



ユーザーガイド

Amazon Verified Permissions



Amazon Verified Permissions: ユーザーガイド

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon の商標およびトレードドレスは、お客様に混乱を招く可能性がある態様、または Amazon の信用を傷つけたり、失わせたりする態様において、Amazon のものではない製品またはサービスに関連して使用してはなりません。Amazon が所有しない他の商標はすべてそれぞれの所有者に帰属します。所有者は必ずしも Amazon との提携や関連があるわけではありません。また、Amazon の支援を受けているとはかぎりません。

Table of Contents

Amazon Verified Permissions とは何でしょうか？	1
Amazon Verified Permissions による認証	1
Cedar ポリシー言語	1
Verified Permissions の利点	2
アプリケーションの開発を加速	2
より安全なアプリケーション	2
エンドユーザー機能	2
関連サービス	2
Verified Permissions へのアクセス	3
Verified Permissions の料金	4
規約と概念	5
認証コード	5
認証リクエスト	6
認証レスポンス	6
考慮済みポリシー	6
コンテキストデータ	6
決定ポリシー	7
エンティティデータ	7
権限、認証、プリンシパル	7
ポリシーの実施	7
ポリシーストア	7
充足ポリシー	8
シダーとの違い	8
名前空間の定義	8
ポリシーテンプレートのサポート	8
スキーマのサポート	9
拡張タイプのサポート	9
エンティティ用の Cedar JSON 形式	9
アクショングループの定義	9
長さやサイズの制限	10
開始	12
AWS アカウントへのサインアップ	12
管理ユーザーの作成	13
Verified Permissions の IAM ポリシー	14

最初のポリシーストアを作成する	15
サンプルポリシーストアの作成	15
サンプルポリシーストア用のテンプレートリンクポリシーの作成	17
サンプルポリシーストアのテスト	18
ポリシーストア	21
ポリシーストアの作成	21
ポリシーストアの切り替え	26
ポリシーストアの削除	26
ポリシーストアスキーマ	28
スキーマの編集-ビジュアル	30
スキーマの編集 - JSON	32
スキーマの削除	32
ポリシーの検証モード	34
ポリシー	36
エンティティフォーマット	36
静的ポリシーの作成	41
静的ポリシーの編集	43
ポリシーの表示	45
ポリシーの例	48
個々のエンティティへのアクセスを許可します。	48
エンティティのグループへのアクセスを許可します。	48
どのエンティティにもアクセスを許可します。	50
エンティティ (ABAC) の属性へのアクセスを許可します。	50
アクセスを拒否する	53
ポリシーテンプレート	55
テンプレートの作成	55
テンプレートにリンクされたポリシーの作成	56
ポリシーテンプレートの編集	59
サンプルポリシーストア用のテンプレートリンクポリシーの例	60
PhotoFlash テンプレートにリンクされたポリシーの例	60
DigitalPetStore	62
TinyToDo テンプレートにリンクされたポリシーの例	62
ID プロバイダ	63
Amazon Cognito とアイデンティティソースとの連携	63
他の ID プロバイダーの使用	64
ID ソースの作成	67

ID ソースの編集	69
Amazon Cognito トークンをスキーマにマッピングする	71
アクセストークンをマッピングする	74
承認モデルの設計	77
正しいモデルは1つ也没有	78
リソースに焦点を当てる	78
複合許可	80
マルチテナンシーの検討	81
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	82
選択方法	83
ポリシー範囲にデータを入力します。	84
すべてのリソースをコンテナに入れる。	85
プリンシパルとリソースを分離	86
属性には権限を埋め込まないでください。	89
きめ細かな許可	91
認証を問い合わせるその他の理由	92
テストベンチ	93
認証	96
API オペレーション	96
API テスト	97
アプリケーションとの統合	99
.....	102
サンプルコンテキストを評価する	104
セキュリティ	110
データ保護	110
データ暗号化	112
アイデンティティおよびアクセス管理	112
対象者	113
アイデンティティによる認証	113
ポリシーを使用したアクセス権の管理	116
Amazon Verified Permissions と IAM の連携方法	119
アイデンティティベースポリシーの例	126
トラブルシューティング	129
コンプライアンス検証	131
耐障害性	132
モニタリング	133

CloudTrail ログ	133
での検証済みアクセス許可情報 CloudTrail	133
Verified Permissions ログファイルのエントリについて	135
AWS CloudFormation リソース	152
Verified Permissions およびAWS CloudFormationテンプレート	152
AWS CloudFormation の詳細はこちら	153
AWS PrivateLink	154
考慮事項	154
インターフェースエンドポイントの作成	154
クォータ	156
リソースのクォータ	156
階層のクォータ	158
1 秒あたりのオペレーションのクォータ	158
ドキュメント履歴	162
.....	clxiii

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証サービスです。Verified Permissions を利用すると、認証を外部化し、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、アプリケーションユーザーの権限をきめ細かく定義します。

トピック

- [Amazon Verified Permissions による認証](#)
- [Cedar ポリシー言語](#)
- [Verified Permissions の利点](#)
- [関連サービス](#)
- [Verified Permissions へのアクセス](#)
- [Verified Permissions の料金](#)

Amazon Verified Permissions による認証

Verified Permissions は、プリンシパルがカスタムアプリケーションの特定のコンテキスト内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検証することで認証を行います。認証済みアクセス権限は、プリンシパルが OpenID Connect などのプロトコル、Amazon Cognito 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、他の手段で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、ユーザーがどこで管理され、どのように認証されたかに依存しません。

Verified Permissions は、顧客が AWS Management Console でポリシーを作成、管理、テストできるようにするサービスです。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証 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 の認証ポリシーは Cedar ポリシー言語を使用して記述されます。Cedar は、認証ポリシーを記述し、そのポリシーに基づいて認証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、許可されたユーザーだけがアプリケーションにアクセスできるよう

にし、各ユーザーが許可されていることだけ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 認証エンジン呼び出し、「このリクエストは認証されていますか?」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 2.4 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

認証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。

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、ユーザーディレクトリや認証サーバーであり、さらには、OAuth 2.0 アクセストークンと AWS 認証情報の認証サービスです。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、API を作成、公開、維持、モニタリング、保護するための AWS のサービスです。WebSocket APIs 詳細については、「[API Gateway デベロッパーガイド](#)」を参照してください。
- AWS IAM Identity Center— 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 アイデン

アイティセンターでは、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アカウントやアプリケーションにわたるアクセスを一元管理できます。詳細については、『[AWS IAM Identity Center 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 Management Console

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することで、Verified Permissions および 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 [AWS Tools for Windows PowerShell](#) という 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 CLI のインストールおよび使用の方法については、[AWS Command Line Interface ユーザーガイド](#)を参照してください。Tools for Windows のインストールと使用については PowerShell、[AWS Tools for Windows PowerShell ユーザーガイド](#)」を参照してください。

- 「[コマンドリファレンス](#)」の「[検証済みアクセス許可](#) AWS CLI」
- での [Amazon Verified Permissions](#) AWS Tools for Windows PowerShell

AWS SDK

AWS には、さまざまなプログラミング言語およびプラットフォーム (Java、Python、Ruby、.NET、iOS、Android など) のライブラリとサンプルコードで構成された SDK (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 「[のツール 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](#)

Verified Permissions API

サービスに HTTPS リクエストを直接発行できる HTTPS API を使用して、プログラムにより Verified Permissions と AWS にアクセスできます。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 か月あたりの認証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の請求、アカウント、イベントについてのご質問は、[AWS Supportにお問い合わせください](#)。

Amazon Verified Permissions Permissions 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

- [認証コード](#)
- [認証リクエスト](#)
- [認証レスポンス](#)
- [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、認証、プリンシパル](#)
- [ポリシーの実施](#)
- [ポリシーストア](#)
- [充足ポリシー](#)
- [Verified Permissions と Cedar の違い](#)

Cedar ポリシー言語の概念

- [認証](#)
- [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](#)

認証コード

承認モデルには、アプリケーションが行う[認証リクエスト](#)のスコープと、それらの要求を評価するための基礎が記述されています。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

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。

認証リクエスト

認証リクエストとは、プリンシパルが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判断するために、アプリケーションが一連のポリシーを評価するために行う検証済み権限のリクエストです。

認証レスポンス

認証レスポンスは、[認証リクエストに対するレスポンス](#)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

考慮済みポリシーとは、Verified Permissions によって[認証リクエスト](#)を評価する際に選択される完全なポリシーのセットです。

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

決定ポリシーは、[認証レスポンス](#)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、認証、プリンシパル

Verified Permissions は、構築したカスタムアプリケーション内のきめ細かい権限と認証を管理します。

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。認証とは、これらの権限を評価して、プリンシパルがアプリケーション内で特定のアクションを実行できるかどうかを判断するプロセスです。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証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。

[認証リクエスト](#)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。

充足ポリシー

充足ポリシーとは、[認証リクエスト](#)のパラメータと一致するポリシーです。

Verified Permissions と Cedar の違い

Amazon Verified Permissionsは、Cedar ポリシー言語エンジンを使用して認証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の違いがあります。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、aws、amazon、または cedar の値を使用して[名前空間](#)を作成することはできません。

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、プロパティなどいくつかのケースで空の文字列を使用できます。

拡張タイプのサポート

Verified Permissions はポリシーで Cedar [拡張タイプ](#)をサポートしていますが、スキーマの定義に、または `IsAuthorized` および `IsAuthorizedWithToken` オペレーションの `entities` パラメーターの一部としてそれらを含めることは現在サポートされていません。

拡張タイプには、固定小数点 ([decimal](#)) データ型と IP アドレス ([ipaddr](#)) データ型があります。

エンティティ用の Cedar JSON 形式

現時点では、Verified Permissions では、[EntityItem](#) 要素の配列である [EntitiesDefinition](#) に定義された構造を使用して、承認リクエストで考慮されるエンティティのリストを渡す必要があります。[Verified Permissionsは、現在のところ、承認リクエストで考慮すべきエンティティのリストを Cedar JSON 形式で渡すことをサポートしていません。](#) Verified Permissions で使用するエンティティのフォーマットに関する具体的な要件については、[Amazon Verified Permissions のエンティティフォーマット](#)を参照してください。

アクショングループの定義

Cedar の認証方法では、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。

長さやサイズの制限

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 に関係のない長さや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	10,000	なし
エンティティタイプ	200	なし
ポリシー ID	64	なし
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	シダー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。

Verified Permissions の開始方法

このチュートリアルを使用して、Amazon Verified Permissions の使用を開始できます。

トピック

- [AWS アカウントへのサインアップ](#)
- [管理ユーザーの作成](#)
- [Verified Permissions のIAMポリシー](#)
- [初めてのVerified Permissions ポリシーストアを作成する](#)

AWS アカウントへのサインアップ

AWS アカウントがない場合は、以下のステップを実行して作成します。

AWS アカウント にサインアップするには

1. <https://portal.aws.amazon.com/billing/signup> を開きます。
2. オンラインの手順に従います。

サインアップ手順の一環として、通話呼び出しを受け取り、電話のキーパッドを使用して検証コードを入力するように求められます。

AWS アカウントにサインアップすると、AWS アカウントのルートユーザーが作成されます。ルートユーザーには、アカウントのすべての AWS のサービスとリソースへのアクセス権があります。セキュリティのベストプラクティスとして、[管理ユーザーに管理アクセスを割り当て](#)、[ルートユーザーアクセスが必要なタスク](#)を実行する場合にのみ、ルートユーザーを使用してください。

サインアップ処理が完了すると、AWS からユーザーに確認メールが送信されます。<https://aws.amazon.com/> の [アカウント] をクリックして、いつでもアカウントの現在のアクティビティを表示し、アカウントを管理することができます。

管理ユーザーの作成

AWS アカウント にサインアップしたら、AWS アカウントのルートユーザー をセキュリティで保護し、AWS IAM Identity Center を有効にして、管理ユーザーを作成します。これにより、日常的なタスクにルートユーザーを使用しないようにします。

AWS アカウントのルートユーザーをセキュリティで保護する

1. [ルートユーザー] を選択し、AWS アカウント のメールアドレスを入力して、アカウント所有者として [AWS Management Console](#) にサインインします。次のページでパスワードを入力します。

ルートユーザーを使用してサインインする方法については、「AWS サインイン User Guide」の「[Signing in as the root user](#)」を参照してください。

2. ルートユーザーの多要素認証 (MFA) を有効にします。

手順については、IAMユーザーガイドの「[AWS アカウント のルートユーザーの仮想 MFA デバイスを有効にする \(コンソール\)](#)」を参照してください。

管理ユーザーを作成する

1. IAM アイデンティティセンターを有効にします。

手順については、「AWS IAM Identity Center ユーザーガイド」の「[AWS IAM Identity Center の有効化](#)」を参照してください。

2. IAM アイデンティティセンターで、管理ユーザーに管理者アクセスを付与します。

IAM アイデンティティセンターディレクトリ をアイデンティティソースとして使用するチュートリアルについては、「AWS IAM Identity Center ユーザーガイド」の「[デフォルトの IAM アイデンティティセンターディレクトリ でユーザーアクセスを設定する](#)」を参照してください。

管理ユーザーとしてサインインする

- IAM アイデンティティセンターのユーザーとしてサインインするには、IAM アイデンティティセンターのユーザーの作成時に E メールアドレスに送信されたサインイン URL を使用します。

IAM アイデンティティセンターのユーザーを使用してサインインする方法については、「AWS サインイン User Guide」の「[Signing in to the AWS access portal](#)」を参照してください。

Verified Permissions のIAM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 API を呼び出すか、AWS Management Console 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 権限を追加する必要があります。

アイデンティティベースポリシーは、IAM ユーザー、ユーザーのグループ、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、および条件を制御します。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 ポリシーを作成](#)」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、許可または拒否するアクションとリソース、また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。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 ポリシーで使用するすべての要素については、IAMユーザーガイドの「[IAMJSON ポリシーエレメントのリファレンス](#)」を参照してください。

[アクション]	説明
CreatePolicyStore	新しいポリシーストアを作成するアクション。
DeletePolicyStore	ポリシーストアを削除するアクション。
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
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。
DeletePolicy	ポリシーストアからポリシーを削除するアクション。
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。

[アクション]	説明
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。
IsAuthorized	認証リクエスト に記述されたパラメータに基づいて 認証レスポンス を取得するアクション。

CreatePolicy アクションに対するアクセス許可のIAMポリシーの例：

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

初めてのVerified Permissions ポリシーストアを作成する

Verified Permissions コンソールに初めてサインインするときに、最初の[ポリシーストア](#)と Cedar ポリシーを作成する方法を選択できます。AWSサインインユーザーガイドの「[AWS へのサインイン方法](#)」のトピックで説明されているように、ユーザータイプに適したサインイン手順に従ってください。コンソールのホームページで、Amazon Verified Permissions サービスを選択します。[使用を開始]を選択します。

