーストアの削除

Verified Permissions ポリシーストアの作成

以下の方法で、ポリシーストアを作成できます。

- ガイド付きセットアップに従う – 最初のポリシーを作成する前に、有効なアクションとプリンシパルタイプを持つリソースタイプを定義します。
- API Gateway と ID ソースのセットアップ – ID プロバイダー (IdP) でサインインするユーザーと、Amazon API Gateway API からのアクションとリソースエンティティを使用して、プリンシパルエンティティを定義します。このオプションは、ユーザーのグループメンバーシップまたは他の属性を使用してアプリケーションが API リクエストを承認する場合にお勧めします。
- サンプルポリシーストアから開始 – 事前定義されたサンプルプロジェクトポリシーストアを選択します。Verified Permissions について学習していて、サンプルポリシーを表示およびテストしたい場合は、このオプションをお勧めします。
- 空のポリシーストアを作成する – スキーマとすべてのアクセスポリシーを自分で定義します。ポリシーストアの設定にすでに慣れている場合は、このオプションをお勧めします。

Guided setup

ガイド付き設定方法を使用してポリシーストアを作成するには

ガイド付き設定ウィザードの指示に従って、ポリシーストアの最初のイテレーションを作成します。最初のリソースタイプのスキーマを作成し、そのリソースタイプに適用できるアクションと、権限を付与するプリンシパルタイプを記述します。次に、最初のポリシーを作成します。このウィザードを完了すると、ポリシーストアに追加したり、スキーマを拡張して他のリソースやプリンシパルタイプを記述したり、追加のポリシーやテンプレートを作成したりできます。

1. Verified Permissions コンソールで、新しいポリシーストアの作成を選択します。
2. 開始オプションセクションで、ガイド付きセットアップを選択します。
3. ポリシーストアの説明を入力します。このテキストは、Weather updates web application など、現在のポリシーストアの 関数へのわかりやすい参照として、組織に適したものにすることができます。
4. 「詳細」セクションに、スキーマの名前空間を入力します。名前空間の詳細については、「」を参照してください名前空間の定義。
5. [次へ] を選択します。
6. 「リソースタイプ」ウィンドウに、リソースタイプの名前を入力します。たとえば、 は Weather updates ウェブアプリケーションのリソース`currentTemperature`である可能性があります。
7. (オプション) [属性を追加] を選択してリソース属性を追加します。リソースの各属性の属性名を入力し、属性タイプを選択します。各属性が必須かどうかを選択します。たとえば、

temperatureFormatはcurrentTemperatureリソースの属性で、華氏または摂氏のいずれかになります。リソースタイプに追加された属性を削除するには、属性の横にある [削除] を選択します。

8. 「アクション」フィールドに、指定したリソースタイプに対して認証するアクションを入力します。リソースタイプにアクションを追加するには、「アクションを追加」を選択します。たとえば、 は「Weather updates web application」のアクションviewTemperatureである場合があります。リソースタイプに追加されたアクションを削除するには、アクションの横にある [削除] を選択します。
9. 「プリンシパルタイプの名前」フィールドに、リソースタイプに指定されたアクションを使用するプリンシパルの名前を入力します。デフォルトでは、ユーザーはこのフィールドに追加されますが、置き換えることができます。
10. [次へ] を選択します。
11. 「プリンシパルタイプ」ウィンドウで、プリンシパルタイプのID ソースを選択します。

    - プリンシパルの ID と属性を Verified Permissions アプリケーションから直接提供する場合は、「カスタム」を選択します。属性を追加するには、[属性を追加] を選択します。Verified Permissions では、スキーマに対してポリシーを検証するときに、指定された属性値を使用します。プリンシパルタイプに追加された属性を削除するには、属性の横にある削除を選択します。
    - プリンシパルの ID と属性が によって生成された ID またはアクセストークンから提供される場合は、Cognito ユーザープールを選択します Amazon Cognito。[ユーザープールを接続] を選択します。AWS リージョン を選択し、接続先のユーザープールのユーザープール ID を入力します。 Amazon Cognito [接続]を選択します。詳細については、「Amazon Cognito デベロッパーガイド」の「Amazon Verified Permissions による認可」を参照してください。 Amazon Cognito
    - プリンシパルの ID と属性が ID トークンやアクセストークンから抽出され、外部 OIDC プロバイダーによって生成された場合は、外部 OIDC プロバイダーを選択し、プロバイダーとトークンの詳細を追加します。
12. [次へ] を選択します。
13. ポリシーの詳細セクションに、最初の Cedar ポリシーに関するポリシーの説明をオプションで入力します。
14. 「プリンシパルの範囲」フィールドで、ポリシーから権限を付与されるプリンシパルを選択します。

- 特定のプリンシパルにポリシーを適用するには、「特定のプリンシパル」を選択します。「アクションの実行を許可するプリンシパル」フィールドでプリンシパルを選択し、プリンシパルのエンティティ ID を入力します。たとえば、 は、Weather updates ウェブアプリケーションのエンティティ識別子`user-id`である場合があります。

  > **Note**
  >
  > を使用している場合 Amazon Cognito、エンティティ識別子は の形式である必要があります`<userpool-id>|<sub>`。

- ポリシーストア内のすべてのプリンシパルにポリシーを適用するには、「すべてのプリンシパル」を選択します。

15. [リソースの範囲] フィールドで、指定したプリンシパルにアクションを許可するリソースを選択します。

    - 特定のリソースにポリシーを適用するには、「特定のリソース」を選択します。「このポリシーが適用される必要があるリソース」フィールドでリソースを選択し、リソースのエンティティ ID を入力します。たとえば、 は Weather updates ウェブアプリケーションのエンティティ識別子`temperature-id`である場合があります。
    - ポリシーストア内のすべてのリソースにポリシーを適用するには、[すべてのリソース] を選択します。

16. [アクションの範囲] フィールドで、指定したプリンシパルに実行を許可するアクションを選択します。

    - 特定のアクションにポリシーを適用するには、「特定のアクションセット」を選択します。「このポリシーが適用される必要があるアクション」フィールドで、アクションの横にあるチェックボックスを選択します。
    - ポリシーストア内のすべてのアクションにポリシーを適用するには、[すべてのアクション] を選択します。

17. ポリシープレビューセクションでポリシーを確認してください。[ポリシーストアを作成]を選択します。

Set up with API Gateway and an identity source

でのセットアップ API Gateway と ID ソース設定方法を使用してポリシーストアを作成するには

API Gateway オプションは、ユーザーのグループまたはロールから認可を決定するように設計された Verified Permissions ポリシーを使用して APIs を保護します。このオプションは、ID ソースグループによる認可をテストするためのポリシーストアと、Lambda オーソライザーによる API を構築します。

IdP のユーザーとそのグループは、プリンシパル (ID トークン) またはコンテキスト (アクセストークン) になります。API のメソッドとパスは API Gateway 、ポリシーが承認するアクションになります。アプリケーションがリソースになります。このワークフローの結果として、Verified Permissions はポリシーストア、Lambda 関数、API Lambda オーソライザーを作成します。このワークフローが完了したら、Lambda オーソライザーを API に割り当てる必要があります。

1. Verified Permissions コンソールで、新しいポリシーストアの作成を選択します。
2. Starting options セクションで、Set up with API Gateway and an identity source を選択し、Next を選択します。
3. リソースとアクションのインポートステップの API で、ポリシーストアのリソースとアクションのモデルとして機能する API を選択します。

    a. API で設定されたステージからデプロイステージを選択し、API のインポートを選択します。API ステージの詳細については、Amazon API Gatewayデベロッパーガイド」の「REST API のステージのセットアップ」を参照してください。

    b. インポートされたリソースとアクションのマップをプレビューします。

    c. リソースまたはアクションを更新するには、 API Gateway コンソールで API パスまたはメソッドを変更し、API のインポートを選択して更新を確認します。

    d. 選択内容に満足したら、次へを選択します。
4. ID ソースで、ID プロバイダータイプを選択します。 Amazon Cognito ユーザープールまたは OpenID Connect (OIDC) IdP タイプを選択できます。
5. [Amazon Cognito] を選択した場合:

    a. ポリシーストアと同じ AWS リージョン および AWS アカウント のユーザープールを選択します。

    b. 承認のために送信する API に渡すトークンタイプを選択します。どちらのトークンタイプにも、この API リンク認可モデルの基盤であるユーザーグループが含まれています。

c. アプリクライアントの検証では、ポリシーストアの範囲をマルチテナントユーザープール内の Amazon Cognito アプリクライアントのサブセットに制限できます。ユーザープール内の 1 つ以上の指定されたアプリケーションクライアントによる認証をユーザーに要求するには、予想されるアプリケーションクライアント IDs を持つトークンのみを受け入れるを選択します。ユーザープールで認証するユーザーを受け入れるには、アプリクライアント IDs を検証しないを選択します。

d. [次へ] を選択します。

6. 外部 OIDC プロバイダーを選択した場合:

a. 発行者 URL に、OIDC 発行者の URL を入力します。これは、認可サーバー、署名キー、および などのプロバイダーに関するその他の情報を提供するサービスエンドポイントですhttps://auth.example.com。発行者 URL は、 で OIDC 検出ドキュメントをホストする必要があります/.well-known/openid-configuration。

b. トークンタイプで、アプリケーションが承認のために送信する OIDC JWT のタイプを選択します。詳細については、「スキーマへの Amazon Cognito トークンのマッピング」および「スキーマへの OIDC トークンのマッピング」を参照してください。

c. (オプション) トークンクレーム - オプションで、トークンクレームの追加を選択し、トークンの名前を入力し、値タイプを選択します。

d. ユーザートークンとグループトークンのクレームで、次の操作を行います。

i. ID ソースのトークンにユーザークレーム名を入力します。これは、通常sub、評価されるエンティティの一意の識別子を保持する ID またはアクセストークンからのクレームです。接続された OIDC IdP の ID は、ポリシーストアのユーザータイプにマッピングされます。

ii. ID ソースのトークンにグループクレーム名を入力します。これは、通常groups、ユーザーのグループのリストを含む ID またはアクセストークンからのクレームです。ポリシーストアは、グループメンバーシップに基づいてリクエストを承認します。

e. 対象者の検証で、ポリシーストアが認可リクエストで受け入れる値を選択してAdd value追加します。

f. [次へ] を選択します。

7. を選択した場合Amazon Cognito、Verified Permissions はユーザープールにグループをクエリします。OIDC プロバイダーの場合は、グループ名を手動で入力します。グループへのアクションの割り当てステップでは、グループメンバーがアクションを実行できるようにするポリシーストアのポリシーを作成します。

a. ポリシーに含めるグループを選択または追加します。

b. 選択した各グループにアクションを割り当てます。

c. [次へ] を選択します。

8. アプリ統合のデプロイで、後で Lambda オーソライザーを手動でアタッチするか、Verified Permissions で今すぐアタッチするかを選択し、Verified Permissions がポリシーストアと Lambda オーソライザーを作成するために実行する手順を確認します。
9. 新しいリソースを作成する準備ができたら、ポリシーストアの作成を選択します。
10. ポリシーストアのステータスステップをブラウザで開いたままにして、Verified Permissions によるリソース作成の進行状況をモニタリングします。
11. 通常約 1 時間後、または Lambda オーソライザーのデプロイステップに Success と表示されたら、オーソライザーを手動でアタッチすることを選択した場合は、オーソライザーを設定します。

    Verified Permissions は、API に Lambda 関数と Lambda オーソライザーを作成します。API を開く を選択して、API に移動します。

    Lambda オーソライザーを割り当てる方法については、Amazon API Gateway デベロッパーガイド」のAPI Gateway 「Lambda オーソライザーを使用する」を参照してください。

    a. API のオーソライザーに移動し、Verified Permissions が作成したオーソライザーの名前を書き留めます。

    b. リソースに移動し、API で最上位のメソッドを選択します。

    c. メソッドリクエスト設定で編集を選択します。

    d. オーソライザーを前にメモしたオーソライザー名に設定します。

    e. HTTP リクエストヘッダーを展開し、名前 または を入力しAUTHORIZATION、必須 を選択します。

    f. API ステージをデプロイします。

    g. 変更を保存します。
12. ID ソースの選択ステップで選択したトークンタイプのユーザープールトークンを使用してオーソライザーをテストします。ユーザープールのサインインとトークンの取得の詳細については、Amazon Cognito デベロッパーガイド」の「ユーザープールの認証フロー」を参照してください。
13. API へのリクエストのAUTHORIZATIONヘッダーでユーザープールトークンを使用して認証を再度テストします。

14. 新しいポリシーストアを確認します。ポリシーを追加および絞り込みます。

Sample policy store

サンプルポリシーストア設定方法を使用してポリシーストアを作成するには

1. 開始オプションセクションで、サンプルポリシーストアを選択します。
2. 「サンプルプロジェクト」セクションで、使用するサンプルのVerified Permissions アプリケーションのタイプを選択します。

   - PhotoFlash は、ユーザーが個々の写真やアルバムを友人と共有できるようにする、顧客向けのサンプル Web アプリケーションです。ユーザーは、自分の写真の閲覧、コメント、再共有を誰に許可するかについて、きめ細かい権限を設定できます。アカウントオーナーは、友達のグループを作成したり、写真をアルバムにまとめたりすることもできます。
   - DigitalPetStoreは、誰でも登録して顧客になることができるサンプルアプリケーションです。顧客は販売するペットの追加、ペットの検索、注文を行うことができます。ペットを追加したお客様は、ペットの飼い主として記録されます。ペットの飼い主は、ペットの詳細を更新したり、ペットの画像をアップロードしたり、ペットリストを削除したりできます。注文した顧客は注文所有者として記録されます。注文所有者は注文の詳細を確認したり、注文をキャンセルしたりできます。ペットショップのマネージャーには管理者権限があります。

     > **Note**
     >
     > DigitalPetStore サンプルポリシーストアにはポリシーテンプレートは含まれていません。PhotoFlash と TinyTodo のサンプルポリシーストアにはポリシーテンプレートが含まれています。

   - TinyToDo は、ユーザがタスクやタスクリストを作成できるようにするサンプルアプリケーションです。リスト所有者はリストを管理および共有したり、リストを閲覧または編集できるユーザーを指定したりできます。
3. サンプルポリシーストアのスキーマの名前空間は、選択したサンプルプロジェクトに基づいて自動的に生成されます。
4. [ポリシーストアを作成]を選択します。

   ポリシーストアは、選択したサンプルポリシーストア用のポリシーとスキーマを使用して作成されます。サンプルポリシーストア用に作成できる、テンプレートにリンクされたポリ

シーの詳細については、Amazon Verified Permissions テンプレートにリンクされたポリシーの例を参照してください。

Empty policy store

「空のポリシーストア」設定方法を使用してポリシーストアを作成するには

1. 開始オプションセクションで、空のポリシーストアを選択します。
2. [ポリシーストアを作成]を選択します。

空のポリシーストアはスキーマなしで作成されます。つまり、ポリシーは検証されません。ポリシー ストアのスキーマの更新の詳細については、「Amazon Verified Permissions ポリシーストアスキーマ」を参照してください。

ポリシーストアのポリシーの作成に関する詳細については、「Amazon Verified Permissions 静的ポリシーの作成」と「Amazon Verified Permissions テンプレートリンクポリシーの作成」を参照してください。

AWS CLI

AWS CLIを使用して空のポリシーストアを作成するには。

ポリシーストアは、`create-policy-store`オペレーションを使用して作成できます。

> **Note**
>
> を使用して作成したポリシーストア AWS CLI は空です。
>
> - スキーマを追加するには、Amazon Verified Permissions ポリシーストアスキーマを参照してください。
> - ポリシーを追加するには、Amazon Verified Permissions 静的ポリシーの作成を参照してください。
> - ポリシーテンプレートを追加するには、Amazon Verified Permissions ポリシーテンプレートの作成を参照してください。

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

```
    "arn": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111",
    "createdDate": "2023-05-16T17:41:29.103459+00:00",
    "lastUpdatedDate": "2023-05-16T17:41:29.103459+00:00",
    "policyStoreId": "PSEXAMPLEabcdefg111111"
}
```

AWS SDKs

`CreatePolicyStore`API を使用してポリシーストアを作成できます。詳細については、Amazon Verified Permissions API リファレンスガイドの「CreatePolicyStore」を参照してください。

AWS SDK を使用した Rust での Amazon Verified Permissions の実装

このトピックでは、 SDK を使用して Rust で Amazon Verified Permissions を実装する AWS 実例を示します。この例では、ユーザーが写真を表示できるかどうかをテストできる認可モデルを開発する方法を示します。サンプルコードは、 の aws-sdk-verifiedpermissions クレートを使用します。これはAWS SDK for Rust、 を操作するための堅牢なツールセットを提供します AWS のサービス。

前提条件

開始する前に、システムに AWS CLI が設定されており、Rust に精通していることを確認してください。

- のインストール手順については AWS CLI、AWS 「CLI インストールガイド」を参照してください。
- の設定手順については AWS CLI、「」の「 の設定 AWS CLI」および「 の設定と認証情報ファイルの設定 AWS CLI」を参照してください。
- Rust の詳細については、rust-lang.org および AWS SDK for Rust デベロッパーガイドを参照してください。

環境の準備ができたら、Rust で Verified Permissions を実装する方法について説明します。

サンプルコードをテストする

サンプルコードは以下を実行します。

- と通信するように SDK クライアントを設定します AWS

- ポリシーストアを作成します。
- スキーマを追加してポリシーストアの構造を定義します。
- 承認リクエストをチェックするポリシーを追加します
- テスト認可リクエストを送信して、すべてが正しく設定されていることを確認します

サンプルコードをテストするには

1. Rust プロジェクトを作成します。
2. の既存のコードを次のコードmain.rsに置き換えます。

```
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. ターミナルに `cargo run` を入力してコードを実行します。

コードが正しく実行されると、ターミナルに決定ポリシーのポリシー ID がDecision: Allow続きます。これは、ポリシーストアが正常に作成され、 AWS SDK for Rust を使用してテストされたことを意味します。

リソースをクリーンアップする

ポリシーストアの探索が完了したら、削除します。

ポリシーストアを削除するには

`delete-policy-store` オペレーションを使用してポリシーストアを削除し、 を削除するポリシーストア ID *PSEXAMPLEabcdefg111111*に置き換えます。

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

このコマンドが成功した場合、出力は生成されません。

API リンクポリシーストア

一般的なユースケースは、Amazon Verified Permissions を使用してAmazon API Gateway でホストされている APIs へのユーザーアクセスを許可することです。 AWS コンソールのウィザードを使用して、Amazon Cognito または任意の OIDC ID プロバイダー (IdP) で管理されているユーザーのロールベースのアクセスポリシーを作成し、Verified Permissions を呼び出してこれらのポリシーを評価するオー AWS Lambda ソライザーをデプロイできます。

ウィザードを完了するには、新しいポリシーストアを作成するときに API Gateway と ID プロバイダーでセットアップを選択し、ステップに従います。

API リンクポリシーストアが作成され、認可リクエスト用に認可モデルとリソースがプロビジョニングされます。ポリシーストアには、Verified Permissions API Gateway に接続する ID ソースと Lambda オーソライザーがあります。ポリシーストアが作成されたら、ユーザーのグループメンバーシップに基づいて API リクエストを承認できます。例えば、Verified Permissions は、`Directors`グループのメンバーであるユーザーにのみアクセス権を付与できます。

アプリケーションが大きくなるにつれて、Cedar ポリシー言語を使用して、ユーザー属性と OAuth 2.0 スコープによるきめ細かな認可を実装できます。例えば、Verified Permissions は、ドメイン に `email` 属性を持つユーザーにのみアクセス権を付与できますmycompany.co.uk。

API の認可モデルを設定した後、残りの責任は、ユーザーを認証し、アプリケーションで API リクエストを生成し、ポリシーストアを維持することです。

デモを確認するには、Amazon Web Services YouTube チャンネルの「Amazon Verified Permissions - Quick Start Overview and Demo」を参照してください。

トピック

- Verified Permissions が API リクエストを承認する方法
- API リンクポリシーストアに関する考慮事項
- 属性ベースのアクセスコントロール (ABAC) の追加
- を使用した本番稼働への移行 AWS CloudFormation
- API リンクポリシーストアのトラブルシューティング

> **Important**
>
> Verified Permissions コンソールの Set up with API Gateway と ID ソースオプションを使用して作成したポリシーストアは、本番環境への即時デプロイを意図していません。最初のポリシーストアで、認可モデルを確定し、ポリシーストアリソースを CloudFormation にエクスポートします。を使用して、プログラムで Verified Permissions を本番環境にデプロイしますAWS Cloud Development Kit (AWS CDK)。詳細については、「を使用した本番稼働への移行 AWS CloudFormation」を参照してください。

API と ID ソースにリンクされたポリシーストアでは、アプリケーションは API にリクエストを行うときに、認可ヘッダーにユーザープールトークンを表示します。ポリシーストアの ID ソースは、Verified Permissions のトークン検証を提供します。トークンは、IsAuthorizedWithToken API を使用して認可リクエスト`principal`で を形成します。Verified Permissions は、アイデンティティ (ID) のグループクレームや、ユーザープールなどのアクセストークンに表示されるように、`cognito:groups`ユーザーのグループメンバーシップに関するポリシーを構築します。API は Lambda オーソライザーでアプリケーションからトークンを処理し、認可決定のために Verified Permissions に送信します。API が Lambda オーソライザーから認可決定を受け取ると、リクエストをデータソースに渡すか、リクエストを拒否します。

Verified Permissions を使用した ID ソースと API Gateway 認可のコンポーネント

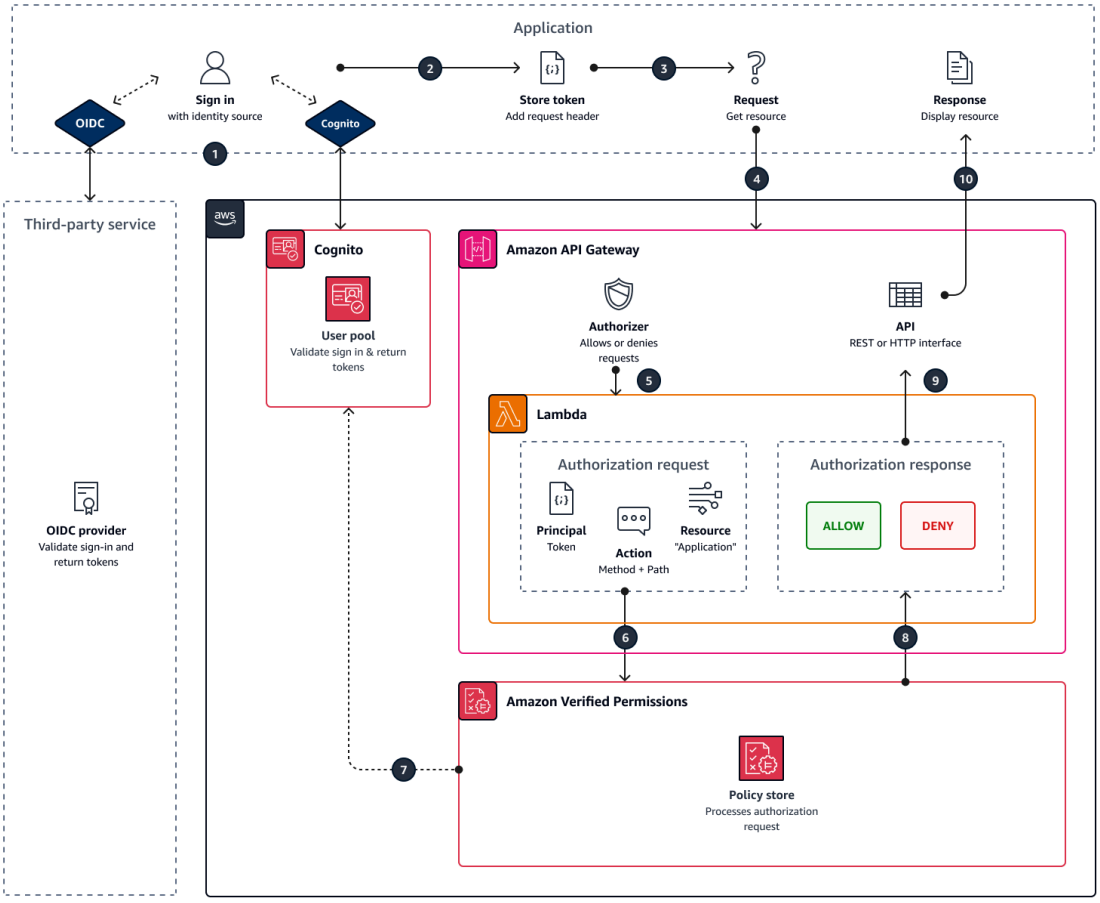
- Amazon Cognito ユーザーを認証してグループ化するユーザープールまたは OIDC IdP。ユーザーのトークンは、ポリシーストアで Verified Permissions が評価するグループメンバーシップとプリンシパルまたはコンテキストを入力します。
- API Gateway REST API。Verified Permissions は、 などの API パスと API メソッドからのアクションを定義しますMyAPI::Action::get /photo。
- API の Lambda 関数と Lambda オーソライザー。Lambda 関数は、ユーザープールからベアラートークンを取得し、Verified Permissions に認可をリクエストして、決定を返します API

Gateway。 API Gateway と ID ソースワークフローでセットアップすると、この Lambda オーソライザーが自動的に作成されます。

- Verified Permissions ポリシーストア。ポリシーストア ID ソースは、 Amazon Cognito ユーザープールまたは OIDC プロバイダーグループです。ポリシーストアスキーマは API の設定を反映し、ポリシーはユーザーグループを許可された API アクションにリンクします。
- IdP でユーザーを認証し、API リクエストにトークンを追加するアプリケーション。

Verified Permissions が API リクエストを承認する方法

新しいポリシーストアを作成し、Set up with API Gateway and an identity source オプションを選択すると、Verified Permissions はポリシーストアスキーマとポリシーを作成します。スキーマとポリシーには、API アクションと、アクションを実行することを許可するユーザーグループが反映されます。Verified Permissions は、Lambda 関数とオーソライザーも作成します。



1. ユーザーは、 Amazon Cognito または別の OIDC IdP を使用してアプリケーションにサインインします。IdP は、ID トークンとアクセストークンをユーザー情報で発行します。

2. アプリケーションは JWTsを保存します。詳細については、「 デベロッパーガイド」の「ユーザープールでのトークンの使用」を参照してください。 Amazon Cognito
3. ユーザーは、アプリケーションが外部 API から取得する必要があるデータをリクエストします。
4. アプリケーションは、 の REST API からデータをリクエストします API Gateway。ID またはアクセストークンをリクエストヘッダーとして追加します。
5. API に認可決定のキャッシュがある場合、前のレスポンスが返されます。キャッシュが無効になっている場合、または API に現在のキャッシュがない場合、 API Gateway はリクエストパラメータをトークンベースの Lambda オーソライザーに渡します。
6. Lambda 関数は、IsAuthorizedWithToken API を使用して Verified Permissions ポリシーストアに認可リクエストを送信します。Lambda 関数は、認可決定の要素を渡します。
   a. プリンシパルとしてのユーザーのトークン。
   b. アクションとして、 API メソッドを API パス、たとえば `GetPhoto`と組み合わせて使用します。
   c. リソース`Application`という用語。
7. Verified Permissions はトークンを検証します。 Amazon Cognito トークンの検証方法の詳細については、「 Amazon Cognito デベロッパーガイド」の「Amazon Verified Permissions による認可」を参照してください。
8. Verified Permissions は、ポリシーストア内のポリシーに対して認可リクエストを評価し、認可決定を返します。
9. Lambda オーソライザーは `Allow`または `Deny`レスポンスを返します API Gateway。
10. API は、アプリケーションにデータまたは`ACCESS_DENIED`レスポンスを返します。アプリケーションは API リクエストの結果を処理して表示します。

API リンクポリシーストアに関する考慮事項

Verified Permissions コンソールで API にリンクされたポリシーストアを構築すると、最終的な本番デプロイのテストが作成されます。本番環境に移行する前に、API とユーザープールの固定設定を確立します。次の要素を考慮してください。

API Gateway がレスポンスをキャッシュする

API リンクポリシーストアでは、Verified Permissions は認可キャッシュ TTL が 120 秒の Lambda オーソライザーを作成します。この値は調整することも、オーソライザーでキャッシュをオフにすることもできます。キャッシュが有効になっているオーソライザーでは、TTL の有効期限が切れるまで、オーソライザーは毎回同じレスポンスを返します。これにより、ユーザー

プールトークンの有効期間を、リクエストされたステージのキャッシュ TTL と等しい期間だけ延長できます。

Amazon Cognito グループは再利用できます

Amazon Verified Permissions は、ユーザーの ID またはアクセストークンの `cognito:groups`クレームからユーザープールユーザーのグループメンバーシップを決定します。このクレームの値は、ユーザーが属するユーザープールグループのフレンドリ名の配列です。ユーザープールグループを一意の識別子に関連付けることはできません。

ポリシーストアに同じグループと同じ名前で削除および再作成するユーザープールグループ。ユーザープールからグループを削除するときは、ポリシーストアからグループへのすべての参照を削除します。

API から派生した名前空間とスキーマはpoint-in-timeです

Verified Permissions は、特定の時点で API をキャプチャします。ポリシーストアを作成するときにのみ API をクエリします。API のスキーマまたは名前が変更された場合は、ポリシーストアと Lambda オーソライザーを更新するか、新しい API リンクポリシーストアを作成する必要があります。Verified Permissions は、API の名前からポリシーストアの名前空間を取得します。

Lambda 関数に VPC 設定がない

Verified Permissions が API オーソライザー用に作成する Lambda 関数は、デフォルトの VPC で起動されます。デフォルトでは です。プライベート VPCs に制限されたネットワークアクセスを持つ APIs は、Verified Permissions でアクセスリクエストを許可する Lambda 関数と通信できません。

Verified Permissions が CloudFormation にオーソライザーリソースをデプロイする

API リンクポリシーストアを作成するには、権限の高いプリンシパルを Verified AWS Permissions コンソールにサインインする必要があります。このユーザーは、複数の にまたがってリソースを作成する CloudFormation スタックをデプロイします AWS のサービス。このプリンシパルには、Verified Permissions、 IAM、Lambda、および でリソースを追加および変更するアクセス許可が必要です API Gateway。ベストプラクティスとして、これらの認証情報を組織内の他の管理者と共有しないでください。

Verified Permissions が作成するリソースの概要を使用した本番稼働への移行 AWS CloudFormationについては、「」を参照してください。

属性ベースのアクセスコントロール (ABAC) の追加

IdP を使用した一般的な認証セッションは、ID トークンとアクセストークンを返します。これらのトークンタイプのいずれかを、API へのアプリケーションリクエストでベアラートークンとして渡すことができます。ポリシーストアの作成時に選択した内容に応じて、Verified Permissions は 2 種類のトークンのいずれかを想定しています。どちらのタイプも、ユーザーのグループメンバーシップに関する情報を保持します。のトークンタイプの詳細については Amazon Cognito、「デベロッパーガイド」の「ユーザープールでのトークンの使用」を参照してください。 Amazon Cognito

ポリシーストアを作成したら、ポリシーを追加および拡張できます。たとえば、新しいグループをユーザープールに追加するときに、ポリシーに追加できます。ポリシーストアは、ユーザープールがトークンでグループを表示する方法をすでに認識しているため、新しいポリシーを持つ新しいグループに対して一連のアクションを許可できます。

また、ポリシー評価のグループベースのモデルを、ユーザープロパティに基づいてより正確なモデルに拡張することもできます。ユーザープールトークンには、認可の決定に役立つ追加のユーザー情報が含まれています。

ID トークン

ID トークンはユーザーの属性を表し、高レベルのきめ細かなアクセスコントロールを備えています。E メールアドレス、電話番号、部門やマネージャーなどのカスタム属性を評価するには、ID トークンを評価します。

アクセストークン

アクセストークンは、OAuth 2.0 スコープを使用したユーザーのアクセス許可を表します。認可レイヤーを追加したり、追加のリソースのリクエストをセットアップしたりするには、アクセストークンを評価します。たとえば、ユーザーが適切なグループに属しており、 API へのアクセスを一般的に許可`PetStore.read`する のようなスコープを保持していることを検証できます。ユーザープールは、実行時にリソースサーバーとトークンのカスタマイズを使用して、カスタムスコープをトークンに追加できます。 https://docs.aws.amazon.com/cognito/latest/developerguide/user-pool-lambda-pre-token-generation.html#user-pool-lambda-pre-token-generation-accesstoken

ID Amazon Cognito トークンとアクセストークンでクレームを処理するポリシーの例については、「スキーマへのトークンのマッピング」と「スキーマへの OIDC トークンのマッピング」を参照してください。

を使用した本番稼働への移行 AWS CloudFormation

API リンクポリシーストアは、 API Gateway API の認可モデルをすばやく構築する方法です。これらは、アプリケーションの認可コンポーネントのテスト環境として機能するように設計されています。テストポリシーストアを作成したら、ポリシー、スキーマ、Lambda オーソライザーの改良に時間を費やします。

API のアーキテクチャを調整し、ポリシーストアのスキーマとポリシーに同等の調整が必要になる場合があります。API リンクポリシーストアは、API アーキテクチャからスキーマを自動的に更新しません。検証済みアクセス許可は、ポリシーストアの作成時にのみ API をポーリングします。API が十分に変更される場合は、新しいポリシーストアでプロセスを繰り返す必要があります。

アプリケーションと認可モデルを本番環境にデプロイする準備ができたら、開発した API リンクポリシーストアをオートメーションプロセスと統合します。ベストプラクティスとして、ポリシーストアスキーマとポリシーを他の AWS アカウント および にデプロイできる AWS CloudFormation テンプレートにエクスポートすることをお勧めします AWS リージョン。

API リンクポリシーストアプロセスの結果は、初期ポリシーストアと Lambda オーソライザーです。Lambda オーソライザーには、いくつかの依存リソースがあります。Verified Permissions は、自動的に生成された CloudFormation スタックにこれらのリソースをデプロイします。本番環境にデプロイするには、ポリシーストアと Lambda オーソライザーリソースをテンプレートに収集する必要があります。API リンクポリシーストアは、次のリソースで構成されます。

1. AWS::VerifiedPermissions::PolicyStore: スキーマを `SchemaDefinition` オブジェクトにコピーします。" 文字を として エスケープします\"。
2. AWS::VerifiedPermissions::IdentitySource: テストポリシーストアから GetIdentitySource の出力から値をコピーし、必要に応じて変更します。
3. 1 つ以上の AWS::VerifiedPermissions::Policy: ポリシーステートメントを `Definition` オブジェクトにコピーします。" 文字を として エスケープします\"。
4. AWS::Lambda::Function、AWS::IAM::Role、AWS::IAM::Policy、AWS::ApiGateway::Authorizer、AWS::La

次のテンプレートは、ポリシーストアの例です。Lambda オーソライザーリソースを既存のスタックからこのテンプレートに追加できます。

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

API リンクポリシーストアのトラブルシューティング

Amazon Verified Permissions API にリンクされたポリシーストアを構築する際の一般的な問題の診断と修正には、こちらの情報を参考にしてください。

トピック

- ポリシーを更新しましたが、承認の決定は変更されませんでした
- Lambda オーソライザーを API にアタッチしたが、認可リクエストを生成していない
- 予期しない認可決定を受け取り、認可ロジックを確認したい
- Lambda オーソライザーから口グを見つけたい
- Lambda オーソライザーが存在しない
- API がプライベート VPC にあり、オーソライザーを呼び出すことができない
- 認可モデルで追加のユーザー属性を処理したい
- 新しいアクション、アクションコンテキスト属性、またはリソース属性を追加する

ポリシーを更新しましたが、承認の決定は変更されませんでした

デフォルトでは、Verified Permissions は認可の決定を 120 秒間キャッシュするように Lambda オーソライザーを設定します。2 分後に再試行するか、オーソライザーのキャッシュを無効にします。詳細については、「Amazon API Gateway デベロッパーガイド」の「API キャッシュを有効にして応答性を高める」を参照してください。 Amazon API Gateway

Lambda オーソライザーを API にアタッチしたが、認可リクエストを生成していない

リクエストの処理を開始するには、オーソライザーをアタッチした API ステージをデプロイする必要があります。詳細については、「Amazon API Gateway デベロッパーガイド」の「REST API のデプロイ」を参照してください。 Amazon API Gateway

予期しない認可決定を受け取り、認可ロジックを確認したい

API リンクポリシーストアプロセスは、オーソライザーの Lambda 関数を作成します。Verified Permissions は、認可決定のロジックをオーソライザー関数に自動的に構築します。ポリシーストアを作成した後に戻って、 関数のロジックを確認および更新できます。

AWS CloudFormation コンソールから Lambda 関数を見つけるには、新しいポリシーストアの概要ページにあるデプロイの確認ボタンを選択します。

AWS Lambda コンソールで関数を見つけることもできます。ポリシーストア AWS リージョン の でコンソールに移動し、プレフィックスが の関数名を検索しますAVPAuthorizerLambda。複数の API リンクポリシーストアを作成した場合は、関数の最終変更時刻を使用してポリシーストアの作成と関連付けます。

Lambda オーソライザーからログを見つけたい

Lambda 関数はメトリクスを収集し、その呼び出し結果を Amazon CloudWatch に記録します。ログを確認するには、Lambda コンソールで関数を見つけ、モニタータブを選択します。CloudWatch ログを表示を選択し、ロググループのエントリを確認します。

Lambda 関数ログの詳細については、 AWS Lambda デベロッパーガイドの「 での Amazon CloudWatch Logs の使用 AWS Lambda」を参照してください。

Lambda オーソライザーが存在しない

API リンクポリシーストアのセットアップが完了したら、Lambda オーソライザーを API にアタッチする必要があります。 API Gateway コンソールでオーソライザーが見つからない場合、ポリシー

ストアの追加リソースが失敗したか、まだデプロイされていない可能性があります。API リンクポリシーストアは、これらのリソースを CloudFormation スタックにデプロイします。

Verified Permissions は、作成プロセスの最後に Check deployment というラベルのリンクを表示します。すでにこの画面から移動している場合は、CloudFormation コンソールに移動し、最近のスタックで というプレフィックスが付いた名前を検索しますAVPAuthorizer-<policy store ID>。CloudFormation は、スタックデプロイの出力に貴重なトラブルシューティング情報を提供します。

CloudFormation スタックのトラブルシューティングについては、AWS CloudFormation 「 ユーザーガイド」のCloudFormation のトラブルシューティング」を参照してください。

API がプライベート VPC にあり、オーソライザーを呼び出すことができない

Verified Permissions は、VPC エンドポイントを介した Lambda オーソライザーへのアクセスをサポートしていません。API とオーソライザーとして機能する Lambda 関数間のネットワークパスを開く必要があります。

認可モデルで追加のユーザー属性を処理したい

API リンクポリシーストアプロセスは、ユーザーのトークンのグループクレームから Verified Permissions ポリシーを取得します。追加のユーザー属性を考慮するように認可モデルを更新するには、それらの属性をポリシーに統合します。

Amazon Cognito ユーザープールの ID トークンとアクセストークンの多くのクレームを Verified Permissions ポリシーステートメントにマッピングできます。たとえば、ほとんどのユーザーは ID トークンに emailクレームを持っています。ID ソースからポリシーへのクレームの追加の詳細については、「スキーマへの Amazon Cognito トークンのマッピング」および「スキーマへの OIDC トークンのマッピング」を参照してください。

新しいアクション、アクションコンテキスト属性、またはリソース属性を追加する

API リンクポリシーストアと作成する Lambda オーソライザーは、point-in-timeリソースです。これらは、作成時の API の状態を反映します。ポリシーストアスキーマは、アクションにコンテキスト属性を割り当てたり、デフォルトApplicationリソースに属性や親を割り当てたりしません。

API にパスとメソッドのアクションを追加するときは、新しいアクションを認識するようにポリシーストアを更新する必要があります。また、Lambda オーソライザーを更新して、新しいアクションの認可リクエストを処理する必要があります。新しいポリシーストアで再度開始することも、既存のポリシーストアを更新することもできます。

既存のポリシーストアを更新するには、 関数を見つけます。自動生成された関数のロジックを調べ、新しいアクション、属性、またはコンテキストを処理するように更新します。次に、スキーマを編集して新しいアクションと属性を含めます。

ポリシーストアの削除

Amazon Verified Permissions ポリシーストアは、 AWS マネジメントコンソール または を使用して削除できます AWS CLI。ポリシーストアを削除すると、ポリシーストア内のスキーマ、ポリシー、ポリシーテンプレートが完全に削除されます。削除されたポリシーストアに関連付けられたポリシーストアエイリアスも削除されます。

削除保護は、ポリシーストアの偶発的な削除を防ぎます。削除保護は、 を通じて作成されたすべての新しいポリシーストアで有効になります AWS マネジメントコンソール。対照的に、API または SDK コールで作成されたすべてのポリシーストアでは無効になっています。

ポリシーストアは、次の理由で削除できます。

- 特定のリージョンで使用可能なポリシーストアのクォータに達しました。詳細については、「リソースのクォータ」を参照してください。
- マルチテナントアプリケーションでテナントをサポートしなくなったため、そのポリシーストアが不要になりました。

AWS マネジメントコンソール

ポリシーストアを削除するには

1. Verified Permissions コンソールを開きます。ポリシーストアを選択します。
2. 左側のナビゲーションペインで、[設定] を選択します。
3. [このポリシーストアを削除] を選択します。
4. テキストボックスに `delete` と入力して、[削除]を選択します。

> **Note**
>
> 削除保護が有効になっている場合は、削除を選択する前に無効にする必要があります。無効にするには、削除保護を無効にするを選択します。

AWS CLI

ポリシーストアを削除するには

delete-policy-store オペレーションを使用してポリシーストアを削除し、 を削除するポリシーストア ID *PSEXAMPLEabcdefg111111*に置き換えます。

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

このコマンドが成功した場合、出力は生成されません。

> **Note**
>
> このポリシーストアで削除保護が有効になっている場合は、まず update-policy-storeオペレーションを実行し、削除保護を無効にする必要があります。
>
> ```
> aws verifiedpermissions update-policy-store \
>     --deletion-protection "DISABLED" \
>     --policy-store-id PSEXAMPLEabcdefg111111
> ```

Amazon Verified Permissions ポリシーストアのエイリアス

ポリシーストアエイリアスは、ポリシーストアのわかりやすい名前です。たとえば、ポリシーストアエイリアスを使用すると、 policy-store-alias/example-policy-storeの代わりに を使用してポリシーストアを参照できますPSEXAMPLEabcdefg111111。ポリシーストアエイリアスは、policyStoreId入力パラメータを受け入れる任意の Verified Permissions オペレーションで使用できます。

CreatePolicyStoreAlias API または AWS::VerifiedPermissions::PolicyStoreAlias CloudFormation リソースを使用して、ポリシーストアのポリシーストアエイリアスを作成できます。

Amazon Verified Permissions API は、各 AWS アカウント およびリージョンのポリシーストアエイリアスを完全に制御します。API には、ポリシーストアエイリアス (CreatePolicyStoreAlias) を作成し、ポリシーストアエイリアス名とポリシーストアエイリアス ARNs (GetPolicyStoreAlias、ListPolicyStoreAliases) を表示し、ポリシーストアエイリアス () を削除するオペレーションが含まれていますDeletePolicyStoreAlias。

トピック

- ポリシーストアエイリアスのプロパティ
- Amazon Verified Permissions ポリシーストアエイリアスの作成
- Amazon Verified Permissions ポリシーストアエイリアスの取得
- Amazon Verified Permissions ポリシーストアエイリアスの削除
- API オペレーションでの Amazon Verified Permissions ポリシーストアエイリアスの使用
- ポリシーストアエイリアスへのアクセスの制御

ポリシーストアエイリアスのプロパティ

Amazon Verified Permissions でのポリシーストアエイリアスの仕組み。

ポリシーストアエイリアスは独立した AWS リソースです

ポリシーストアエイリアスは、ポリシーストアのプロパティではありません。ポリシーストアエイリアスに対して実行するアクションは、関連するポリシーストアには影響しません。ポリシーストアエイリアスは、関連するポリシーストアに影響を与えずに削除できます。ポリシーストアを削除すると、そのポリシーストアに関連付けられたすべてのポリシーストアエイリアスも削除されます。

各ポリシーストアエイリアスには、ポリシーストアエイリアスを一意に識別する Amazon リソースネーム (ARN) があります。ポリシーストアエイリアスを IAM ポリシーのリソースとして指定すると、ポリシーは関連するポリシーストアではなく、ポリシーストアエイリアスを参照します。

各ポリシーストアエイリアスには 2 つの形式があります

ポリシーストアエイリアスを作成するときは、ポリシーストアエイリアス名を指定します。Amazon Verified Permissions は、ポリシーストアエイリアス ARN を作成します。

- ポリシーストアエイリアス ARN は、ポリシーストアエイリアスを一意に識別する Amazon リソースネーム (ARN) です。

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

- AWS アカウント および リージョンで一意のポリシーストアエイリアス名。Amazon Verified Permissions API では、ポリシーストアのエイリアス名には常に というプレフィックスが付けられますpolicy-store-alias/。

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

ポリシーストアエイリアスはシークレットではありません

ポリシーストアのエイリアスは、CloudTrail ログやその他の出力でプレーンテキストで表示される場合があります。ポリシーストアのエイリアス名に機密情報や機密情報を含めないでください。

各ポリシーストアエイリアスは、一度に 1 つのポリシーストアに関連付けられます。

ポリシーストアエイリアスとそれに関連付けられたポリシーストアは、同じ AWS アカウント およびリージョンに属している必要があります。ポリシーストアエイリアスは、同じ AWS アカウントおよびリージョン内の任意のポリシーストアに関連付けることができます。

たとえば、このListPolicyStoreAliases出力は、example-policy-storeポリシーストアエイリアスが policyStoreIdプロパティで表される 1 つのターゲットポリシーストアにのみ関連付けられていることを示しています。

