aws

ユーザーガイド

AWS IoT TwinMaker

AWS IoT TwinMaker: ユーザーガイド

Table of Contents

とは何ですか AWS IoT TwinMaker?

AWS IoT TwinMaker は、 AWS IoT 物理システムとデジタルシステムの運用可能なデジタルツインを構築するために使用できるサービスです。 AWS IoT TwinMaker 現実世界のさまざまなセンサー、カメラ、エンタープライズアプリケーションからの測定と分析を使用してデジタルビジュアライゼーションを作成し、実際の工場、建物、または産業プラントの追跡に役立ちます。この実際のデータを使用して、オペレーションのモニタリング、エラーの診断と修正、およびオペレーションの最適化を行うことができます。

デジタルツインは、システムとそのすべての物理コンポーネントとデジタルコンポーネントをライブデジタルで表現したものです。データによって動的に更新され、システムの実際の構造、状態、動作を模倣します。これを利用してビジネスの成果を上げることができます。

エンドユーザーは、ユーザーインターフェースアプリケーションを使用してデジタルツインからのデータを操作します。

仕組み

デジタルツインを作成するための最小要件を満たすには、以下を実行する必要があります。

- デバイス、機器、スペース、プロセスを物理的な場所でモデル化します。
- これらのモデルを、センサーデータのカメラフィードなどの重要なコンテキスト情報を保存するデータソースに接続します。
- ビジネス上の意思決定をより効率的に行えるよう、ユーザーがデータやインサイトを理解するのに役立つビジュアライゼーションを作成します。
- デジタルツインをエンドユーザーが利用できるようにして、ビジネスの成果を高めましょう。

AWS IoT TwinMaker 以下の機能を提供することで、これらの課題に対処します。

- エンティティコンポーネントシステムナレッジグラフ:デバイス、機器、スペース、 AWS IoT TwinMaker プロセスをナレッジグラフでモデリングするためのツールを提供します。

  このナレッジグラフにはシステムに関するメタデータが含まれており、さまざまな場所にあるデータに接続できます。 AWS IoT TwinMaker には、 AWS IoT SiteWise および Kinesis Video Streams に保存されているデータ用のコネクタが組み込まれています。他の場所に保存されているデータへのカスタムコネクタを作成することも可能です。

ナレッジグラフとコネクタを組み合わせることで、異なる場所にあるデータをクエリするための単一のインターフェースが提供されます。

- Scene Composer: AWS IoT TwinMaker コンソールには、3D でシーンを作成するためのシーン構成ツールがあります。ウェブ表示用に最適化され、.gltfまたは.glb形式に変換された3D/CADモデルをアップロードします。次に、シーンコンポーザーを使用して複数のモデルを1つのシーンに配置し、それぞれの操作を視覚的に表現します。

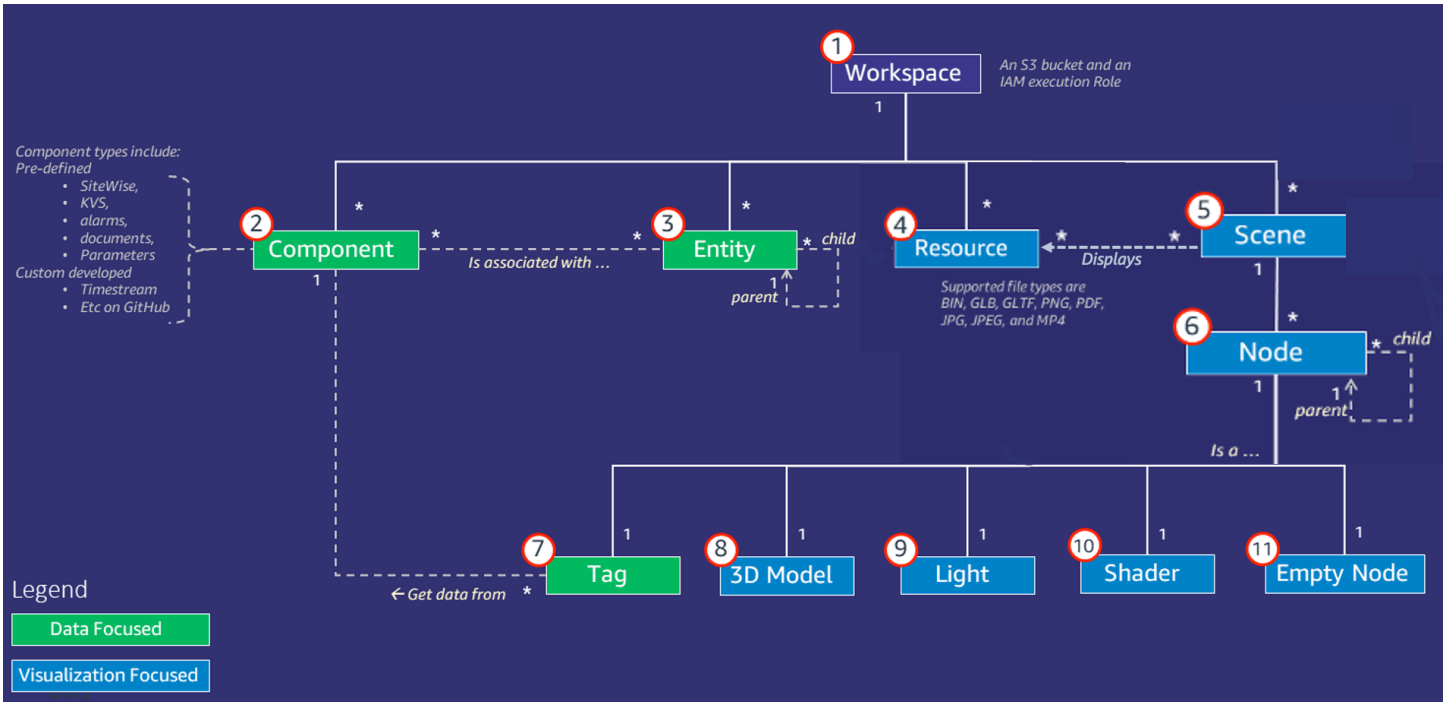
  シーン内のデータをオーバーレイすることもできます。たとえば、センサーからの温度データに接続するタグをシーンの場所に作成できます。これにより、データが位置と関連付けられます。

- アプリケーション: AWS IoT TwinMaker エンドユーザー向けのダッシュボードアプリケーションの構築に使用できる Grafana と Amazon Managed Grafana 用のプラグインを提供します。

- サードパーティツール: AWS IoT TwinMaker Mendixはと提携して、産業用IoT向けの完全なソリューションを提供しています。Kinesis Video Streams AWS などのサービスで Mendix ローコードアプリケーション開発プラットフォーム (LCAP) を使い始めるには AWS IoT TwinMaker、ワークショップ「リーン・デイリー・マネジメント・アプリケーション開発プラットフォーム (LCAP) ウィズ・メンディックス」をご覧ください。 AWS IoT TwinMaker AWS IoT SiteWise

主要コンセプトとコンポーネント

次の図は、の主要概念がどのように組み合わされているかを示しています。 AWS IoT TwinMaker



> **Note**
>
> 図中のアスタリスク (*) は関係を示しています。 one-to-many 各リレーションシップのクォータについては、エンドポイントとクォータをご覧ください。AWS IoT TwinMaker

以下のセクションでは、図に示されている概念について説明します。

ワークスペース

ワークスペースは、デジタルツインアプリケーションの最上位コンテナです。デジタルツイン用のエンティティ、コンポーネント、シーンアセット、その他のリソースの論理セットをこのワークスペース内で作成します。また、デジタルツインアプリケーションとそれに含まれるリソースへのアクセスを管理するためのセキュリティ境界としても機能します。各ワークスペースは、ワークスペースデータが保存されるAmazon S3バケットにリンクされます。IAMロールを使用してワークスペースへのアクセスを制限します。

ワークスペースには複数のコンポーネント、エンティティ、シーン、リソースを含めることができます。コンポーネントタイプ、エンティティ、シーン、またはリソースは1つのワークスペース内にのみ存在します。

エンティティ-コンポーネントモデル

AWS IoT TwinMaker には、 entity-component-based ナレッジグラフを使用してシステムをモデル化するためのツールが用意されています。エンティティコンポーネントアーキテクチャを使用して、物理システムの表現を作成できます。このエンティティコンポーネントモデルは、エンティティ、コンポーネント、リレーションシップで構成されています。エンティティコンポーネントシステムの詳細については、エンティティコンポーネントシステムを参照してください。

エンティティ

エンティティは、デジタルツイン内の要素をデジタル表現したもので、その要素の機能をキャプチャします。この要素には、物理的な機器、コンセプト、プロセスなどがあります。エンティティにはコンポーネントが関連付けられています。これらのコンポーネントは、関連するエンティティのデータとコンテキストを提供します。

を使用すると AWS IoT TwinMaker、エンティティをカスタム階層に整理して、より効率的に管理できます。エンティティとコンポーネントシステムのデフォルトビューは階層型です。

コンポーネント

コンポーネントはシーン内のエンティティのコンテキストとデータを提供します。エンティティにコンポーネントを追加します。コンポーネントの有効期間はエンティティの有効期間と関連しています。

コンポーネントは、ドキュメントのリストや地理的位置の座標などの静的データを追加できます。また、や他の時系列クラウドヒストリアンなどの時系列データを含むシステムなど AWS IoT SiteWise 、他のシステムに接続する機能を持つこともできます。

コンポーネントは、データソースと AWS IoT TwinMakerの接続を記述したJSONドキュメントによって定義されます。コンポーネントには、外部データソースやに組み込まれているデータソースを記述できます。 AWS IoT TwinMakerコンポーネントは、JSONドキュメントで指定されているLambda関数を使用して外部データソースにアクセスします。ワークスペースには多数のコンポーネントを含めることができます。コンポーネントは、関連するエンティティを通じてタグにデータを提供します。

AWS IoT TwinMaker には、コンソールから追加できる組み込みコンポーネントがいくつか用意されています。独自のカスタムコンポーネントを作成して、タイムストリームテレメトリや地理空間座標などのデータソースに接続することもできます。例としては、 TimeStreamテレメトリ、地理空間コンポーネント、Snowflakeなどのサードパーティのデータソースへのコネクタなどがあります。

AWS IoT TwinMaker には、一般的なユースケース向けに以下の種類の組み込みコンポーネントが用意されています。

- ドキュメント、指定したURLにあるユーザーマニュアルや画像など。
- 時系列、 AWS IoT SiteWiseからのセンサーデータなど。
- アラーム、外部データソースからの時系列アラームなど。
- ビデオ、Kinesis Video Streamsに接続されたIPカメラからのビデオ。
- カスタムコンポーネント、追加のデータソースに接続するためのカスタムコンポーネント。たとえば、カスタムコネクタを作成して、外部に保存されている時系列データに AWS IoT TwinMaker エンティティを接続できます。

データソース

データソースはデジタルツインのソースデータの場所です。 AWS IoT TwinMaker 次の 2 種類のデータソースをサポートします。

- 階層コネクタ、これにより、外部モデルを AWS IoT TwinMakerに継続的に同期できます。
- 時系列コネクタ、これにより、 AWS IoT SiteWiseなどの時系列データベースに接続できます。

プロパティ

プロパティは、コンポーネントに含まれる、静的な値と時系列に連動する値の両方です。エンティティにコンポーネントを追加すると、コンポーネント内のプロパティにエンティティの現在の状態に関する詳細が記述されます。

AWS IoT TwinMaker 次の 3 種類のプロパティをサポートします。

- 単一値、 non-time-seriesプロパティ — これらのプロパティは通常、静的なキーと値のペアで、AWS IoT TwinMaker 関連するエンティティのメタデータと一緒に直接格納されます。
- 時系列プロパティ — AWS IoT TwinMaker これらのプロパティの時系列ストアへの参照を保存します。デフォルトは最新の値です。
- リレーションシッププロパティ—これらのプロパティには、別のエンティティまたはコンポーネントへの参照が格納されます。たとえば、`seen_by`は、カメラエンティティを、そのカメラによって直接視覚化される別のエンティティに関連付けることができるリレーションシップコンポーネントです。

統合データクエリインターフェイスを使用して、異種データソース間でプロパティ値をクエリできます。

視覚化

AWS IoT TwinMaker これを使用して、デジタルツインを 3 次元で表現し、それを Grafana に表示します。シーンを作成するには、既存のCADまたはその他の3Dファイルタイプを使用します。次に、データオーバーレイを使用してデジタルツインに関連するデータを追加します。

シーン

シーンは、接続先のデータを視覚的に把握できる 3 次元表現です。 AWS IoT TwinMakerシーンは、環境全体で単一のgltf (GL Transmission Format) またはglb 3Dモデルを使用して作成することも、複数のモデルを組み合わせて作成することもできます。シーンには、シーンの注目ポイントを示すタグも含まれています。

シーンはビジュアライゼーションの最上位のコンテナです。シーンは1つ以上のノードで構成されています。

ワークスペースには複数のシーンを含めることができます。たとえば、ワークスペースには施設の各フロアにつき1つのシーンを含めることができます。

リソース

シーンにはリソースが表示され、コンソールにはノードとして表示されます。 AWS IoT TwinMaker シーンには多数のリソースが含まれる場合があります。

リソースとは、シーンを作成するために使用される画像やglTFベースの3次元モデルのことです。リソースは1つの機器を表すことも、サイト全体を表すこともできます。

リソースをシーンに配置するには、.gltfまたは.glbファイルをワークスペースリソースライブラリにアップロードし、シーンに追加します。

ユーザーインタフェースを改善

AWS IoT TwinMaker を使用すると、センサーデータなどの重要なコンテキストや情報をシーン内の場所に追加するデータオーバーレイでシーンを拡張できます。

ノード：ノードはタグ、ライト、3次元モデルのインスタンスです。空にしてシーン階層に構造を追加することもできます。たとえば、複数のノードを1つの空のノードにまとめることができます。

タグ：タグは、（エンティティを通じて）コンポーネントからのデータを表すノードの一種です。タグは、1つのコンポーネントにのみ関連付けることができます。タグは、シーンの特定のx,y,z座標位置に追加される注釈です。タグは、エンティティプロパティを使用してこのシーンパーツをナレッジグラフに接続します。タグを使用して、シーン内のアイテム（アラームなど）の動作や外観を設定できます。

ライト：シーンにライトを追加して特定のオブジェクトにピントを合わせたり、オブジェクトに影を落として物理的な位置を示すことができます。

3次元モデル：3次元モデルは、リソースとしてインポートされた.gltfまたは.glbファイルを視覚的に表現したものです。

> **Note**
>
> AWS IoT TwinMaker 重大な人身傷害や死亡につながったり、環境や物的損害を引き起こす可能性のある危険な環境や重要なシステムの運用における使用、またはそれらに関連する使用を目的としたものではありません。

の使用を通じて収集されたデータは、 AWS IoT TwinMaker その使用事例に応じて正確性を評価する必要があります。 AWS IoT TwinMaker 物理システムが安全に動作しているかどうかを評価する目的で、人間が物理システムを監視する代わりに使用すべきではありません。

の開始方法 AWS IoT TwinMaker

このセクションのトピックでは、以下を行う方法について説明します。

- 新しいワークスペースを作成して設定する。
- エンティティを作成してコンポーネントを追加する。

前提条件:

最初のワークスペースとシーンを作成するには、次の AWS リソースが必要です。

- AWS アカウント。
- の IAM サービスロール AWS IoT TwinMaker。このロールは、AWS IoT TwinMaker コンソール で新しい AWS IoT TwinMaker ワークスペースを作成すると、デフォルトで自動的に生成されます。

  で新しい IAM サービスロール AWS IoT TwinMaker を自動的に作成することを選択しない場合は、作成済みのロールを指定する必要があります。

  このサービスロールを作成、管理する手順については、「???」を参照してください。

  IAM サービスロールの詳細については、「AWS のサービスにアクセス許可を委任するロールの作成」を参照してください。

  > **Important**
  >
  > このサービスロールには、サービスが Amazon S3 バケットを読み書きするためのアクセス許可を付与するポリシーがアタッチされている必要があります。 は、このロール AWS IoT TwinMaker を使用して、ユーザーに代わって他のサービスにアクセスします。また、サービスがロールを引き受け AWS IoT TwinMaker られるように、このロールと の間に信頼関係を割り当てる必要があります。ツインが他の AWS サービスとやり取りする場合は、それらのサービスに必要なアクセス許可も追加します。

トピック

- AWS IoT TwinMakerのサービスロールを作成、管理する
- ワークスペースの作成
- 最初のエンティティを作成する

- AWS アカウントのセットアップ

AWS IoT TwinMakerのサービスロールを作成、管理する

AWS IoT TwinMaker では、サービスロールを使用して、ユーザーに代わって他のサービスのリソースにアクセスすることを許可する必要があります。このロールには、 との信頼関係が必要です AWS IoT TwinMaker。ワークスペースを作成したら、このロールをワークスペースに割り当てる必要があります。このトピックは、一般的なシナリオでアクセス許可を構成する方法を示すポリシーの例を含んでいます。

信頼を割り当てる

次のポリシーは、ロールと の間に信頼関係を確立します AWS IoT TwinMaker。この信頼関係をワークスペースに使用するロールに割り当てます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iottwinmaker.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Amazon S3 のアクセス許可

次のポリシーでは、Amazon S3 バケットの読み書きをロールで許可します。ワークスペースは Amazon S3 にリソースを格納するため、Amazon S3 のアクセス許可は、すべてのワークスペースで必要です。

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

```
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucket*",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:DeleteObject"
      ],
      "Resource": [
        "arn:aws:s3:::*/DO_NOT_DELETE_WORKSPACE_*"
      ]
    }
  ]
}
```

> **Note**
>
> ワークスペースを作成すると、 は、ワークスペースで使用されていることを示すファイルを Amazon S3 バケットに AWS IoT TwinMaker 作成します。このポリシーは、ワークスペースを削除するときにそのファイルを削除する AWS IoT TwinMaker アクセス許可を付与します。
>
> AWS IoT TwinMaker は、ワークスペースに関連する他のオブジェクトを配置します。ワークスペースを削除するときは、お客様自身でこれらのオブジェクトも削除する必要があります。

特定の Amazon S3 バケットにアクセス許可を割り当てる

AWS IoT TwinMaker コンソールでワークスペースを作成するときに、 で AWS IoT TwinMaker Amazon S3 バケットを作成するように選択できます。このバケットに関する情報は、次の AWS CLI コマンドを使用して確認できます。

```
  aws iottwinmaker get-workspace --workspace-id workspace name
```

次の例は、このコマンドの出力形式を示しています。

```
{
    "arn": "arn:aws:iottwinmaker:region:account Id:workspace/workspace name",
    "creationDateTime": "2021-11-30T11:30:00.000000-08:00",
    "description": "",
    "role": "arn:aws:iam::account Id:role/service role name",
    "s3Location": "arn:aws:s3:::bucket name",
    "updateDateTime": "2021-11-30T11:30:00.000000-08:00",
    "workspaceId": "workspace name"
}
```

特定の Amazon S3 バケットにアクセス許可を割り当てるようにポリシーを更新するには、*#####*の値を使用します。

次のポリシーでは、ロールによる特定の Amazon S3 バケットの読み書きを許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucket*",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::bucket name",
        "arn:aws:s3:::bucket name/*"
      ]
    },
    {
      "Effect": "Allow",
```

```
      "Action": [
        "s3:DeleteObject"
      ],
      "Resource": [
        "arn:aws:s3:::iottwinmakerbucket/DO_NOT_DELETE_WORKSPACE_*"
      ]
    }
  ]
}
```

ビルトインコネクタのアクセス許可

ワークスペースが組み込みコネクタを使用して他の AWS サービスとやり取りする場合は、このポリシーにそれらのサービスのアクセス許可を含める必要があります。com.amazon.iotsitewise.connector コンポーネントタイプを使用する場合は、 AWS IoT SiteWiseのアクセス許可を含める必要があります。コンポーネントタイプの詳細については、「???」を参照してください。

> **Note**
>
> カスタムコンポーネントタイプを使用して他の AWS サービスとやり取りする場合は、コンポーネントタイプに関数を実装する Lambda 関数を実行するアクセス許可をロールに付与する必要があります。詳細については、「???」を参照してください。

次の例は、ポリシー AWS IoT SiteWise に を含める方法を示しています。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucket*",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:PutObject"
      ],
      "Resource": [
```

```
        "arn:aws:s3:::bucket name",
        "arn:aws:s3:::bucket name/*"
      ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "iotsitewise:DescribeAsset"
        ],
        "Resource": "asset ARN"
        },
    {
        "Effect": "Allow",
        "Action": [
            "iotsitewise:DescribeAssetModel"
        ],
        "Resource": "asset model ARN"
        },
    {
      "Effect": "Allow",
      "Action": [
        "s3:DeleteObject"
      ],
      "Resource": [
        "arn:aws:s3:::*/DO_NOT_DELETE_WORKSPACE_*"
      ]
    }
  ]
}
```

com.amazon.iotsitewise.connector コンポーネントタイプを使用していて、 からプロパティデータを読み取る必要がある場合は AWS IoT SiteWise、ポリシーに次のアクセス許可を含める必要があります。

```
...
{
    "Action": [
        "iotsitewise:GetPropertyValueHistory",
    ],
    "Resource": [
        "AWS IoT SiteWise asset resource ARN"
```

```
    ],
    "Effect": "Allow"
},
...
```

com.amazon.iotsitewise.connector コンポーネントタイプを使用していて、 にプロパティデータを書き込む必要がある場合は AWS IoT SiteWise、ポリシーに次のアクセス許可を含める必要があります。

```
...
{
    "Action": [
        "iotsitewise:BatchPutPropertyValues",
    ],
    "Resource": [
        "AWS IoT SiteWise asset resource ARN"
    ],
    "Effect": "Allow"
},
...
```

com.amazon.iotsitewise.connector.edgevideo コンポーネントタイプを使用する場合は、 AWS IoT SiteWise および Kinesis Video Streams のアクセス許可を含める必要があります。次のポリシー例は、ポリシーに AWS IoT SiteWise および Kinesis Video Streams アクセス許可を含める方法を示しています。

```
...
{
    "Action": [
        "iotsitewise:DescribeAsset",
        "iotsitewise:GetAssetPropertyValue"
    ],
    "Resource": [
        "AWS IoT SiteWise asset resource ARN for the Edge Connector for Kinesis Video
 Streams"
    ],
    "Effect": "Allow"
},
```

```
{
    "Action": [
        "iotsitewise:DescribeAssetModel"
    ],
    "Resource": [
        "AWS IoT SiteWise model resource ARN for the Edge Connector for Kinesis Video
 Streams"
    ],
    "Effect": "Allow"
},
{
    "Action": [
        "kinesisvideo:DescribeStream"
    ],
    "Resource": [
        "Kinesis Video Streams stream ARN"
    ],
    "Effect": "Allow"
},
...
```

外部データソースへのコネクタのアクセス許可

外部データソースに接続する関数を使用するコンポーネントタイプを作成する場合、その関数を実装する Lambda 関数を使用するアクセス許可をサービスロールに付与する必要があります。コンポーネントタイプと関数の作成の詳細については、「???」を参照してください。

次の例では、サービスロールに Lambda 関数を使用するアクセス許可を付与します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucket*",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:PutObject"
      ],
      "Resource": [
```

```
        "arn:aws:s3:::bucket name",
        "arn:aws:s3:::bucket name/*"
      ]
    },
    {
        "Action": [
            "lambda:invokeFunction"
        ],
        "Resource": [
            "Lambda function ARN"
        ],
        "Effect": "Allow"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:DeleteObject"
      ],
      "Resource": [
        "arn:aws:s3:::*/DO_NOT_DELETE_WORKSPACE_*"
      ]
    }
  ]
}
```

IAM コンソール、、および IAM API を使用してロールを作成し、ポリシー AWS CLIと信頼関係を割り当てる方法の詳細については、「 にアクセス許可を委任するロールの作成 AWS のサービス」を参照してください。

Athena データコネクタを使用するようにワークスペース IAM ロールを変更する

AWS IoT TwinMaker Athena 表形式データコネクタ を使用するには、 AWS IoT TwinMaker ワークスペースの IAM ロールを更新する必要があります。ワークスペース IAM ロールに次のアクセス許可を追加する:

> **Note**
>
> この IAM 変更は、 AWS Glue および Amazon S3 に保存されている Athena 表形式データでのみ機能します。Athena を他のデータソースで使用するには、Athena の IAM ロールを設定する必要があります。「Athena の ID とアクセス管理」を参照してください。

```
{
    "Effect": "Allow",
    "Action": [
        "athena:GetQueryExecution",
        "athena:GetQueryResults",
        "athena:GetTableMetadata",
        "athena:GetWorkGroup",
        "athena:StartQueryExecution",
        "athena:StopQueryExecution"
    ],
    "Resource": [
        "athena resouces arn"
    ]
},// Athena permission
{
    "Effect": "Allow",
    "Action": [
        "glue:GetTable",
        "glue:GetTables",
        "glue:GetDatabase",
        "glue:GetDatabases"
    ],
    "Resource": [
        "glue resouces arn"
    ]
},// This is an example for accessing aws glue
{
    "Effect": "Allow",
    "Action": [
        "s3:ListBucket",
        "s3:GetObject"
    ],
    "Resource": [
        "Amazon S3 data source bucket resources arn"
    ]
```

```
}, // S3 bucket for storing the tabular data.
{
    "Effect": "Allow",
    "Action": [
        "s3:GetBucketLocation",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:ListBucketMultipartUploads",
        "s3:ListMultipartUploadParts",
        "s3:AbortMultipartUpload",
        "s3:CreateBucket",
        "s3:PutObject",
        "s3:PutBucketPublicAccessBlock"
    ],
    "Resource": [
        "S3 query result bucket resources arn"
    ]
} // Storing the query results
```

Athena IAM 設定の詳細については、「Athena の ID とアクセス管理」をご覧ください。

ワークスペースの作成

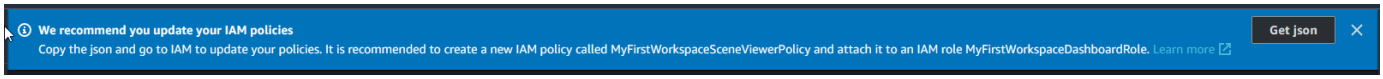
最初のワークスペースを作成して設定するには、次の手順に従います。

> **Note**
>
> このトピックでは、単一のリソースで簡単なワークスペースを作成する方法を説明します。複数のリソースを持つフル機能のワークスペースの場合は、サンプル Github リポジトリでAWS IoT TwinMaker サンプルセットアップを試してください。

1. 「AWS IoT TwinMaker 」 コンソールのホームページで、左側のナビゲーションペインで 「ワークスペース」 を選択します。
2. 「ワークスペース」 ページで、「ワークスペースを作成」 をクリックします。
3. 「ワークスペースを作成」 ページに、ワークスペース名を入力します。
4. (オプション) ワークスペースの説明を入力します。

5. 「S3 リソース」 で 「S3 バケットを作成」 を選択します。このオプションは、ワークスペースに関連する情報とリソースを が AWS IoT TwinMaker 保存する Amazon S3 バケットを作成します。各ワークスペースには独自のバケットが必要です。
6. 「実行ロール」 で、「新しいロールを自動生成」 またはこのワークスペース用に作成したカスタム IAM ロールを選択します。

   新しいロール の自動生成 を選択した場合、 は、前のステップで指定した Amazon Amazon S3バケットの読み取りと書き込みのアクセス許可を含め、他の AWS のサービスにアクセスするためのアクセス許可を新しいサービスロールに付与するポリシーをロールにア AWS IoT TwinMaker タッチします。このアクセス許可をロールに割り当てる方法の詳細については、「???」を参照してください。
7. 「ワークスペースを作成」 を選択します。次のバナーは、「ワークスペース」 ページの上部に表示されます。



8. 「JSON を取得」 を選択します。Grafana ダッシュボードを表示するユーザーとアカウント用に AWS IoT TwinMaker 作成した IAM ロールに、表示される IAM ポリシーを追加することをお勧めします。このロールの名前は次のパターンに従います: *workspace-name* DashboardRole。ポリシーを作成してロールにアタッチする方法については、「ロールのアクセス許可ポリシーの変更 (コンソール)」を参照してください。

   次の例には、ダッシュボードロールに追加するポリシーが含まれています。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::iottwinmaker-workspace-workspace-name-lower-case-account-id",
        "arn:aws:s3:::iottwinmaker-workspace-workspace-name-lower-case-account-id/
*"
      ]
    },
    {
```

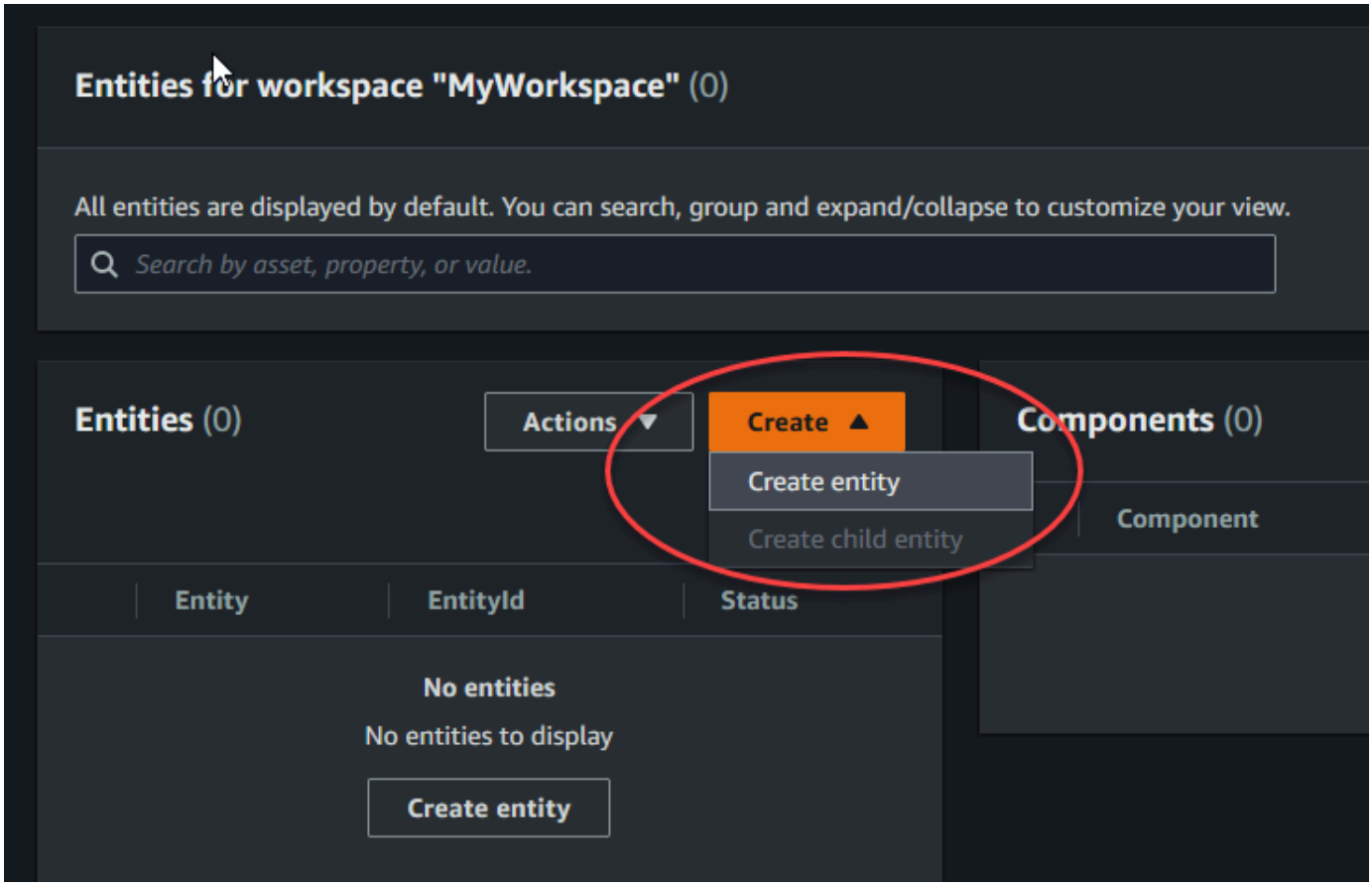
```
      "Effect": "Allow",
      "Action": [
        "iottwinmaker:Get*",
        "iottwinmaker:List*"
      ],
      "Resource": [
        "arn:aws:iottwinmaker:us-east-1:account-id:workspace/workspace-name",
        "arn:aws:iottwinmaker:us-east-1:account-id:workspace/workspace-name/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iottwinmaker:ListWorkspaces",
      "Resource": "*"
    }
  ]
}
```

これで、最初のエンティティを使用してワークスペースのデータモデルを作成する準備ができました。これを行う手順については、「最初のエンティティを作成する」を参照してください。

最初のエンティティを作成する

最初の エンティティ を作成するには、次の手順を実行します。

1. 「ワークスペース」 ページでワークスペースを選択し、左側のペインで 「エンティティ」 を選択します。
2. 「エンティティ」 ページで 「作成」 を選択し、「エンティティの作成」 を選択します。



3. 「エンティティの作成」 ウィンドウに、エンティティ名を入力します。この例では **`CookieMixer`** エンティティを使用します。
4. (オプション) エンティティの説明を入力します。
5. 「エンティティの作成」 を選択します。

エンティティには、ワークスペース内の各項目に関するデータが含まれます。components. AWS IoT TwinMaker provides を追加することで、データをエンティティに配置します。 には、次の組み込みコンポーネントタイプが用意されています。

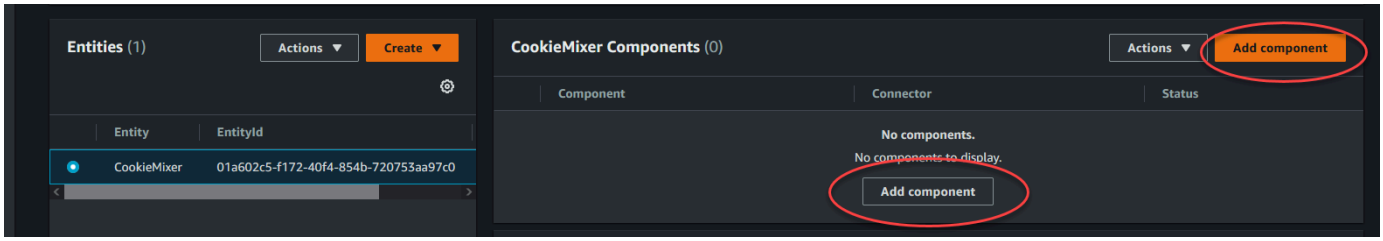
- パラメータ: キーと値のプロパティのセットを追加します。
- ドキュメント: エンティティに関する情報を含むドキュメント名と URL を追加します。
- アラーム: アラーム時系列データソースに接続します。
- SiteWise コネクタ : AWS IoT SiteWise アセットで定義されている時系列プロパティを取得します。

- Kinesis Video Streams 用エッジコネクタ： AWS IoT Greengrass KVS 用エッジコネクタからビデオデータを取得します AWS IoT Greengrass。詳細については、「AWS IoT TwinMakerビデオインテグレーション」を参照してください。

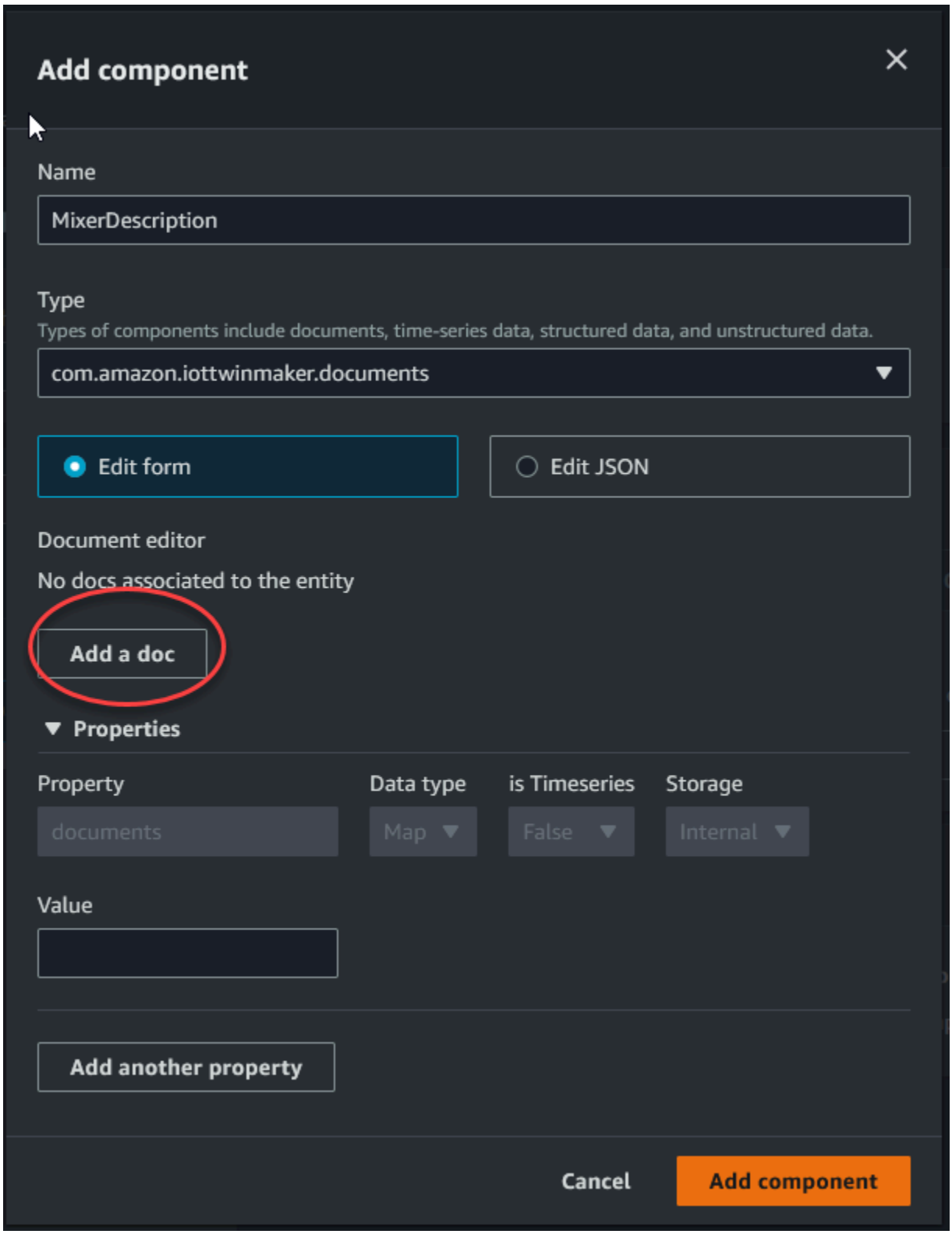
左側のペインで 「コンポーネントタイプ」 を選択すると、これらのコンポーネントタイプとその定義を確認できます。「コンポーネントタイプ」 ページでは、新しいコンポーネントタイプを作成することもできます。コンポーネントの作成の詳細については、「コンポーネントタイプの使用と作成」を参照してください。