サンプルポリシーストアの作成

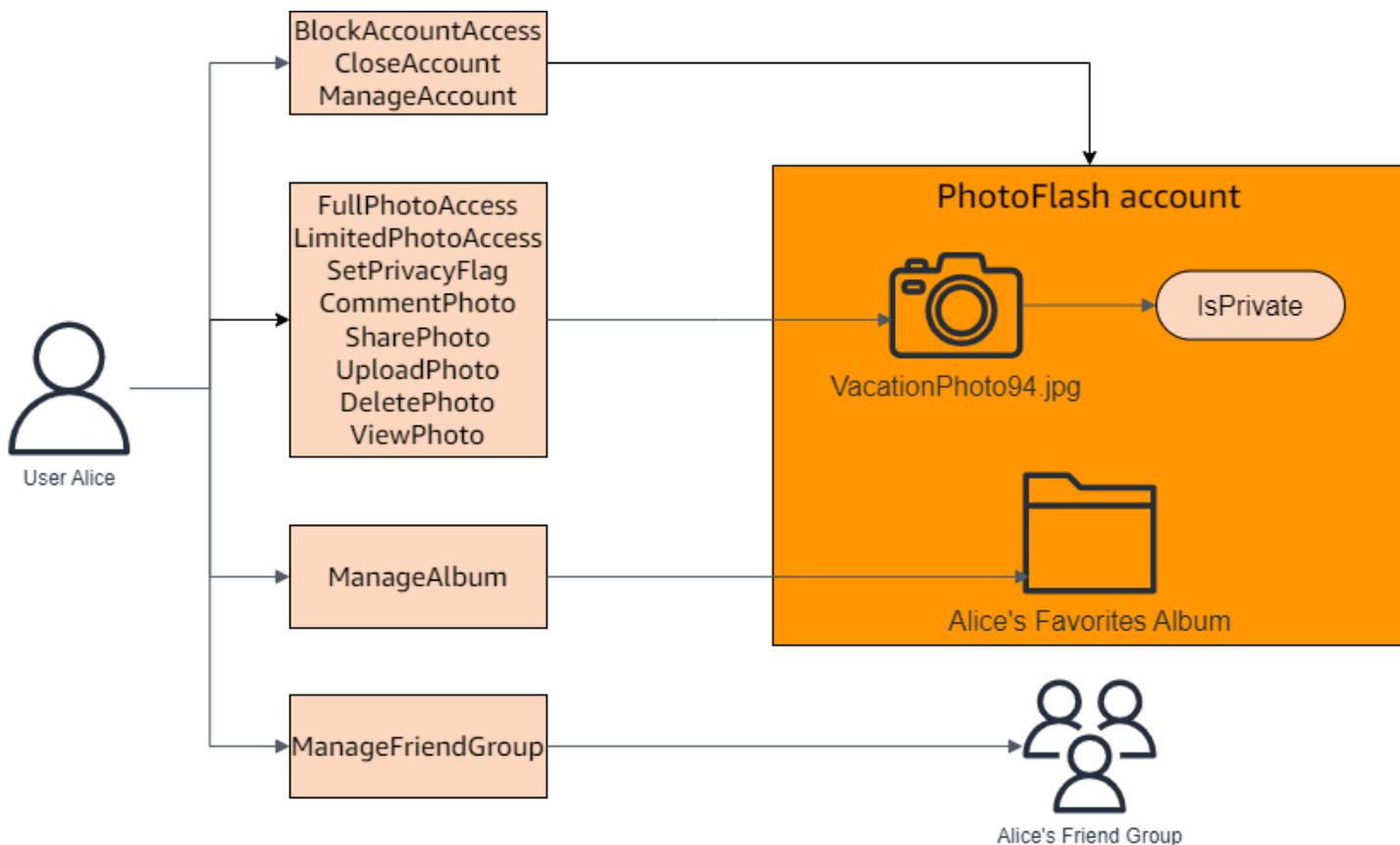
Verified Permissions を初めて使用する場合は、サンプルポリシーストアの 1 つを使用して、Verified Permissions の仕組みを理解しておくことをお勧めします。サンプルポリシーストアには、定義済みのポリシーとスキーマが用意されています。

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. [設定方法] 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。このチュートリアルでは、PhotoFlashポリシーストアを選択します。
3. サンプルポリシーストアのスキーマのネームスペースは、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、ポリシー、ポリシーテンプレート、およびサンプルポリシーストアのスキーマを使用して作成されます。

次の図は、PhotoFlash サンプルポリシーストアアクションと、それらが適用されるリソースタイプの関係を示しています。



サンプルポリシーストア用のテンプレートリンクポリシーの作成

PhotoFlash サンプルポリシーストアには、ポリシー、ポリシーテンプレート、スキーマが含まれています。サンプルポリシーストアに含まれる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できます。

サンプルポリシーストア用のテンプレートリンクポリシーを作成するには

1. <https://console.aws.amazon.com/verifiedpermissions/> にある 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. 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ます。説明は、非プライベート共有写真へのフルアクセスを付与し、次へを選択します。
5. プリンシパルには、 と入力します `PhotoFlash::User::"Alice"`。リソースには、 と入力します `PhotoFlash::Album::"Bob-Vacation-Album"`。
6. 「テンプレートにリンクされたポリシーの作成」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。

7. PhotoFlash サンプルポリシーストア用に別のテンプレートにリンクされたポリシーを作成します。[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の作成] を選択します。
8. 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、「非プライベート共有写真への制限付きアクセスを付与する」という説明を付けて、「次へ」を選択します。
9. プリンシパルには、 と入力します `PhotoFlash::FriendGroup::"MySchoolFriends"`。リソースには、 と入力します `PhotoFlash::Album::"Alice's favorite album"`。
10. 「テンプレートにリンクされたポリシーの作成」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。

チュートリアル次のセクションで、新しいテンプレートにリンクされたポリシーをテストします。のテンプレートにリンクされたポリシーの作成に使用できる値のその他の例については PhotoFlash、「」を参照してください [PhotoFlash 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のテスト

サンプルポリシーストアとテンプレートにリンクされた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[承認リクエスト](#)を実行することで、サンプルの Verified Permissions 静的ポリシーと新しいテンプレートにリンクされたポリシーをテストできます。

サンプルポリシーストアを作成した時期によっては、ポリシーテンプレートがこの手順のリファレンスと異なる場合があります。チュートリアルはこの部分を開始する前に、ポリシーストア PhotoFlash の例に続く各ポリシーテンプレートがあることを確認してください。ポリシーがこれらのポリシーと一致しない場合は、既存のポリシーを編集するか、サンプルプロジェクトオプションから新しいポリシーストアを作成します PhotoFlash。

非プライベート共有写真へのフルアクセスを許可する

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

非プライベート共有写真への制限付きアクセスを許可する

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

サンプルポリシーストアポリシーをテストするには

1. <https://console.aws.amazon.com/verifiedpermissions/> にある 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 PhotoFlash::User を選択します。テキストボックスにユーザーの ID を入力します。例えば、Alice です。

5. プリンシパルには「親を追加」を選択しないでください。
6. Account: Entity 属性で、PhotoFlash::Account エンティティが選択されていることを確認します。アカウントの識別子を入力します。例えば、Alice-account です。
7. リソースセクションで、PhotoFlash::Photo リソースタイプを選択します。テキストボックスに写真の識別子を入力します。例えば、photo.jpeg です。
8. 親を追加を選択し、エンティティタイプに PhotoFlash::Account を選択します。ユーザーの [アカウント: エンティティ] フィールドに指定した写真の親アカウントに同じ識別子を入力します。例えば、Alice-account です。
9. アクションセクションで、有効なアクションのリストから PhotoFlash::Action::" ViewPhoto を選択します。
10. 「追加のエンティティ」セクションで、「このエンティティを追加」を選択して、推奨するアカウントエンティティを追加します。
11. ページ上部の [承認リクエストを実行] を選択して、サンプルポリシーストア内の Cedar ポリシーの承認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どうかの決定が表示されるはずですが。

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ている静的ポリシーと、前のセクションで作成したテンプレートにリンクされたポリシーに基づく承認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]	認証決定
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

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]	認証決定
PhotoFlas h::User Alice	PhotoFlas h::Account Alice-account	PhotoFlas h::Photo Bob-Photo .jpeg	PhotoFlas h::Album Bob-Vacation-Album	PhotoFlas h::Action ::ViewPhoto"	許可
PhotoFlas h::User Alice	PhotoFlas h::Account Alice-account	PhotoFlas h::Photo Bob-Photo .jpeg	PhotoFlas h::Album Bob-Vacation-Album	PhotoFlas h::Action ::DeletePhoto"	拒否
PhotoFlas h::User Alice	PhotoFlas h::Account Alice-account	PhotoFlas h::Photo Bob-Photo .jpeg、IsPrivate : Boolean true	PhotoFlas h::Album Bob-Vacation-Album	PhotoFlas h::Action ::ViewPhoto"	拒否
PhotoFlas h::User Jane、PhotoFlas h::Friend Group MySchoolFriends	PhotoFlas h::Account Jane-account	PhotoFlas h::Photo photo.jpeg	PhotoFlas h::Album Aliceのお気に入りアルバム	PhotoFlas h::Action ::ViewPhoto"	許可
PhotoFlas h::User Jane、PhotoFlas h::Friend Group MySchoolFriends	PhotoFlas h::Account Jane-account	PhotoFlas h::Photo photo.jpeg	PhotoFlas h::Album Aliceのお気に入りアルバム	PhotoFlas h::Action ::DeletePhoto"	拒否

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。各ポリシーストアには、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が含まれています。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。[認証リクエスト](#)を行う際には、ポリシーストアを指定する必要があります。

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。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。詳細については、「Cedar ポリシー言語リファレンスガイド」の「[名前空間](#)」を参照してください。

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。

- **ガイド付き設定** — 最初のポリシーを作成する前に、有効なアクションを含むリソースタイプとプリンシパルタイプを定義します。
- **サンプルポリシーストア** — 定義済みの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** — 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. 「設定方法」セクションで、「ガイド付き設定」を選択します。

2. ポリシーストアの説明を入力します。このテキストは、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。例えば、Weather はを更新します。
3. 「詳細」セクションに、スキーマの名前空間を入力します。
4. [次へ]をクリックします。
5. 「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。
6. (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。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。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。
7. 「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。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。
8. 「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。
9. [次へ]をクリックします。
10. 「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。プリンシパル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。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 Amazon Cognito 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。[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、接続する Amazon Cognito ユーザープールのユーザープール ID を入力します。[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 による認証](#)」を参照してください。
11. [次へ]をクリックします。
12. 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。

13. 「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。
 - 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。
14. [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。
 - 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。
15. [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。
16. 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。

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. [設定方法] 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達と共有できるようにする、顧客向けのウェブアプリケーションのサンプルです。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 および TinyTodo 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、ユーザーがタスクと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、[Verified Permissions サンプルポリシーストア用のテンプレートリンクポリシーの例](#)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. 設定方法セクションで、「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「[テンプレートにリンクされたポリシーの作成](#)」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

AWS CLIを使用して作成したポリシーストアは空です。

- 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、[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

`CreatePolicyStoreAPI`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、[CreatePolicyStore](#)「Amazon Verified Permissions API リファレンスガイド」の「」を参照してください。

Verified Permissions ポリシーストアの切り替え

AWS Management Console

ポリシーストアを切り替えるか、追加のポリシーストアを作成するには

1. Verified Permissions コンソール <https://console.aws.amazon.com/verifiedpermissions/> 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、[現在のポリシーストア] の横にある [切り替え] を選択します。
3. 既存のポリシーストアを切り替えたり、追加のポリシーストアを作成したりできます。
 - ポリシーストアを切り替えるには、切り替え先のポリシーストアのポリシーストア ID を選択します。
 - 新しいポリシーストアを作成するには、[新しいポリシーストアを作成] を選択します。
「[Verified Permissions ポリシーストアの作成](#)」の手順に従います

AWS CLI

ポリシーストアを切り替えるか、追加のポリシーストアを作成するには

AWS CLIは「デフォルト」のポリシーストアを維持しません。代わりに、ほとんどのAWS CLIコマンドは--policy-store-idを使用して各コマンドに使用するポリシーストアを指定します。

新しいポリシーストアを作成するには、[create-policy-store](#) コマンドを使用します。

検証済み権限ポリシーストアの削除

AWS Management Console

ポリシーストアを削除するには

1. Verified Permissions コンソール <https://console.aws.amazon.com/verifiedpermissions/> 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。

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できます。

```
$ 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。

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。

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カ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 Management Console

スキーマを作成するには

1. <https://console.aws.amazon.com/verifiedpermissions/> の「Verified Permissions」コンソールを開きます。ポリシーストアを選択してください。
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。

ポリシーストアは、AWS CLI 次の例のようなコマンドを実行して作成できます。

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": "PSEXAMPLEeabcdefg111111",
"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 検証済みアクセス許可 API リファレンスガイドのを参照してください[PutSchema](#)。

ビジュアルモードでのスキーマの編集

検証済みアクセス権限コンソールで [スキーマ] を選択すると、ビジュアルモードには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。この最上位のビューまたはエンティティの詳細から、[スキーマの編集] を選択してスキーマの更新を開始できます。ビジュアルモードは、ネストされたレコードなどの一部のスキーマ形式では使用できません。

ビジュアルスキーマエディターは、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。[展開] を選択すると、スキーマのエンティティの関係を最大限に把握できます。

アクションダイアグラム

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、プリンシパルが実行できるアクション、アクションを実行できるリソースが表示されます。エンティティ間の線は、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できることを示しています。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、2 つ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。[アクション | リソースタイプ | プリンシパル・タイプ] でフィルタを選択すると、エンティティが独自の接続のみを含むビューに表示されます。

エンティティタイプ図

エンティティタイプ図は、プリンシパルとリソースの関係に焦点を当てています。スキーマ内の複雑なネストされた親関係を理解したい場合は、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の詳細が表示されます。

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、特定のアクションやエンティティタイプの詳細をすぐに確認したい場合に便利です。任意のエンティティを選択すると、詳細が表示されます。

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

1. <https://console.aws.amazon.com/verifiedpermissions/> にある Verified Permissions コンソールを開きます。ポリシーストアを選択してください。
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。エンティティ関係図を確認し、スキーマに加えたい変更を計画します。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、他のエンティティと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、スキーマに対するポリシーを検証する際に、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. 「属性を追加」を選択すると、承認リクエスト内のアクションのコンテキストに追加できる属性が追加されます。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、スキーマに対する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 モードでのスキーマの編集

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

1. <https://console.aws.amazon.com/verifiedpermissions/> にある 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. [変更の保存] をクリックします。

スキーマの削除

AWS Management Console

Verified Permissions スキーマを削除するには

1. <https://console.aws.amazon.com/verifiedpermissions/> にある Verified Permissions コンソールを開きます。ポリシーストアを選択してください。
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. [ドメインを削除] を選択します。

AWS CLI

Verified Permissions スキーマを削除するには

スキーマを削除コマンドはありません。cedarJsonフィールドに空のスキーマを指定してput-schemaコマンドを実行すると、ポリシーストア内のスキーマを削除できます。空のスキーマは、波かっこ {} で表されます。

```
$ aws verifiedpermissions put-schema \  
  --policy-store-id PSEXAMPLEabcdefgh111111 \  
  --definition cedarJson='{}' {  
    "policyStoreId": "PSEXAMPLEabcdefgh111111",  
    "namespaces": [],  
    "createdDate": "2023-06-14T21:55:27.347581Z",  
    "lastUpdatedDate": "2023-06-19T17:55:04.95944Z"  
  }
```

Amazon Verified Permissions ポリシーの検証モード

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。検証に失敗した場合、認証済みアクセス権限はリクエストを拒否します。

AWS Management Console

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 <https://console.aws.amazon.com/verifiedpermissions/> 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。

⚠ 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は、ユニバーサルユニーク識別子 (UUID) などの値を使用してください。

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に、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 == User::"a1b2c3d4-e5f6-a1b2-c3d4-EXAMPLE11111", // alice`

- 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 ID は、AWS CloudTrail トレイルで共有されるログエントリに含まれます。

Amazon Verified Permissions のエンティティフォーマット

Amazon Verified Permissions では、Cedar ポリシー言語を使用してポリシーを作成します。サポートされるポリシーの構文とデータ型は、Cedar ポリシー言語リファレンスガイドの「[Cedar での基本的なポリシー構築](#)」および「[Cedar でサポートされるデータ型](#)」のトピックで概説されている構文

とデータ型と一致します。ただし、Verified Permissions と Cedar では、承認リクエストを行う際のエンティティのフォーマットに違いがあります。

Verified Permissions のエンティティの JSON フォーマットは、以下の点で Cedar と異なります。

- 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

次の例は、リスト内のエンティティが Cedar を使用してフォーマットする方法を示しています。

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

```
}  
]
```

次の例は、前の Cedar リストの例と同じエンティティを「Verified Permissions」でフォーマットする方法を示しています。

```
{  
  "Set": [  
    {  
      "Record": {  
        "number": {  
          "Long": 1  
        }  
      }  
    },  
    {  
      "Record": {  
        "sentence": {  
          "String": "Here is an example sentence"  
        }  
      }  
    },  
    {  
      "Record": {  
        "question": {  
          "Boolean": false  
        }  
      }  
    }  
  ]  
}
```

以下の例は、承認リクエスト内のポリシーを評価するための Cedar エンティティのフォーマット方法を示しています。

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

次の例は、前の Cedar の例と同じエンティティが「検証済みアクセス権」でどのようにフォーマットされるかを示しています。

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

Amazon Verified Permissions 静的ポリシーの作成

Cedar 静的ポリシーを作成して、プリンシパルがアプリケーションの特定のリソースに対して特定のアクションを実行することを許可または拒否できます。

AWS Management Console

静的ポリシーを作成するには

1. <https://console.aws.amazon.com/verifiedpermissions/> の「Verified Permissions」コンソールを開きます。ポリシーストアを選択してください。
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。
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。

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. [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ポリシーを作成できます。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\"  
}
```

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

ポリシーストア内の既存の Cedar 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だけです。静的ポリシーの特定の要素のみを変更できます。

- 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。

- ポリシーを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に変更する。
- 静的ポリシーの効果をpermitまたはforbidから変更する。
- 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。

テンプレートにリンクされたポリシーを変更するには、代わりに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、「[ポリシーテンプレートの編集](#)」を参照してください。

AWS Management Console

静的ポリシーを編集するには

1. <https://console.aws.amazon.com/verifiedpermissions/> にある Verified Permissions コンソールを開きます。ポリシーストアを選択してください。
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 PSEXAMPLEEabcdefg111111  
  