```
{
    "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"
}
```

複数のエイリアスを同じポリシーストアに関連付けることができます

たとえば、 example-policy-storeおよび example-policy-store-2 エイリアスを同じポリシーストアに関連付けることができます。

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

ポリシーストアエイリアスは、 AWS アカウント および リージョンで一意である必要があります

たとえば、各 AWS アカウント とリージョンexample-policy-storeに という名前のポリシーストアエイリアスを 1 つだけ持つことができます。ポリシーストアエイリアスでは、大文字と小文字が区別されます。ポリシーストアのエイリアス名を変更することはできません。ただし、24 時間の予約期間が終了したら、ポリシーストアエイリアスを削除し、目的の名前で新しいポリシーストアエイリアスを作成できます。

異なるリージョンで同じ名前のポリシーストアエイリアスを作成できます。各ポリシーストアエイリアスには一意の ARN があります。コードが などのポリシーストアエイリアス名を参照している場合はpolicy-store-alias/example-policy-store、複数のリージョンで実行できます。各リージョンでは、異なるポリシーストアを使用します。

ポリシーストアエイリアスがソフト削除とハード削除をサポート

デフォルトでは、ポリシーストアエイリアスが削除されると、ソフト削除されます。ポリシーストアのエイリアス名は 24 時間予約されます。この期間中に同じ名前のポリシーストアエイリアスを作成しようとすると、リクエストは拒否されます。この期間中、 は PendingDeletion状態のポリシーストアエイリアスGetPolicyStoreAliasを返します。を HardDeleteとして指定することで、ポリシーストアエイリアスをハード削除することもできますdeletionMode。ハード削除されたポリシーストアエイリアスはすぐに削除され、ポリシーストアエイリアス名はすぐに再利用できるようになります。詳細については、「Amazon Verified Permissions ポリシーストアエイリアスの削除」を参照してください。

エイリアスを使用してポリシーストアを識別できます

ポリシーストアエイリアスを使用して、 を受け入れるすべてのオペレーションでポリシーストアを識別できます policyStoreId (例: IsAuthorized)。このような場合、ポリシーストアのエイリアス名には のプレフィックスを付ける必要がありますpolicy-store-alias/。ポリシーストアエイリアスを使用して、DeletePolicyStoreオペレーションのポリシーストアを識別することはできません。

ポリシーストアエイリアス名またはポリシーストアエイリアス ARN を使用して、IAM ポリシーの Resource要素でポリシーストアを識別することはできません。ポリシーストアエイリアスを介して参照されるときにポリシーストアへのアクセスを制御するには、「」を参照してくださいポリシーストアエイリアスへのアクセスの制御。

Amazon Verified Permissions ポリシーストアエイリアスの作成

わかりやすい名前を使用してポリシーストアを参照するポリシーストアエイリアスを作成できます。ポリシーストアエイリアスの名前は、 AWS アカウント および リージョンごとに一意である必要があります。ポリシーストアエイリアスは、ポリシーストアエイリアスと同じリージョンで同じ AWS アカウント およびアクティブな によって所有されているポリシーストアにのみ関連付けることができます。ポリシーストアエイリアスは、独自の ARNsと IAM 認可を持つ個別のリソースです。

デフォルトでは、同じポリシーストアに関連付けることができるポリシーストアエイリアスは 10 個のみです。

> **Note**
>
> CreatePolicyStoreAlias はべき等です。既存のポリシーストアエイリアスに一致するポリシーストアエイリアス名とポリシーストア ID を使用して CreatePolicyStoreAlias オ

ペレーションを呼び出すと、CreatePolicyStoreAliasオペレーションは成功し、既存のポリシーストアエイリアスを返します。ただし、既存のポリシーストアエイリアス名ではなく別のポリシーストア ID を使用して CreatePolicyStoreAlias オペレーションを呼び出すと、オペレーションは を返しますConflictException。

AWS CLI

ポリシーストアエイリアスを作成するには

CreatePolicyStoreAlias オペレーションを使用して、ポリシーストアエイリアスを作成できます。次の例では、 という名前のポリシーストアエイリアスを作成します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"
}
```

Amazon Verified Permissions ポリシーストアエイリアスの取得

GetPolicyStoreAlias オペレーションを使用してポリシーストアエイリアスに関する情報を取得し、特定のポリシーストアエイリアスの詳細を取得するか、 AWS アカウント およびリージョン内のすべてのポリシーストアエイリアスを一覧表示するListPolicyStoreAliasesオペレーションを使用できます。

ポリシーストアエイリアスの取得

GetPolicyStoreAlias オペレーションを使用して、関連するポリシーストア ID など、特定のポリシーストアエイリアスに関する詳細を取得します。

AWS CLI

ポリシーストアエイリアスの詳細を取得するには

GetPolicyStoreAlias オペレーションを使用して、ポリシーストアエイリアスを取得できます。次の の例では、 という名前のポリシーストアエイリアスの詳細を取得しますexample-policy-store。

```
$ 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"
}
```

ポリシーストアエイリアスの一覧表示

ListPolicyStoreAliases オペレーションを使用して、 AWS アカウント および リージョン内のすべてのポリシーストアエイリアスを一覧表示します。filter パラメータを使用して、特定のポリシーストアに関連付けられたポリシーストアエイリアスのみを一覧表示できます。

AWS CLI

すべてのポリシーストアエイリアスを一覧表示するには

ListPolicyStoreAliases オペレーションを使用して、ポリシーストアエイリアスを一覧表示できます。次の の例では、us-west-2 リージョンの 123456789012 AWS アカウント が所有するすべてのポリシーストアエイリアスを一覧表示します。

```
$ 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"
        }
    ]
}
```

特定のポリシーストアのポリシーストアエイリアスを一覧表示するには

filter パラメータを使用して、特定のポリシーストアに関連付けられたエイリアスのみを一覧表示します。

```
$ 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"
```

```
        }
    ]
}
```

Amazon Verified Permissions ポリシーストアエイリアスの削除

不要になったポリシーストアエイリアスは削除できます。ポリシーストアエイリアスを削除しても、関連付けられたポリシーストアには影響しません。ポリシーストアを削除すると、そのポリシーストアに関連付けられているすべてのポリシーストアエイリアスが削除されます。

Amazon Verified Permissions は、ポリシーストアエイリアスに対して 2 つの削除モードをサポートしています。

- ソフト削除 (デフォルト): ポリシーストアエイリアスが PendingDeletion状態になります。ポリシーストアのエイリアス名は 24 時間予約されており、この期間に再利用することはできません。この期間中、 は PendingDeletion状態のポリシーストアエイリアスGetPolicyStoreAliasを返します。これは、 を指定しない場合deletionMode、または を指定した場合のデフォルトの動作ですSoftDelete。
- ハード削除: ポリシーストアのエイリアスはすぐに削除されます。ポリシーストアのエイリアス名はすぐに再利用できるようになります。ハード削除を実行するには、 を HardDeleteとして指定しますdeletionMode。

ポリシーストアエイリアスをソフト削除する

デフォルトでは、ポリシーストアエイリアスを削除すると、ソフト削除が実行されます。ポリシーストアエイリアスは PendingDeletion状態になり、ポリシーストアエイリアス名は 24 時間予約されます。

AWS CLI

ポリシーストアエイリアスをソフト削除するには

DeletePolicyStoreAlias オペレーションを使用して、ポリシーストアエイリアスをソフト削除できます。次の例では、 という名前のポリシーストアエイリアスをソフト削除しますexample-policy-store。

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

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

ソフト削除モードを明示的に指定することもできます。

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

ポリシーストアエイリアスのハード削除

ポリシーストアエイリアスをすぐに削除するには、 を HardDeleteとして指定しますdeletionMode。ハード削除されたポリシーストアエイリアスは PendingDeletion状態に移行せず、ポリシーストアエイリアス名はすぐに再利用できるようになります。

以前にソフト削除されたポリシーストアエイリアスをハード削除した場合、ポリシーストアエイリアスは直ちに削除されます。

> **Important**
>
> Amazon Verified Permissions は結果整合性があります。ポリシーストアエイリアスをハード削除し、別のポリシーストアを指すようにすぐに再作成すると、ポリシーストアエイリアスを参照するリクエストは、以前に関連付けられたポリシーストアに短時間解決し続ける可能性があります。予期しない認可結果を回避するには、同じ名前のポリシーストアエイリアスを再作成する前に、削除が伝播される時間を確保します。

AWS CLI

ポリシーストアエイリアスをハード削除するには

deletion-mode パラメータを に設定して DeletePolicyStoreAlias オペレーションを使用して、ポリシーストアエイリアスをハード削除できますHardDelete。次の例では、 という名前のポリシーストアエイリアスをすぐに削除しますexample-policy-store。

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

API オペレーションでの Amazon Verified Permissions ポリシーストアエイリアスの使用

IsAuthorized、IsAuthorizedWithToken、GetPolicyStore などのpolicyStoreIdパラメータを受け入れる Amazon Verified Permissions オペレーションは、ポリシーストア ID の代わりにポリシーストアエイリアス名を受け入れることができます。

> **Important**
>
> ポリシーストアエイリアスをpolicyStoreIdパラメータの値として使用する場合は、policy-store-alias/ プレフィックスを含める必要があります。たとえば、 policy-store-alias/example-policy-storeではなく を使用しますexample-policy-store。

オペレーションでのポリシーストアエイリアスの使用

次のIsAuthorizedコマンドは、 という名前のポリシーストアエイリアスexample-policy-storeを使用してポリシーストアを識別します。

AWS 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**
>
> DeletePolicyStore オペレーションの policyStoreIdフィールドの代わりにポリシーストアエイリアスを使用することはできません。

全体でのポリシーストアエイリアスの使用 AWS リージョン

エイリアスの最も強力な使用法の 1 つは、アプリケーションを複数の AWS リージョンで実行する場合です。たとえば、各リージョンで異なるポリシーストアを使用するグローバルアプリケーションがあるとします。

- us-east-1 では、 を使用しますPSEXAMPLEabcdefg111111。
- eu-west-1 では、 を使用しますPSEXAMPLEabcdefg222222。

リージョンごとに異なるバージョンのアプリケーションを作成するか、ディクショナリステートメントまたはスイッチステートメントを使用して、リージョンごとに適切なポリシーストアを選択できます。ただし、各リージョンで同じポリシーストアエイリアス名を持つポリシーストアエイリアスを作成する方がはるかに簡単です。ポリシーストアのエイリアス名では大文字と小文字が区別されることに注意してください。

AWS 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
```

次に、コードでポリシーストアエイリアスを使用します。コードが各リージョンで実行されると、ポリシーストアエイリアスはそのリージョン内の関連するポリシーストアを参照します。

AWS 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
```

ただし、ポリシーストアのエイリアスが削除されるリスクがあります。この場合、アプリケーションがポリシーストアのエイリアス名を使用しようとすると失敗し、ポリシーストアのエイリアスを再作

成または更新する必要がある場合があります。このリスクを軽減するには、アプリケーションで使用するポリシーストアエイリアスを管理するアクセス許可をプリンシパルに付与することに注意してください。

ポリシーストアエイリアスへのアクセスの制御

ポリシーストアエイリアスを管理するプリンシパルには、それらのポリシーストアエイリアス、および一部のオペレーションでは、ポリシーストアエイリアスが関連付けられているポリシーストアを操作するアクセス許可が必要です。これらのアクセス許可は、 ポリシーを使用して IAM 指定できます。

以下のセクションでは、ポリシーストアエイリアスの作成と管理に必要なアクセス許可について説明します。

verifiedpermissions:CreatePolicyStoreAlias

ポリシーストアエイリアスを作成するには、プリンシパルに、ポリシーストアエイリアスと関連付けられたポリシーストアの両方に対して次のアクセス許可が必要です。

- verifiedpermissions:CreatePolicyStoreAlias ポリシーストアエイリアスの 。このアクセス許可を、 IAM ポリシーストアエイリアスの作成が許可されているプリンシパルにアタッチされた ポリシーに付与します。

  次のポリシーステートメントの例では、 Resource要素内の特定のポリシーストアエイリアスを指定します。ただし、複数のポリシーストアエイリアス ARNs を一覧表示したり、 などのポリシーストアエイリアスパターンを指定したりできます"sample*"。Resource の値を指定"*"して、プリンシパルが AWS アカウント および リージョンにポリシーストアエイリアスを作成できるようにすることもできます。

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

- verifiedpermissions:CreatePolicyStoreAlias 関連付けられたポリシーストアの 。このアクセス許可は、 IAM ポリシーで指定する必要があります。

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

verifiedpermissions:GetPolicyStoreAlias

特定のポリシーストアエイリアスの詳細を取得するには、プリンシパルに IAM ポリシー内のポリシーストアエイリアスに対するverifiedpermissions:GetPolicyStoreAliasアクセス許可が必要です。

次のポリシーステートメントの例では、特定のポリシーストアエイリアスを取得するアクセス許可をプリンシパルに付与します。

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

verifiedpermissions:ListPolicyStoreAliases

AWS アカウント および リージョンのポリシーストアエイリアスを一覧表示するには、プリンシパルに IAM ポリシーのverifiedpermissions:ListPolicyStoreAliasesアクセス許可が必要です。このポリシーは特定のポリシーストアまたはポリシーストアエイリアスリソースに関連付けられていないため、ポリシーのリソース要素の値は である必要があります"*"。

たとえば、次の IAM ポリシーステートメントは、 内のすべてのポリシーストアエイリアスを一覧表示するアクセス許可をプリンシパルに付与します AWS アカウント。

```
{
  "Sid": "IAMPolicyForListingAliases",
  "Effect": "Allow",
```

```
  "Action": "verifiedpermissions:ListPolicyStoreAliases",
  "Resource": "*"
}
```

verifiedpermissions:DeletePolicyStoreAlias

ポリシーストアエイリアスを削除するには、プリンシパルにポリシーストアエイリアスのみのアクセス許可が必要です。

> **Note**
>
> ポリシーストアエイリアスを削除しても、関連するポリシーストアには影響しませんが、ポリシーストアエイリアスを参照するアプリケーションはエラーを受け取ります。ポリシーストアエイリアスを誤って削除した場合は、24 時間の予約期間後に再作成できます。

プリンシパルには、ポリシーストアエイリアスのverifiedpermissions:DeletePolicyStoreAliasアクセス許可が必要です。このアクセス許可を、 IAM ポリシーストアエイリアスの削除が許可されているプリンシパルにアタッチされた ポリシーに付与します。

次のポリシーステートメントの例では、 Resource要素のポリシーストアエイリアスを指定します。ただし、複数のポリシーストアエイリアス ARNs を一覧表示したり、 などのポリシーストアエイリアスパターンを指定したりできます"sample*"。Resource の値を指定"*"して、プリンシパルが AWS アカウント および リージョンのポリシーストアエイリアスを削除できるようにすることもできます。

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

Policy Store エイリアスのアクセス許可の制限

ポリシーストアエイリアスを使用して、policyStoreIdフィールドを入力として受け入れる任意のオペレーションでポリシーストアを参照できます。これを行うと、Amazon Verified Permissions は

ポリシーストアエイリアスverifiedpermissions:GetPolicyStoreAliasに対して を、関連付けられたポリシーストアに対してリクエストされたオペレーションを承認します。

たとえば、ポリシーストアエイリアスを使用してIsAuthorizedオペレーションを実行する場合、プリンシパルには以下の両方が必要です。

- verifiedpermissions:GetPolicyStoreAlias ポリシーストアエイリアスの アクセス許可
- verifiedpermissions:IsAuthorized 関連付けられたポリシーストアの アクセス許可

次のポリシー例では、特定のポリシーストアエイリアスIsAuthorizedを使用して を呼び出すアクセス許可を付与します。

```
{
  "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"
}
```

プリンシパルが使用できるポリシーストアエイリアスを制限するには、 アクセスverifiedpermissions:GetPolicyStoreAlias許可を制限します。たとえば、次のポリシーでは、プリンシパルが で始まるもの以外のポリシーストアエイリアスを使用することを許可します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 ポリシーストアスキーマ

スキーマは、アプリケーションでサポートされているエンティティタイプの構造と、アプリケーションが承認リクエストで提供する可能性のあるアクションを宣言したものです。Verified Permissions と Cedar がスキーマを処理する方法の違いを確認するには、「」を参照してくださいスキーマのサポート。

詳細については、Cedar ポリシー言語リファレンスガイドの「Cedar スキーマ形式」を参照してください。

> **Note**
>
> Verified Permissions でのスキーマの使用は任意ですが、プロダクションソフトウェアではスキーマの使用を強くお勧めします。新しいポリシーを作成すると、Verified Permissionsはスキーマを使用してスコープと条件で参照されるエンティティと属性を検証できるため、システムの動作を混乱させる恐れのあるポリシーの入力ミスやミスを防ぐことができます。ポリシー検証を有効にする場合は、新しいポリシーはすべてスキーマに準拠している必要があります。

AWS マネジメントコンソール

スキーマを作成するには

1. Verified Permissions コンソールを開きます。ポリシーストアを選択します。
2. 左側のナビゲーションペインで、[スキーマ] を選択します。
3. [スキーマの作成] を選択します。

AWS CLI

新しいスキーマを送信するか、 AWS CLIを使用して既存のスキーマを上書きする。

ポリシーストアを作成するには、次の例のような AWS CLI コマンドを実行します。

次の Cedar コンテンツを含むスキーマを考えてみましょう。

```
{
    "MySampleNamespace": {
```

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

まず JSON を 1 行の文字列にエスケープし、その前にそのデータ型：cedarJsonの宣言を記述する必要があります。次の例では、JSON スキーマのエスケープバージョンを含むschema.jsonファイルの次の内容を使用しています。

> **Note**
>
> この例では、読みやすいように行を折り返しています。コマンドが受け付けるには、ファイル全体を 1 行にまとめる必要があります。

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

```
$ aws verifiedpermissions put-schema \
    --definition file://schema.json \
    --policy-store PSEXAMPLEabcdefg111111
```

```
{
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "namespaces": [
        "MySampleNamespace"
    ],
    "createdDate": "2023-07-17T21:07:43.659196+00:00",
    "lastUpdatedDate": "2023-08-16T17:03:53.081839+00:00"
}
```

AWS SDKs

PutSchemaAPI を使用してポリシーストアを作成できます。詳細については、Amazon Verified Permissions API リファレンスガイドの「PutSchema」を参照してください。

ポリシーストアスキーマの編集

Amazon Verified Permissions コンソールでスキーマを選択すると、スキーマを構成するエンティティタイプとアクションが表示されます。スキーマの編集は、ビジュアルモードまたは JSON モードで表示できます。ビジュアルモードでは、さまざまなウィザードを使用して新しいタイプとアクションを追加することで、スキーマを更新できます。JSON モードを使用すると、JSON エディタでスキーマの JSON コードを直接更新できます。

Visual Mode

ビジュアルスキーマエディタは、スキーマ内のエンティティ間の関係を示す一連の図から始まります。拡大を選択して、図の表示を最大化します。使用可能な図は 2 つあります。

- アクション図 – アクション図ビューには、ポリシーストアで設定したプリンシパルのタイプ、実行できるアクション、およびアクションを実行できるリソースが一覧表示されます。エンティティ間の行は、プリンシパルがリソースに対してアクションを実行できるようにするポリシーを作成する機能を示します。アクション図に 2 つのエンティティ間の関係が示されていない場合は、ポリシーで許可または拒否する前に、それらのエンティティ間の関係を作成する必要があります。エンティティを選択するとプロパティの概要が表示され、ドリルダウンすると詳細が表示されます。この [アクション | リソースタイプ | プリンシパルタイプ] でフィルタリングを選択すると、独自の接続のみを持つビューにエンティティが表示されます。
- エンティティタイプの図 – エンティティタイプの図は、プリンシパルとリソースの関係に焦点を当てています。スキーマ内のネストされた複雑な親関係を理解するには、この図を確認してください。エンティティにカーソルを合わせると、そのエンティティの親関係がドリルダウンされます。

図の下には、スキーマ内のエンティティタイプとアクションのリストビューがあります。リストビューは、特定のアクションまたはエンティティタイプの詳細をすぐに表示する場合に便利です。エンティティを選択して詳細を表示します。

Verified Permissions スキーマをビジュアルモードで編集するには

1. Verified Permissions コンソールを開きます。ポリシーストアを選択します。
2. 左側のナビゲーションペインで、[スキーマ] を選択します。
3. [ビジュアルモード] を選択します。エンティティと関係の図を確認し、スキーマに加える変更を計画します。オプションで、1 つのエンティティでフィルタリングして、他のエンティティへの個々の接続を調べることができます。
4. [Edit schema] を選択します。
5. 「詳細」セクションに、スキーマの名前空間を入力します。
6. 「エンティティタイプ」セクションで、「新しいエンティティタイプを追加」を選択します。
7. エンティティの名前を入力します。
8. (オプション)「親を追加」を選択して、新しいエンティティが属する親エンティティを追加します。エンティティに追加された親を削除するには、親の名前の横にある [削除] を選択します。
9. 属性を追加するには、[属性を追加] を選択します。エンティティの各属性の属性名を入力し、属性タイプを選択します。Verified Permissions では、スキーマに対してポリシーを検証するときに、指定された属性値を使用します。各属性が必須かどうかを選択します。エンティティに追加された属性を削除するには、属性の横にある [削除] を選択します。
10. [エンティティタイプを追加] を選択して、エンティティをスキーマに追加します。
11. [アクション] セクションで、[新しいアクションを追加] を選択します。
12. アクションの名前を入力します。
13. (オプション)「リソースを追加」を選択して、アクションが適用されるリソースタイプを追加します。アクションに追加されたリソースタイプを削除するには、リソースタイプ名の横にある [削除] を選択します。
14. (オプション)「プリンシパルの追加」を選択して、アクションが適用されるプリンシパルタイプを追加します。アクションに追加されたプリンシパルタイプを削除するには、プリンシパルタイプの名前の横にある [削除] を選択します。

15. 属性の追加 を選択して、承認リクエストのアクションのコンテキストに追加できる属性を追加します。属性名を入力し、各属性の属性タイプを選択します。Verified Permissions では、スキーマに対してポリシーを検証するときに、指定された属性値を使用します。各属性が必須かどうかを選択します。アクションに追加された属性を削除するには、属性の横にある [削除] を選択します。
16. [アクションを追加] を選択します。
17. すべてのエンティティタイプとアクションをスキーマに追加したら、[変更を保存] を選択します。

JSON mode

更新中に、JSON エディタがコードを JSON 構文と照合して検証し、編集時にエラーと警告を識別するため、問題をすばやく見つけやすくなります。さらに、JSON のフォーマットについて心配する必要はありません。更新を行ったら、JSON 形式を選択するだけで、形式は予想される JSON 形式に合わせて更新されます。

JSON モードでVerified Permissions スキーマを編集するには

1. Verified Permissions コンソールを開きます。ポリシーストアを選択します。
2. 左側のナビゲーションペインで、[スキーマ] を選択します。
3. JSON モードを選択し、次に [スキーマを編集] を選択します。
4. JSON スキーマのコンテンツを「コンテンツ」フィールドに入力します。構文エラーをすべて解決するまで、更新内容をスキーマに保存することはできません。JSONをフォーマットを選択すると、スキーマの JSON 構文を推奨される間隔とインデントでフォーマットできます。
5. [Save changes] (変更の保存) をクリックします。

Amazon Verified Permissions ポリシー検証モードの有効化

Verified Permissions でポリシー検証モードを設定して、ポリシーの変更をポリシーストア内の スキーマ と照合して検証するかどうかを制御できます。

> **Important**
>
> ポリシー検証をオンにすると、ポリシーまたはポリシーテンプレートを作成または更新しようとする試みはすべて、ポリシーストア内のスキーマと照合されて検証されます。Verified Permissions は、検証が失敗した場合にリクエストの試行を拒否します。このため、アプリケーションの開発中は検証をオフにし、テスト中はオンのままにして、アプリケーションが本番稼働中はオンのままにしておくことをお勧めします。

AWS マネジメントコンソール

ポリシーストアのポリシー検証モードを設定するには

1. Verified Permissions コンソールを開きます。ポリシーストアを選択します。
2. [設定] を選択します。
3. 「ポリシー検証モード」セクションで、「変更」を選択します。
4. 次のいずれかを行います：
   - ポリシー検証を有効にして、すべてのポリシー変更がスキーマに照らして検証されるようにするには、「厳格 (推奨)」ラジオボタンを選択します。
   - ポリシー変更のポリシー検証を無効にするには、「オフ」ラジオボタンを選択します。`confirm`を入力して、ポリシーの更新がスキーマに対して検証されなくなることを確認します。
5. [変更を保存] をクリックします。