この例では、エンティティに関する説明情報を追加する簡単なドキュメントコンポーネントを作成します。

1. エンティティページで、エンティティを選択し、コンポーネントの追加を選択します。



2. 「コンポーネントを追加」 ウィンドウに、コンポーネント名前を入力します。この例では クッキーミキサー エンティティを使用しているため、「名前」 フィールドに **`MixerDescription`** を入力します。



3. ドキュメントを追加 を選択し、ドキュメント名 と外部 URL の値を入力します。ドキュメントコンポーネントを使用すると、エンティティに関する重要な情報を含む外部 URLsのリストを保存できます。

4. 「コンポーネントを追加」 を選択します。

これで、最初のシーンを作成する準備ができました。これを行う手順については、「AWS IoT TwinMaker シーンの作成と編集」を参照してください。

AWS アカウントのセットアップ

がない場合は AWS アカウント、次の手順を実行して作成します。

にサインアップするには AWS アカウント

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、電話キーパッドで検証コードを入力するように求められます。

   にサインアップすると AWS アカウント、 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、ユーザーに管理アクセスを割り当て、ルートユーザーのみを使用してルートユーザーアクセスが必要なタスクを実行してください。

にサインアップする AWS アカウント

がない場合は AWS アカウント、次の手順を実行して作成します。

にサインアップするには AWS アカウント

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、電話キーパッドで検証コードを入力するように求められます。

   にサインアップすると AWS アカウント、 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、ユーザーに管理アクセスを割り当て、ルートユーザーのみを使用してルートユーザーアクセスが必要なタスクを実行してください。

AWS サインアッププロセスが完了すると、 から確認メールが送信されます。https://aws.amazon.com/ の [マイアカウント] を選んで、いつでもアカウントの現在のアクティビティを表示し、アカウントを管理できます。

管理アクセスを持つユーザーを作成する

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

のセキュリティ保護 AWS アカウントのルートユーザー

1. ルートユーザーを選択し、 AWS アカウント E メールアドレスを入力して、アカウント所有者AWS Management Consoleとして 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 ユーザーガイドの「ルートユーザーとしてサインインする」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 「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。

管理アクセスを持つ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、ユーザーに管理アクセスを付与します。

   を ID ソース IAM アイデンティティセンターディレクトリ として使用する方法のチュートリアルについては、「 ユーザーガイド」の「デフォルト でユーザーアクセス IAM アイデンティティセンターディレクトリを設定するAWS IAM Identity Center 」を参照してください。

管理アクセス権を持つ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 Identity Center 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 ユーザーガイド」の AWS 「 アクセスポータルにサインインする」を参照してください。

追加のユーザーにアクセス権を割り当てる

1. IAM アイデンティティセンターで、最小特権のアクセス許可を適用するというベストプラクティスに従ったアクセス許可セットを作成します。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「権限設定を作成する」を参照してください。
2. グループにユーザーを割り当て、そのグループにシングルサインオンアクセス権を割り当てます。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「グループの参加」を参照してください。

コンポーネントタイプの使用と作成

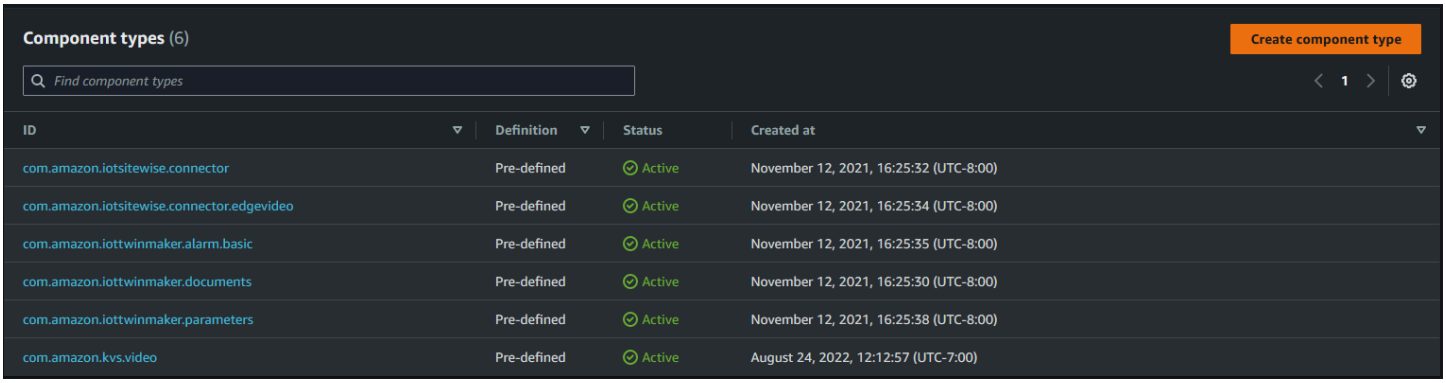
このトピックでは、 AWS IoT TwinMaker コンポーネントタイプの作成に使用する値と構造について説明します。CreateComponentTypeAPI に渡すか、 AWS IoT TwinMaker コンソールのコンポーネントタイプエディターを使用してリクエストオブジェクトを作成する方法を示します。

コンポーネントは、プロパティのコンテキストと、関連するエンティティのデータを提供します。

組み込みコンポーネントタイプ

AWS IoT TwinMaker コンソールでワークスペースを選択し、左側のペインで [Component types] を選択すると、以下のコンポーネントタイプが表示されます。

- com.amazon.iotsitewise.resourcesync: アセットとアセットモデルを自動的に同期し、エンティティ、コンポーネント、コンポーネントタイプに変換するコンポーネントタイプです。 AWS IoT SiteWise AWS IoT TwinMaker アセット同期の使用方法について詳しくは、「AWS IoT SiteWiseとのアセット同期」を参照してください。 AWS IoT SiteWise
- com.amazon.iottwinmaker.alarm.basic：外部ソースからエンティティにアラームデータを引き出す基本的なアラームコンポーネントです。このコンポーネントには、特定のデータソースに接続する関数は含まれていません。これは、アラームコンポーネントは抽象コンポーネントであり、データソースとそのソースから読み取る関数を指定する別のコンポーネントタイプに継承できることを意味します。
- com.amazon.iottwinmaker.documents：エンティティに関する情報を含むドキュメントのタイトルとURLを単純にマッピングしたものです。
- com.amazon.iotsitewise.connector.edgevideo: Kinesis ビデオストリーム用エッジコネクタコンポーネントを使用して IoT デバイスからビデオをエンティティに取り込むコンポーネント。 AWS IoT Greengrass Kinesis Video Streams AWS IoT GreengrassAWS IoT TwinMaker 用エッジコネクタコンポーネントはコンポーネントではなく、IoT AWS IoT Greengrass デバイスにローカルにデプロイされるビルド済みのコンポーネントです。
- com.amazon.iotsitewise.connector： AWS IoT SiteWise データをエンティティに取り込むコンポーネント。
- com.amazon.iottwinmaker.parameters：静的なキーと値のペアをエンティティに追加するコンポーネント。
- com.amazon.kvs.video: Kinesis ビデオストリームからエンティティに動画を取り込むコンポーネント。 AWS IoT TwinMaker



AWS IoT TwinMaker コンポーネントタイプのコア機能

以下のリストは、コンポーネントタイプのコア機能を示しています。

- プロパティ定義:PropertyDefinitionRequestこのオブジェクトは、シーンコンポーザーで設定できるプロパティを定義したり、外部データソースから取得したデータを設定したりできるプロパティを定義します。設定した静的プロパティはに保存されます。 AWS IoT TwinMakerデータソースから取得した時系列プロパティやその他のプロパティは外部に保存されます。

  プロパティ定義はPropertyDefinitionRequestマップの文字列内で指定します。各文字列はマップ内で一意でなければなりません。

- 関数:FunctionRequestこのオブジェクトは、外部データソースからの読み取りと場合によっては外部データソースへの書き込みを行う Lambda 関数を指定します。

  外部に保存されている値を持つプロパティを含むが、値を取得するための対応する関数がないコンポーネントタイプは、抽象コンポーネントタイプです。抽象コンポーネントタイプから具象コンポーネントタイプを拡張することができます。抽象コンポーネントタイプをエンティティに追加することはできません。シーンコンポーザーには表示されません。

  文字列の中にある関数をFunctionRequestマップに指定します。文字列には、以下の定義済み関数タイプのいずれかを指定する必要があります。

  - dataReader：外部ソースからデータを取得する関数。
  - dataReaderByEntity：外部ソースからデータを取得する関数。

    このタイプのデータリーダーを使用する場合、GetPropertyValueHistoryAPI オペレーションはこのコンポーネントタイプのプロパティに対するエンティティ固有のクエリのみをサポートします。（componentName+entityIdのプロパティ値履歴のみをリクエストできます。）
  - dataReaderByComponentType：外部ソースからデータを取得する関数。

このタイプのデータリーダーを使用する場合、GetPropertyValueHistoryAPI オペレーションはこのコンポーネントタイプのプロパティに対するエンティティ間クエリのみをサポートします。（プロパティ値の履歴を要求できるのは、componentTypeIdの場合のみです。）

- dataWriter：外部ソースにデータを書き込む関数。
- schemaInitializer：コンポーネントタイプを含むエンティティを作成するたびに、プロパティ値を自動的に初期化する関数。

非抽象コンポーネントタイプには、3種類のデータリーダー関数のうちの1つが必要です。

アラームを含むタイムストリームテレメトリコンポーネントを実装するLambda関数の例については、「AWS IoT TwinMaker サンプル」のデータリーダーを参照してください。

> **Note**
>
> アラームコネクタは抽象アラームコンポーネントタイプを継承するため、Lambda関数はalarm_key値を返す必要があります。この値を返さないと、Grafanaはそれをアラームとして認識しません。これはアラームを返すすべてのコンポーネントに必要です。

- 継承：コンポーネントタイプは継承によってコードの再利用性を促進します。コンポーネントタイプは最大10個の親コンポーネントタイプを継承できます。

  extendsFromパラメータを使用して、コンポーネントタイプがプロパティと機能を継承するコンポーネントタイプを指定します。
- IsSingleton：一部のコンポーネントには、位置座標など、1つのエンティティに複数回含めることができないプロパティが含まれています。isSingletonパラメータの値をtrueに設定すると、コンポーネントタイプをエンティティに1回だけ含めることができます。

プロパティ定義を作成

次の表は、PropertyDefinitionRequestのパラメータの説明です。

パラメータ	説明
isExternalId	プロパティが外部に保存されているプロパティ値の一意の識別子（AWS IoT SiteWise アセッ

パラメータ	説明
	ト ID など) であるかどうかを指定するブール値。 このプロパティのデフォルト値はfalseです。
isStoredExternally	プロパティ値を外部に保存するかどうかを指定するブール値。 このプロパティのデフォルト値はfalseです。
isTimeSeries	プロパティが時系列データで構成されるかどうかを指定するブール値。 このプロパティのデフォルト値はfalseです。
isRequiredInEntity	そのコンポーネントタイプを使用するエンティティ内のプロパティに値が必要かどうかを指定するブール値。
dataType	プロパティのデータ型 (文字列、マップ、リスト、測定単位など) DataTypeを指定するオブジェクト。
defaultValue	DataValueプロパティのデフォルト値を指定するオブジェクト。
configuration	string-to-string 外部データソースへの接続に必要な追加情報を指定するマップ。

関数の作成

次の表は、FunctionRequestのパラメータの説明です。

パラメータ	説明
implementedBy	外部データソースに接続する Lambda DataConnector関数を指定するオブジェクト。

パラメータ	説明
requiredProperties	関数が外部データソースから読み書きするために必要なプロパティのリスト。
scope	関数のスコープ。ワークスペース全体をスコープとする関数にはWorkspace を使用します。コンポーネントを含むエンティティに限定されたスコープを持つ関数にはEntityを使用します。

コンポーネントタイプの作成と拡張の方法を示す例については、???を参照してください。

コンポーネントタイプの例

このトピックでは、コンポーネントタイプの主要な概念を実装する方法を示す例を示します。

アラーム (要約)

次の例は、コンソールに表示される抽象アラームコンポーネントタイプです。 AWS IoT TwinMaker implementedBy値を持たないdataReaderからなるfunctionsリストが含まれています。

```
{
  "componentTypeId": "com.example.alarm.basic:1",
  "workspaceId": "MyWorkspace",
  "description": "Abstract alarm component type",
  "functions": {
    "dataReader": {
         "isInherited": false
    }
  },
  "isSingleton": false,
  "propertyDefinitions": {
    "alarm_key": {
      "dataType": { "type": "STRING" },
      "isExternalId": true,
      "isRequiredInEntity": true,
      "isStoredExternally": false,
      "isTimeSeries": false
```

```
    },
    "alarm_status": {
      "dataType": {
        "allowedValues": [
          {
            "stringValue": "ACTIVE"
          },
          {
            "stringValue": "SNOOZE_DISABLED"
          },
          {
            "stringValue": "ACKNOWLEDGED"
          },
          {
            "stringValue": "NORMAL"
          }
        ],
        "type": "STRING"
      },
      "isRequiredInEntity": false,
      "isStoredExternally": true,
      "isTimeSeries": true
    }
  }
}
```

注記：

componentTypeIdとworkspaceIDの値は必須です。componentTypeIdの値はワークスペースに固有である必要があります。alarm_keyの値は、関数が外部ソースからアラームデータを取得するために使用できる一意の識別子です。キーの値は必須で、 AWS IoT TwinMakerに格納されています。alarm_status時系列値は外部ソースに保存されます。

その他の例はAWS IoT TwinMaker サンプルにあります。

タイムストリームテレメトリ

次の例は、特定のタイプのコンポーネント (アラームや Cookie ミキサーなど) に関するテレメトリデータを外部ソースから取得する単純なコンポーネントタイプです。コンポーネントタイプが継承するLambda関数を指定します。

```
{
    "componentTypeId": "com.example.timestream-telemetry",
    "workspaceId": "MyWorkspace",
    "functions": {
        "dataReader": {
            "implementedBy": {
                "lambda": {
                    "arn": "lambdaArn"
                }
            }
        }
    },
    "propertyDefinitions": {
        "telemetryType": {
            "dataType": { "type": "STRING" },
            "isExternalId": false,
            "isStoredExternally": false,
            "isTimeSeries": false,
            "isRequiredInEntity": true
        },
        "telemetryId": {
            "dataType": { "type": "STRING" },
            "isExternalId": false,
            "isStoredExternally": false,
            "isTimeSeries": false,
            "isRequiredInEntity": true
        }
    }
}
```

アラーム（抽象アラームから継承）

次の例は、抽象アラームコンポーネントタイプとタイムストリームテレメトリコンポーネントタイプの両方を継承しています。アラームデータを取得する独自のLambda関数を指定します。

```
{
    "componentTypeId": "com.example.cookiefactory.alarm",
    "workspaceId": "MyWorkspace",
    "extendsFrom": [
        "com.example.timestream-telemetry",
        "com.amazon.iottwinmaker.alarm.basic"
```

```
    ],
    "propertyDefinitions": {
        "telemetryType": {
            "defaultValue": {
                "stringValue": "Alarm"
            }
        }
    },
    "functions": {
        "dataReader": {
            "implementedBy": {
                "lambda": {
                    "arn": "lambdaArn"
                }
            }
        }
    }
}
```

> **Note**
>
> アラームコネクタは抽象アラームコンポーネントタイプを継承するため、Lambda関数はalarm_key値を返す必要があります。この値を返さないと、Grafanaはそれをアラームとして認識しません。これはアラームを返すすべてのコンポーネントに必要です。

機器の例

このセクションの例では、潜在的な機器のモデリング方法を示します。これらの例を参考にして、独自のプロセスで機器をモデル化する方法についていくつかのアイデアを得ることができます。

クッキーミキサー

次の例は、タイムストリームテレメトリコンポーネントタイプを継承しています。クッキーミキサーの回転速度と温度に関する追加の時系列プロパティを指定します。

```
{
    "componentTypeId": "com.example.cookiefactory.mixer",
    "workspaceId": "MyWorkspace",
```

```
    "extendsFrom": [
        "com.example.timestream-telemetry"
    ],
    "propertyDefinitions": {
        "telemetryType": {
            "defaultValue" : { "stringValue": "Mixer" }
        },
        "RPM": {
            "dataType": { "type": "DOUBLE" },
            "isTimeSeries": true,
            "isStoredExternally": true
        },
        "Temperature": {
            "dataType": { "type": "DOUBLE" },
            "isTimeSeries": true,
            "isStoredExternally": true
        }
    }
}
```

水タンク

次の例は、タイムストリームテレメトリコンポーネントタイプを継承しています。水タンクの容積と流量に関する追加の時系列プロパティを指定します。

```
{
    "componentTypeId": "com.example.cookiefactory.watertank",
    "workspaceId": "MyWorkspace",
    "extendsFrom": [
        "com.example.timestream-telemetry"
    ],
    "propertyDefinitions": {
        "telemetryType": {
            "defaultValue" : { "stringValue": "WaterTank" }
        },
        "tankVolume1": {
            "dataType": { "type": "DOUBLE" },
            "isTimeSeries": true,
            "isStoredExternally": true
        },
        "tankVolume2": {
```

```
            "dataType": { "type": "DOUBLE" },
            "isTimeSeries": true,
            "isStoredExternally": true
        },
        "flowRate1": {
            "dataType": { "type": "DOUBLE" },
            "isTimeSeries": true,
            "isStoredExternally": true
        },
        "flowrate2": {
            "dataType": { "type": "DOUBLE" },
            "isTimeSeries": true,
            "isStoredExternally": true
        }
    }
}
```

スペースロケーション

次の例にはプロパティが含まれており、その値はに格納されています。 AWS IoT TwinMaker値はユーザーによって指定され、内部に保存されるため、値を取得するための関数は必要ありません。また、この例では、RELATIONSHIPデータタイプを使用して別のコンポーネントタイプとの関係を指定します。

このコンポーネントは、デジタルツインにコンテキストを追加するための軽量なメカニズムを提供します。これを使用して、どこにあるかを示すメタデータを追加できます。この情報は、どのカメラが機器や空間を認識できるかを判断したり、特定の場所に人を派遣する方法を知ったりするためのロジックにも使用できます。

```
{
    "componentTypeId": "com.example.cookiefactory.space",
    "workspaceId": "MyWorkspace",
    "propertyDefinitions": {
        "position":  {"dataType": {"nestedType": {"type": "DOUBLE"},"type": "LIST"}},
        "rotation":  {"dataType": {"nestedType": {"type": "DOUBLE"},"type": "LIST"}},
        "bounds":  {"dataType": {"nestedType": {"type": "DOUBLE"},"type": "LIST"}},
        "parent_space" : { "dataType": {"type": "RELATIONSHIP"}}
    }
}
```

AWS IoT TwinMaker 一括オペレーション

を使用して metadataTransferJob 、 AWS IoT TwinMaker リソースを大規模に転送および管理します。を使用すると、一括操作を実行し、 と Amazon metadataTransferJob S3 の間でリソース AWS IoT TwinMaker AWS IoT SiteWise を転送できます。 Amazon S3

一括オペレーションは、次のシナリオで使用できます。

- 開発用アカウントから本番稼働用アカウントへの移行など、アカウント間のアセットとデータの一括移行。
- 大規模なアセットのアップロードや編集などの大規模な AWS IoT アセット管理。
- AWS IoT TwinMaker および へのアセットの一括インポート AWS IoT SiteWise。
- `revit` や ファイルなどの既存のオントロジー`BIM`ファイルからの AWS IoT TwinMaker エンティティの一括インポート。

トピック

主要な概念と用語

AWS IoT TwinMaker 一括オペレーションでは、次の概念と用語を使用します。

- インポート: リソースを AWS IoT TwinMaker ワークスペースに移動するアクション。例えば、ローカルファイル、Amazon S3 バケット内のファイル、または から AWS IoT TwinMaker ワークスペース AWS IoT SiteWise などです。
- エクスポート: ワークスペースから AWS IoT TwinMaker ローカルマシンまたは Amazon S3 バケットにリソースを移動するアクション。
- ソース : リソースを移動する開始場所。

  例えば、Amazon S3 バケットはインポートソースであり、 AWS IoT TwinMaker ワークスペースはエクスポートソースです。
- 送信先 : リソースの移動先となる場所。

例えば、Amazon S3 バケットはエクスポート先、 AWS IoT TwinMaker ワークスペースはインポート先です。

- AWS IoT SiteWise スキーマ： との間でリソースをインポートおよびエクスポートするために使用されるスキーマ AWS IoT SiteWise。
- AWS IoT TwinMaker スキーマ： との間でリソースをインポートおよびエクスポートするために使用されるスキーマ AWS IoT TwinMaker。
- AWS IoT TwinMaker 最上位リソース: 既存の APIsで使用されるリソース。具体的には、エンティティまたは ですComponentType。
- AWS IoT TwinMaker サブレベルリソース: メタデータ定義で使用されるネストされたリソースタイプ。具体的には、コンポーネント です。
- メタデータ: AWS IoT SiteWise および AWS IoT TwinMaker リソースを正常にインポートまたはエクスポートするために必要なキー情報。
- metadataTransferJob: の実行時に作成されたオブジェクト`CreateMetadataTransferJob`。

AWS IoT TwinMaker metadataTransferJob 機能

このトピックでは、一括操作を実行する AWS IoT TwinMaker 場合の動作、つまり metadataTransferJob の処理方法について説明します。また、リソースの転送に必要なメタデータを使用してスキーマを定義する方法についても説明します。 AWS IoT TwinMaker バルクオペレーションは次の機能をサポートしています。

- 最上位レベルのリソースの作成または置換： AWS IoT TwinMaker は、新しいリソースを作成するか、リソース ID によって一意に識別されるすべての既存のリソースを置き換えます。

  例えば、システム内にエンティティが存在する場合、エンティティ定義は `Entity`キーの下のテンプレートで定義されている新しいものに置き換えられます。
- サブリソースの作成または置き換え：

  EntityComponent レベルから作成または置換できるのは、コンポーネントのみです。エンティティは既に存在している必要があります。存在しない場合、アクションは を生成します ValidationException。

  プロパティレベルまたはリレーションシップレベルから、プロパティまたはリレーションシップを作成または置き換えることができ、それを含む が既に存在している EntityComponent 必要があります。

- サブリソースの削除：

  AWS IoT TwinMaker は、サブリソースの削除もサポートしています。サブリソースは、コンポーネント、プロパティ、または関係にすることができます。

  コンポーネントを削除する場合は、エンティティレベルから削除する必要があります。

  プロパティまたは関係を削除する場合は、エンティティまたは EntityComponent レベルから削除する必要があります。

  サブリソースを削除するには、上位レベルのリソースを更新し、サブリソースの定義を省略します。

- 最上位レベルのリソースを削除しない： AWS IoT TwinMaker 最上位レベルのリソースは削除されません。最上位リソースとは、エンティティまたは を指します ComponentType。
- 1 つのテンプレートで同じ最上位リソースのサブリソース定義はありません。

  同じテンプレート内の同じエンティティの完全なエンティティ定義とサブリソース (プロパティなど) 定義を指定することはできません。

  entityId が Entity で使用されている場合、エンティティ、プロパティ、または関係で同じ ID EntityComponentを使用することはできません。

  で entityId または componentName の組み合わせが使用されている場合 EntityComponent、、プロパティ EntityComponent、またはリレーションシップで同じ組み合わせを使用することはできません。

  entityId、componentName、propertyName の組み合わせがプロパティまたはリレーションシップで使用されている場合、プロパティまたはリレーションシップで同じ組み合わせを使用することはできません。

- ExternalId は のオプションです AWS IoT TwinMaker。 ExternalId を使用して、 リソースを識別できます。

一括インポートおよびエクスポートオペレーションの実行

このトピックでは、一括インポートおよびエクスポートオペレーションを実行する方法と、転送ジョブのエラーを処理する方法について説明します。CLI コマンドを使用した転送ジョブの例を示します。

AWS IoT TwinMaker API リファレンスには、 CreateMetadataTransferJobおよびその他の API アクションに関する情報が含まれています。

トピック

metadataTransferJob 前提条件

を実行する前に、次の前提条件を完了してください metadataTransferJob。

- AWS IoT TwinMaker ワークスペースを作成します。ワークスペースは、 のインポート先またはエクスポート元にすることができます metadataTransferJob。ワークスペースの作成については、「」を参照してくださいワークスペースの作成。
- リソースを保存する Amazon S3 バケットを作成します。Amazon S3 の使用の詳細については、「Amazon S3 とは」を参照してください。

IAM アクセス許可

一括操作を実行する場合は、Amazon S3 AWS IoT TwinMaker AWS IoT SiteWise、、およびローカルマシン間の AWS リソースの交換を許可するアクセス許可を持つ IAM ポリシーを作成する必要があります。IAM ポリシーの作成の詳細については、「IAM ポリシーの作成」を参照してください。

AWS IoT TwinMaker AWS IoT SiteWise および Amazon S3 のポリシーステートメントを以下に示します。

- AWS IoT TwinMaker ポリシー：

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

```
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetBucketLocation",
            "s3:ListBucket",
            "s3:AbortMultipartUpload",
            "s3:ListBucketMultipartUploads",
            "s3:ListMultipartUploadParts"
        ],
        "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iottwinmaker:GetWorkspace",
                "iottwinmaker:CreateEntity",
                "iottwinmaker:GetEntity",
                "iottwinmaker:UpdateEntity",
                "iottwinmaker:GetComponentType",
                "iottwinmaker:CreateComponentType",
                "iottwinmaker:UpdateComponentType",
                "iottwinmaker:ListEntities",
                "iottwinmaker:ListComponentTypes",
                "iottwinmaker:ListTagsForResource",
                "iottwinmaker:TagResource",
                "iottwinmaker:UntagResource"
            ],
            "Resource": "*"
        }
    ]
}
```

- AWS IoT SiteWise ポリシー：

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetBucketLocation",
            "s3:ListBucket",
            "s3:AbortMultipartUpload",
            "s3:ListBucketMultipartUploads",
```

```
            "s3:ListMultipartUploadParts"
        ],
        "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iotsitewise:CreateAsset",
                "iotsitewise:CreateAssetModel",
                "iotsitewise:UpdateAsset",
                "iotsitewise:UpdateAssetModel",
                "iotsitewise:UpdateAssetProperty",
                "iotsitewise:ListAssets",
                "iotsitewise:ListAssetModels",
                "iotsitewise:ListAssetProperties",
                "iotsitewise:ListAssetModelProperties",
                "iotsitewise:ListAssociatedAssets",
                "iotsitewise:DescribeAsset",
                "iotsitewise:DescribeAssetModel",
                "iotsitewise:DescribeAssetProperty",
                "iotsitewise:AssociateAssets",
                "iotsitewise:DisassociateAssets",
                "iotsitewise:AssociateTimeSeriesToAssetProperty",
                "iotsitewise:DisassociateTimeSeriesFromAssetProperty",
                "iotsitewise:BatchPutAssetPropertyValue",
                "iotsitewise:BatchGetAssetPropertyValue",
                "iotsitewise:TagResource",
                "iotsitewise:UntagResource",
                "iotsitewise:ListTagsForResource"
            ],
            "Resource": "*"
        }
    ]
}
```

- Amazon S3 ポリシー :

```
{
    "Effect": "Allow",
    "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:GetBucketLocation",
        "s3:ListBucket",
```

```
            "s3:AbortMultipartUpload",
            "s3:ListBucketMultipartUploads",
            "s3:ListMultipartUploadParts"
        ],
        "Resource": "*"
}
```

または、単一の Amazon S3 バケットにのみアクセスするように Amazon S3 ポリシーの範囲を設定することもできます。次のポリシーを参照してください。

Amazon S3 シングルバケットスコープポリシー

```
{
    "Effect": "Allow",
    "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:GetBucketLocation",
        "s3:ListBucket",
        "s3:AbortMultipartUpload",
        "s3:ListBucketMultipartUploads",
        "s3:ListMultipartUploadParts"
    ],
    "Resource": [
        "arn:aws:s3:::bucket name",
        "arn:aws:s3:::bucket name/*"
    ]
}
```

のアクセスコントロールを設定する metadataTransferJob

ユーザーがアクセスできるジョブの種類を制御するには、 の呼び出しに使用されるロールに次の IAM ポリシーを追加します AWS IoT TwinMaker。

> **Note**
>
> このポリシーでは、Amazon S3 との間でリソースを転送するジョブの AWS IoT TwinMaker インポートおよびエクスポートへのアクセスのみが許可されます。 Amazon S3

```
{
    "Effect": "Allow",
    "Action": [
        "iottwinmaker:*DataTransferJob*"
    ],
    "Resource": "*",
    "Condition": {
        "StringLikeIfExists": {
            "iottwinmaker:sourceType": [
                "s3",
                "iottwinmaker"
            ],
            "iottwinmaker:destinationType": [
                "iottwinmaker",
                "s3"
            ]
        }
    }
}
```

一括オペレーションを実行する

このセクションでは、一括インポートおよびエクスポートオペレーションを実行する方法について説明します。

Amazon S3 から にデータをインポートする AWS IoT TwinMaker

1. スキーマを使用して、転送するリソースを指定します AWS IoT TwinMaker metadataTransferJob 。スキーマファイルを作成して Amazon S3 バケットに保存します。

   スキーマの例については、「」を参照してください[メタデータテンプレートをインポートする](#)。
2. リクエストボディを作成し、JSON ファイルとして保存します。リクエストボディは、転送ジョブの送信元と送信先を指定します。Amazon S3 バケットをソースとして指定し、 AWS IoT TwinMaker ワークスペースを送信先として指定してください。

   リクエストボディの例を次に示します。

   ```
   {
       "metadataTransferJobId": "your-transfer-job-Id",
       "sources": [{
           "type": "s3",
   ```

```
        "s3Configuration": {
            "location": "arn:aws:s3:::your-S3-bucket-name/your_import_data.json"
        }
    }],
    "destination": {
        "type": "iottwinmaker",
        "iotTwinMakerConfiguration": {
            "workspace": "arn:aws:iottwinmaker:us-
east-1:111122223333:workspace/your-worksapce-name"
        }
    }
}
```

リクエストボディに付けたファイル名を記録します。次のステップで必要になります。この例では、リクエストボディの名前は ですcreateMetadataTransferJobImport.json。

3. 次の CLI コマンドを実行して を呼び出します CreateMetadataTransferJob (input-json ファイル名を、リクエストボディに付けた名前に置き換えます) 。

```
aws iottwinmaker create-metadata-transfer-job --region us-east-1 \
--cli-input-json file://createMetadataTransferJobImport.json
```

これにより、 が作成され metadataTransferJob 、選択したリソースの転送プロセスが開始されます。

から Amazon S3 AWS IoT TwinMaker にデータをエクスポートする

1. 適切なフィルターを使用して JSON リクエストボディを作成し、エクスポートするリソースを選択します。この例では、以下を使用します。

```
{
    "metadataTransferJobId": "your-transfer-job-Id",
    "sources": [{
        "type": "iottwinmaker",
        "iotTwinMakerConfiguration": {
            "workspace": "arn:aws:iottwinmaker:us-
east-1:111122223333:workspace/your-workspace-name",
            "filters": [{
                "filterByEntity": {
                    "entityId": "parent"
                }},
```

```
                {
                "filterByEntity": {
                    "entityId": "child"
                }},
                {
                "filterByComponentType": {
                    "componentTypeId": "component.type.minimal"
                }}
            ]
        }
    }],
    "destination": {
        "type": "s3",
        "s3Configuration": {
            "location": "arn:aws:s3:::your-S3-bucket-location"
        }
    }
}
```

filters 配列では、エクスポートするリソースを指定できます。この例では、、、entity、および でフィルタリングしますcomponentType。

AWS IoT TwinMaker ワークスペースをソースとして指定し、Amazon S3 バケットをメタデータ転送ジョブの送信先として指定してください。

リクエストボディを保存してファイル名を記録します。次のステップで必要になります。この例では、リクエストボディに という名前を付けましたcreateMetadataTransferJobExport.json。

2. 次の CLI コマンドを実行して を呼び出します CreateMetadataTransferJob (input-json ファイル名を、リクエストボディに付けた名前に置き換えます) 。

```
aws iottwinmaker create-metadata-transfer-job --region us-east-1 \
--cli-input-json file://createMetadataTransferJobExport.json
```

これにより、 が作成され metadataTransferJob 、選択したリソースの転送プロセスが開始されます。

転送ジョブのステータスを確認または更新するには、次のコマンドを使用します。

- ジョブをキャンセルするには、 CancelMetadataTransferJob API アクションを使用します。を呼び出すと CancelMetadataTransferJob、API は実行中の のみをキャンセルし metadataTransferJob、すでにエクスポートまたはインポートされたリソースはこの API コールの影響を受けません。
- 特定のジョブに関する情報を取得するには、 GetMetadataTransferJob API アクションを使用します。

  または、次の CLI コマンドを使用して、既存の転送ジョブ GetMetadataTransferJob で を呼び出すこともできます。

```
aws iottwinmaker get-metadata-transfer-job --job-id ExistingJobId
```

  存在しない AWS IoT TwinMaker インポートジョブまたはエクスポートジョブ GetMetadataTransferJob で を呼び出すと、レスポンスで `ResourceNotFoundException` エラーが発生します。
- 現在のジョブを一覧表示するには、 ListMetadataTransferJobs API アクションを使用します。

  を destinationType ListMetadataTransferJobs AWS IoT TwinMaker として、 を sourceType `s3`として呼び出す CLI の例を次に示します。 sourceType

```
aws iottwinmaker list-metadata-transfer-jobs --destination-type iottwinmaker --
source-type s3
```

> **Note**
>
> sourceType パラメータと destinationType パラメータの値は、インポートまたはエクスポートジョブの送信元と送信先に合わせて変更できます。

これらの API アクションを呼び出す CLI コマンドのその他の例については、「」を参照してくださいAWS IoT TwinMaker metadataTransferJob 例。

転送ジョブ中にエラーが発生した場合は、「」を参照してくださいエラー処理。

エラー処理

転送ジョブを作成して実行したら、 を呼び出し GetMetadataTransferJobて発生したエラーを診断できます。

```
aws iottwinmaker get-metadata-transfer-job \
--metadata-transfer-job-id your_metadata_transfer_job_id \
--region us-east-1
```

ジョブの状態が に変わったらCOMPLETED、ジョブの結果を確認できます。GetMetadataTransferJob は、以下のフィールドを含む MetadataTransferJobProgress というオブジェクトを返します。

- failedCount : 転送プロセス中に失敗したリソースの数を示します。
- skippedCount :転送プロセス中にスキップされたリソースの数を示します。
- succeededCount : 転送プロセス中に成功したリソースの数を示します。
- totalCount : 転送プロセスに関係するリソースの総数を示します。

さらに、署名付き URL を含む reportUrl 要素が返されます。転送ジョブにさらに調査したいエラーがある場合は、この URL を使用して完全なエラーレポートをダウンロードできます。

メタデータテンプレートをインポートする

1 回の一括インポートオペレーションで、多数のコンポーネント、componentTypes、またはエンティティをインポートできます。このセクションの例では、これを行う方法を示します。

template: Importing entities

エンティティをインポートするジョブには、次のテンプレート形式を使用します。

```
{
  "entities": [
    {
      "description": "string",
      "entityId": "string",
      "entityName": "string",
      "parentEntityId": "string",
      "tags": {
        "string": "string"
      },
      "components": {
        "string": {
          "componentTypeId": "string",
          "description": "string",
          "properties": {
```

```
            "string": {
              "definition": {
                "configuration": {
                  "string": "string"
                },
                "dataType": "DataType",
                "defaultValue": "DataValue",
                "displayName": "string",
                "isExternalId": "boolean",
                "isRequiredInEntity": "boolean",
                "isStoredExternally": "boolean",
                "isTimeSeries": "boolean"
              },
              "value": "DataValue"
            }
          },
          "propertyGroups": {
            "string": {
              "groupType": "string",
              "propertyNames": [
                "string"
              ]
            }
          }
        }
      }
    }
  ]
}
```

template: Importing componentTypes

componentTypes をインポートするジョブには、次のテンプレート形式を使用します。

```
{
  "componentTypes": [
    {
      "componentTypeId": "string",
      "componentTypeName": "string",
      "description": "string",
      "extendsFrom": [
        "string"
      ],
      "functions": {
```

```
        "string": {
          "implementedBy": {
            "isNative": "boolean",
            "lambda": {
              "functionName": "Telemetry-tsDataReader",
              "arn": "Telemetry-tsDataReaderARN"
            }
          },
          "requiredProperties": [
            "string"
          ],
          "scope": "string"
        }
      },
      "isSingleton": "boolean",
      "propertyDefinitions": {
        "string": {
          "configuration": {
            "string": "string"
          },
          "dataType": "DataType",
          "defaultValue": "DataValue",
          "displayName": "string",
          "isExternalId": "boolean",
          "isRequiredInEntity": "boolean",
          "isStoredExternally": "boolean",
          "isTimeSeries": "boolean"
        }
      },
      "propertyGroups": {
        "string": {
          "groupType": "string",
          "propertyNames": [
            "string"
          ]
        }
      },
      "tags": {
        "string": "string"
      }
    }
  ]
}
```

template: Importing components

コンポーネントをインポートするジョブには、次のテンプレート形式を使用します。

```
{
  "entityComponents": [
    {
      "entityId": "string",
      "componentName": "string",
      "componentTypeId": "string",
      "description": "string",
      "properties": {
        "string": {
          "definition": {
            "configuration": {
              "string": "string"
            },
            "dataType": "DataType",
            "defaultValue": "DataValue",
            "displayName": "string",
            "isExternalId": "boolean",
            "isRequiredInEntity": "boolean",
            "isStoredExternally": "boolean",
            "isTimeSeries": "boolean"
          },
          "value": "DataValue"
        }
      },
      "propertyGroups": {
        "string": {
          "groupType": "string",
          "propertyNames": [
            "string"
          ]
        }
      }
    }
  ]
}
```

AWS IoT TwinMaker metadataTransferJob 例

メタデータ転送を管理するには、次のコマンドを使用します。

- CreateMetadataTransferJob API アクション。

  CLI コマンドの例：

  ```
  aws iottwinmaker create-metadata-transfer-job --region us-east-1 \
  --cli-input-json file://yourTransferFileName.json
  ```

- ジョブをキャンセルするには、 CancelMetadataTransferJob API アクションを使用します。

  CLI コマンドの例：