{  
  "createdDate": "2023-06-12T20:33:37.382907+00:00",  
  "lastUpdatedDate": "2023-06-12T20:33:37.382907+00:00",  
  "policyId": "SPEXAMPLEEabcdefg111111",  
  "policyStoreId": "PSEXAMPLEEabcdefg111111",  
  "policyType": "STATIC",  
  "principal": {  
    "entityId": "janeFriends",  
    "entityType": "UserGroup"  
  },  
  "resource": {  
    "entityId": "vacationFolder",  
    "entityType": "Album"  
  }  
}
```

ポリシーの表示

AWS Management Console

Verified Permissions ポリシーを表示するには

1. <https://console.aws.amazon.com/verifiedpermissions/> にある 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。作成したポリシーがすべて表示されます。
3. 「検索」テキストボックスを選択して、ポリシーをプリンシパルまたはリソースでフィルタリングします。
4. ポリシーの横にあるラジオボタンを選択すると、ポリシーの作成、更新日時、ポリシーの内容など、ポリシーに関する詳細が表示されます。

5. ポリシーを削除するには、ポリシーの横にあるラジオボタンを選択し、次に [削除] を選択します。[ポリシーを削除] を選択してポリシーの削除を確認します。

AWS CLI

ポリシーストアで使用可能なすべてのポリシーを一覧表示するには

[GetPolicy](#) オペレーションを使用してポリシーのリストを表示できます。次の例では、静的ポリシーとテンプレートにリンクされたポリシーを含むリストを取得します。

```
$ aws verifiedpermissions list-policies \
  --policy-store-id PSEXAMPLEabcdefgh111111
{
  "Policies": [
    {
      "createdAt": "2023-05-17T18:38:31.359864+00:00",
      "definition": {
        "static": {
          "Description": "Grant everyone of janeFriends UserGroup access
to the vacationFolder Album"
        }
      },
      "lastUpdatedDate": "2023-05-18T16:15:04.366237+00:00",
      "policyId": "SPEXAMPLEabcdefgh111111",
      "policyStoreId": "PSEXAMPLEabcdefgh111111",
      "policyType": "STATIC",
      "resource": {
        "entityId": "publicFolder",
        "entityType": "Album"
      }
    },
    {
      "createdAt": "2023-05-22T18:57:53.298278+00:00",
      "definition": {
        "templateLinked": {
          "policyTemplateId": "PTEXAMPLEabcdefgh111111"
        }
      },
      "lastUpdatedDate": "2023-05-22T18:57:53.298278+00:00",
      "policyId": "TPEXAMPLEabcdefgh111111",
      "policyStoreId": "PSEXAMPLEabcdefgh111111",
      "policyType": "TEMPLATELINKED",
      "principal": {
```

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

個々のポリシーの詳細を表示するには

[GetPolicy](#) オペレーションを使用してポリシーの詳細を取得できます。次の例では、テンプレートにリンクされたポリシーの詳細を取得します。

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

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

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

Amazon Verified Permissions

以下の検証済み権限ポリシーの例は、『Cedar ポリシー言語リファレンスガイド』の「[スキーマの例](#)」 PhotoFlash セクションで説明されている仮想アプリケーション用に定義されたスキーマに基づいています。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します。](#)
- [エンティティのグループへのアクセスを許可します。](#)
- [どのエンティティにもアクセスを許可します。](#)
- [エンティティ \(ABAC\) の属性へのアクセスを許可します。](#)
- [アクセスを拒否する](#)

個々のエンティティへのアクセスを許可します。

この例では、ユーザー alice に写真 VacationPhoto94.jpg の表示を許可するポリシーを作成する方法を示します。

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

エンティティのグループへのアクセスを許可します。

この例では、グループ alice_friends 内の誰でも写真 VacationPhoto94.jpg を閲覧できるようにするポリシーを作成する方法を示します。

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

この例では、ユーザー `alice` がアルバム `alice_vacation` 内の写真を表示できるようにするポリシーを作成する方法を示します。

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

この例では、ユーザー `alice` がアルバム `alice_vacation` 内の写真を表示、編集、または削除することを許可するポリシーの作成方法を示します。

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

この例では、アルバム `alice_vacation` 内のユーザー `alice` に権限を許可するポリシーを作成する方法を示します。 `admin` は、写真を表示、編集、削除する権限を含むスキーマ階層で定義されたグループです。

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

この例では、アルバム `alice_vacation` 内のユーザー `alice` に権限を許可するポリシーを作成する方法を示します。 `viewer` は、写真を表示してコメントする権限を含むスキーマ階層で定義されたグループです。ユーザー `alice` には、ポリシーにリストされている 2 番目のアクションによっても `edit` 権限が付与されます。

```
permit(  

```

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

どのエンティティにもアクセスを許可します。

この例では、認証されたプリンシパルにアルバムalice_vacationを表示することを許可するポリシーの作成方法を示します。

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

この例では、ユーザー alice が jane アカウント内のすべてのアルバムを一覧表示し、各アルバム内の写真を一覧表示し、アカウント内の写真を表示できるようにするポリシーを作成する方法を示します。

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

この例では、ユーザーaliceがアルバムjane_vaction内のリソースに対して任意のアクションを実行することを許可するポリシーの作成方法を示します。

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

エンティティ (ABAC) の属性へのアクセスを許可します。

属性ベースのアクセスコントロール (ABAC) は、属性に基づいて権限を定義する認可戦略です。Verified Permissionsでは、プリンシパル、アクション、リソースに属性を添付できます。する

と、これらの属性は、リクエストのコンテキストを構成するプリンシパル、アクション、リソースの属性を評価するポリシーの `when` および `unless` 句内で参照できるようになります。

以下の例では、『Cedar ポリシー言語リファレンスガイド』の「[スキーマの例](#)」 PhotoFlash セクションで説明されている仮定のアプリケーションで定義されている属性を使用しています。

この例では、ジョブ レベル 5 以上の HardwareEngineering 部門のプリンシパルに、アルバム `device_prototypes` 内の写真の表示と一覧表示を許可するポリシーを作成する方法を示します。

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

この例では、ユーザー `alice` がファイルタイプ `JPEG` の任意のリソースの表示をすることを許可するポリシーの作成方法を示します。

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

アクションにはコンテキスト属性があります。context これらの属性は承認リクエストに渡す必要があります。この例は、`alicereadonly` ユーザーが任意の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方法を示しています。 `appliesTo` スキーマのアクションのプロパティを設定することもできます。たとえば、 `ViewPhotoPhotoFlash::Photo` ユーザーが特定の種類のリソースに対してのみ認証を試みることができるようにしたい場合に、リソースに有効なアクションを指定します。

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

```
resource
) when {
    context has readOnly &&
    context.readOnly == true
};
```

ただし、スキーマ内のアクションのプロパティを設定するより適切な方法は、アクションを機能的なアクショングループにまとめることです。たとえば、ReadOnlyPhotoAccessという名前のアクションを作成し、PhotoFlash::Action::"ViewPhoto"ReadOnlyPhotoAccessアクショングループのメンバーになるように設定できます。この例は、Alice にそのグループ内の読み取り専用アクションへのアクセスを許可するポリシーを作成する方法を示しています。

```
permit(
    principal == PhotoFlash::User::"alice",
    action,
    resource
) when {
    action in PhotoFlash::Action::"ReadOnlyPhotoAccess"
};
```

この例では、すべてのプリンシパルがowner属性を持つリソースに対して任意のアクションを実行することを許可するポリシーを作成する方法を示します。

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

この例では、プリンシパルのdepartment属性がリソースのdepartment属性と一致する場合に、プリンシパルに任意のリソースの表示を許すポリシーを作成する方法を示します。

Note

ポリシー条件に指定された属性がないエンティティは、承認決定時にポリシーが無視され、そのエンティティではそのポリシーの評価が失敗します。たとえば、department属性を持たないプリンシパルには、このポリシーではどのリソースへのアクセスも許可されません。

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

この例では、プリンシパルがリソースの owner である場合、またはプリンシパルがリソースの admins グループの一部である場合に、プリンシパルがリソースに対して任意の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方法を示します。

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

アクセスを拒否する

ポリシーにポリシーの効果に関する forbid が含まれている場合、アクセス許可を付与するのではなく、アクセス許可を制限します。

Important

認可中に、permit ポリシーと forbid ポリシーの両方が適用される場合は、どちらが優先されます。

以下の例では、『Cedar ポリシー言語リファレンスガイド』の「[スキーマの例](#)」 PhotoFlash セクションで説明されている仮定のアプリケーションで定義されている属性を使用しています。

この例では、ユーザー alice がリソースに対して readOnly を除くすべてのアクションを実行することを拒否するポリシーを作成する方法を示します。

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

この例では、プリンシパルがリソースの `owner` 属性を持っていない限り、`private` 属性を持つすべてのリソースへのアクセスを拒否するポリシーを作成する方法を示します。

```
forbid (  
  principal,  
  action,  
  resource  
)  
when {  
  resource.private  
}  
unless {  
  principal == resource.owner  
};
```

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

Verified Permissions で Cedar ポリシーテンプレートを作成して、システムのアクセス制御ルールを定義できます。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含む Cedar ポリシーです。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、複数のプリンシパルやリソースにアタッチできます。ポリシーテンプレートの更新は、そのテンプレートを使用するすべてのプリンシパルとリソースに反映されます。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。

ポリシーテンプレートを使用して、アプリケーション全体で共有できるポリシーを作成することをお勧めします。たとえば、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読み取り、編集、コメントの権限を付与するエディター用のポリシーテンプレートを作成できます。

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

プリンシパルをリソースの編集者に指定すると、アプリケーションはそのテンプレートを使用してポリシーをインスタンス化し、プリンシパルにリソースの読み取り、編集、コメントアクションを実行する権限を与えることができます。

テンプレートの作成

AWS Management Console

ポリシーテンプレートを作成するには

1. <https://console.aws.amazon.com/verifiedpermissions/> にある 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 PSEXAMPLEabcdefghijklmnop111111
{
  "createdDate": "2023-05-18T21:17:47.284268+00:00",
  "lastUpdatedDate": "2023-05-18T21:17:47.284268+00:00",
  "policyStoreId": "PSEXAMPLEabcdefghijklmnop111111",
  "policyTemplateId": "PTEXAMPLEabcdefghijklmnop111111"
}
```

テンプレートにリンクされたポリシーの作成

テンプレートにリンクされたポリシーを作成して、ポリシーテンプレートにリンクできます。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の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すべてのポリシーは、その時点以降の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。

AWS Management Console

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 <https://console.aws.amazon.com/verifiedpermissions/> 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。たとえば、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。入力するとAdminUsers、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の作成] 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、ここで示されるステートメントを使用して示されるステートメントを使用して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 == Action::"view",
    resource == 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": "pt-4651be67-c128-4d22-8e67-9b068980c631",
    "principal": {
      "entityType": "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 PSEXAMPLEEabcdefg111111
{
  "createdDate": "2023-05-22T18:57:53.298278+00:00",
  "lastUpdatedDate": "2023-05-22T18:57:53.298278+00:00",
  "policyId": "TPEXAMPLEEabcdefg111111",
  "policyStoreId": "PSEXAMPLEEabcdefg111111",
  "policyType": "TEMPLATELINKED",
  "principal": {
    "entityId": "alice",
    "entityType": "User"
  },
  "resource": {
    "entityId": "VacationPhoto94.jpg",
    "entityType": "Photo"
  }
}
```



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

Verified Permissions サンプルポリシーストア用のテンプレートリンクポリシーの例

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 テンプレートにリンクされたポリシーの例

この例では、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ます。このポリシーでは、個々のユーザーと写真とのプライベート以外の共有写真へのアクセスを制限します。

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"  
);
```

この例では、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ます。個々のユーザーとアルバムとのプライベート共有以外の写真への制限付きアクセスを許可します。

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

この例は、ポリシーテンプレートを使用して、友達グループと個々の写真との非公開の共有写真への制限付きアクセスを許可するテンプレートリンクポリシーを作成する方法を示しています。

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

この例では、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ます。友達グループとアルバムとの非公開の共有写真への制限付きアクセスを許可します。

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

この例では、ポリシーテンプレートを使用して、友達グループと個々の写真とのプライベート共有以外の写真へのフルアクセスを付与するテンプレートリンクポリシーを作成する方法を示します。

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

この例では、アカウントの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ます。

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

DigitalPetStore

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。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ているポリシーを表示できます。

TinyToDo テンプレートにリンクされたポリシーの例

この例では、個々のユーザーとタスク リストに閲覧者アクセスを与え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"  
);
```

この例は、個々のユーザーとタスクリストに編集者アクセスを許可するポリシーテンプレートを使用して、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 プロバイダでの Amazon Verified Permissions の使用

Verified Permissions で [OpenID Connect \(OIDC\)](#) ID プロバイダー (IdPs) を使用して、Verified Permissions ポリシーでプリンシパルとして使用するユーザー属性を渡すことができます。

Amazon Cognito とアイデンティティソースとの連携

ID ソースは、ポリシーストアへの認証リクエストを簡略化するために ID プロバイダが参照するユーザー情報の集まりです。ID ソースを作成して、Verified Permissions アプリケーションのプリンシパルに関する情報を提供できます。AWS リージョン、Amazon Cognito ユーザープール ID、およびアイデンティティソースのプリンシパルタイプを指定できます。Verified Permissions は同じの Amazon Cognito ユーザープールでのみ機能するため AWS アカウント、別のアカウントの ID ソースを指定することはできません。

ユーザープール JSON Web トークン (JWT) クレームには、ユーザー属性が含まれます。カスタムクレームを追加して、Verified Permissions が下す認証決定を通知できます。ID トークンのクレームには `cognito:username` と `cognito:groups` が含まれます。詳細については、「Amazon Cognito デベロッパーガイド」の「[Amazon Cognito ユーザープール](#)」を参照してください。

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、JWT は署名と有効性を備えた自己完結型の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 では、取り消されたトークン、またはユーザーに対して発行され、後で削除されたトークンに基づいてアクセスを許可できるということです。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。

ユーザープールには、カスタム属性を追加できます。カスタム属性は、標準属性と区別するために、`custom:` プレフィックスを必要とします。たとえば、ユーザープールに `custom:joblevel` 属性を追加できます。詳細については、「Amazon Cognito デベロッパーガイド」の「[カスタム属性](#)」を参照してください。

Amazon Cognito : ユーザープールのクレームと文字を含む属性を使用して Verified Permissions で Cedar ポリシーを記述する場合、Cedar ポリシーの構文に準拠するには、Cedar ポリシーでコロン (:) ではなくピリオド (.) を使用してそれらを参照する必要があります。たとえば、cognito:username と cognito:groups をそれぞれ cognito.username と cognito.groups に変更する必要があります。

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 cognito または custom クレームが含まれている場合、を含む認証リクエスト [IsAuthorizedWithToken](#) はで失敗します ValidationException。

この例では、プリンシパルに対応する Amazon Cognito ユーザープールのカスタムクレームを参照する、ポリシーの作成方法を示します。

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

詳細については、Amazon Cognito デベロッパーガイドの「[Amazon Verified Permissions による認証](#)」を参照してください。

他の ID プロバイダーの使用

任意の OpenID Connect (OIDC) プロバイダー (IdP) の JSON ウェブトークン (JWT) からエンティティ属性を抽出し、検証済みの権限に解析できます。

この例では、OIDC 準拠の IDP から Verified Permissions を呼び出す方法を示します。¹

```
async function authorizeUsingJwtToken(jwtToken) {  
  
  const payload = await verifier.verify(jwtToken);  
  
  var principalEntity = {  
    entityType: "PhotoFlash::User", // the application needs to fill in the  
    relevant user type
```

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

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

  return authResult;
}

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

```
value.forEach((item) => {
  attributeItem.push({
    string: item,
  });
});
attributes[key] = {
  set: attributeItem,
};
} 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 need to use the claim that
represents the user-id
  }
};
return [entityItem];
}
```

1 このコード例では、OIDC 互換 によって署名された JWTs を検証するために [aws-jwt-verify](#) ライブラリを使用しています IdPs。

Amazon Verified Permissions ID ソースの作成

次の手順では、Amazon Cognito ID ソースをポリシーストアに追加します。

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。

プリンシパルエンティティタイプは、AWS CLIまたは [CreateIdentitySource](#) Verified Permissions API [create-identity-source](#) でを使用して ID ソースを作成するときに除外できます。ただし、エンティティタイプが空白の場合は、エンティティタイプが の ID ソースが作成されます。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 Management Console

Amazon Cognito ユーザープール ID ソースを作成するには

1. <https://console.aws.amazon.com/verifiedpermissions/> にある 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 ソースを変更] を選択します。
4. Cognito ユーザープールの詳細セクションで、AWS リージョンを選択し、ID ソースのユーザープール ID を入力します。
5. 「プリンシパルの詳細」セクションで、ID ソースのプリンシパル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. 「クライアントアプリケーション検証」セクションで、クライアントアプリケーション ID 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。

7. [ID ソースを作成]を選択します。

8. Cedar ポリシー内の ID またはアクセストークンから抽出した属性を参照できるようにするには、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種類を 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 トークンの属性を Cedar の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、[Amazon Cognito トークンを Verified Permissions スキーマにマッピングする](#)を参照してください。

AWS CLI

Amazon Cognito 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、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。

以下のconfig.txtファイルには、create-identity-sourceコマンドの--設定パラメータで使用される Amazon Cognito ユーザープールの詳細が含まれています。

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

コマンド:

```
$ 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"  
}
```

Cedar ポリシーの ID 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できるようにするには、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種類を 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 トークンの属性を Cedar の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、[Amazon Cognito トークンを Verified Permissions スキーマにマッピングする](#)を参照してください。

認証されたユーザーの 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 Cognito デベロッパーガイドの「[Amazon Verified Permissions による認証](#)」を参照してください。

Amazon Verified Permissions ID ソースの編集

AWS Management Console

Amazon Cognito ユーザープールの ID ソースを更新するには

1. <https://console.aws.amazon.com/verifiedpermissions/> にある 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セクションで、AWS リージョンを選択し、ID ソースにユーザープール ID を入力します。
6. 「プリンシパルの詳細」セクションでは、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. 「クライアントアプリケーション検証」セクションで、クライアントアプリケーション ID 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. [変更を保存] をクリックします。
9. 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 ユーザープール 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。次の例では、指定した ID ソースを別の Amazon Cognito ユーザープールを使用するように更新します。

以下の config.txt ファイルには、create-identity-source コマンドの --configuration パラメータで使用される Amazon Cognito ユーザープールの詳細が含まれています。

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

コマンド:

```
$ aws verifiedpermissions update-identity-source \
  --update-configuration file://config.txt \
  --policy-store-id 123456789012
{
  "createdDate": "2023-05-19T20:30:28.214829+00:00",
  "identitySourceId": "ISEXAMPLEabcdefgh111111",
```

```
"lastUpdatedDate": "2023-05-19T20:30:28.214829+00:00",  
"policyStoreId": "PSEXAMPLEEabcdefg111111"  
}
```

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 Verified Permissions スキーマにマッピングする

Amazon Cognito ID トークンには、Verified Permissions スキーマの属性の命名に影響する 4 種類の属性があります。Amazon Cognito を Verified Permissions ポリシーストアの ID ソースとして使用し、トークン API アクションを使用して検証される Amazon Cognito [isAuthorizedWith](#) トークンの属性を使用してポリシーを記述するには、Amazon Cognito 属性をスキーマに追加する必要があります。

- cognito 名前空間プレフィックスが付いた Amazon Cognito 固有のクレーム (cognito:username など)。

Note

cognito:groups Amazon Cognito 固有のクレームは、現在、Verified Permissions スキーマにはマッピングされていません。Amazon Cognito グループでは、一意であることが保証されておらず、再利用できる識別子を使用できます。認証決定にこれらを含めると、アプリケーションにセキュリティ上のリスクが生じる可能性があります。

- custom 名前空間のプレフィックスが付いたカスタムクレーム (custom:employmentStoreCode など)。
- 名前空間プレフィックスが付いていない標準クレーム (email など)。
- tenant、department、clearance など、トークンのカスタマイズ中に追加される一時的なクレーム。

認証されたユーザーの 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 Cognito デベロッパーガイドの「[Amazon Verified Permissions による認証](#)」を参照してください。

次の ID トークンの例には、4 種類の属性がそれぞれ含まれています。これには、Amazon Cognito 固有クレーム `cognito:username`、カスタムクレーム `custom:employmentStoreCode`、標準クレーム `email`、および一時的なクレーム `tenant` が含まれます。

```
{
  "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_wBEbEZKaX",
  "cognito:username": "alice",
  "custom:employmentStoreCode": "petstore-dallas",
  "origin_jti": "5b9f50a3-05da-454a-8b99-b79c2349de77",
  "aud": "52n97d5afhfiuXXX",
  "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"
}
```

ID ストアを作成して Amazon Cognito ユーザープールに関連付けるときは、トークンが `IsAuthorizedWithToken` 評価リクエストに渡されるときに Cedar が生成するプリンシパルタイプを指定します。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。そのプリンシパルタイプと、Cedar ポリシーから参照できるようにする属性を定義する必要があります。

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、「[JSON モードでのスキーマの編集](#)」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 `CognitoUser` 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。

```
"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
      }
    }
  }
}
```

Amazon Cognito 属性を反映するようにスキーマを更新したら、その属性を参照するポリシーを作成できます。

```
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 アクセストークンのマッピング

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

- `client_id`は、アプリケーションのクライアント ID を表します。この値は、マシンの ID を表す認可 machine-to-machine シナリオで使用できます。
- `scope`は [OAuth 2.0 標準スコープ](#)で、トークン所有者の承認済みスコープを表します。

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`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_wBEbEZKaX",
  "client_id": "52n97d5afhfiu1c4di1k5m8f60",
  "origin_jti": "a1b2c3d4-e5f6-a1b2-c3d4-TOKEN11111111",
  "event_id": "bda909cb-3e29-4bb8-83e3-ce6808f49011",
  "token_use": "access",
  "scope": "aws.cognito.signin.user.admin",
  "auth_time": 1688092966,
  "exp": 1688096566,
  "iat": 1688092966,
  "jti": "a1b2c3d4-e5f6-a1b2-c3d4-TOKEN22222222",
  "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、「[JSON モードでのスキーマの編集](#)」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": "String"
            },
            "client_id": {
              "type": "String"
            }
          }
        }
      }
    },
    "type": "Record"
  }
}
```

Amazon Cognito 属性を反映するようにスキーマを更新したら、その属性を参照するポリシーを作成できます。

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

```
context.token.scope like "*aws.cognito.signin.user.admin*"
};
```

アプリケーションの認証モデルの設計

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て、アプリケーション UI で権限を管理する際にエンドユーザーに表示されるべき内容を明確に理解する必要があります。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？ それらは互いに関係を持っていますか？ たとえば、ファイルはフォルダ内に存在しますか？
- 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？
- エンドユーザーが「Admin」、「Operator」、「ReadOnly」などの事前定義されたアクセス許可から選択するか、アドホックポリシーステートメントを作成するか。それとも両方一緒ですか？
- 親フォルダから権限を継承するファイルなど、権限はリソース全体に継承されるべきですか？
- 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りますか？
- 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。その代わりに、他のモデルと同様、何よりもまずユーザーエクスペリエンスの設計作業であり、インターフェースのモックアップ、論理図、権限が製品内でユーザーに表示される内容にどのように影響するかについての全体的な説明などの成果物として現れることがあります。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。

このセクションでは、設計作業への取り組み方、注意すべき点、検証済み権限をうまく使用するためのベストプラクティスに関する一般的なガイダンスを提供します。

ここで紹介するガイドラインに加えて、[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](#)を考慮することを忘れないでください。

トピック

- [正規の「正しい」モデルはありません](#)
- [API オペレーションではなく、リソースに重点を置く](#)
- [複合認証は正常](#)
- [マルチテナンシーに関する考慮事項](#)
- [可能な場合は、ポリシースコープを入力します。](#)
- [すべてのリソースはコンテナ内にあります](#)
- [プリンシパルをリソースコンテナから分離する](#)
- [属性内に権限を埋め込まないでください](#)
- [モデルではきめ細かな権限を、ユーザーインターフェースでは集約した権限を優先します](#)
- [認証を問い合わせる他の理由も検討してください](#)

正規の「正しい」モデルはありません

認証モデルを設計する場合、一意に正しい回答は 1 つもありません。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証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 に似たオペレーティングシステムでファイルを作成しても、親フォルダから自動的にアクセス許可が継承されることはありません。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。

認証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。認証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、効果的な認証モデルを作成する上で最も役立つ前提条件となります。

API オペレーションではなく、リソースに重点を置く

消費者向けのほとんどのアプリケーションでは、権限はアプリケーションがサポートするリソースをモデルにしています。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。簡単でしょうか？

しかし、この API に焦点を当てたアプローチは最適とは言えません。API は、顧客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やリソースの代用にすぎないからです。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。

API のセマンティクスは無視して、基礎となるデータやリソース、およびそれらの関連付け操作に集中することをおすすめします。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	許可
<code>listMembersOfGroup</code>	グループに <code>viewGroupMembership</code> 権限が必要です。
<code>listGroupMembershipsForUser</code>	ユーザーに <code>viewGroupMembership</code> 権限が必要です。
<code>isUserInGroups</code>	ユーザーに <code>viewGroupMembership</code> 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。

複合認証は正常

複合認証とは、アプリケーションのインターフェースのボタンをクリックするなどの単一のユーザーアクティビティで、そのアクティビティが許可されているかどうかを判断するために複数の個別認証クエリが必要になる場合に発生します。たとえば、ファイルシステムの新しいディレクトリにファイルを移動する場合、ソースディレクトリからファイルを削除する権限、移動先ディレクトリにファイルを追加する権限、およびファイル自体を操作する権限 (アプリケーションによって異なります) という 3 つの異なる権限が必要になる場合があります。

認証モデルの設計に慣れていない場合、認証に関するすべての決定は 1 つの認証クエリで解決可能でなければならないと考えるかもしれません。しかし、これではモデルが過度に複雑になり、ポリシーステートメントが複雑になる可能性があります。実際には、複合認証を使用すると、より単純な認証モデルを作成するのに役立ちます。適切に設計された認証モデルの 1 つの基準は、個々のアクションを十分に分解すれば、ファイルの移動などの複合操作をプリミティブの直感的な集合体で表現できることです。

複合認証が発生するもう 1 つの状況は、権限を付与するプロセスに複数の関係者が関与する場合があります。ユーザーがグループのメンバーになることができる組織ディレクトリを考えてみましょう。簡単な方法は、グループオーナーに誰でも追加できる権限を与えることです。しかし、まずユーザーに追加に同意してもらいたい場合はどうすればよいのでしょうか。これにより、ユーザーとグループの両方がメンバーシップに同意しなければならないというハンドシェイク契約が導入されます。これを実現するために、ユーザーにバインドされた別の権限を導入して、ユーザーを任意のグループに追加で

きるのか、特定のグループに追加できるのかを指定できます。呼び出し元が後でメンバーをグループに追加しようとする場合、アプリケーションは権限の両側を強制する必要があります。つまり、呼び出し元には指定されたグループにメンバーを追加する権限があり、追加される個々のユーザーには追加する権限があるということです。N ウェイ ハンドシェイクが存在する場合、合意の各部分を強制するために N 複合認証クエリを監視するのが一般的です。

複数のリソースが関わっていて、権限をモデル化する方法が不明な設計上の課題に直面している場合、複合認証シナリオを使用している可能性があります。このような場合は、操作を複数の個別認証チェックに分解することで解決策が見つかるかもしれません。

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利用する企業やテナントなど、複数のお客様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。承認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。

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、すべての認証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、さまざまなポリシーストアに認可の決定をクエリします。

どちらの戦略も、AWS 請求に影響する可能性のある比較的大量の承認リクエストを作成します。では、どのようにアプローチを設計すべきですか？ Verified Permissions マルチテナンシー認証戦略に影響する可能性のある一般的な条件を次に示します。

テナントポリシーの分離

テナントデータを保護するには、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。

認証フロー

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管理

[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を管理するオーバーヘッドのレベルは、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。

テナントのオフオンボーディング

一部のテナントは、スキーマやそのケースに固有の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、データを削除したい場合、作業レベルは他のテナントとの分離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コォータがあります。クォータの詳細については、「[リソースのコォータ](#)」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で、独自の時間とリソースコミットメントが必要です。

考慮事項	共有ポリシーストアの エフォートレベル	テナントごとのポリシー ストアのエフォートレベル
テナントポリシーの分離	中。Must include tenant identifiers in policies and authorization requests.	低。Isolation is default behavior. Tenant-specific

policies are inaccessible to other tenants.

認証フロー	低。 All queries target one policy store.	中。 Must maintain mappings between each tenant and their policy store ID.
テンプレートとスキーマ管理	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.
テナントのオフオンボーディング	中。 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。アーキテクチャ上の決定を行う前に、2つのアプローチとその考慮事項を慎重に比較します。

アプリケーションにテナント固有のポリシーが必要ではなく、単一の [ID ソース](#) を使用する場合、すべてのテナントに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す。これにより、承認フローとグローバルポリシー管理がよりシンプルになります。1つの共有ポリシーストアを使用してテナントをオフオンボーディングすると、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、労力が軽減されます。

ただし、アプリケーションで多数のテナント固有のポリシーが必要な場合や、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。各ポリシーストアにテナントごとのアクセス許可を付与する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。テナントのオフオンボーディングには、ポリシーストアの削除が必要です。shared-policy-store 環境では、テナント固有のポリシーを検索して削除する必要があります。

可能な場合は、ポリシースコープを入力します。

ポリシースコープは、Cedar ポリシーステートメントの `permit` または `forbid` キーワードの後に開き括弧の間にある部分です。