AWS CLI

ポリシーストアの検証モードを設定するには

ポリシーストアの検証モードを変更するには、UpdatePolicyStore オペレーションを使用し、ValidationSettings パラメータに別の値を指定します。

```
$ 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"
    }
}
```

詳細については、Cedar ポリシー言語リファレンスガイドの「ポリシー検証」を参照してください。

Amazon Verified Permissions ポリシー

ポリシーは、プリンシパルがリソースに対して 1 つ以上のアクションを実行することを許可または禁止するステートメントです。各ポリシーは、他のすべてのポリシーとは独立して評価されます。Cedar ポリシーの構造と評価方法の詳細については、「Cedar ポリシー言語リファレンスガイド」の「スキーマに対する Cedar ポリシーの検証」を参照してください。

オプションで、ポリシーにポリシー名を割り当てることができます。ポリシー名は、ポリシーストア内のすべてのポリシーで一意で、プレフィックスが である必要がありますname/。policyId パラメータを受け入れるコントロールプレーンオペレーションでは、ポリシー ID の代わりにポリシー名を使用できます。次の例では、ポリシー名を使用して でポリシーを取得しますGetPolicy。

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

> **Important**
>
> プリンシパル、リソース、アクションを参照する Cedar ポリシーを作成する場合、それらの各要素に使用される一意の識別子を定義できます。次のベスト プラクティスに従うことを強くお勧めします。
>
> - すべてのプリンシパル識別子とリソース識別子に汎用一意識別子 (UUIDs) を使用します。
>
> たとえば、ユーザー jane が会社を退職し、後で別のユーザーに jane という名前を使用させた場合、その新しいユーザーは、まだ User::"jane" を参照しているポリシーによって付与されているすべてのものに自動的にアクセスできるようになります。Cedar は、新しいユーザーと古いユーザーを区別できません。これは、プリンシパル ID とリソース ID の両方に適用されます。ポリシーに古い ID が含まれているために意図せずアクセスを許可してしまうことがないよう、常に一意性が保証され、再利用されない識別子を使用してください。
>
> エンティティに UUID を使用する場合は、その後に//コメント指定子とエンティティの「わかりやすい」名前を付けることをお勧めします。これにより、ポリシーがわかりやすくなります。例: principal == Role::"a1b2c3d4-e5f6-a1b2-c3d4-EXAMPLE11111"、// administrators

- プリンシパル、リソースの固有識別子の一部に、個人を特定する情報、機密性の高い情報を含めないでください。これらの識別子は、 AWS CloudTrail 証跡で共有されている口グエントリに含まれます。

トピック

- Amazon Verified Permissions 静的ポリシーの作成
- Amazon Verified Permissions の静的ポリシーの編集
- コンテキストの追加
- Amazon Verified Permissions テストベンチの使用
- Amazon Verified Permissions

Amazon Verified Permissions 静的ポリシーの作成

プリンシパルの静的ポリシーを作成して、アプリケーションの指定されたリソースに対して指定されたアクションを実行することをプリンシパルに許可または禁止できます。静的ポリシーには、`principal`と`resource` に特定の値が含まれており、認可の決定に使用できる準備ができています。

AWS マネジメントコンソール

静的ポリシーを作成するには

1. Verified Permissions コンソールを開きます。ポリシーストアを選択します。
2. 左側のナビゲーションペインで、[ポリシー] を選択します。
3. [ポリシーを作成] を選択し、次に [静的ポリシーを作成] を選択します。

> **Note**
>
> 使用するポリシーステートメントがある場合は、ステップ 8 に進み、次のページのポリシーセクションにポリシーを貼り付けます。

4. 「ポリシー効果」セクションで、リクエストがポリシーに一致した場合にポリシーを「許可」か「禁止」にするかを選択します。許可を選択すると、ポリシーはプリンシパルがリソースに対してアクションを実行することを許可します。逆に、Forbid を選択した場合、ポリシーはプリンシパルがリソースに対してアクションを実行することを許可しません。

5. 「プリンシパルの範囲」フィールドで、ポリシーを適用するプリンシパルの範囲を選択します。

   - 特定のプリンシパルにポリシーを適用するには、「特定のプリンシパル」を選択します。ポリシーで指定されたアクションを実行することを許可または禁止するプリンシパルのエンティティタイプと識別子を指定します。
   - プリンシパルのグループにポリシーを適用するには、「プリンシパルのグループ」を選択します。「プリンシパルのグループ」フィールドにプリンシパルのグループ名を入力します。
   - ポリシーストア内のすべてのプリンシパルにポリシーを適用するには、「すべてのプリンシパル」を選択します。

6. [リソース範囲] フィールドで、ポリシーを適用するリソースの範囲を選択します。

   - 特定のリソースにポリシーを適用するには、「特定のリソース」を選択します。ポリシーを適用するリソースのエンティティタイプと識別子を指定します。
   - [リソースグループ] を選択して、ポリシーをリソースグループに適用します。「リソースグループ」フィールドにリソースグループ名を入力します。
   - ポリシーストア内のすべてのリソースにポリシーを適用するには、[すべてのリソース] を選択します。

7. [アクションの範囲] セクションで、ポリシーを適用するリソースの範囲を選択します。

   - [特定のアクションセット] を選択して、ポリシーをアクションセットに適用します。アクションの横にあるチェックボックスを選択して、ポリシーを適用します。
   - ポリシーストア内のすべてのアクションにポリシーを適用するには、[すべてのアクション] を選択します。

8. [次へ]をクリックします。

9. 「ポリシー」セクションで、Cedar ポリシーを確認します。「フォーマット」を選択すると、ポリシーの構文を推奨間隔とインデントでフォーマットできます。詳細については、「Cedar ポリシー言語リファレンスガイド」の「Cedar における基本ポリシー構築」を参照してください。

10. 「詳細」セクションに、ポリシーの説明を任意で入力します。

11. [Create policy] (ポリシーの作成) を選択します。

AWS CLI

静的ポリシーを作成するには

CreatePolicy オペレーションを使用して、静的インスタンスを作成できます。次の例は、単純な静的ポリシーを作成します。

```
$ 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"
}
```

ポリシー名を使用してポリシーを作成するには

オプションで、ポリシーの作成時にポリシー名を指定できます。名前は、ポリシーストア内のすべてのポリシーで一意で、プレフィックスが である必要がありますname/。ポリシー ID の代わりに名前を使用できます。

```
$ 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**
>
> ポリシーストア内の別のポリシーに既に関連付けられている名前を指定すると、`ConflictException`エラーが発生します。

Amazon Verified Permissions の静的ポリシーの編集

ポリシーストアで既存の静的ポリシーを編集できます。直接更新できるのは、静的ポリシーのみです。テンプレートにリンクされたポリシーを変更するには、ポリシーテンプレートを更新する必要があります。詳細については、「Amazon Verified Permissions ポリシーテンプレートの編集」を参照してください。

静的ポリシーの次の要素を変更できます。

- ポリシーが参照する`action`。
- `when`や`unless`などの条件句。

静的ポリシーの次の要素は変更できません。これらの要素のいずれかを変更するには、ポリシーを削除して再作成する必要があります。

- 静的ポリシーからテンプレートにリンクされたポリシーへのポリシー。
- `permit` または `forbid` からの静的ポリシーの効果。
- 静的ポリシーによって参照される`principal`。
- 静的ポリシーによって参照される`resource`。

AWS マネジメントコンソール

静的ポリシーを編集するには

1. Verified Permissions コンソールを開きます。ポリシーストアを選択します。
2. 左側のナビゲーションペインで、[ポリシー] を選択します。
3. 編集する静的ポリシーの横にあるラジオボタンを選択して、[編集] を選択します。

4. 「ポリシー本文」セクションで、静的ポリシーの action または 条件句を更新します。ポリシーのポリシー効果、principal、またはresourceは更新できません。
5. [ポリシーを更新] を選択します。

> **Note**
>
> ポリシーストアでポリシー検証が有効になっている場合、静的ポリシーを更新すると、Verified Permissions はポリシーストア内のスキーマと照合してポリシーを検証します。更新された静的ポリシーが検証に合格しない場合、操作は失敗し、更新は保存されません。

AWS CLI

静的ポリシーを編集するには

UpdatePolicy オペレーションを使用して静的ポリシーを編集できます。次の例では、単純な静的ポリシーを編集します。

この例では、definition.txtファイルを使用してポリシー定義を格納しています。

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

以下のコマンドはそのファイルを参照します。

```
$ 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"
    }
}
```

ポリシーの名前を更新するには

ポリシーを更新するときに、ポリシー名を設定または更新できます。名前は、ポリシーストア内のすべてのポリシーで一意で、プレフィックスが である必要がありますname/。更新リクエストに名前フィールドを含めない場合、既存の名前は変更されません。名前を削除するには、空の文字列に設定します。

```
$ 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"
    }
}
```

コンテキストの追加

コンテキストは、ポリシーの決定に関連する情報ですが、プリンシパル、アクション、またはリソースのアイデンティティの一部ではありません。アクセストークンクレームはコンテキストです。アクションは、ソース IP アドレスのセットからのみ、またはユーザーが MFA でサインインした場合にのみ許可できます。アプリケーションはこのコンテキストセッションデータにアクセスできるため、承認リクエストに入力する必要があります。Verified Permissions 認可リクエストのコンテキストデータは、 `contextMap`要素で JSON 形式である必要があります。

このコンテンツを説明する例は、サンプルポリシーストアからのものです。その後、テスト環境に DigitalPetStore サンプルポリシーストアを作成します。

次のコンテキストオブジェクトは、サンプル DigitalPetStore ポリシーストアに基づいて、アプリケーションの各 Cedar データ型のいずれかを宣言します。

```
"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"
    }
  }
}
```

認可コンテキストのデータ型

ブール値

バイナリtrueまたはfalse値。この例では、 trueのブール値 は、顧客が注文の表示をリクエストする前に多要素認証を実行したMfaAuthenticatedことを示します。

設定

コンテキスト要素のコレクション。セットメンバーは、この例のようにすべて同じタイプにすることも、ネストされたセットを含む異なるタイプにすることもできます。この例では、顧客は 3 つの異なるアカウントに関連付けられています。

String

文字で囲まれた文字、数字、記号のシーケンス"。この例では、UserAgent文字列は、顧客が注文の表示をリクエストするために使用したブラウザを表します。

Long

整数。この例では、 は、顧客が過去の 4 つの注文を表示するように求めた結果、このリクエストがバッチの一部であるRequestedOrderCountことを示します。

レコード

属性のコレクション。これらの属性は、リクエストコンテキストで宣言する必要があります。スキーマを持つポリシーストアには、このエンティティとスキーマ内のエンティティの属性を含める必要があります。この例では、NetworkInfoレコードには、ユーザーの発信元 IP、クライアントによって決定されるその IP の位置情報、および転送中の暗号化に関する情報が含まれています。

EntityIdentifier

リクエストの entities要素で宣言されたエンティティと属性への参照。この例では、ユーザーの注文は従業員 によって承認されましたBob。

このサンプルコンテキストをサンプル DigitalPetStore アプリケーションでテストするには、リクエスト entities、ポリシーストアスキーマ、静的ポリシーを Customer Role - Get Order という説明で更新する必要があります。

認可コンテキストを受け入れるように DigitalPetStore を変更する

当初、 DigitalPetStore はそれほど複雑なポリシーストアではありません。提示したコンテキストをサポートするために事前設定されたポリシーやコンテキスト属性は含まれません。このコンテキスト情報を使用して認可リクエストの例を評価するには、ポリシーストアと認可リクエストに次の変更を加えます。アクセストークン情報をコンテキストとするコンテキストの例については、Amazon Cognito 「アクセストークンのマッピング」および「OIDC アクセストークンのマッピング」を参照してください。

Schema

新しいコンテキスト属性をサポートするために、ポリシーストアスキーマに次の更新を適用します。GetOrder をactions次のように更新します。

```
"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"
    ]
  }
}
```

リクエストコンテキストNetworkInfoで という名前recordのデータ型を参照するには、の前にスキーマに以下を追加して、スキーマに commonType コンストラクトを作成しますactions。commonType コンストラクトは、異なるエンティティに適用できる属性の共有セットです。

```
"commonTypes": {
  "NetworkInfo": {
    "attributes": {
```

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

Policy

次のポリシーは、指定された各コンテキスト要素が満たす必要がある条件を設定します。Customer Role - Get Order という説明を含む既存の静的ポリシーに基づいて構築されます。このポリシーでは、最初は、リクエストを行うプリンシパルがリソースの所有者である必要があるだけです。

```
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*"
};
```

これで、注文を取得するリクエストが、リクエストに追加した追加のコンテキスト条件を満たすことが要求されました。

1. ユーザーは MFA でサインインしている必要があります。
2. ユーザーのウェブブラウザには、文字列 が含まれているUser-Agent必要がありますMy UserAgent。
3. ユーザーは 4 つ以下の注文を表示するようにリクエストしている必要があります。
4. ユーザーのアカウントコードの 1 つは である必要があります111122223333。
5. ユーザーの IP アドレスは米国から送信され、暗号化されたセッションにあり、IP アドレスはで始まる必要があります192.0.2.。
6. 従業員は注文を承認している必要があります。認可リクエストの entities要素で、 のロールBobを持つユーザーを宣言しますEmployee。

Request body

適切なスキーマとポリシーを使用してポリシーストアを設定したら、この認可リクエストを Verified Permissions API オペレーション IsAuthorized に提示できます。entities セグメントにはBob、ロールが のユーザーである の定義が含まれていることに注意してくださいEmployee。

```
{
  "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"
}
```

Amazon Verified Permissions テストベンチの使用

Verified Permissions テストベンチを使用して、認可リクエストを実行して Verified Permissions ポリシーをテストおよびトラブルシューティングします。テストベンチでは、指定したパラメータを使用して、ポリシーストア内の Cedar ポリシーがリクエストを認可するかどうかを判断します。認可リクエストのテスト中に、ビジュアルモードと JSON モードを切り替えることができます。Cedar

ポリシーがどのように構成され、評価されるかについての詳細は、Cedar ポリシー言語リファレンスガイドの「Cedar における基本ポリシー構築」を参照してください。

> **Note**
>
> Verified Permissions を使用して認可リクエストを行う場合は、追加のエンティティセクションのリクエストの一部としてプリンシパルとリソースのリストを指定できます。ただし、アクションの詳細を含めることはできません。スキーマで指定するか、リクエストから推測する必要があります。追加のエンティティセクションにはアクションを設定できません。

テストベンチの視覚的な概要とデモンストレーションについては、 AWS YouTube チャンネルの「Amazon Verified Permissions - Policy Creation and Testing (Primer Series #3)」を参照してください。

Visual mode

> **Note**
>
> テストベンチのビジュアルモードを使用するには、ポリシーストアにスキーマが定義されている必要があります。

ポリシーをビジュアルモードでテストするには

1. Verified Permissions コンソールを開きます。ポリシーストアを選択します。
2. 左側のナビゲーションペインで、[テストベンチ] を選択します。
3. [ビジュアルモード] を選択します。
4. [プリンシパル] セクションで、スキーマのプリンシパルタイプから[プリンシパルによるアクション] を選択します。テキストボックスにプリンシパルの識別子を入力します。
5. (オプション) [親を追加] を選択して、指定したプリンシパルの親エンティティを追加します。プリンシパルに追加された親を削除するには、親の名前の横にある [削除] を選択します。
6. 指定したプリンシパルの各属性の属性値を指定します。テストベンチは、シミュレートされた認可リクエストで指定された属性値を使用します。
7. [リソース] セクションで、[プリンシパルがアクションを実行しているリソース] を選択します。テキストボックスにリソースの識別子を入力します。

8. (オプション) [親を追加] を選択して、指定したリソースの親エンティティを追加します。リソースに追加された親を削除するには、親の名前の横にある [削除] を選択します。
9. 指定したリソースの各属性に Attribute 値 を指定します。テストベンチは、シミュレートされた認可リクエストで指定された属性値を使用します。
10. [アクション] セクションで、指定したプリンシパルとリソースに対して有効なアクションのリストから、プリンシパルが実行しているアクション を選択します。
11. 指定したアクションの各属性の属性値を指定します。テストベンチは、シミュレートされた認可リクエストで指定された属性値を使用します。
12. (オプション) エンティティの追加 セクションで、エンティティの追加 を選択して、承認決定のために評価するエンティティを追加します。
13. ドロップダウンリストから[エンティティ識別子] を選択し、エンティティ識別子を入力します。
14. (オプション) [親を追加] を選択して、指定したエンティティの親エンティティを追加します。エンティティに追加された親を削除するには、親の名前の横にある [削除] を選択します。
15. 指定したエンティティの各属性に [属性値] を指定します。テストベンチは、シミュレートされた認可リクエストで指定された属性値を使用します。
16. [確認] を選択して、テストベンチにエンティティを追加します。
17. 認可リクエストの実行を選択して、ポリシーストアの Cedar ポリシーの認可リクエストをシミュレートします。テストベンチには、リクエストを許可するか拒否するかの決定と、満たされているポリシーまたは評価中に発生したエラーに関する情報が表示されます。

JSON mode

JSON モードでポリシーをテストするには

1. Verified Permissions コンソールを開きます。ポリシーストアを選択します。
2. 左側のナビゲーションペインで、[テストベンチ] を選択します。
3. [JSON モード] を選択します。
4. [リクエストの詳細] セクションで、スキーマを定義している場合は、スキーマのプリンシパルタイプから[プリンシパルによるアクション] を選択します。テキストボックスにプリンシパルの識別子を入力します。

   スキーマが定義されていない場合は、[プリンシパルによるアクション] テキストボックスにプリンシパルを入力します。

5. スキーマが定義されている場合は、スキーマ内のリソースタイプから [リソース] を選択します。テキストボックスにリソースの識別子を入力します。

   スキーマが定義されていない場合は、[リソース] テキストボックスにリソースを入力します。
6. スキーマが定義されている場合は、指定したプリンシパルとリソースの有効なアクションのリストから [アクション] を選択します。

   スキーマが定義されていない場合は、[アクション] テキストボックスにアクションを入力します。
7. コンテキストフィールドに、シミュレートするリクエストのコンテキストを入力します。リクエストコンテキストは、認可の決定に使用できる追加情報です。
8. エンティティフィールドに、認可決定について評価されるエンティティとその属性の階層を入力します。
9. 認可リクエストの実行を選択して、ポリシーストア内の Cedar ポリシーの認可リクエストをシミュレートします。テストベンチには、リクエストを許可するか拒否するかの決定と、満たされているポリシーまたは評価中に発生したエラーに関する情報が表示されます。

Amazon Verified Permissions

ここに含まれるポリシーの例の一部は、基本的な Cedar ポリシーの例であり、一部は Verified Permissions 固有です。基本的なものは、Cedar ポリシー言語リファレンスガイドにリンクされており、そこに含まれています。Cedar ポリシー構文の詳細については、「Cedar ポリシー言語リファレンスガイド」の「Cedar における基本的なポリシー構築」を参照してください。

ポリシーの例

- 個々のエンティティへのアクセスを許可する
- エンティティのグループへのアクセスを許可する
- 任意のエンティティへのアクセスを許可する
- エンティティの属性へのアクセスを許可する (ABAC)
- アクセスを拒否する
- 角括弧表記を使用してトークン属性を参照します
- ドット表記を使用して属性を参照します
- Amazon Cognito ID トークン属性を反映

- OIDC ID トークン属性を反映
- Amazon Cognito アクセストークン属性を反映
- OIDC アクセストークン属性を反映

角括弧表記を使用してトークン属性を参照します

次の例は、ブラケット表記を使用してトークン属性を参照するポリシーを作成する方法を示しています。

Verified Permissions のポリシーでトークン属性を使用する方法の詳細については、「スキーマへの Amazon Cognito トークンのマッピング」および「スキーマへの OIDC トークンのマッピング」を参照してください。

```
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.*"
};
```

ドット表記を使用して属性を参照します

次の例は、ドット表記を使用して属性を参照するポリシーを作成する方法を示しています。

Verified Permissions のポリシーでトークン属性を使用する方法の詳細については、「スキーマへの Amazon Cognito トークンのマッピング」および「スキーマへの OIDC トークンのマッピング」を参照してください。

```
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"
};
```

Amazon Cognito ID トークン属性を反映

次の例は、ID トークン属性を参照するポリシーの作成方法を示しています Amazon Cognito。

Verified Permissions のポリシーでトークン属性を使用する方法の詳細については、「スキーマへの Amazon Cognito トークンのマッピング」および「スキーマへの OIDC トークンのマッピング」を参照してください。

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

OIDC ID トークン属性を反映

次の例は、OIDC プロバイダーから ID トークン属性を参照するポリシーを作成する方法を示しています。

Verified Permissions のポリシーでトークン属性を使用する方法の詳細については、「スキーマへの Amazon Cognito トークンのマッピング」および「スキーマへの OIDC トークンのマッピング」を参照してください。

```
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*"
};
```

Amazon Cognito アクセストークン属性を反映

次の例は、アクセストークン属性を参照するポリシーを作成する方法を示しています Amazon Cognito。

Verified Permissions のポリシーでトークン属性を使用する方法の詳細については、「スキーマへの Amazon Cognito トークンのマッピング」および「スキーマへの OIDC トークンのマッピング」を参照してください。

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

OIDC アクセストークン属性を反映

次の例は、OIDC プロバイダーからアクセストークン属性を参照するポリシーを作成する方法を示しています。

Verified Permissions のポリシーでトークン属性を使用する方法の詳細については、「スキーマへの Amazon Cognito トークンのマッピング」および「スキーマへの OIDC トークンのマッピング」を参照してください。

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

Amazon Verified Permissions ポリシーテンプレートとテンプレートにリンクされたポリシー

Verified Permissions では、ポリシーテンプレートは、principal、、resourceまたはその両方のプレースホルダーを持つポリシーです。ポリシーテンプレートのみを使用して認可リクエストを処理することはできません。認可リクエストを処理するには、ポリシーテンプレートに基づいてテンプレートにリンクされたポリシーを作成する必要があります。ポリシーテンプレートを使用すると、ポリシーを一度定義してから、複数のプリンシパルとリソースで使用できます。ポリシーテンプレートの更新は、テンプレートを使用するすべてのポリシーに反映されます。詳細については、Cedar ポリシー言語リファレンスガイドの「Cedar ポリシーテンプレート」を参照してください。

必要に応じて、ポリシーテンプレート名をポリシーテンプレートに割り当てることができます。ポリシーテンプレート名は、ポリシーストア内で一意で、プレフィックスが である必要がありますname/。policyTemplateId パラメータを受け入れるコントロールプレーンオペレーションでは、ポリシーテンプレート ID の代わりにポリシーテンプレート名を使用できます。GetPolicyTemplate と のみ出力で名前ListPolicyTemplatesを返します。次の例では、ポリシーテンプレート名を使用して、 でポリシーテンプレートを取得しますGetPolicyTemplate。

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

たとえば、次のポリシーテンプレートはRead、ポリシーテンプレートを使用するプリンシパルとリソースに Edit、、および アクセスComment許可を提供します。

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

このテンプレートEditorに基づいて という名前のポリシーを作成する場合、プリンシパルが特定のリソースの エディタとして指定されていると、アプリケーションはプリンシパルがリソースに対して読み取り、編集、コメントするためのアクセス許可を提供するポリシーを作成します。

静的ポリシーとは異なり、テンプレートにリンクされたポリシーは動的です。前の例で、ポリシーテンプレートからCommentアクションを削除すると、そのテンプレートにリンクまたは基づいている

ポリシーはそれに応じて更新され、ポリシーで指定されたプリンシパルは対応するリソースにコメントできなくなります。

テンプレートにリンクされたポリシーの例については、「」を参照してくださいAmazon Verified Permissions テンプレートにリンクされたポリシーの例。

Amazon Verified Permissions ポリシーテンプレートの作成

Verified Permissions でポリシーテンプレートを作成するには AWS マネジメントコンソール、、AWS CLI、または AWS SDKs を使用します。ポリシーテンプレートを使用すると、ポリシーを一度定義してから、複数のプリンシパルとリソースで使用できます。ポリシーテンプレートを作成したら、テンプレートにリンクされたポリシーを作成して、特定のプリンシパルとリソースでポリシーテンプレートを使用できます。詳細については、「Amazon Verified Permissions テンプレートリンクポリシーの作成」を参照してください。

AWS マネジメントコンソール

ポリシーテンプレートを作成するには

1. Verified Permissions コンソールを開きます。ポリシーストアを選択します。
2. 左側のナビゲーションペインで、[ポリシーテンプレート] を選択します。
3. [ポリシーテンプレートを作成] を選択します。
4. [詳細] セクションにポリシーテンプレートの説明を入力します。
5. [ポリシーテンプレート本文] セクションでは、プレースホルダー`?principal` および`?resource` を使用して、このテンプレートに基づいて作成されたポリシーが、付与する権限をカスタマイズできるようにします。[フォーマット] を選択すると、ポリシーテンプレートの構文を推奨される間隔とインデントでフォーマットできます。
6. [ポリシーテンプレートを作成] を選択します。

AWS CLI

ポリシーテンプレートを作成するには

CreatePolicyTemplate オペレーションを使用して、ポリシーテンプレートを作成できます。次の例では、プリンシパルのプレースホルダーを含むポリシーテンプレートを作成します。

ファイル `template1.txt` には次のものが含まれています。

```
"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"
}
```

ポリシーテンプレート名を使用してポリシーテンプレートを作成するには

オプションで、ポリシーテンプレートの作成時にポリシーテンプレート名を指定できます。名前は、ポリシーストア内のすべてのポリシーテンプレートで一意で、プレフィックスが である必要がありますname/。ポリシーテンプレート ID の代わりに名前を使用できます。

```
$ 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**
>
> ポリシーストア内の別のポリシーテンプレートに既に関連付けられている名前を指定すると、ConflictException エラーが発生します。

Amazon Verified Permissions テンプレートリンクポリシーの作成

テンプレートにリンクされたポリシー、またはポリシーテンプレートに基づくポリシーは、、 AWS CLI、または AWS SDKs AWS マネジメントコンソールを使用して作成できます。テンプレートにリンクされたポリシーは、ポリシーテンプレートにリンクされたままになります。ポリシーテンプレートでポリシーステートメントを変更すると、そのテンプレートにリンクされたポリシーは、その時点から行われたすべての承認決定に対して新しいステートメントを自動的に使用します。

テンプレートにリンクされたポリシーの例については、「」を参照してくださいAmazon Verified Permissions テンプレートにリンクされたポリシーの例。

AWS マネジメントコンソール

ポリシーテンプレートをインスタンス化してテンプレートにリンクされたポリシーを作成するには

1. Verified Permissions コンソールを開きます。ポリシーストアを選択します。
2. 左側のナビゲーションペインで、[ポリシー] を選択します。
3. [ポリシーを作成] を選択し、[テンプレートにリンクされたポリシーを作成] を選択します。
4. 使用するポリシーテンプレートの横にあるラジオボタンを選択して、[次へ] を選択します。
5. テンプレートにリンクされたポリシーのこの特定のインスタンスに使用するプリンシパルとリソースを入力します。指定した値は [ポリシーステートメントのプレビュー] フィールドに表示されます。

   > **Note**
   >
   > プリンシパルとリソースの値は、静的ポリシーと同じ形式にする必要があります。例えば、プリンシパルの `AdminUsers` グループを指定するには、`Group::"AdminUsers"` と入力します。`AdminUsers` を入力すると、検証エラーが表示されます。

6. [テンプレートにリンクされたポリシーを作成] を選択します。

   テンプレートにリンクされた新しいポリシーが [ポリシー] の下に表示されます。

AWS CLI

ポリシーテンプレートをインスタンス化してテンプレートにリンクされたポリシーを作成するには

既存のポリシーテンプレートを参照し、テンプレートで使用されるすべてのプレースホルダの値を指定するテンプレートリンクポリシーを作成できます。

次の例では、次のステートメントがあるテンプレートを使用する、テンプレートにリンクされるポリシーを作成しています。

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

また、次の definition.txt ファイルを使用して definition パラメータの値を指定します。

```
{
    "templateLinked": {
        "policyTemplateId": "PTEXAMPLEabcdefg111111",
        "principal": {
            "entityType": "PhotoFlash::User",
            "entityId": "alice"
        }
    }
}
```

出力には、テンプレートから取得したリソースと、定義パラメータから取得したプリンシパルの両方が表示されます。

```
$ 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"
    }
}
```

Amazon Verified Permissions ポリシーテンプレートの編集

Verified Permissions のポリシーテンプレートを編集または更新するには、 AWS マネジメントコンソール、 AWS CLI、または AWS SDKs を使用します。ポリシーテンプレートを編集すると、テンプレートにリンクされているか、テンプレートに基づいているポリシーが自動的に更新されます。そのため、ポリシーテンプレートを編集するときは注意し、アプリケーションを壊す変更を誤って導入しないようにしてください。

ポリシーテンプレートの次の要素を変更できます。

- ポリシーテンプレートによってaction参照される
- when や などの条件unless句

ポリシーテンプレートの次の要素は変更できません。これらの要素のいずれかを変更するには、ポリシーテンプレートを削除して再作成する必要があります。

- permit または からのポリシーテンプレートの効果 forbid
- ポリシーテンプレートによってprincipal参照される
- ポリシーテンプレートによってresource参照される

AWS マネジメントコンソール

ポリシーテンプレートを編集するには

1. Verified Permissions コンソールを開きます。ポリシーストアを選択します。
2. 左側のナビゲーションペインで、[ポリシーテンプレート] を選択します。コンソールには、現在のポリシーストアで作成したすべてのポリシーテンプレートが表示されます。

3. ポリシーテンプレートの横にあるラジオボタンを選択すると、ポリシーテンプレートの作成と更新のタイミングやポリシーテンプレートの内容など、ポリシーテンプレートに関する詳細が表示されます。
4. [編集] を選択してポリシーテンプレートを編集します。必要に応じて [ポリシーの説明] と [ポリシー本文] を更新し、[ポリシーテンプレートを更新] を選択します。
5. ポリシーテンプレートの横にあるラジオボタンを選択して [削除] を選択すると、ポリシーテンプレートを削除できます。[OK] を選択して、ポリシーテンプレートの削除を確定します。

AWS CLI

ポリシーテンプレートを編集するには

UpdatePolicy オペレーションを使用して、静的ポリシーを作成できます。次の例では、ファイルに定義されている新しいポリシーにポリシー本文を置き換えることにより、指定されたポリシーテンプレートを更新します。

`template1.txt` ファイルの内容。

```
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"
}
```

ポリシーテンプレートの名前を更新するには

ポリシーテンプレートを更新するときに、ポリシーテンプレート名を設定または更新できます。名前は、ポリシーストア内のすべてのポリシーテンプレートで一意で、プレフィックスが である必要がありますname/。更新リクエストに名前フィールドを含めない場合、既存の名前は変更されません。名前を削除するには、空の文字列に設定します。

```
$ 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 Verified Permissions テンプレートにリンクされたポリシーの例

サンプルポリシーストア方法を使用して Verified Permissions でポリシーストアを作成すると、選択したサンプルプロジェクトの事前定義ポリシー、ポリシーテンプレート、およびスキーマを使用してポリシーストアが作成されます。以下の Verified Permissions テンプレートにリンクされたポリシー例は、サンプルポリシーストアとそれぞれのポリシー、ポリシーテンプレート、およびスキーマで使用できます。

PhotoFlash の例

次の例は、ポリシーテンプレートを使用するテンプレートにリンクされたポリシーを作成する方法を示しています。個々のユーザーと写真を持つ非プライベート共有写真への制限付きアクセスを付与します。

> **Note**
>
> Cedar のポリシー言語では、エンティティをin自体とみなします。したがって、principal in User::"Alice"はprincipal == User::"Alice"と同等です。

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

次の例は、ポリシーテンプレートを使用するテンプレートにリンクされたポリシーを作成する方法を示しています。個々のユーザーとアルバムを持つ非プライベート共有写真への制限付きアクセスを付与します。

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

次の例は、ポリシーテンプレートを使用するテンプレートにリンクされたポリシーを作成する方法を示しています。友人グループと個々の写真を持つ非プライベート共有写真への制限付きアクセスを付与します。

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

次の例は、ポリシーテンプレートを使用するテンプレートにリンクされたポリシーを作成する方法を示しています。友人グループとアルバムを持つ非プライベート共有写真への制限付きアクセスを付与します。

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

次の例は、ポリシーテンプレートを使用するテンプレートにリンクされたポリシーを作成する方法を示しています。友人グループと個々の写真を使用して、非プライベート共有写真へのフルアクセスを付与します。

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

次の例は、 アカウントからポリシーテンプレートブロックユーザーを使用するテンプレートにリンクされたポリシーを作成する方法を示しています。

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

DigitalPetStore の例

DigitalPetStore サンプルポリシーストアにはポリシーテンプレートは一切含まれていません。DigitalPetStore サンプルポリシーストアを作成した後、左側のナビゲーションペインで [ポリシー] を選択すると、ポリシーストアに含まれるポリシーを表示できます。

TinyToDo の例

次の例は、個々のユーザーとタスクリストに対するビューワーアクセスを許可するポリシーテンプレートを使用するテンプレートにリンクされたポリシーを作成する方法を示しています。

```
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"
);
```

次の例は、個々のユーザーとタスクリストに対するエディタアクセスを許可するポリシーテンプレートを使用するテンプレートにリンクされたポリシーを作成する方法を示しています。