  ```
  aws iottwinmaker cancel-metadata-transfer-job
  --region us-east-1 \
  --metadata-transfer-job-id job-to-cancel-id
  ```

  を呼び出すと CancelMetadataTransferJob、特定のメタデータ転送ジョブのみがキャンセルされ、エクスポートまたはインポート済みのリソースは影響を受けません。

- 特定のジョブに関する情報を取得するには、 GetMetadataTransferJob API アクションを使用します。

  CLI コマンドの例：

  ```
  aws iottwinmaker get-metadata-transfer-job \
  --metadata-transfer-job-id your_metadata_transfer_job_id \
  --region us-east-1 \
  ```

- 現在のジョブを一覧表示するには、 ListMetadataTransferJobs API アクションを使用します。

  JSON ファイル ListMetadataTransferJobs を使用して、返された結果をフィルタリングできます。CLI を使用して次の手順を参照してください。

  1. CLI 入力 JSON ファイルを作成して、使用するフィルターを指定します。

     ```
     {
         "sourceType": "s3",
         "destinationType": "iottwinmaker",
         "filters": [{
     ```

```
        "workspaceId": "workspaceforbulkimport"
    },
    {
        "state": "COMPLETED"
    }]
}
```

これを保存してファイル名を記録します。CLI コマンドを入力するときに必要になります。

2. 次の CLI コマンドの引数として JSON ファイルを使用します。

```
aws iottwinmaker list-metadata-transfer-job --region us-east-1 \
--cli-input-json file://ListMetadataTransferJobsExample.json
```

AWS IoT TwinMaker メタデータ転送ジョブスキーマ

metadataTransferJob スキーマのインポート： Amazon S3 バケットにアップロードするときに、この AWS IoT TwinMaker メタデータスキーマを使用してデータを検証します。

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "title": "IoTTwinMaker",
  "description": "Metadata transfer job resource schema for IoTTwinMaker",
  "definitions": {
    "ExternalId": {
      "type": "string",
      "minLength": 1,
      "maxLength": 128,
      "pattern": "[a-zA-Z0-9][a-zA-Z_\\-0-9.:]*[a-zA-Z0-9]+"
    },
    "Description": {
      "type": "string",
      "minLength": 0,
      "maxLength": 512
    },
    "DescriptionWithDefault": {
      "type": "string",
      "minLength": 0,
      "maxLength": 512,
      "default": ""
    },
    "ComponentTypeName": {
```

```
      "description": "A friendly name for the component type.",
      "type": "string",
      "pattern": ".*[^\\u0000-\\u001F\\u007F]*.*",
      "minLength": 1,
      "maxLength": 256
    },
    "ComponentTypeId": {
      "description": "The ID of the component type.",
      "type": "string",
      "pattern": "[a-zA-Z_.\\-0-9:]+",
      "minLength": 1,
      "maxLength": 256
    },
    "ComponentName": {
      "description": "The name of the component.",
      "type": "string",
      "pattern": "[a-zA-Z_\\-0-9]+",
      "minLength": 1,
      "maxLength": 256
    },
    "EntityId": {
      "description": "The ID of the entity.",
      "type": "string",
      "minLength": 1,
      "maxLength": 128,
      "pattern": "[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}|^[a-zA-
Z0-9][a-zA-Z_\\-0-9.:]*[a-zA-Z0-9]+"
    },
    "EntityName": {
      "description": "The name of the entity.",
      "type": "string",
      "minLength": 1,
      "maxLength": 256,
      "pattern": "[a-zA-Z_0-9-.][a-zA-Z_0-9-. ]*[a-zA-Z0-9]+"
    },
    "ParentEntityId": {
      "description": "The ID of the parent entity.",
      "type": "string",
      "minLength": 1,
      "maxLength": 128,
      "pattern": "\\$ROOT|^[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]
{12}|^[a-zA-Z0-9][a-zA-Z_\\-0-9.:]*[a-zA-Z0-9]+",
      "default": "$ROOT"
    },
```

```
    "DisplayName": {
      "description": "A friendly name for the property.",
      "type": "string",
      "pattern": ".*[^\\u0000-\\u001F\\u007F]*.*",
      "minLength": 0,
      "maxLength": 256
    },
    "Tags": {
      "description": "Metadata that you can use to manage the entity / componentType",
      "patternProperties": {
        "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-@]*)$": {
          "type": "string",
          "minLength": 1,
          "maxLength": 256
        }
      },
      "existingJavaType": "java.util.Map<String,String>",
      "minProperties": 0,
      "maxProperties": 50
    },
    "Relationship": {
      "description": "The type of the relationship.",
      "type": "object",
      "properties": {
        "relationshipType": {
          "description": "The type of the relationship.",
          "type": "string",
          "pattern": ".*",
          "minLength": 1,
          "maxLength": 256
        },
        "targetComponentTypeId": {
          "description": "The ID of the target component type associated with this
relationship.",
          "$ref": "#/definitions/ComponentTypeId"
        }
      },
      "additionalProperties": false
    },
    "DataValue": {
      "description": "An object that specifies a value for a property.",
      "type": "object",
      "properties": {
        "booleanValue": {
```

```
          "description": "A Boolean value.",
          "type": "boolean"
        },
        "doubleValue": {
          "description": "A double value.",
          "type": "number"
        },
        "expression": {
          "description": "An expression that produces the value.",
          "type": "string",
          "pattern": "(^\\$\\{Parameters\\.[a-zA-z]+([a-zA-z_0-9]*)}$)",
          "minLength": 1,
          "maxLength": 316
        },
        "integerValue": {
          "description": "An integer value.",
          "type": "integer"
        },
        "listValue": {
          "description": "A list of multiple values.",
          "type": "array",
          "minItems": 0,
          "maxItems": 50,
          "uniqueItems": false,
          "insertionOrder": false,
          "items": {
            "$ref": "#/definitions/DataValue"
          },
          "default": null
        },
        "longValue": {
          "description": "A long value.",
          "type": "integer",
          "existingJavaType": "java.lang.Long"
        },
        "stringValue": {
          "description": "A string value.",
          "type": "string",
          "pattern": ".*",
          "minLength": 1,
          "maxLength": 256
        },
        "mapValue": {
          "description": "An object that maps strings to multiple DataValue objects.",
```

```
          "type": "object",
          "patternProperties": {
            "[a-zA-Z_\\-0-9]+": {
              "$ref": "#/definitions/DataValue"
            }
          },
          "additionalProperties": {
            "$ref": "#/definitions/DataValue"
          }
        },
        "relationshipValue": {
          "description": "A value that relates a component to another component.",
          "type": "object",
          "properties": {
            "TargetComponentName": {
              "type": "string",
              "pattern": "[a-zA-Z_\\-0-9]+",
              "minLength": 1,
              "maxLength": 256
            },
            "TargetEntityId": {
              "type": "string",
              "pattern": "[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}|
^[a-zA-Z0-9][a-zA-Z_\\-0-9.:]*[a-zA-Z0-9]+",
              "minLength": 1,
              "maxLength": 128
            }
          },
          "additionalProperties": false
        }
      },
      "additionalProperties": false
    },
    "DataType": {
      "description": "An object that specifies the data type of a property.",
      "type": "object",
      "properties": {
        "allowedValues": {
          "description": "The allowed values for this data type.",
          "type": "array",
          "minItems": 0,
          "maxItems": 50,
          "uniqueItems": false,
          "insertionOrder": false,
```

```
          "items": {
            "$ref": "#/definitions/DataValue"
          },
          "default": null
        },
        "nestedType": {
          "description": "The nested type in the data type.",
          "$ref": "#/definitions/DataType"
        },
        "relationship": {
          "description": "A relationship that associates a component with another
 component.",
          "$ref": "#/definitions/Relationship"
        },
        "type": {
          "description": "The underlying type of the data type.",
          "type": "string",
          "enum": [
            "RELATIONSHIP",
            "STRING",
            "LONG",
            "BOOLEAN",
            "INTEGER",
            "DOUBLE",
            "LIST",
            "MAP"
          ]
        },
        "unitOfMeasure": {
          "description": "The unit of measure used in this data type.",
          "type": "string",
          "pattern": ".*",
          "minLength": 1,
          "maxLength": 256
        }
      },
      "required": [
        "type"
      ],
      "additionalProperties": false
    },
    "PropertyDefinition": {
      "description": "An object that specifies information about a property.",
      "type": "object",
```

```
      "properties": {
        "configuration": {
          "description": "An object that specifies information about a property.",
          "patternProperties": {
            "[a-zA-Z_\\-0-9]+": {
              "type": "string",
              "pattern": "[a-zA-Z_\\-0-9]+",
              "minLength": 1,
              "maxLength": 256
            }
          },
          "existingJavaType": "java.util.Map<String,String>"
        },
        "dataType": {
          "description": "An object that contains information about the data type.",
          "$ref": "#/definitions/DataType"
        },
        "defaultValue": {
          "description": "An object that contains the default value.",
          "$ref": "#/definitions/DataValue"
        },
        "displayName": {
          "description": "An object that contains the default value.",
          "$ref": "#/definitions/DisplayName"
        },
        "isExternalId": {
          "description": "A Boolean value that specifies whether the property ID comes
from an external data store.",
          "type": "boolean",
          "default": null
        },
        "isRequiredInEntity": {
          "description": "A Boolean value that specifies whether the property is
required.",
          "type": "boolean",
          "default": null
        },
        "isStoredExternally": {
          "description": "A Boolean value that specifies whether the property is stored
externally.",
          "type": "boolean",
          "default": null
        },
        "isTimeSeries": {
```

```
          "description": "A Boolean value that specifies whether the property consists
 of time series data.",
          "type": "boolean",
          "default": null
        }
      },
      "additionalProperties": false
    },
    "PropertyDefinitions": {
      "type": "object",
      "patternProperties": {
        "[a-zA-Z_\\-0-9]+": {
          "$ref": "#/definitions/PropertyDefinition"
        }
      },
      "additionalProperties": {
        "$ref": "#/definitions/PropertyDefinition"
      }
    },
    "Property": {
      "type": "object",
      "properties": {
        "definition": {
          "description": "The definition of the property",
          "$ref": "#/definitions/PropertyDefinition"
        },
        "value": {
          "description": "The value of the property.",
          "$ref": "#/definitions/DataValue"
        }
      },
      "additionalProperties": false
    },
    "Properties": {
      "type": "object",
      "patternProperties": {
        "[a-zA-Z_\\-0-9]+": {
          "$ref": "#/definitions/Property"
        }
      },
      "additionalProperties": {
        "$ref": "#/definitions/Property"
      }
    },
```

```
    "PropertyName": {
      "type": "string",
      "pattern": "[a-zA-Z_\\-0-9]+"
    },
    "PropertyGroup": {
      "description": "An object that specifies information about a property group.",
      "type": "object",
      "properties": {
        "groupType": {
          "description": "The type of property group.",
          "type": "string",
          "enum": [
            "TABULAR"
          ]
        },
        "propertyNames": {
          "description": "The list of property names in the property group.",
          "type": "array",
          "minItems": 1,
          "maxItems": 256,
          "uniqueItems": true,
          "insertionOrder": false,
          "items": {
            "$ref": "#/definitions/PropertyName"
          },
          "default": null
        }
      },
      "additionalProperties": false
    },
    "PropertyGroups": {
      "type": "object",
      "patternProperties": {
        "[a-zA-Z_\\-0-9]+": {
          "$ref": "#/definitions/PropertyGroup"
        }
      },
      "additionalProperties": {
        "$ref": "#/definitions/PropertyGroup"
      }
    },
    "Component": {
      "type": "object",
      "properties": {
```

```
        "componentTypeId": {
          "$ref": "#/definitions/ComponentTypeId"
        },
        "description": {
          "$ref": "#/definitions/Description"
        },
        "properties": {
          "description": "An object that maps strings to the properties to set in the
 component type. Each string in the mapping must be unique to this object.",
          "$ref": "#/definitions/Properties"
        },
        "propertyGroups": {
          "description": "An object that maps strings to the property groups to set in
 the entity component. Each string in the mapping must be unique to this object.",
          "$ref": "#/definitions/PropertyGroups"
        }
      },
      "required": [
        "componentTypeId"
      ],
      "additionalProperties": false
    },
    "RequiredProperty": {
      "type": "string",
      "pattern": "[a-zA-Z_\\-0-9]+"
    },
    "LambdaFunction": {
      "type": "object",
      "properties": {
        "arn": {
          "type": "string",
          "pattern": "arn:((aws)|(aws-cn)|(aws-us-gov)|(\\${partition})):lambda:(([a-
z0-9-]+)|(\\${region})):([0-9]{12}|(\\${accountId})):function:[/a-zA-Z0-9_-]+",
          "minLength": 1,
          "maxLength": 128
        }
      },
      "additionalProperties": false,
      "required": [
        "arn"
      ]
    },
    "DataConnector": {
      "description": "The data connector.",
```

```
      "type": "object",
      "properties": {
        "isNative": {
          "description": "A Boolean value that specifies whether the data connector is
native to IoT TwinMaker.",
          "type": "boolean"
        },
        "lambda": {
          "description": "The Lambda function associated with this data connector.",
          "$ref": "#/definitions/LambdaFunction"
        }
      },
      "additionalProperties": false
    },
    "Function": {
      "description": "The function of component type.",
      "type": "object",
      "properties": {
        "implementedBy": {
          "description": "The data connector.",
          "$ref": "#/definitions/DataConnector"
        },
        "requiredProperties": {
          "description": "The required properties of the function.",
          "type": "array",
          "minItems": 1,
          "maxItems": 256,
          "uniqueItems": true,
          "insertionOrder": false,
          "items": {
            "$ref": "#/definitions/RequiredProperty"
          },
          "default": null
        },
        "scope": {
          "description": "The scope of the function.",
          "type": "string",
          "enum": [
            "ENTITY",
            "WORKSPACE"
          ]
        }
      },
      "additionalProperties": false
```

```
    },
    "Entity": {
      "type": "object",
      "properties": {
        "description": {
          "description": "The description of the entity.",
          "$ref": "#/definitions/DescriptionWithDefault"
        },
        "entityId": {
          "$ref": "#/definitions/EntityId"
        },
        "entityExternalId": {
          "description": "The external ID of the entity.",
          "$ref": "#/definitions/ExternalId"
        },
        "entityName": {
          "$ref": "#/definitions/EntityName"
        },
        "parentEntityId": {
          "$ref": "#/definitions/ParentEntityId"
        },
        "tags": {
          "$ref": "#/definitions/Tags"
        },
        "components": {
          "description": "A map that sets information about a component.",
          "type": "object",
          "patternProperties": {
            "[a-zA-Z_\\-0-9]+": {
              "$ref": "#/definitions/Component"
            }
          },
          "additionalProperties": {
            "$ref": "#/definitions/Component"
          }
        }
      },
      "required": [
        "entityId",
        "entityName"
      ],
      "additionalProperties": false
    },
    "ComponentType": {
```

```
      "type": "object",
      "properties": {
        "description": {
          "description": "The description of the component type.",
          "$ref": "#/definitions/DescriptionWithDefault"
        },
        "componentTypeId": {
          "$ref": "#/definitions/ComponentTypeId"
        },
        "componentTypeExternalId": {
          "description": "The external ID of the component type.",
          "$ref": "#/definitions/ExternalId"
        },
        "componentTypeName": {
          "$ref": "#/definitions/ComponentTypeName"
        },
        "extendsFrom": {
          "description": "Specifies the parent component type to extend.",
          "type": "array",
          "minItems": 1,
          "maxItems": 256,
          "uniqueItems": true,
          "insertionOrder": false,
          "items": {
            "$ref": "#/definitions/ComponentTypeId"
          },
          "default": null
        },
        "functions": {
          "description": "a Map of functions in the component type. Each function's key
must be unique to this map.",
          "type": "object",
          "patternProperties": {
            "[a-zA-Z_\\-0-9]+": {
              "$ref": "#/definitions/Function"
            }
          },
          "additionalProperties": {
            "$ref": "#/definitions/Function"
          }
        },
        "isSingleton": {
          "description": "A Boolean value that specifies whether an entity can have
more than one component of this type.",
```

```
          "type": "boolean",
          "default": false
        },
        "propertyDefinitions": {
          "description": "An map of the property definitions in the component type.
Each property definition's key must be unique to this map.",
          "$ref": "#/definitions/PropertyDefinitions"
        },
        "propertyGroups": {
          "description": "An object that maps strings to the property groups to set in
the component type. Each string in the mapping must be unique to this object.",
          "$ref": "#/definitions/PropertyGroups"
        },
        "tags": {
          "$ref": "#/definitions/Tags"
        }
      },
      "required": [
        "componentTypeId"
      ],
      "additionalProperties": false
    },
    "EntityComponent": {
      "type": "object",
      "properties": {
        "entityId": {
          "$ref": "#/definitions/EntityId"
        },
        "componentName": {
          "$ref": "#/definitions/ComponentName"
        },
        "componentExternalId": {
          "description": "The external ID of the component.",
          "$ref": "#/definitions/ExternalId"
        },
        "componentTypeId": {
          "$ref": "#/definitions/ComponentTypeId"
        },
        "description": {
          "description": "The description of the component.",
          "$ref": "#/definitions/Description"
        },
        "properties": {
```

```
          "description": "An object that maps strings to the properties to set in the
component. Each string in the mapping must be unique to this object.",
          "$ref": "#/definitions/Properties"
        },
        "propertyGroups": {
          "description": "An object that maps strings to the property groups to set in
the component. Each string in the mapping must be unique to this object.",
          "$ref": "#/definitions/PropertyGroups"
        }
      },
      "required": [
        "entityId",
        "componentTypeId",
        "componentName"
      ],
      "additionalProperties": false
    }
  },
  "additionalProperties": false,
  "properties": {
    "entities": {
      "type": "array",
      "uniqueItems": false,
      "items": {
        "$ref": "#/definitions/Entity"
      }
    },
    "componentTypes": {
      "type": "array",
      "uniqueItems": false,
      "items": {
        "$ref": "#/definitions/ComponentType"
      }
    },
    "entityComponents": {
      "type": "array",
      "uniqueItems": false,
      "items": {
        "$ref": "#/definitions/EntityComponent"
      },
      "default": null
    }
  }
```

```
}
```

という新しい componentType を作成しcomponent.type.intial、 というエンティティを作成する例を次に示しますinitial。

```
{
  "componentTypes": [
    {
      "componentTypeId": "component.type.initial",
      "tags": {
        "key": "value"
      }
    }
  ],
  "entities": [
    {
      "entityName": "initial",
      "entityId": "initial"
    }
  ]
}
```

既存のエンティティを更新する例を次に示します。

```
{
  "componentTypes": [
    {
      "componentTypeId": "component.type.initial",
      "description": "updated"
    }
  ],
  "entities": [
    {
      "entityName": "parent",
      "entityId": "parent"
    },
    {
      "entityName": "child",
      "entityId": "child",
      "components": {
        "testComponent": {
          "componentTypeId": "component.type.initial",
          "properties": {
```

```
            "testProperty": {
              "definition": {
                "configuration": {
                  "alias": "property"
                },
                "dataType": {
                  "relationship": {
                    "relationshipType": "parent",
                    "targetComponentTypeId": "test"
                  },
                  "type": "STRING",
                  "unitOfMeasure": "t"
                },
                "displayName": "displayName"
              }
            }
          }
        },
        "parentEntityId": "parent"
      }
    ],
    "entityComponents": [
      {
        "entityId": "initial",
        "componentTypeId": "component.type.initial",
        "componentName": "entityComponent",
        "description": "additionalDescription",
        "properties": {
          "additionalProperty": {
            "definition": {
              "configuration": {
                "alias": "additionalProperty"
              },
              "dataType": {
                "type": "STRING"
              },
              "displayName": "additionalDisplayName"
            },
            "value": {
              "stringValue": "test"
            }
          }
        }
```

```
    }
  ]
}
```

AWS IoT TwinMaker データコネクター

AWS IoT TwinMaker コネクタベースのアーキテクチャを使用しているため、独自のデータストアのデータをに接続できます。 AWS IoT TwinMakerつまり、使用する前にデータを移行する必要がないということです。 AWS IoT TwinMaker現在、 AWS IoT TwinMaker はのファーストパーティコネクタをサポートしています。 AWS IoT SiteWiseにモデリングデータとプロパティデータを保存すれば AWS IoT SiteWise、独自のコネクタを実装する必要はありません。モデリングデータまたはプロパティデータを Timestream、DynamoDB、Snowflake などの他のデータストアに保存する場合、 AWS Lambda AWS IoT TwinMaker 必要に応じてコネクタを呼び出せるように、 AWS IoT TwinMaker データコネクタインターフェイスを使用してコネクタを実装する必要があります。

トピック

AWS IoT TwinMaker データコネクター

コネクタは、送信されたクエリを解決し、結果またはエラーを返すために、基になるデータストアにアクセスする必要があります。

利用可能なコネクタ、そのリクエストインターフェース、レスポンスインターフェースについては、以下のトピックを参照してください。

コネクタインターフェースで使用されるプロパティの詳細については、GetPropertyValueHistoryAPIアクションを参照してください。

> **Note**
>
> 一部のコネクタでは、リクエストインターフェースとレスポンスインターフェースの両方に、開始時刻と終了時刻のプロパティ用に2つのタイムスタンプフィールドがあります。`startDateTime`、`endDateTime`のどちらもエポック秒を表すのに長い数字を使用していますが、これはもうサポートされていません。後方互換性を維持するため、このフィールドにはタイムスタンプ値を送信しますが、APIのタイムスタンプ形式と一致する`startTime`フィールドと`endTime`フィールドを使用することをお勧めします。

トピック

スキーマ イニシャライザ コネクタ

コンポーネントタイプまたはエンティティライフサイクルでスキーマ イニシャライザを使用して、基になるデータソースからコンポーネントタイプまたはコンポーネントプロパティを取得できます。スキーマ イニシャライザは、APIアクションを明示的に呼び出してpropertiesをセットアップしなくても、コンポーネントタイプまたはコンポーネントプロパティを自動的にインポートします。

SchemaInitializer リクエストインターフェース

```
{
  "workspaceId": "string",
  "entityId": "string",
  "componentName": "string",
  "properties": {
    // property name as key,
    // value is of type PropertyRequest
    "string": "PropertyRequest"
  }
}
```

> **Note**
>
> このリクエストインターフェースのプロパティマップはPropertyRequestです。詳細については、を参照してくださいPropertyRequest。

SchemaInitializer レスポンスインターフェース

```
{
  "properties": {
    // property name as key,
    // value is of type PropertyResponse
    "string": "PropertyResponse"
  }
}
```

> **Note**
>
> このリクエストインターフェースのプロパティマップはPropertyResponseです。詳細については、を参照してくださいPropertyResponse。

DataReaderByEntity

DataReaderByEntity は、1 つのコンポーネントのプロパティの時系列値を取得するために使用されるデータプレーンコネクタです。

このコネクタのプロパティタイプ、構文、形式については、GetPropertyValueHistoryAPI アクションを参照してください。

DataReaderByEntityリクエストインターフェース

```
{
  "startDateTime": long, // In epoch sec, deprecated
  "startTime": "string", // ISO-8601 timestamp format
  "endDateTime": long, // In epoch sec, deprecated
  "endTime": "string", // ISO-8601 timestamp format
  "properties": {
    // A map of properties as in the get-entity API response
    // property name as key,
    // value is of type PropertyResponse
    "string": "PropertyResponse"
   },
  "workspaceId": "string",
  "selectedProperties": List:"string",
  "propertyFilters": List:PropertyFilter,
```

```
  "entityId": "string",
  "componentName": "string",
  "componentTypeId": "string",
  "interpolation": InterpolationParameters,
  "nextToken": "string",
  "maxResults": int,
  "orderByTime": "string"
  }
```

DataReaderByEntityレスポンスインターフェース

```
{
  "propertyValues": [
    {
      "entityPropertyReference": EntityPropertyReference, // The same
 as EntityPropertyReference
      "values": [
        {
        "timestamp": long, // Epoch sec, deprecated
        "time": "string", // ISO-8601 timestamp format
        "value": DataValue // The same as DataValue
        }
      ]
    }
  ],
  "nextToken": "string"
}
```

DataReaderByComponentType

同じコンポーネントタイプに含まれる共通プロパティの時系列値を取得するには、DataReaderByEntityデータプレーンコネクタを使用します。たとえば、コンポーネントタイプで時系列プロパティを定義していて、そのコンポーネントタイプを使用するコンポーネントが複数ある場合、特定の時間範囲内のすべてのコンポーネントでそれらのプロパティをクエリできます。一般的な使用例としては、複数のコンポーネントのアラームステータスをクエリしてエンティティをグローバルに把握したい場合です。

このコネクタのプロパティタイプ、構文、形式については、GetPropertyValueHistoryAPI アクションを参照してください。

DataReaderByComponentType リクエストインターフェース

```
{
  "startDateTime": long, // In epoch sec, deprecated
  "startTime": "string", // ISO-8601 timestamp format
  "endDateTime": long, // In epoch sec, deprecated
  "endTime": "string", // ISO-8601 timestamp format
  "properties": { // A map of properties as in the get-entity API response
    // property name as key,
    // value is of type PropertyResponse
    "string": "PropertyResponse"
   },
  "workspaceId": "string",
  "selectedProperties": List:"string",
  "propertyFilters": List:PropertyFilter,
  "componentTypeId": "string",
  "interpolation": InterpolationParameters,
  "nextToken": "string",
  "maxResults": int,
  "orderByTime": "string"
}
```

DataReaderByComponentType レスポンスインターフェース

```
{
  "propertyValues": [
    {
      "entityPropertyReference": EntityPropertyReference, // The same
 as EntityPropertyReference
      "entityId": "string",
      "componentName": "string",
      "values": [
        {
        "timestamp": long, // Epoch sec, deprecated
        "time": "string", // ISO-8601 timestamp format
        "value": DataValue // The same as DataValue
        }
      ]
    }
  ],
  "nextToken": "string"
}
```

DataReader

DataReader DataReaderByEntity との両方に対応できるデータプレーンコネクタです DataReaderByComponentType。

このコネクタのプロパティタイプ、構文、形式については、GetPropertyValueHistoryAPI アクションを参照してください。

DataReader リクエストインターフェース

`EntityId`および`componentName`はオプションです。

```
{
  "startDateTime": long, // In epoch sec, deprecated
  "startTime": "string", // ISO-8601 timestamp format
  "endDateTime": long, // In epoch sec, deprecated
  "endTime": "string", // ISO-8601 timestamp format
  "properties": { // A map of properties as in the get-entity API response
    // property name as key,
    // value is of type PropertyRequest
    "string": "PropertyRequest"
  },

  "workspaceId": "string",
  "selectedProperties": List:"string",
  "propertyFilters": List:PropertyFilter,
  "entityId": "string",
  "componentName": "string",
  "componentTypeId": "string",
  "interpolation": InterpolationParameters,
  "nextToken": "string",
  "maxResults": int,
  "orderByTime": "string"
}
```

DataReader レスポンスインターフェース

```
{
  "propertyValues": [
    {
      "entityPropertyReference": EntityPropertyReference, // The same
 as EntityPropertyReference
      "values": [
```

```
        {
        "timestamp": long, // Epoch sec, deprecated
        "time": "string", // ISO-8601 timestamp format
        "value": DataValue // The same as DataValue
        }
      ]
    }
  ],
  "nextToken": "string"
}
```

AttributePropertyValueReaderByEntity

AttributePropertyValueReaderByEntity は、1 つのエンティティの静的プロパティの値を取得するために使用できるデータプレーンコネクタです。

このコネクタのプロパティタイプ、構文、形式については、 GetPropertyValueAPI アクションを参照してください。

AttributePropertyValueReaderByEntity リクエストインターフェース

```
{
  "properties": {
    // property name as key,
    // value is of type PropertyResponse
    "string": "PropertyResponse"
  }

  "workspaceId": "string",
  "entityId": "string",
  "componentName": "string",
  "selectedProperties": List:"string",
}
```

AttributePropertyValueReaderByEntity レスポンスインターフェース

```
{
  "propertyValues": {
    "string": { // property name as key
        "propertyReference": EntityPropertyReference, // The same
 as EntityPropertyReference
        "propertyValue": DataValue // The same as DataValue
```

```
    }
}
```

DataWriter

DataWriter は、単一コンポーネントのプロパティの時系列データポイントを基になるデータストアに書き戻すために使用できるデータプレーンコネクタです。

このコネクタのプロパティタイプ、構文、形式については、BatchPutPropertyValuesAPI アクションを参照してください。

DataWriter リクエストインターフェース

```
{
  "workspaceId": "string",
  "properties": {
    // entity id as key
    "String": {
      // property name as key,
      // value is of type PropertyResponse
      "string": PropertyResponse
    }
  },
  "entries": [
    {
      "entryId": "string",
      "entityPropertyReference": EntityPropertyReference, // The same
 as EntityPropertyReference
      "propertyValues": [
        {
        "timestamp": long, // Epoch sec, deprecated
        "time": "string", // ISO-8601 timestamp format
        "value": DataValue // The same as DataValue
        }
      ]
    }
  ]
}
```

DataWriter レスポンスインターフェース

```
{
```

```
  "errorEntries": [
    {
      "errors": List:BatchPutPropertyError // The value is a list of
 type BatchPutPropertyError
    }
  ]
}
```

例

以下のJSONサンプルは、複数のコネクタのレスポンスおよびリクエスト構文の例です。

- SchemaInitializer:

  以下の例は、コンポーネントタイプのライフサイクルにおけるスキーマイニシャライザーを示しています。

  リクエスト：

```
{
  "workspaceId": "myWorkspace",
  "properties": {
    "modelId": {
      "definition": {
          "dataType": { "type": "STRING" },
          "isExternalId": true,
          "isFinal": true,
          "isImported": false,
          "isInherited": false,
          "isRequiredInEntity": true,
          "isStoredExternally": false,
          "isTimeSeries": false,
          "defaultValue": {
              "stringValue": "myModelId"
          }
      },
      "value": {
          "stringValue": "myModelId"
      }
    },
    "tableName": {
      "definition": {
          "dataType": { "type": "STRING" },
```

```
          "isExternalId": false,
          "isFinal": false,
          "isImported": false,
          "isInherited": false,
          "isRequiredInEntity": false,
          "isStoredExternally": false,
          "isTimeSeries": false,
          "defaultValue": {
              "stringValue": "myTableName"
          }
      },
      "value": {
          "stringValue": "myTableName"
      }
    }
  }
}
```

レスポンス：

```
{
  "properties": {
    "myProperty1": {
      "definition": {
        "dataType": {
          "type": "DOUBLE",
          "unitOfMeasure": "%"
        },
        "configuration": {
          "myProperty1Id": "idValue"
        },
        "isTimeSeries": true
      }
    },
    "myProperty2": {
      "definition": {
        "dataType": { "type": "STRING" },
        "isTimeSeries": false,
        "defaultValue": {
          "stringValue": "property2Value"
        }
      }
    }
```

```
  }
}
```

- エンティティ ライフサイクルのスキーマ イニシャライザー：

  リクエスト：

```
{
  "workspaceId": "myWorkspace",
  "entityId": "myEntity",
  "componentName": "myComponent",
  "properties": {
    "assetId": {
      "definition": {
          "dataType": { "type": "STRING" },
          "isExternalId": true,
          "isFinal": true,
          "isImported": false,
          "isInherited": false,
          "isRequiredInEntity": true,
          "isStoredExternally": false,
          "isTimeSeries": false
      },
      "value": {
          "stringValue": "myAssetId"
      }
    },
    "tableName": {
      "definition": {
          "dataType": { "type": "STRING" },
          "isExternalId": false,
          "isFinal": false,
          "isImported": false,
          "isInherited": false,
          "isRequiredInEntity": false,
          "isStoredExternally": false,
          "isTimeSeries": false
      },
      "value": {
          "stringValue": "myTableName"
      }
    }
  }
```

```
}
```

レスポンス：

```
{
  "properties": {
    "myProperty1": {
      "definition": {
        "dataType": {
          "type": "DOUBLE",
          "unitOfMeasure": "%"
        },
        "configuration": {
          "myProperty1Id": "idValue"
        },
        "isTimeSeries": true
      }
    },
    "myProperty2": {
      "definition": {
        "dataType": { "type": "STRING" },
        "isTimeSeries": false
      },
      "value": {
        "stringValue": "property2Value"
      }
    }
  }
}
```

- DataReaderByEntity と DataReader:

  リクエスト：

```
{
  "workspaceId": "myWorkspace",
  "entityId": "myEntity",
  "componentName": "myComponent",
  "selectedProperties": [
    "Temperature",
    "Pressure"
  ],
  "startTime": "2022-04-07T04:04:42Z",
```

```
    "endTime": "2022-04-07T04:04:45Z",
    "maxResults": 4,
    "orderByTime": "ASCENDING",
    "properties": {
        "assetId": {
            "definition": {
                "dataType": { "type": "STRING" },
                "isExternalId": true,
                "isFinal": true,
                "isImported": false,
                "isInherited": false,
                "isRequiredInEntity": true,
                "isStoredExternally": false,
                "isTimeSeries": false
            },
            "value": {
                "stringValue": "myAssetId"
            }
        },
        "Temperature": {
            "definition": {
                "configuration": {
                    "temperatureId": "xyz123"
                },
                "dataType": {
                    "type": "DOUBLE",
                    "unitOfMeasure": "DEGC"
                },
                "isExternalId": false,
                "isFinal": false,
                "isImported": true,
                "isInherited": false,
                "isRequiredInEntity": false,
                "isStoredExternally": false,
                "isTimeSeries": true
            }
        },
        "Pressure": {
            "definition": {
                "configuration": {
                    "pressureId": "xyz456"
                },
                "dataType": {
                    "type": "DOUBLE",
```

```
                    "unitOfMeasure": "MPA"
                },
                "isExternalId": false,
                "isFinal": false,
                "isImported": true,
                "isInherited": false,
                "isRequiredInEntity": false,
                "isStoredExternally": false,
                "isTimeSeries": true
            }
        }
    }
}
```

レスポンス：

```
{
  "propertyValues": [
    {
      "entityPropertyReference": {
        "entityId": "myEntity",
        "componentName": "myComponent",
        "propertyName": "Temperature"
      },
      "values": [
        {
          "time": "2022-04-07T04:04:42Z",
          "value": {
            "doubleValue": 588.168
          }
        },
        {
          "time": "2022-04-07T04:04:43Z",
          "value": {
            "doubleValue": 592.4224
          }
        }
      ]
    }
  ],
  "nextToken": "qwertyuiop"
}
```

- AttributePropertyValueReaderByEntity:

  リクエスト：

```
{
  "workspaceId": "myWorkspace",
  "entityId": "myEntity",
  "componentName": "myComponent",
  "selectedProperties": [
    "manufacturer",
  ],
  "properties": {
    "assetId": {
      "definition": {
        "dataType": { "type": "STRING" },
        "isExternalId": true,
        "isFinal": true,
        "isImported": false,
        "isInherited": false,
        "isRequiredInEntity": true,
        "isStoredExternally": false,
        "isTimeSeries": false
      },
      "value": {
          "stringValue": "myAssetId"
      }
    },
    "manufacturer": {
      "definition": {
        "dataType": { "type": "STRING" },
        "configuration": {
            "manufacturerPropId": "M001"
        },
        "isExternalId": false,
        "isFinal": false,
        "isImported": false,
        "isInherited": false,
        "isRequiredInEntity": false,
        "isStoredExternally": true,
        "isTimeSeries": false
      }
    }
  }
```

```
}
```

レスポンス：

```
{
  "propertyValues": {
    "manufacturer": {
      "propertyReference": {
        "propertyName": "manufacturer",
        "entityId": "myEntity",
        "componentName": "myComponent"
      },
      "propertyValue": {
        "stringValue": "Amazon"
      }
    }
  }
}
```

- DataWriter:

  リクエスト：

```
{
  "workspaceId": "myWorkspaceId",
  "properties": {
    "myEntity": {
      "Temperature": {
          "definition": {
              "configuration": {
                  "temperatureId": "xyz123"
              },
              "dataType": {
                  "type": "DOUBLE",
                  "unitOfMeasure": "DEGC"
              },
              "isExternalId": false,
              "isFinal": false,
              "isImported": true,
              "isInherited": false,
              "isRequiredInEntity": false,
              "isStoredExternally": false,
```

```
              "isTimeSeries": true
          }
      }
    }
  },
  "entries": [
    {
      "entryId": "myEntity",
      "entityPropertyReference": {
        "entityId": "myEntity",
        "componentName": "myComponent",
        "propertyName": "Temperature"
      },
      "propertyValues": [
        {
          "timestamp": 1626201120,
          "value": {
            "doubleValue": 95.6958
          }
        },
        {
          "timestamp": 1626201132,
          "value": {
            "doubleValue": 80.6959
          }
        }
      ]
    }
  ]
}
```

レスポンス：

```
{
    "errorEntries": [
        {
            "errors": [
                {
                    "errorCode": "409",
                    "errorMessage": "Conflict value at same timestamp",
                    "entry": {
                        "entryId": "myEntity",
```

```
                        "entityPropertyReference": {
                            "entityId": "myEntity",
                            "componentName": "myComponent",
                            "propertyName": "Temperature"
                        },
                        "propertyValues": [
                            "time": "2022-04-07T04:04:42Z",
                            "value": {
                                "doubleValue": 95.6958
                            }
                        ]
                    }
                }
            ]
        }
    ]
}
```

AWS IoT TwinMaker Athena テーブルデータコネクター

Athena表形式データコネクタを使用すると、 AWS IoT TwinMakerにあるAthenaデータストアにアクセスして使用できます。大量のデータ移行作業を行わなくても、Athenaデータを使用してデジタルツインを構築できます。Athena データソースのデータにアクセスするには、事前構築済みのコネクタを使用するか、カスタム Athena コネクタを作成できます。

AWS IoT TwinMaker Athena データコネクタの前提条件

Athena 表形式データコネクタを使用する前に、以下の前提条件を満たしてください。

- マネージド Athena テーブルとそれに関連付けられた Amazon S3 リソースを作成します。Athena の使用については、Athenaのドキュメントを参照してください。。
- AWS IoT TwinMaker ワークスペースを作成します。ワークスペースは、AWS IoT TwinMaker コンソールで作成できます。
- ワークスペースの IAM ロールを Athena 権限で更新します。詳細については、「Athena データコネクタを使用するようにワークスペース IAM ロールを変更する」を参照してください。
- AWS IoT TwinMakerのエンティティコンポーネントシステムとエンティティの作成方法に慣れてください。詳細については、「最初のエンティティを作成する」を参照してください。
- AWS IoT TwinMakerのデータコネクタに慣れてください。詳細については、「AWS IoT TwinMaker データコネクター」を参照してください。