```

Effect ———— permit (
Scope ———— principal == User::"e3527bb8-f74a-48da-818c-f7e6ef79bf7c",
               action == Photo::"readFile",
               resource in Album::"615e85bc-f03d-4915-b4eb-4c184b8da25d"
)
Conditions ———— when {
                  resource.private == false
                  };

```

可能な限り、principalおよびresourceの値を入力することをお勧めします。これにより、Verified Permissions がポリシーをインデックス化してより効率的に取得できるようになり、パフォーマンスが向上します。同じアクセス権限を多くの異なるプリンシパルまたはリソースに付与する必要がある場合は、ポリシーテンプレートを使用してプリンシパルとリソースの各ペアに添付することをお勧めします。

1つのwhen句にプリンシパルとリソースのリストを含む大規模なポリシーを1つ作成することは避けてください。そうすると、スケーラビリティの限界や運用上の課題にぶつかる可能性があります。たとえば、ポリシー内の大きなリストから1人のユーザーを追加または削除するには、ポリシー全体を読み取り、リストを編集し、新しいポリシーをすべて記述し、ある管理者が別の管理者の変更を上書きした場合の同時実行エラーを処理する必要があります。対照的に、きめ細かな権限を多数使用すれば、ユーザーを追加または削除するのは、そのユーザーに適用される1つのポリシーを追加または削除するのと同じくらい簡単です。

すべてのリソースはコンテナ内にあります

承認モデルを設計するときは、すべてのアクションを特定のリソースに関連付ける必要があります。viewFileのようなアクションでは、それを適用できるリソースは直感的です。個々のファイルでも、フォルダ内の複数のファイルでもかまいません。ただし、createFileのような操作は直感的ではありません。ファイルを作成する機能をモデル化する場合、どのリソースに適用されますか？ファイルがまだ存在していないため、ファイルそのものであってはなりません。

これはリソース作成に関する一般的な問題の一例です。リソース作成はブートストラップの問題です。リソースがまだ存在しない場合でも、リソースを作成する権限を何かに付与する方法が必要です。解決策は、すべてのリソースは必ず何らかのコンテナ内に存在しなければならない、権限のアンカーポイントとなるのはコンテナ自体であることを認識することです。たとえば、システムにフォルダがすでに存在する場合、ファイルの作成機能はそのフォルダに対する権限としてモデル化できます。なぜなら、そのフォルダは新しいリソースをインスタンス化するための権限が必要な場所だからです。

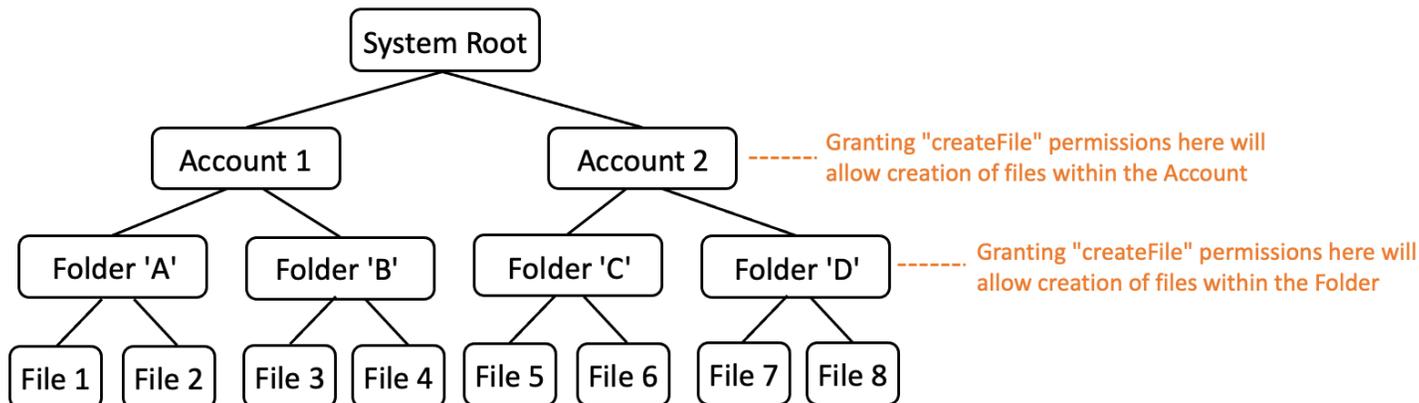
```
permit (  
    principal == User::"6688f676-1aa9-456a-acf4-228340b54e9d",  
    action == Action::"createFile",  
    resource == Folder::"c863f89b-461f-4fc2-b638-e5fa5f79a48b"  
);
```

しかし、フォルダーが存在しない場合はどうなるのでしょうか？これは、リソースがまだ存在しないアプリケーションのまったく新しい顧客アカウントかもしれません。このような状況でも、「顧客はどこで新しいファイルを作成できるのか」と尋ねれば直感的に理解できるコンテキストがまだ残っています。ランダムな顧客アカウント内でファイルを作成できないようにしたくはないでしょう。むしろ、暗黙のコンテキストがあります。それは、顧客自身のアカウント境界です。したがって、アカウント自体がリソース作成のコンテナであり、これを次の例のようなポリシーで明示的にモデル化できます。

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

しかし、アカウントも存在しない場合はどうなるでしょうか？ 顧客登録ワークフローを設計して、システム内に新しいアカウントを作成することもできます。その場合は、プロセスでアカウントを作成できる一番外側の境界を格納するコンテナが必要になります。このルートレベルのコンテナはシステム全体を表し、「システムルート」のような名前を付けることもできます。ただし、これが必要かどうか、またどのような名前を付けるかは、アプリケーションオーナーの判断に委ねられます。

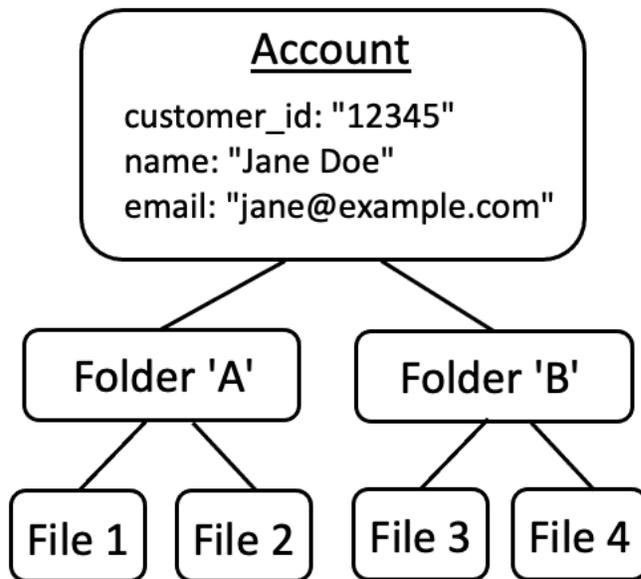
したがって、このサンプルアプリケーションでは、結果のコンテナ階層は次のようになります。



これは1つのサンプル階層です。他のものも有効です。覚えておくべきことは、リソースの作成は常にリソースコンテナのコンテキスト内で行われるということです。これらのコンテナはアカウントの境界のように暗黙的である場合があり、見落としがちです。承認モデルを設計するときは、これらの暗黙的な前提条件を必ず書き留めて、承認モデルで正式に文書化して表現できるようにしてください。

プリンシパルをリソースコンテナから分離する

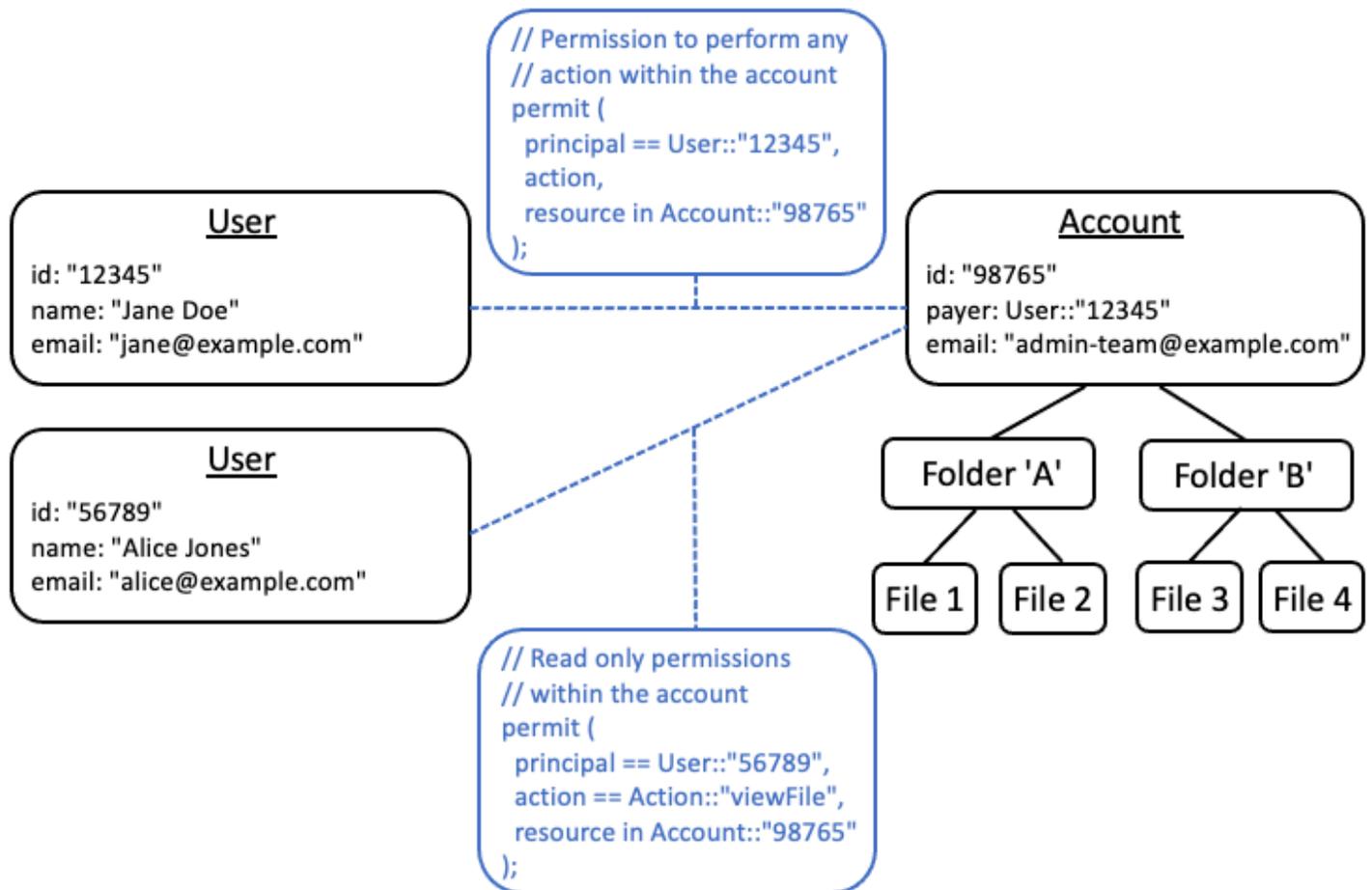
リソース階層を設計する際、特に消費者向けアプリケーションによくある傾向の1つは、顧客のユーザー ID を顧客アカウント内のリソースのコンテナとして使用することです。



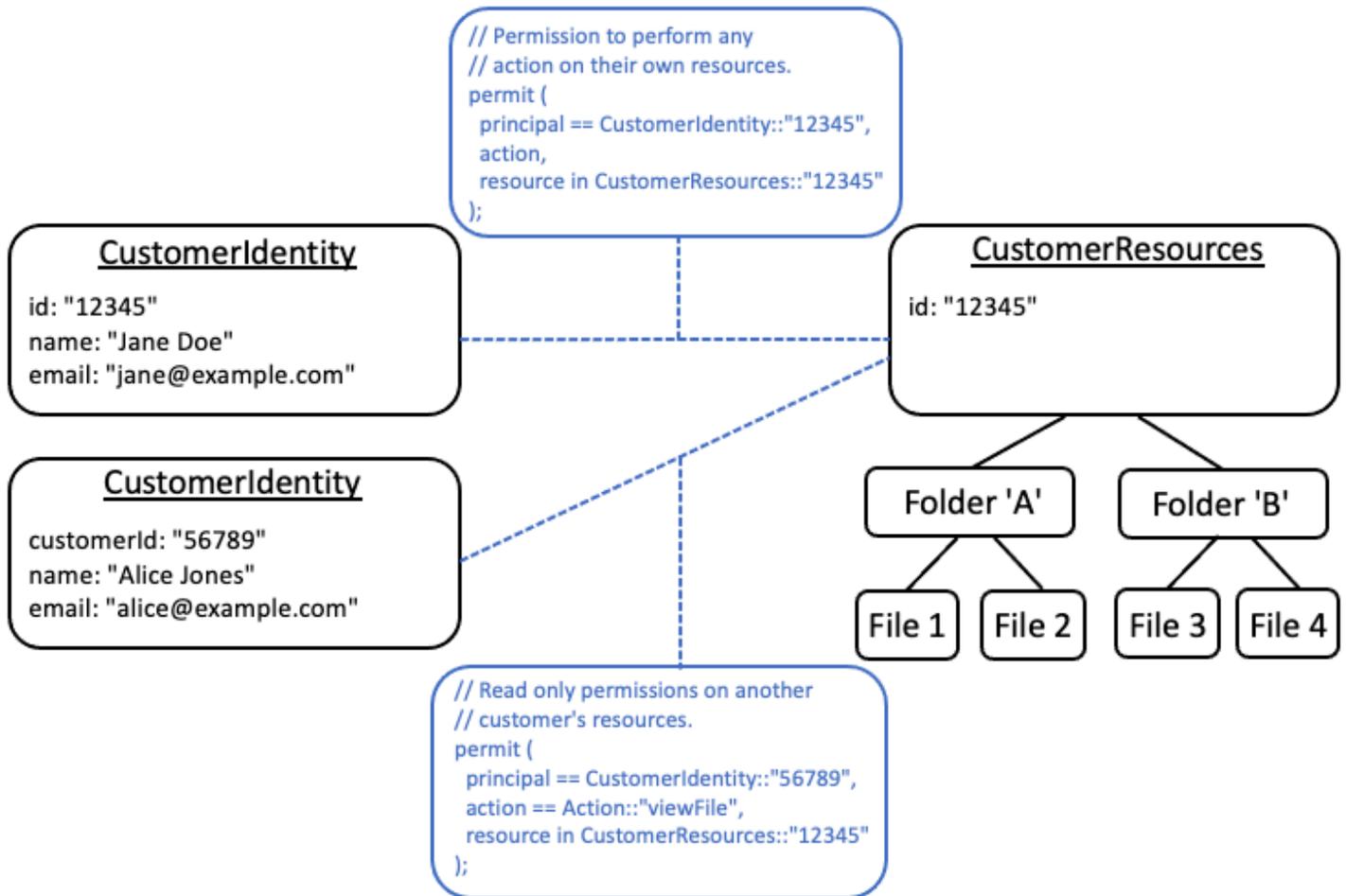
この戦略はアンチパターンとして扱うことをおすすめします。これは、機能が豊富なアプリケーションでは、アクセスを他のユーザーに委任する傾向が自然にあるためです。たとえば、他のユーザーがアカウントリソースを共有できる「ファミリー」アカウントを導入することを選択できます。同様に、企業の顧客は、複数の従業員をアカウントの一部の運営者に指定したい場合があります。また、アカウントの所有権を別のユーザーに移したり、複数のアカウントのリソースを統合したりする必要が生じる場合もあります。

ユーザー ID をアカウントのリソースコンテナとして使用すると、以前のシナリオを実現するのが難しくなります。さらに憂慮すべき点は、この方法で他のユーザーにアカウントコンテナへのアクセス権が付与されると、ジェーンの電子メールやログイン認証情報の変更など、ユーザー ID 自体を変更するためのアクセス権限を誤って付与されてしまう可能性があることです。

そのため、可能な場合は、プリンシパルをリソースコンテナから切り離し、「管理者権限」や「所有権」などの概念を使用してプリンシパルをリソースコンテナから分離し、プリンシパル間の接続をモデル化するのがより回復力のあるアプローチです。



既存のアプリケーションではこのような切り離されたモデルを実現できない場合は、承認モデルを設計する際にできる限りそれを模倣することを検討することをおすすめします。たとえば、ユーザー ID、ログイン資格情報、および所有するリソースをカプセル化する Customer という名前の 1 つの概念だけを持つアプリケーションは、これを Customer Identity の 1 つの論理エンティティ (名前、電子メールなどを含む) を含む認可モデルにマッピングできます。Customer Resources または Customer Account 用の別個の論理エンティティは、それらが所有するすべてのリソースの親ノードとして機能します。両方のエンティティは同じ Id を共有できますが、Type は異なります。



属性内に権限を埋め込まないでください

属性は承認決定の入力として使用するのがベストです。ユーザーに「PermittedFolders」という名前の属性を宣言するなどして、権限そのものを表すために属性を使用しないでください。

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

そして、その後にポリシー内でその属性を使用する:

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

このアプローチは、特定のプリンシパルが特定のフォルダにアクセスできる単純な承認モデルを、トレードオフを伴う属性ベースのアクセス制御 (ABAC) モデルに変えます。そのトレードオフの 1 つは、リソースに対する権限を誰が持っているかをすばやく判断するのが難しくなることです。前述の例では、特定のフォルダーに誰がアクセスできるかを判断するには、そのフォルダーが各自の属性にリストされているかどうかをすべてのユーザーに繰り返し確認する必要があります。そのためには、ユーザーがアクセス権限を持つ場合にアクセスを許可するポリシーがあることを特に認識した上で行う必要があります。

このアプローチのもう 1 つのリスクは、権限が 1 つの User レコードにまとめられている場合のスケーリング要因です。ユーザーが多くのものにアクセスできると、User レコードの累積サイズが大きくなり、データを保存しているシステムの上限に達する可能性があります。

代わりに、複数の個別のポリシーを使用してこのシナリオを表現し、繰り返しを最小限に抑えるためにポリシーテンプレートを使用することをおすすめします。

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

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

permit (
```

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

Verified Permissionsは、権限評価中に多数の個別のきめ細かいポリシーを効率的に処理できます。このような方法でモデル化すると、時間が経つにつれて管理しやすくなり、監査しやすくなります。

モデルではきめ細かな権限を、ユーザーインターフェースでは集約した権限を優先します

設計者が後で後悔することが多い戦略の1つは、ReadやWriteなどの非常に広範なアクションを含む認可モデルを設計し、後でより詳細なアクションが必要であることに気づくことです。より細かい詳細度の必要性は、よりきめ細かいアクセス制御を求める顧客のフィードバックによって、または最小権限のアクセス許可を奨励するコンプライアンスおよびセキュリティ監査によって促進される可能性があります。

きめ細かい権限があらかじめ定義されていないと、アプリケーションコードやポリシーステートメントをユーザーによりきめ細かく設定した権限に変更するために、複雑な変換が必要になる可能性があります。たとえば、以前にコースベースのアクションに対して承認されたアプリケーションコードを、詳細なアクションを使用するように変更する必要があります。さらに、移行を反映するようにポリシーを更新する必要があります。

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

このようなコストのかかる移行を回避するには、事前にきめ細かい権限を定義したほうがよいでしょう。ただし、後でエンドユーザーが多数のきめ細かい権限を理解せざるを得なくなった場合、特にほとんどの顧客がReadやWriteなどのきめ細かな制御に満足している場合、これはトレードオフにつながる可能性があります。両方の利点を最大限に活用するには、ポリシー テンプレートやアクショングループなどのメカニズムを使用して、きめ細かいアクセス許可を Read や Write などの事前定

義されたコレクションにグループ化できます。このアプローチを使用すると、顧客にはコースベースの権限のみが表示されます。しかし、舞台裏では、きめ細かなアクセス許可を粒度の細かいアクションの集合としてモデル化することで、アプリケーションの将来性を確保しています。顧客または監査人のどちらかが要求すると、きめ細かい権限が公開される可能性があります。

認証を問い合わせる他の理由も検討してください

通常、承認チェックはユーザーのリクエストと関連付けられます。このチェックは、ユーザーにそのリクエストを実行する権限があるかどうかを判断する方法です。ただし、認証データを使用してアプリケーションのインターフェースの設計に影響を与えることもできます。たとえば、エンドユーザーがアクセスできるリソースのみのリストを表示するホーム画面を表示したい場合があります。リソースの詳細を表示するときに、ユーザーがそのリソースに対して実行できる操作だけをインターフェースに表示したい場合があります。