```
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"
);
```

ID ソースとトークンを使用してアプリケーションを保護する

Amazon Verified Permissions で外部 ID プロバイダー (IdP) を表す ID ソースを作成して、アプリケーションをすばやく保護します。ID ソースは、ポリシーストアと信頼関係がある IdP で認証されたユーザーからの情報を提供します。アプリケーションが ID ソースからのトークンを使用して認可リクエストを行うと、ポリシーストアはユーザープロパティとアクセス許可から認可を決定できます。Amazon Cognito ユーザープールまたはカスタム OpenID Connect (OIDC) IdP を ID ソースとして追加できます。

Verified Permissions で OpenID Connect (OIDC) ID プロバイダー (IdPs) を使用できます。アプリケーションは、OIDC 準拠の ID プロバイダーによって生成された JSON ウェブトークン (JWTs) を使用して認可リクエストを生成できます。トークンのユーザー ID はプリンシパル ID にマッピングされます。ID トークンを使用すると、Verified Permissions は属性クレームをプリンシパル属性にマッピングします。アクセストークンでは、これらのクレームはコンテキストにマッピングされます。どちらのトークンタイプでも、 のようなクレームgroupsをプリンシパルグループにマッピングし、ロールベースのアクセスコントロール (RBAC) を評価するポリシーを構築できます。

> **Note**
>
> Verified Permissions は、IdP トークンからの情報に基づいて認可の決定を行いますが、IdP と直接やり取りすることはありません。

Amazon Cognito ユーザープールまたは OIDC ID プロバイダーを使用して Amazon API Gateway REST APIs の認可ロジックを構築するstep-by-stepのチュートリアルについては、 AWS セキュリティブログのAPI Gateway APIs」または「独自の ID プロバイダー Amazon Cognito を持ち込む」を参照してください。

トピック

- 適切な ID プロバイダーの選択
- Amazon Cognito ID ソースの使用
- OIDC ID ソースの使用

適切な ID プロバイダーの選択

Verified Permissions はさまざまな IdPs で動作しますが、アプリケーションで使用する IdP を決定するときは、次の点を考慮してください。

次の Amazon Cognito 場合に を使用します。

- 既存の ID インフラストラクチャなしで新しいアプリケーションを構築している
- セキュリティ機能が組み込まれた AWSマネージドユーザープールが必要
- ソーシャル ID プロバイダーの統合が必要です
- トークン管理を簡素化したい

OIDC プロバイダーは、次の場合に使用します。

- 既存の ID インフラストラクチャ (Auth0、Okta、Azure AD) がある
- 一元化されたユーザー管理を維持する必要があります
- 特定の IdPs のコンプライアンス要件がある

Amazon Cognito ID ソースの使用

Verified Permissions は Amazon Cognito ユーザープールと密接に連携します。 Amazon Cognito JWTsは予測可能な構造です。Verified Permissions はこの構造を認識し、含まれている情報から最大限のメリットを得ます。たとえば、ID トークンまたはアクセストークンを使用して、ロールベースのアクセスコントロール (RBAC) 認可モデルを実装できます。

新しい Amazon Cognito ユーザープール ID ソースには、次の情報が必要です。

- AWS リージョン。
- ユーザープール ID。
- ID ソースに関連付けるプリンシパルエンティティタイプ。例: `MyCorp::User`。
- ID ソースに関連付けるプリンシパルグループエンティティタイプ。例: `MyCorp::UserGroup`。
- ポリシーストアへのリクエストを許可するユーザープールのクライアント IDs。

Verified Permissions は同じ 内の Amazon Cognito ユーザープールでのみ機能するため AWS アカウント、別のアカウントで ID ソースを指定することはできません。Verified Permissions は、ユーザープールプリンシパルを操作するポリシーで参照する必要がある ID ソース識別子であるエンティティプレフィックスを、 などのユーザープールの ID に設定しますus-west-2_EXAMPLE。この場

合、ID を持つユーザープール内のユーザーを a1b2c3d4-5678-90ab-cdef-EXAMPLE22222 として参照します。 us-west-2_EXAMPLE|a1b2c3d4-5678-90ab-cdef-EXAMPLE22222

ユーザープールトークンクレームには、属性、スコープ、グループ、クライアント IDs、カスタムデータを含めることができます。Amazon Cognito JWTs には、Verified Permissions の承認決定に役立つさまざまな情報を含める機能があります。具体的には次のとおりです。

1. cognito: プレフィックス付きのユーザー名およびグループクレーム
2. を使用したカスタムユーザー属性 custom: prefix
3. 実行時に追加されたカスタムクレーム
4. sub や などの OIDC 標準クレーム email

これらのクレームの詳細と、 の Verified Permissions ポリシーでそれらを管理する方法について説明しますAmazon Cognito トークンをスキーマにマッピングする。

> **Important**
>
> Amazon Cognito トークンは有効期限が切れる前に取り消すことができますが、JWTsは署名と有効性を備えたステートレスリソースと見なされます。JSON ウェブトークン RFC 7519 に準拠するサービスは、トークンをリモートで検証することが想定されており、発行者による検証は必須ではありません。つまり、Verified Permissions は、後で削除されたユーザーに対して取り消されたトークンまたは発行されたトークンに基づいてアクセスを許可できます。このリスクを軽減するために、有効期間をできるだけ短くしてトークンを作成し、ユーザーのセッションを継続する権限を削除したい場合は更新トークンを取り消すことをお勧めします。詳細については、「トークン失効によるユーザーセッションの終了」を参照してください。

次の例は、プリンシパルに関連付けられた Amazon Cognito ユーザープールクレームの一部を参照するポリシーを作成する方法を示しています。

```
permit(
     principal,
     action,
     resource == ExampleCo::Photo::"VacationPhoto94.jpg"
)
when {
     principal["cognito:username"]) == "alice" &&
```

```
     principal["custom:department"]) == "Finance"
};
```

次の例は、Cognito ユーザープール内のユーザーであるプリンシパルを参照するポリシーを作成する方法を示しています。プリンシパル ID は の形式であることに注意してください"<userpool-id>|<sub>"。

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

Verified Permissions のユーザープール ID ソースの Cedar ポリシーは、英数字とアンダースコア () 以外の文字を含むクレーム名に特別な構文を使用します_。これには、cognito:usernameや などの:文字を含むユーザープールプレフィックスクレームが含まれますcustom:department。cognito:username または custom:departmentクレームを参照するポリシー条件を記述するには、principal["custom:department"]それぞれprincipal["cognito:username"]および として記述します。

> **Note**
>
> トークンに cognito:または custom:プレフィックスを持つクレームと、リテラル値cognitoまたは を持つクレーム名が含まれている場合custom、IsAuthorizedWithToken を使用した認可リクエストは で失敗しますValidationException。

クレームのマッピングの詳細については、「」を参照してくださいAmazon Cognito トークンをスキーマにマッピングする。 Amazon Cognito ユーザーの認可の詳細については、「Amazon Cognito デベロッパーガイド」の「Amazon Verified Permissions を使用した認可」を参照してください。Amazon Cognito

トピック

- Amazon Verified Permissions Amazon Cognito ID ソースの作成
- Amazon Verified Permissions Amazon Cognito ID ソースの編集
- Amazon Cognito トークンをスキーマにマッピングする
- のクライアントとオーディエンスの検証 Amazon Cognito

Amazon Verified Permissions Amazon Cognito ID ソースの作成

次の手順では、ID ソースを既存のポリシーストアに追加します。

Verified Permissions コンソールで新しいポリシーストアを作成するときに、ID ソースを作成することもできます。このプロセスでは、ID ソーストークンのクレームをエンティティ属性に自動的にインポートできます。ガイド付きセットアップを選択するか、 API Gateway と ID プロバイダーオプションを使用してセットアップします。これらのオプションでは、初期ポリシーも作成されます。

> **Note**
>
> ID ソース は、ポリシーストアを作成するまでは左側のナビゲーションペインには表示されません。作成する ID ソースは、現在のポリシーストアに関連付けられます。

Verified Permissions API の create-identity-source AWS CLI または CreateIdentitySource を使用して ID ソースを作成するときに、プリンシパルエンティティタイプを除外できます。ただし、空のエンティティタイプは、エンティティタイプが の ID ソースを作成しますAWS::Cognito。このエンティティ名は、ポリシーストアスキーマと互換性がありません。 Amazon Cognito ID をポリシーストアスキーマと統合するには、プリンシパルエンティティタイプをサポートされているポリシーストアエンティティに設定する必要があります。

AWS マネジメントコンソール

Amazon Cognito ユーザープール ID ソースを作成するには

1. Verified Permissions コンソールを開きます。ポリシーストアを選択します。
2. 左側にあるナビゲーションペインで、[ID ソース] を選択します。
3. [ID ソースを作成]を選択します。
4. Cognito ユーザープールの詳細で、 を選択し、ID ソースのユーザープール ID AWS リージョン を入力します。
5. プリンシパル設定のプリンシパルタイプで、このソースからプリンシパルのエンティティタイプを選択します。接続された Amazon Cognito ユーザープールの ID は、選択したプリンシパルタイプにマッピングされます。
6. グループ設定で、ユーザープールcognito:groupsクレームをマッピングする場合は Cognito グループを使用するを選択します。プリンシパルタイプの親であるエンティティタイプを選択します。

7. クライアントアプリケーションの検証で、クライアントアプリケーション IDsを検証するかどうかを選択します。

   - クライアントアプリケーション ID を検証するには、「クライアントアプリケーション ID が一致するトークンのみを受け入れる」を選択します。検証するクライアントアプリケーション ID ごとに [新しいクライアントアプリケーション ID を追加] を選択します。追加したクライアントアプリケーション ID を削除するには、クライアントアプリケーション ID の横にある [削除] を選択します。
   - クライアントアプリケーション ID を検証したくない場合は、[クライアントアプリケーション ID を検証しない]を選択します。

8. [ID ソースを作成]を選択します。
9. (オプション) ポリシーストアにスキーマがある場合、Cedar ポリシーのアイデンティティトークンまたはアクセストークンから抽出した属性を参照する前に、スキーマを更新して、ID ソースが作成するプリンシパルのタイプを Cedar に認識させる必要があります。スキーマへの追加には、Cedar ポリシーで参照したい属性を含める必要があります。 Amazon Cognito トークン属性を Cedar プリンシパル属性にマッピングする方法の詳細については、「」を参照してくださいAmazon Cognito トークンをスキーマにマッピングする。

   > **Note**
   >
   > API リンクポリシーストアを作成するか、ポリシーストアの作成時に API Gateway と ID プロバイダーのセットアップを使用するとき、Verified Permissions はユーザープールにユーザー属性をクエリし、プリンシパルタイプにユーザープール属性が入力されるスキーマを作成します。

10. トークンからの情報を使用して認可を決定するポリシーを作成します。詳細については、「Amazon Verified Permissions 静的ポリシーの作成」を参照してください。

ID ソースの作成、スキーマの更新、ポリシーの作成が完了したら、 を使用して Verified Permissions `IsAuthorizedWithToken`に認可の決定を依頼します。詳細については、Amazon Verified Permissions API リファレンスガイドのIsAuthorizedWithToken」を参照してください。

AWS CLI

Amazon Cognito ユーザープール ID ソースを作成するには

CreateIdentitySource オペレーションを使用して ID ソースを作成できます。次の の例では、Amazon Cognito ユーザープールから認証された ID にアクセスできる ID ソースを作成します。

1. `create-identity-source` コマンドの `--configuration`パラメータで使用する Amazon Cognito ユーザープールの以下の詳細を含む`config.txt`ファイルを作成します。

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

2. 次のコマンドを実行して ID Amazon Cognito ソースを作成します。

```
$ 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. (オプション) ポリシーストアにスキーマがある場合、Cedar ポリシーのアイデンティティトークンまたはアクセストークンから抽出した属性を参照する前に、スキーマを更新して、ID ソースが作成するプリンシパルのタイプを Cedar に認識させる必要があります。スキーマへの追加には、Cedar ポリシーで参照したい属性を含める必要があります。 Amazon Cognito トークン属性を Cedar プリンシパル属性にマッピングする方法の詳細については、「」を参照してくださいAmazon Cognito トークンをスキーマにマッピングする。

> **Note**
>
> API リンクポリシーストアを作成するか、ポリシーストアの作成時に API Gateway と ID プロバイダーでセットアップを使用するとき、Verified Permissions はユーザープールにユーザー属性をクエリし、プリンシパルタイプにユーザープール属性が入力されるスキーマを作成します。

4. トークンからの情報を使用して認可を決定するポリシーを作成します。詳細については、「Amazon Verified Permissions 静的ポリシーの作成」を参照してください。

ID ソースの作成、スキーマの更新、ポリシーの作成が完了したら、 を使用して Verified Permissions `IsAuthorizedWithToken`に認可の決定を依頼します。詳細については、「Amazon Verified Permissions API リファレンスガイド」の「IsAuthorizedWithToken」を参照してください。

Verified Permissions で認証されたユーザーに Amazon Cognito アクセストークンと ID トークンを使用する方法の詳細については、「Amazon Amazon Cognitoデベロッパーガイド」の「Amazon Verified Permissions による認可」を参照してください。

Amazon Verified Permissions Amazon Cognito ID ソースの編集

ID ソースの一部のパラメータは、作成後に編集できます。ID ソースのタイプを変更することはできません。ID ソースを削除し、OIDC または OIDC Amazon Cognito に切り替える新しい ID ソースを作成する必要があります Amazon Cognito。ポリシーストアスキーマが ID ソース属性と一致する場合は、ID ソースに加えた変更を反映するようにスキーマを個別に更新する必要があることに注意してください。

AWS マネジメントコンソール

Amazon Cognito ID ソースを更新するには

1. Verified Permissions コンソールを開きます。ポリシーストアを選択します。
2. 左側にあるナビゲーションペインで、[ID ソース] を選択します。
3. 編集する ID ソースの ID を選択します。
4. [編集] を選択します。
5. Cognito ユーザープールの詳細で、 を選択し AWS リージョン 、ID ソースのユーザープール ID を入力します。
6. プリンシパルの詳細で、ID ソースのプリンシパルタイプを更新できます。接続された Amazon Cognito ユーザープールの ID は、選択したプリンシパルタイプにマッピングされます。
7. グループ設定で、ユーザープール`cognito:groups`クレームをマッピングする場合は Cognito グループを使用するを選択します。プリンシパルタイプの親であるエンティティタイプを選択します。

8. クライアントアプリケーションの検証で、クライアントアプリケーション IDsを検証するかどうかを選択します。

   - クライアントアプリケーション ID を検証するには、「クライアントアプリケーション ID が一致するトークンのみを受け入れる」を選択します。検証するクライアントアプリケーション ID ごとに [新しいクライアントアプリケーション ID を追加] を選択します。追加したクライアントアプリケーション ID を削除するには、クライアントアプリケーション ID の横にある [削除] を選択します。
   - クライアントアプリケーション ID を検証したくない場合は、「クライアントアプリケーション ID を検証しない」を選択します。

9. [変更を保存] をクリックします。
10. ID ソースのプリンシパルタイプを変更した場合は、更新されたプリンシパルタイプが正しく反映されるようにスキーマを更新する必要があります。

ID ソースを削除するには、ID ソースの横にあるラジオボタンを選択し、次に [ID ソースを削除] を選択します。テキストボックスにdeleteを入力し、[ID ソースを削除] を選択して ID ソースの削除を確定します。

AWS CLI

Amazon Cognito ID ソースを更新するには

UpdateIdentitySource オペレーションを使用して ID ソースを更新できます。次の の例では、指定された ID ソースを更新して、別の Amazon Cognito ユーザープールを使用します。

1. update-identity-source コマンドの --configurationパラメータで使用する Amazon Cognito ユーザープールの以下の詳細を含むconfig.txtファイルを作成します。

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

2. 次のコマンドを実行して、 Amazon Cognito ID ソースを更新します。

```
$ 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**
>
> ID ソースのプリンシパルタイプを変更する場合、更新されたプリンシパルタイプを正しく反映するようにスキーマを更新する必要があります。

Amazon Cognito トークンをスキーマにマッピングする

ID ソースをポリシーストアに追加し、プロバイダークレーム、またはトークンをポリシーストアスキーマにマッピングする場合があります。このプロセスを自動化するには、ガイド付きセットアップを使用して ID ソースでポリシーストアを作成するか、ポリシーストアの作成後にスキーマを手動で更新します。トークンをスキーマにマッピングしたら、それらを参照するポリシーを作成できます。

ユーザーガイドのこのセクションには、次の情報が含まれています。

- ポリシーストアスキーマに属性を自動的に入力できる場合
- Verified Permissions ポリシーで Amazon Cognito トークンクレームを使用する方法
- ID ソースのスキーマを手動で構築する方法

API リンクポリシーストアとガイド付きセットアップで作成された ID ソースを持つポリシーストアでは、ID (ID) トークン属性をスキーマに手動でマッピングする必要はありません。Verified Permissions にユーザープールの属性を指定し、ユーザー属性が入力されたスキーマを作成できます。ID トークン認可では、Verified Permissions はクレームをプリンシパルエンティティの属性にマッピングします。次の条件で Amazon Cognito 、トークンをスキーマに手動でマッピングする必要がある場合があります。

- サンプルから空のポリシーストアまたはポリシーストアを作成しました。
- アクセストークンの使用をロールベースのアクセスコントロール (RBAC) を超えて拡張したい。
- Verified Permissions REST API、 AWS SDK、または を使用してポリシーストアを作成します AWS CDK。

Verified Permissions ポリシーストアで ID ソース Amazon Cognito として を使用するには、スキーマにプロバイダー属性が必要です。スキーマは固定されており、プロバイダートークンが IsAuthorizedWithToken または BatchIsAuthorizedWithToken API リクエストで作成するエンティティに対応している必要があります。ID トークンのプロバイダー情報からスキーマを自動的に入力する方法でポリシーストアを作成した場合は、ポリシーを作成する準備が整います。ID ソースのスキーマなしでポリシーストアを作成する場合は、API リクエストを使用して作成されたエンティティに一致するプロバイダー属性をスキーマに追加する必要があります。その後、プロバイダートークンの属性を使用してポリシーを記述できます。

Verified Permissions で認証されたユーザーに Amazon Cognito ID とアクセストークンを使用する方法の詳細については、「Amazon Amazon Cognitoデベロッパーガイド」の「Amazon Verified Permissions による認可」を参照してください。

トピック

- ID トークンをスキーマにマッピングする
- アクセストークンをマッピングする
- Amazon Cognito コロン区切りクレームの代替表記
- スキーママッピングについて知っておくべきこと

ID トークンをスキーマにマッピングする

Verified Permissions は、ID トークンクレームをユーザーの名前とタイトル、グループメンバーシップ、連絡先情報などの属性として処理します。ID トークンは、属性ベースのアクセスコントロール (ABAC) 認可モデルで最も役立ちます。Verified Permissions でリクエストを行っているユーザーに基づいてリソースへのアクセスを分析する場合は、ID ソースの ID トークンを選択します。

Amazon Cognito ID トークンは、ほとんどの OIDC 依存パーティライブラリで機能します。追加のクレームで OIDC の機能を拡張します。アプリケーションは、Amazon Cognito ユーザープール認証 API オペレーション、またはユーザープールがホストする UI を使用してユーザーを認証できます。詳細については、「 Amazon Cognito デベロッパーガイド」の「API とエンドポイントの使用」を参照してください。

Amazon Cognito ID トークンでの便利なクレーム

cognito:username および *preferred_username*

ユーザーのユーザー名のバリエーション。

sub

ユーザーの一意のユーザー識別子 (UUID)

custom: プレフィックスが付いたクレーム

などのカスタムユーザープール属性のプレフィックスcustom:employmentStoreCode。

標準のクレーム

email や などの標準 OIDC クレームphone_number。詳細については、「エラーセット 2 を組み込んだ OpenID Connect Core 1.0 の標準クレーム」を参照してください。 OpenID

cognito:groups

ユーザーのグループメンバーシップ。ロールベースのアクセスコントロール (RBAC) に基づく認可モデルでは、このクレームはポリシーで評価できるロールを示しています。

一時的なクレーム

ユーザーのプロパティではないが、実行時にユーザープールのトークン生成前 Lambda トリガーによって追加されるクレーム。一時的なクレームは標準クレームに似ていますが、tenantや など、標準外ですdepartment。

: 区切り文字を持つ Amazon Cognito 属性を参照するポリシーでは、 形式の属性を参照しますprincipal["*cognito:username*"]。ロールクレームcognito:groupsはこのルールの例外です。Verified Permissions は、このクレームの内容をユーザーエンティティの親エンティティにマッピングします。

Amazon Cognito ユーザープールからの ID トークンの構造の詳細については、「 Amazon Cognito デベロッパーガイド」の「ID トークンの使用」を参照してください。

次の ID トークンの例には、4 種類の属性があります。これには、 Amazon Cognito特定のクレーム cognito:username、カスタムクレーム custom:employmentStoreCode、標準クレーム email、および一時クレーム が含まれます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"
}
```

Amazon Cognito ユーザープールで ID ソースを作成するときは、Verified Permissions が で認可リクエストで生成するプリンシパルエンティティのタイプを指定しますIsAuthorizedWithToken。その後、ポリシーはそのリクエストを評価する一環として、そのプリンシパルの属性をテストできます。スキーマは ID ソースのプリンシパルタイプと属性を定義し、Cedar ポリシーで参照できます。

ID トークングループクレームから派生するグループエンティティのタイプも指定します。認可リクエストでは、Verified Permissions はグループクレームの各メンバーをそのグループエンティティタイプにマッピングします。ポリシーでは、そのグループエンティティをプリンシパルとして参照できます。

以下の例は、サンプルの ID トークンの属性をVerified Permissions スキーマに反映する方法を示しています。スキーマの編集の詳細については、「ポリシーストアスキーマの編集」を参照してください。ID ソース構成でプリンシパルタイプ User が指定されている場合は、次の例のようなものを含めて、それらの属性を Cedar で使用できるようにすることができます。

```
"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
            }
         }
      }
   }
```

このスキーマに対して検証するポリシーの例については、「」を参照してくださいAmazon Cognito ID トークン属性を反映。

アクセストークンをマッピングする

Verified Permissions は、グループクレーム以外のアクセストークンクレームをアクションの属性またはコンテキスト属性として処理します。グループメンバーシップに加えて、IdP からのアクセストークンには API アクセスに関する情報が含まれる場合があります。アクセストークンは、ロールベースのアクセスコントロール (RBAC) を使用する認可モデルに役立ちます。グループメンバーシップ以外のアクセストークンクレームに依存する認可モデルでは、スキーマ設定に追加の労力が必要です。

Amazon Cognito アクセストークンには、認可に使用できるクレームがあります。

Amazon Cognito アクセストークンでの便利なクレーム

client_id

OIDC 証明書利用者のクライアントアプリケーションの ID。クライアント ID を使用すると、Verified Permissions は、承認リクエストがポリシーストアの許可されたクライアントからのものであることを確認できます。machine-to-machine (M2M) 認可では、リクエスト元のシステムはクライアントシークレットを使用してリクエストを承認し、認可の証拠としてクライアント ID とスコープを提供します。

scope

トークンのベアラーのアクセス許可を表す OAuth 2.0 スコープ。

cognito:groups

ユーザーのグループメンバーシップ。ロールベースのアクセスコントロール (RBAC) に基づく認可モデルでは、このクレームはポリシーで評価できるロールを示しています。

一時的なクレーム

アクセス許可ではないが、実行時にユーザープールのトークン生成前 Lambda トリガーによって追加されるクレーム。一時的なクレームは標準クレームに似ていますが、 tenantや など、標準外ですdepartment。アクセストークンをカスタマイズすると、 AWS 請求書にコストがかかります。

Amazon Cognito ユーザープールからのアクセストークンの構造の詳細については、「 Amazon Cognito デベロッパーガイド」の「アクセストークンの使用」を参照してください。

Amazon Cognito アクセストークンは、Verified Permissions に渡されるとコンテキストオブジェクトにマッピングされます。アクセストークンの属性は `context.token.attribute_name` を使用して参照できます。以下のアクセストークンの例には、`client_id` と `scope` のクレームの両方が含まれています。

```
{
    "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"
}
```

次の例は、サンプルアクセストークンの属性を Verified Permissions スキーマに反映する方法を示しています。スキーマの編集の詳細については、「ポリシーストアスキーマの編集」を参照してください。

```
{
   "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"
         }
```

```
      }
   }
}
```

このスキーマに対して検証するポリシーの例については、「」を参照してくださいAmazon Cognito アクセストークン属性を反映。

Amazon Cognito コロン区切りクレームの代替表記

Verified Permissions の起動時に、 `cognito:groups` や などの Amazon Cognito トークンクレームに推奨されるスキーマは、これらのコロン区切り文字列を階層区切り文字`.`として使用するように`custom:store`変換しました。この形式はドット表記と呼ばれます。たとえば、ポリシー`principal.cognito.groups`で への参照`cognito:groups`が になりました。この形式は引き続き使用できますが、ブラケット表記を使用してスキーマとポリシーを構築することをお勧めします。この形式では、ポリシー`principal["cognito:groups"]`で への参照`cognito:groups`がになります。Verified Permissions コンソールからユーザープール ID トークンの自動生成されたスキーマは、角括弧表記を使用します。

ID ソースの Amazon Cognito 手動で構築されたスキーマとポリシーでは、ドット表記を引き続き使用できます。他のタイプの OIDC IdP のスキーマ`:`またはポリシーでは、 または他の英数字以外の文字でドット表記を使用することはできません。

ドット表記のスキーマは、次の例に示すように、`:`キャラクターの各インスタンスを `cognito`または`custom`初期フレーズの子としてネストします。

```
"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
         }
      }
   }
}
```

このスキーマに対して検証し、ドット表記を使用するポリシーの例については、「」を参照してください[ドット表記を使用して属性を参照します](#)。

スキーママッピングについて知っておくべきこと

属性マッピングはトークンタイプによって異なります

アクセストークン認可では、Verified Permissions はクレームを[コンテキスト](#)にマッピングします。ID トークン認可では、Verified Permissions はクレームをプリンシパル属性にマッピングします。Verified Permissions コンソールで作成したポリシーストアの場合、空のポリシーストアとサンプルポリシーストアのみが ID ソースを持たず、ID トークン認可のためにユーザープール属性をスキーマに入力する必要があります。アクセストークン認可は、グループメンバークレームを含むロールベースのアクセスコントロール (RBAC) に基づいており、他のクレームをポリシーストアスキーマに自動的にマッピングしません。

ID ソース属性は必要ありません

Verified Permissions コンソールで ID ソースを作成すると、必須としてマークされた属性はありません。これにより、クレームの欠落が認可リクエストで検証エラーを引き起こすのを防ぐことができます。必要に応じて属性を に設定できますが、すべての認可リクエストに存在する必要があります。

RBAC はスキーマに属性を必要としません

ID ソースのスキーマは、ID ソースを追加するときに行うエンティティの関連付けによって異なります。ID ソースは、1 つのクレームをユーザーエンティティタイプにマッピングし、1 つのクレームをグループエンティティタイプにマッピングします。これらのエンティティマッピングは、アイデンティティソース設定の中核です。この最小限の情報を使用して、ロールベースのアクセスコントロール (RBAC) モデルで、特定のユーザーおよびユーザーが属する可能性のある特定のグループに対して認可アクションを実行するポリシーを作成できます。スキーマにトークンクレームを追加すると、ポリシーストアの認可範囲が拡張されます。ID トークンのユーザー属性には、属性ベースのアクセスコントロール (ABAC) 認可に貢献できるユーザーに関する情報があります。アクセストークンのコンテキスト属性には、OAuth 2.0 スコープなどの情報があり、プロバイダーからの追加のアクセスコントロール情報を提供できますが、追加のスキーマ変更が必要です。

Verified Permissions コンソールの API Gateway と ID プロバイダーのセットアップとガイド付きセットアップオプションは、ID トークンクレームをスキーマに割り当てます。アクセストークンクレームの場合はこの限りではありません。非グループアクセストークンクレームをスキーマに追加するには、JSON モードでスキーマを編集し、 commonTypes 属性を追加する必要があります。詳細については、「アクセストークンをマッピングする」を参照してください。

トークンタイプを選択する

ポリシーストアが ID ソースと連携する方法は、ID トークンを処理するかアクセストークンを処理するかという ID ソース設定の重要な決定によって異なります。 Amazon Cognito ID プロバイダーでは、API リンクポリシーストアを作成するときにトークンタイプを選択できます。API リンクポリシーストアを作成するときは、ID トークンまたはアクセストークンの認可を設定するかどうかを選択する必要があります。この情報は、Verified Permissions がポリシーストアに適用するスキーマ属性と、 API Gateway API の Lambda オーソライザーの構文に影響します。特に、Verified Permissions コンソールの属性への ID トークンクレームの自動マッピングを活用する場合は、ID ソースを作成する前に、処理するトークンタイプを早期に決定します。トークンタイプを変更するには、ポリシーとスキーマをリファクタリングするための多大な労力が必要です。以下のトピックでは、ポリシーストアでの ID トークンとアクセストークンの使用について説明します。

Cedar パーサーには一部の文字に角括弧が必要です

ポリシーは通常、 のような形式のスキーマ属性を参照しますprincipal.username。トークンクレーム名に表示される/可能性がある :、、 .などのほとんどの英数字以外の文字の場合、Verified Permissions は principal.cognito:usernameや などの条件値を解析できませんcontext.ip-address。代わりにcontext["ip-address"]、これらの条件を角括弧表記でそれぞれ principal["cognito:username"]または の形式でフォーマットする必要があります。アンダースコア文字_はクレーム名の有効な文字であり、この要件に対する唯一の英数字以外の例外です。

このタイプのプリンシパル属性のスキーマの部分的な例は、次のようになります。

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

このタイプのコンテキスト属性のスキーマの部分的な例は、次のようになります。

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

```
}
```

このスキーマに対して検証するポリシーの例については、「」を参照してください角括弧表記を使用してトークン属性を参照します。

のクライアントとオーディエンスの検証 Amazon Cognito

ID ソースをポリシーストアに追加すると、Verified Permissions には、ID トークンとアクセストークンが意図したとおりに使用されていることを確認する設定オプションがあります。この検証は、`IsAuthorizedWithToken`および `BatchIsAuthorizedWithToken` API リクエストの処理で行われます。この動作は、ID トークンとアクセストークン、および Amazon Cognito と OIDC ID ソースによって異なります。Amazon Cognito ユーザープールプロバイダーを使用すると、Verified Permissions は ID トークンとアクセストークンの両方でクライアント ID を検証できます。OIDC プロバイダーを使用すると、Verified Permissions は ID トークンのクライアント ID とアクセストークンの対象者を検証できます。

クライアント ID は、アプリケーションが使用する ID プロバイダーインスタンスに関連付けられた識別子です`1example23456789`。例: 。対象者は、 など、アクセストークンの目的の証明書利用者または送信先に関連付けられた URL パスです`https://mytoken.example.com`。アクセストークンを使用する場合、`aud`クレームは常に対象者に関連付けられます。

Amazon Cognito ID トークンには、アプリケーションクライアント ID を含む `aud`クレームがあります。アクセストークンには、アプリケーションクライアント ID も含まれる`client_id`クレームがあります。

ID ソースにクライアントアプリケーション検証に 1 つ以上の値を入力すると、Verified Permissions はこのアプリケーションクライアント IDs のリストを ID トークン`aud`クレームまたはアクセストークン`client_id`クレームと比較します。Verified Permissions は、 Amazon Cognito ID ソースの証明書利用者 URL を検証しません。

JWTs のクライアント側の認可

アプリケーションで JSON ウェブトークンを処理し、ポリシーストア ID ソースを使用せずにそのクレームを Verified Permissions に渡すことができます。JSON Web Token (JWT) からエンティティ属性を抽出し、Verified Permissions に解析できます。

この例では、JWT.1 を使用してアプリケーションから Verified Permissions を呼び出す方法を示します。

```
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] このコード例では、aws-jwt-verify ライブラリを使用して OIDC 互換 IdPs によって署名された JWT を検証しています。

OIDC ID ソースの使用

準拠の OpenID Connect (OIDC) IdP をポリシーストアの ID ソースとして設定することもできます。OIDC プロバイダーは Amazon Cognito ユーザープールに似ています。認証の積として JWTs を生成します。OIDC プロバイダーを追加するには、発行者 URL を指定する必要があります

新しい OIDC ID ソースには、次の情報が必要です。

- 発行者 URL。Verified Permissions は、この URL で`.well-known/openid-configuration`エンドポイントを検出できる必要があります。
- ワイルドカードを含まない CNAME レコード。たとえば、 `a.example.com` を にマッピングすることはできません`*.example.net`。逆に、 `*.example.com` を にマッピングすることはできませんa.example.net。
- 認可リクエストで使用するトークンタイプ。この場合、ID トークンを選択します。
- ID ソースに関連付けるユーザーエンティティタイプ。例: `MyCorp::User`。
- ID ソースに関連付けるグループエンティティタイプ。例: `MyCorp::UserGroup`。
- ID トークンの例、または ID トークン内のクレームの定義。
- ユーザーおよびグループエンティティ IDs に適用するプレフィックス。CLI と API では、このプレフィックスを選択できます。API Gateway を使用してセットアップし、ID プロバイダーまたはガイド付きセットアップオプションを使用して作成したポリシーストアでは、Verified Permissions は発行者名から を引いたプレフィックスを割り当てます。たとえばhttps://、 ですMyCorp::User::"auth.example.com|a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"。

API オペレーションを使用して OIDC ソースからのリクエストを承認する方法の詳細については、「」を参照してください認可に使用できる API オペレーション。

次の例は、経理部門の従業員の年末レポートへのアクセスを許可し、機密分類を持ち、衛星局にいないポリシーを作成する方法を示しています。Verified Permissions は、プリンシパルの ID トークンのクレームからこれらの属性を取得します。

プリンシパルでグループを参照するときは、ポリシーを正しく評価するために `in`演算子を使用する必要があります。

```
permit(
     principal in MyCorp::UserGroup::"MyOIDCProvider|Accounting",
     action,
     resource in MyCorp::Folder::"YearEnd2024"
```

```
) when {
     principal.jobClassification == "Confidential" &&
     !(principal.location like "SatelliteOffice*")
};
```

トピック

- Amazon Verified Permissions OIDC ID ソースの作成
- Amazon Verified Permissions OIDC ID ソースの編集
- OIDC トークンをスキーマにマッピングする
- OIDC プロバイダーのクライアントとオーディエンスの検証

Amazon Verified Permissions OIDC ID ソースの作成

次の手順では、ID ソースを既存のポリシーストアに追加します。

Verified Permissions コンソールで新しいポリシーストアを作成するときに、ID ソースを作成することもできます。このプロセスでは、ID ソーストークンのクレームをエンティティ属性に自動的にインポートできます。ガイド付きセットアップを選択するか、 API Gateway と ID プロバイダーオプションを使用してセットアップします。これらのオプションでは、初期ポリシーも作成されます。

> **Note**
>
> ID ソース は、ポリシーストアを作成するまでは左側のナビゲーションペインには表示されません。作成する ID ソースは、現在のポリシーストアに関連付けられます。

Verified Permissions API の create-identity-source AWS CLI または CreateIdentitySource を使用して ID ソースを作成するときに、プリンシパルエンティティタイプを除外できます。ただし、空のエンティティタイプは、エンティティタイプが の ID ソースを作成しますAWS::Cognito。このエンティティ名は、ポリシーストアスキーマと互換性がありません。 Amazon Cognito ID をポリシーストアスキーマと統合するには、プリンシパルエンティティタイプをサポートされているポリシーストアエンティティに設定する必要があります。

AWS マネジメントコンソール

OpenID Connect (OIDC) ID ソースを作成するには

1. Verified Permissions コンソールを開きます。ポリシーストアを選択します。

2. 左側にあるナビゲーションペインで、[ID ソース] を選択します。
3. [ID ソースを作成]を選択します。
4. 外部 OIDC プロバイダーを選択します。
5. 発行者 URL に、OIDC 発行者の URL を入力します。これは、認可サーバー、署名キー、および などのプロバイダーに関するその他の情報を提供するサービスエンドポイントですhttps://auth.example.com。発行者 URL は、 で OIDC 検出ドキュメントをホストする必要があります/.well-known/openid-configuration。
6. トークンタイプで、アプリケーションが承認のために送信する OIDC JWT のタイプを選択します。詳細については、「OIDC トークンをスキーマにマッピングする」を参照してください。
7. スキーマエンティティへのトークンクレームをマップで、ID ソースのユーザーエンティティとユーザークレームを選択します。ユーザーエンティティは、OIDC プロバイダーのユーザーを参照するポリシーストア内のエンティティです。ユーザークレームは、通常sub、評価対象のエンティティの一意の識別子を保持する ID またはアクセストークンからのクレームです。接続された OIDC IdP の ID は、選択したプリンシパルタイプにマッピングされます。
8. (オプション) スキーマエンティティへのトークンクレームのマップで、アイデンティティソースのグループエンティティとグループクレームを選択します。グループエンティティはユーザーエンティティの親です。グループクレームはこのエンティティにマッピングされます。グループクレームは、評価されるエンティティのユーザーグループ名の文字列、JSON、またはスペースで区切られた文字列を含む ID またはアクセストークンgroupsからのクレームで、通常は です。接続された OIDC IdP の ID は、選択したプリンシパルタイプにマッピングされます。
9. 検証 - オプションで、ポリシーストアが認可リクエストで受け入れるクライアント IDs またはオーディエンス URLs があれば入力します。
10. [ID ソースを作成]を選択します。
11. (オプション) ポリシーストアにスキーマがある場合、Cedar ポリシーのアイデンティティトークンまたはアクセストークンから抽出する属性を参照する前に、スキーマを更新して、ID ソースが作成するプリンシパルのタイプを Cedar に認識させる必要があります。スキーマへの追加には、Cedar ポリシーで参照したい属性を含める必要があります。OIDC トークン属性を Cedar プリンシパル属性にマッピングする方法の詳細については、「」を参照してくださいOIDC トークンをスキーマにマッピングする。
12. トークンからの情報を使用して認可を決定するポリシーを作成します。詳細については、「Amazon Verified Permissions 静的ポリシーの作成」を参照してください。

ID ソースの作成、スキーマの更新、ポリシーの作成が完了したら、 を使用して Verified Permissions `IsAuthorizedWithToken`に認可の決定を依頼します。詳細については、Amazon Verified Permissions API リファレンスガイドの「IsAuthorizedWithToken」を参照してください。

AWS CLI

OIDC ID ソースを作成するには

CreateIdentitySource オペレーションを使用して ID ソースを作成できます。次の例では、OIDC ID プロバイダー (IdP) から認証された ID にアクセスできる ID ソースを作成します。

1. `create-identity-source` コマンドの `--configuration`パラメータで使用する OIDC IdP の以下の詳細を含む`config.txt`ファイルを作成します。

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

2. 次のコマンドを実行して、OIDC ID ソースを作成します。