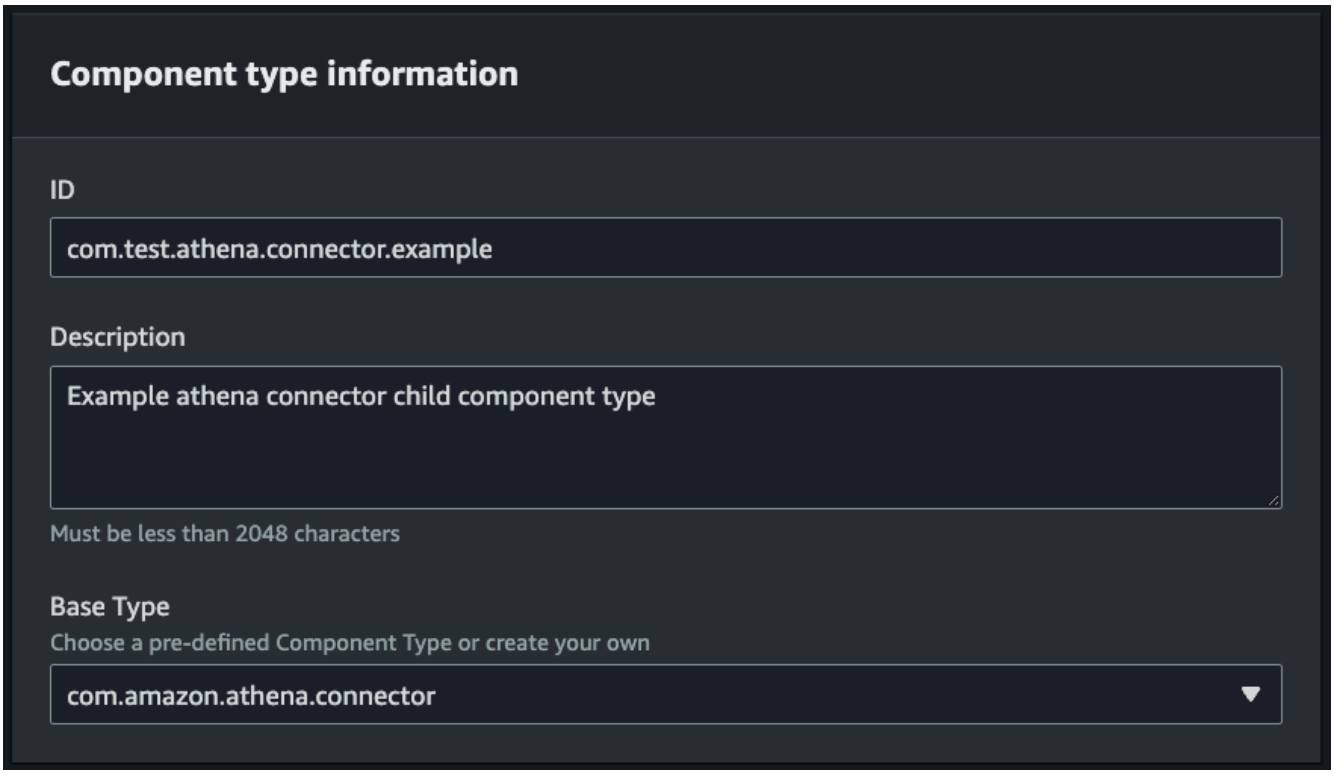
Athenaデータコネクタを使用する

Athenaデータコネクタを使用するには、Athenaコネクタをコンポーネントタイプとして使用してコンポーネントを作成する必要があります。次に、 AWS IoT TwinMakerで使用するために、コンポーネントをシーン内のエンティティにアタッチします。

Athenaデータコネクタを使用してコンポーネントタイプを作成する

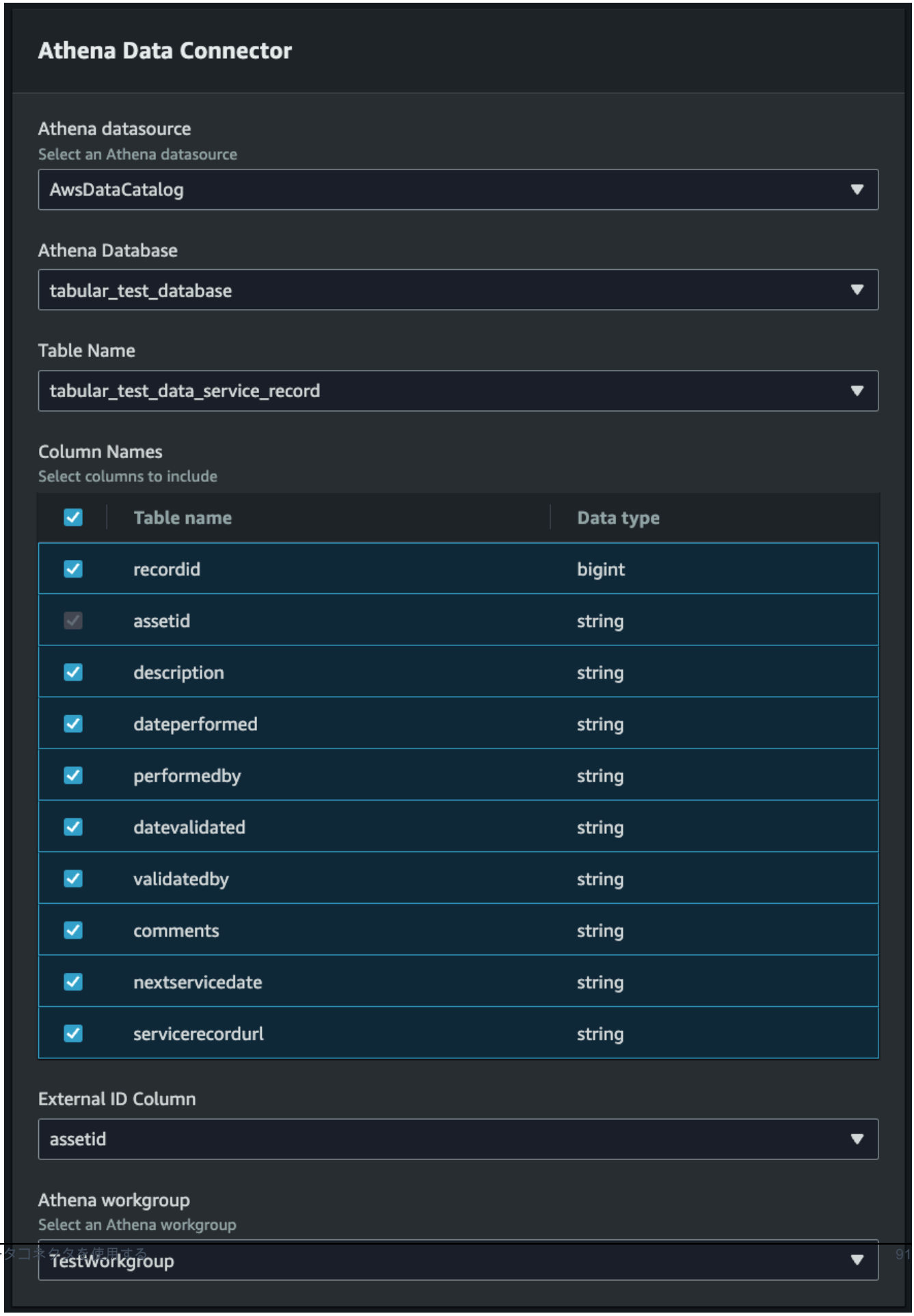
以下の手順に従って、Athena AWS IoT TwinMaker 表形式データコネクタでコンポーネントタイプを作成します。

1. AWS IoT TwinMaker コンソールに移動します。
2. 既存のフローを開くか、新しく作成します。
3. 左側のナビゲーションメニューから「コンポーネントタイプ」を選択し、「コンポーネントタイプの作成」を選択してコンポーネントタイプ作成ページを開きます。
4. コンポーネントタイプの作成ページで、IDフィールドにユースケースに一致するIDを入力します。



5. ベースタイプを選択します。ドロップダウンリストから、com.amazon.athena.connectorというラベルが付いているAthena表形式データコネクタを選択します。

6. 以下のフィールドでAthenaリソースを選択して、コンポーネントタイプのデータソースを設定します。
   - Athena データソースを選択してください。
   - Athenaデータベースを選択します。
   - テーブル名を選択します。
   - Athena workGroupを選択します。
7. データソースとして使用するAthenaリソースを選択したら、含める列をテーブルから選択します。
8. 外部ID列名を選択します。前のステップから外部ID列として使用する列を選択します。外部 ID は Athena アセットを表し、 AWS IoT TwinMaker それをエンティティにマップするために使用される ID です。


Athena Data Connector
Athena datasource
Select an Athena datasource
AwsDataCatalog
Athena Database
tabular_test_database
Table Name
tabular_test_data_service_record
Column Names
Select columns to include
Table name
Data type
recordid
bigint
assetid
string
description
string
dateperformed
string
performedby
string
datevalidated
string
validatedby
string
comments
string
nextservicedate
string
serviceRecordUrl
string
External ID Column
assetid
Athena workgroup
Select an Athena workgroup
TestWorkgroup

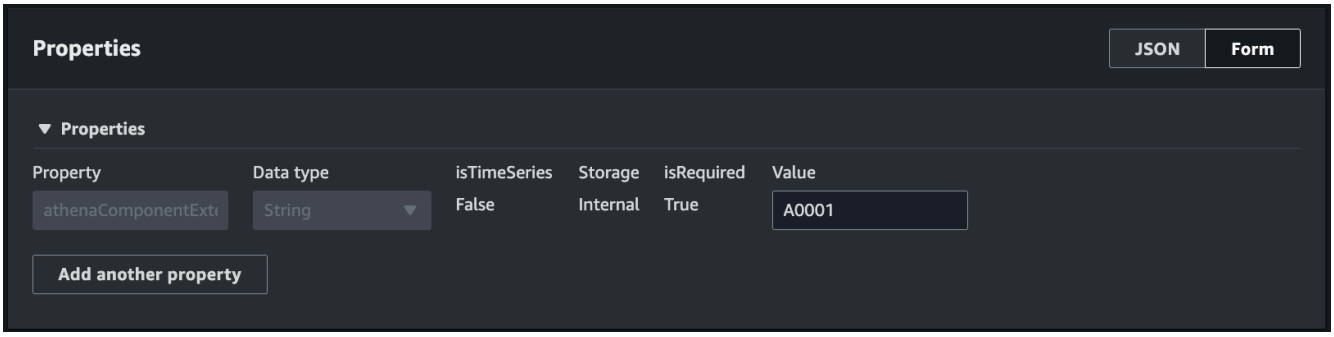
9. (オプション) AWS これらのリソースにタグを追加すると、リソースをグループ化して整理できます。
10. 「コンポーネントタイプの作成」を選択して、コンポーネントタイプの作成を終了します。

Athenaデータコネクタタイプのコンポーネントを作成し、エンティティにアタッチします。

以下の手順を使用して、Athena AWS IoT TwinMaker 表形式データコネクタを使用してコンポーネントを作成し、エンティティにアタッチします。

> **Note**
>
> この手順を完了するには、Athena表形式データコネクタをデータソースとして使用する既存のコンポーネントタイプが必要です。このウォークスルーを開始する前に、前の手順「Athenaデータコネクタを使用してコンポーネントタイプを作成する」を参照してください。

1. AWS IoT TwinMaker コンソールに移動します。
2. 既存のフローを開くか、新しく作成します。
3. 左側のナビゲーションメニューから [Entities] を選択し、コンポーネントを追加するエンティティを選択するか、新しいエンティティを作成します。
4. 新しいエンティティを作成します。
5. 次に「コンポーネントを追加」を選択します。次に、「コンポーネント名」フィールドに、ユースケースに合った名前を入力します。
6. 「コンポーネントタイプ」ドロップダウンメニューから、前の手順で作成したコンポーネントタイプIDを選択します。
7. コンポーネント情報とコンポーネント名を入力し、 ComponentType 以前に作成した子を選択します。これは Athena ComponentType データコネクタで作成したものです。
8. プロパティセクションに、コンポーネントの athenaComponentExternalID を入力します。



9. 「コンポーネントを追加」を選択して、コンポーネントの作成を終了します。

これで、Athenaデータコネクタをコンポーネントタイプとして使用するコンポーネントの作成とエンティティへのアタッチが完了しました。

Athena表形式データコネクタJSONリファレンスの使用

以下の例は、Athena表形式データコネクタの完全なJSONリファレンスです。これをリソースとして使用して、カスタムデータコネクタとコンポーネントタイプを作成します。

```
{
    "componentTypeId": "com.amazon.athena.connector",
    "description": "Athena connector for syncing tabular data",
    "workspaceId":"AmazonOwnedTypesWorkspace",
    "propertyGroups": {
        "tabularPropertyGroup": {
            "groupType": "TABULAR",
            "propertyNames": []
        }
    },
    "propertyDefinitions": {
        "athenaDataSource": {
            "dataType": { "type": "STRING" },
            "isRequiredInEntity": true
        },
        "athenaDatabase": {
            "dataType": { "type": "STRING" },
            "isRequiredInEntity": true
        },
        "athenaTable": {
            "dataType": { "type": "STRING" },
            "isRequiredInEntity": true
        },
```

```
        "athenaWorkgroup": {
            "dataType": { "type": "STRING" },
            "isRequiredInEntity": true
        },
        "athenaExternalIdColumnName": {
            "dataType": { "type": "STRING" },
            "isRequiredInEntity": true,
            "isExternalId": false
        },
        "athenaComponentExternalId": {
            "dataType": { "type": "STRING" },
            "isStoredExternally": false,
            "isRequiredInEntity": true,
            "isExternalId": true
        }
    },
    "functions": {
        "tabularDataReaderByEntity": {
            "implementedBy": {
                "isNative": true
            }
        }
    }
}
```

Athenaデータコネクタを使用する

GrafanaではAthenaテーブルを使用しているエンティティを表示できます。詳細については、「AWS IoT TwinMaker Grafanaダッシュボードのインテグレーション」を参照してください。

Athenaテーブルを作成して使用してデータを保存する方法については、Athenaのドキュメントを参照してください。

Athenaデータコネクタのトラブルシューティング

このトピックでは、Athena データコネクタの設定時に発生する可能性のある一般的な問題について説明します。

Athenaワークグループの場所：

> Athenaコネクタコンポーネントタイプを作成する場合、Athenaワークグループには出力場所を設定する必要があります。ワークグループの仕組みをご覧ください。

IAMロールの権限がありません：

> ComponentType の作成、エンティティへの Ca コンポーネントの追加、または API の実行時に AWS IoT TwinMaker; ワークスペースロールに Athena API アクセス権限がない場合があります。GetPropertyValue IAM 権限を更新するには、「のサービスロールの作成と管理」を参照してください。 AWS IoT TwinMaker

Athenaの表形式データをGrafanaで視覚化する

Grafana プラグインを使用すると、Athena への API 呼び出しやインタラクションを行わずに、選択したプロパティに基づく並べ替えやフィルタリングなどの追加機能を備えたダッシュボードパネルであるGrafana a上の表形式のデータを視覚化できます。 AWS IoT TwinMakerこのトピックでは、Athenaの表形式データを視覚化するようにGrafanaを設定する方法を説明します。

前提条件

Athenaの表形式データを視覚化するようにGrafanaパネルを設定する前に、以下の前提条件を確認してください。

- これでGrafana環境がセットアップされました。詳細については、「AWS IoT TwinMaker Grafana のインテグレーション」を参照してください。
- Grafanaデータソースを設定できます。詳細については、「AWS IoT TwinMaker Grafana」を参照してください。
- 新しいダッシュボードを作成し、新しいパネルを追加することはよくご存知のことと思います。

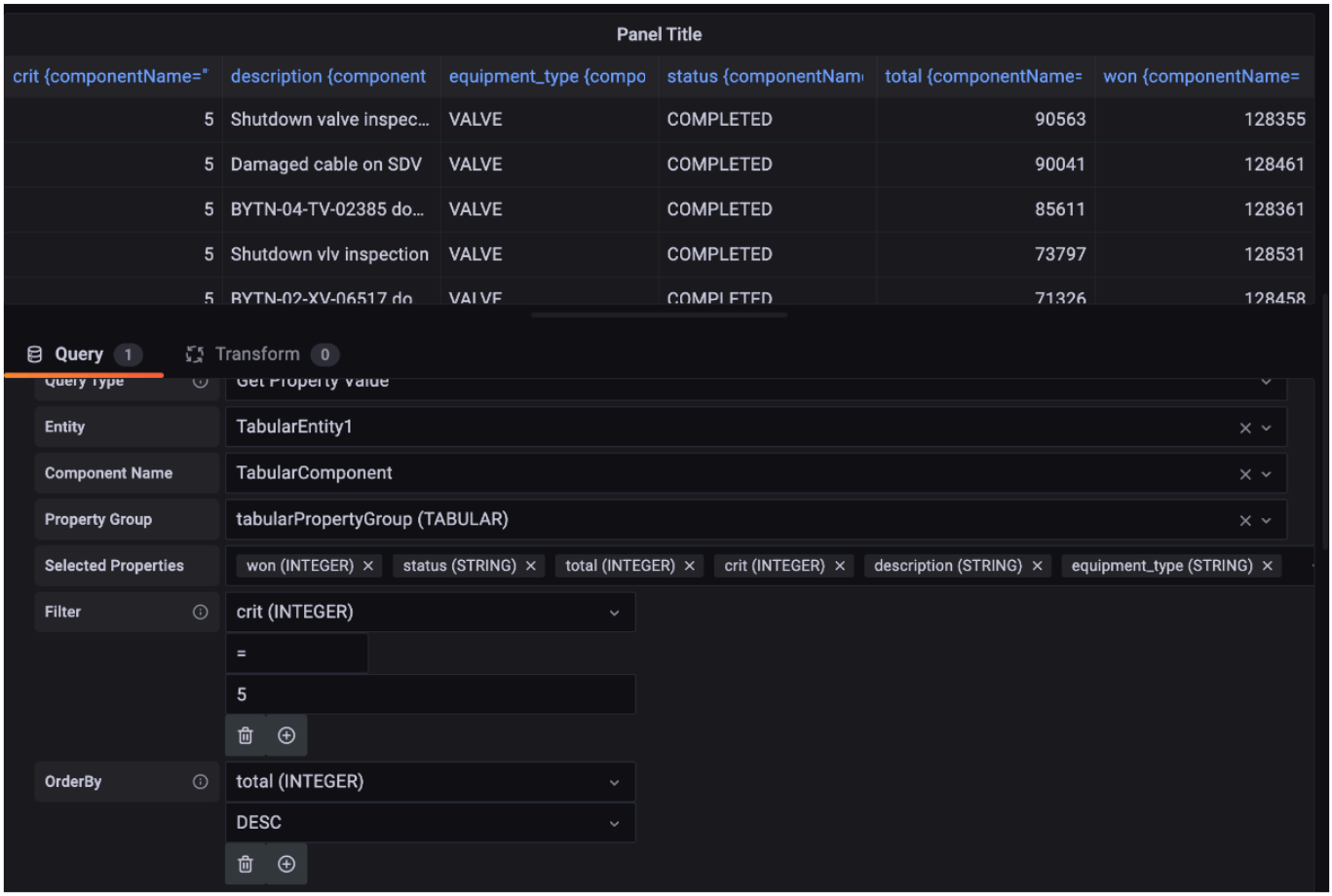
Athenaの表形式データをGrafanaで視覚化する

この手順では、Grafanaパネルを使用してAthenaの表形式のデータを視覚化する方法を示します。

1. AWS IoT TwinMaker Grafana ダッシュボードを開きます。
2. パネル設定でテーブルパネルを選択します。
3. クエリ設定でデータソースを選択します。
4. 「プロパティ値の取得」クエリを選択します。
5. エンティティを選択します。
6. Athenaベースコンポーネントタイプを拡張するComponentTypeを持つコンポーネントを選択します。

7. Athenaテーブルのプロパティグループを選択します。
8. プロパティグループから任意の数のプロパティを選択します。
9. フィルターとプロパティの順序のリストを使用して表形式の条件を設定します。次のオプションを設定します。

   - フィルター：プロパティ値の式を定義してデータをフィルターします。
   - OrderBy: プロパティのデータを昇順または降順のどちらで返すかを指定します。



AWS IoT TwinMaker 時系列データコネクタの開発

このセクションでは、プロセスで時系列データコネクタを開発する方法について説明します。 step-by-stepさらに、3Dモデル、エンティティ、コンポーネント、アラーム、およびコネクタを含む、クッキーファクトリーサンプル全体に基づく時系列データコネクタの例を紹介します。Cookie [AWS IoT TwinMaker GitHub ファクトリのサンプルソースはサンプルリポジトリにあります]。

トピック

AWS IoT TwinMaker 時系列データコネクタの前提条件

時系列データコネクタを開発する前に、次のタスクを完了することをお勧めします。

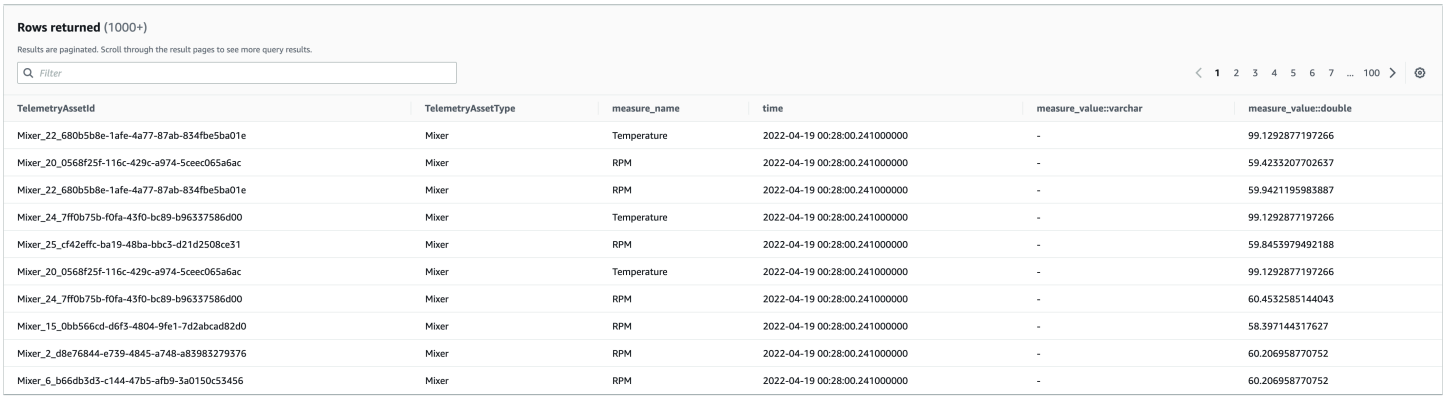
- AWS IoT TwinMaker ワークスペースを作成します。
- AWS IoT TwinMaker コンポーネントタイプを作成します。
- AWS IoT TwinMaker エンティティを作成します。
- （オプション）「コンポーネントタイプの使用と作成」をお読みください。
- （オプション）AWS IoT TwinMaker データ コネクタ インターフェースを読んで、 AWS IoT TwinMaker データコネクタの一般的な理解を深めてください。

> **Note**
>
> 完全に実装されたコネクタの例については、クッキーファクトリーの実装例をご覧ください。

時系列データコネクタの背景

クッキーミキサーと水タンクを備えた工場で働いているところを想像してみてください。 AWS IoT TwinMaker こうした物理エンティティのデジタルツインを構築して、さまざまな時系列メトリクスをチェックして運用状態を監視できるようにしたいと考えています。

現場のセンサをセットアップし、測定データをすでにTimestreamデータベースにストリーミングしています。オーバーヘッドを最小限に抑えながら、 AWS IoT TwinMaker で測定データを表示し、整理できるようにしたいものです。このタスクは時系列データコネクタを使用して実行できます。以下の画像は、時系列コネクタを使用して入力されるテレメトリテーブルの例を示しています。



Rows returned (1000+)

Results are paginated. Scroll through the result pages to see more query results.

Filter

1 2 3 4 5 6 7 ... 100

TelemetryAssetId	TelemetryAssetType	measure_name	time	measure_value::varchar	measure_value::double
Mixer_22_680b5b8e-1afe-4a77-87ab-834fbe5ba01e	Mixer	Temperature	2022-04-19 00:28:00.241000000	-	99.1292877197266
Mixer_20_0568f25f-116c-429c-a974-5ceec065a6ac	Mixer	RPM	2022-04-19 00:28:00.241000000	-	59.4233207702637
Mixer_22_680b5b8e-1afe-4a77-87ab-834fbe5ba01e	Mixer	RPM	2022-04-19 00:28:00.241000000	-	59.9421195983887
Mixer_24_7ff0b75b-f0fa-43f0-bc89-b96337586d00	Mixer	Temperature	2022-04-19 00:28:00.241000000	-	99.1292877197266
Mixer_25_cf42effc-ba19-48ba-bbc3-d21d2508ce31	Mixer	RPM	2022-04-19 00:28:00.241000000	-	59.8453979492188
Mixer_20_0568f25f-116c-429c-a974-5ceec065a6ac	Mixer	Temperature	2022-04-19 00:28:00.241000000	-	99.1292877197266
Mixer_24_7ff0b75b-f0fa-43f0-bc89-b96337586d00	Mixer	RPM	2022-04-19 00:28:00.241000000	-	60.4532585144043
Mixer_15_0bb566cd-d6f3-4804-9fe1-7d2abcad82d0	Mixer	RPM	2022-04-19 00:28:00.241000000	-	58.397144317627
Mixer_2_d8e76844-e739-4845-a748-a83983279376	Mixer	RPM	2022-04-19 00:28:00.241000000	-	60.206958770752
Mixer_6_b66db3d3-c144-47b5-afb9-3a0150c53456	Mixer	RPM	2022-04-19 00:28:00.241000000	-	60.206958770752

このスクリーンショットで使用されているデータセットと Timestream テーブルは、サンプルリポジトリにありますAWS IoT TwinMaker 。 GitHub 前のスクリーンショットに示されている結果を生成する、実装用のクッキー ファクトリのサンプルコネクタも参照してください。

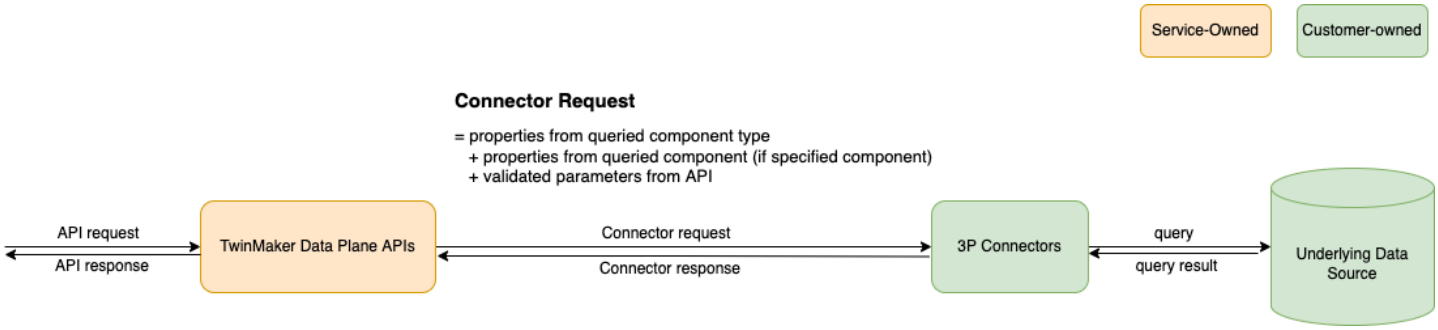
時系列データコネクタのデータフロー

データプレーンクエリでは、コンポーネントとコンポーネントタイプの定義から、 AWS IoT TwinMaker コンポーネントとコンポーネントタイプの両方の対応するプロパティを取得します。AWS IoT TwinMaker クエリ内の API AWS Lambda クエリパラメータとともにプロパティを関数に転送します。

AWS IoT TwinMaker Lambda 関数を使用して、データソースからのクエリにアクセスして解決し、それらのクエリの結果を返します。Lambda関数は、データプレーンのコンポーネントとコンポーネントタイプのプロパティを使用して最初のリクエストを解決します。

Lambdaクエリの結果はAPIレスポンスにマッピングされ、ユーザーに返されます。

AWS IoT TwinMaker データコネクタインターフェイスを定義し、それを使用して Lambda 関数とやり取りします。データコネクタを使用すると、データ移行の手間をかけずに AWS IoT TwinMaker APIからデータソースをクエリできます。次の図は、前の段落で説明した基本的なデータフローの概要を示しています。



時系列データコネクタの開発

以下の手順は、機能的な時系列データコネクタまで段階的に構築する開発モデルの概要を示しています。基本的なステップは次のとおりです。

1. 有効な基本コンポーネントタイプの作成

   コンポーネントタイプでは、コンポーネント間で共有される共通のプロパティを定義します。コンポーネントタイプの定義について詳しくは、「コンポーネントタイプの使用と作成」を参照してください。

   AWS IoT TwinMaker エンティティ・コンポーネント・モデリング・パターンを使用するため、各コンポーネントはエンティティにアタッチされます。各物理アイテムを 1 つのエンティティとしてモデル化し、異なるデータソースを独自のコンポーネントタイプでモデル化することをお勧めします。

   次の例では、1つのプロパティを使用したTimestreamテンプレートの例を示します。

```
{"componentTypeId": "com.example.timestream-telemetry",
    "workspaceId": "MyWorkspace",
    "functions": {
        "dataReader": {
            "implementedBy": {
                "lambda": {
                    "arn": "lambdaArn"
                }
            }
        }
    },
    "propertyDefinitions": {
        "telemetryType": {
            "dataType": { "type": "STRING" },
            "isExternalId": false,
```

```
            "isStoredExternally": false,
            "isTimeSeries": false,
            "isRequiredInEntity": true
        },
        "telemetryId": {
            "dataType": { "type": "STRING" },
            "isExternalId": true,
            "isStoredExternally": false,
            "isTimeSeries": false,
            "isRequiredInEntity": true
        },
        "Temperature": {
            "dataType": { "type": "DOUBLE" },
            "isExternalId": false,
            "isTimeSeries": true,
            "isStoredExternally": true,
            "isRequiredInEntity": false
        }
    }
}
```

コンポーネントタイプの主な要素は次のとおりです。

- `telemetryId`このプロパティは、対応するデータソース内の物理アイテムの固有キーを識別します。データコネクタはこのプロパティをフィルター条件として使用し、特定のアイテムに関連する値のみをクエリします。さらに、データプレーンAPIレスポンスに`telemetryId`プロパティ値を含めると、クライアント側がIDを取得し、必要に応じて逆引きを行うことができます。
- `lambdaArn`フィールドは、コンポーネントタイプが関与するLambda関数を識別します。
- `isRequiredInEntity`フラグはIDの作成を強制します。このフラグは、コンポーネントの作成時にアイテムの ID もインスタンス化されるために必要です。
- `TelemetryId`は Timestream テーブルで項目を識別できるように、外部 ID としてコンポーネントタイプに追加されます。

2. そのコンポーネントタイプでコンポーネントを作成

   作成したコンポーネントタイプを使用するには、コンポーネントを作成して、データを取得したいエンティティにアタッチする必要があります。以下のステップでは、そのコンポーネントを作成するプロセスを詳しく説明します。

   a. AWS IoT TwinMaker コンソールに移動します。

b. コンポーネントタイプを作成したのと同じワークスペースを選択して開きます。

c. エンティティのページに移動します。

d. 新しいエンティティを作成するか、テーブルから既存のエンティティを選択します。

e. 使用するエンティティを選択したら、「コンポーネントの追加」を選択して「コンポーネントの追加」ページを開きます。

f. コンポーネントに名前を付け、タイプには1でテンプレートで作成したコンポーネントタイプを選択します。有効な基本コンポーネントタイプを作成します。

3. コンポーネントタイプをLambdaコネクタに呼び出すようにする

Lambdaコネクタは、データソースにアクセスし、入力に基づいてクエリステートメントを生成し、それをデータソースに転送する必要があります。次の例は、これを行う JSON リクエストテンプレートを示しています。

```
{
  "workspaceId": "MyWorkspace",
  "entityId": "MyEntity",
  "componentName": "TelemetryData",
  "selectedProperties": ["Temperature"],
  "startTime": "2022-08-25T00:00:00Z",
  "endTime": "2022-08-25T00:00:05Z",
  "maxResults": 3,
  "orderByTime": "ASCENDING",
  "properties": {
      "telemetryType": {
          "definition": {
              "dataType": { "type": "STRING" },
              "isExternalId": false,
              "isFinal": false,
              "isImported": false,
              "isInherited": false,
              "isRequiredInEntity": false,
              "isStoredExternally": false,
              "isTimeSeries": false
          },
          "value": {
              "stringValue": "Mixer"
          }
      },
      "telemetryId": {
          "definition": {
```

```
                "dataType": { "type": "STRING" },
                "isExternalId": true,
                "isFinal": true,
                "isImported": false,
                "isInherited": false,
                "isRequiredInEntity": true,
                "isStoredExternally": false,
                "isTimeSeries": false
            },
            "value": {
                "stringValue": "item_A001"
            }
        },
        "Temperature": {
            "definition": {
                "dataType": { "type": "DOUBLE", },
                "isExternalId": false,
                "isFinal": false,
                "isImported": true,
                "isInherited": false,
                "isRequiredInEntity": false,
                "isStoredExternally": false,
                "isTimeSeries": true
            }
        }
    }
}
```

リクエストの主要な要素：

- selectedPropertiesは、Timestream計測の対象となるプロパティを入力するリストです。
- startDateTime、startTime、EndDateTime、endTimeの各フィールドでは、リクエストの時間範囲を指定します。これにより、返される測定値のサンプル範囲が決まります。
- entityIdは、データのクエリ元となるエンティティの名前です。
- componentNameは、データのクエリ元となるコンポーネントの名前です。
- orderByTimeフィールドを使用して、結果が表示される順序を整理します。

前述のリクエスト例では、特定のアイテムの特定の時間枠内に、選択したプロパティの一連のサンプルを、選択した時間順序で取得することを想定しています。レスポンスステートメントは、以下のように要約できます。

```
{
  "propertyValues": [
    {
      "entityPropertyReference": {
        "entityId": "MyEntity",
        "componentName": "TelemetryData",
        "propertyName": "Temperature"
      },
      "values": [
        {
          "time": "2022-08-25T00:00:00Z",
          "value": {
            "doubleValue": 588.168
          }
        },
        {
          "time": "2022-08-25T00:00:01Z",
          "value": {
            "doubleValue": 592.4224
          }
        },
        {
          "time": "2022-08-25T00:00:02Z",
          "value": {
            "doubleValue": 594.9383
          }
        }
      ]
    }
  ],
  "nextToken": "..."
}
```

4. コンポーネントタイプを2つのプロパティを持つように更新

   次のJSONテンプレートは、2つのプロパティを持つ有効なコンポーネントタイプを示しています。

```
{
    "componentTypeId": "com.example.timestream-telemetry",
    "workspaceId": "MyWorkspace",
    "functions": {
        "dataReader": {
            "implementedBy": {
                "lambda": {
                    "arn": "lambdaArn"
                }
            }
        }
    },
    "propertyDefinitions": {
        "telemetryType": {
            "dataType": { "type": "STRING" },
            "isExternalId": false,
            "isStoredExternally": false,
            "isTimeSeries": false,
            "isRequiredInEntity": true
        },
        "telemetryId": {
            "dataType": { "type": "STRING" },
            "isExternalId": true,
            "isStoredExternally": false,
            "isTimeSeries": false,
            "isRequiredInEntity": true
        },
        "Temperature": {
            "dataType": { "type": "DOUBLE" },
            "isExternalId": false,
            "isTimeSeries": true,
            "isStoredExternally": true,
            "isRequiredInEntity": false
        },
        "RPM": {
            "dataType": { "type": "DOUBLE" },
            "isExternalId": false,
            "isTimeSeries": true,
            "isStoredExternally": true,
            "isRequiredInEntity": false
        }
    }
```

```
}
```

5. 2番目のプロパティを処理するようにLambdaコネクタを更新

   AWS IoT TwinMaker データプレーン API は、1 回のリクエストで複数のプロパティをクエリすることをサポートし、コネクタへの 1 AWS IoT TwinMaker 回のリクエストに続いて次のリストを提供します。`selectedProperties`

   次のJSONリクエストは、2つのプロパティのリクエストをサポートするようになった変更後のテンプレートを示しています。

```
{
  "workspaceId": "MyWorkspace",
  "entityId": "MyEntity",
  "componentName": "TelemetryData",
  "selectedProperties": ["Temperature", "RPM"],
  "startTime": "2022-08-25T00:00:00Z",
  "endTime": "2022-08-25T00:00:05Z",
  "maxResults": 3,
  "orderByTime": "ASCENDING",
  "properties": {
      "telemetryType": {
          "definition": {
              "dataType": { "type": "STRING" },
              "isExternalId": false,
              "isFinal": false,
              "isImported": false,
              "isInherited": false,
              "isRequiredInEntity": false,
              "isStoredExternally": false,
              "isTimeSeries": false
          },
          "value": {
              "stringValue": "Mixer"
          }
      },
      "telemetryId": {
          "definition": {
              "dataType": { "type": "STRING" },
              "isExternalId": true,
              "isFinal": true,
              "isImported": false,
              "isInherited": false,
```

```
                "isRequiredInEntity": true,
                "isStoredExternally": false,
                "isTimeSeries": false
            },
            "value": {
                "stringValue": "item_A001"
            }
        },
        "Temperature": {
            "definition": {
                "dataType": { "type": "DOUBLE" },
                "isExternalId": false,
                "isFinal": false,
                "isImported": true,
                "isInherited": false,
                "isRequiredInEntity": false,
                "isStoredExternally": false,
                "isTimeSeries": true
            }
        },
        "RPM": {
            "definition": {
                "dataType": { "type": "DOUBLE" },
                "isExternalId": false,
                "isFinal": false,
                "isImported": true,
                "isInherited": false,
                "isRequiredInEntity": false,
                "isStoredExternally": false,
                "isTimeSeries": true
            }
        }
    }
}
```

同様に、次の例に示すように、対応するレスポンスも更新されます。

```
{
  "propertyValues": [
    {
      "entityPropertyReference": {
        "entityId": "MyEntity",
        "componentName": "TelemetryData",
```

```
        "propertyName": "Temperature"
      },
      "values": [
        {
          "time": "2022-08-25T00:00:00Z",
          "value": {
            "doubleValue": 588.168
          }
        },
        {
          "time": "2022-08-25T00:00:01Z",
          "value": {
            "doubleValue": 592.4224
          }
        },
        {
          "time": "2022-08-25T00:00:02Z",
          "value": {
            "doubleValue": 594.9383
          }
        }
      ]
    },
    {
      "entityPropertyReference": {
        "entityId": "MyEntity",
        "componentName": "TelemetryData",
        "propertyName": "RPM"
      },
      "values": [
        {
          "time": "2022-08-25T00:00:00Z",
          "value": {
            "doubleValue": 59
          }
        },
        {
          "time": "2022-08-25T00:00:01Z",
          "value": {
            "doubleValue": 60
          }
        },
        {
          "time": "2022-08-25T00:00:02Z",
```

```
          "value": {
            "doubleValue": 60
          }
        }
      ]
    }
  ],
  "nextToken": "..."
}
```