このような状況では、承認モデルにトレードオフが生じる可能性があります。たとえば、属性ベースのアクセス制御 (ABAC) ポリシーに大きく依存していると、「誰が何にアクセスできるのか」という質問にすばやく答えることが難しくなる可能性があります。これは、その質問に答えるには、各ルールをすべてのプリンシパルとリソースと照らし合わせて調べ、一致するものがあるかどうかを判断する必要がありますからです。そのため、ユーザーがアクセスできるリソースのみを一覧表示するように最適化する必要がある製品では、ロールベースのアクセス制御 (RBAC) モデルを使用する場合があります。RBAC を使用すると、ユーザーに割り当てられているすべてのポリシーを繰り返し処理してリソースアクセスを決定するのが簡単になります。

テストベンチ

Verified Permissions テストベンチでは、Verified Permissions ポリシーに対して [認証リクエスト](#) を実行することで、そのポリシーをテストし、トラブルシューティングすることができます。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証するかどうかを判断します。認証リクエストをテストする際に、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 を使用して認証リクエストを行う場合、「追加のエンティティ」セクションに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提供できます。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。

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 <https://console.aws.amazon.com/verifiedpermissions/> 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証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証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証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)「追加のエンティティ」セクションで「エンティティを追加」を選択し、認証決定を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証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. [認証リクエストを実行] を選択して、ポリシーストア内の Cedar ポリシーの認証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<https://console.aws.amazon.com/verifiedpermissions/> 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、認証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、認証決定の際に評価されるエンティティの階層とその属性を入力します。
9. 「認証リクエストを実行」を選択して、ポリシーストア内の Cedar ポリシーの認証リクエストをシミュレートします。テストベンチには、要求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 での承認の実装

ポリシーストア、ポリシー、テンプレート、スキーマ、承認モデルを構築したら、Amazon Verified Permissions に対するリクエストの承認を開始する準備が整います。Verified Permissions 認証を実装するには、のポリシーの設定AWSとアプリケーション内の統合を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の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 DCR アクセス許可と API アクセス許可に役立ちます。

OneDrive アクセス許可

アプリケーション TAK 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のある正確な形式、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。例えば、ユーザーがサインインするときに、「資金の転送」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。例えば、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を変更できるかどうかを判断することができます。

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分散システムの一部であり、多くの場合、外部 APIs から情報を取得します。Verified Permissions が「移管資金」ボタンの表示を許可したバンキングアプリの例では、ユーザーが移管を開始するときに、より複雑な承認決定を行う必要があります。Verified Permissions は、転送先アカウントとして適格な転送先アカウントを一覧表示する API リクエストを承認し、転送先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](#) から取得されています。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。

認証用の API オペレーション

Verified Permissions API には、次の認証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 を使用した認証リクエストへのエンドポイントです。プリンシパル、アクション、リソース、コンテキスト、エンティティ要素を送信する必要があります。Verified Permissions は、ポリシーストアスキーマに対してリクエスト内のエンティティを検証します。その後、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、Amazon Cognito JSON ウェブトークン (JWTs) のユーザーデータから認証リクエストを生成します。Verified Permissions は、ポリシーストアの ID ソースとして Amazon Cognito 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プリンシパルに属性を入力します。

[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、[クォータの使用量](#)を最小限に抑え、複雑なネストされたアクションの承認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。1 つのリソースのバッチ認証を使用すると、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。1 つのプリンシパルのバッチ認証を使用すると、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。Verified Permissions は、リクエストごとに決定を返します。

認証モデルのテスト

アプリケーションをデプロイする際の Verified Permissions 認証決定の効果を理解するために、ポリシーの開発時に [テストベンチ](#) 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評価できます。テストベンチは、ポリシーストア内の認証リクエストとレスポンスを評価する AWS Management Console 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次のステップです。Verified Permissions API は [IsAuthorized](#)、[IsAuthorizedWithToken](#)、および [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 BatchIsAuthorized](#) として受け入れられます。認証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<https://console.aws.amazon.com/verifiedpermissions/>を開きます。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DigitalPetStore。
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[IsAuthorized](#)の からテストベンチリクエストを入力します。次の詳細では、DigitalPetStoreサンプルを参照する例 4 の条件を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、 を選択してDigitalPetStore::User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DigitalPetStore::Role、「顧客」と入力します。
 - c. リソースを「1234」の順序に設定します。プリンシパルが で動作しているリソースで、 を選択してDigitalPetStore::Order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と入力 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 を実行しているアクションで、 を選択しますDigitalPetStore::Action::"GetOrder"。
4. 認証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「満たされたポリシー」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。カスタマーロール - 注文の取得という説明の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に注意してください。

REST API

1. Verified Permissions コンソール<https://console.aws.amazon.com/verifiedpermissions/>を開きます。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DigitalPetStore。
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[IsAuthorized](#)の から、DigitalPetStoreサンプルを参照する例 4 のリクエストボディをコピーします。

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ボディに貼り付け、 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ように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Userである必要があるという条件を削除Resourceし、注文Bobを表示でき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を ActiveUsersまたは のメンバーにできるようにします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。

アプリケーションや AWS SDKsとの統合

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。承認モデルを策定してテストしたら、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証リクエストに入力する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 で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、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にユーザーの詳細が表示されます。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](#) から発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 に認証リクエストを生成します。Verified Permissions は、ポリシーストア内のポリシーに対してリクエストを評価し、認可の決定を返します。
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 「[のツール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](#)

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証の簡素化](#)から `IsAuthorized` 生成されています。

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

その他のデベロッパーリソース

- [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 アプリのカスタム承認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用の使用権限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証を簡素化する](#)

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。一連の送信元 IP アドレスからのみ、またはユーザーが MFA でサインインした場合にのみ、アクションを許可できます。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 認証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](#) から取得されています。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。

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 1 つを宣言します。

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

```
    "IPAddress": {
      "string": "192.0.2.178"
    },
    "Country": {
      "string": "United States of America"
    },
    "SSL": {
      "boolean": true
    }
  },
  "approvedBy": {
    "entityIdentifier": {
      "entityId": "Bob",
      "entityType": "DigitalPetStore::User"
    }
  }
}
```

認証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。

文字列

文字で囲まれた"文字、数字、または記号のシーケンス。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、このリクエストが顧客が過去の4つの注文を表示しようとした結果である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エンティティの属性をスキーマに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。

サンプルDigitalPetStoreアプリケーションでこのサンプルコンテキストをテストするには、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 DigitalPetStore ように を変更する

最初は、DigitalPetStore は非常に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ていません。このコンテキスト情報を含む承認リクエストの例を評価するには、ポリシーストアと承認リクエストに次の変更を加えます。

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。actions 次のように `GetOrder` の を更新します。

```
"GetOrder": {
  "memberOf": [],
  "appliesTo": {
    "resourceTypes": [
      "Order"
    ],
    "context": {
      "type": "Record",
      "attributes": {
        "UserAgent": {
          "required": true,
```

```
        "type": "String"
      },
      "approvedBy": {
        "name": "User",
        "required": true,
        "type": "Entity"
      },
      "AccountCodes": {
        "type": "Set",
        "required": true,
        "element": {
          "type": "Long"
        }
      },
      "RequestedOrderCount": {
        "type": "Long",
        "required": true
      },
      "MfaAuthorized": {
        "type": "Boolean",
        "required": true
      }
    }
  },
  "principalTypes": [
    "User"
  ]
}
}
```

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、次のようにスキーマに [commonType](#) コンストラクトを作成します。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。

Note

Verified Permissions ビジュアルスキーマエディタは現在、commonTypeコンストラクトをサポートしていません。スキーマに追加すると、ビジュアルモードでスキーマを表示できなくなります。

```
"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によって満たされる必要がある条件を設定します。既存の静的ポリシーに基づいて構築され、「カスタマーロール - 注文の取得」という説明が付きます。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.MfaAuthorized == true &&
  context.UserAgent like "*My UserAgent*" &&
  context.RequestedOrderCount <= 4 &&
  context.AccountCodes.contains(111122223333) &&
  context.NetworkInfo.Country like "*United States*" &&
  context.NetworkInfo.SSL == true &&
  context.NetworkInfo.IPAddress like "192.0.2.*" &&
  context.approvedBy in DigitalPetStore::Role::"Employee"
};
```

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を要求しました。

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 件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発信され、暗号化されたセッションにあり、ユーザーの 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する必要があります。承認リクエストの entities要素では、 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、この認証リクエストを Verified Permissions API オペレーション に提示できます [IsAuthorized](#)。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": {
      "MfaAuthorized": {
        "boolean": true
      }
    },
    "UserAgent": {
      "string": "My UserAgent 1.12"
    }
  }
}
```

```
  },
  "RequestedOrderCount": {
    "long": 4
  },
  "AccountCodes": {
    "set": [
      {"long": 111122223333},
      {"long": 444455556666},
      {"long": 123456789012}
    ]
  },
  "NetworkInfo": {
    "record": {
      "IPAddress": {"string": "192.0.2.178"},
      "Country": {"string": "United States of America"},
      "SSL": {"boolean": true}
    }
  },
  "approvedBy": {
    "entityIdentifier": {
      "entityId": "Bob",
      "entityType": "DigitalPetStore::User"
    }
  }
},
"entities": {
  "entityList": [
    {
      "identifier": {
        "entityType": "DigitalPetStore::User",
        "entityId": "Alice"
      },
      "attributes": {
        "memberId": {
          "string": "801b87f2-1a5c-40b3-b580-eacad506d4e6"
        }
      },
      "parents": [
        {
          "entityType": "DigitalPetStore::Role",
          "entityId": "Customer"
        }
      ]
    }
  ]
}
```