```
$ 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. (オプション) ポリシーストアにスキーマがある場合、Cedar ポリシーのアイデンティティトークンまたはアクセストークンから抽出する属性を参照する前に、スキーマを更新して、ID ソースが作成するプリンシパルのタイプを Cedar に認識させる必要があります。スキーマへの追加には、Cedar ポリシーで参照したい属性を含める必要があります。OIDC トークン属性を Cedar プリンシパル属性にマッピングする方法の詳細については、「」を参照してくださいOIDC トークンをスキーマにマッピングする。
4. トークンからの情報を使用して認可を決定するポリシーを作成します。詳細については、「Amazon Verified Permissions 静的ポリシーの作成」を参照してください。

ID ソースの作成、スキーマの更新、ポリシーの作成が完了したら、 を使用して Verified Permissions `IsAuthorizedWithToken`に認可の決定を依頼します。詳細については、Amazon Verified Permissions API リファレンスガイドのIsAuthorizedWithToken」を参照してください。

Amazon Verified Permissions OIDC ID ソースの編集

ID ソースの一部のパラメータは、作成後に編集できます。ID ソースのタイプを変更することはできません。ID ソースを削除し、OIDC または OIDC Amazon Cognito に切り替える新しい ID ソースを作成する必要があります Amazon Cognito。ポリシーストアスキーマが ID ソース属性と一致する場合は、ID ソースに加えた変更を反映するようにスキーマを個別に更新する必要があることに注意してください。

AWS マネジメントコンソール

OIDC ID ソースを更新するには

1. Verified Permissions コンソールを開きます。ポリシーストアを選択します。
2. 左側にあるナビゲーションペインで、[ID ソース] を選択します。
3. 編集する ID ソースの ID を選択します。
4. [編集] を選択します。
5. OIDC プロバイダーの詳細で、必要に応じて発行者 URL を変更します。
6. トークンクレームをスキーマ属性にマップで、必要に応じてユーザーとグループのクレームとポリシーストアエンティティタイプの関連付けを変更します。エンティティタイプを変更したら、ポリシーとスキーマ属性を更新して、新しいエンティティタイプに適用する必要があります。
7. 対象者の検証で、適用する対象者値を追加または削除します。

8. [Save changes] (変更の保存) をクリックします。

ID ソースを削除するには、ID ソースの横にあるラジオボタンを選択し、次に [ID ソースを削除] を選択します。テキストボックスにdeleteを入力し、[ID ソースを削除] を選択して ID ソースの削除を確定します。

AWS CLI

OIDC ID ソースを更新するには

UpdateIdentitySource オペレーションを使用して ID ソースを更新できます。次の の例では、指定された ID ソースを更新して、別の OIDC プロバイダーを使用します。

1. update-identity-source コマンドの --configurationパラメータで使用する OIDC IdP の以下の詳細を含むconfig.txtファイルを作成します。

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

2. 次のコマンドを実行して、OIDC ID ソースを更新します。

```
$ 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**
>
> ID ソースのプリンシパルタイプを変更する場合、更新されたプリンシパルタイプを正しく反映するようにスキーマを更新する必要があります。

OIDC トークンをスキーマにマッピングする

ID ソースをポリシーストアに追加し、プロバイダークレーム、またはトークンをポリシーストアスキーマにマッピングする場合があります。このプロセスを自動化するには、ガイド付きセットアップを使用して ID ソースでポリシーストアを作成するか、ポリシーストアの作成後にスキーマを手動で更新します。トークンをスキーマにマッピングしたら、それらを参照するポリシーを作成できます。

ユーザーガイドのこのセクションには、次の情報が含まれています。

- ポリシーストアスキーマに属性を自動的に入力できる場合
- ID ソースのスキーマを手動で構築する方法

API リンクポリシーストアとガイド付きセットアップで作成された ID ソースを持つポリシーストアでは、ID (ID) トークン属性をスキーマに手動でマッピングする必要はありません。Verified Permissions にユーザープールの属性を指定し、ユーザー属性が入力されたスキーマを作成できます。ID トークン認可では、Verified Permissions はクレームをプリンシパルエンティティの属性にマッピングします。

Verified Permissions ポリシーストアで OIDC ID プロバイダー (IdP) を ID ソースとして使用するには、スキーマにプロバイダー属性が必要です。スキーマは固定されており、プロバイダートークンが IsAuthorizedWithToken または BatchIsAuthorizedWithToken API リクエストで作成するエンティティに対応している必要があります。ID トークンのプロバイダー情報からスキーマを自動的に入力する方法でポリシーストアを作成した場合は、ポリシーを作成する準備が整います。ID ソースのスキーマなしでポリシーストアを作成する場合は、API リクエストを使用して作成されたエンティティに一致するプロバイダー属性をスキーマに追加する必要があります。その後、プロバイダートークンの属性を使用してポリシーを記述できます。

トピック

ID トークンをスキーマにマッピングする

Verified Permissions は、ID トークンクレームをユーザーの名前とタイトル、グループメンバーシップ、連絡先情報などの属性として処理します。ID トークンは、属性ベースのアクセスコントロール (ABAC) 認可モデルで最も役立ちます。Verified Permissions でリクエストを行っているユーザーに基づいてリソースへのアクセスを分析する場合は、ID ソースの ID トークンを選択します。

OIDC プロバイダーからの ID トークンの操作は、 Amazon Cognito ID トークンの操作とほぼ同じです。違いはクレームにあります。IdP には、標準の OIDC 属性があるか、カスタムスキーマがある場合があります。Verified Permissions コンソールで新しいポリシーストアを作成するときに、ID トークンの例を使用して OIDC ID ソースを追加するか、トークンクレームをユーザー属性に手動でマッピングできます。Verified Permissions は IdP の属性スキーマを認識しないため、この情報を指定する必要があります。

詳細については、「Verified Permissions ポリシーストアの作成」を参照してください。

以下は、OIDC ID ソースを持つポリシーストアのスキーマの例です。

```
"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"
            }
         }
      }
}
```

このスキーマに対して検証するポリシーの例については、「」を参照してくださいOIDC ID トークン属性を反映。

アクセストークンをマッピングする

Verified Permissions は、グループクレーム以外のアクセストークンクレームをアクションの属性またはコンテキスト属性として処理します。グループメンバーシップに加えて、IdP からのアクセストークンには API アクセスに関する情報が含まれる場合があります。アクセストークンは、ロールベースのアクセスコントロール (RBAC) を使用する認可モデルに役立ちます。グループメンバーシップ以外のアクセストークンクレームに依存する認可モデルでは、スキーマ設定に追加の労力が必要です。

外部 OIDC プロバイダーからのほとんどのアクセストークンは、 Amazon Cognito アクセストークンと密接に一致します。OIDC アクセストークンは、Verified Permissions に渡されるとコンテキストオブジェクトにマッピングされます。アクセストークンの属性は `context.token.`*`attribute_name`* を使用して参照できます。次の OIDC アクセストークンの例には、基本クレームの例が含まれています。

```
{
    "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"
}
```

次の例は、サンプルアクセストークンの属性を Verified Permissions スキーマに反映する方法を示しています。スキーマの編集の詳細については、「ポリシーストアスキーマの編集」を参照してください。

```
{
   "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"
         }
```

```
        }
    }
}
```

このスキーマに対して検証するポリシーの例については、「」を参照してくださいOIDC アクセストークン属性を反映。

スキーママッピングについて知っておくべきこと

属性マッピングはトークンタイプによって異なります

アクセストークン認可では、Verified Permissions はクレームをコンテキストにマッピングします。ID トークン認可では、Verified Permissions はクレームをプリンシパル属性にマッピングします。Verified Permissions コンソールで作成したポリシーストアの場合、空のポリシーストアとサンプルポリシーストアのみが ID ソースを持たず、ID トークン認可のためにユーザープール属性をスキーマに入力する必要があります。アクセストークン認可は、グループメンバークレームを含むロールベースのアクセスコントロール (RBAC) に基づいており、他のクレームをポリシーストアスキーマに自動的にマッピングしません。

ID ソース属性は必要ありません

Verified Permissions コンソールで ID ソースを作成すると、必須としてマークされた属性はありません。これにより、クレームの欠落が認可リクエストで検証エラーを引き起こすのを防ぐことができます。必要に応じて属性を に設定できますが、すべての認可リクエストに存在する必要があります。

RBAC はスキーマに属性を必要としません

ID ソースのスキーマは、ID ソースを追加するときに作成するエンティティの関連付けによって異なります。ID ソースは、1 つのクレームをユーザーエンティティタイプにマッピングし、1 つのクレームをグループエンティティタイプにマッピングします。これらのエンティティマッピングは、アイデンティティソース設定の中核です。この最小限の情報を使用して、ロールベースのアクセスコントロール (RBAC) モデルで、特定のユーザーおよびユーザーがメンバーである可能性のある特定のグループに対して認可アクションを実行するポリシーを作成できます。スキーマにトークンクレームを追加すると、ポリシーストアの承認範囲が拡張されます。ID トークンのユーザー属性には、属性ベースのアクセスコントロール (ABAC) 認可に貢献できるユーザーに関する情報があります。アクセストークンのコンテキスト属性には、OAuth 2.0 スコープなどの情報があり、プロバイダーからの追加のアクセスコントロール情報を提供できますが、追加のスキーマ変更が必要です。

Verified Permissions コンソールの API Gateway と ID プロバイダーのセットアップとガイド付きセットアップオプションは、ID トークンクレームをスキーマに割り当てます。これは、アクセストークンクレームには当てはまりません。非グループアクセストークンクレームをスキーマに追加す

るには、JSON モードでスキーマを編集し、 commonTypes 属性を追加する必要があります。詳細については、「アクセストークンをマッピングする」を参照してください。

OIDC グループのクレームが複数の形式をサポート

OIDC プロバイダーを追加するときに、ポリシーストアのユーザーのグループメンバーシップにマッピングする ID トークンまたはアクセストークンのグループクレームの名前を選択できます。検証されたアクセス許可は、次の形式でグループのクレームを認識します。

1. スペースのない文字列: `"groups": "MyGroup"`
2. スペース区切りリスト: `"groups": "MyGroup1 MyGroup2 MyGroup3"`。各文字列はグループです。
3. JSON (カンマ区切り) リスト: `"groups": ["MyGroup1", "MyGroup2", "MyGroup3"]`

> **Note**
>
> Verified Permissions は、スペース区切りグループクレームの各文字列を個別のグループとして解釈します。グループ名をスペース文字で 1 つのグループとして解釈するには、クレーム内のスペースを置き換えるか削除します。たとえば、 という名前のグループを `My Group`としてフォーマットします`MyGroup`。

トークンタイプを選択する

ポリシーストアが ID ソースと連携する方法は、ID トークンを処理するかアクセストークンを処理するかという ID ソース設定の重要な決定によって異なります。OIDC プロバイダーでは、ID ソースを追加するときにトークンタイプを選択する必要があります。ID トークンまたはアクセストークンを選択できます。選択したトークンタイプは、ポリシーストアで処理されないトークンタイプを除外します。特に、Verified Permissions コンソールの属性への ID トークンクレームの自動マッピングのメリットを享受したい場合は、ID ソースを作成する前に、処理するトークンタイプを早期に決定してください。トークンタイプを変更するには、ポリシーとスキーマをリファクタリングするための多大な労力が必要です。以下のトピックでは、ポリシーストアでの ID トークンとアクセストークンの使用について説明します。

Cedar パーサーには一部の文字に角括弧が必要です

ポリシーは通常、 のような形式のスキーマ属性を参照します`principal.username`。トークンクレーム名に表示される/可能性がある `:`、、 `.`などのほとんどの英数字以外の文字の場

合、Verified Permissions は principal.cognito:usernameや などの条件値を解析できませんcontext.ip-address。代わりにcontext["ip-address"]、これらの条件を括弧表記でそれぞれ principal["cognito:username"]または の形式でフォーマットする必要があります。アンダースコア文字_はクレーム名の有効な文字であり、この要件に対する唯一の英数字以外の例外です。

このタイプのプリンシパル属性のスキーマの部分的な例は、次のようになります。

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

このタイプのコンテキスト属性のスキーマの一部の例は次のようになります。

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

```
            }
    }
    },
       "principalTypes": [
          "User"
       ]
    }
}
```

このスキーマに対して検証するポリシーの例については、「」を参照してください角括弧表記を使用してトークン属性を参照します。

OIDC プロバイダーのクライアントとオーディエンスの検証

ID ソースをポリシーストアに追加すると、Verified Permissions には、ID トークンとアクセストークンが意図したとおりに使用されていることを確認する設定オプションがあります。この検証は、`IsAuthorizedWithToken`および `BatchIsAuthorizedWithToken` API リクエストの処理で行われます。この動作は、ID トークンとアクセストークン、および Amazon Cognito と OIDC ID ソースによって異なります。Amazon Cognito ユーザープールプロバイダーを使用すると、Verified Permissions は ID トークンとアクセストークンの両方でクライアント ID を検証できます。OIDC プロバイダーを使用すると、Verified Permissions は ID トークンのクライアント ID とアクセストークンの対象者を検証できます。

クライアント ID は、アプリケーションが使用する ID プロバイダーインスタンスに関連付けられた識別子です`1example23456789`。例: 。対象者は、 など、アクセストークンの目的の証明書利用者または送信先に関連付けられた URL パスです`https://mytoken.example.com`。アクセストークンを使用する場合、`aud`クレームは常に対象者に関連付けられます。

OIDC ID トークンには、 などのクライアント IDs を含む`aud`クレームがあります`1example23456789`。

OIDC Access トークンには、 などのトークンのオーディエンス URL を含む`aud`クレームと`https://myapplication.example.com`、 などのクライアント IDs を含む`client_id`クレームがあります`1example23456789`。

ポリシーストアを設定するときは、ポリシーストアがトークンの対象者を検証するために使用する対象者検証の値を 1 つ以上入力します。

- ID トークン – Verified Permissions は、`aud`クレーム内のクライアント ID の少なくとも 1 つのメンバーが対象者検証値と一致することを確認して、クライアント IDs を検証します。

- アクセストークン – Verified Permissions は、audクレームの URL が対象者検証値と一致することを確認して対象者を検証します。aud クレームが存在しない場合、対象者は cidまたは client_idクレームを使用して検証できます。ID プロバイダーに正しい対象者のクレームと形式を確認してください。

JWTs のクライアント側の認可

アプリケーションで JSON ウェブトークンを処理し、ポリシーストア ID ソースを使用せずにそのクレームを Verified Permissions に渡すことができます。JSON Web Token (JWT) からエンティティ属性を抽出し、Verified Permissions に解析できます。

この例では、JWT.1 を使用してアプリケーションから Verified Permissions を呼び出す方法を示します。

```
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] このコード例では、aws-jwt-verify ライブラリを使用して OIDC 互換 IdPs によって署名された JWT を検証しています。

Amazon Verified Permissions の統合

Amazon Verified Permissions の統合は、コードを最小限に抑え、フレームワーク固有のベストプラクティスに従いながら、アプリケーションにきめ細かな認可を実装するのに役立ちます。これらの統合により、アプリケーションを Verified Permissions にシームレスに接続するミドルウェアコンポーネントとユーティリティが提供されます。

統合を使用すると、次のことができます。

- 数分で認可を実装する
- フレームワーク固有のパターンと規則に従う
- メンテナンスオーバーヘッドの削減
- 潜在的なセキュリティ実装エラーを最小限に抑える
- 認可コードではなくビジネスロジックに焦点を当てる

アプリケーションに追加すると、統合は次の操作を行います。

1. フレームワーク固有のミドルウェアを使用して受信リクエストをインターセプトする
2. リクエストから関連する認可コンテキストを抽出する
3. Verified Permissions を使用して認可の決定を決定する
4. 認可結果に基づいてアクセスコントロールを適用する

Verified Permissions は現在、次のフレームワークをサポートしています。

- Node.js アプリケーションの Express.js

Express と Amazon Verified Permissions の統合

Verified Permissions Express 統合は、Express.js アプリケーションに認可を実装するためのミドルウェアベースのアプローチを提供します。この統合により、既存のルートハンドラーを変更することなく、きめ細かな認可ポリシーを使用して API エンドポイントを保護できます。統合は、リクエストを傍受し、定義されたポリシーに照らして評価し、承認されたユーザーのみが保護されたリソースにアクセスできるようにすることで、認可チェックを自動的に処理します。

このトピックでは、ポリシーストアの作成から認可ミドルウェアの実装とテストまで、Express 統合の設定について説明します。これらのステップに従うことで、最小限のコード変更で Express アプリケーションに堅牢な認可コントロールを追加できます。

このトピックでは、次のGitHubリポジトリを参照します。

- cedar-policy/authorization-for-expressjs - Express.js の Cedar 認可ミドルウェア
- verifiedpermissions/authorization-clients-js - JavaScript 用の Verified Permissions 認可クライアント
- verifiedpermissions/examples/express-petstore - Express.js ミドルウェアを使用した実装例

前提条件

Express 統合を実装する前に、以下を確認してください。

- Verified Permissions にアクセスできるAWS アカウント
- Node.js と npm がインストールされている
- Express.js アプリケーション
- OpenID Connect (OIDC) ID プロバイダー (などAmazon Cognito)
- AWS CLI 適切なアクセス許可で設定されている

統合のセットアップ

ステップ 1: ポリシーストアを作成する

以下を使用してポリシーストアを作成します AWS CLI。

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

> **Note**
> レスポンスで返されたポリシーストア ID を保存して、以降のステップで使用します。

ステップ 2: 依存関係をインストールする

Express アプリケーションに必要なパッケージをインストールします。

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

認可の設定

ステップ 1: Cedar スキーマを生成してアップロードする

スキーマは、アプリケーションのエンティティタイプやユーザーが実行できるアクションなど、アプリケーションの認可モデルを定義します。スキーマの名前空間を定義することをお勧めします。この例では、YourNamespace を使用します。Verified Permissions ポリシーストアにスキーマをアタッチし、ポリシーが追加または変更されると、サービスは自動的にポリシーをスキーマに対して検証します。

@cedar-policy/authorization-for-expressjs パッケージは、アプリケーションの OpenAPI 仕様を分析し、Cedar スキーマを生成できます。具体的には、パスオブジェクトが仕様で必要です。

OpenAPI 仕様がない場合は、 express-openapi-generator パッケージの簡単な手順に従って OpenAPI 仕様を生成できます。

OpenAPI 仕様からスキーマを生成します。

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

次に、 で使用する Cedar スキーマをフォーマットします AWS CLI。必要な特定の形式の詳細については、「」を参照してくださいポリシーストアスキーマ。スキーマのフォーマットにヘルプが必要な場合は、Verifiedpermissions/examples GitHub リポジトリprepare-cedar-schema.shにというスクリプトがあります。以下は、 v2.cedarschema.forAVP.jsonファイルに Verified Permissions 形式のスキーマを出力するスクリプトの呼び出し例です。

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

フォーマットされたスキーマをポリシーストアにアップロードし、 をポリシーストア ID policy-store-id に置き換えます。

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

ステップ 2: 認可ポリシーを作成する

ポリシーが設定されていない場合、Cedar はすべての認可リクエストを拒否します。Express フレームワーク統合は、以前に生成されたスキーマに基づいてサンプルポリシーを生成することで、このプロセスをブートストラップするのに役立ちます。

本番アプリケーションでこの統合を使用する場合は、Infrastructure as a Code (IaaC) ツールを使用して新しいポリシーを作成することをお勧めします。詳細については、「の使用 AWS CloudFormation」を参照してください。

Cedar ポリシーのサンプルを生成します。

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

これにより、 /policies ディレクトリにサンプルポリシーが生成されます。その後、ユースケースに基づいてこれらのポリシーをカスタマイズできます。例えば、次のようになります。

```
// 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
);
```

で使用するポリシーをフォーマットします AWS CLI。必要な形式の詳細については、 AWS CLI リファレンスの「create-policy」を参照してください。ポリシーの書式設定にヘルプが必要な場合は、確認済みアクセス許可/例GitHubリポジトリconvert_cedar_policies.shに というスクリプトがあります。以下は、そのスクリプトへの呼び出しです。

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

フォーマットされたポリシーを Verified Permissions にアップロードし、 をポリシーファイルのパスと名前policy_1.jsonに、 をポリシーストア ID policy-store-id に置き換えます。

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

ステップ 3: ID プロバイダーを接続する

デフォルトでは、Verified Permissions オーソライザーミドルウェアは API リクエストの認可ヘッダー内で提供される JSON ウェブトークン (JWT) を読み取り、ユーザー情報を取得します。Verified Permissions は、認可ポリシーの評価に加えて、トークンを検証できます。

userPoolArn および で、次のような名前identity-source-configuration.txtの ID ソース設定ファイルを作成しますclientId。

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

次の AWS CLI コマンドを実行して ID ソースを作成し、 をポリシーストア ID policy-store-id に置き換えます。

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

認可ミドルウェアの実装

Express アプリケーションを更新して、認可ミドルウェアを含めます。この例では ID トークンを使用していますが、アクセストークンを使用することもできます。詳細については、 のauthorization-for-expressjs」を参照してください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);
```

統合のテスト

異なるユーザートークンを使用して API エンドポイントにリクエストを行うことで、認可実装をテストできます。認可ミドルウェアは、定義されたポリシーに対して各リクエストを自動的に評価します。

たとえば、異なるアクセス許可を持つ異なるユーザーグループを設定した場合:

- 管理者: すべてのリソースと管理機能へのフルアクセス
- 従業員: リソースを表示、作成、更新できる
- 顧客: リソースのみを表示可能

異なるユーザーでサインインし、さまざまなオペレーションを試みることで、アクセス許可ポリシーが期待どおりに動作していることを検証できます。Express アプリケーションのターミナルには、認可の決定に関する追加の詳細を示すログ出力が表示されます。

トラブルシューティング

認可に失敗した場合は、以下を試してください。

- ポリシーストア ID が正しいことを確認する
- ID ソースが正しく設定されていることを確認します。
- ポリシーの形式が正しいことを確認する
- JWT トークンが有効であることを確認する

次の手順

基本的な統合を実装したら、次の点を考慮してください。

- 特定の認可シナリオ用のカスタムマッパーの実装
- 認可決定のモニタリングとログ記録の設定
- さまざまなユーザーロールの追加ポリシーの作成

Amazon Verified Permissions での認可の実装

ポリシーストア、ポリシー、テンプレート、スキーマ、認可モデルを構築したら、Amazon Verified Permissions を使用してリクエストの承認を開始する準備が整います。Verified Permissions 認可を実装するには、 の認可ポリシーの設定 AWS をアプリケーションの統合と組み合わせる必要があります。Verified Permissions をアプリケーションと統合するには、 AWS SDK を追加し、Verified Permissions API を呼び出し、ポリシーストアに対して認可決定を生成するメソッドを実装します。

Verified Permissions による認可は、アプリケーションの UX アクセス許可と API アクセス許可に役立ちます。

UX アクセス許可

アプリケーション UX へのユーザーアクセスを制御します。ユーザーがアクセスする必要がある正確なフォーム、ボタン、グラフィック、その他のリソースのみを表示することを許可できます。たとえば、ユーザーがサインインすると、「資金の移管」ボタンがアカウントに表示されるかどうかを判断できます。ユーザーが実行できるアクションを制御することもできます。たとえば、同じバンキングアプリで、ユーザーがトランザクションのカテゴリの変更を許可されているかどうかを判断できます。

API アクセス許可

データへのユーザーアクセスを制御します。アプリケーションは多くの場合、分散システムの一部であり、外部 APIs。Verified Permissions が「送金資金」ボタンの表示を許可した銀行アプリの例では、ユーザーが送金を開始するときに、より複雑な認可決定を行う必要があります。Verified Permissions は、対象となる移管先アカウントを一覧表示する API リクエストを承認し、次に移管を他のアカウントにプッシュするリクエストを承認できます。

このコンテンツを説明する例は、サンプルポリシーストアからのものです。これを行うには、テスト環境に DigitalPetStore サンプルポリシーストアを作成します。

バッチ認可を使用して UX アクセス許可を実装するエンドツーエンドのサンプルアプリケーションについては、 AWS セキュリティブログの「Use Amazon Verified Permissions for fine-grained authorization at scale」を参照してください。

トピック

- 認可に使用できる API オペレーション
- 認可モデルのテスト

- 認可モデルとアプリケーションの統合

認可に使用できる API オペレーション

Verified Permissions API には、次の認可オペレーションがあります。

IsAuthorized

`IsAuthorized` API オペレーションは、Verified Permissions を使用した認可リクエストへのエントリポイントです。プリンシパル、アクション、リソース、コンテキスト、エンティティの要素を送信する必要があります。Verified Permissions は、リクエスト内のエンティティに適用されるリクエストされたポリシーストア内のすべてのポリシーに対してリクエストを評価します。

IsAuthorizedWithToken

`IsAuthorizedWithToken` オペレーションは、JSON ウェブトークン (JWTs) のユーザーデータから認可リクエストを生成します。Verified Permissions は、ポリシーストアの ID ソース Amazon Cognito として などの OIDC プロバイダーと直接連携します。Verified Permissions は、ユーザーの ID またはアクセストークンのクレームからリクエストのすべての属性をプリンシパルに入力します。ID ソースのユーザー属性またはグループメンバーシップからアクションとリソースを承認できます。

`IsAuthorizedWithToken` リクエストにグループまたはユーザープリンシパルタイプに関する情報を含めることはできません。指定した JWT にすべてのプリンシパルデータを入力する必要があります。

BatchIsAuthorized

`BatchIsAuthorized` オペレーションは、1 つの API リクエストで 1 つのプリンシパルまたはリソースに対して複数の認可決定を処理します。このオペレーションは、クォータの使用を最小限に抑え、最大 30 個の複雑なネストされたアクションごとに認可決定を返す 1 つのバッチオペレーションにリクエストをグループ化します。単一のリソースのバッチ認可を使用すると、ユーザーがリソースに対して実行できるアクションをフィルタリングできます。単一のプリンシパルのバッチ認可を使用すると、ユーザーがアクションを実行できるリソースをフィルタリングできます。

BatchIsAuthorizedWithToken

`BatchIsAuthorizedWithToken` オペレーションは、1 つの API リクエストで 1 つのプリンシパルに対して複数の承認決定を処理します。プリンシパルは、ポリシーストア ID ソースによっ

て ID またはアクセストークンで提供されます。このオペレーションは、クォータの使用を最小限に抑え、アクションとリソースに対する最大 30 個のリクエストごとに認可決定を返す単一のバッチオペレーションにリクエストをグループ化します。ポリシーでは、属性またはユーザーディレクトリのグループメンバーシップからのアクセスを許可できます。

と同様にIsAuthorizedWithToken、グループまたはユーザープリンシパルタイプに関する情報をBatchIsAuthorizedWithTokenリクエストに含めることはできません。指定した JWT にすべてのプリンシパルデータを入力する必要があります。

認可モデルのテスト

アプリケーションをデプロイする際の Amazon Verified Permissions 認可決定の影響を理解するために、 Amazon Verified Permissions テストベンチの使用および Verified Permissions への HTTPS REST API リクエストを使用してポリシーを開発する際にポリシーを評価できます。テストベンチは、ポリシーストア内の認可リクエストとレスポンスを評価する AWS マネジメントコンソール ための のツールです。

Verified Permissions REST API は、概念的な理解からアプリケーション設計に移行する際の開発の次のステップです。Verified Permissions API は、IsAuthorized、IsAuthorizedWithToken、BatchIsAuthorized を使用した認可リクエストを、リージョンサービスエンドポイントへの署名付き AWS API リクエストとして受け入れます。認可モデルをテストするには、任意の API クライアントでリクエストを生成し、ポリシーが想定どおりに認可決定を返していることを確認できます。

例えば、次の手順を使用してサンプルポリシーストアIsAuthorizedでテストできます。

Test bench

1. Verified Permissions コンソールで Verified Permissions コンソールを開きます。DigitalPetStore という名前のサンプルポリシーストアからポリシーストアを作成します。
2. 新しいポリシーストアでテストベンチを選択します。
3. 「Verified Permissions API リファレンス」のIsAuthorized」からテストベンチリクエストを入力します。以下の詳細では、例 4 の条件を、 DigitalPetStore サンプルを参照するようにレプリケートします。

    a. Alice をプリンシパルとして設定します。アクションを実行するプリンシパル で、DigitalPetStore::Userを選択して と入力しますAlice。

   b. Alice のロールを顧客として設定します。親を追加を選択し、 を選択してDigitalPetStore::Role、「顧客」と入力します。

   c. リソースを順序「1234」に設定します。プリンシパルが動作しているリソース で、DigitalPetStore::Orderを選択して と入力します1234。

   d. DigitalPetStore::Order リソースには owner 属性が必要です。Alice を注文の所有者として設定します。選択DigitalPetStore::Userして入力する Alice

   e. Alice が注文の表示をリクエストしました。プリンシパルが実行するアクションで、 を選択しますDigitalPetStore::Action::"GetOrder"。

4. 認可リクエストの実行 を選択します。変更されていないポリシーストアでは、このリクエストは ALLOW 決定になります。決定を返した「満たす」ポリシーに注意してください。
5. 左側のナビゲーションバーから [ポリシー] を選択します。顧客ロール - 注文の取得の説明とともに静的ポリシーを確認します。
6. プリンシパルが顧客ロールにあり、リソースの所有者であったため、Verified Permissions がリクエストを許可したことを確認します。

REST API

1. Verified Permissions コンソールで Verified Permissions コンソールを開きます。DigitalPetStore という名前のサンプルポリシーストアからポリシーストアを作成します。
2. 新しいポリシーストアのポリシーストア ID を書き留めます。
3. 「Verified Permissions API リファレンス」のIsAuthorized」から、 DigitalPetStore サンプルを参照する例 4 のリクエスト本文をコピーします。
4. API クライアントを開き、ポリシーストアのリージョンサービスエンドポイントへのリクエストを作成します。例に示すように、ヘッダーを入力します。
5. サンプルリクエスト本文に貼り付け、 の値を前にメモしたポリシーストア ID policyStoreId に変更します。
6. リクエストを送信し、結果を確認します。デフォルトの DigitalPetStore ポリシーストアでは、このリクエストは ALLOW 決定を返します。

テスト環境のポリシー、スキーマ、リクエストを変更して、結果を変更し、より複雑な決定を行うことができます。

1. Verified Permissions から決定を変更する方法でリクエストを変更します。例えば、Alice のロールを に変更Employeeするか、順序 1234 の owner 属性を に変更しますBob。
2. 承認の決定に影響する方法でポリシーを変更します。例えば、顧客ロール - 注文を取得するという説明でポリシーを変更して、 が の所有者であるUser必要があるという条件を削除ResourceしBob、 が注文を表示するようにリクエストを変更します。
3. スキーマを変更して、ポリシーがより複雑な決定を行うことを許可します。Alice が新しい要件を満たすことができるようにリクエストエンティティを更新します。例えば、スキーマを編集して、 Userが ActiveUsersまたは のメンバーになるようにしますInactiveUsers。アクティブなユーザーのみが自分の注文を表示できるようにポリシーを更新します。Alice がアクティブまたは非アクティブなユーザーになるようにリクエストエンティティを更新します。

認可モデルとアプリケーションの統合

アプリケーションに Amazon Verified Permissions を実装するには、アプリケーションに適用するポリシーとスキーマを定義する必要があります。認可モデルを準備してテストしたら、次のステップは、適用時点から API リクエストの生成を開始することです。これを行うには、ユーザーデータを収集して認可リクエストに入力するようにアプリケーションロジックを設定する必要があります。

アプリが Verified Permissions を使用してリクエストを承認する方法

1. 現在のユーザーに関する情報を収集します。通常、ユーザーの詳細は、JWT やウェブセッション Cookie など、認証されたセッションの詳細で提供されます。このユーザーデータは、ポリシーストアにリンクされた Amazon Cognito ID ソースまたは別の OpenID Connect (OIDC) プロバイダーから送信される場合があります。
2. ユーザーがアクセスするリソースに関する情報を収集します。通常、ユーザーが新しいアセットをロードするためにアプリを必要とする選択を行うと、アプリケーションはリソースに関する情報を受け取ります。
3. ユーザーが実行するアクションを決定します。
4. ユーザーの試行されたオペレーションのプリンシパル、アクション、リソース、エンティティを使用して、Verified Permissions に認可リクエストを生成します。Verified Permissions は、ポリシーストアのポリシーに対してリクエストを評価し、認可決定を返します。
5. アプリケーションは Verified Permissions から許可または拒否のレスポンスを読み取り、ユーザーのリクエストに決定を適用します。

Verified Permissions API オペレーションは AWS SDKs。Verified Permissions をアプリケーションに含めるには、選択した言語の AWS SDK をアプリケーションパッケージに統合します。

詳細と AWS SDKs「Tools for Amazon Web Services」を参照してください。

以下は、さまざまな AWS SDKs。

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

次の AWS SDK for JavaScript の例は、Amazon Verified Permissions と Amazon Cognito を使用したきめ細かな認可の簡素化からIsAuthorized派生しています。

```
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"
                }
            }
```