> **Note**
>
> この場合のページ分割に関しては、リクエスト内のページサイズはすべてのプロパティに適用されます。つまり、クエリのプロパティが5つで、ページサイズが100の場合、ソースに十分なデータポイントがあれば、プロパティごとに100データポイント、合計500データポイントが表示されるはずです。

実装例については、「Snowflake コネクタのサンプル」を参照してください。 GitHub

データコネクタの改善

例外処理

Lambdaコネクタが例外をスローしても安全です。データプレーン API 呼び出しでは、 AWS IoT TwinMaker サービスは Lambda 関数が応答を返すのを待ちます。コネクタ実装が例外を投げると、AWS IoT TwinMaker 例外タイプをに変換してConnectorFailure、コネクタ内で問題が発生したことを API クライアントに認識させます。

ページネーションの処理

この例では、Timestreamはページネーションをネイティブにサポートするユーティリティ関数を提供しています。ただし、SQLなどの他のクエリインターフェースでは、効率的なページネーションアルゴリズムを実装するために余分な労力が必要になる場合があります。SQLインターフェースでページ分割を処理するSnowflakeコネクタの例があります。

AWS IoT TwinMaker コネクタのレスポンスインターフェースから新しいトークンが返されると、トークンは暗号化されてから API クライアントに返されます。トークンが別のリクエストに含まれ

ている場合は、Lambda AWS IoT TwinMaker コネクタに転送する前に復号化します。トークンに機密情報を追加しないことをお勧めします。

コネクタのテスト

コネクタをコンポーネントタイプにリンクした後でも実装を更新することはできますが、 AWS IoT TwinMakerとインテグレートする前にLambdaコネクタを検証することを強くお勧めします。

Lambdaコネクタをテストするには複数の方法があります：LambdaコンソールでLambdaコネクタをテストすることも、 AWS CDKでローカルにテストすることもできます。

Lambda 関数のテストの詳細については、「Lambda 関数のテスト」と「アプリケーションのローカルテスト」を参照してください。 AWS CDK

セキュリティ

Timestreamのセキュリティベストプラクティスに関するドキュメントについては、「Timestreamのセキュリティ」を参照してください。

SQL インジェクション防止の例については、 AWS IoT TwinMaker GitHub サンプルリポジトリの次の Python スクリプトを参照してください。

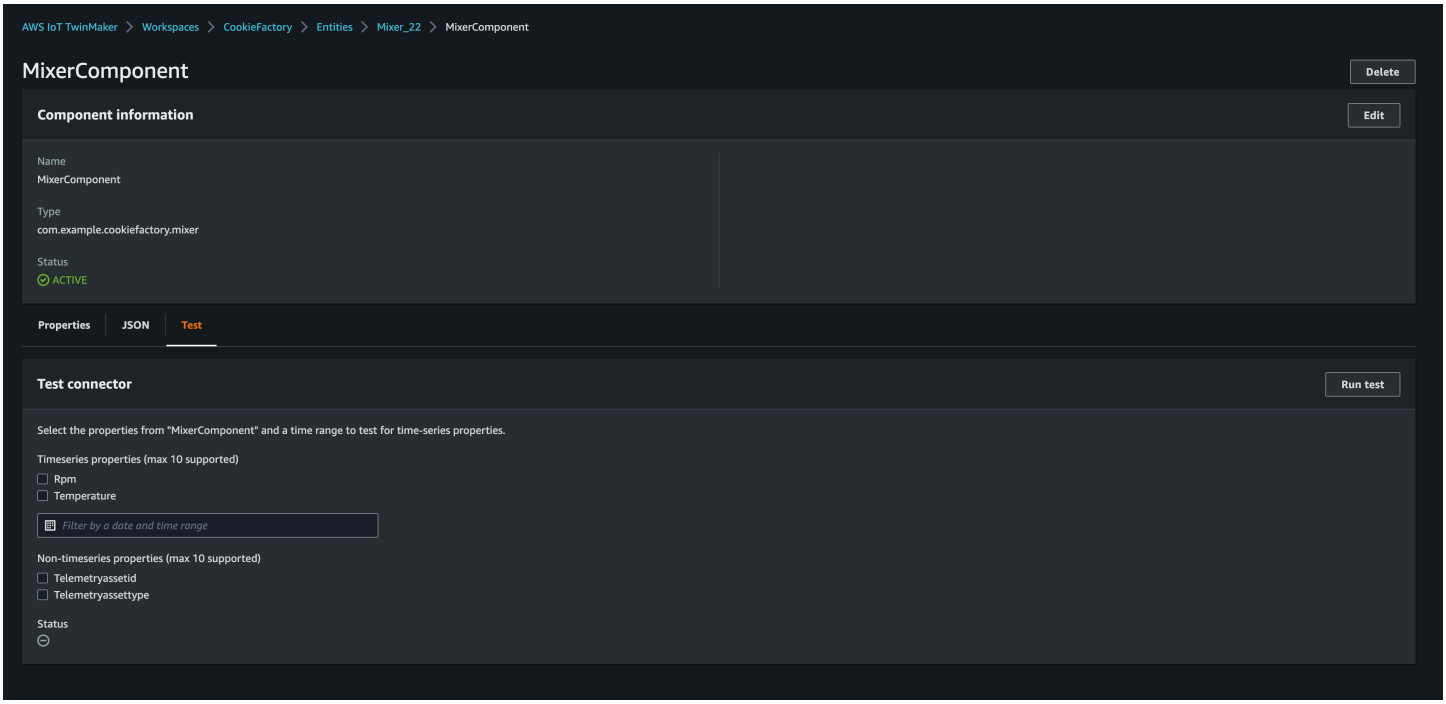
AWS IoT TwinMaker リソースの作成

Lambda 関数を実装したら、AWS IoT TwinMaker コンソールまたは API を使用してコンポーネントタイプ、エンティティ、 AWS IoT TwinMaker コンポーネントなどのリソースを作成できます。

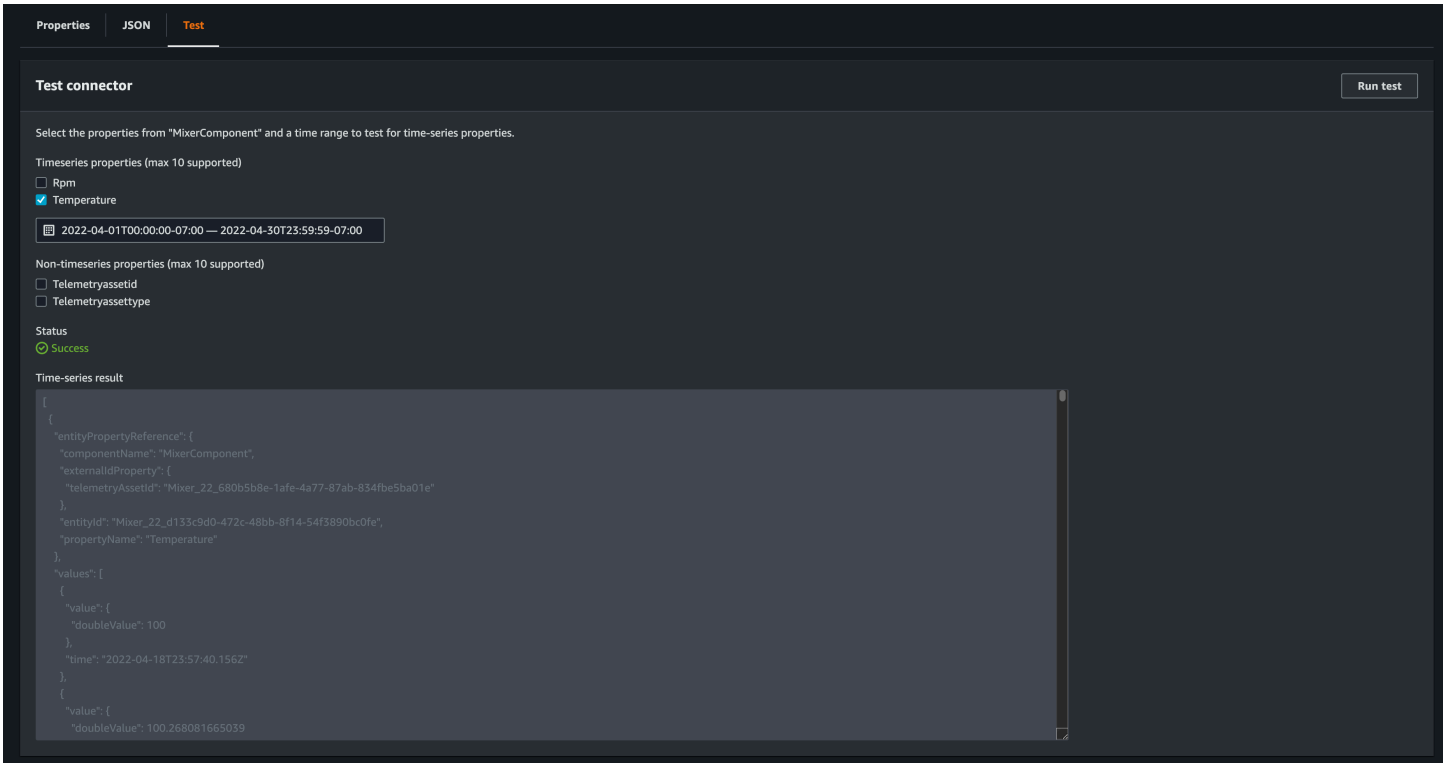
> **Note**
>
> GitHub サンプルの設定手順に従うと、 AWS IoT TwinMaker すべてのリソースが自動的に使用可能になります。AWS IoT TwinMaker GitHub サンプル内のコンポーネントタイプ定義を確認できます。コンポーネントタイプがコンポーネントによって一度使用されると、そのコンポーネントタイプのプロパティ定義と関数は更新できません。

インテグレーションテスト

AWS IoT TwinMaker との統合テストを実施して、データプレーンクエリが機能することを確認することをお勧めします end-to-end。GetPropertyValueHistoryAPI を使用して実行することも、AWS IoT TwinMaker コンソールで簡単に実行することもできます。



AWS IoT TwinMaker コンソールで [コンポーネントの詳細] に移動し、[テスト] の下にコンポーネントのすべてのプロパティが一覧表示されているのがわかります。コンソールのテストエリアでは、non-time-series プロパティだけでなく時系列プロパティもテストできます。時系列プロパティには API を使用し、 non-time-series プロパティには GetPropertyValueHistoryAPI GetPropertyValueを使用することもできます。Lambdaコネクタが複数のプロパティクエリをサポートしている場合、複数のプロパティを選択できます。



次のステップ

AWS IoT TwinMaker Grafanaダッシュボードを設定してメトリクスを視覚化できるようになりました。また、AWS IoT TwinMaker GitHub サンプルリポジトリにある他のデータコネクタサンプルを調べて、ユースケースに合っているかどうかを確認することもできます。

AWS IoT TwinMakerクッキー ファクトリ時系列コネクタの例

クッキーファクトリの Lambda 関数の完全なコードは、で入手できます。 GitHubコネクタをコンポーネントタイプにリンクした後でも実装を更新することはできますが、 AWS IoT TwinMakerとインテグレートする前にLambdaコネクタを検証することを強くお勧めします。Lambda関数のテストは、Lambdaコンソールで行うか、 AWS CDKのローカルで行います。Lambda 関数のテストの詳細については、「Lambda 関数のテスト」と「アプリケーションのローカルテスト」を参照してください。 AWS CDK

クッキーファクトリのコンポーネントタイプの例

コンポーネントタイプでは、コンポーネント間で共有される共通のプロパティを定義します。クッキーファクトリの例では、同じタイプの物理コンポーネントが同じ測定値を共有するため、コンポーネントタイプで測定スキーマを定義できます。一例として、以下の例ではミキサータイプを定義しています。

```
{
    "componentTypeId": "com.example.cookiefactory.mixer"
    "propertyDefinitions": {
        "RPM": {
            "dataType": { "type": "DOUBLE" },
            "isTimeSeries": true,
            "isRequiredInEntity": false,
            "isExternalId": false,
            "isStoredExternally": true
        },
        "Temperature": {
            "dataType": { "type": "DOUBLE" },
            "isTimeSeries": true,
            "isRequiredInEntity": false,
            "isExternalId": false,
            "isStoredExternally": true
        }
    }
}
```

たとえば、物理コンポーネントの Timestream データベースには測定値、SQL データベースにはメンテナンスレコード、アラームシステムにはアラームデータがある場合があります。複数のコンポーネントを作成してエンティティに関連付けると、さまざまなデータソースがエンティティにリンクされ、エンティティコンポーネントグラフにデータが入力されます。この場合、各コンポーネントには、telemetryId対応するデータソース内のコンポーネントの固有キーを識別するプロパティが必要です。telemetryIdプロパティを指定することには 2 つの利点があります。1 つは、プロパティをデータコネクターでフィルター条件として使用して、特定のコンポーネントの値のみをクエリできること、もう 1 つは、データプレーン API telemetryId レスポンスにプロパティ値を含めると、クライアント側が ID を取得し、必要に応じて逆引き検索を実行できることです。

TelemetryIdをコンポーネントタイプに外部 ID として追加すると、TimeStreamテーブル内のコンポーネントが識別されます。

```
{
    "componentTypeId": "com.example.cookiefactory.mixer"
    "propertyDefinitions": {
        "telemetryId": {
            "dataType": { "type": "STRING" },
            "isTimeSeries": false,
            "isRequiredInEntity": true,
            "isExternalId": true,
```

```
            "isStoredExternally": false
        },
        "RPM": {
            "dataType": { "type": "DOUBLE" },
            "isTimeSeries": true,
            "isRequiredInEntity": false,
            "isExternalId": false,
            "isStoredExternally": true
        },
        "Temperature": {
            "dataType": { "type": "DOUBLE" },
            "isTimeSeries": true,
            "isRequiredInEntity": false,
            "isExternalId": false,
            "isStoredExternally": true
        }
    }
}
```

同様に、以下のJSONの例に示すように、WaterTankのコンポーネントタイプがあります。

```
{
  "componentTypeId": "com.example.cookiefactory.watertank",
  "propertyDefinitions": {
    "flowRate1": {
      "dataType": { "type": "DOUBLE" },
      "isTimeSeries": true,
      "isRequiredInEntity": false,
      "isExternalId": false,
      "isStoredExternally": true
    },
    "flowrate2": {
      "dataType": { "type": "DOUBLE" },
      "isTimeSeries": true,
      "isRequiredInEntity": false,
      "isExternalId": false,
      "isStoredExternally": true
    },
    "tankVolume1": {
      "dataType": { "type": "DOUBLE" },
      "isTimeSeries": true,
      "isRequiredInEntity": false,
      "isExternalId": false,
```

```
      "isStoredExternally": true
    },
    "tankVolume2": {
      "dataType": { "type": "DOUBLE" },
      "isTimeSeries": true,
      "isRequiredInEntity": false,
      "isExternalId": false,
      "isStoredExternally": true
    },
    "telemetryId": {
      "dataType": { "type": "STRING" },
      "isTimeSeries": false,
      "isRequiredInEntity": true,
      "isExternalId": true,
      "isStoredExternally": false
    }
  }
}
```

エンティティスコープ内のプロパティ値のクエリを目的としている場合、TelemetryTypeはコンポーネントタイプのオプションプロパティです。例については、AWS IoT TwinMaker GitHub サンプルリポジトリ内の定義済みコンポーネントタイプを参照してください。同じテーブルにはアラームタイプも埋め込まれているので、TelemetryTypeが定義され、TelemetryIdやTelemetryTypeなどの共通プロパティを親コンポーネントタイプに抽出して、他の子タイプと共有できます。

Lambdaの例

Lambdaコネクタは、データソースにアクセスし、入力に基づいてクエリステートメントを生成し、それをデータソースに転送する必要があります。Lambdaに送信されるリクエスト例を、次のJSON例で示すことができます。

```
{
    'workspaceId': 'CookieFactory',
    'selectedProperties': ['Temperature'],
    'startDateTime': 1648796400,
    'startTime': '2022-04-01T07:00:00.000Z',
    'endDateTime': 1650610799,
    'endTime': '2022-04-22T06:59:59.000Z',
    'properties': {
        'telemetryId': {
            'definition': {
                'dataType': { 'type': 'STRING' },
```

```
                'isTimeSeries': False,
                'isRequiredInEntity': True,
                'isExternalId': True,
                'isStoredExternally': False,
                'isImported': False,
                'isFinal': False,
                'isInherited': True,
            },
            'value': {
                'stringValue': 'Mixer_22_680b5b8e-1afe-4a77-87ab-834fbe5ba01e'
            }
        }
        'Temperature': {
            'definition': {
                'dataType': { 'type': 'DOUBLE' },
                'isTimeSeries': True,
                'isRequiredInEntity': False,
                'isExternalId': False,
                'isStoredExternally': True,
                'isImported': False,
                'isFinal': False,
                'isInherited': False
            }
        }
        'RPM': {
            'definition': {
                'dataType': { 'type': 'DOUBLE' },
                'isTimeSeries': True,
                'isRequiredInEntity': False,
                'isExternalId': False,
                'isStoredExternally': True,
                'isImported': False,
                'isFinal':False,
                'isInherited': False
            }
        },
    'entityId': 'Mixer_22_d133c9d0-472c-48bb-8f14-54f3890bc0fe',
    'componentName': 'MixerComponent',
    'maxResults': 100,
    'orderByTime': 'ASCENDING'
}
```

Lambda 関数の目的は、特定のエンティティの過去の測定データをクエリすることです。 AWS IoT TwinMaker コンポーネントとプロパティのマップが提供されるため、コンポーネント ID にはインスタンス化された値を指定する必要があります。たとえば、コンポーネントタイプレベルのクエリ (アラームのユースケースによくある) を処理し、ワークスペース内のすべてのコンポーネントのアラームステータスを返すには、プロパティマップにコンポーネントタイプのプロパティ定義があります。

最も単純なケースでは、前述のリクエストのように、特定のコンポーネントの特定の時間枠における一連の温度サンプルを、時間の昇順で取得する必要があります。このクエリステートメントは、以下のように要約できます。

```
...
SELECT measure_name, time, measure_value::double
    FROM {database_name}.{table_name}
    WHERE time < from_iso8601_timestamp('{request.start_time}')
    AND time >= from_iso8601_timestamp('{request.end_time}')
    AND TelemetryId = '{telemetry_id}'
    AND measure_name = '{selected_property}'
    ORDER BY time {request.orderByTime}
...
```

AWS IoT TwinMaker シーンの作成と編集

シーンはデジタルツインを 3 次元で視覚化したものです。これらはデジタルツインを編集する主な方法です。アラーム、時系列データ、カラーオーバーレイ、タグ、ビジュアルルールをシーンに追加して、デジタルツインを視覚化する方法を実際のユースケースとともに説明します。

このセクションでは、次のトピックについて説明します。

最初のシーンを作成する前に

シーンはデジタルツインを表現するリソースに依存しています。これらのリソースは 3D モデル、データ、またはテクスチャファイルで構成されています。リソースのサイズと複雑さ、シーン内の要素 (照明など)、コンピューターハードウェアは、 AWS IoT TwinMaker シーンのパフォーマンスに影響します。このトピックの情報を活用して、ラグや読み込み時間を短縮し、シーンのフレームレートを向上できます。

リソースを にインポートする前に最適化する AWS IoT TwinMaker

を使用して AWS IoT TwinMaker 、デジタルツインをリアルタイムで操作できます。シーンを最大限に活用するには、リソースをリアルタイム環境での使用で最適化することをお勧めします。

3D モデルはパフォーマンスに大きな影響を与える可能性があります。複雑なモデルジオメトリやメッシュはパフォーマンスを低下させる可能性があります。例えば、工業用 CAD モデルは詳細度が高いです。 AWS IoT TwinMaker シーンで使用する前に、これらのモデルのメッシュを圧縮し、ポリゴン数を減らすことをお勧めします。用の新しい 3D モデルを作成する場合は AWS IoT

TwinMaker、詳細レベルを確立し、すべてのモデルにわたって維持する必要があります。ユースケースの視覚化や解釈に影響を与えない詳細をモデルから削除します。

モデルを圧縮してファイルサイズを小さくするには、DRACO 3D データ圧縮などのオープンソースのメッシュ圧縮ツールを使用します。

最適化されていないテクスチャもパフォーマンスに影響する可能性があります。テクスチャに透明度が必要ない場合は、PNG 形式よりも PEG 画像形式を選択することを検討してください。ベーシス・ユニバーサル テクスチャ圧縮などのオープンソースのテクスチャ圧縮ツールを使用して、テクスチャファイルを圧縮できます。

AWS IoT TwinMakerでのパフォーマンスのベストプラクティス

で最高のパフォーマンスを得るには AWS IoT TwinMaker、以下の制限とベストプラクティスに注意してください。

- AWS IoT TwinMaker シーンレンダリングのパフォーマンスはハードウェアによって異なります。パフォーマンスはコンピューターのハードウェア構成によって異なります。
- AWS IoT TwinMaker内のすべてのオブジェクトのポリゴンの総数を 100 万未満にすることをお勧めします。
- シーンごとに合計で 200 のオブジェクトを作成することをお勧めします。シーン内のオブジェクト数を 200 以上に増やすと、シーンのフレームレートが下がる可能性があります。
- シーン内のすべての一意の 3D アセットの合計サイズは 100 MB を超えないようにすることをお勧めします。そうでない場合は、ブラウザやハードウェアによっては、読み込み時間が遅くなったり、パフォーマンスが低下したりする可能性があります。
- シーンにはデフォルトでアンビエント照明があります。シーンにライトを追加して特定のオブジェクトにピントを合わせたり、オブジェクトに影を落としたりすることができます。シーンごとに 1 つのライトを使用することをお勧めします。必要に応じてライトを使用し、シーン内で現実世界のライトを複製することは避けてください。

詳細はこちら

シーンのパフォーマンスを向上させるために使用できる最適化テクニックの詳細についてはリソースを活用してください。

- で使用するために OBJ モデルを F に変換して圧縮する方法 AWS IoT TwinMaker
- 3D モデルをウェブコンテンツ用に最適化する

- WebGL のパフォーマンスを向上させるためのシーンの最適化

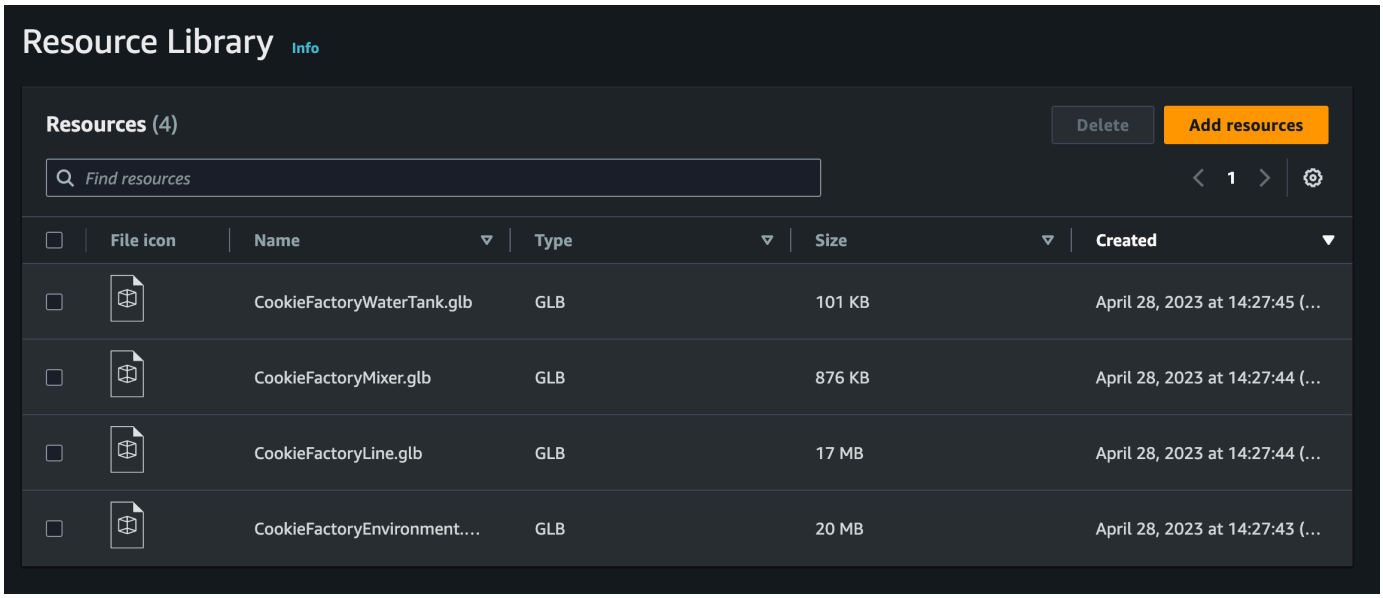
リソースライブラリへの AWS IoT TwinMaker リソースのアップロード

リソースライブラリを使用して、デジタルツインアプリケーションのシーンに配置したいリソースを制御および管理できます。リソース AWS IoT TwinMaker を認識するには、リソースライブラリコンソールページを使用してリソースをアップロードします。

コンソールを使用して Resource Library にファイルをアップロードする

AWS IoT TwinMaker コンソールを使用して Resource Library にファイルを追加するには、次の手順に従います。

1. 左側のナビゲーションメニューの Workspace で、リソースライブラリ を選択します。
2. リソースを追加 を選択し、アップロードするファイルを選択します。



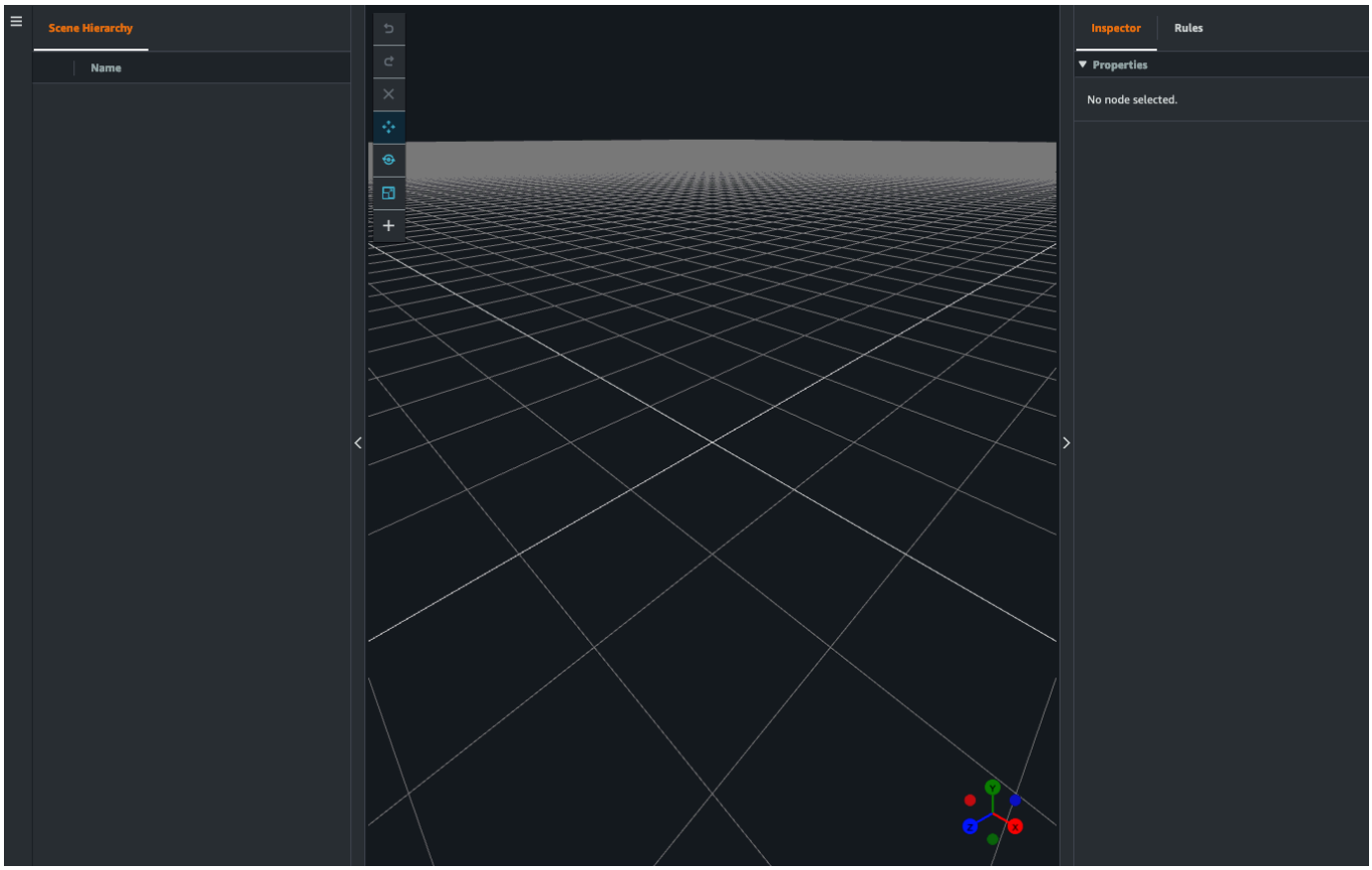
シーンを作成する

このセクションでは、デジタルツインを編集できるようにシーンを設定します。リソースライブラリにアップロードされた 3D モデルをインポートし、ウィジェットを追加し、プロパティデータをオブジェクトにバインドしてデジタルツインを完了できます。シーンオブジェクトには、建物全体やスペース、または物理的な場所に配置された個々の機器を含めることができます。

> **Note**
>
> シーンを作成する前に、ワークスペースを作成する必要があります。

でシーンを作成するには、次の手順を使用します AWS IoT TwinMaker。

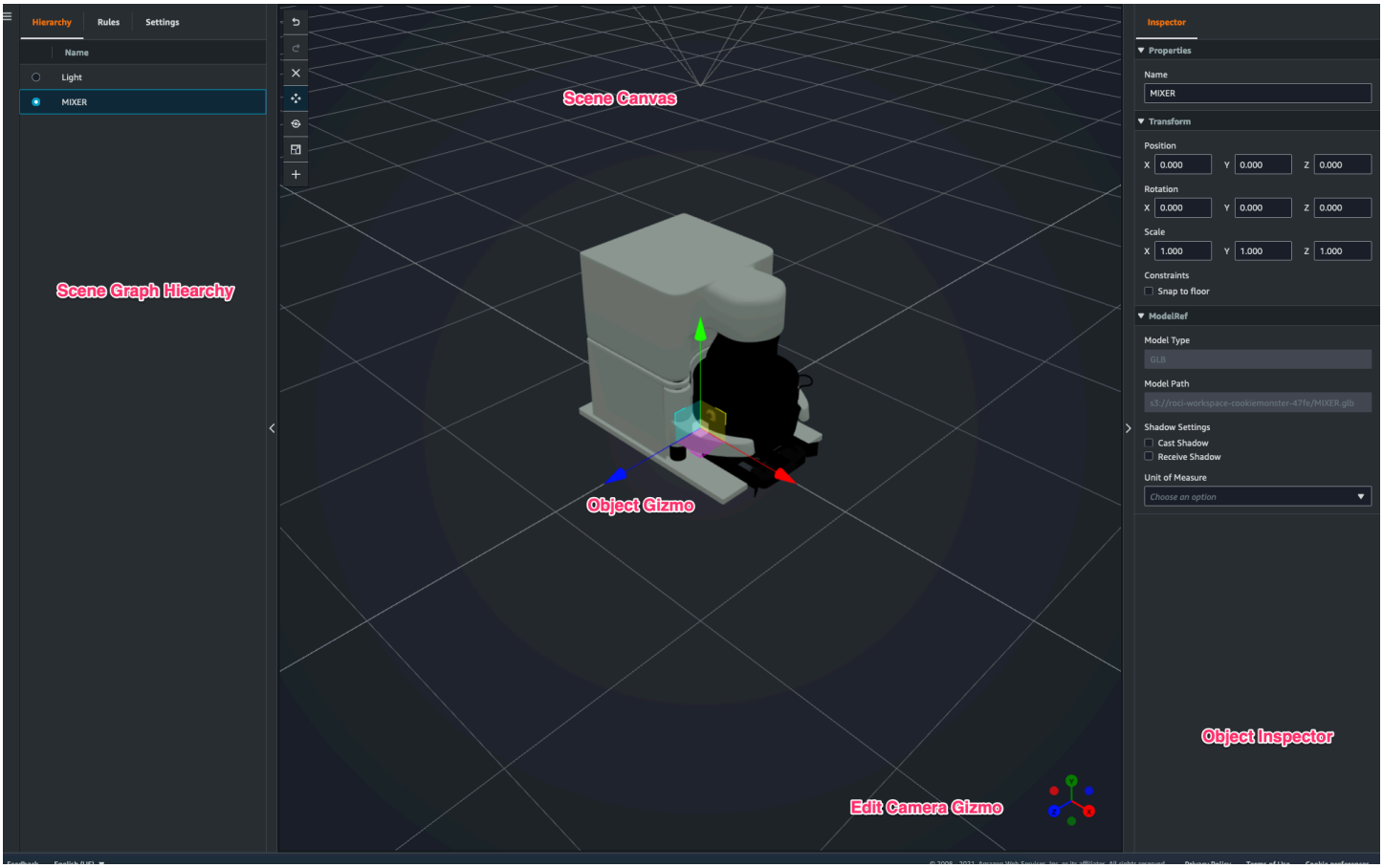
1. シーンペインを開くには、ワークスペースの左側のナビゲーションでシーン を選択します。
2. 「シーンの作成」 を選択します。新しいシーン作成ペインが開きます。
3. 「シーン作成」 ペインに新しいシーンの名前と説明を入力します。標準または階層バンドルの料金プランがある場合は、シーンタイプを選択できます。動的シーン を使用することをお勧めします。
4. シーンを作成する準備ができたら、「シーンの作成」 を選択します。新しいシーンが開き、作業できる状態になります。



シーンで 3D AWS IoT TwinMaker ナビゲーションを使用する

AWS IoT TwinMaker シーンには、シーンの 3D スペースを効率的にナビゲートするために使用できるナビゲーションコントロールのセットがあります。3D スペースやシーンで表されるオブジェクトを操作するには、次のウィジェットとメニューオプションを使用します。

- インスペクター:「インスペクター」 ウィンドウを使用して、階層内の選択したエンティティまたはコンポーネントのプロパティと設定を表示および編集します。
- シーンキャンバス: シーンキャンバスは、使用したい任意の 3D リソースを配置したり向きを変えたりできる 3D スペースです。
- シーングラフ階層: このパネルを使用して、シーンに存在するすべてのエンティティを表示できます。ウィンドウの左側に表示されます。
- オブジェクトギズモ: このギズモを使用して、キャンバス上でオブジェクトを移動します。シーンキャンバスで選択した 3D オブジェクトの中央に表示されます。
- カメラ編集ギズモ: カメラ編集ギズモを使用すると、シーンビューカメラの現在の向きをすばやく確認したり、表示角度を変更したりできます。このギズモはシーンビューの右下隅にあります。
- ズームコントロール: シーンキャンバス上を移動するには、右クリックして移動したい方向にドラッグします。回転するには、左クリックしてドラッグして回転します。ズームするには、マウスのスクロールホイールを使用するか、ラップトップのトラックパッドで指をつまんで離します。



階層ペインのシーンボタンには、ボタンのレイアウト順に次の機能が表示されます。

- 元に戻す: シーン内の最後の変更を取り消します。
- やり直し: シーン内の最後の変更をやり直します。
- プラス (+): このボタンを使用すると、「空のノードを追加」、「3D モデルを追加」、「タグを追加」、「ライトを追加」、「モデルシェーダを追加」の各アクションにアクセスできます。
- ナビゲーション方法の変更: シーンカメラのナビゲーションオプションである 「オービット」 と 「パン」 にアクセスできます。
- ゴミ箱 (削除): このボタンを使用すると、シーン内の選択したオブジェクトを削除できます。
- オブジェクト操作ツール: このボタンを使用すると、選択したオブジェクトを移動、回転、スケールできます。

固定カメラをエンティティに追加する

固定カメラビューを AWS IoT TwinMaker シーン内のエンティティにアタッチできます。これらのカメラは 3D モデルに固定遠近感を与えるため、シーン内の視点を目的のエンティティにすばやく簡単に移動できます。

1. 「AWS IoT TwinMaker 」 コンソールでシーンに移動します。
2. シーン階層メニューで、カメラをアタッチするエンティティを選択します。
3. 「+」 ボタンを押し、ドロップダウンオプションから 「現在のビューからカメラを追加」 を選択します。現在の視点カメラをエンティティに適用するには。
4. インスペクターでカメラを設定し、次の設定を調整できます。
   - カメラ名
   - カメラの位置と回転
   - カメラの焦点距離
   - ズームレベル
   - ニアクリッピングプレーンとファークリッピングプレーン
5. カメラを配置した後でカメラにアクセスするには。カメラを追加したエンティティを階層内で選択します。エンティティの下に表示されているカメラ名を探します。
6. エンティティから配置されたカメラを選択すると、シーンのカメラビューは配置されたカメラの設定されたパースペクティブにスナップされます。

シーン拡張編集

AWS IoT TwinMaker シーンには、シーンに存在するリソースを強化、編集、操作するための一連のツールが用意されています。

以下のトピックでは、 AWS IoT TwinMaker シーンで拡張編集機能を使用する方法について説明します。

- シーンオブジェクトのターゲットを絞った配置
- サブモデル選択
- シーン階層内のエンティティ編集

シーンオブジェクトのターゲットを絞った配置

AWS IoT TwinMaker では、シーンにオブジェクトを正確に配置して追加できます。この強化編集機能により、シーン内のタグ、エンティティ、ライト、モデルを配置する場所をより細かく制御できます。

1. 「AWS IoT TwinMaker 」 コンソールでシーンに移動します。
2. 「+」 ボタンを押し、ドロップダウンオプションからオプションの 1 つを選択します。モデル、ライト、タグなど、「+」 メニューにあるものなら何でもかまいません。

   シーンの 3D スペースでカーソルを動かすと、カーソルの周りにターゲットが表示されます。
3. ターゲットを使用して、シーンに要素を正確に配置します。

サブモデル選択

AWS IoT TwinMaker では、シーン内の 3D モデルのサブモデルを選択し、タグ、ライト、ルールなどの標準プロパティを適用できます。

3D モデルファイル形式には、モデルのサブエリアを大きなモデル内のサブモデルとして指定できるメタデータが含まれています。たとえば、モデルがろ過システムの場合、タンク、パイプ、モーターなどのシステムの個々の部分はろ過の 3D モデルのサブモデルとしてマークされます。