```
    },
    {
      "identifier": {
        "entityType": "DigitalPetStore::User",
        "entityId": "Bob"
      },
      "attributes": {
        "memberId": {
          "string": "49d9b81e-735d-429c-989d-93bec0bcfd8b"
        }
      },
      "parents": [
        {
          "entityType": "DigitalPetStore::Role",
          "entityId": "Employee"
        }
      ]
    },
    {
      "identifier": {
        "entityType": "DigitalPetStore::Order",
        "entityId": "1234"
      },
      "attributes": {
        "owner": {
          "entityIdentifier": {
            "entityType": "DigitalPetStore::User",
            "entityId": "Alice"
          }
        }
      },
      "parents": []
    }
  ],
  "policyStoreId": "PSEXAMPLEabcdefgh111111"
}
```

Amazon Verified Permissions のセキュリティ

AWS では、クラウドセキュリティを最優先事項としています。AWS のユーザーは、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を利用できます。

セキュリティは、AWS とユーザーの間の責任共有です。[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して説明しています。

- クラウドのセキュリティ - AWS は、AWS クラウドで AWS のサービスを実行するインフラストラクチャを保護する責任を担います。また、AWS は、ユーザー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](#)の一環として、サードパーティーの監査が定期的にセキュリティの有効性をテストおよび検証しています。Amazon Verified Permissions に適用されるコンプライアンスプログラムについては、「[コンプライアンスプログラムによるAWS対象サービス](#)」「」を参照してください。
- クラウド内のセキュリティ - ユーザーの責任は、使用する AWS のサービスに応じて異なります。またお客様は、データの機密性、企業要件、適用法令と規制などのその他の要因に対しても責任を担います。

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とセキュア化に役立つその他の AWS のサービスを使用する方法も学びます。

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

「AWS [責任共有モデル](#)」「」「」は、Amazon Verified Permissions 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を負います。顧客は、このインフラストラクチャでホストされて

い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、使用される AWS のサービスのセキュリティ構成と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のよくある質問](#)」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 ブログを参照してください。

- データを保護するため、AWS アカウント の認証情報を保護し、AWS IAM Identity Center または AWS Identity and Access Management(IAM) を使用して個々のユーザーをセットアップすることをお勧めします。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アクセス許可のみを各ユーザーに付与できます。
- 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。
 - AWS CloudTrail で API とユーザーアクティビティログをセットアップします。
 - AWS のサービス 内でデフォルトである、すべてのセキュリティ管理に加え、AWS の暗号化ソリューションを使用します。
 - Amazon Macie 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ます。これらは、Amazon S3 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。
 - 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機密情報やセンシティブ情報は、タグや [Name] フィールドなどの自由形式のフィールドに配置しないことを強くお勧めします。これには、コンソール、API、AWS CLI、または AWS SDK を使用して検証済み権限または他の AWS のサービス を操作する場合があります。名前に使用する自由記述のテキストフィールドやタグに入力したデータは、課金や診断ログに使用される場合があります。外部サーバーへ URL を供給する場合は、そのサーバーへのリクエストを検証するために、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AWS マネージドキーで自動的に暗号化するため、顧客が管理するキーを使用する必要はなく、サポートもされていません。

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 リソースへのアクセスを安全に制御するのに役立つ AWS のサービスです。IAM 管理者は、検証済みアクセス許可リソースの使用を認証 (サインイン) および許可 (アクセス許可を持つ) できるユーザーを制御します。IAMは追加料金なしでご利用いただけるAWS のサービスです。

トピック

- [対象者](#)
- [アイデンティティによる認証](#)
- [ポリシーを使用したアクセス権の管理](#)
- [Amazon Verified Permissions と IAM の連携方法](#)
- [Amazon Verified Permissions の アイデンティティベースのポリシー例](#)
- [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](#)

対象者

AWS Identity and Access Management (IAM) の使用方法は、検証済みアクセス許可で行う作業によって異なります。

サービスユーザー – Verified Permissions サービスを使用してジョブを実行する場合、管理者は必要な資格情報と権限を提供します。作業を行うためにより多くの検証済みアクセス許可機能を使用すると、追加のアクセス許可が必要になる場合があります。アクセスの管理方法を理解しておく、管理者に適切な許可をリクエストするうえで役立ちます。Verified Permissions の機能にアクセスできない場合は、「[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](#)」を参照してください。

サービス管理者 – 社内の Verified Permissions リソースを担当している場合は、Verified Permissions に対する完全なアクセス権があると思われます。サービスのユーザーがどの Verified Permissions やリソースにアクセスするかを決めるのは管理者の仕事です。その後で、サービスユーザーのアクセス許可を変更するために、IAM 管理者にリクエストを送信する必要があります。IAM の基本概念については、このページの情報を確認します。会社が Verified Permissions で IAM を使用する方法の詳細については、[Amazon Verified Permissions と IAM の連携方法](#) を参照してください。

IAM 管理者 – IAM 管理者の場合は、検証済みのアクセス許可へのアクセスを管理するポリシーを作成する方法について詳しく学ぶことができます。IAM で使用できる Verified Permissions のアイデンティティベースのポリシーの例を表示するには、「[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。

アイデンティティによる認証

認証とは、アイデンティティ認証情報を使用して AWS にサインインする方法です。AWS アカウントのルートユーザー、IAM ユーザー、または IAM ロールを引き受けることによって認証 (AWS にサインイン) を受ける必要があります。

ID ソースから提供された認証情報を使用して、フェデレーテッドアイデンティティとして AWS にサインインできます。AWS IAM Identity Center フェデレーテッドアイデンティティ の例としては、(IAM アイデンティティセンター) ユーザー、会社のシングルサインオン認証、Google または Facebook の認証情報などがあります。フェデレーテッドアイデンティティとしてサインインする場合、管理者は以前に IAM ロールを使用して ID フェデレーションを設定しています。フェデレーションを使用して AWS にアクセスする場合、間接的にロールを引き受けることとなります。

ユーザーのタイプに応じて、AWS Management Console または AWS アクセスポータルにサインインできます。AWS への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。

プログラムで AWS にアクセスする場合、AWS は Software Development Kit (SDK) とコマンドラインインターフェイス (CLI) を提供し、認証情報でリクエストに暗号で署名します。AWS ツールを使用しない場合は、リクエストに自分で署名する必要があります。推奨される方法を使用して自分でリクエストに署名する方法の詳細については、IAM ユーザー ガイドの「[AWS API リクエストの署名](#)」を参照してください。

使用する認証方法を問わず、追加のセキュリティ情報の提供が求められる場合もあります。例えば、AWS では多要素認証 (MFA) を使用してアカウントのセキュリティを高めることを推奨しています。詳細については、AWS IAM Identity Center ユーザー ガイドの「[多要素認証](#)」と、IAM ユーザーガイドの AWS における [多要素認証 \(MFA\) の使用](#) を参照してください。

AWS アカウントのルートユーザー

AWS アカウントを作成する場合は、そのアカウントのすべての AWS のサービスとリソースに対して完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。このアイデンティティは AWS アカウントのルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサインインすることによってアクセスできます。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。ルートユーザーの認証情報は保護し、ルートユーザーでしか実行できないタスクを実行するときに使用します。ルートユーザーとしてサインインする必要があるタスクの完全なリストについては、IAM ユーザー ガイドの「[ルートユーザー資格情報が必要なタスク](#)」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、管理者アクセスを必要とするユーザーを含む人間のユーザーに対し、ID プロバイダーとのフェデレーションを使用して、一時的な認証情報の使用により、AWS のサービスにアクセスすることを要求します。

フェデレーテッドアイデンティティ は、エンタープライズユーザーディレクトリ、ウェブ ID プロバイダー、AWS Directory Service、アイデンティティセンターディレクトリのユーザーか、または ID ソースから提供された認証情報を使用して AWS のサービス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 が AWS アカウントにアクセスすると、ロールが継承され、ロールは一時的な認証情報を提供します。

アクセスを一元管理する場合は、AWS IAM Identity Center を使用することをお勧めします。IAM アイデンティティセンター でユーザーとグループを作成するか、すべての AWS アカウントとアプリケーションで使用するために、独自の ID ソースで一連のユーザーとグループに接続して同期することもできます。IAM アイデンティティセンター の詳細については、「[AWS IAM Identity Center ユー](#)

「IAM ユーザーガイド」の「[What is 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、1人のユーザーまたは1つのアプリケーションに対して特定の権限を持つAWSアカウント内のアイデンティティです。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を保有するIAMユーザーを作成する代わりに、一時的な認証情報を使用することをお勧めします。ただし、IAMユーザーでの長期的な認証情報が必要な特定のユースケースがある場合は、アクセスキーをローテーション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「[長期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションする](#)」を参照してください。

[IAMグループ](#)は、ユーザーの集団を指定するアイデンティティです。グループとしてサインインすることはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できます。多数のユーザーグループがある場合、グループを使用することで権限の管理が容易になります。たとえば、IAM Admin という名前のグループを設定して、そのグループにIAMリソースを管理するアクセス許可を与えることができます。

ユーザーは、ロールとは異なります。ユーザーは1人の人または1つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユーザーには永続的な長期の認証情報がありますが、ロールでは一時的な認証情報が提供されます。詳細については、「IAMユーザーガイド」の「[IAMユーザー \(ロールではなく\) の作成が適している場合](#)」を参照してください。

IAM ロール

[IAMロール](#)は、特定の許可を持つ、AWSアカウント内のアイデンティティです。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。[ロールを切り替えて](#)、AWS Management ConsoleでIAM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ロールと一時的な認証情報は、次の状況で役立ちます。

- フェデレーションユーザーユーザーアクセス – フェデレーションアイデンティティに権限を割り当てるには、ロールを作成してそのロールの権限を定義します。フェデレーションアイデンティティが認証されると、そのアイデンティティはロールに関連付けられ、ロールで定義されている権限

限が付与されます。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 ID プロバイダーのロールの作成](#)」を参照してください。IAM アイデンティティセンターを使用する場合、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制御するために、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。権限セットの詳細については、「AWS IAM Identity Center ユーザーガイド」の「[権限セット](#)」を参照してください。

- 一時的な IAM ユーザー許可 – IAM ユーザーまたはロールは、IAM ロールを引き受けて、特定のタスクに対するさまざまな権限を一時的に引き受けることができます。
- クロスアカウントアクセス 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権を付与する主な方法は、ロールを使用することです。ただし、一部の AWS のサービスでは、(ロールをプロキシとして使用する代わりに) リソースにポリシーを直接アタッチできます。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。
- Amazon EC2上で実行されるアプリケーション – IAM ロールを使用して、EC2 インスタンス上で実行され、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。AWS ロールを EC2 インスタンスに割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスに添付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、IAM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権限を付与する](#)」を参照してください。

IAM ロールと IAM ユーザーのどちらを使用するかを確認するには、IAM ユーザー ガイドの「[いつ IAM ロール\(ユーザーではなく\)を作成するか](#)」を参照してください。

ポリシーを使用したアクセス権の管理

AWS でアクセス権を管理するには、ポリシーを作成して AWS アイデンティティまたはリソースにアタッチします。ポリシーは AWS のオブジェクトであり、アイデンティティやリソースに関連付けて、これらの権限を定義します。AWS は、プリンシパル(ユーザー、ルートユーザー、またはロールセッション) がリクエストを行うと、これらのポリシーを評価します。ポリシーでの権限により、リクエストが許可されるか拒否されるかが決まります。大半のポリシーは JSON ドキュメントとして AWS に保存されます。JSON ポリシードキュメントの構造と内容の詳細については、「IAM ユーザーガイド」の「[JSON ポリシー概要](#)」を参照してください。

管理者は AWS JSON 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。

デフォルトでは、ユーザーやロールに権限はありません。IAM管理者は、リソースに必要なアクションを実行するためのアクセス許可をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを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方法を問わず、アクションのアクセス許可を定義します。例えば、iam:GetRole アクションを許可するポリシーがあるとします。このポリシーがあるユーザーは、AWS Management Console、AWS CLI、または AWS API からロール情報を取得できます。

アイデンティティベースポリシー

アイデンティティベースポリシーは、IAM ユーザー、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 権限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、および条件を制御します。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。

アイデンティティベースポリシーは、さらにインラインポリシーまたはマネージドポリシーに分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれます。管理ポリシーは、AWS アカウント内の複数のユーザー、グループ、およびロールにアタッチできるスタンドアロンポリシーです。マネージドポリシーには、AWS マネージドポリシーとカスタマー管理ポリシーがあります。マネージドポリシーとインラインポリシーのどちらを選択する方法については、IAM ユーザーガイドの「[マネージド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、リソースにアタッチする 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 ロール信頼ポリシーや 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。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は、[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、フェデレーションユーザー、または AWS のサービスを含めることができます。

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で IAM の AWS マネージド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権限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。

Simple Storage Service (Amazon S3)、AWS WAF、および Amazon VPC は、ACL をサポートするサービスの例です。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 では、他の一般的ではないポリシータイプをサポートしています。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- **アクセス許可の境界** - アクセス許可の境界は、アイデンティティベースのポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる許可の上限を設定する高度な機能です。エンティティに権限の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとその権限の境界の共通部分になります。Principal フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、権限の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、許可は無効になります。アクセス許可の境界の詳細については、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)** - SCP は、AWS Organizations で組織や組織単位 (OU) の最大権限を指定する JSON ポリシーです。AWS Organizations は、顧客のビジネスが所有する複数の AWS アカウントをグループ化し、一元的に管理するサービスです。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP はメンバーアカウントのエンティティに対する権限を制限します (各 AWS アカウントのルートユーザーなど)。Organizations と SCP の詳細については、『AWS Organizations ユーザーガイド』の「[SCP の仕組み](#)」を参照してください。
- **セッションポリシー** - セッションポリシーは、ロールまたはフェデレーションユーザーの一時的なセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果としてセッションの権限の範囲は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポリシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合もあ

ります。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。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権限を理解するのがさらに難しくなります。複数のポリシータイプが関係する場合に、AWS がリクエストを許可するかどうか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 IAM の連携方法

IAM を使用して検証済みアクセス許可へのアクセスを管理する前に、検証済みアクセス許可で利用できる IAM 機能を確認してください。

Amazon Verified Permissions で使用できる IAM 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	いいえ
ポリシーアクション	あり
ポリシーリソース	はい
ポリシー条件キー	いいえ
ACL	なし
ABAC (ポリシー内のタグ)	いいえ
一時的な認証情報	あり
プリンシパル権限	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 サービスがほとんどの IAM 機能でどのように動作するかの詳細を取得するには、IAM ユーザーガイドの [IAM で動作する AWS サービス](#) 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

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 権限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、および条件を制御します。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 のアイデンティティベースのポリシーでは、許可または拒否するアクションとリソース、また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。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。

Verified Permissions の アイデンティティベースのポリシー例

Amazon Verified Permissions アイデンティティベースの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 ロール信頼ポリシーや 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。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

で定義されます。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。プリンシパルには、アカウント、ユーザー、ロール、フェデレーションユーザー、または AWS のサービスを含めることができます。

クロスアカウントアクセスを有効にするには、全体のアカウント、または別のアカウントの IAM エンティティを、リソースベースのポリシーのプリンシパルとして指定します。リソースベースのポリシーにクロスアカウントのプリンシパルを追加しても、信頼関係は半分しか確立されない点に注意してください。プリンシパルとリソースが異なる AWS アカウントにある場合、信頼できるアカウントの IAM 管理者は、リソースにアクセスするための許可をプリンシパルエンティティ (ユーザーまたはロール) に付与する必要があります。IAM 管理者は、アイデンティティベースのポリシーをエンティティにアタッチすることで権限を付与します。ただし、リソースベースのポリシーで、同じアカウントのプリンシパルへのアクセス権が付与されている場合は、アイデンティティベースのポリシーをさらに付与する必要はありません。詳細については、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との相違点](#)」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 あり

管理者は AWS JSON 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んなリソースにどんな条件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。ポリシーアクションの名前は通常、関連する AWS API オペレーションと同じです。一致する API オペレーションのない権限のみのアクションなど、いくつかの例外があります。また、ポリシーに複数アクションが必要なオペレーションもあります。これらの追加アクションは、依存アクションと呼ばれます。

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権限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については、「サービス認証リファレンス」の「[Amazon Verified Permissions によって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 の アイデンティティベースの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	あり
------------------	----

管理者は AWS JSON 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んなリソースにどんな条件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 Resource 要素は、アクションが適用される 1 つ以上のオブジェクトを指定します。ステートメントには、Resource または NotResource 要素を含める必要があります。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。これは、リソースレベルの権限と呼ばれる特定のリソースタイプをサポートするアクションに対して実行できます。

オペレーションのリスト化など、リソースレベルの権限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": "*" 
```

Amazon VPC リソースのタイプとその ARN のリストを確認するには、「サービス認証リファレンス」の「[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 AWS JSON 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んなリソースにどんな条件でアクションを実行できるかということです。

Condition 要素 (または Condition ブロック) を使用すると、ステートメントが有効になる条件を指定できます。Condition 要素はオプションです。equal や less than などの[条件演算子](#)を使用して条件式を作成することによって、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。

1つのステートメントに複数の Condition 要素を指定するか、1つの Condition 要素に複数のキーを指定すると、AWS は AND 論理演算子を使用してそれら进行评估します。単一の条件キーに複数の値を指定すると、AWS は OR 論理演算子を使用して条件进行评估します。ステートメントの権限が付与される前にすべての条件が満たされる必要があります。

条件を指定する際にプレースホルダー変数も使用できます。例えば IAM ユーザーに、IAM ユーザー名がタグ付けされている場合のみリソースにアクセスできる権限を付与することができます。詳細については、IAMユーザーガイドの「[IAMポリシーエレメント: 変数およびタグ](#)」を参照してください。

AWS はグローバル条件キーとサービス固有の条件キーをサポートしています。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権限を持つかを制御します。ACL はリソーススペ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 (ポリシー内のタグ) のサポート いいえ

属性ベースのアクセス制御 (ABAC) は、属性に基づいて権限を定義する認可戦略です。AWS では、これらの属性はタグと呼ばれます。タグは、IAM エンティティ (ユーザーまたはロール)、および多数のAWSリソースにアタッチできます。エンティティとリソースのタグ付けは、ABAC の最初の手順です。その後、プリンシパルのタグがアクセスしようとしているリソースのタグと一致した場合に操作を許可するように ABAC ポリシーを設計します。

ABAC は、急成長する環境やポリシー管理が煩雑になる状況で役立ちます。

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 [Condition 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 つの条件キーのすべてをサポートする場合、そのサービスでのサポート状況の値は「はい」になります。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 とは?](#)」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 での一時的な認証情報の使用

一時的な認証情報のサポート

あり

AWS のサービスには、一時的な認証情報を使用してサインインしても機能しないものがあります。どの AWS のサービスが一時認証情報と連携するかなどの追加情報については、IAMユーザーガイドの「[IAM と連携するAWS のサービス](#)」を参照してください。

ユーザー名とパスワード以外の方法で AWS Management Console にサインインする場合は、一時的な認証情報を使用していることになります。例えば、会社の Single Sign-On (SSO) リンクを使用して AWS にアクセスすると、そのプロセスは自動的に一時認証情報を作成します。また、ユーザーとしてコンソールにサインインしてからロールを切り替える場合も、一時的な認証情報が自動的に作成されます。ロールの切り替えに関する詳細については、IAMユーザーガイドの「[ロールへの切り替え \(コンソール\)](#)」を参照してください。

一時認証情報は、AWS CLI または AWS API を使用して手動で作成できます。作成後、一時的な認証情報を使用して AWS にアクセスできるようになります。AWS は、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することをお勧めします。詳細については、「[IAM の一時的なセキュリティ認証情報](#)」を参照してください。

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 あり

IAM ユーザーまたはロールを使用して AWS でアクションを実行するユーザーは、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行してから、別のサービスの別のアクションを開始することがあります。FAS は、AWS のサービスを呼び出すプリンシパルの権限を、AWS のサービスのリクエストと合わせて使用し、ダウンストリームのサービスに対してリクエストを行います。FAS リクエストは、サービスが、完了するために他の AWS のサービスまたはリソースとのやりとりを必要とするリクエストを受け取ったときにのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。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、IAM 内からサービスロールを作成、変更、削除できます。詳細については、「IAM ユーザーガイド」の「[AWS のサービスに許可を委任するロールの作成](#)」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

サービスリンクロールは、AWS のサービスにリンクされているサービスロールの一種です。サービスがロールを引き受け、ユーザーに代わってアクションを実行できるようになります。サービスリンクロールは、AWS アカウントに表示され、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、「[IAM と連携する AWS サービス](#)」を参照してください。表の中から、[サービスにリンクされたロール] 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。

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 Management Console、AWS Command Line Interface (AWS CLI)、または AWS API を使用してタスクを実行することもできません。IAM 管理者は、リソースに必要なアクションを実行するための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 JSON ポリシー ドキュメントの例を使用して 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「[IAM ポリシーの作成](#)」を参照してください。

各リソースタイプの ARN の形式を含む、Verified Permissions によって定義されたアクションとリソースタイプの詳細については、サービス認証リファレンスの「[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

- [ポリシーのベストプラクティス](#)
- [Verified Permissions コンソールの使用](#)
- [ユーザーが自分のアクセス許可を表示できるようにする方法](#)

ポリシーの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を実行すると、AWS アカウント に料金が発生する可能性があります。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 マネージドポリシーを使用して開始し、最小特権の権限に移行する – ユーザーとワークロードへの権限の付与を開始するには、多くの一般的なユースケースのために権限を付与する AWS マネージドポリシーを使用します。これらは AWS アカウントで使用できます。ユースケースに応じた AWS カスタマー マネージドポリシーを定義することで、権限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「[IAM のポリシーとアクセス許可](#)」を参照してください。

- IAMポリシーで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、アクションやリソースへのアクセスを制限できます。例えば、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。また、AWS CloudFormation などの特定の AWS のサービスを介して使用する場合、条件を使用して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。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、安全で機能的な権限を確保する。 – IAM Access Analyzer は、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) と IAM のベスト プラクティスに準拠しているかどうかを確認します。IAM Access Analyzer 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を作成できるようサポートします。詳細については、「IAMユーザーガイド」の「[IAM Access Analyzer 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 – AWS アカウント内の IAM ユーザーまたはルートユーザーを要求するシナリオがある場合は、セキュリティを強化するために MFA をオンにします。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「[MFA 保護 API アクセスの設定](#)」を参照してください。

IAM でのベストプラクティスの詳細については、「IAMユーザーガイド」の「[IAM でのセキュリティのベストプラクティス](#)」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。これらの許可は、AWS アカウントの Verified Permissions リソースに関する詳細をリストおよび表示することを許可する必要があります。最小限必要なアクセス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) では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権限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を許可します。

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、Verified Permissions *ConsoleAccess* または *ReadOnlyAWS* 管理ポリシーもエンティティ

にアタッチします。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。

ユーザーが自分のアクセス許可を表示できるようにする方法

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 か AWS API を使用してプログラマ的に、このアクションを完了する権限が含まれています。

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

Amazon Verified Permissions 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](#)
- [iam を実行する権限がありません。PassRole](#)
- [自分の AWS アカウント 以外のユーザーに Verified Permissions リソースへのアクセスを許可したい](#)

Verified Permissions でアクションを実行する権限がありません

「I am not authorized to perform an action in Amazon Bedrock」というエラーが表示された場合、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。

次の例は、mateojackson という IAM ユーザーがコンソールを使用して架空の *my-example-widget* リソースに関する詳細を表示しようとしたとき、架空の `verifiedpermissions:GetWidget` アクセス許可がない場合に発生するエラーを示しています。

```
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。管理者とは、サインイン認証情報を提供した担当者です。

iam を実行する権限がありません。PassRole

`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やサービスリンクロールを作成せずに、既存のロールを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権限が必要です。

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権限が必要です。Mary には、ロールをサービスに渡す権限がありません。

```
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、Mary 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。管理者とは、サインイン認証情報を提供した担当者です。