```
          }
        ]
    }
});
```

その他の開発者リソース

- Amazon Verified Permissions ワークショップ
- Amazon Verified Permissions - リソース
- Amazon Verified Permissions を使用して ASP.NET Core アプリのカスタム認可ポリシープロバイダーを実装する
- Amazon Verified Permissions を使用してビジネスアプリケーションのエンタイトルメントサービスを構築する
- Amazon Verified Permissions と Amazon Cognito を使用してきめ細かな認可を簡素化する

Amazon Verified Permissions のセキュリティ

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

セキュリティは、 AWS とお客様の間で共有される責任です。責任共有モデルでは、これをクラウドのセキュリティおよびクラウド内のセキュリティと説明しています。

- クラウドのセキュリティ – AWS は、 で AWS サービスを実行するインフラストラクチャを保護する責任があります AWS クラウド。 AWS また、 は、お客様が安全に使用できるサービスも提供します。AWS コンプライアンスプログラムコンプライアンスプログラムの一環として、サードパーティーの監査者は定期的にセキュリティの有効性をテストおよび検証。Amazon Verified Permissions に適用されるコンプライアンスプログラムの詳細については、「コンプライアンスプログラムAWS による対象範囲内のサービスコンプライアンスプログラム」を参照してください。
- クラウド内のセキュリティ – お客様の責任は、使用する AWS サービスによって決まります。また、ユーザーは、データの機密性、会社の要件、適用される法律や規制など、その他の要因についても責任を負います。

このドキュメントは、Verified Permissions 使用時における責任共有モデルの適用法を理解するのに役立ちます。以下のトピックでは、セキュリティとコンプライアンスの目的を満たすように Verified Permissions を設定する方法について説明します。また、Verified Permissions リソースのモニタリングや保護に役立つ他の AWS サービスの使用方法についても説明します。

トピック

- Amazon Verified Permission でのデータ保護
- Amazon Verified Permissions のためのアイデンティティとアクセス管理
- Amazon Verified Asmission のコンプライアンス検証
- Amazon Verified Permissions

Amazon Verified Permission でのデータ保護

責任 AWS 共有モデル、Amazon Verified Permissions でのデータ保護に適用されます。このモデルで説明されているように、「 AWS 」は、「 AWS クラウド」のすべてを実行するグローバルインフラストラクチャを保護する責任があります。ユーザーは、このインフラストラクチャでホストさ

れるコンテンツに対する管理を維持する責任があります。このコンテンツには、 AWS のサービス使用する のセキュリティ設定および管理タスクが含まれます。データプライバシーの詳細については、データプライバシーに関するよくある質問を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティブログに投稿された「AWS 責任共有モデルおよび GDPR」のブログ記事を参照してください。

- データ保護の目的で、認証情報を保護し AWS アカウント 、 AWS IAM アイデンティティセンターまたは AWS Identity and Access Management () を使用して個々のユーザーを設定することをお勧めしますIAM。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。
- 次の方法でデータを保護することをお勧めします。
  - 各アカウントで多要素認証 (MFA) を使用します。
  - SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 が必要です。
  - で API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。
  - AWS 暗号化ソリューションと、 内のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
  - などの高度なマネージドセキュリティサービスを使用して Amazon Macie、 に保存されている機密データの検出と保護を支援します Amazon S3。
  - コマンドラインインターフェイスまたは API を使用して AWS にアクセスするときに FIPS 140-2 検証済みの暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-2」を参照してください。
- お客様の E メールアドレスなどの極秘または機密情報は、タグ、または名前フィールドなどの自由形式のテキストフィールドに配置しないことを強くお勧めします。これは、コンソール、API、または SDK AWS のサービス を使用して Verified Permissions AWS CLIまたは他の を使用する場合も同様です。 AWS SDKs タグ、または名前に使用される自由記述のテキストフィールドに入力したデータは、請求または診断ログに使用される場合があります。外部サーバーに URL を提供する場合、そのサーバーへのリクエストを検証できるように、認証情報を URL に含めないことを強くお勧めします。
- アクション名には、機密情報を含めないでください。
- また、エンティティ (リソースとプリンシパル) には、常に一意で、変更できない、再利用できない識別子を使用することを強くお勧めします。テスト環境では、`jane` タイプのエンティティの名前に `bob` や `User` などの単純なエンティティ識別子を使用することを選択できます。ただし、実稼働システムでは、セキュリティ上の理由から、再利用できない一意の値を使用することが重要です。ユニバーサルユニーク識別子 (UUID) などの値を使用することをおすすめします。たとえば、

会社を辞めるユーザーjaneを考えてみましょう。後で、他の人に名前janeを使わせます。その新しいユーザーは、まだUser::"jane"を参照しているポリシーによって付与されたすべてのものに自動的にアクセスできるようになります。Verified Permissions と Cedar は、新しいユーザーと以前のユーザーを区別できません。

このガイダンスはプリンシパル識別子とリソース識別子の両方に適用されます。ポリシーに古い識別子が含まれているために意図せずアクセスを許可してしまうことがないように、常に一意であることが保証され、再利用されない識別子を使用してください。

- Long および Decimal 値を定義するために指定した文字列が各タイプの有効範囲内であることを確認してください。また、算術演算子を使用しても有効範囲外の値にならないようにしてください。範囲を超えると、操作によりオーバーフロー例外が発生します。エラーとなるポリシーは無視されます。つまり、許可ポリシーが予期せずアクセスを許可しなかったり、禁止ポリシーが予期せずアクセスをブロックできなかったりする可能性があります。

データ暗号化

Amazon Verified Permissions は、 ポリシーなどのすべての顧客データを で自動的に暗号化します AWS マネージドキー。Amazon Verified Permissions では、お客様は カスタマー管理キー を使用してデータを暗号化することもできます。

暗号化にカスタマーマネージドキーを使用する方法の詳細については、「」を参照してくださいthe section called “カスタマーマネージドキー”。

Amazon Verified Permissions でのリソースの暗号化

Amazon Verified Permissions は、デフォルトで暗号化を提供し、 AWS 所有の暗号化キーを使用して保管中の機密データを保護します。Amazon Verified Permissions では、追加の保護レイヤーとして AWS Key Management Service 、(AWS KMS) カスタマー管理キー(CMK) を使用してポリシーストアを暗号化できます。この機能により、保管時の暗号化による機密データの保護が保証され、以下に役立ちます。

- アプリケーション側の運用負担を軽減して機密データを保護する
- 独自の を介して承認ポリシーの詳細を表示できるユーザーを管理する AWS KMS カスタマー管理キー
- セキュリティを重視したアプリケーションを構築して、暗号化のコンプライアンスと厳格な規制要件を満たす

以下のセクションでは、新しいポリシーストアの暗号化を設定し、暗号化キーを管理する方法について説明します。

AWS KMS Amazon Verified Permissions のキータイプ

Amazon Verified Permissions は と統合 AWS KMS して、顧客データの暗号化/復号に使用される暗号化キーを管理します。キーの種類と詳細については、「AWS KMS デベロッパーガイド」の「AWS Key Management Service concepts」を参照してください。新しいポリシーストアを作成するときに、次の AWS KMS キータイプから選択してデータを暗号化できます。

AWS 所有キー

デフォルトの暗号化タイプ。Amazon Verified Permissions は、追加料金なしでキーを所有し、作成時に保管中のリソースデータを暗号化します。Verified Permissions が所有するキーを使用してデータを暗号化/復号するために、コードまたはアプリケーションに追加の設定は必要ありません。

カスタマーマネージドキー

AWS アカウントでキーを作成、所有、管理します。 AWS KMS キーを完全に制御できます。 カスタマー管理キーには AWS KMS 料金が適用されます。詳細については、AWS KMS 料金表ページを参照してください。キータイプの詳細については、「 AWS KMS デベロッパーガイド」の「カスタマーマネージドキー」を参照してください。

最上位リソース (ポリシーストア) カスタマー管理キー の暗号化に を指定すると、Verified Permissions はそのキーを使用してリソースとその子リソースを暗号化します。を使用してポリシーストアを暗号化するには カスタマー管理キー、キーポリシーで Verified Permissions へのアクセスを許可する必要があります。キーポリシーは、アクセスを制御する カスタマー管理キー ために にアタッチするリソースベースのポリシーです。詳細については、「the section called “Amazon Verified Permissions の AWS KMS キーの使用を許可する”」を参照してください。

さらに、 を使用して暗号化されたポリシーストアを作成したり カスタマー管理キー、 によって暗号化されたポリシーストアに API コールを行うには カスタマー管理キー、呼び出しを行う IAM ユーザーまたはロールもキーにアクセスできる必要があります。Verified Permissions がキーにアクセスできない場合、そのキーによって暗号化されたリソースを含む承認の決定は古くなったり不正確になったりする可能性があります。キーにアクセスできない場合、そのキーによって暗号化されたリソースのread/update/deleteはできず、暗号化にキーを使用するための作成呼び出しは失敗します。

> **Note**
>
> Verified Permissions 保管時の暗号化は、Verified Permissions が利用可能なすべての AWS リージョンで使用できます。

> **Important**
>
> カスタマー管理キー を使用してポリシーストアを暗号化すると、暗号化に別のキーを使用するようにリソースを更新したり、そのポリシーストアからキーを削除したりすることはできません。

Amazon Verified Permissions での AWS KMS および データキーの使用

Amazon Verified Permissions の保管時の暗号化機能では、 AWS KMS キーとデータキーの階層を使用してリソースデータを保護します。

> **Note**
>
> Amazon Verified Permissions は対称 AWS KMS キーのみをサポートします。非対称 AWS KMS キーを使用して Amazon Verified Permissions リソースを暗号化することはできません。

AWS 所有キーの使用

Amazon Verified Permissions は、デフォルトですべてのリソースを AWS 所有キーで暗号化します。これらのキーは、アカウントリソースを保護するために無料で使用でき、毎年ローテーションします。キーを表示、管理、使用、または監査する必要がないため、データ保護のためのアクションは必要ありません。 AWS 所有キーの詳細については、「 AWS KMS デベロッパーガイド」のAWS「 所有キー」を参照してください。

カスタマーマネージドキーの使用

暗号化 カスタマー管理キー 用の を選択すると、次の利点があります。

- AWS KMS キーの作成と管理には、キーポリシーとキーへのアクセスを制御する IAM ポリシーの設定が含まれます AWS KMS 。 AWS KMS キーを有効または無効にしたり、自動キーローテー

ションを有効または無効にしたり、使用されなくなったときに AWS KMS キーを削除したりできます。

- インポートされたキーマテリアル カスタマー管理キー で を使用するか、所有および管理しているカスタムキーストア カスタマー管理キー で を使用できます。
- AWS KMS AWS CloudTrail ログで への Amazon Verified Permissions API コールを調べることで、Verified Permissions リソースの暗号化と復号化を監査できます。

Amazon Verified Permissions が暗号化/復号に カスタマー管理キーを使用するには、特定のキーポリシーを追加して、Amazon Verified Permissions がユーザーに代わってリソースを暗号化/復号できるようにする必要があります。

Amazon Verified Permissions の AWS KMS キーの使用を許可する

少なくとも、Amazon Verified Permissions には、 に対する次のアクセス許可が必要です カスタマー管理キー。

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

キーポリシーの例を以下に示します。

```
{
    "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": "*"
}
```

ソースコンテキストについて

ソースコンテキストは、特定のキーに対して AWS KMS アクションを実行しようとするソース発信者に関する情報を提供します。これにより、コンテキストをデータのソースにバインドすることで、暗号化されたデータの混乱や誤用を防ぐことができます。

お客様は、次のキーポリシーステートメントなど、キーポリシーの追加条件としてソースコンテキストを使用できます。

```
{
    "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/*"
                }
            }
        }
    ]
}
```

このキーポリシーは、ソースアカウントがこの AWS KMS キーが存在するアカウントと同じ場合、Verified Permissions がユーザーに代わって AWS KMS 呼び出しを行うことを許可します。これらの値は、CMK キーの AWS CloudTrail 監査ログを確認するときに検証可能である必要があります。グローバル AWS 条件キーの詳細については、「aws:SourceArnまたはaws:SourceAccount条件キーの使用」を参照してください。

暗号化コンテキストについて

暗号化コンテキストは、暗号化整合性チェックのための追加の認証済みデータを含むキーと値のペアのセットです。データを暗号化するリクエストに暗号化コンテキストを含めると、 は暗号化コンテキストを暗号化されたデータに AWS KMS 暗号化バインドします。データを復号するには、同じ暗号化コンテキストを渡す必要があります。

Amazon Verified Permissions は、すべての AWS KMS 暗号化オペレーションで同じ暗号化コンテキストを使用し、Verified Permissions が暗号化/復号プロセスのためにユーザーに代わって AWS KMS 呼び出しを行うときにログ内で AWS CloudTrail 検証できます。デフォルトでは、Verified Permissions は、リソースを暗号化するときに次の暗号化コンテキストのキーと値のペアを使用します。

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

Amazon Verified Permissions では、暗号化/復号プロセス中に含めたい追加のメタデータの一部としてカスタム暗号化コンテキストを追加することもできます。つまり、キーポリシーは、以下の例のようなアクセス許可をよりきめ細かく付与できます。

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

このキーポリシーは、暗号化コンテキストマップにキー が含まれ、その値が の形式に従い、キーarn:aws:verifiedpermissions::111122223333:policy-store/*と値のペア が含まれている場合aws:verifiedpermissions:policy-store-arn、Verified Permissions がユーザーに代わって AWS KMS 呼び出しを行うことを許可します"policy_owner": "Tim"。カスタム暗号化コンテキストを設定する方法については、the section called “暗号化ポリシーストアの作成”「」を参照してください。

> **Note**
>
> 暗号化コンテキストに基づく条件を持つキーポリシーでは、各キーと値のペアをチェックするのではなく、暗号化コンテキストマップのサブセットにすることをお勧めします。サービスとその依存関係のアップストリームでは、表示されないキーと値のペアが追加される場合があり、キーポリシーで暗号化コンテキストマップの正確な外観に基づいて条件付きで が許可されていると、Verified Permissions のキーアクセスに影響する可能性があります。

kms:ViaService について

kms:ViaService 条件キーは、 AWS KMS キーの使用を、指定された AWS サービスからのリクエストに制限します。この条件キーは、転送アクセスセッション (FAS) にのみ適用されます。の詳細についてはkms:ViaService、 AWS KMS デベロッパーガイドのkms:ViaService」を参照してください。

たとえば、次のキーポリシーステートメントでは、 kms:ViaService条件キーを使用して、リクエストカスタマー管理キーが に代わって米国東部 (バージニア北部) リージョンの Amazon Verified Permissions から送信された場合にのみ、指定されたアクションに を使用することを許可しますBrentRole。

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

これは、Verified Permissions が AWS KMS ユーザーに代わって暗号化/復号を にリクエストするときに、Verified Permissions が ID、アクセス許可、およびセッション属性を渡すために必要です。FAS リクエストの詳細については、IAM 「 ユーザーガイド」の「Forward Access Sessions」を参照してください。

AWS KMS キーポリシーを完了する

前のセクションの概念に基づいて、Amazon Verified Permissions が暗号化/復号化に CMK を使用できるようにするキーポリシーの例を次に示します。

```
{
    "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**
>
> Amazon Verified Permissions で既に使用されている AWS KMS キーのキーポリシーを変更するときは注意が必要です。Verified Permissions は、最上位のリソース作成時に最初に AWS KMS キーを設定するときに暗号化と復号のアクセス許可を検証しますが、後続のポリシー変更をオンデマンドで検証することはできません。必要なアクセス許可を不注意で削除すると、認可の決定と定期的な Verified Permissions サービスフローが中断される可能性があります。Amazon Verified Permissions の カスタマー管理キーに関連する一般的なエラーのトラブルシューティングのガイダンスについては、「」を参照してくださいthe section called “Amazon Verified Permissions のカスタマーマネージドキーのトラブルシューティング”。

暗号化されたリソースに必要な IAM ポリシー

アカウント内の IAM ロールを介して Verified Permissions を呼び出すお客様は、対応する IAM ポリシーに、リソースの暗号化と復号 カスタマー管理キー のために を利用するための適切なアクセス許可があることを確認する必要があります。

によって暗号化されたポリシーストアを作成する場合 カスタマー管理キー、次の IAM ポリシーは、必要な最低限のアクション AWS KMS と、それを実行するための Verified Permissions アクションを示しています。

```
{
    "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**
>
> (Get* および List* オペレーション) を取得し、 によって暗号化されたポリシーストアを削除するには カスタマー管理キー、追加のアクセス許可は必要ありません。

で暗号化されたポリシーストアの更新 カスタマー管理キー、 (Get* および List* オペレーション) の取得、 で暗号化されたポリシーストアの子リソースの更新および削除を行う場合 カスタマー管理キー、次の IAM ポリシーは、必要最小限のポリシー AWS KMS と Verified Permissions アクションを示しています。

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

単一の IAM ポリシーとして、お客様は IAM ロールポリシーに以下を追加するだけです。

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

暗号化されたポリシーストアの管理

ポリシーストアは、関連するすべてのポリシーリソースを含むエントリレベルのコンテナです。ポリシーストアと子リソースの階層の詳細については、「Amazon Verified Permissions ユーザーガイド」の「Amazon Verified Permissions ポリシーストア」を参照してください。

Verified Permissions でポリシーストアを作成すると、 AWS KMS キーを使用して保管時の暗号化を有効にできます。これにより、以下が保証されます。

- ポリシーストアとその子リソースに対するすべての読み取り、更新、削除オペレーションは、復号プロセス カスタマー管理キー に提供された を利用します。
- 認可決定呼び出し (IsAuthorized、BatchIsAuthorized、IsAuthorizedWithToken など) では、復号プロセス カスタマー管理キー に提供された が使用されます。

暗号化ポリシーストアの作成

暗号化されたポリシーストアを作成する前に、 カスタマー管理キー 使用している に、Amazon Verified Permissions がキーを暗号化/復号化に使用するための適切なキーポリシーステートメントが設定されていることを確認してください。必要なアクセス許可the section called “Amazon Verified Permissions の AWS KMS キーの使用を許可する”については、「」を参照してください。

の使用 AWS CLI:

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

encrypted.json は次のようになります。

```
{
    "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>",
            ...
        }
    }
}
```

を カスタマー管理キー ARN keyに置き換え、 <ENCRYPTION_CONTEXT_KEY>と<ENCRYPTION_CONTEXT_VALUE>ペアを目的のencryptionContextキーと値のペアに置き換えます。キーと値のペアを追加しない場合は、 を完全に省略encryptionContextできます。

> **Important**
>
> aws:verifiedpermissions:policy-store-arn カスタム暗号化コンテキストにキーと値のペアを含めないでください。これは自動的に追加され、渡されたカスタム暗号化コンテキストのキーと値のペアの一部である場合、検証エラーが発生します。

ポリシーストアの子リソースで使用可能な APIs「Amazon Verified Permissions API リファレンスガイド」の「アクション」を参照してください。

> **Note**
>
> Amazon Verified Permissions リソースで AWS KMS カスタマー管理キー 使用されているが、 AWS KMS キーポリシーが正しくないために削除、無効化、またはアクセスできない場合、リソースの復号は失敗するため、承認の決定が古くなります。アクセスの喪失は、状況に応じて、一時的 (キーポリシーを修正できる) または永続的 (削除されたキーを復元できない) にすることができます。 AWS KMS キーの削除や無効化など、重要なオペレーションへのアクセスを制限することをお勧めします。また、Amazon Verified Permissions にアクセスできない AWS 万一、特権ユーザーがアクセスできるように、ブレークAWS グラスアクセス手順を設定することをお勧めします。

を使用した Amazon Verified Permissions インタラクションのモニタリング AWS KMS

Amazon Verified Permissions による の使用 カスタマー管理キー 状況をモニタリングできます AWS CloudTrail。Verified Permissions AWS KMS を介した への各リクエストには、リクエストパラメータで使用されている暗号化コンテキストとキー ARN (カスタマー管理キー) が含まれます。

の AWS CloudTrail ログエントリの例`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"
```

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

の AWS CloudTrail ログエントリの例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"
        }
    },
    ...
}
```

の AWS CloudTrail ログエントリの例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",
        ...
    },
    ...
}
```

ログエントリには、Amazon Verified Permissions のプリンシパルのinvokedBy参照と、requestParametersマップencryptionContext/sourceEncryptionContext/destinationEncryptionContextに含まれていることに注意してください。

の AWS CloudTrail ログエントリの例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"
    },
    ...
}
```

ログエントリには Amazon Verified Permissions のプリンシパルのinvokedBy参照が含まれていることに注意してください。

AWS CloudTrail ログエントリの詳細については、AWS CloudTrail 「 ユーザーガイド」のAWS CloudTrail 「イベントについて」を参照してください。

制限事項

このトピックでは、Verified Permissions の現在の制限と、リソースの暗号化のための カスタマー管理キーの使用について説明します。

- 一度有効にすると、ポリシーストアの暗号化を無効にすることはできません
- 暗号化なしでポリシーストアを作成した後、 によって暗号化されるようにポリシーストアを更新することはできません。 カスタマー管理キー
- 既存の暗号化されたポリシーストア カスタマー管理キー の への Verified Permissions アクセスを取り消すと、認可の決定が古くなる可能性があります。
- を使用してポリシーストアを作成した後は カスタマー管理キー、カスタム暗号化コンテキスト値を変更することはできません。これらは、暗号化されたポリシーストアの作成時に設定された静的値です。

Amazon Verified Permissions のカスタマーマネージドキーのトラブルシューティング

このトピックでは、Amazon Verified Permissions を使用する際に発生する可能性がある一般的な カスタマー管理キー 関連エラーと、それらを解決するためのトラブルシューティング手順について説明します。

アクセス拒否: AWS KMS アクセス許可の問題

エラー:「リソースがこのリージョンに存在しないか、リソースベースのポリシーがアクセスを許可していないか、リソースベースのポリシーが明示的にアクセスを拒否しているため、サービスまたは呼び出し元は指定された AWS KMS キーを使用する権限がありません」

これは、サービスまたは呼び出し元が IAM ポリシー/AWS KMS キーポリシーに必要なkms:*アクション (複数可) アクセス許可を持っていないか、参照されているキーが存在しないか、存在しなくなったことを意味します。

を使用したトラブルシューティング AWS CloudTrail:

- でkms.amazonaws.comイベントを検索する AWS CloudTrail
- 許可されていないと特定された AWS KMS オペレーションのイベント名を検索します (例: Decrypt、ReEncryptGenerateDataKeyWithoutPlaintext、、 DescribeKeyなど) 。
- errorCode フィールドと errorMessageフィールドを確認する
- どのプリンシパルがオペレーションを試みたか userIdentity を確認する

この問題を解決するには、ポリシーと AWS KMS キー IAM ポリシーで適切な AWS KMS オペレーションアクセス許可をユーザーまたは IAM プリンシパルに付与します。詳細については、「the section called “AWS KMS キーポリシーを完了する”」を参照してください。

検証例外: AWS KMS キー設定

エラー:「設定済み AWS KMS キーに有効な設定がありません」

つまり、参照されているキーは、現在の設定のため カスタマー管理キー 、暗号化のためにサービスで使用することはできません。理由には、キーが無効になっている、キーがサポートされていない EncryptionAlgorithm、またはキーがサポートされていない KeyUsage タイプが含まれる場合があります。

スロットリング例外: AWS KMS レート制限

エラー:「呼び出すことができるレートを超えました AWS KMS」

このエラーは、キーの暗号化オペレーション AWS KMS の制限を超過したことを意味します: https://docs.aws.amazon.com/kms/latest/developerguide/requests-per-second.html。

関連情報

- Verified Permissions Policy ストアの管理
- AWS KMS ベストプラクティス
- AWS KMS 暗号化コンテキスト
- AWS CloudTrail Integration
- AWS CloudTrail ログエントリの例

Amazon Verified Permissions のためのアイデンティティとアクセス管理

AWS Identity and Access Management (IAM) は、管理者が AWS resources へのアクセスを安全に制御 AWS のサービス するのに役立つ です。 IAM 管理者は、誰を認証 (サインイン) し、誰に Verified Permissions リソースの使用を許可する (アクセス許可を付与 AWS のサービス する) かを制御します。 IAM は、追加料金なしで使用できる です。

トピック

オーディエンス

AWS Identity and Access Management (IAM) の使用方法は、ロールによって異なります。

- サービスユーザー - 機能にアクセスできない場合は、管理者にアクセス許可をリクエストします (「Amazon Verified Permissions アイデンティティとアクセスのトラブルシューティング」を参照)。
- サービス管理者 - ユーザーアクセスを決定し、アクセス許可リクエストを送信します (「Amazon Verified Permissions と の連携方法 IAM」を参照)
- IAM 管理者 - アクセスを管理するポリシーを記述する (「」を参照Amazon Verified Permissions のアイデンティティベースのポリシー例)

アイデンティティを使用した認証

認証とは、ID 認証情報 AWS を使用して にサインインする方法です。、IAM ユーザー AWS アカウントのルートユーザー、または IAM ロールを引き受けることで認証される必要があります。

AWS IAM アイデンティティセンター (IAM Identity Center)、シングルサインオン認証、Google/Facebook 認証情報などの ID ソースからの認証情報を使用して、フェデレーティッド ID としてサインインできます。サインインの詳細については、「AWS サインイン ユーザーガイド」の「AWS アカウントにサインインする方法」を参照してください。

プログラムによるアクセスの場合、 は SDK と CLI AWS を提供してリクエストを暗号化して署名します。詳細については、「 IAM ユーザーガイド」のAWS 「API リクエストの署名バージョン 4」を参照してください。

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、まず、すべての AWS のサービス および リソースへの完全なアクセス権を持つ AWS アカウント root ユーザーと呼ばれる 1 つのサインインアイデンティティから始めます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザー認証情報を必要とするタスクについては、IAM 「 ユーザーガイド」の「ルートユーザー認証情報を必要とするタスク」を参照してください。

フェデレーテッドアイデンティティ

ベストプラクティスとして、人間のユーザーが一時的な認証情報 AWS のサービス を使用して にアクセスするには、ID プロバイダーとのフェデレーションを使用する必要があります。

フェデレーティッド ID は、エンタープライズディレクトリ、ウェブ ID プロバイダー、または ID Directory Service ソースの認証情報 AWS のサービス を使用して にアクセスするユーザーです。フェデレーテッドアイデンティティは、一時的な認証情報を提供するロールを引き受けます。

アクセスを一元管理する場合は、 AWS IAM アイデンティティセンターをお勧めします。詳細については、「AWS IAM アイデンティティセンター ユーザーガイド」の「IAM アイデンティティセンターとは」を参照してください。

IAM ユーザーとグループ

IAM ユーザーは、特定の個人やアプリケーションに対する特定のアクセス許可を持つアイデンティティです。長期認証情報を持つ IAM ユーザーの代わりに一時的な認証情報を使用することをお勧めします。詳細については、「 IAM ユーザーガイド」の「ID プロバイダーとのフェデレーション AWS を使用して にアクセスする必要がある」を参照してください。

IAM グループは IAM ユーザーのコレクションを指定し、大量のユーザーのアクセス許可の管理を容易にします。詳細については、「 ユーザーガイド」の IAM 「ユーザーのユースケース」を参照してください。 IAM

IAM ロール

IAM ロールは、特定のアクセス許可 AWS アカウント を持つ 内の ID です。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。 IAM ロールを切り替える AWS マネジメントコンソール ことで、 で ロールを一時的に引き受けることができます。ロールを引き受けるには、 AWS CLI または AWS API オペレーションを呼び出すか、カスタム URL を使用します。ロールの使用方法の詳細については、「 IAM ユーザーガイド」のIAM 「ロールの使用」を参照してください。

IAM 一時的な認証情報を持つ ロールは、以下の状況で役立ちます。

- フェデレーションユーザーアクセス – フェデレーテッド ID に許可を割り当てるには、ロールを作成してそのロールの許可を定義します。フェデレーテッド ID が認証されると、その ID はロールに関連付けられ、ロールで定義されている許可が付与されます。フェデレーションのロールの詳細については、 IAM ユーザーガイドの「サードパーティー ID プロバイダーのロールを作成する (フェデレーション)」を参照してください。IAM Identity Center を使用する場合は、許可セットを設定します。ID が認証後にアクセスできるものをコントロールするために、IAM アイデンティティセンター は権限セットを IAMのロールに関連付けます。アクセス許可セットの詳細については、AWS IAM アイデンティティセンター User Guide の Permission sets を参照してください。
- 一時的な IAM ユーザーアクセス許可 – IAM ユーザーまたはロールは、特定のタスクに対して異なるアクセス許可を一時的に引き受ける IAM ロールを引き受けることができます。
- クロスアカウントアクセス IAM ロールを使用して、自分のアカウントのリソースにアクセスすることを別のアカウントの信頼済みプリンシパルに許可できます。クロスアカウントアクセスを許可する主な方法は、ロールを使用することです。ただし、一部の では AWS のサービス、 (プロキシとしてロールを使用する代わりに) リソースに直接ポリシーをアタッチできます。クロスアカウントアクセスのロールとリソースベースのポリシーの違いについては、「 IAM ユーザーガイド」のIAM 「ロールとリソースベースのポリシーの違い」を参照してください。
- で実行されているアプリケーション Amazon EC2 – IAM ロールを使用して、EC2 インスタンスで実行され、 AWS CLI または AWS API リクエストを行うアプリケーションの一時的な認証情報を管理できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。EC2 インスタンスに AWS ロールを割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスにアタッチされたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、「 ユーザーガイド」の「 IAM ロールを使用して Amazon EC2 インスタンスで実行されているアプリケーションにアクセス許可を付与する」を参照してください。 IAM

IAM ロールと IAM ユーザーのどちらを使用するかについては、IAM 「 ユーザーガイド」の「 (ユーザーではなく) IAM ロールを作成するタイミング」を参照してください。

ポリシーを使用したアクセスの管理

でアクセスを制御する AWS には、ポリシーを作成し、ID AWS またはリソースにアタッチします。ポリシーは、ID またはリソースに関連付けられたときにアクセス許可を定義します。 は、プリンシパルがリクエストを行うときにこれらのポリシー AWS を評価します。ほとんどのポリシーは JSON ドキュメント AWS として に保存されます。JSON ポリシードキュメントの詳細については、IAM 「 ユーザーガイド」の「JSON ポリシーの概要」を参照してください。

管理者は、ポリシーを使用して、どのプリンシパルがどのリソースに対して、どのような条件でアクションを実行できるかを定義することで、誰が何にアクセスできるかを指定します。

デフォルトでは、ユーザーやロールにアクセス許可はありません。 IAM 管理者は IAM ポリシーを作成し、ロールに追加します。ロールは、ユーザーが引き受けることができます。 IAM ポリシーは、オペレーションの実行に使用された方法に関係なく、アクセス許可を定義します。

アイデンティティベースのポリシー

アイデンティティベースのポリシーは、アイデンティティ (ユーザー、グループ、またはロール) にアタッチできる JSON アクセス許可ポリシードキュメントです。これらのポリシーは、アイデンティティがどのリソースに対してどのような条件下でどのようなアクションを実行できるかを制御します。ID ベースのポリシーを作成する方法については、「 IAM ユーザーガイド」の「カスタマー管理ポリシーを使用したカスタム IAM アクセス許可の定義」を参照してください。

アイデンティティベースのポリシーは、インラインポリシー (単一の ID に直接埋め込む) または管理ポリシー (複数の ID にアタッチされたスタンドアロンポリシー) にすることができます。管理ポリシーとインラインポリシーを選択する方法については、 IAM ユーザーガイドの「管理ポリシーとインラインポリシーの選択」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。例としては、 IAM ロール信頼ポリシーと Amazon S3 バケットポリシーなどがあります。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシー IAM では、 から AWS 管理ポリシーを使用することはできません。

アクセスコントロールリスト (ACL)

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするためのアクセス許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon S3、 AWS WAF、および Amazon VPC は、ACLs。ACL の詳細については、Amazon Simple Storage Service デベロッパーガイド の アクセスコントロールリスト (ACL) の概要 を参照してください。

その他のポリシータイプ

AWS は、より一般的なポリシータイプによって付与されるアクセス許可の上限を設定できる追加のポリシータイプをサポートしています。

- アクセス許可の境界 — アイデンティティベースのポリシーが IAM エンティティに付与できるアクセス許可の上限を設定します。詳細については、「 IAM ユーザーガイド」の IAM 「エンティティのアクセス許可の境界」を参照してください。
- サービスコントロールポリシー (SCP) - AWS Organizations内の組織または組織単位の最大のアクセス許可を指定します。詳細については、「AWS Organizations ユーザーガイド」の「サービスコントロールポリシー」を参照してください。
- リソースコントロールポリシー (RCP) – は、アカウント内のリソースで利用できる最大数のアクセス許可を定義します。詳細については、「AWS Organizations ユーザーガイド」の「リソースコントロールポリシー (RCP)」を参照してください。
- セッションポリシー – ロールまたはフェデレーションユーザーの一時セッションを作成する際にパラメータとして渡される高度なポリシーです。詳細については、IAM ユーザーガイドの「セッションポリシー」を参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成されるアクセス許可を理解するのがさらに難しくなります。が複数のポリシータイプが関与する場合にリクエストを許可するかどうか AWS を決定する方法については、「 IAM ユーザーガイド」の「ポリシー評価ロジック」を参照してください。

Amazon Verified Permissions と の連携方法 IAM

IAM を使用して Verified Permissions へのアクセスを管理する前に、Verified Permissions で使用できる IAM 機能を確認してください。

IAM Amazon Verified Permissions で使用できる機能

IAM 機能	Verified Permissions サポート
アイデンティティベースのポリシー	あり
リソースベースのポリシー	なし
ポリシーアクション	あり
ポリシーリソース	あり
ポリシー条件キー	いいえ
ACL	なし
ABAC (ポリシー内のタグ)	あり
一時的な認証情報	あり
プリンシパルアクセス権限	あり
サービスロール	いいえ
サービスリンクロール	いいえ

Verified Permissions およびその他の AWS のサービスがほとんどの IAM 機能と連携する方法の概要を把握するには、「 IAM ユーザーガイド」のAWS 「 と連携する のサービス IAM」を参照してください。

Verified Permissions の アイデンティティベースのポリシー

アイデンティティベースポリシーをサポートする	あり

アイデンティティベースポリシーは、IAM ユーザー、ユーザーグループ、ロールなど、アイデンティティにアタッチできる JSON 許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。ID ベースのポリシーを作成する方法については、「 IAM ユーザーガイド」の「カスタマー管理ポリシーによるカスタム IAM アクセス許可の定義」を参照してください。

IAM アイデンティティベースのポリシーでは、許可または拒否されたアクションとリソース、およびアクションが許可または拒否される条件を指定できます。JSON ポリシーで使用できるすべての エレメントについては、 IAM ユーザーガイド の「IAM JSON ポリシーエレメントリファレンス」を参照してください。

Verified Permissions の アイデンティティベースのポリシー例

Verified Permissions ID ベースポリシーの例を確認するには、「Amazon Verified Permissions の アイデンティティベースのポリシー例」を参照してください。

Verified Permissions 内のリソースベースのポリシー

リソースベースのポリシーのサポート	なし

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーの例としては、 IAM ロールの信頼ポリシーと Amazon S3 バケットポリシーがあります。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスをコントロールできます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーで、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

クロスアカウントアクセスを有効にするには、リソースベースのポリシーのプリンシパルとして、別のアカウントのアカウントまたは IAM エンティティ全体を指定できます。詳細については、「 IAM ユーザーガイド」の「 でのクロスアカウントリソースアクセス IAM」を参照してください。

Verified Permissions のポリシーアクション

ポリシーアクションに対するサポート	はい

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

JSON ポリシーの Action 要素にはポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。このアクションは関連付けられたオペレーションを実行するためのアクセス許可を付与するポリシーで使用されます。

Verified Permissions アクションのリストを確認するには、「サービス認可リファレンス」の「Amazon Verified Permissions で定義されるアクション」を参照してください。

Verified Permissions のポリシーアクションは、アクションの前にプレフィックスを使用します。