シーンでサポートされている 3D ファイル形式: GLB と GLTF。

1. 「AWS IoT TwinMaker 」 コンソールでシーンに移動します。
2. シーンにモデルがない場合は、「+」 メニューからオプションを選択してモデルを追加します。
3. シーン階層にリストされているモデルを選択します。選択すると、階層にはモデルの下にサブモデルが表示されます。

   > **Note**
   >
   > サブモデルが表示されない場合は、そのモデルにサブモデルが設定されていない可能性があります。
4. サブモデルの表示を切り替えるには、階層内のサブモデルの名前の右側にある目のアイコンを押します。

5. 名前や位置などのサブモデルデータを編集するには、サブモデルを選択してシーンインスペクターを開きます。インスペクターメニューを使用して、サブモデルデータを更新または変更します。
6. タグ、ライト、ルール、その他のプロパティをサブモデルに追加するには、階層内でサブモデルを選択した状態で 「+」 を押します。

シーン階層内のエンティティ編集

AWS IoT TwinMaker シーンを使用すると、階層テーブル内のエンティティのプロパティを直接編集できます。次の手順は、階層メニューからエンティティに対して実行できるアクションを示しています。

1. 「AWS IoT TwinMaker 」 コンソールでシーンに移動します。
2. シーン階層を開き、操作するエンティティのサブ要素を選択します。
3. 要素を選択したら、「+」 ボタンを押し、ドロップダウンから次のいずれかのオプションを選択します。
   - 空のノードを追加
   - 3D モデルを追加
   - ライトを追加
   - 現在の視点からカメラを追加
   - タグを追加
   - モデルシェーダーを追加
   - モーションインジケータを追加
4. ドロップダウンからいずれかのオプションを選択すると、その選択が手順 2 で選択した要素の子としてシーンに適用されます。
5. 子要素を選択し、階層内を新しい親にドラッグすることで、子要素の順序を変更したり、要素を再ペアレント化したりできます。

エンティティに注釈を追加します。

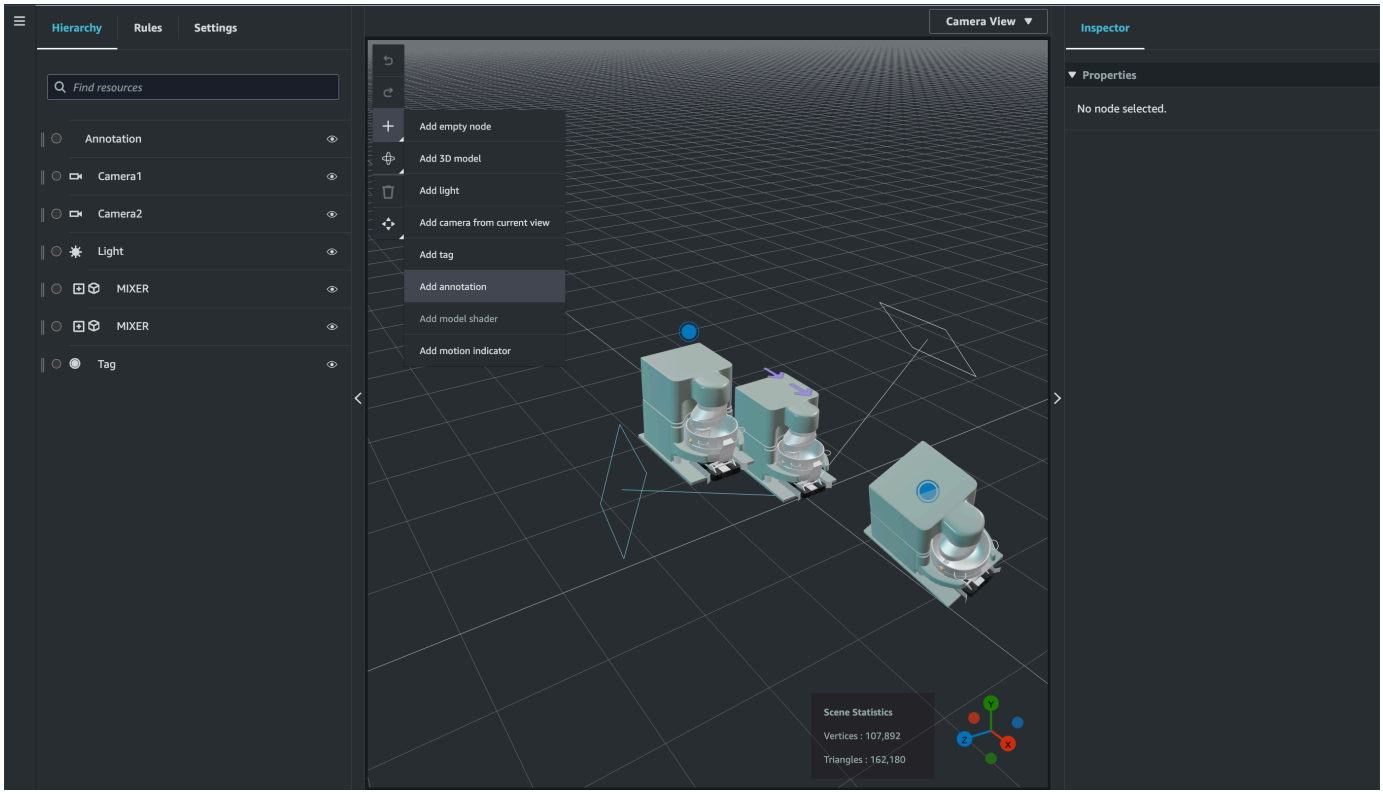
AWS IoT TwinMaker シーンコンポーザーを使用すると、シーン階層内の任意の要素に注釈を付けることができます。注釈はマークダウンで作成されます。

マークダウンでの記述の詳細については、マークダウン構文に関する公式ドキュメントの「基本構文」を参照してください。

> **Note**
>
> AWS IoT TwinMaker 注釈とオーバーレイの Markdown 構文のみ。HTML ではありません。

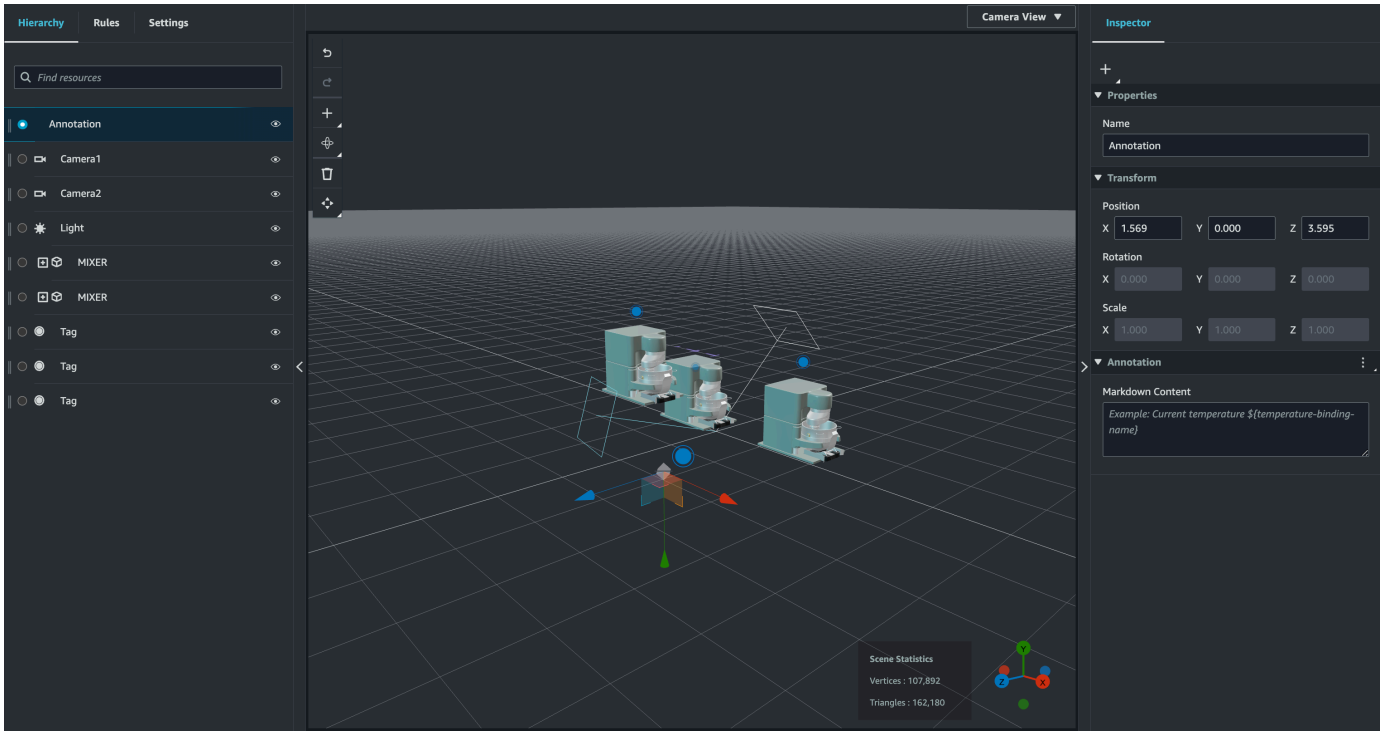
エンティティに注釈を追加

1. 「AWS IoT TwinMaker 」 コンソールでシーンに移動します。
2. シーン階層から注釈を付ける要素を選択します。階層内の要素が選択されていない場合は、ルートに注釈を追加できます。
3. 「+」 ボタンを押して、「注釈を追加」オプションを選択します。

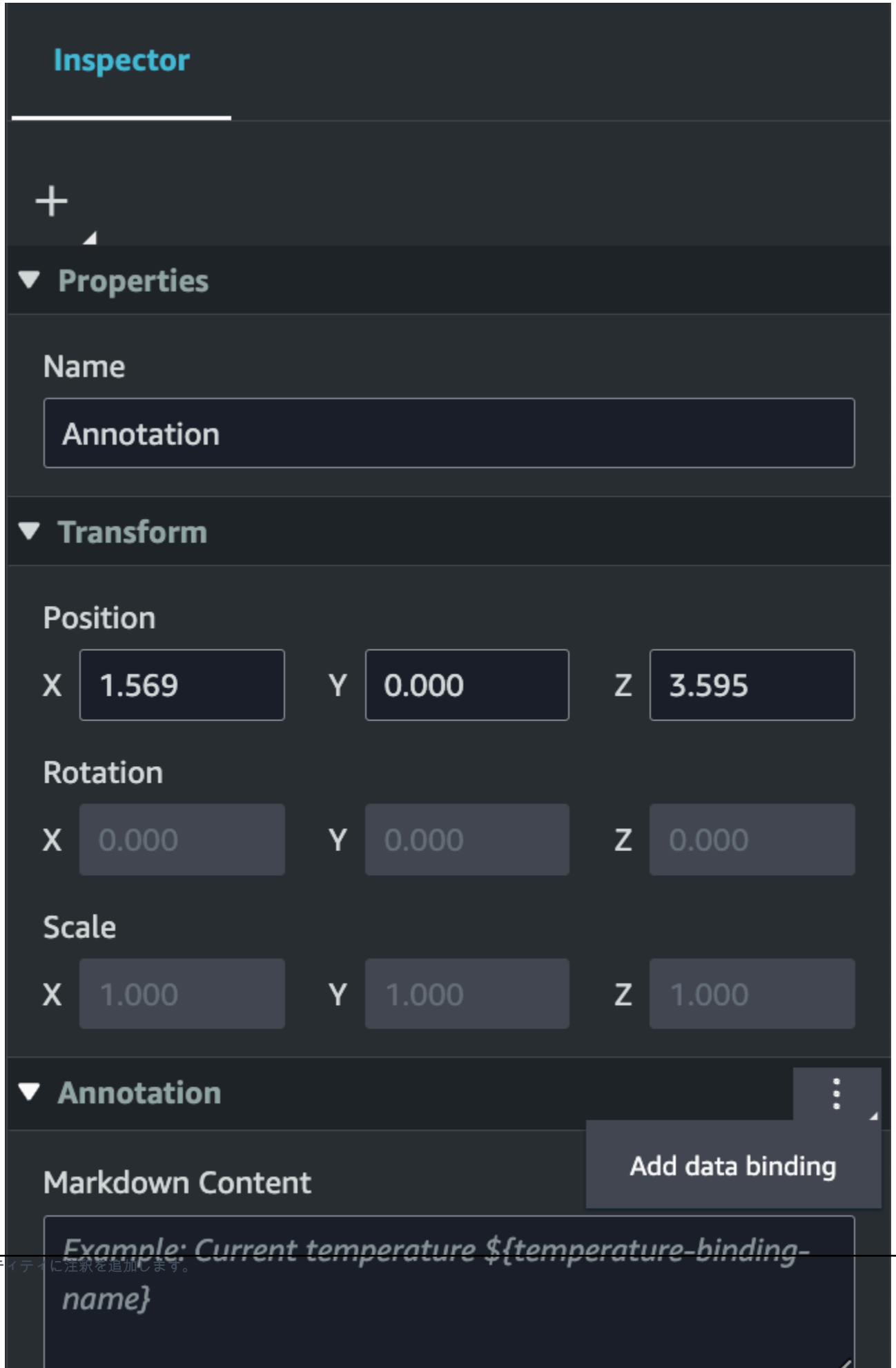


4. 左側の 「インスペクター」 ウィンドウで、「注釈」 セクションまでスクロールします。マークダウン構文を使用して、注釈に表示させるテキストを記述します。

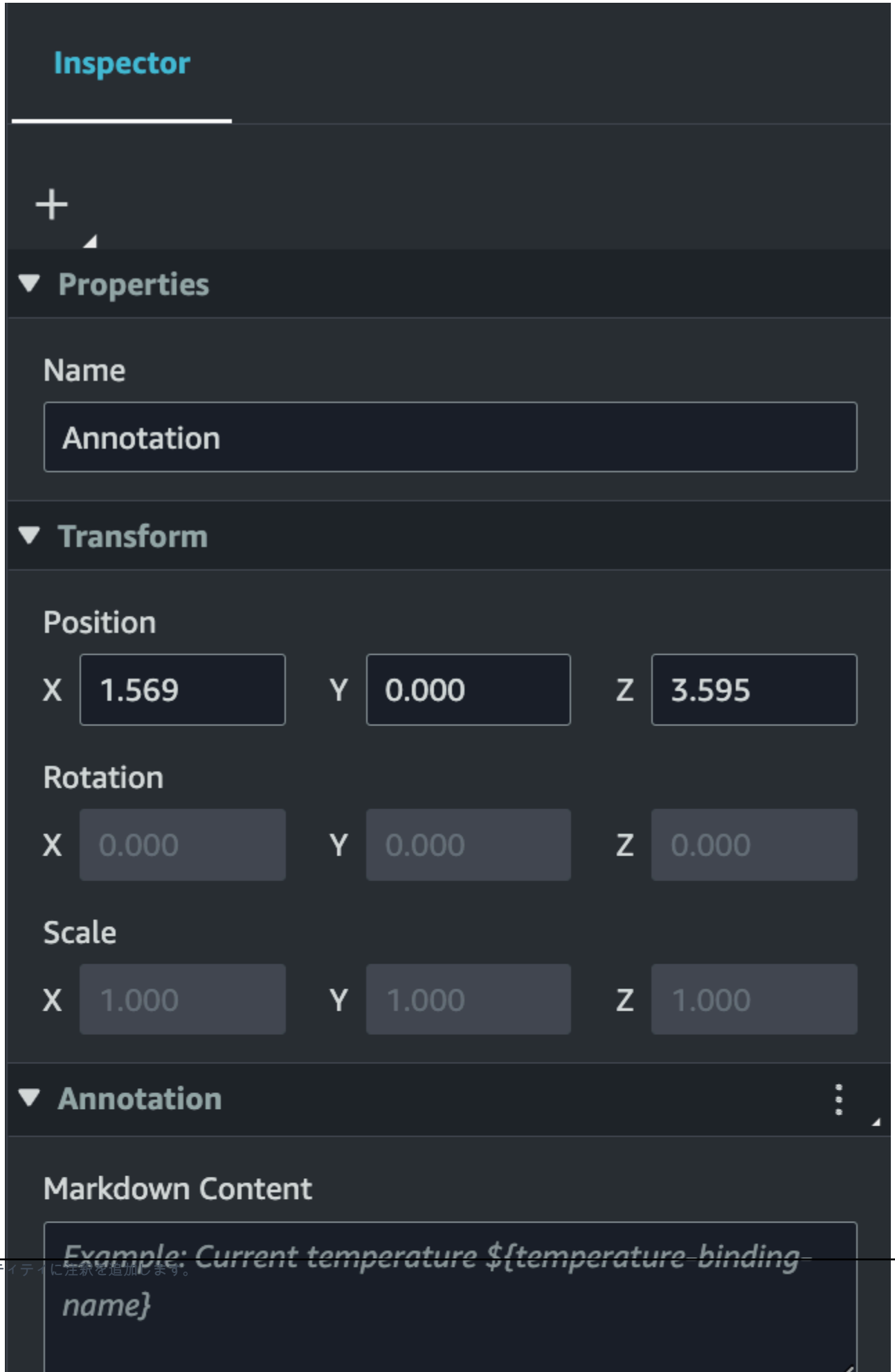
   マークダウンでの記述の詳細については、マークダウン構文に関する公式ドキュメントの「基本構文」を参照してください。



5. AWS IoT TwinMaker シーンデータを注釈にバインドするには、「データバインディングを追加」を選択し、エンティティ ID を追加し、データを表示するエンティティのコンポーネント名とプロパティ名を選択します。バインディング名を更新してマークダウン変数として使用し、データを注釈に表示できます。



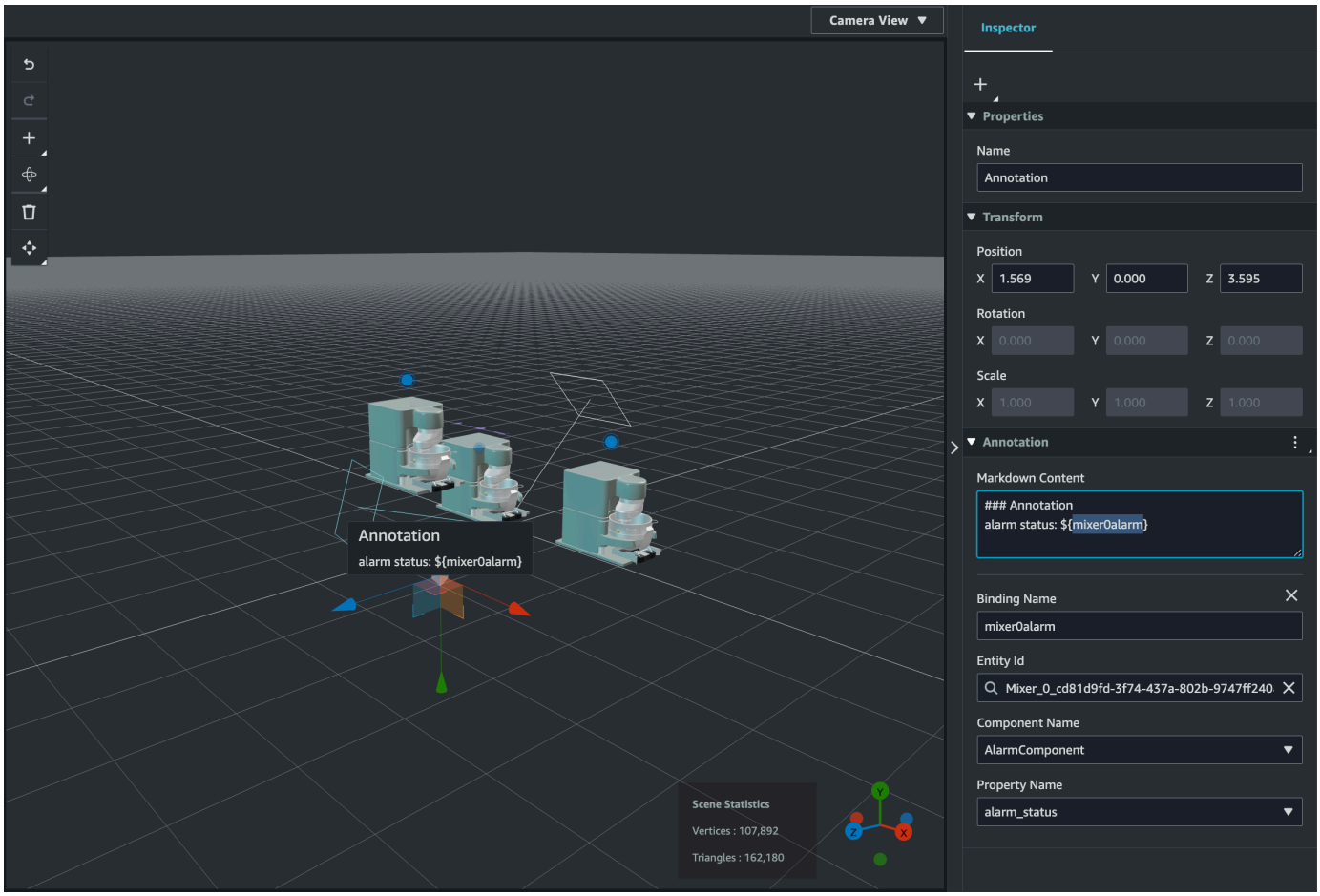
エンティティに注釈を追加します。


Inspector
Properties
Name
Annotation
Transform
Position
X 1.569
Y 0.000
Z 3.595
Rotation
X 0.000
Y 0.000
Z 0.000
Scale
X 1.000
Y 1.000
Z 1.000
Annotation
Markdown Content
Example: Current temperature ${temperature-binding-name}

6. バインディング名は注釈の変数を表すために使用されます。

   バインディング名を入力して、 の AWS IoT TwinMaker変数構文を使用して、アノテーション内のエンティティの時系列の最新の履歴値を表示します。 ${*variable-name*}

   例として、このオーバーレイでは、注釈内の `mixer0alarm` の値が構文 `${mixer0alarm}` とともに表示されます。



タグにオーバーレイを追加

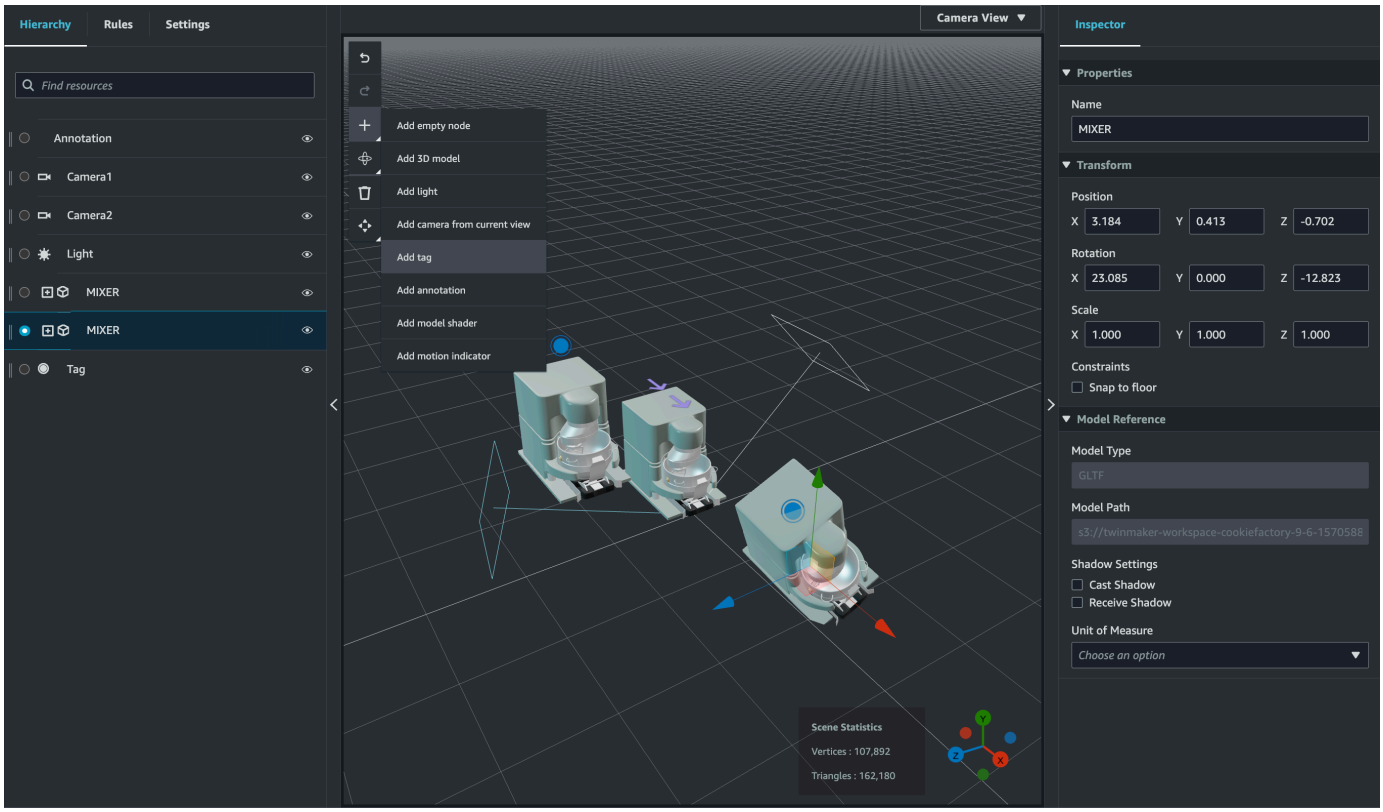
AWS IoT TwinMaker シーンのオーバーレイを作成できます。シーンオーバーレイはタグに関連付けられており、シーンエンティティに関連付けられた重要なデータを表示するために使用できます。オーバーレイはマークダウンで作成およびレンダリングされます。

マークダウンでの記述の詳細については、マークダウン構文に関する公式ドキュメントの「基本構文」を参照してください。

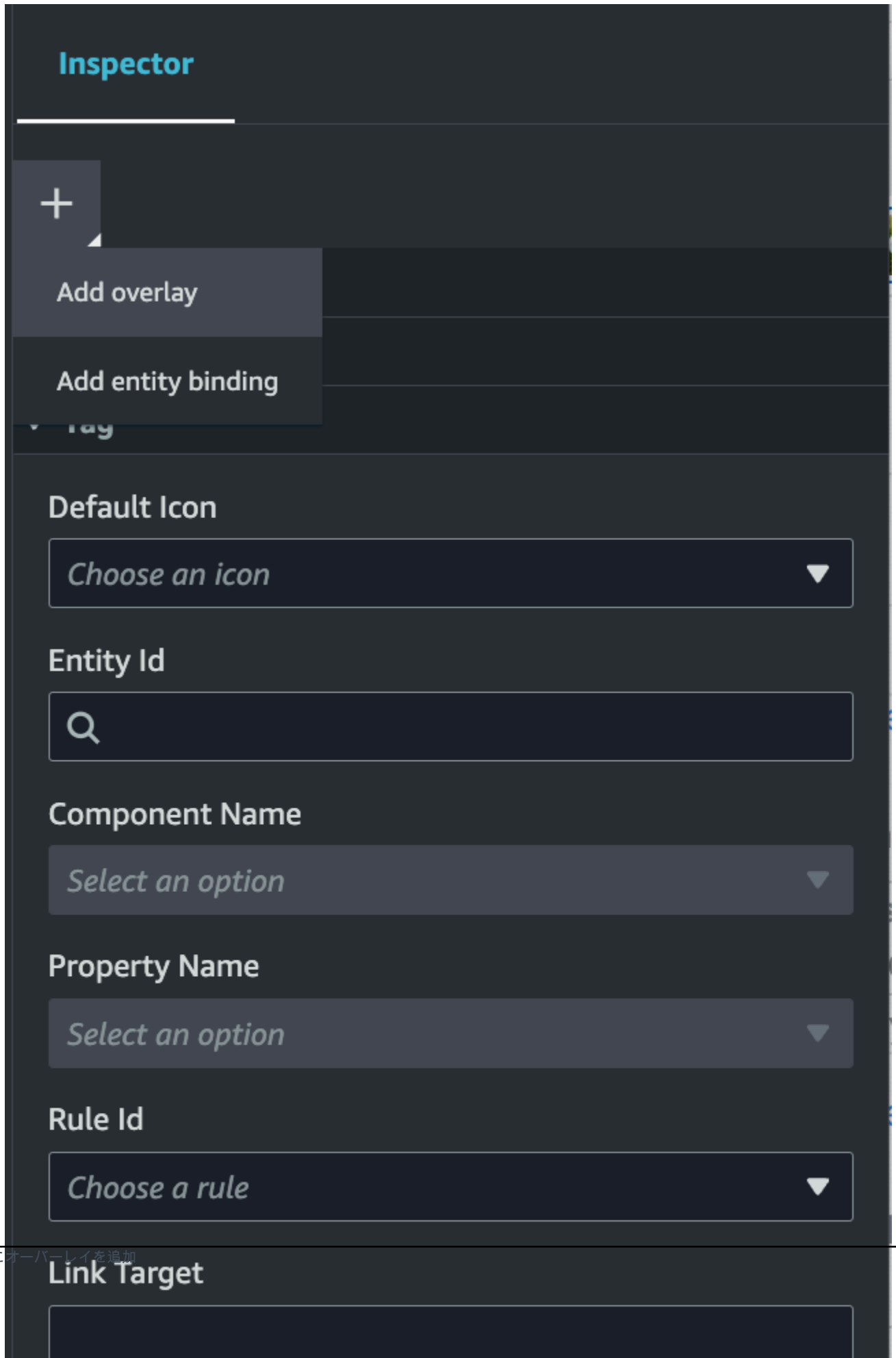
> **Note**
>
> デフォルトでは、オーバーレイに関連付けられたタグが選択されている場合にのみ、オーバーレイがシーンに表示されます。シーン設定でこれを切り替えて、すべてのオーバーレイを一度に表示することができます。

1. 「AWS IoT TwinMaker 」 コンソールでシーンに移動します。
2. AWS IoT TwinMaker オーバーレイはタグシーンに関連付けられており、既存のタグを更新したり、新しいタグを追加したりできます。

   「+」 ボタンを押して、「タグを追加」 オプションを選択します。



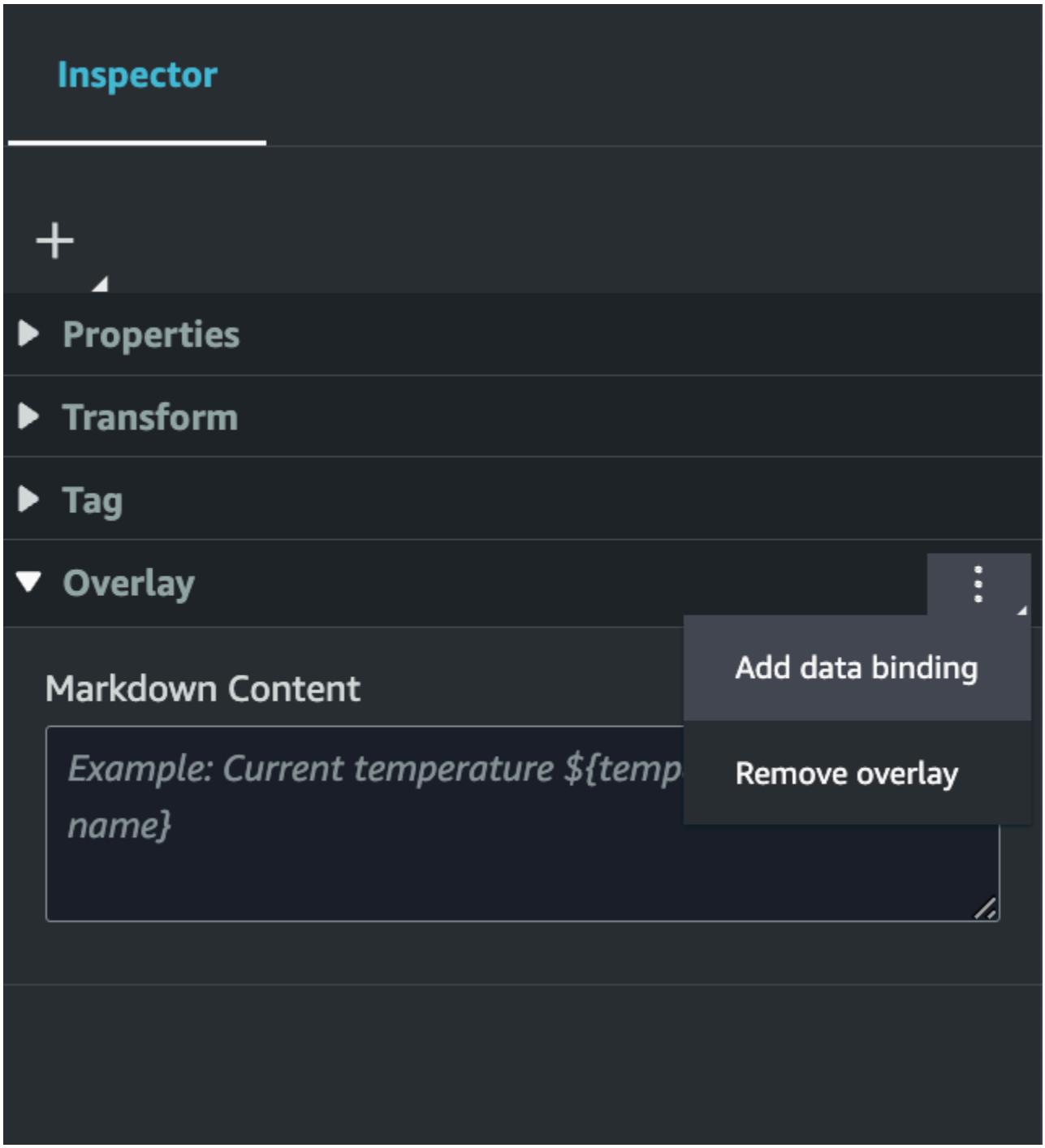
3. 右側の Inspector パネルで + (プラス記号) ボタンを選択し、オーバーレイの追加 を選択します。


Inspector
Add overlay
Add entity binding
Default Icon
Choose an icon
Entity Id
Component Name
Select an option
Property Name
Select an option
Rule Id
Choose a rule
Link Target

4. マークダウン構文で、オーバーレイに表示させるテキストを記述します。

   マークダウンでの記述の詳細については、マークダウン構文に関する公式ドキュメントの「基本構文」を参照してください。

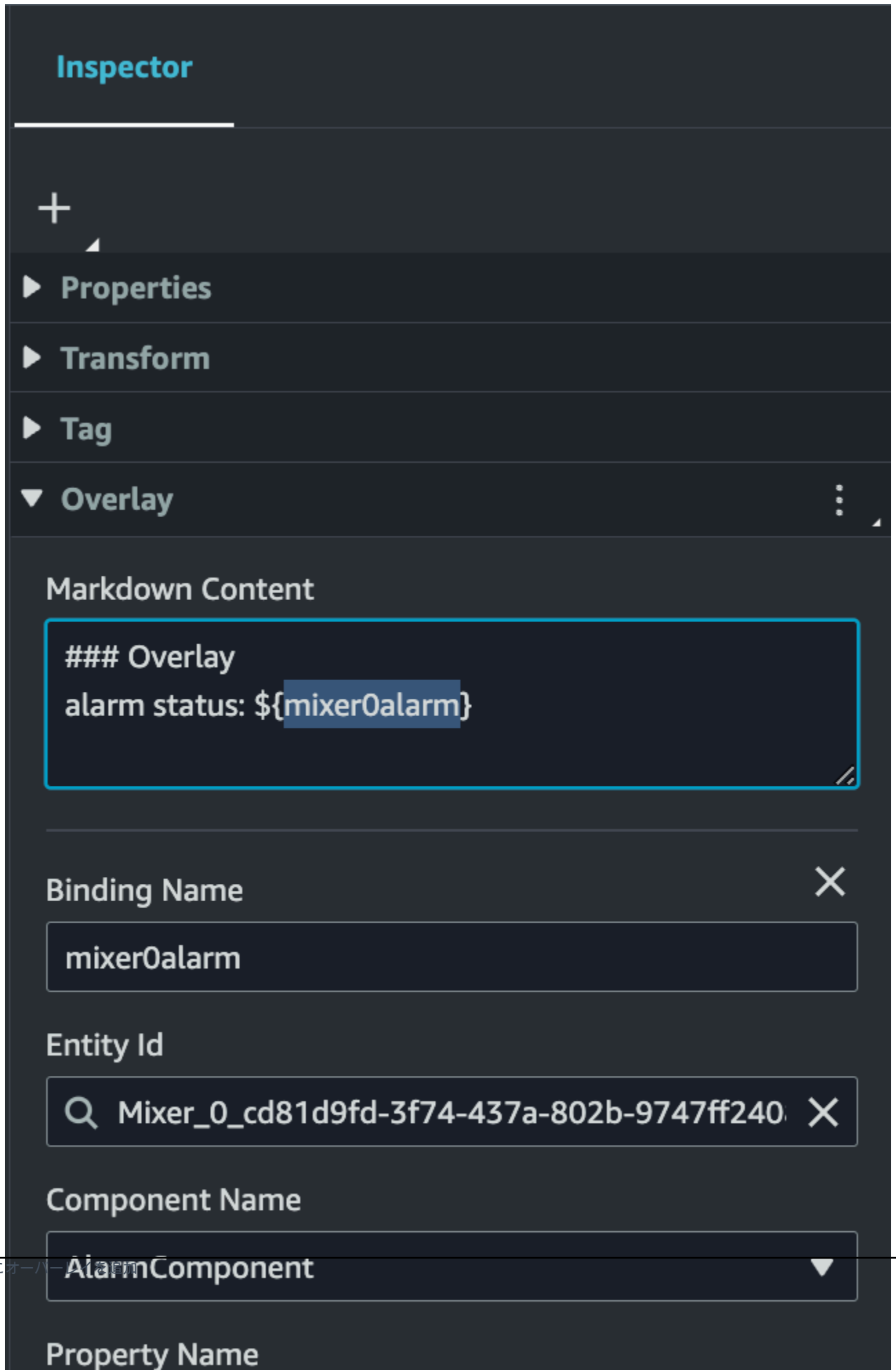
5. AWS IoT TwinMaker シーンデータをオーバーレイにバインドするには、「データバインディングを追加」を選択します。