自分の AWS アカウント 以外のユーザーに Verified Permissions リソース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制御リスト (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 IAM の連携方法](#)」を参照してください。
- 所有している AWS アカウント 全体のリソースへのアクセス権を提供する方法については、「IAM ユーザーガイド」の「[所有している別の AWS アカウント アカウントへのアクセス権を IAM ユーザーに提供](#)」を参照してください。
- サードパーティーの AWS アカウント にリソースへのアクセス権を提供する方法については、「IAM ユーザーガイド」の「[第三者が所有する AWS アカウント へのアクセス権を付与する](#)」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でのロールとリソースベースのポリシーの使用の違いの詳細については、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との相違点](#) 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、「[コンプライアンスプログラム別の範囲](#)」の「AWS のサービス」と「」の「AWS のサービス」を参照し、関心のあるコンプライアンスプログラムを選択してください。一般的な情報については、「[AWS コンプライアンスプログラム](#)」を参照してください。

AWS Artifact 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。詳細については、「[Downloading Reports in AWS Artifact](#)」を参照してください。

AWS のサービスを使用する際のユーザーのコンプライアンス責任は、ユーザーのデータの機密性や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。AWS では、コンプライアンスに役立つ次のリソースを提供しています。

- [セキュリティとコンプライアンスのクイックスタートガイド](#) - これらのデプロイガイドでは、アーキテクチャ上の考慮事項について説明し、セキュリティとコンプライアンスに重点を置いたベースライン環境を AWS にデプロイするための手順を示します。
- [Amazon Web Servicesに関するHIPAAのセキュリティとコンプライアンスのためのアーキテクチャの設計ホワイトペーパー](#) - このホワイトペーパーは、企業が AWS を使用して HIPAA 対象アプリケーションを作成する方法を説明します。

Note

すべての AWS のサービスが HIPAA 適格であるわけではありません。詳細については、「[HIPAA 対応サービスのリファレンス](#)」を参照してください。

- [AWS コンプライアンスのリソース](#) - このワークブックおよびガイドのコレクションは、顧客の業界と拠点に適用されるものである場合があります。
- [AWS Customer Compliance Guide](#) — コンプライアンスの観点から見た責任共有モデルを理解できます。このガイドは、AWS のサービスを保護するためのベストプラクティスを要約したものであり、複数のフレームワーク (米国標準技術研究所 (NIST)、ペイメントカード業界セキュリティ標準評議会 (PCI)、国際標準化機構 (ISO) など) にわたるセキュリティ統制へのガイダンスがまとめられています。

- 「AWS Config デベロッパーガイド」の「[ルールでのリソースの評価](#)」 - AWS Config サービスは、自社のプラクティス、業界ガイドライン、および規制に対するリソースの設定の準拠状態を評価します。
- [AWS Security Hub](#) - この AWS のサービスは、AWS 内のセキュリティ状態の包括的なビューを提供します。Security Hub では、セキュリティコントロールを使用して AWS リソースを評価し、セキュリティ業界標準とベストプラクティスに対するコンプライアンスをチェックします。サポートされているサービスとコントロールのリストについては、「[Security Hub のコントロールリファレンス](#)」を参照してください。
- [AWS Audit Manager](#) - この AWS のサービスは、AWS の使用状況を継続的に監査して、リスクの管理方法や、規制および業界標準へのコンプライアンスの管理方法を簡素化するために役立ちます。

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ゾーンを中心に構築されています。AWS リージョン には、低レイテンシー、高いスループット、そして高度の冗長ネットワークで接続されている物理的に独立・隔離された複数のアベイラビリティゾーンがあります。アベイラビリティゾーンを使用すると、中断することなくゾーン間で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できます。アベイラビリティゾーンは、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、耐障害性、および拡張性が優れています。

Verified Permissions ポリシーストアを作成すると、個々の AWS リージョン 内に作成され、そのリージョンのアベイラビリティゾーンを構成するデータセンター全体に自動的に複製されます。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 とアベイラビリティ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 のモニタリング

モニタリングは、Amazon Verified Permissions やその他の 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ための重要な部分です。は、検証済みのアクセス許可を監視し、何か問題がある場合にレポートし、必要に応じて自動アクションを実行するための次の監視ツールを提供します。

- AWS CloudTrail は、AWS アカウントにより、またはそのアカウントに代わって行われた API コールや関連イベントを取得し、指定した Amazon S3 バケットにログファイルを配信します。AWS を呼び出したユーザーとアカウント、呼び出し元の IP アドレス、および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。

AWS CloudTrail を使用して Amazon Verified Permissions API 呼び出しをログに記録する

Amazon Verified Permissions は AWS CloudTrail、Verified Permissions のユーザー、ロール、または AWS サービスによって実行されたアクションを記録するサービスであると統合されています。は、Verified Permissions のすべての API コールをイベントとして CloudTrail 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 CloudTrail イベントなど、Amazon S3 バケットへの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、コンソールのイベント履歴で最新の CloudTrail イベントを表示できます。で収集された情報を使用して CloudTrail、Verified Permissions に対するリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

の詳細については CloudTrail、「[AWS CloudTrail ユーザーガイド](#)」を参照してください。

での検証済みアクセス許可情報 CloudTrail

CloudTrail アカウントを作成する AWS アカウントと、は 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 CloudTrail イベント履歴 の他の AWS サービス イベントとともに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、「[イベント履歴を使用した CloudTrail イベントの表示](#)」を参照してください。

Verified Permissions のイベントを含む、AWS アカウント 内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、はログファイル CloudTrail 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 ログで収集されたデータをより詳細に分析し、それに基づく対応を行うように他の 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:

- [「証跡作成の概要」](#)
- [CloudTrail でサポートされているサービスと統合](#)
- [の Amazon SNS 通知の設定 CloudTrail](#)
- [複数のリージョンからの CloudTrail ログファイルの受信と複数のアカウントからの CloudTrail ログファイルの受信](#)

すべての Verified Permissions アクションは によってログに記録 CloudTrail され、[「Amazon Verified Permissions API リファレンスガイド」](#)に記載されています。例えば、CreateIdentitySource、および ListPolicyStores アクションを呼び出すと DeletePolicy、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、誰がリクエストを生成したかという情報が含まれます。アイデンティティ情報は、以下を判別するために役立ちます。

- リクエストが、ルートまたは AWS Identity and Access Management (IAM) ユーザー認証情報のどちらを使用して送信されたかどうか。
- 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 サービスによって送信されたかどうか。

詳細については、「[CloudTrail userIdentity 要素](#)」を参照してください。

[IsAuthorized](#) や などのデータイベント [IsAuthorizedWithToken](#) は、証跡またはイベントデータストアの作成時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されている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 ログファイルには、1 つ以上のログエントリが含まれます。イベントは任意の送信元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 コール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。

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": "PSEXAMPLEEabcdefg111111"
},
"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/
PSEXAMPLEEabcdefg111111"
    }
],
"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": "b6edaeeee-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": "PSEXAMPLEEabcdefg111111"  
  },  
  "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/  
PSEXAMPLEEabcdefg111111"  
    }  
  ],  
  "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"
}
```

AWS CloudFormationによる Amazon Verified Permissions リソースの作成

Amazon Verified Permissions は、AWS リソースのモデル化と設定を支援するサービスである AWS CloudFormation と統合されており、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ます。必要なすべての AWS リソース (ポリシー ストアなど) を説明するテンプレートを作成すると、AWS CloudFormation がそれらのリソースをプロビジョニングして設定します。

AWS CloudFormation を使用すると、テンプレートを再利用して検証済みアクセス許可リソースを一貫して繰り返し設定できます。リソースを一度記述するだけで、同じリソースを複数の AWS アカウント とリージョンで何度でもプロビジョニングできます。

Important

Amazon Cognito ID は、Amazon Verified Permissions と同じ AWS リージョン のすべてで利用できるわけではありません。Amazon Cognito ID (Unrecognized resource types: AWS::Cognito::UserPool, AWS::Cognito::UserPoolClient など) に関する AWS CloudFormation からのエラーを受け取った場合は、Amazon Cognito ID が利用できる地理的に最も近い AWS リージョン 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 およびAWS 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、[AWS 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または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には、AWS CloudFormation スタックにプロビジョニングしたいリソースを記述します。JSON や YAML に不慣れな方は、AWS CloudFormation Designer を使えば、AWS CloudFormation テンプレートを使いこなすことができます。詳細については、「AWS CloudFormation ユーザーガイド」の「[AWS CloudFormation Designer とは](#)」を参照してください。

Verified Permissions は、AWS CloudFormation でのアイデンティティ ソース、ポリシー、ポリシー ストア、およびポリシー テンプレートの作成をサポートします。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 CloudFormation の詳細はこちら

AWS CloudFormation の詳細については、以下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](#)
- [AWS 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](#)

インターフェイスエンドポイント (AWS PrivateLink) を使用して Amazon Verified Permissions にアクセスする

AWS PrivateLink を使用して、VPC と Amazon Verified Permissions 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、AWS Direct Connect 接続のいずれかを使用せずに、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ト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 Guide (AWS PrivateLink ガイド)」の「[Access an AWS のサービス using an interface VPC endpoint](#) (インターフェイス VPC エンドポイントを使用してにアクセスする)」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` です。

Amazon Verified Permissions のクォータ

AWS アカウント 各サービスには、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。AWS 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 認証済み権限に関連するクォータは以下のとおりです。

トピック

- [リソースのクォータ](#)
- [階層のクォータ](#)
- [1秒あたりのオペレーションのクォータ](#)

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: 1,000	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	No	ポリシーストアのために定義できる ID ソースの最大数。
認可リクエストサイズ ¹	1 MB	No	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	No	各ポリシーの最大サイズ
スキーマサイズ	100,000 bytes	No	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	No	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。

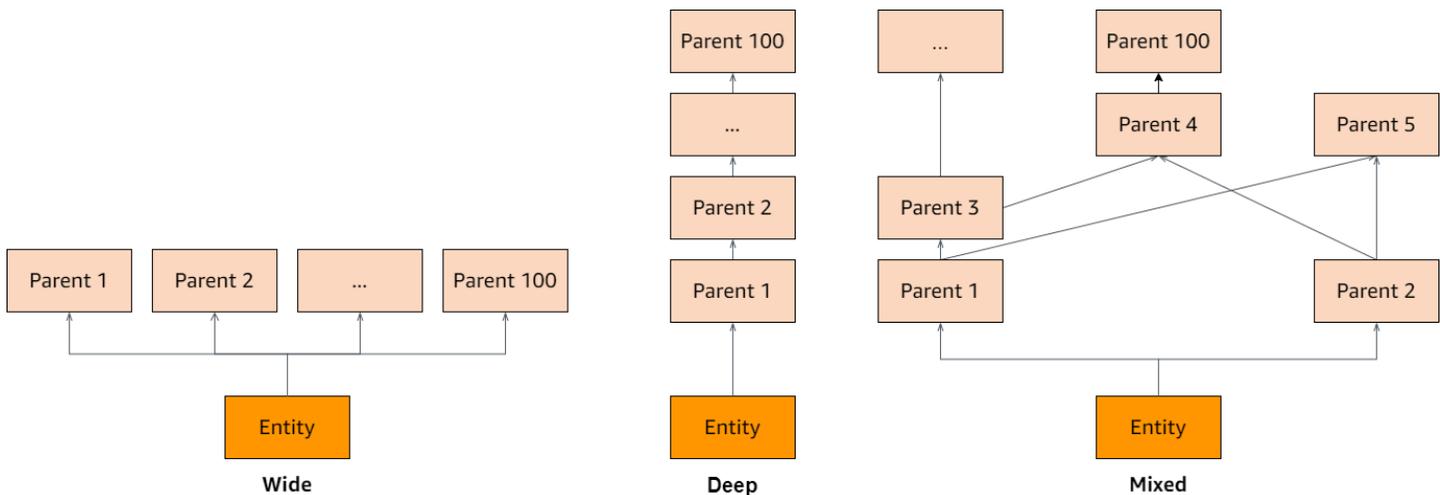
¹ 認証リクエストのクォータは、とで同じです。 [IsAuthorizedIsAuthorizedWithToken](#)

² 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。

階層のクォータ

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	No	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	No	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	No	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が API AWS リージョン オペレーションのクォータを超えた場合に、サービスエンドポイントへのリクエストを制限します。1 秒あたり

のリクエスト数がクォータを超えたり、書き込み操作を同時に行おうとしたりすると、Verified Permissions が例外を返すことがあります。[現在の RPS クォータは、Service Quotas で確認できます。](#)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、再試行と指数関数的バックオフに対応するようにアプリケーションを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](#)」と「[ワークロードの API スロットリングの管理と監視](#)」を参照してください。

名前	デフォルト	引き上げ可能	説明
BatchIsAuthorized 1 秒あたりのリクエスト数 (1 アカウント、1 リージョンあたり)	サポートされている各リージョン: 30	はい	1 BatchIsAuthorized 秒あたりの最大リクエスト数。
CreatePolicy 1 アカウントあたりのリージョンごとの 1 秒あたりのリクエスト数	サポートされている各リージョン: 10	はい	1 CreatePolicy 秒あたりの最大リクエスト数。
CreatePolicyStore 1 アカウントあたりのリージョンごとの 1 秒あたりのリクエスト数	サポートされている各リージョン: 1	No	1 CreatePolicyStore 秒あたりの最大リクエスト数。
CreatePolicyTemplate 1 アカウントあたりのリージョンごとの 1 秒あたりのリクエスト数	サポートされている各リージョン: 10	はい	1 CreatePolicyTemplate 秒あたりの最大リクエスト数。
DeletePolicy 1 アカウントあたりのリージョンごとの 1 秒あたりのリクエスト数	サポートされている各リージョン: 10	はい	1 DeletePolicy 秒あたりの最大リクエスト数。
DeletePolicyStore 1 アカウントあたりのリージョンごとの 1 秒あたりのリクエスト数	サポートされている各リージョン: 1	No	1 DeletePolicyStore 秒あたりの最大リクエスト数。

名前	デフォルト	引き上げ可能	説明
DeletePolicyTemplate 1 アカウントあたりのリージョンごとの 1 秒あたりのリクエスト数	サポートされている各リージョン: 10	はい	1 DeletePolicyTemplate 秒あたりの最大リクエスト数。
GetPolicy 1 アカウントあたりのリージョンごとの 1 秒あたりのリクエスト数	サポートされている各リージョン: 10	はい	1 GetPolicy 秒あたりの最大リクエスト数。
GetPolicyTemplate 1 アカウントあたりのリージョンごとの 1 秒あたりのリクエスト数	サポートされている各リージョン: 10	はい	1 GetPolicyTemplate 秒あたりの最大リクエスト数。
GetSchema 1 アカウントあたりのリージョンごとの 1 秒あたりのリクエスト数	サポートされている各リージョン: 10	はい	1 GetSchema 秒あたりの最大リクエスト数。
IsAuthorized 1 アカウントあたりのリージョンごとの 1 秒あたりのリクエスト数	サポートされている各リージョン: 200	はい	1 IsAuthorized 秒あたりの最大リクエスト数。
IsAuthorizedWithToken 1 アカウントあたりのリージョンごとの 1 秒あたりのリクエスト数	サポートされている各リージョン: 200	はい	1 IsAuthorizedWithToken 秒あたりの最大リクエスト数。
ListPolicies 1 アカウントあたりのリージョンごとの 1 秒あたりのリクエスト数	サポートされている各リージョン: 10	はい	1 ListPolicies 秒あたりの最大リクエスト数。
ListPolicyStores 1 アカウントあたりのリージョンごとの 1 秒あたりのリクエスト数	サポートされている各リージョン: 10	はい	1 ListPolicyStores 秒あたりの最大リクエスト数。

名前	デフォルト	引き上げ可能	説明
ListPolicyTemplates 1 アカウントあたりのリージョンごとの 1 秒あたりのリクエスト数	サポートされている各リージョン: 10	はい	1 ListPolicyTemplates 秒あたりの最大リクエスト数。
PutSchema 1 アカウントあたりのリージョンごとの 1 秒あたりのリクエスト数	サポートされている各リージョン: 10	はい	1 PutSchema 秒あたりの最大リクエスト数。
UpdatePolicy 1 アカウントあたりのリージョンごとの 1 秒あたりのリクエスト数	サポートされている各リージョン: 10	はい	1 UpdatePolicy 秒あたりの最大リクエスト数。
UpdatePolicyTemplate 1 アカウントあたりのリージョンごとの 1 秒あたりのリクエスト数	サポートされている各リージョン: 10	はい	1 UpdatePolicyTemplate 秒あたりの最大リクエスト数。

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
AWS CloudFormation の統合	Verified Permissions は、AWS CloudFormation でのアイデンティティ ソース、ポリシー、ポリシー ストア、およびポリシー テンプレートの作成をサポートします。	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。