```
verifiedpermissions
```

単一のステートメントで複数のアクションを指定するには、アクションをカンマで区切ります。

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

ワイルドカード (*) を使用して複数アクションを指定できます。例えば、Get という単語で始まるすべてのアクションを指定するには次のアクションを含めます。

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

Verified Permissions ID ベースポリシーの例を確認するには、「Amazon Verified Permissions のアイデンティティベースのポリシー例」を参照してください。

Verified Permissions のポリシーリソース

ポリシーリソースに対するサポート	はい

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

`Resource` JSON ポリシー要素はアクションが適用されるオブジェクトを指定します。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。リソースレベルのアクセス許可をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

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

Verified Permissions リソースタイプとその ARNs「Amazon Verified Permissions で定義されるリソースタイプ」を参照してください。 どのアクションで各リソースの ARN を指定できるかについては、「Amazon Verified Permissions で定義されるアクション」を参照してください。

Verified Permissions の条件キー

サービス固有のポリシー条件キーのサポート	いいえ

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

`Condition` 要素は、定義された基準に基づいてステートメントが実行される時期を指定します。イコールや未満などの条件演算子を使用して条件式を作成して、ポリシーの条件とリクエスト内の値を一致させることができます。すべての AWS グローバル条件キーを確認するには、「 IAM ユーザーガイド」のAWS 「 グローバル条件コンテキストキー」を参照してください。

Verified Permissions の ACL

ACL のサポート	なし

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための許可を持つかをコントロールします。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Verified Permissionsのある ABAC

ABAC のサポート (ポリシー内のタグ)	はい

属性ベースのアクセス制御 (ABAC) は、タグと呼ばれる属性に基づいてアクセス許可を定義する認可戦略です。 IAM エンティティと AWS リソースにタグをアタッチし、プリンシパルのタグがリソースのタグと一致するときにオペレーションを許可するように ABAC ポリシーを設計できます。

タグに基づいてアクセスを管理するには、`aws:ResourceTag/key-name`、`aws:RequestTag/key-name`、または `aws:TagKeys` の条件キーを使用して、ポリシーの条件要素でタグ情報を提供します。

サービスがすべてのリソースタイプに対して 3 つの条件キーすべてをサポートする場合、そのサービスの値はありです。サービスが一部のリソースタイプに対してのみ 3 つの条件キーのすべてをサポートする場合、値は「部分的」になります。

ABAC の詳細については、IAM 「 ユーザーガイド」の「ABAC 認可によるアクセス許可の定義」を参照してください。ABAC をセットアップするステップを説明するチュートリアルについては、「IAM ユーザーガイド」の「属性に基づくアクセスコントロール (ABAC) を使用する」を参照してください。

Verified Permissions での一時認証情報の使用

一時的な認証情報のサポート	はい

一時的な認証情報は、 AWS リソースへの短期的なアクセスを提供し、フェデレーションまたはスイッチロールの使用時に自動的に作成されます。長期的なアクセスキーを使用する代わりに、一時的な認証情報を動的に生成 AWS することをお勧めします。詳細については、「 ユーザーガイド」の「IAM と で動作する一時的なセキュリティ認証情報」を参照してください。 AWS のサービスIAM IAM

Verified Permissions のクロスサービスプリンシパル許可

プリンシパル権限のサポート	はい

転送アクセスセッション (FAS) は、 を呼び出すプリンシパルのアクセス許可と AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストをリクエストする を使用します。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。

Verified Permissions のサービスロール

サービスロールのサポート	いいえ

サービスロールとは、サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロール です。 IAM 管理者は、 内からサービスロールを作成、変更、削除できます IAM。詳細については、「 IAM ユーザーガイド」の「 にアクセス許可を委任するロールを作成する AWS のサービス」を参照してください。

Verified Permissions のサービスにリンクされたロールのアクセス許可

サービスにリンクされたロールのサポート	いいえ

サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。 IAM 管理者は、サービスにリンクされたロールのアクセス許可を表示できますが、編集することはできません。

サービスにリンクされたロールの作成または管理の詳細については、 AWS が使用する のサービス IAMを参照してください。表の「サービスリンクロール」列に Yes と記載されたサービスを見つけます。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[はい] リンクを選択します。

IAM Verified Permissions の ポリシー

Verified Permissions は、アプリケーション内のユーザーの権限を管理します。アプリケーションが Verified Permissions APIs を呼び出すか、 AWS マネジメントコンソール ユーザーが Verified Permissions ポリシーストアで Cedar ポリシーを管理できるようにするには、必要な IAM アクセス許可を追加する必要があります。

ID ベースのポリシーは、 IAM ユーザー、ユーザーのグループ、ロールなどの ID にアタッチできる JSON アクセス許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティベースのポリシーを作成する方法については、「 IAM ユーザーガイド」のIAM 「ポリシーの作成」を参照してください。

IAM アイデンティティベースのポリシーでは、許可または拒否されたアクションとリソース、およびアクションが許可または拒否される条件 (以下を参照) を指定できます。プリンシパルは、それが添付されているユーザーまたはロールに適用されるため、アイデンティティベースのポリシーでは指定できません。JSON ポリシーで使用できるすべての要素については、 IAM 「 ユーザーガイド」のIAM 「JSON ポリシー要素リファレンス」を参照してください。

[アクション]	説明
CreateIdentitySource	新しい ID ソースを作成するアクション。
CreatePolicy	ポリシーストアに Cedar ポリシーを作成するアクション。静的ポリシーまたはポリシーテンプレートにリンクされたポリシーを作成できます。
CreatePolicyStore	新しいポリシーストアを作成するアクション。
CreatePolicyTemplate	新しいポリシーテンプレートを作成するアクション。
DeleteIdentitySource	ID ソースを削除するアクション。
DeletePolicy	ポリシーストアからポリシーを削除するアクション。
DeletePolicyStore	ポリシーストアを削除するアクション。
DeletePolicyTemplate	ポリシーテンプレートを削除するアクション。
GetIdentitySource	ID ソースを取得するアクション。
GetPolicy	指定されたポリシーに関する情報を取得するためのアクション。
GetPolicyStore	指定されたポリシーストアに関する情報を取得するアクション。
GetPolicyTemplate	ポリシーテンプレートを取得するアクション。
GetSchema	スキーマを取得するアクション。

[アクション]	説明
IsAuthorized	認可リクエストで説明されているパラメータに基づいて認可レスポンスを取得するアクション。 ???
IsAuthorizedWithToken	プリンシパルが ID トークンから取得される認可リクエストで説明されているパラメータに基づいて認可レスポンスを取得するアクション。
ListIdentitySources	内のすべての ID ソースを一覧表示するアクション AWS アカウント。
ListPolicies	ポリシーストア内のすべてのポリシーを一覧表示するアクション。
ListPolicyStores	内のすべてのポリシーストアを一覧表示するアクション AWS アカウント。
ListPolicyTemplates	内のすべてのポリシーテンプレートを一覧表示するアクション AWS アカウント。
ListTagsForResource	リソースのすべてのタグを一覧表示するアクション。
PutSchema	ポリシーストアにスキーマを追加するアクション。
TagResource	リソースにタグを追加するアクション。
UpdateIdentitySource	ID ソースを更新するアクション。
UpdatePolicy	ポリシーストアのポリシーを更新するアクション。
UpdatePolicyStore	ポリシーストアを更新するアクション。
UpdatePolicyTemplate	ポリシーテンプレートを更新するアクション。
UntagResource	リソースからタグを削除するアクション。

CreatePolicy アクションへのアクセス許可の IAM ポリシーの例:

JSON

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

Amazon Verified Permissions の アイデンティティベースのポリシー例

デフォルトでは、ユーザーおよびロールには Verified Permissions リソースを作成または変更する権限はありません。また、、 AWS Command Line Interface (AWS CLI) AWS マネジメントコンソール、または AWS API を使用してタスクを実行することはできません。 IAM 管理者は、必要なリソースに対してアクションを実行するアクセス許可をユーザーとロールに付与する IAM ポリシーを作成する必要があります。次に、管理者はこれらのポリシーを必要とするユーザーに、ポリシーをアタッチする必要があります。

これらのサンプル JSON ポリシードキュメントを使用して IAM アイデンティティベースのポリシーを作成する方法については、IAM 「 ユーザーガイド」の「ポリシーの作成 IAM」を参照してください。

各リソースタイプの ARNs「Amazon Verified Permissions のアクション、リソース、および条件キー」を参照してください。

トピック

- ポリシーに関するベストプラクティス
- Verified Permissions コンソールの使用
- 自分の権限の表示をユーザーに許可する

ポリシーに関するベストプラクティス

アイデンティティベースのポリシーは、誰かがアカウント内のVerified Permissions リソースを作成、アクセス、または削除できるかどうかを決定します。これらのアクションでは、 AWS アカウントに費用が発生する場合があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS 管理ポリシーを開始し、最小特権のアクセス許可に移行する – ユーザーとワークロードにアクセス許可の付与を開始するには、多くの一般的なユースケースにアクセス許可を付与するAWS 管理ポリシーを使用します。これらは で使用できます AWS アカウント。ユースケースに固有の AWS カスタマー管理ポリシーを定義することで、アクセス許可をさらに減らすことをお勧めします。詳細については、「IAM ユーザーガイド」の「AWS マネージドポリシー」または「AWS ジョブ機能の管理ポリシー」を参照してください。
- 最小特権のアクセス許可を適用する – IAM ポリシーでアクセス許可を設定する場合は、タスクの実行に必要なアクセス許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権アクセス許可とも呼ばれています。IAM を使用してアクセス許可を適用する方法の詳細については、「 IAM ユーザーガイド」の「 のポリシーとアクセス許可 IAM」を参照してください。
- IAM ポリシーの条件を使用してアクセスをさらに制限する – ポリシーに条件を追加して、アクションとリソースへのアクセスを制限できます。たとえば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。条件を使用して、サービスアクションが などの特定の を通じて使用されている場合に AWS のサービス、サービスアクションへのアクセスを許可することもできます CloudFormation。詳細については、IAM ユーザーガイドの「IAM JSON ポリシー要素: 条件」を参照してください。
- IAM Access Analyzer を使用して IAM ポリシーを検証し、安全で機能的なアクセス許可を確保する – IAM Access Analyzer は、ポリシーがポリシー言語 (JSON) と IAM のベストプラクティスに準拠するように、新規および既存の IAM ポリシーを検証します。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、「 IAM ユーザーガイド」の「IAM Access Analyzer を使用したポリシーの検証」を参照してください。
- 多要素認証 (MFA) を要求する – で IAM ユーザーまたはルートユーザーを必要とするシナリオがある場合は AWS アカウント、MFA をオンにしてセキュリティを強化します。API オペレーションが呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細については、「 IAM ユーザーガイド」の「Secure API access with MFA」を参照してください。

のベストプラクティスの詳細については IAM、「 IAM ユーザーガイド」の「 のセキュリティのベストプラクティス IAM」を参照してください。

Verified Permissions コンソールの使用

Amazon Verified Permissions コンソールにアクセスするには、許可の最小限のセットが必要です。これらのアクセス許可により、 内の Verified Permissions リソースの詳細を一覧表示および表示できます AWS アカウント。最小限必要な許可よりも制限が厳しいアイデンティティベースのポリシーを作成すると、そのポリシーを持つエンティティ (ユーザーまたはロール) に対してコンソールが意図したとおりに機能しません。

AWS CLI または AWS API のみを呼び出すユーザーには、最小限のコンソールアクセス許可を付与する必要はありません。代わりに、実行しようとしている API オペレーションに一致するアクションのみへのアクセスが許可されます。

ユーザーとロールが引き続き Verified Permissions コンソールを使用できるようにするには、エンティティに Verified Permissions *ConsoleAccess*または *ReadOnly* AWS 管理ポリシーもアタッチします。詳細については、「IAM ユーザーガイド」の「ユーザーへのアクセス許可の追加」を参照してください。

自分の権限の表示をユーザーに許可する

この例では、ユーザーアイデンティティにアタッチされたインラインおよびマネージドポリシーの表示を IAM ユーザーに許可するポリシーの作成方法を示します。このポリシーには、コンソールで、または AWS CLI または AWS API を使用してプログラムでこのアクションを実行するアクセス許可が含まれています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
```

```
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS Amazon Verified Permissions の マネージドポリシー

ユーザー、グループ、ロールにアクセス許可を追加するには、自分でポリシーを記述するよりも AWS 管理ポリシーを使用する方が簡単です。必要なアクセス許可のみをチームに提供するIAM カスタマー管理ポリシーを作成するには、時間と専門知識が必要です。すぐに開始するには、 AWS マネージドポリシーを使用できます。これらのポリシーは、一般的なユースケースをターゲット範囲に含めており、 AWS アカウントで利用できます。 AWS 管理ポリシーの詳細については、「 IAM ユーザーガイド」の「 AWS 管理ポリシー」を参照してください。

AWS サービスは、 AWS 管理ポリシーを維持および更新します。 AWS 管理ポリシーのアクセス許可は変更できません。サービスでは新しい機能を利用できるようにするために、 AWS マネージドポリシーに権限が追加されることがあります。この種類の更新はポリシーがアタッチされている、すべてのアイデンティティ (ユーザー、グループおよびロール) に影響を与えます。新しい機能が立ち上げられた場合や、新しいオペレーションが使用可能になった場合に、各サービスが AWS マネージドポリシーを更新する可能性が最も高くなります。サービスは AWS マネージドポリシーからアクセス許可を削除しないため、ポリシーの更新によって既存のアクセス許可が損なわれることはありません。

さらに、 は、複数の サービスにまたがるジョブ関数の マネージドポリシー AWS をサポートしています。例えば、ReadOnlyAccess AWS 管理ポリシーは、すべての AWS サービスとリソースへの読

み取り専用アクセスを提供します。サービスが新機能を起動すると、 は新しいオペレーションとリソースの読み取り専用アクセス許可 AWS を追加します。職務機能ポリシーのリストと説明については、IAM 「 ユーザーガイド」のAWS 「 職務機能の管理ポリシー」を参照してください。

AWS マネージドポリシー: AmazonVerifiedPermissionsFullAccess

`AmazonVerifiedPermissionsFullAccess` 管理ポリシーは、Verified Permissions へのフルアクセスを許可します。 Amazon Cognitoベースの ID ソースを使用するには、AmazonCognitoReadOnly ポリシーなどの別のポリシーをアタッチする必要があります。

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/*"
      ]
    }
  ]
}
```

AWS マネージドポリシー: AmazonVerifiedPermissionsReadOnlyAccess

AmazonVerifiedPermissionsReadOnlyAccess 管理ポリシーは、Verified Permissions への読み取り専用アクセスを許可します。

このポリシーは、認可クエリ APIs IsAuthorizedや など、Amazon Verified Permissions のすべての読み取りオペレーションへのアクセスを許可しますIsAuthorizedWithToken。

> **Note**
>
> BatchIsAuthorized および へのアクセスBatchIsAuthorizedWithTokenはIsAuthorizedWithToken、それぞれIsAuthorizedおよび へのアクセスが付与されると自動的に付与されます。

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/*"
      ]
    }
  ]
}
```

AWS 管理ポリシーに対する Verified Permissions の更新

このサービスがこれらの変更の追跡を開始してからの Verified Permissions の AWS マネージドポリシーの更新に関する詳細を表示します。このページの変更に関する自動アラートについては、Verified Permissions ドキュメント履歴ページの RSS フィードにサブスクライブしてください。

変更	説明	日付
AmazonVerifiedPermissionsFullAccess – 新しいポリシー	Verified Permissions に、Verified Permissions へのフルアクセスを許可する新しいポリシーが追加されました。	2024 年 10 月 11 日
AmazonVerifiedPermissionsReadOnlyAccess – 新しいポリシー	Verified Permissions に、認可クエリ APIsIsAuthorized や など、Amazon Verified Permissions のすべての読み取りオペレーションへのアクセスを許可する新しいポリシーが追加されましたIsAuthorizedWithToken 。	2024 年 10 月 11 日
Verified Permissions が変更の追跡を開始しました	Verified Permissions は、AWS 管理ポリシーの変更の追跡を開始しました。	2024 年 10 月 11 日

Amazon Verified Permissions アイデンティティとアクセスのトラブルシューティング

以下の情報を使用して、Amazon Verified Permissions と IAMの使用時に発生する可能性がある一般的な問題の診断と修正に役立てます。

トピック

Verified Permissions でアクションを実行する権限がありません

アクションを実行する権限がないというエラーが表示された場合は、そのアクションを実行できるようにポリシーを更新する必要があります。

次のエラー例は、mateojackson IAM ユーザーがコンソールを使用して、ある *my-example-widget* リソースに関する詳細情報を表示しようとしたことを想定して、その際に必要な verifiedpermissions:*GetWidget* アクセス許可を持っていない場合に発生するものです。

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

この場合、verifiedpermissions:*GetWidget* アクションを使用して *my-example-widget* リソースへのアクセスを許可するように、mateojackson ユーザーのポリシーを更新する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン認証情報を提供した担当者が管理者です。

iam: PassRole を実行する権限がありません

iam:PassRole アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更新して Verified Permissions にロールを渡すことができるようにする必要があります。

一部の AWS のサービス では、新しいサービスロールまたはサービスにリンクされたロールを作成する代わりに、既存のロールをそのサービスに渡すことができます。そのためには、サービスにロールを渡すアクセス許可が必要です。

以下の例のエラーは、marymajor という IAM ユーザーがコンソールを使用して Verified Permissions でアクションを実行しようする場合に発生します。ただし、このアクションをサービスが実行するには、サービスロールから付与されたアクセス許可が必要です。Mary には、ロールをサービスに渡すアクセス許可がありません。

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

この場合、Mary のポリシーを更新してメアリーに iam:PassRole アクションの実行を許可する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン資格情報を提供した担当者が管理者です。

自分の 以外のユーザーに Verified Permissions リソース AWS アカウント へのアクセスを許可したい

他のアカウントのユーザーや組織外の人が、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセスコントロールリスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください:

- Verified Permissions がこれらの機能をサポートしているかどうかを確認するには、「Amazon Verified Permissions と の連携方法 IAM」を参照してください。
- 所有 AWS アカウント している のリソースへのアクセスを提供する方法については、「 IAM ユーザーガイド」の「所有 AWS アカウント している別の の IAM ユーザーへのアクセスを提供する」を参照してください。
- リソースへのアクセスをサードパーティーに提供する方法については AWS アカウント、「 IAM ユーザーガイド」の「サードパーティーが所有する へのアクセスを提供する AWS アカウント」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、「IAM ユーザーガイド」の「外部で認証されたユーザー (ID フェデレーション) へのアクセスの許可」を参照してください。
- クロスアカウントアクセスにロールとリソースベースのポリシーを使用する違いについては、「 IAM ユーザーガイド」の「 でのクロスアカウントリソースアクセス IAM」を参照してください。

Amazon Verified Asmission のコンプライアンス検証

AWS のサービス が特定のコンプライアンスプログラムの範囲内にあるかどうかを確認するには、「コンプライアンスAWS のサービス プログラムによるスコープ」の「コンプライアンス」を参照して、関心のあるコンプライアンスプログラムを選択します。一般的な情報については、AWS 「コンプライアンスプログラム」を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、「Downloading Reports in AWS Artifact」を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS のサービス は、お客様のデータの機密性、貴社のコンプライアンス目的、適用される法律および規制によって決まります。を使用する際のコンプライアンス責任の詳細については AWS のサービス、AWS 「 セキュリティドキュメント」を参照してください。

Amazon Verified Permissions

AWS グローバルインフラストラクチャは、 AWS リージョン およびアベイラビリティーゾーンを中心に構築されています。 は、低レイテンシー、高スループット、および高度に冗長なネットワークで接続された、物理的に分離および分離された複数のアベイラビリティーゾーン AWS リージョン を提供します。アベイラビリティーゾーンでは、ゾーン間で中断することなく自動的にフェイルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性が高く、フォールトトレラントで、スケーラブルです。

Verified Permissions ポリシーストアを作成すると、そのポリシーストアは個々の 内に作成され AWS リージョン、そのリージョンのアベイラビリティーゾーンを構成するデータセンター間で自動的にレプリケートされます。現時点では、検証済みアクセス権限はリージョン間のレプリケーションをサポートしていません。

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

Amazon Verified Permissions API コールのモニタリング

モニタリングは、Amazon Verified Permissions およびその他の AWS ソリューションの信頼性、可用性、パフォーマンスを維持する上で重要な部分です。 には、Verified Permissions をモニタリングし、問題が発生したときに報告し、必要に応じて自動アクションを実行するための以下のツール AWS が用意されています。

- AWS CloudTrail は、 AWS アカウントによって、またはアカウントに代わって行われた API コールおよび関連イベントをキャプチャし、指定した Amazon S3 バケットにログファイルを配信します。呼び出し元のユーザーとアカウント AWS、呼び出し元の送信元 IP アドレス、呼び出しの発生日時を特定できます。詳細については、「AWS CloudTrail ユーザーガイド」を参照してください。

CloudTrail を使用した Verified Permissions のモニタリングの詳細については、「」を参照してくださいを使用した Amazon Verified Permissions API コールのログ記録 AWS CloudTrail。

を使用した Amazon Verified Permissions API コールのログ記録 AWS CloudTrail

Amazon Verified Permissions は AWS CloudTrail、Verified Permissions のユーザー、ロール、または のサービスによって実行されたアクションを記録する AWS サービスである と統合されています。CloudTrail は、Verified Permissions のすべての API 呼び出しをイベントとしてキャプチャします。キャプチャされた呼び出しには、Verified Permissions コンソールの呼び出しと、検証済みパーミッション API オペレーションへのコード呼び出しが含まれます。証跡を作成する場合は、Verified Permissions のイベントなど、 Amazon S3 バケットへの CloudTrail イベントの継続的な配信を有効にすることができます。証跡を設定しない場合でも、CloudTrail コンソールで最新の管理アクションイベントをイベント履歴で表示できますが、 などの API コールのイベントは表示できませんisAuthorized。CloudTrail によって収集された情報を使用して、Verified Permissions に対して行われたリクエスト、リクエストの作成元の IP アドレス、リクエストの作成者、リクエストの作成日時、その他の詳細を確認できます。

CloudTrail の詳細については、「AWS CloudTrail ユーザーガイド」を参照してください。

CloudTrail でのVerified Permissions 情報

CloudTrail は、アカウントの作成 AWS アカウント 時に で有効になります。Verified Permissions でアクティビティが発生すると、そのアクティビティはイベント履歴の他の AWS サービスイベントと

ともに CloudTrail イベントに記録されます。最近のイベントは、 AWS アカウントで表示、検索、ダウンロードできます。詳細については、 CloudTrail イベント履歴でのイベントの表示を参照してください。

Verified Permissions のイベントなど AWS アカウント、 のイベントの継続的な記録については、証跡を作成します。証跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、コンソールで証跡を作成するときに、証跡がすべての AWS リージョンに適用されます。証跡は、 AWS パーティション内のすべてのリージョンからのイベントをログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、CloudTrail ログで収集されたイベントデータをさらに分析して処理するように他の AWS サービスを設定できます。詳細については、次を参照してください:

- 追跡を作成するための概要
- CloudTrail がサポートされているサービスと統合
- 「CloudTrail の Amazon SNS 通知の設定」
- 複数のリージョンから CloudTrail ログファイルを受け取るおよび複数のアカウントから CloudTrail ログファイルを受け取る

すべてのVerified Permissionsアクションは、CloudTrail が記録します。これらの説明については、「Amazon Verified Permissions API リファレンスガイド」を参照してください。たとえば、`CreateIdentitySource`、`DeletePolicy`、`ListPolicyStores` の各アクションを呼び出すと、CloudTrail ログファイルにエントリが生成されます。

各イベントまたはログエントリには、リクエストの生成者に関する情報が含まれます。アイデンティティ情報は、以下を判別するのに役立ちます。

- リクエストが root または AWS Identity and Access Management (IAM) ユーザー認証情報を使用して行われたかどうか。
- リクエストがロールまたはフェデレーションユーザーのテンポラリなセキュリティ認証情報を使用して行われたかどうか。
- リクエストが別の AWS サービスによって行われたかどうか。

詳細については、「CloudTrail userIdentity エレメント」を参照してください。

IsAuthorized や IsAuthorizedWithToken などのデータ イベントは、証跡データストアまたはイベントデータストアを作成するときに、デフォルトではログに記録されません。CloudTrail データイベント

を記録するには、アクティビティを収集する、サポート対象のリソースまたはリソースタイプを明示的に追加する必要があります。詳細については、「AWS CloudTrail ユーザーガイド」の「データイベント」を参照してください。

Verified Permissions ログファイルのエントリについて

証跡は、指定した Amazon S3 バケットにイベントをログファイルとして配信できるようにする設定です。CloudTrail のログファイルは、単一か複数のログエントリを含みます。イベントは、任意の出典からの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエストパラメータなどに関する情報が含まれます。CloudTrail ログファイルは、パブリック API 呼び出しの順序付けられたスタックトレースではないため、特定の順序では表示されません。

認可 API コールの場合、決定などのレスポンス要素は、 `additionalEventData`ではなく に含まれます`responseElements`。

トピック

- IsAuthorized
- BatchIsAuthorized
- CreatePolicyStore
- ListPolicyStores
- DeletePolicyStore
- PutSchema
- GetSchema
- CreatePolicyTemplate
- DeletePolicyTemplate
- CreatePolicy
- GetPolicy
- CreateIdentitySource
- GetIdentitySource
- ListIdentitySources
- DeleteIdentitySource

> **Note**
>
> こちらの例では、データのプライバシーを保護するため一部のフィールドが削除されています。

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"
}
```

を使用した Amazon Verified Permissions リソースの作成 AWS CloudFormation

Amazon Verified Permissions は と統合されています。これは AWS CloudFormation、 AWS リソースとインフラストラクチャの作成と管理に費やす時間を短縮できるように、リソースのモデル化とセットアップに役立つサービスです。必要なすべての AWS リソース (ポリシーストアなど) を記述するテンプレートを作成し、それらのリソースを CloudFormation プロビジョニングして設定します。

を使用すると CloudFormation、テンプレートを再利用して Verified Permissions リソースを一貫して繰り返しセットアップできます。リソースを 1 回記述し、複数の AWS アカウント およびリージョンで同じリソースを何度もプロビジョニングします。

> **Important**
>
> Amazon Cognito Identity は、Amazon Verified Permissions AWS リージョン と同じ のすべてで利用できるわけではありません。などの Amazon Cognito Identity CloudFormation に関して からエラーが発生した場合は`Unrecognized resource types: AWS::Cognito::UserPool, AWS::Cognito::UserPoolClient`、Amazon Cognito Identity が利用可能な地理的に最も近い AWS リージョン 場所に Amazon Cognito ユーザープールとクライアントを作成することをお勧めします。Verified Permissions ID ソースを作成するときには、この新しく作成したユーザープールを使用してください。

Verified Permissions と CloudFormation テンプレート

検証済み権限および関連サービスのリソースをプロビジョニングおよび構成するには、CloudFormation テンプレートを理解する必要があります。テンプレートは、JSON や YAML でフォーマットされたテキストファイルです。これらのテンプレートは、 CloudFormation スタックでプロビジョニングするリソースを記述します。JSON または YAML に慣れていない場合は、CloudFormation デザイナー を使用して CloudFormation テンプレートの使用を開始できます。詳細については、 AWS CloudFormation ユーザーガイドのCloudFormation 「デザイナーとは」を参照してください。

Verified Permissions は、アイデンティティソース、ポリシー、ポリシーストア、ポリシーテンプレート、ポリシーストアエイリアスの作成をサポートしています CloudFormation。Verified

Permissions リソースの JSON および YAML テンプレートの例などの詳細については、「AWS CloudFormation ユーザーガイド」の「Amazon Verified Permissions リソースタイプのリファレンス」を参照してください。

AWS CDK コンストラクト

AWS Cloud Development Kit (AWS CDK) は、コードでクラウドインフラストラクチャを定義し、それをプロビジョニングするためのオープンソースのソフトウェア開発フレームワークです CloudFormation。コンストラクトまたは再利用可能なクラウドコンポーネントを使用してテンプレートを作成できます CloudFormation 。その後、これらのテンプレートを使用してクラウドインフラストラクチャをデプロイできます。

詳細と AWS CDK のダウンロードについては、AWS 「 クラウド開発キット」を参照してください。

以下は、コンストラクトなどの Verified Permissions AWS CDK リソースのドキュメントへのリンクです。

- Amazon Verified Permissions L2 CDK コンストラクト

の詳細 CloudFormation

詳細については CloudFormation、次のリソースを参照してください。

- AWS CloudFormation
- AWS CloudFormation ユーザーガイド
- CloudFormation API リファレンス
- AWS CloudFormation コマンドラインインターフェイスユーザーガイド

を使用して Amazon Verified Permissions にアクセスする AWS PrivateLink

を使用して AWS PrivateLink 、VPC と Amazon Verified Permissions の間にプライベート接続を作成できます。インターネットゲートウェイ、NAT デバイス、VPN 接続、または Direct Connect 接続を使用せずに、VPC 内にあるかのように Verified Permissions にアクセスできます。VPC のインスタンスは、パブリック IP アドレスがなくても Verified Permissions にアクセスできます。

このプライベート接続を確立するには、 AWS PrivateLinkを利用したインターフェイスエンドポイントを作成します。インターフェイスエンドポイントに対して有効にする各サブネットにエンドポイントネットワークインターフェイスを作成します。これらは、Verified Permissions 宛てのトラフィックのエントリポイントとして機能するリクエスタ管理型ネットワークインターフェイスです。

詳細については、AWS PrivateLink ガイドのAWS PrivateLinkから AWS のサービス にアクセスするを参照してください。

Verified Permissions に関する考慮事項

Verified Permissions のインターフェイスエンドポイントを設定する前に、「AWS PrivateLink ガイド」の「考慮事項」を確認してください。

Verified Permissions は、インターフェイスエンドポイントを介してすべての API アクションの呼び出しをサポートしています。

Verified Permissions では、VPC エンドポイントポリシーがサポートされません。デフォルトで、エンドポイント経由での Verified Permissions への完全なアクセスが許可されます。または、セキュリティグループをエンドポイントのネットワークインターフェイスに関連付けて、インターフェイスエンドポイントを介して Verified Permissions へのトラフィックを制御することもできます。

Verified Permissions のインターフェイスエンドポイントの作成

Amazon VPC コンソールまたは AWS Command Line Interface (AWS CLI) を使用して、Verified Permissions のインターフェイスエンドポイントを作成できます。詳細については、「AWS PrivateLink ガイド」の「インターフェイスエンドポイントを作成」を参照してください。

以下のサービス名を使用して、 Verified Permissions のインターフェイスエンドポイントを作成します。

```
com.amazonaws.region.verifiedpermissions
```

インターフェイスエンドポイントのプライベート DNS を有効にすると、リージョンのデフォルト DNS 名を使用して、Verified Permissions に API リクエストを実行できます。例えば、verifiedpermissions.us-east-1.amazonaws.com。

インターフェイスエンドポイントのエンドポイントポリシーを作成する

エンドポイントポリシーは、インターフェイスエンドポイントにアタッチできる IAM リソースです。デフォルトのエンドポイントポリシーでは、インターフェイスエンドポイントを介して Verified Permissions へのフルアクセスを許可します。VPC から Verified Permissions に許可されるアクセスを制御するには、インターフェイスエンドポイントにカスタムエンドポイントポリシーをアタッチします。

エンドポイントポリシーは以下の情報を指定します。

- アクションを実行できるプリンシパル (AWS アカウント、IAM ユーザー、IAM ロール)。
- 実行可能なアクション。
- このアクションを実行できるリソース。

詳細については、AWS PrivateLink ガイドのControl access to services using endpoint policies (エンドポイントポリシーを使用してサービスへのアクセスをコントロールする)を参照してください。

例: Verified Permissions アクションの VPC エンドポイントポリシー

以下は、カスタムエンドポイントポリシーの例です。このポリシーをインターフェイスエンドポイントにアタッチすると、すべてのリソースのすべてのプリンシパルに対して、リストされている Verified Permissions アクションへのアクセスが許可されます。

```
{
   "Statement": [
      {
         "Principal": "*",
         "Effect": "Allow",
         "Action": [
            "verifiedpermissions:IsAuthorized",
            "verifiedpermissions:IsAuthorizedWithToken",
```

```
            "verifiedpermissions:GetPolicy"
         ],
         "Resource":"*"
      }
   ]
}
```

Amazon Verified Permissions のクォータ

AWS アカウント には、サービスごとに AWS 、以前は制限と呼ばれていたデフォルトのクォータがあります。特に明記されていない限り、クォータは地域固有です。一部のクォータについては引き上げをリクエストできますが、その他のクォータについては引き上げることはできません。

Verified Permissions のクォータを表示するには、[Service Quotas コンソール] を開きます。ナビゲーションペインで [AWS サービス] を選択し、[Verified Permissions] を選択します。

クォータの引き上げをリクエストするには、Service Quotas ユーザーガイド の「クォータ引き上げリクエスト」を参照してください。Service Quotas でクォータがまだ利用できない場合は、[上限引き上げ] フォームを使用してください。

AWS アカウント には、Verified Permissions に関連する次のクォータがあります。

トピック

リソースのクォータ

名前	デフォルト	引き上げ可能	説明
1 アカウントのリージョンあたりのポリシーストア	サポートされている各リージョン: 30,000	はい	ポリシーストアの最大数。
ポリシーストアあたりのポリシーテンプレート	サポートされている各リージョン: 40	[はい]	ポリシーストア内のポリシーテンプレートの最大数。

名前	デフォルト	引き上げ可能	説明
ポリシーストアあたりの ID ソース	1	[いいえ]	ポリシーストアのために定義できる ID ソースの最大数。
ポリシーストアあたりのポリシーストアエイリアス	10	[Yes (はい)]	1 つのポリシーストアに関連付けることができるポリシーストアエイリアスの最大数。
認可リクエストサイズ[1]	1 MB	[いいえ]	認可リクエストの最大サイズ。
ポリシーサイズ	10,000 バイト	はい	各ポリシーの最大サイズ
スキーマサイズ	100,000 bytes	はい	ポリシーストアのスキーマの最大サイズ。
リソースあたりのポリシーサイズ	200,000 バイト2	はい	特定のリソースを参照するすべてのポリシーの最大サイズ。

1 認可リクエストのクォータは、IsAuthorized と IsAuthorizedWithToken の両方で同じです。

2 1 つのリソースを対象とするすべてのポリシーの合計サイズに対するデフォルトの制限は 200,000 バイトです。同様に、スコープがリソースを未定義のままにしてすべてのリソースに適用するすべてのポリシーの合計サイズは、デフォルトで 200,000 バイトに制限されます。テンプレートにリンクされたポリシーの場合、ポリシーテンプレートのサイズは 1 回のみカウントされ、さらに各テンプレートにリンクされたポリシーをインスタンス化するために使用するパラメータの各セットのサイズ

もカウントされることに注意してください。この制限は、ポリシー設計が特定の制約を満たしていれば引き上げることができます。このオプションを確認する必要がある場合は、 にお問い合わせください サポート。

テンプレートにリンクされたポリシーサイズの例

テンプレートにリンクされたポリシーがリソースクォータあたりのポリシーサイズにどのように寄与するかは、プリンシパルとリソースの長さの合計を取ることで判断できます。プリンシパルまたはリソースが指定されていない場合、その部分の長さは 0 です。リソースが指定されていない場合、そのサイズは"unspecified"リソースクォータにカウントされます。テンプレート本文自体のサイズは、ポリシーのサイズには影響しません。

次のテンプレートを見てみましょう。

```
@id("template1")
permit (
  principal in ?principal,
  action in [Action::"view", Action::"comment"],
  resource in ?resource
)
unless {
  resource.tag =="private"
};
```

そのテンプレートから次のポリシーを作成しましょう。

```
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
}
```

次に、それぞれの principalおよび の文字をカウントして、これらのポリシーのサイズを計算しますresource。各文字は 1 バイトとしてカウントされます。

のサイズは、プリンシパルの長さ User::"alice" (13) にリソースの長さ Photo::"car.jpg" (16) を加えたpolicy1ものになります。これらを追加すると、13 + 16 = 29 バイトになります。

のサイズは、プリンシパルの長さ User::"bob" (11) にリソースの長さ Photo::"boat.jpg" (17) を加えたpolicy2ものになります。これらを追加すると、11 + 17 = 28 バイトになります。

のサイズは、プリンシパルの長さ User::"jane" (12) にリソースの長さ Photo::"car.jpg" (16) を加えたpolicy3ものになります。これらを追加すると、12 + 16 = 28 バイトになります。

のサイズは、プリンシパルの長さ User::"jane" (12) にリソースの長さ (0) を加えたpolicy4ものになります。これらを追加すると、12 + 0 = 12 バイトになります。

policy2 はリソース を参照する唯一のポリシーであるためPhoto::"boat.jpg"、リソースの合計サイズは 28 バイトです。

policy1 と policy3の両方がリソース を参照するためPhoto::"car.jpg"、合計リソースサイズは 29 + 28 = 57 バイトです。

policy4 は"unspecified"リソースを参照する唯一のポリシーであるため、リソースの合計サイズは 12 バイトです。

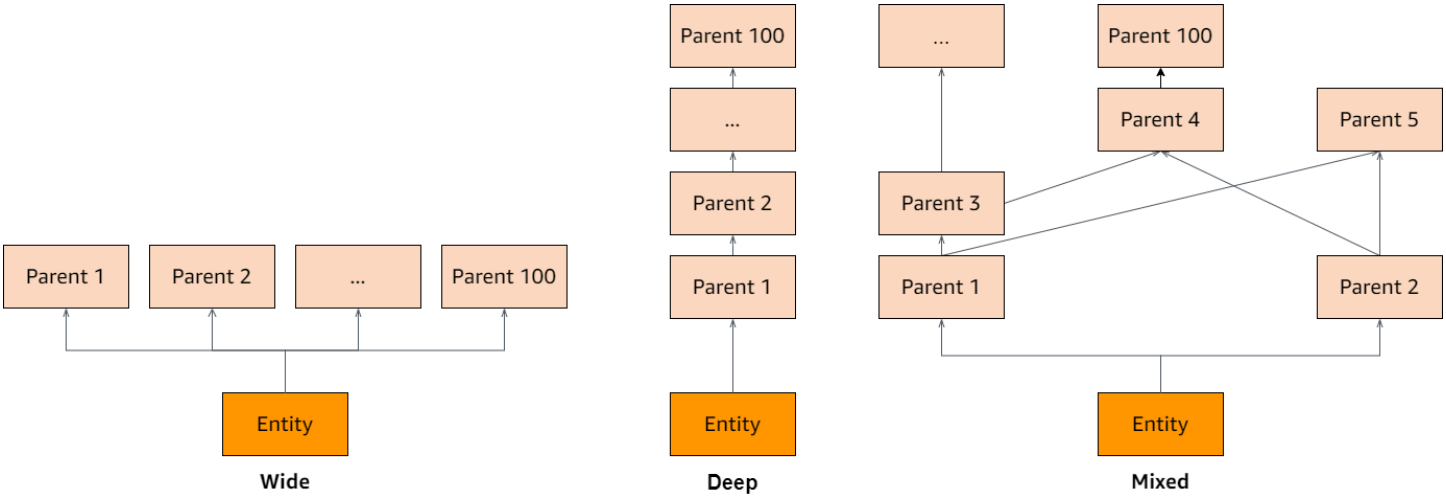
階層のクォータ

> **Note**
>
> 次のクォータは集計されます。つまり、まとめて追加されます。グループの推移的親の最大数は、リストされている数です。例えば、プリンシパルあたりの推移的親の制限が 100 の場

合、アクションとリソースの両方にプリンシパルの親が 100 人、親が 0 人いるか、親の合計が最大 100 人になる可能性があります。

名前	デフォルト	引き上げ可能	説明
プリンシパルあたりの推移的親数	100	[いいえ]	各プリンシパルの推移的親の最大数。
1 アクションあたりの推移的親数	100	[いいえ]	各アクションの推移的親の最大数。
リソースあたりの推移的親数	100	[いいえ]	各リソースの推移的親の最大数。

以下の図は、エンティティ (プリンシパル、アクション、またはリソース) に対して推移的親を定義する方法を示しています。



1 秒あたりのオペレーションのクォータ

Verified Permissions は、アプリケーションリクエスト AWS リージョン が API オペレーションのクォータを超えると、 のサービスエンドポイントへのリクエストを調整します。Verified Permissions は、1 秒あたりのリクエストのクォータを超えた場合、または同時書き込みオペレーションを試みた場合に例外を返すことがあります。現在の RPS クォータは、Service Quotas で表示できます。アプリケーションがオペレーションのクォータを超えないようにするには、再試行と エクスポネンシャルバックオフ用に最適化する必要があります。詳細については、「バックオフパターンによる再試行」および「ワークロードでの API スロットリングの管理とモニタリング」を参照してください。

名前	デフォルト	引き上げ可能	説明
BatchGetPolicy リクエスト/秒/リージョン/ポリシーストア	サポートされている各リージョン: 10	あり	ポリシーストアあたりの 1 秒あたりの BatchGetPolicy リクエストの最大数。
BatchIsAuthorized数	サポートされている各リージョン: 30	あり	ポリシーストアあたりの 1 秒あたりの BatchIsAuthorized リクエストの最大数。
BatchIsAuthorizedWithToken リクエスト/秒/リージョン/ポリシーストア	サポートされている各リージョン: 30	あり	ポリシーストアあたりの 1 秒あたりの BatchIsAuthorizedWithToken リクエストの最大数。
CreateIdentitySource リクエスト/秒/リージョン/ポリシーストア	サポートされている各リージョン: 1	あり	ポリシーストアあたりの 1 秒あたりの CreateIdentitySource リクエストの最大数。

名前	デフォルト	引き上げ可能	説明
ポリシーストアあたりのリージョンごとの 1 秒あたりの CreatePolicy リクエスト数	サポートされている各リージョン: 10	あり	ポリシーストアあたりの 1 秒あたりの CreatePolicy リクエストの最大数。
1 秒、1 リージョン、1 アカウントあたりの CreatePolicyStore リクエスト	サポートされている各リージョン: 1	[いいえ]	CreatePolicyStore リクエストの 1 秒あたりの最大数。
CreatePolicyTemplate リクエスト/秒/リージョン/ポリシーストア	サポートされている各リージョン: 10	あり	ポリシーストアあたりの 1 秒あたりの CreatePolicyTemplate リクエストの最大数。
DeleteIdentitySource リクエスト/秒/リージョン/ポリシーストア	サポートされている各リージョン: 1	あり	ポリシーストアあたりの 1 秒あたりの DeleteIdentitySource リクエストの最大数。
DeletePolicy リクエスト/秒/リージョン/ポリシーストア	サポートされている各リージョン: 10	あり	ポリシーストアあたりの 1 秒あたりの DeletePolicy リクエストの最大数。
1 秒、1 リージョン、1 アカウントあたりの DeletePolicyStore リクエスト	サポートされている各リージョン: 1	[いいえ]	DeletePolicyStore リクエストの 1 秒あたりの最大数。
DeletePolicyTemplate リクエスト/秒/リージョン/ポリシーストア	サポートされている各リージョン: 10	あり	ポリシーストアあたりの 1 秒あたりの DeletePolicyTemplate リクエストの最大数。

名前	デフォルト	引き上げ可能	説明
ポリシーストアあたりのリージョンごとの 1 秒あたりの GetIdentitySource リクエスト数	サポートされている各リージョン: 10	あり	ポリシーストアあたりの 1 秒あたりの GetIdentitySource リクエストの最大数。
ポリシーストアごとのリージョンごとの 1 秒あたりの GetPolicy リクエスト	サポートされている各リージョン: 10	あり	ポリシーストアあたりの 1 秒あたりの GetPolicy リクエストの最大数。
1 秒、1 リージョン、1 アカウントあたりの GetPolicyStore リクエスト数	サポートされている各リージョン: 10	あり	1 秒あたりの GetPolicyStore リクエストの最大数。
ポリシーストアごとのリージョンごとの 1 秒あたりの GetPolicyTemplate リクエスト	サポートされている各リージョン: 10	あり	ポリシーストアあたりの 1 秒あたりの GetPolicyTemplate リクエストの最大数。
ポリシーストアごとのリージョンごとの 1 秒あたりの GetSchema リクエスト	サポートされている各リージョン: 10	あり	ポリシーストアあたりの 1 秒あたりの GetSchema リクエストの最大数。
IsAuthorized リクエスト/秒/リージョン/ポリシーストア	サポートされている各リージョン: 200	あり	ポリシーストアあたりの 1 秒あたりの IsAuthorized リクエストの最大数。
IsAuthorizedWithToken リクエスト/秒/リージョン/ポリシーストア	サポートされている各リージョン: 200	あり	ポリシーストアあたりの 1 秒あたりの IsAuthorizedWithToken リクエストの最大数。

名前	デフォルト	引き上げ可能	説明
ListIdentitySources リクエスト/秒/リージョン/ポリシーストア	サポートされている各リージョン: 10	あり	ポリシーストアあたりの 1 秒あたりの ListIdentitySources リクエストの最大数。
ListPolicies リクエスト/秒/リージョン/ポリシーストア	サポートされている各リージョン: 10	あり	ポリシーストアあたりの 1 秒あたりの ListPolicies リクエストの最大数。
1 秒、1 リージョン、1 アカウントあたりの ListPolicyStores リクエスト	サポートされている各リージョン: 10	あり	1 秒あたりの ListPolicyStores リクエストの最大数。
ListPolicyTemplates リクエスト/秒/リージョン/ポリシーストア	サポートされている各リージョン: 10	あり	ポリシーストアあたりの 1 秒あたりの ListPolicyTemplates リクエストの最大数。
PutSchema リクエスト/秒/リージョン/ポリシーストア	サポートされている各リージョン: 10	あり	ポリシーストアあたりの 1 秒あたりの PutSchema リクエストの最大数。
UpdateIdentitySource リクエスト/秒/リージョン/ポリシーストア	サポートされている各リージョン: 1	あり	ポリシーストアあたりの 1 秒あたりの UpdateIdentitySource リクエストの最大数。
UpdatePolicy リクエスト/秒/リージョン/ポリシーストア	サポートされている各リージョン: 10	あり	ポリシーストアあたりの 1 秒あたりの UpdatePolicy リクエストの最大数。

名前	デフォルト	引き上げ可能	説明
1 秒、1 リージョン、1 アカウントあたりの UpdatePolicyStore リクエスト数	サポートされている各リージョン: 10	いいえ	1 秒あたりの UpdatePolicyStore リクエストの最大数。
UpdatePolicyTemplate リクエスト/秒/リージョン/ポリシーストア	サポートされている各リージョン: 10	あり	ポリシーストアあたりの 1 秒あたりの UpdatePolicyTemplate リクエストの最大数。

Amazon Verified Permissions と Cedar ポリシー言語の用語と概念

Amazon Verified Permissions を使用するには、以下の概念を理解する必要があります。

Verified Permissions の概念

- 認可コード
- 認可リクエスト
- 認可レスポンス
- 考慮済みポリシー
- コンテキストデータ
- 決定ポリシー
- エンティティデータ
- 権限、認可、プリンシパル
- ポリシーの実施
- ポリシーストア
- ポリシーストアエイリアス
- ポリシー名
- ポリシーテンプレート名
- 充足ポリシー
- Amazon Verified Permissions と Cedar ポリシー言語の違い

Cedar ポリシー言語の概念

- 認可
- エンティティ
- グループと階層
- 名前空間
- Policy
- ポリシーテンプレート

- Schema

認可コード

認可モデルは、アプリケーションによって行われた認可リクエストの範囲と、それらのリクエストを評価するための基盤を記述します。さまざまな種類のリソース、それらのリソースで実行されるアクション、およびそれらのアクションを実行するプリンシパルのタイプに基づいて定義されます。また、それらのアクションが実行される背景も考慮されます。

ロールベースのアクセスコントロール (RBAC) は、ロールを定義して一連の権限に関連付ける評価基準です。その後、これらのロールを 1 個またはそれ以上の ID に割り当てることができます。割り当てられた ID には、そのロールに関連する権限が付与されます。ロールに関連付けられている権限が変更されると、その変更はそのロールが割り当てられているすべての ID に自動的に影響します。Cedar はプリンシパルグループを活用することで RBAC の意思決定を支援できます。

属性ベースのアクセスコントロール (ABAC) は、ID に関連する権限をその ID の属性によって決定する評価基準です。Cedar は、プリンシパルの属性を参照するポリシー条件を使用することで、ABAC の決定を支援できます。

Cedar のポリシー言語では、属性ベースの条件を持つユーザーグループに対して権限を定義できるため、RBAC と ABAC を 1 つのポリシーにまとめることができます。

認可リクエスト

認可リクエストは、プリンシパルが特定のコンテキストのリソースに対してアクションを実行できるかどうかを決定するために、一連のポリシーを評価するアプリケーションによる Verified Permissions から行われるリクエストです。

認可レスポンス

認可レスポンスは、認可リクエストへのレスポンスです。これには、許可または拒否の決定に加えて、決定ポリシーの ID などの追加情報が含まれます。

考慮済みポリシー

考慮されるポリシーは、Verified Permissions が承認リクエストを評価する際に含めるために選択するポリシーの完全なセットです。

コンテキストデータ

コンテキストデータは、評価すべき追加情報を提供する属性値です。

決定ポリシー

ポリシーの決定は、認可レスポンスを決定するポリシーです。たとえば、充足ポリシーが2つあり、1 つが拒否、もう 1 つが許容である場合、拒否ポリシーが決定ポリシーになります。充足した許可ポリシーが複数あり、充足した禁止ポリシーが存在しない場合、決定ポリシーが複数存在することになります。当てはまるポリシーがなく、レスポンスが拒否された場合、決定ポリシーは存在しません。

エンティティデータ

エンティティデータは、プリンシパル、アクション、リソースに関するデータです。ポリシー評価に関連するエンティティデータは、エンティティ階層の上位にあるグループメンバーシップと、プリンシパルおよびリソースの属性値です。

権限、認可、プリンシパル

Verified Permissions は、構築するカスタムアプリケーション内できめ細かなアクセス許可と認可を管理します。

プリンシパルとは、ユーザー名やマシン ID などの識別子に結び付けられた ID を持つ、アプリケーションのユーザー (人間か機械かを問わない) です。認証プロセスによって、プリンシパルが本当に本人が主張する ID であるかどうかが決まります。

その ID には、そのプリンシパルがそのアプリケーション内で許可される操作を決定する一連のアプリケーション権限が関連付けられています。認可は、プリンシパルがアプリケーションで特定のアクションを実行することを許可されているかどうかを判断するために、これらのアクセス許可を評価するプロセスです。これらの権限はポリシーとして表現できます。

ポリシーの実施

ポリシー実施とは、Verified Permissions 以外のアプリケーション内で評価決定を強制するプロセスです。Verified Permissions の評価で拒否が返された場合、強制はプリンシパルがリソースにアクセスできないようにします。

ポリシーストア

ポリシーストアはポリシーとテンプレートのコンテナです。各ストアには、ストアに追加されたポリシーを検証するためのスキーマが含まれています。デフォルトでは、各アプリケーションには独自のポリシーストアがありますが、複数のアプリケーションが 1 つのポリシーストアを共有できます。アプリケーションが認可リクエストを行うと、そのリクエストを評価するために使用されるポリシーストアが特定されます。ポリシーストアはポリシーセットを分離する方法を提供するため、マルチテナントアプリケーションで使用して各テナントのスキーマとポリシーを格納できます。1 つのアプリケーションで、テナントごとに別々のポリシーストアを設定できます。

認可リクエストを評価する際、Verified Permissions はリクエストに関連するポリシーストア内のポリシーのサブセットのみを考慮します。関連性はポリシーのスコープに基づいて決定されます。スコープは、ポリシーが適用される特定のプリンシパルとリソース、およびプリンシパルがリソースに対して実行できるアクションを識別します。スコープを定義すると、考慮するポリシーのセットが絞り込まれるため、パフォーマンスの向上に役立ちます。

ポリシーストアエイリアス

ポリシーストアエイリアスは、ポリシーストアのわかりやすい名前です。ポリシーストアエイリアスを使用して、`policyStoreId`パラメータを受け入れる任意の Verified Permissions オペレーションでポリシーストアを識別できます。ポリシーストアエイリアスは、独自の ARN を持つ独立した AWS リソースです。 ARNs 各エイリアスは一度に 1 つのポリシーストアに関連付けられ、複数のエイリアスを同じポリシーストアに関連付けることができます。詳細については、「Amazon Verified Permissions ポリシーストアのエイリアス」を参照してください。

ポリシー名

ポリシー名は、ポリシーのオプションフレンドリ名です。ポリシー名は、ポリシーストア内のすべてのポリシーで一意で、プレフィックスが である必要がありますname/。`policyId` パラメータを受け入れるコントロールプレーンオペレーションでは、ポリシー ID の代わりにポリシー名を使用できます。名前は、ポリシーを作成または更新するときに設定できます。`GetPolicy` と のみ出力の名前`ListPolicies`を返します。

ポリシーテンプレート名

ポリシーテンプレート名は、ポリシーテンプレートのオプションフレンドリ名です。ポリシーテンプレート名は、ポリシーストア内のすべてのポリシーテンプレートで一意で、プレフィックスが であ

る必要がありますname/。policyTemplateId パラメータを受け入れるコントロールプレーンオペレーションでは、ポリシーテンプレート ID の代わりにポリシーテンプレート名を使用できます。名前は、ポリシーテンプレートを作成または更新するときに設定できます。GetPolicyTemplate とのみ出力の名前ListPolicyTemplatesを返します。

充足ポリシー

満たすポリシーは、認可リクエストのパラメータに一致するポリシーです。

Amazon Verified Permissions と Cedar ポリシー言語の違い

Amazon Verified Permissionsは、Cedar ポリシー言語エンジンを使用して認可タスクを実行します。ただし、ネイティブの Cedar 実装と「Verified Permissions」にある Cedar の実装にはいくつか違いがあります。このトピックでは、これらの違いについて説明します。

名前空間の定義

Cedar のVerified Permissions実装は、ネイティブの Cedar 実装と以下の違いがあります。

- Verified Permissionsは、ポリシーストアで定義されたスキーマ内の名前空間1つだけをサポートします。
- Verified Permissions では、空の文字列、または aws、、 amazonの値を含む名前空間を作成することはできませんcedar。

ポリシーテンプレートのサポート

Verified Permissions と Cedar は両方とも、principal と resource のスコープ内でのみプレースホルダーを許可します。ただし、Verified Permissions では、principal と resource のどちらも制約されていないことも必要です。

以下のポリシーは Cedar では有効ですが、principalには制約がないためVerified Permissionsでは拒否されます。

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

principalとresourceの両方に制約があるため、以下の例はいずれも Cedar と Verified Permissionsの両方で有効です。

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

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

スキーマのサポート

Verified Permissions では、すべてのスキーマ JSON キー名が空でない文字列である必要があります。Cedar では、プロパティや名前空間など、場合によっては空の文字列を使用できます。

アクショングループの定義

Cedar の認可方法では、認可リクエストをポリシーと照らし合わせて評価する際に考慮すべきエンティティのリストが必要です。

アプリケーションが使用するアクションとアクショングループをスキーマで定義できます。ただし、Cedar は評価リクエストにスキーマを含めません。代わりに、Cedar は送信したポリシーとポリシーテンプレートの検証にのみスキーマを使用します。Cedar は評価リクエスト時にスキーマを参照しないため、スキーマにアクショングループを定義した場合でも、承認 API オペレーションに渡す必要があるエンティティリストの一部として、アクショングループのリストも含める必要があります。

これはVerified Permissions によって自動的に行われます。スキーマで定義したアクショングループは、`IsAuthorized`または`IsAuthorizedWithToken`オペレーションのパラメータとして渡したエンティティリストに自動的に追加されます。

エンティティフォーマット

`entityList` パラメータを使用した Verified Permissions のエンティティの JSON 形式は、以下の点で Cedar とは異なります。

- Verified Permissions では、JSON オブジェクトのすべてのキーと値のペアが、という名前の JSON オブジェクトでラップされている必要があります。`Record`
- Verified Permissions の JSON リストは、キー名が `Set` で値が Cedar の元の JSON リストである JSON キーと値のペアでラップする必要があります。
- `String`、`Long`、および `Boolean` タイプ名の場合、Cedar の各キーと値のペアは、Verified permissions の JSON オブジェクトに置き換えられます。オブジェクトの名前は元のキー名です。JSON オブジェクト内には、キーと値のペアが 1 つあり、キー名はスカラー値 (`String`、`Long`、または`Boolean`)の型名 で、値は Cedar エンティティの値です。

- Cedar エンティティと Verified Permissions エンティティの構文フォーマットは、以下の点で異なります。

Cedar フォーマット	Verified Permissions フォーマット
uid	Identifier
type	EntityType
id	EntityId
attrs	Attributes
parents	Parents

Example- リスト

次の例は、エンティティのリストをそれぞれ Cedar と Verified Permissions で表現する方法を示しています。

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- ポリシー評価

次の例は、Cedar および Verified Permissions の認可リクエストでポリシーを評価するためにエンティティがどのようにフォーマットされるかを示しています。

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": []
    }
]
```

長さとサイズの制限

Verified Permissionsは、スキーマ、ポリシー、ポリシーテンプレートを格納するためのポリシーストア形式のストレージをサポートします。このストレージが原因で、認証済みアクセス権限には Cedar に関係のない長さとサイズの制限が課されます。

オブジェクト	Verified Permissionsの制限 (バイト単位)	シダー制限
ポリシーのサイズ [1]	10,000	なし
インラインポリシーの説明	150	Cedar には適用されません。
ポリシーテンプレートサイズ	10,000	なし
スキーマサイズ	100,000	なし
エンティティタイプ	200	なし
ポリシー ID	64	なし

オブジェクト	Verified Permissionsの制限 (バイト単位)	シダー制限
ポリシーテンプレート ID	64	なし
エンティティ ID	200	なし
ポリシーストア ID	64	Cedarには適用されません。

[1] Verified Permissions では、ポリシーストアで作成されたポリシーのプリンシパル、アクション、リソースの合計サイズに基づいて、ポリシーストアごとのポリシーに制限があります。1 つのリソースに関連するすべてのポリシーの合計サイズは 200,000 バイトを超えることはできません。テンプレートにリンクされたポリシーの場合、ポリシーテンプレートのサイズに、テンプレートにリンクされた各ポリシーのインスタンス化に使用される各パラメータセットのサイズを加えたものが 1 回だけカウントされます。

Amazon Verified Permissions から Cedar 4 へのアップグレードに関するよくある質問

Amazon Verified Permissions は、使用する Cedar のバージョンをバージョン 2 からバージョン 4 にアップグレードしています。Cedar は、ポリシーストアでポリシー、ポリシーテンプレート、スキーマを記述するために使用するオープンソース言語です。Verified Permissions での Cedar 4 のサポートにより、`is`演算子タグやエンティティタグなどの新機能を使用して、より表現力の高いポリシーを記述できます。

Amazon Verified Permissions は、ポリシーストアを Cedar 4 に自動的にアップグレードします。ただし、Cedar 2 用に記述された一部のポリシー、スキーマ、認可リクエストは、Cedar 4 と互換性がありません。ポリシーストアの場合は、自動的にアップグレードされません。Cedar 4 にアップグレードする前に、ポリシー、ポリシーテンプレート、スキーマ、またはアプリケーションコードを変更する必要がある場合があります。

トピック

Amazon Verified Permissions を呼び出すときに Cedar 2 がサポートされなくなったというエラーが表示されるのはなぜですか?

2026 年 4 月以降、Cedar 2 を使用するポリシーストアでは認可 APIs (`IsAuthorized`、`BatchIsAuthorized`、`IsAuthorizedWithToken`および

BatchIsAuthorizedWithToken) が無効になっています。この場合、次のエラーメッセージが返されます。

```
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.
```

Verified Permissions を呼び出すときにこのエラーメッセージが表示された場合は、ポリシーストアが Cedar 4 と互換性がないため、 を新しいバージョンに移行できなかったことを意味します。Amazon Verified Permissions を引き続き使用するには、Cedar 4 を使用する新しいポリシーストアを作成するか、 にお問い合わせください サポート。

一部のポリシー、ポリシーテンプレート、スキーマが Cedar 4 と互換性がないのはなぜですか?

Cedar チームは、バグを修正し、言語を簡素化するために、Cedar 2 以降、下位互換性のないいくつかの変更を行いました。これらの変更には次の場合が含まれます。

- ポリシー、ポリシーテンプレート、スキーマの構文の変更
- より多くのエラーを検出する、より正確なポリシー検証ツール
- などの組み込み関数の動作の変更 isInRange

下位互換性のない変更の完全なリストについては、Cedar 変更ログ(*)で とマークされた項目を探します。

ポリシーストアで Cedar 2 と Cedar 4 のどちらを使用しているかを確認するにはどうすればよいですか?

ポリシーストアが使用する Cedar のバージョンは、Amazon Verified Permissions コンソールまたは GetPolicyStoreオペレーションを使用して確認できます。

> **Note**
>
> 同じ AWS アカウント とリージョンのすべてのポリシーストアは、同じバージョンの Cedar を使用します。

Console

Cedar バージョンのポリシーストアを確認するには (コンソール)

1. にサインイン AWS マネジメントコンソール し、https://console.aws.amazon.com/verifiedpermissions/ で Amazon Verified Permissions コンソールを開きます。
2. ナビゲーションペインから、ポリシーストアを選択し、確認するポリシーストアを選択します。
3. ナビゲーションペインで [設定] を選択します。
4. 詳細ボックスで、Cedar バージョンフィールドを見つけます。

フィールドは、ポリシーストアが Cedar 2 を使用している場合と、Cedar 4 `CEDAR_4`を使用している場合に読み取り`CEDAR_2`ます。

CLI

Cedar バージョンのポリシーストアを確認するには (AWS CLI)

1. まだインストールしていない場合は、 AWS Command Line Interface (AWS CLI) をインストールして設定します。詳細については、「AWS CLIの最新バージョンをインストールまたは更新します。」を参照してください。
2. `get-policy-store` コマンドを使用します。次の例では、*`policy-store-id`* をポリシーストアの識別子に置き換えます。

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

出力の `cedarVersion`フィールドには、ポリシーストアが使用している Cedar のバージョンが表示されます。例えば、次のようになります。

```
{
    "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"
}
```

フィールドは、ポリシーストアが Cedar 2 を使用している場合と、Cedar 4 `CEDAR_4`を使用している場合に読み取り`CEDAR_2`ます。

Cedar 4 にアップグレードするにはどうすればよいですか?

Amazon Verified Permissions では、ほとんどのお客様が Cedar 4 にアップグレードされています。ポリシーストアを作成したことがない場合、作成する新しいポリシーストアは Cedar 4 を使用します。既存のお客様の場合は、すでに Cedar 4 にアップグレードされている可能性があります。ポリシーストアが使用している Cedar のバージョンを確認するには[ポリシーストアで Cedar 2 と Cedar 4 のどちらを使用しているかを確認するにはどうすればよいですか?](#)、「」を参照してください。

アップグレードされていない場合、Verified Permissions は、Cedar 4 と互換性のないポリシーストアの 1 つでポリシー、ポリシーテンプレート、スキーマ、または認可リクエストを検出しました。2025 年後半に、互換性のないリソースについて説明する E メール通知が送信されます。より早くアップグレードするには、 でケースを開きます サポート。

> **Important**
>
> 同じ のすべてのポリシーストアは、同じバージョンの Cedar AWS アカウント を使用します。アカウント内の 1 つのポリシーストアが Cedar 4 と互換性がない場合、そのアカウントのどのポリシーストアでも Cedar 4 を使用することはできません。

ポリシーストアを Cedar 4 から Cedar 2 にダウングレードできますか?

いいえ。ポリシーストアを Cedar 4 にアップグレードした後に問題が発生した場合は、 でケースを開きます サポート。

ポリシーストアが Cedar 2 に設定されているというエラーメッセージが表示されるのはなぜですか?

Amazon Verified Permissions の一部の機能は、Cedar 4 の新機能に依存しています。ポリシーストアで Cedar 4 が使用されていない場合、次の API フィールドは使用できません。

- IsAuthorized、BatchIsAuthorized、IsAuthorizedWithTokenおよび BatchIsAuthorizedWithTokenオペレーションの場合:
  - `datetime`、 `decimal` または `attributes`contextフィールドdurationの値

ポリシーストアがアップグレードされるまで、Cedar 2 の後に導入されたポリシー、ポリシーテンプレート、スキーマで構文またはデータ型を使用することはできません。

スキーマを Cedar 4 と互換性を持たせるにはどうすればよいですか?

Verified Permissions コンソールは、スキーマの互換性の問題を自動的に修正できます。スキーマを自動的に修正できない場合、コンソールに手動で修正するエラーのリストが表示されます。

> **Important**
>
> Amazon Verified Permissions コンソールのコードエディタには、ポリシーストアで Cedar 2 を使用している場合でも、常に Cedar 4 からのエラーと警告が表示されます。変更の保存ボタンまたは Verified Permissions API を使用して、Cedar 4 と互換性のないスキーマを引き続き更新できます。

コンソールを使用してスキーマを修正するには

1. にサインイン AWS マネジメントコンソール し、検証済みのアクセス許可で Amazon Verified Permissions コンソールを開きます。
2. ナビゲーションペインで、ポリシーストアを選択し、確認するポリシーストアを選択します。
3. ナビゲーションペインでスキーマを選択します。
4. スキーマを自動的に修正できる場合は、「「修正」をクリックして互換性のあるバージョンをプレビューする」というバナーが表示されます。修正を選択します。

5. スキーマに加えられた変更を確認し、更新されたスキーマのプレビューをクリックします。
6. 更新されたスキーマを確認し、変更の保存をクリックします。

スキーマを自動的に修正できない場合は、 コンソールで修正するエラーのリストを確認できます。

1. 上記の説明に従って、スキーマの編集ページを開きます。
2. JSON モードを選択します。
3. コードエディタの左側にあるガーターの赤いエラーアイコンにカーソルを合わせます。エラーメッセージがツールヒントに表示されます。

発生する可能性のある一般的なエラーとその解決方法は次のとおりです。

が JSON からスキーマを解析できませんでした: `` `field-name` ``

Cedar 2 では、Cedar スキーマの一部として意味がない場合でも、型定義などのスキーマの一部に任意のフィールドを含めることができます。Cedar 4 では、これは許可されなくなりました。このエラーを解決するには、JSON スキーマから *field-name* というフィールドを削除します。有効なスキーマフィールドのリストについては、Cedar のドキュメントを参照してください。

不明な拡張タイプ `` `extension-name` ``

Cedar 2 では、 `type`が である属性を宣言するときに`Extension`、値が有効な拡張タイプ名であるかどうかにかかわらず、 `name`フィールドに任意の値を指定できます。これは Cedar 4 のエラーになりました。これを解決するには、*extension-name* を有効な拡張タイプ名に置き換えます。有効な拡張タイプ名のリストは、Cedar ドキュメントで確認できます。

スキーマのエラーを解決する方法が不明な場合は、 にお問い合わせください。 サポート

ポリシーとテンプレートを Cedar 4 と互換性を持たせるにはどうすればよいですか?

Verified Permissions コンソールには、Cedar 4 と互換性のないポリシーまたはテンプレートのエラーが表示されます。

コンソールでポリシーまたはテンプレートのエラーを表示するには

1. にサインイン AWS マネジメントコンソール し、検証済みのアクセス許可で Amazon Verified Permissions コンソールを開きます。

2. ナビゲーションペインから、ポリシーストアを選択し、確認するポリシーストアを選択します。
3. 必要に応じて、ナビゲーションペインでポリシーまたはポリシーテンプレートを選択します。
4. 互換性のないポリシーまたはテンプレートを選択します。
5. [編集] を選択します。
6. コードエディタの左側にあるガーターの赤いエラーアイコンにカーソルを合わせます。エラーメッセージがツールヒントに表示されます。

発生する可能性のある一般的なエラーとその解決方法は次のとおりです。

空のセットリテラルは ポリシーで禁止されています

Cedar 2 では、 構文を使用して、セットが空かどうか`mySet == []`を確認できます。Cedar 4 では、この構文を使用するポリシーはスキーマに対して検証されなくなりました。ポリシー`mySet == []`の を に置き換えます`mySet.isEmpty()`。

Amazon Verified Permissions ユーザーガイドのドキュメント履歴

次の表に、Verified Permissions のドキュメントリリースを示します。

変更	説明	日付
ポリシーストアエイリアスのハード削除	24 時間PendingDeletion の予約期間をバイパスして、ポリシーストアエイリアスをハード削除して即時削除できるようになりました。	2026 年 4 月 17 日
ポリシー名とポリシーテンプレート名	ポリシーとポリシーテンプレートに名前を割り当てて、フレンドリ名で参照できるようになりました。	2025 年 3 月 4 日
ポリシーストアのエイリアス	ポリシーストアエイリアスを作成して、フレンドリ名でポリシーストアを参照できるようになりました。	2025 年 2 月 26 日
新しい AWS 管理ポリシー	Verified Permissions で AmazonVerifiedPermissionsFullAccess および AmazonVerifiedPermissionsReadOnlyAccess IAM 管理ポリシーを使用できるようになりました。	2024 年 10 月 11 日
OIDC ID ソース	OpenID Connect (OIDC) ID プロバイダーからユーザーを承認できるようになりました。	2024 年 6 月 8 日

ID ソーストークンを使用したバッチ認可	1 つの BatchIsAuthorizedWithToken API リクエストでユーザープールから Amazon Cognito ユーザーを承認できるようになりました。	2024 年 4 月 5 日
を使用したポリシーストアの作成 API Gateway	既存の API と Amazon Cognito ユーザープールからポリシーストアを作成できるようになりました。	2024 年 4 月 1 日
コンテキストの概念と例	Verified Permissions を使用した認可リクエストのコンテキストに関する情報を追加しました。	2024 年 2 月 1 日
認可の概念と例	Verified Permissions を使用した認可リクエストに関する情報を追加しました。	2024 年 2 月 1 日
AWS CloudFormation 統合	Verified Permissions は、 での ID ソース、ポリシー、ポリシーストア、ポリシーテンプレートの作成をサポートしています CloudFormation。	2023 年 6 月 30 日
初回リリース	Amazon Verified Permissions ユーザーガイドの初回リリース	2023 年 6 月 13 日

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