バインディング名とエンティティ ID を追加し、データを表示するエンティティのコンポーネント名とプロパティ名を選択します。

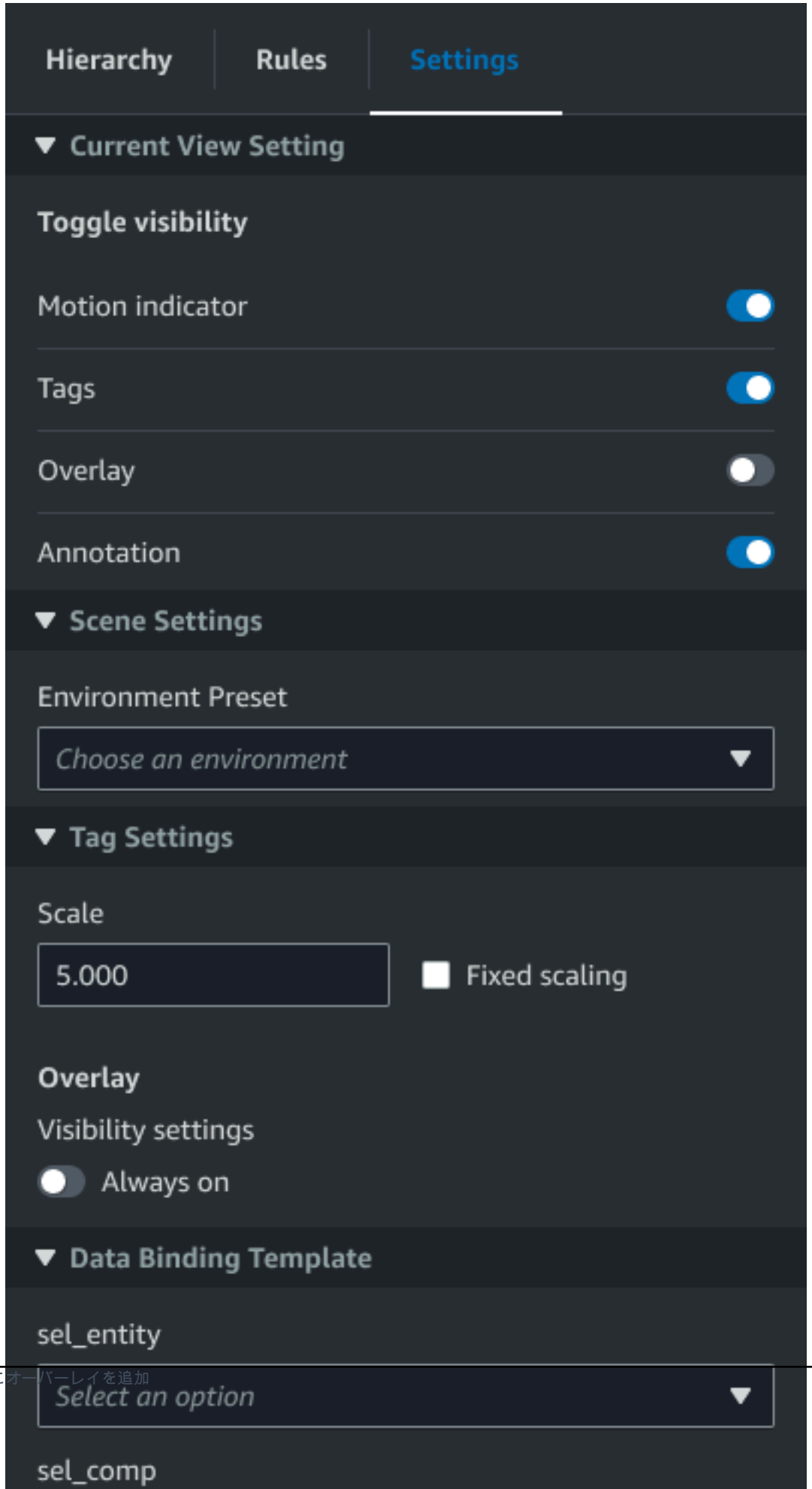
6. の AWS IoT TwinMaker変数構文 を使用して、エンティティの時系列データの最新の履歴値をオーバーレイに表示できます${*variable-name*}。

   例として、このオーバーレイでは、mixer0alarm の値が構文 ${mixer0alarm} とともにオーバーレイに表示されます。


Inspector
Properties
Transform
Tag
Overlay
Markdown Content
Overlay
alarm status: ${mixer0alarm}
Binding Name
mixer0alarm
Entity Id
Mixer_0_cd81d9fd-3f74-437a-802b-9747ff240
Component Name
AlarmComponent
Property Name

7. オーバーレイの可視性を有効にするには、左上の設定タブを開き、すべてのオーバーレイが一度に表示されるようにオーバーレイのトグルがオンになっていることを確認します。

> **Note**
>
> デフォルトでは、オーバーレイに関連付けられたタグが選択されている場合にのみ、オーバーレイがシーンに表示されます。


Hierarchy
Rules
Settings
Current View Setting
Toggle visibility
Motion indicator
Tags
Overlay
Annotation
Scene Settings
Environment Preset
Choose an environment
Tag Settings
Scale
5.000
Fixed scaling
Overlay
Visibility settings
Always on
Data Binding Template
sel_entity
Select an option
sel_comp

シーンを編集

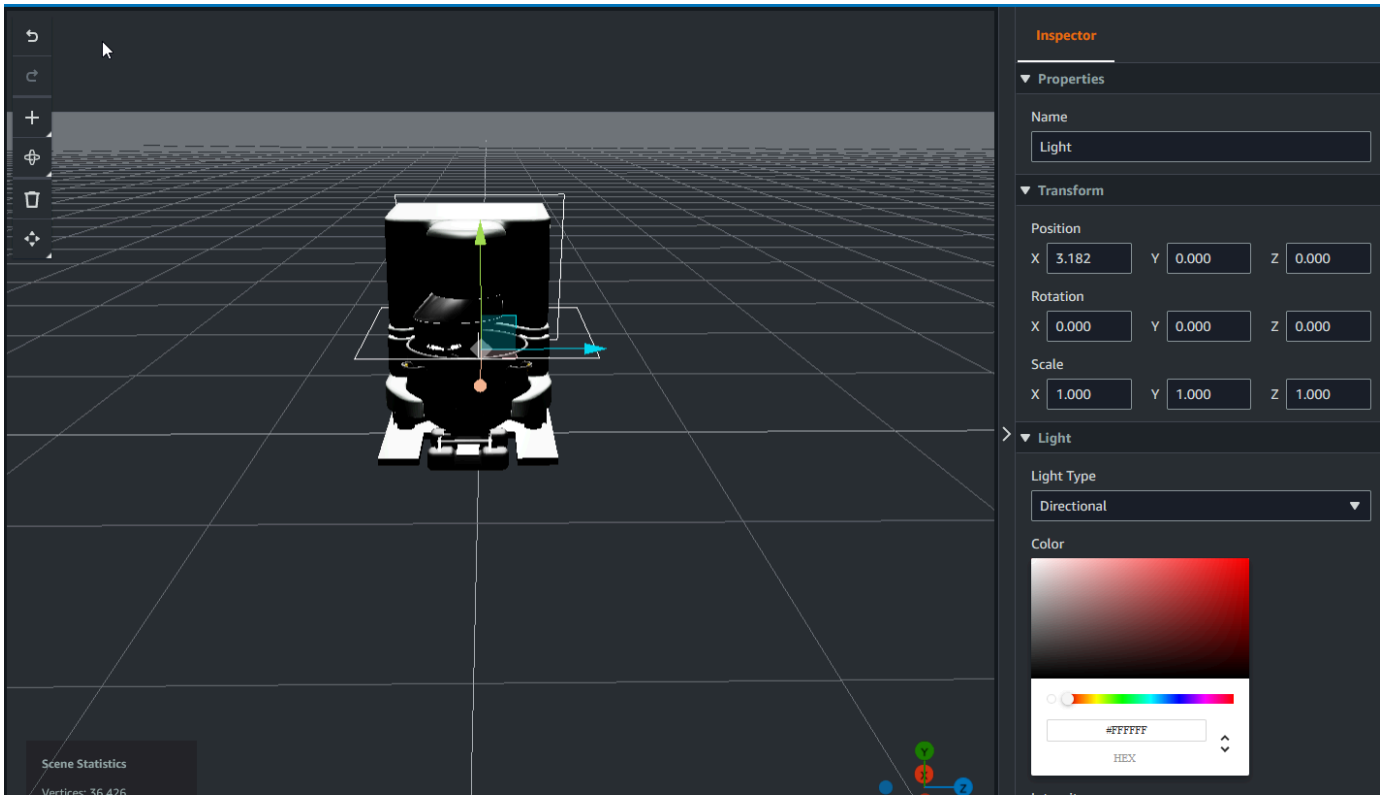
シーンを作成したら、エンティティやコンポーネントを追加したり、拡張ウィジェットをシーンに設定したりできます。エンティティコンポーネントとウィジェットを使用してデジタルツインをモデル化し、ユースケースに合った機能を提供します。

シーンにモデルを追加

シーンにモデルを追加するには、次の手順に従います。

> **Note**
>
> シーンにモデルを追加するには、まずモデルを AWS IoT TwinMaker リソースライブラリにアップロードする必要があります。詳細については、「リソースライブラリへの AWS IoT TwinMaker リソースのアップロード」を参照してください。

1. シーンコンポーザーページで、プラス (+) 記号を選択し、次に「3Dモデルの追加」を選択します。
2. 「リソースライブラリからリソースを追加」ウィンドウで、CookieFactorMixer.glb ファイルを選択し、「 を追加」を選択します。シーンコンポーザーが開きます。
3. オプション：プラス (+) 記号を選択し、「ライトを追加」を選択します。
4. 各ライトオプションを選択して、シーンにどのように影響するかを確認してください。



> **Note**
>
> シーンにはデフォルトのアンビエントライティングがあります。フレームレートの低下を防ぐには、シーンに追加するライトの数を制限することを検討してください。

モデルシェーダー拡張 UI ウィジェットをシーンに追加する

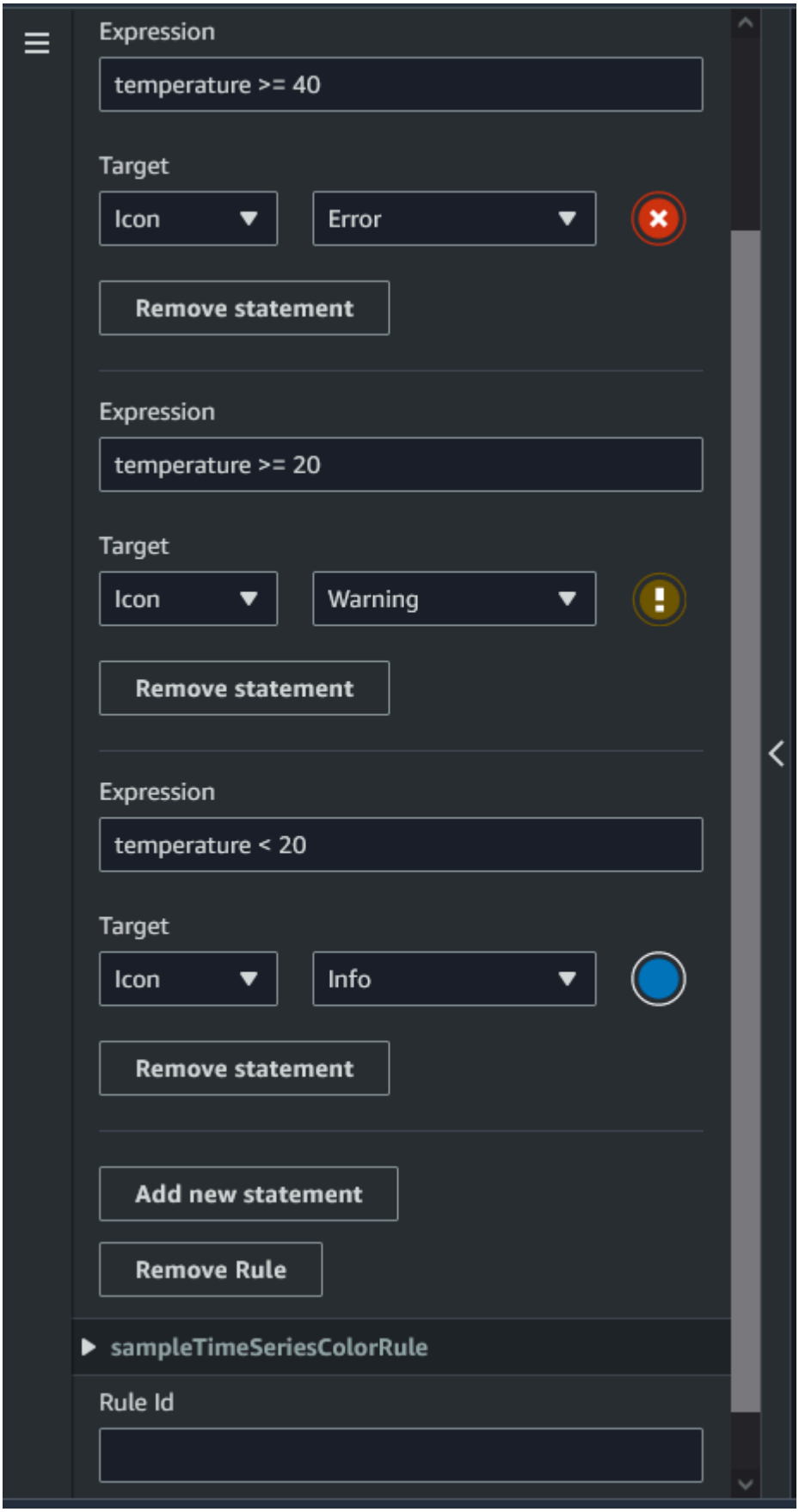
モデルシェーダーウィジェットは、定義した条件下でオブジェクトの色を変更できます。たとえば、シーン内のクッキーミキサーの色をミキサーの温度データに基づいて変更するカラーウィジェットを作成できます。

選択したオブジェクトにモデルシェーダーウィジェットを追加するには、次の手順に従います。

1. ウィジェットを追加する階層内のオブジェクトを選択します。+ ボタンを押して、モデルシェーダー を選択します。
2. 新しいビジュアルルールグループを追加するには、まず以下の手順に従って を作成し ColorRule、次にルール ID の オブジェクトの Inspector パネルで を選択しますColorRule。
3. entityID を選択し ComponentName、モデルシェーダーを PropertyName バインドします。

シーンのビジュアルルールを作成します

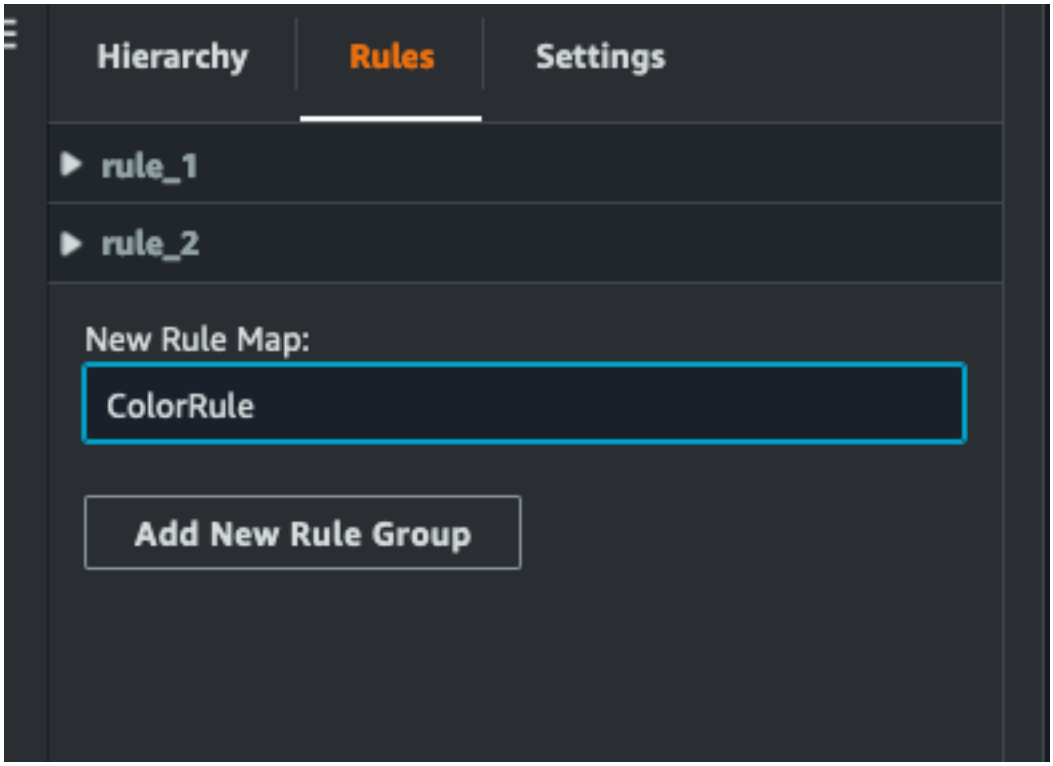
ビジュアルルールマップを使用して、タグやモデルシェーダーなどの拡張 UI ウィジェットの視覚的な外観を変更するデータ駆動型条件を指定できます。サンプルルールも用意されていますが、独自のルールを作成することもできます。次の例は、ビジュアルルールを示しています。


Expression
temperature >= 40
Target
Icon
Error
Remove statement
Expression
temperature >= 20
Target
Icon
Warning
Remove statement
Expression
temperature < 20
Target
Icon
Info
Remove statement
Add new statement
Remove Rule
sampleTimeSeriesColorRule
Rule Id

上の図は、ID が「温度」で以前に定義されたデータプロパティが特定の値と照合されるときのルールを示しています。例えば、「温度」が 40 以上の場合、状態はタグの外観を赤い円に変更します。Grafanaダッシュボードでターゲットを選択すると、同じデータソースを使用するように設定されている詳細パネルに入力されます。

次の手順は、メッシュカラー化拡張UIレイヤーの新しいビジュアルルールグループを追加する方法を示しています。

1. コンソールのルールタブで、 ColorRule テキストフィールドに などの名前を入力し、新しいルールグループを追加 を選択します。



2. ユースケースに合わせてルールを定義します。例えば、データプロパティ「温度」に基づいて作成できます。報告された値は 20 未満です。ルール式には、次の構文を使用します。 未満は < 、 より大きいは > 、 以下は <=、 以上は >=、 等しいは ==。 (詳細については、「Apache Commons JEXL 構文」を参照してください。)
3. ターゲットを色に設定します。などの色を定義するには、16 進値`#fcba03`を使用します。(16 進値の詳細については、「16 進数」を参照してください。)

シーン用のタグの作成

タグは、シーンの特定のx,y,z座標位置に追加される注釈です。タグはエンティティプロパティを使用してシーンパーツをナレッジグラフに接続します。タグを使用して、シーン内のアイテム（アラームなど）の動作や外観を設定できます。

> **Note**
>
> タグに機能を追加するには、タグにビジュアルルールを適用します。

以下の手順で、シーンにタグを追加します。

1. 階層内のオブジェクトを選択し、「+」ボタンを選択し、「タグを追加」を選択します。
2. タグに名前を付けます。次に、ビジュアルルールを適用するには、ビジュアルグループIDを選択します。
3. ドロップダウンリストで、EntityID 、 ComponentName、および を選択します PropertyName。
4. データパスフィールドにデータを入力するには、 の作成 DataFrameLabelを選択します。

3D タイルモデル形式

シーンでの 3D タイルの使用

に 3D シーンをロードするときに長い待機時間が発生したり、複雑な 3D モデルをナビゲートするときにレンダリングパフォーマンス AWS IoT TwinMaker が低下したりする場合は、モデルを 3D タイルに変換することをお勧めします。このセクションでは、3D タイルの形式と使用可能なサードパーティー製ツールについて説明します。3D タイルがユースケースに適しているかどうかを判断し、使用開始に役立てるには、「」を読んでください。

複雑なモデルのユースケース

AWS IoT TwinMaker シーン内の 3D モデルは、モデルが次の場合、ロード時間の遅延やナビゲーションの遅延などのパフォーマンスの問題を引き起こす可能性があります。

- 大きい： ファイルサイズが 100MBを超えています。
- 高密度： 数百または数千の異なるメッシュで構成されます。
- 複雑な：メッシュジオメトリには、複雑なシェイプを形成するために数百万の三角形があります。

3D タイル形式

3D Tiles 形式は、モデルジオメトリをストリーミングし、3D レンダリングのパフォーマンスを向上させるためのソリューションです。これにより、 AWS IoT TwinMaker シーンに 3D モデルを瞬時にロードでき、カメラビューに表示される内容に基づいてモデルのチャンクにロードすることで 3D インタラクションを最適化します。

3D Tiles 形式は Cesium によって作成されました。Cesium には、3D モデルを Cesium Ion と呼ばれる 3D Tiles に変換するマネージドサービスがあります。これは現在、3D タイルを作成するための最適なソリューションであり、サポートされている形式の複雑なモデルにこれをお勧めします。Cesium を登録し、Cesium の料金ページでビジネス要件に基づいて適切なサブスクリプションプランを選択できます。

AWS IoT TwinMaker シーンに追加できる 3D Tiles モデルを準備するには、Cesium Ion に記載されている指示に従ってください。

- Cesium Ion にモデルをインポートする

Cesium 3D タイルを にアップロードする AWS

モデルが 3D タイルに変換されたら、モデルファイルをダウンロードし、 AWS IoT TwinMaker ワークスペースの Amazon S3 バケットにアップロードします。

1. 3D Tiles モデルアーカイブ を作成してダウンロードします。
2. アーカイブをフォルダに解凍します。
3. 3D Tiles フォルダ全体をワークスペースに関連付けられた Amazon S3 バケットにアップロードします AWS IoT TwinMaker 。 (「Amazon S3 ユーザーガイド」の「オブジェクトのアップロード」を参照してください。) Amazon S3
4. 3D Tiles モデルが正常にアップロードされると、 AWS IoT TwinMaker Resource Library にタイプ の Amazon S3 フォルダパスが表示されます`Tiles3D`。

> **Note**
>
> AWS IoT TwinMaker Resource Library は、3D Tiles モデルを直接アップロードすることはできません。

での 3D タイルの使用 AWS IoT TwinMaker

AWS IoT TwinMaker は、ワークスペース S3 バケットにアップロードされた 3D タイルモデルを認識しています。モデルには、同じ Amazon S3 ディレクトリで使用可能な `tileset.json`とすべての依存ファイル (.gltf、.b3dm、.i3dm、.cmpt、.pnts) が必要です。 Amazon S3 Amazon S3 ディレクトリパスは、 タイプのリソースライブラリに表示されます`Tiles3D`。

シーンに 3D タイルモデルを追加するには、次の手順に従います。

1. シーンコンポーザーページで、プラス (+) 記号を選択し、次に「3Dモデルの追加」を選択します。
2. リソースライブラリからリソースを追加ウィンドウで、タイプ の 3D タイルモデルへのパスを選択し`Tiles3D`、 を追加を選択します。
3. キャンバスをクリックして、モデルをシーンに配置します。

3D タイルの違い

3D Tiles は現在、ジオメトリメタデータとセマンティックメタデータをサポートしていません。つまり、元のモデルのメッシュ階層はサブモデル選択機能では使用できません。3D タイルモデルにウィジェットを追加することはできますが、モデルシェーダー、分割された 3D 変換、サブモデルメッシュのエンティティバインディングなどのサブモデルに微調整された機能は使用できません。

シーンの背景のコンテキストとして機能する大きなアセットには、3D タイル変換を使用することをお勧めします。サブモデルをさらに分割して注釈を付ける場合は、別の glTF /glb アセットとして抽出し、シーンに直接追加する必要があります。これは、Blender などの無料および一般的な 3D ツールで実行できます。

ユースケースの例：

- 詳細な機械室と床、電気ボックス、およびパイプのパイプを備えた工場の 1GB モデルがあります。関連するプロパティデータがしきい値を超えた場合、電気ボックスとパイプは赤く光る必要があります。
- モデル内のボックスメッシュとパイプメッシュを分離し、Blender を使用して別の glTF にエクスポートします。
- 電気要素やパイプ要素なしでファクトリを 3D タイルモデルに変換し、S3 にアップロードします。
- オリジン (0,0,0) の AWS IoT TwinMaker シーンに 3D Tiles モデルと glTF モデルの両方を追加します。

- glTF の電気ボックスとパイプサブモデルにモデルシェーダーコンポーネントを追加して、プロパティルールに基づいてメッシュを赤にします。

動的シーン

AWS IoT TwinMaker scenes は、シーンノードと設定をエンティティコンポーネントに保存することで、ナレッジグラフのパワーを引き出します。 AWS IoT TwinMaker コンソールを使用して動的シーンを作成し、3D シーンをより簡単に管理、構築、レンダリングできます。

主な機能：

- すべての 3D シーンノードオブジェクト、設定、およびデータバインディングは、ナレッジグラフクエリに基づいて「動的」にレンダリングされます。
- Grafana またはカスタムアプリケーションで読み取り専用のシーンビューワーを使用すると、30 秒間隔でシーンの更新を取得できます。

静的シーンと動的シーン

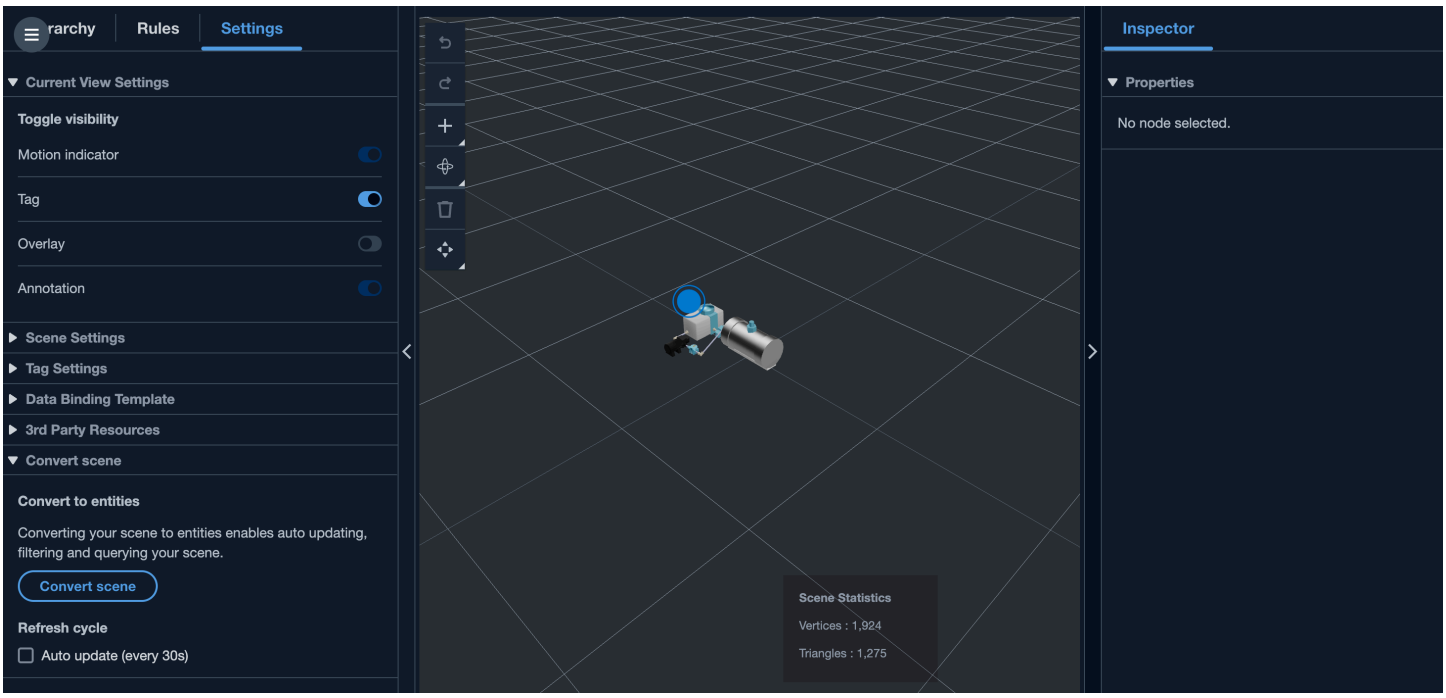
静的シーンは、すべてのシーンノードと設定の詳細を含む S3 に保存されているシーン JSON ファイルで構成されます。シーンへの変更は、JSON ドキュメントに加え、S3 に保存する必要があります。基本的な料金プラン がある場合は、静的シーンが唯一のオプションです。

動的シーンはシーンのグローバル設定を持つシーン JSON ファイルで構成され、他のすべてのシーンノードとノード設定はエンティティコンポーネントとしてナレッジグラフに保存されます。動的シーンは、標準および階層型バンドルの料金プランでのみサポートされます。料金プランのアップグレード方法については、AWS IoT TwinMaker 価格設定モードの切り替え「」を参照してください）。

以下の手順に従って、既存の静的シーンを動的シーンに変換できます。

- 「AWS IoT TwinMaker 」 コンソールでシーンに移動します。
- 左側のパネルで、設定タブをクリックします。
- パネルの下部にあるシーンの変換セクションを展開します。
- シーンの変換ボタンをクリックし、確認 をクリックします。

> **Warning**
>
> 静的シーンから動的シーンへの変換は元に戻せません。



シーンコンポーネントタイプとエンティティ

シーン固有のエンティティコンポーネントを作成するには、次の 1P コンポーネントタイプがサポートされています。

- com.amazon.iottwinmaker.3d.component.camera カメラウィジェット の設定を保存するコンポーネントタイプ。
- com.amazon.iottwinmaker.3d.component.dataoverlay 注釈ウィジェットまたはタグウィジェットのオーバーレイの設定を保存するコンポーネントタイプ。
- com.amazon.iottwinmaker.3d.component.light ライトウィジェットの設定を保存するコンポーネントタイプ。
- com.amazon.iottwinmaker.3d.component.modelref シーンで使用される 3D モデルの設定と S3 の場所を保存するコンポーネントタイプ。
- com.amazon.iottwinmaker.3d.component.modelshader モデルシェーダーの設定を 3D モデルに保存するコンポーネントタイプ。

- com.amazon.iottwinmaker.3d.component.motionindicator モーションインジケータウィジェットの設定を保存するコンポーネントタイプ。
- com.amazon.iottwinmaker.3d.component.submodelref 3D モデルのサブモデルの設定を保存するコンポーネントタイプ。
- com.amazon.iottwinmaker.3d.component.tag タグウィジェット の設定を保存するコンポーネントタイプ。
- com.amazon.iottwinmaker.3d.node 3D 変換、名前、汎用プロパティなどのシーンノードの基本設定を保存するコンポーネントタイプ。

動的シーンの概念

動的シーンエンティティは、 というラベルのグローバルエンティティの下に保存されます$SCENES。各シーンは、ルートエンティティと、シーンノード階層に一致する子エンティティの階層で構成されます。ルートの下にある各シーンノードには、com.amazon.iottwinmaker.3d.node コンポーネントと、ノードのタイプ (3D モデル、ウィジェットなど) のコンポーネントがあります。

> **Warning**
>
> シーンエンティティを手動で削除しないでください。シーンが壊れている可能性があります。シーンを部分的にまたは完全に削除する場合は、シーンコンポーザーページを使用してシーンノードを追加および削除し、シーンページを使用してシーンを選択および削除します。

AWS IoT TwinMaker UI コンポーネントを使用してカスタマイズされたウェブアプリケーションを作成する

AWS IoT TwinMaker は、 AWS IoT アプリケーションデベロッパー向けのオープンソースの UI コンポーネントを提供します。これらの UI コンポーネントを使用すると、デベロッパーはデジタルツインに対応した AWS IoT TwinMaker 機能を使用してカスタマイズされたウェブアプリケーションを構築できます。

AWS IoT TwinMaker UI コンポーネントは、 IoT AWS IoT アプリケーションデベロッパーが複雑な IoT IoT アプリケーションの開発を簡素化できるようにするオープンソースのクライアント側ライブラリである Application Kit の一部です。

AWS IoT TwinMaker UI コンポーネントには以下が含まれます。

- AWS IoT TwinMaker ソース：

  データを取得し、データやデジタルツインとやり取りするための AWS IoT TwinMaker データコネクタコンポーネント。

  詳細については、「AWS IoT TwinMaker ソースドキュメント」を参照してください。
- シーンビューアー:

  デジタルツインをレンダリングして操作できるように `@react-three/fiber` 上に構築された 3D レンダリングコンポーネント。

  詳細については、「シーンビューアードキュメント」を参照してください。
- ビデオプレーヤー:

  を介して Kinesis Video Streams からビデオをストリーミングできるビデオプレーヤーコンポーネント AWS IoT TwinMaker。

  詳細については、「ビデオプレイヤードキュメント」を参照してください。

AWS IoT Application Kit の使用の詳細については、Application AWS IoT Kit Github ページを参照してください。

AWS IoT Application Kit を使用して新しいウェブアプリケーションを起動する方法については、 IoT App Kit の公式ドキュメントページを参照してください。

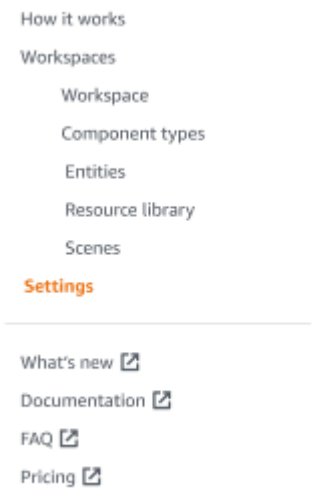
AWS IoT TwinMaker 価格設定モードの切り替え

AWS IoT TwinMaker 現在、ベーシック、スタンダード、階層型バンドルの3つの価格設定モードがあります。スタンダード料金モードが、デフォルトとして設定されています。

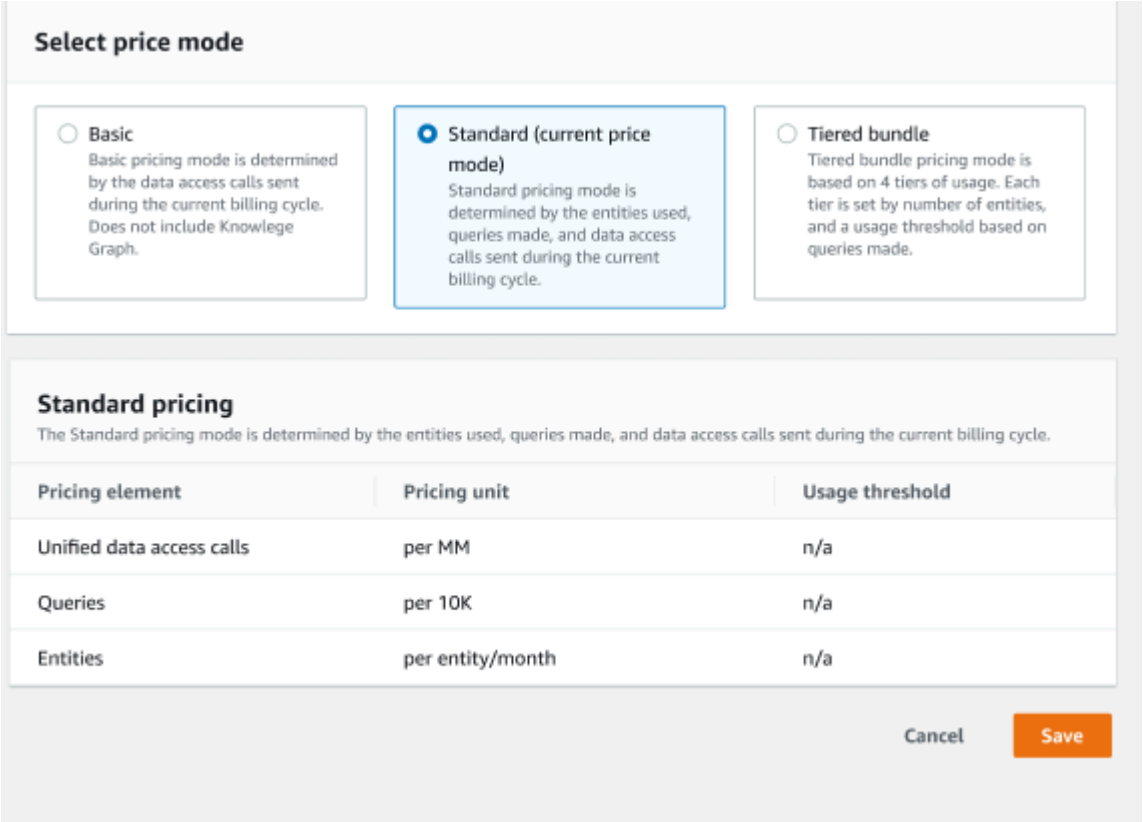
使用量ベースから階層型ベースの料金モードへの変更はいつでも可能ですが、変更は次回の請求サイクルの開始時に有効になります。使用量ベースから階層型ベースの料金モードに切り替えた後は、その後 3 回の使用サイクルの間は、使用料ベースの料金モードに戻すことはできません。ベーシックからスタンダードに切り替えると、変更は直ちに有効になります。詳細とコスト情報については、「価格設定」を参照してくださいAWS IoT TwinMaker 。

以下の手順では、AWS IoT TwinMaker コンソールで料金モードを切り替える方法を説明します。

1. AWS IoT TwinMaker コンソールを開きます。
2. 左側のナビゲーションペインで [設定] を選択します。「価格設定」ページが開きます。



3. 「料金モードを変更」 を選択します。
4. 次のスクリーンショットのように、「スタンダード」 または 「階層型バンドル」 モードのいずれかを選択します。



5. 「保存」 を選択して、新しい料金モードを確定します。
6. これで、料金モードが変更されました。

> **Note**
>
> 使用量ベースから階層型ベースの料金モードへの変更はいつでも可能ですが、変更は次回の請求サイクルの開始時に有効になります。使用量ベースから階層型ベースの料金モードに切り替えた後は、その後 3 回の使用サイクルの間は、使用料ベースの料金モードに戻すことはできません。ベーシックからスタンダードに切り替えると、変更は直ちに有効になります。

AWS IoT TwinMaker ナレッジグラフ

AWS IoT TwinMaker ナレッジグラフは、 AWS IoT TwinMaker ワークスペースに含まれるすべての情報を整理し、視覚的なグラフ形式で表示します。エンティティ、コンポーネント、コンポーネントタイプに対してクエリを実行して、 AWS IoT TwinMaker リソース間のリレーションシップを示す視覚的なグラフを生成できます。

以下のトピックでは、ナレッジグラフを使用し、統合する方法について説明します。

トピック

AWS IoT TwinMaker ナレッジグラフのコアコンセプト

このトピックでは、ナレッジグラフ機能の主要な概念と用語について説明します。

ナレッジグラフの仕組み:

ナレッジグラフは、CreateEntityエンティティとそのコンポーネントと既存の UpdateEntityAPI との関係を作成します。リレーションシップは、エンティティのコンポーネントで定義される特別なデータタイプ RELATIONSHIP のプロパティにすぎません。 AWS IoT TwinMaker Knowledge Graph は ExecuteQueryAPI を呼び出し、エンティティ内のデータまたはエンティティ間の関係に基づいてクエリを実行します。Knowledge Graphは、クエリの記述に役立つグラフマッチ構文サポートを新たに追加した柔軟なPartiQLクエリ言語（ AWS 多くのサービスで使用されています）を使用しています。呼び出しが行われた後は、結果を表として表示したり、接続されたノードやエッジのグラフとして視覚化したりできます。

ナレッジグラフの主要用語:

- エンティティグラフ: ワークスペース内のノードとエッジの収集。
- ノード: ワークスペース内のすべてのエンティティがエンティティグラフのノードになります。

- エッジ: エンティティのコンポーネントに定義されているすべてのリレーションシッププロパティがエンティティグラフのエッジになります。さらに、 parentEntityId エンティティのフィールドを使用して定義された階層的な親子関係は、関係名が "" isChildOf のエンティティグラフのエッジにもなります。すべてのエッジは方向性のあるエッジです。
- リレーションシップ: AWS IoT TwinMaker リレーションシップはエンティティのコンポーネントの特殊なタイプのプロパティです。 AWS IoT TwinMaker CreateEntityまたはUpdateEntityAPI を使用してリレーションシップを定義および編集できます。では AWS IoT TwinMaker、リレーションシップはエンティティのコンポーネントで定義する必要があります。リレーションシップを独立したリソースとして定義することはできません。リレーションシップは、あるエンティティから別のエンティティへの方向性がある必要があります。

AWS IoT TwinMaker ナレッジグラフクエリの実行方法

AWS IoT TwinMaker ナレッジグラフを使用する前に、以下の前提条件を満たしていることを確認してください。

- ワークスペースを作成します。 AWS IoT TwinMaker ワークスペースは、AWS IoT TwinMaker コンソールで作成できます。
- AWS IoT TwinMakerのエンティティコンポーネントシステムとエンティティの作成方法に慣れてください。詳細については、「最初のエンティティを作成する」を参照してください。
- AWS IoT TwinMakerのデータコネクタに慣れてください。詳細については、「AWS IoT TwinMaker データコネクター」を参照してください。

> **Note**
>
> AWS IoT TwinMaker Knowledge Graph を使用するには、標準バンドル価格設定モードまたは階層型バンドルの価格設定モードになっている必要があります。詳細については、「AWS IoT TwinMaker 価格設定モードの切り替え」を参照してください。

次の手順では、クエリを作成、実行、保存、および編集する方法を示します。

クエリエディタを開く

ナレッジグラフのクエリエディターに移動するには

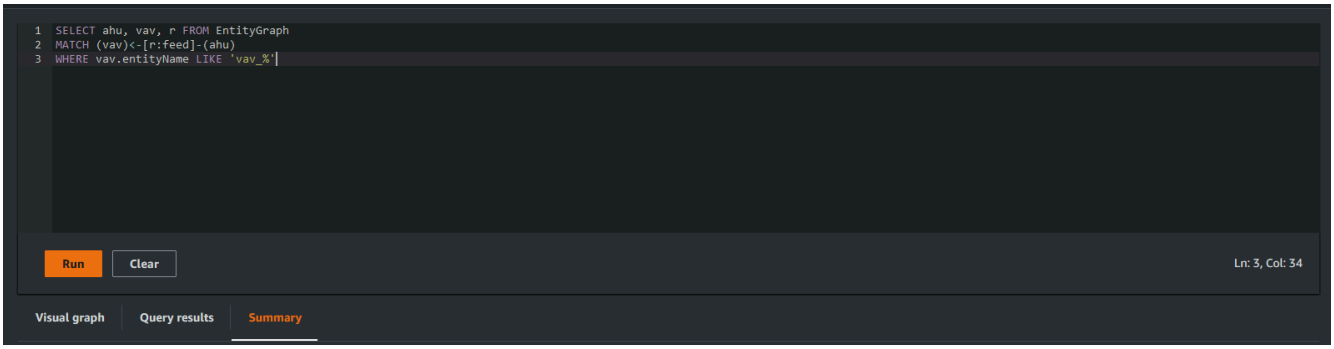
1. AWS IoT TwinMaker コンソールを開きます。

2. ナレッジグラフを使用したいワークスペースを開きます。
3. 左のナビゲーションメニューの 「クエリエディタ」 を選択します。
4. クエリエディターが開きます。ワークスペースのリソースに対してクエリを実行できるようになりました。

クエリを実行する

クエリを実行してグラフを生成するには

1. クエリエディタで、「エディタ」 タブを選択して構文エディタを開きます。
2. エディタースペースで、ワークスペースのリソースに対して実行したいクエリを記述します。



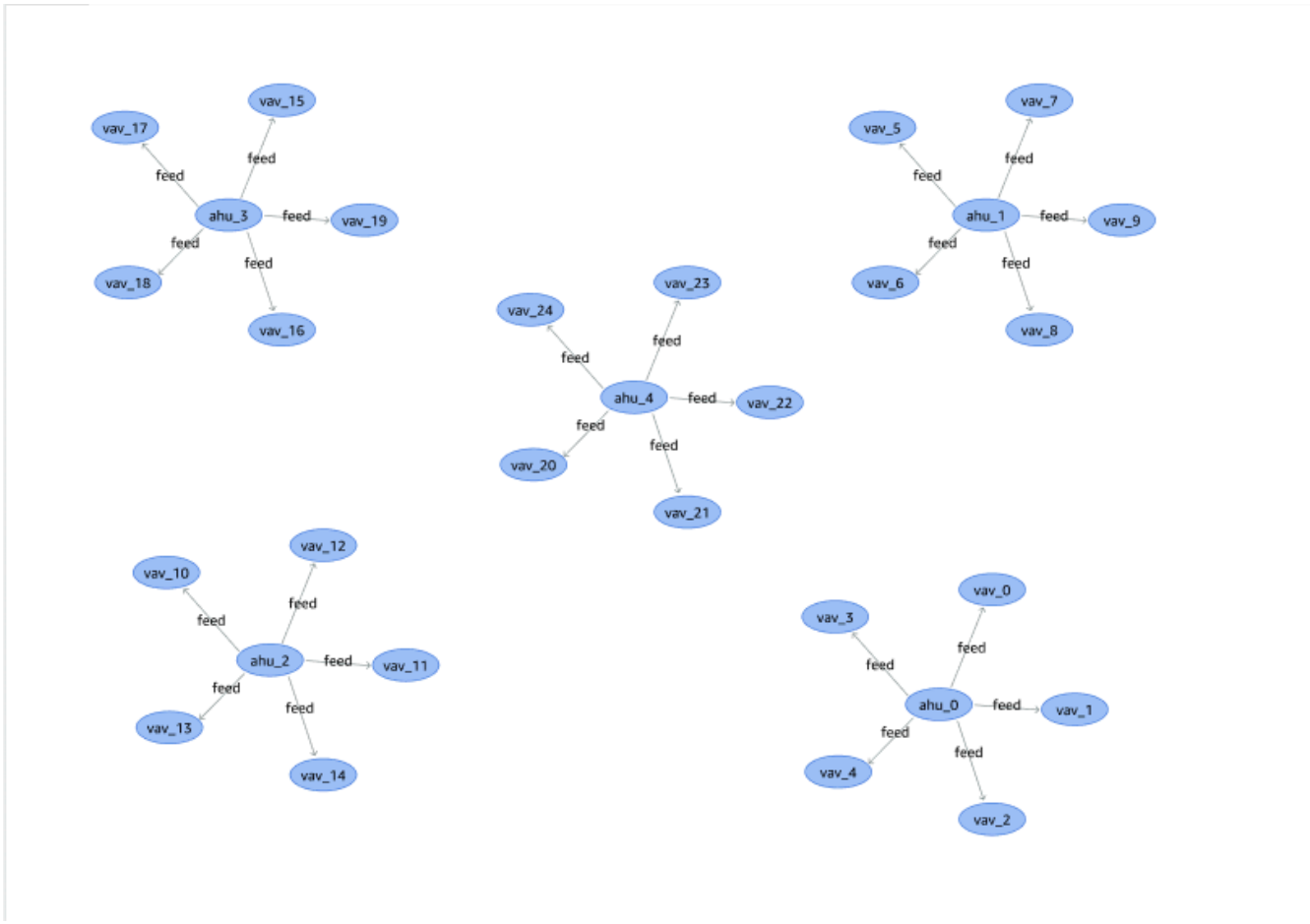
この例では、vav_%リクエストは名前に含まれるエンティティを検索し、feed次のコードを使用してそれらのエンティティ間の関係に基づいて整理します。

```
SELECT ahu, vav, r FROM EntityGraph
MATCH (vav)<-[r:feed]-(ahu)
WHERE vav.entityName LIKE 'vav_%'
```

> **Note**
>
> ナレッジグラフの構文は PartiQL を使用します。この構文の詳細については、を参照してくださいAWS IoT TwinMaker ナレッジグラフの追加リソース。

3. [Run query] を選択して、作成したリクエストを実行します。

リクエストに基づいてグラフが生成されます。



上のグラフ例は、ステップ 2 のクエリ例に基づいています。

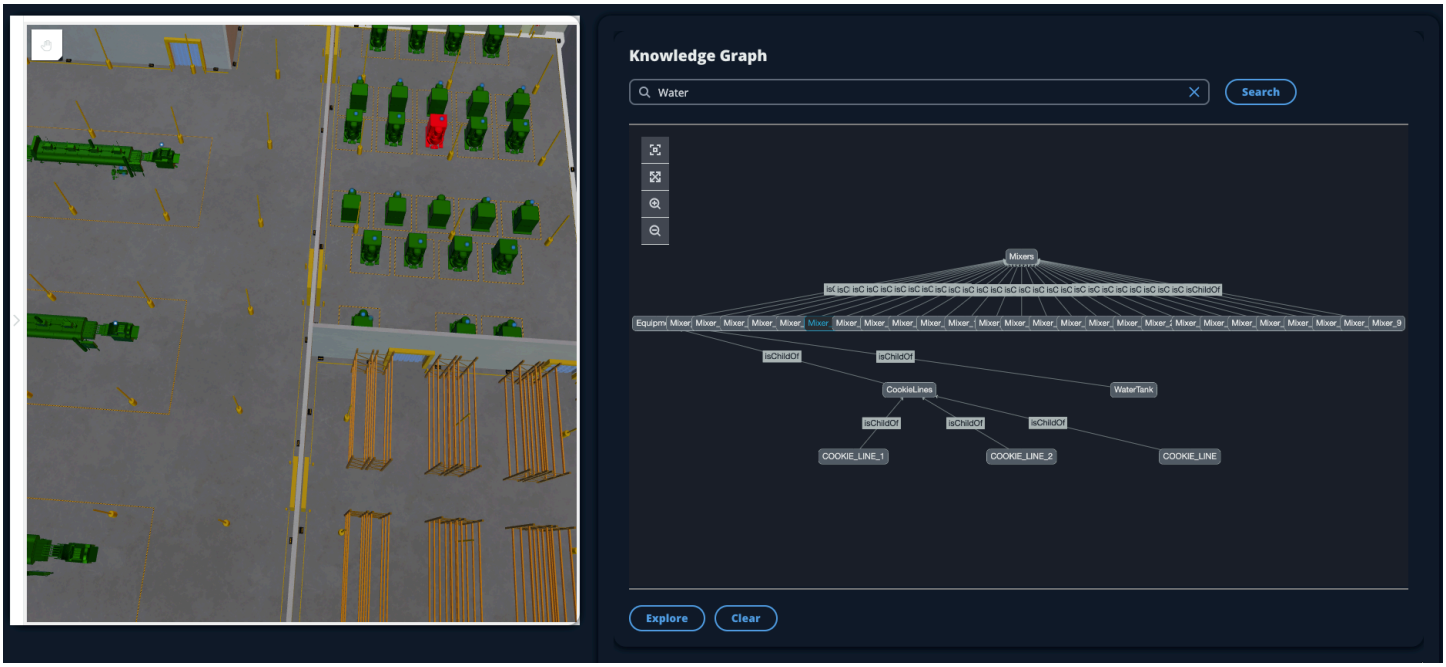
4. クエリの結果もリストに表示されます。[Results] を選択すると、クエリ結果がリストに表示されます。
5. オプションで [名前を付けてエクスポート] を選択し、クエリ結果を JSON または CSV 形式でエクスポートします。

コンソールでのナレッジグラフの基本的な使用方法を説明します。ナレッジグラフ構文の詳細と例については、「AWS IoT TwinMaker ナレッジグラフの追加リソース」を参照してください。

ナレッジグラフシーンの統合

AWS IoT アプリキットのコンポーネントを使用して、 AWS IoT TwinMaker ナレッジグラフをシーンに統合するウェブアプリケーションを構築できます。これにより、シーン内に存在する 3D ノード (機器やシステムを表す 3D モデル) に基づいてグラフを生成できます。シーンの 3D ノードをグラフ化するアプリケーションを作成するには、まず 3D ノードをワークスペース内のエンティティにバ

インドします。このマッピングでは、シーン内に存在する 3D AWS IoT TwinMaker モデルとワークスペース内のエンティティとの関係をグラフ化します。その後、Web アプリケーションを作成し、シーン内の 3D モデルを選択し、それらのモデルと他のエンティティとの関係をグラフ形式で調べることができます。



AWS IoT AWS IoT TwinMaker アプリキットのコンポーネントを利用してシーンにグラフを生成する動作中のウェブアプリケーションの例については、github AWS IoT TwinMaker のサンプル react アプリを参照してください。

AWS IoT TwinMaker シーングラフの前提条件

AWS IoT TwinMaker シーンでナレッジグラフを使用する Web アプリを作成する前に、以下の前提条件を満たしてください。

- ワークスペースを作成します。 AWS IoT TwinMaker ワークスペースは、AWS IoT TwinMaker コンソールで作成できます。
- AWS IoT TwinMakerのエンティティコンポーネントシステムとエンティティの作成方法に慣れてください。詳細については、「最初のエンティティを作成する」を参照してください。
- 3D AWS IoT TwinMaker モデルを含むシーンを作成します。
- AWS IoT TwinMaker AWS IoT のアプリキットのコンポーネントに慣れてください。 AWS IoT TwinMaker コンポーネントの詳細については、を参照してくださいAWS IoT TwinMaker UI コンポーネントを使用してカスタマイズされたウェブアプリケーションを作成する。

- ナレッジグラフの概念と主要な用語に慣れてください。AWS IoT TwinMaker ナレッジグラフのコアコンセプト を参照してください。

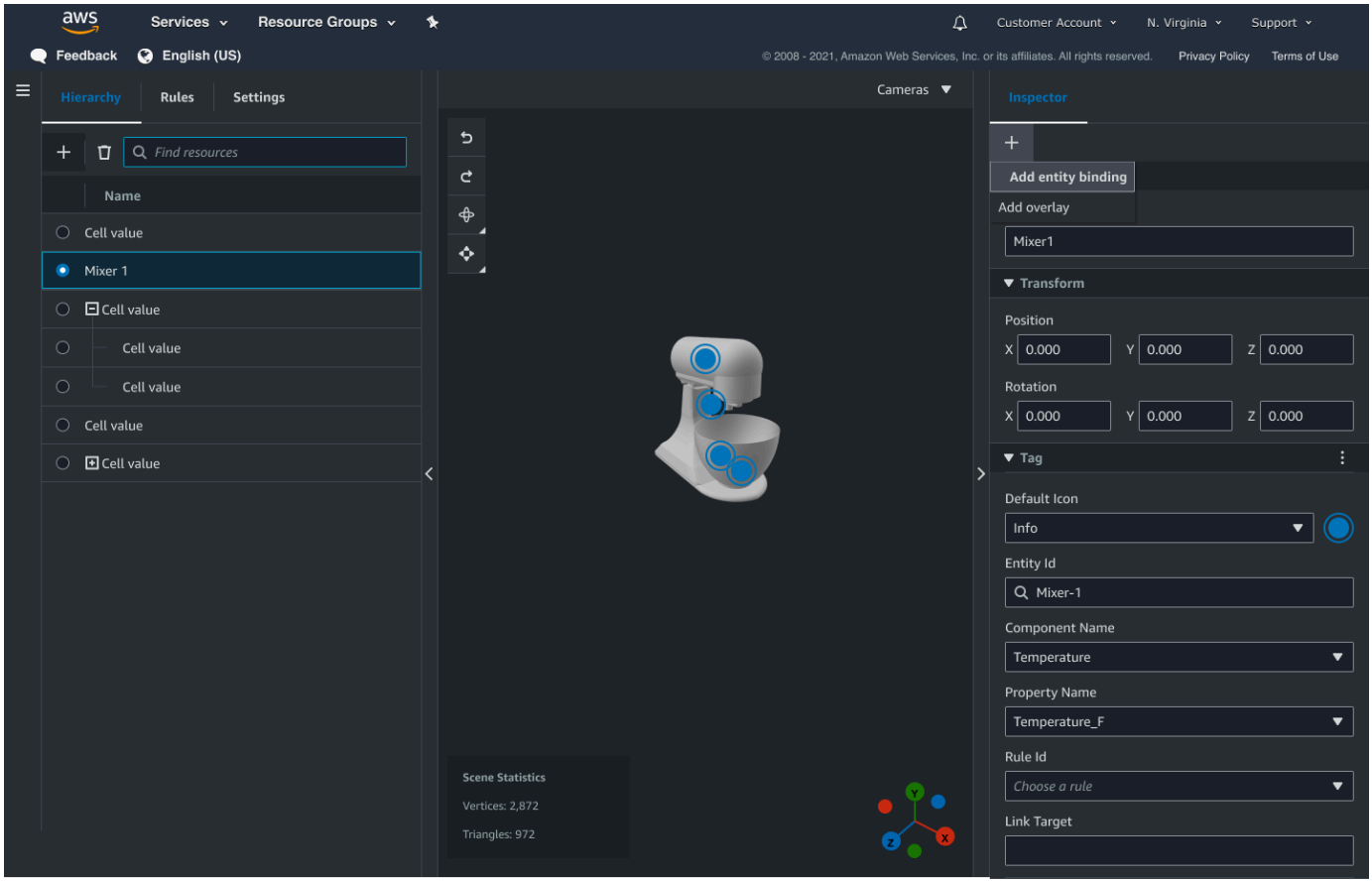
> **Note**
>
> AWS IoT TwinMaker ナレッジグラフと関連機能を使用するには、標準バンドル価格設定モードまたは階層型バンドル価格設定モードのいずれかに設定する必要があります。 AWS IoT TwinMaker 料金の詳細については、を参照してくださいAWS IoT TwinMaker 価格設定モードの切り替え。

シーンで 3D ノードをバインドする

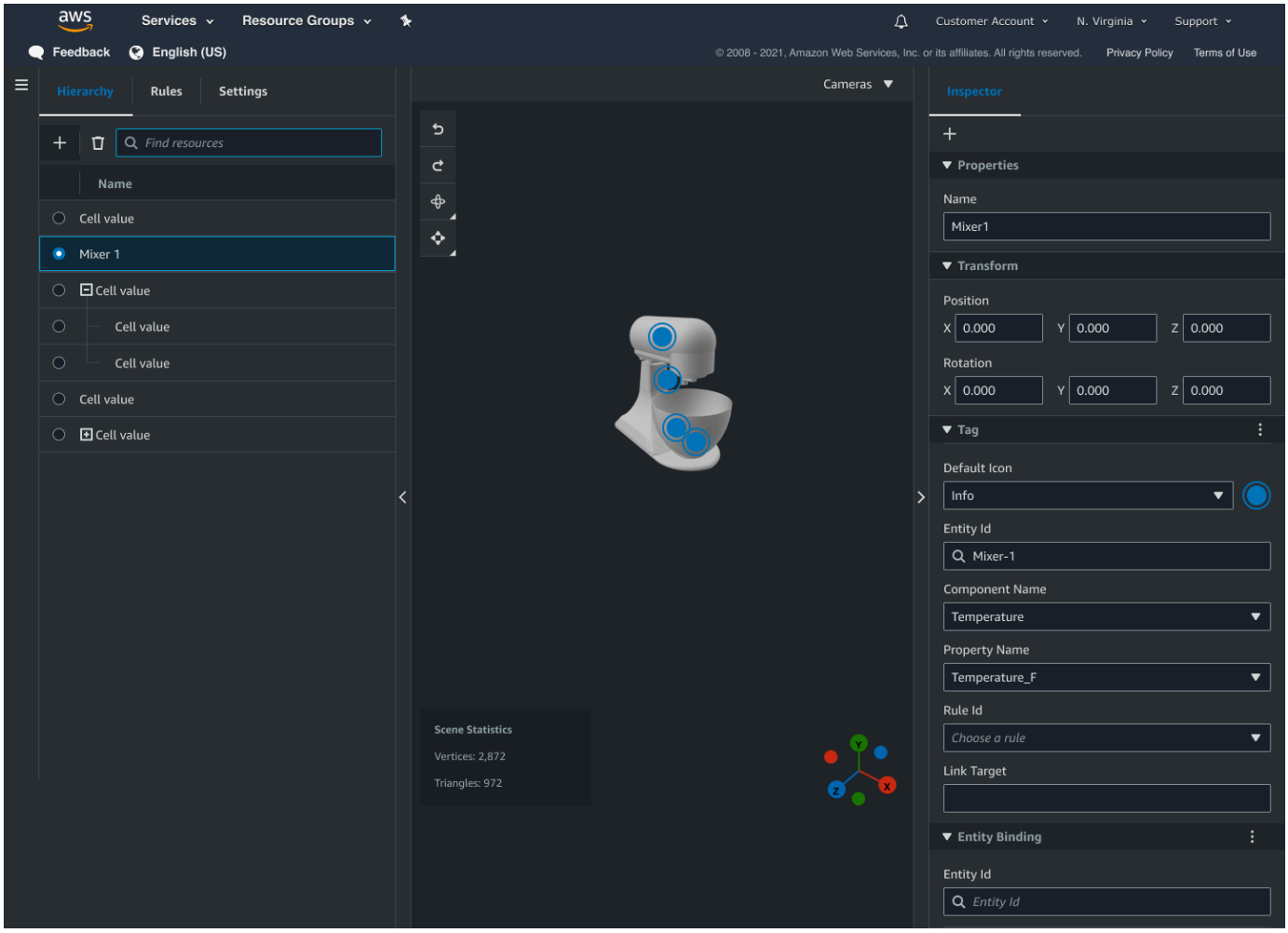
ナレッジグラフをシーンと統合する Web アプリを作成する前に、シーン内に存在する 3D モデル (3D ノードと呼ばれる) を関連するワークスペースエンティティにバインドします。たとえば、シーン内にミキサー機器のモデルがあり、対応するエンティティという名前がある場合`mixer_0`、ミキサーのモデルとミキサーを表すエンティティとの間にデータバインディングを作成して、モデルとエンティティをグラフ化できるようにします。

データバインディングアクションを実行するには

1. AWS IoT TwinMaker コンソールにログインします。
2. ワークスペースを開き、バインドしたい 3D ノードを含むシーンを選択します。
3. シーンコンポーザーでノード (3D モデル) を選択します。ノードを選択すると、画面の右側にインスペクターパネルが開きます。
4. インスペクターパネルで、パネルの上部に移動し、+ ボタンを選択します。次に、「エンティティバインディングを追加」オプションを選択します。ドロップダウンが開き、現在選択しているノードにバインドするエンティティを選択できます。



5. データバインディングのドロップダウンメニューから、3D モデルにマップするエンティティ ID を選択します。「コンポーネント名」と「プロパティ名」フィールドで、バインドするコンポーネントとプロパティを選択します。



「エンティティ ID」、「コンポーネント名」、「プロパティ名」 の各フィールドを選択したら、バインドは完了です。

6. グラフ化したいすべてのモデルとエンティティに対してこのプロセスを繰り返します。

> **Note**
>
> シーンタグでも同じデータバインディング操作を実行できます。エンティティの代わりにタグを選択し、同じ手順でタグをノードにバインドします。

ウェブアプリケーションを作成

エンティティをバインドしたら、 AWS IoT アプリキットライブラリを使用して、シーンを表示したり、シーンノードとエンティティの関係を調べたりできるナレッジグラフウィジェットを備えた Web アプリを構築します。

以下のリソースを使用して独自のアプリを作成します。

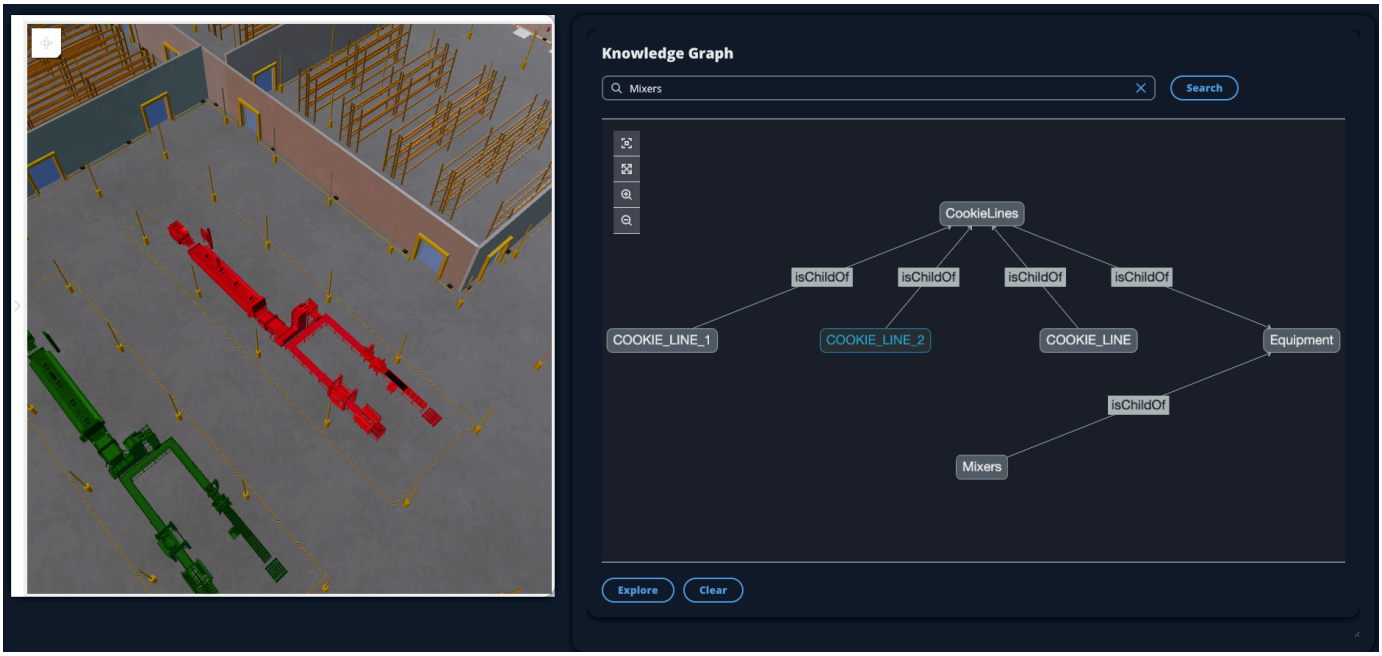
- React AWS IoT TwinMaker アプリケーションのサンプル github Readme ドキュメンテーション。
- github にある react AWS IoT TwinMaker アプリのサンプルソースです。
- AWS IoT アプリキット入門ドキュメント。
- AWS IoT アプリキットのビデオプレーヤーコンポーネントのドキュメンテーション。
- AWS IoT アプリキットのシーンビューアーコンポーネントのドキュメンテーション。

以下の手順は、Web アプリのシーンビューアーコンポーネントの機能を示しています。

> **Note**
>
> この手順は、 AWS IoT TwinMaker サンプル React AWS IoT アプリの App Kit シーンビューアーコンポーネントの実装に基づいています。

1. AWS IoT TwinMaker サンプル React アプリのシーンビューアーコンポーネントを開きます。検索フィールドにエンティティ名またはエンティティ名の一部 (大文字と小文字を区別して検索) を入力し、[検索] ボタンを選択します。モデルがエンティティ ID にバインドされている場合、シーン内のモデルが強調表示され、エンティティのノードがシーンビューアーパネルに表示されます。

2. すべてのリレーションシップのグラフを生成するには、シーンビューアーウィジェットでノードを選択し、Explore ボタンを選択します。



3. クリアボタンを押すと、現在のグラフ選択がクリアされ、最初からやり直すことができます。

Grafana AWS IoT TwinMaker でナレッジグラフを使用する方法

このセクションでは、 AWS IoT TwinMaker Grafana ダッシュボードにクエリエディターパネルを追加してクエリを実行および表示する方法を説明します。

AWS IoT TwinMaker クエリエディターの前提条件

Grafana AWS IoT TwinMaker でナレッジグラフを使用する前に、以下の前提条件を満たしてください。

- AWS IoT TwinMaker ワークスペースを作成します。ワークスペースは、AWS IoT TwinMaker コンソールで作成できます。
- Grafana AWS IoT TwinMaker で使用するように設定します。手順については、「AWS IoT TwinMaker Grafana ダッシュボードの統合」を参照してください。

> **Note**
>
> AWS IoT TwinMaker Knowledge Graph を使用するには、標準または段階的なバンドル価格設定モードになっている必要があります。詳細については、「AWS IoT TwinMaker 価格設定モードの切り替え」を参照してください。

AWS IoT TwinMaker クエリ編集者の権限

Grafana AWS IoT TwinMaker のクエリエディターを使用するには、アクションの権限を持つ IAM ロールが必要です。`iottwinmaker:ExecuteQuery`この例のように、その権限をワークスペースダッシュボードロールに追加します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "{s3Arn}",
                "{s3Arn}/"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iottwinmaker:Get",
                "iottwinmaker:List",
                "iottwinmaker:ExecuteQuery"
            ],
            "Resource": [
                "{workspaceArn}",
                "{workspaceArn}/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iottwinmaker:ListWorkspaces",
            "Resource": "*"
```

```
        }
    ]
}
```

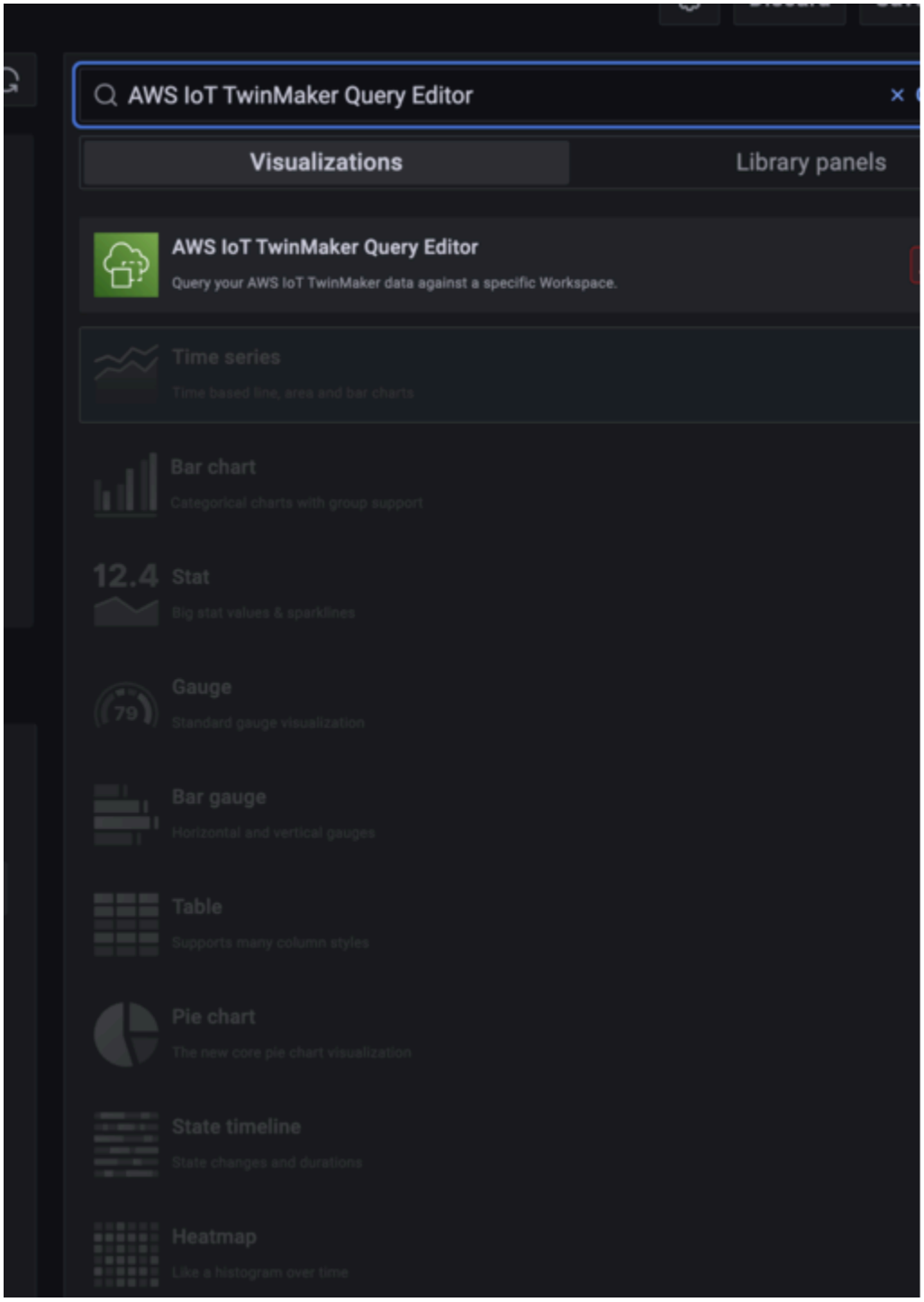
> **Note**
>
> AWS IoT TwinMaker Grafana データソースを設定するときは、必ず Assume role ARN フィールドにこの権限を持つロールを使用してください。追加したら、ワークスペースの横にあるドロップダウンからワークスペースを選択できます。

詳細については、「ダッシュボード IAM ロールの作成」を参照してください。

クエリエディターパネルを設定します。 AWS IoT TwinMaker

ナレッジグラフ用の新しい Grafana ダッシュボードパネルを設定するには

1. AWS IoT TwinMaker Grafana ダッシュボードを開きます。
2. 新しいダッシュボードパネルを作成します。パネルの作成方法の詳細な手順については、Grafana ドキュメントの「ダッシュボードの作成」を参照してください。
3. ビジュアライゼーションのリストから [クエリエディター] を選択します。AWS IoT TwinMaker



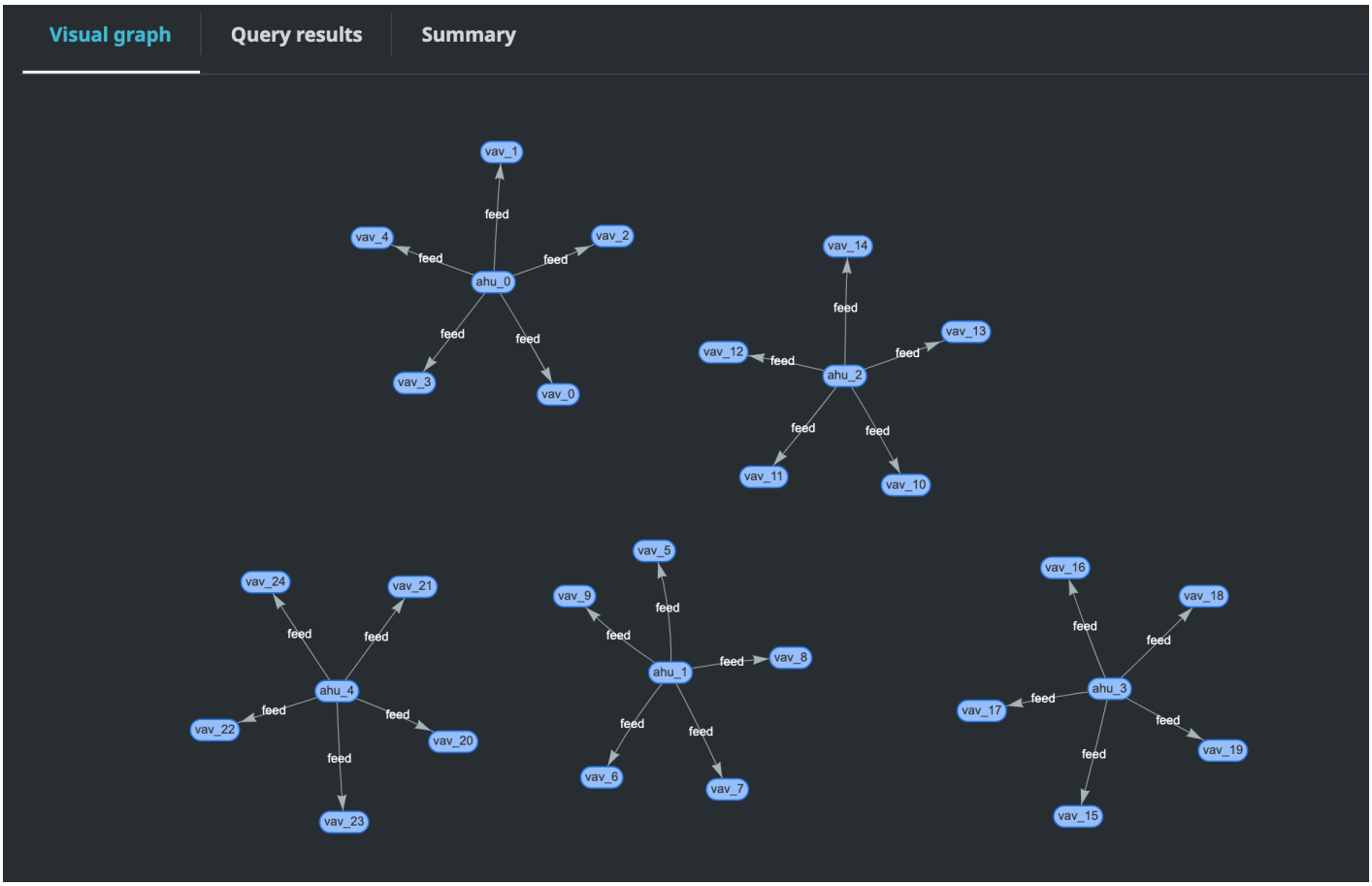
4. クエリを実行するデータソースを選択します。
5. (オプション) 表示されたフィールドに新しいパネルの名前を追加します。
6. [適用] を選択して保存し、新しいパネルを確定します。

ナレッジグラフパネルは、 AWS IoT TwinMaker コンソールに用意されているクエリエディターと同様に機能します。パネルで作成したクエリを実行、記述、クリアできます。クエリの記述方法の詳細については、を参照してくださいAWS IoT TwinMaker ナレッジグラフの追加リソース。

AWS IoT TwinMaker クエリエディターの使用方法

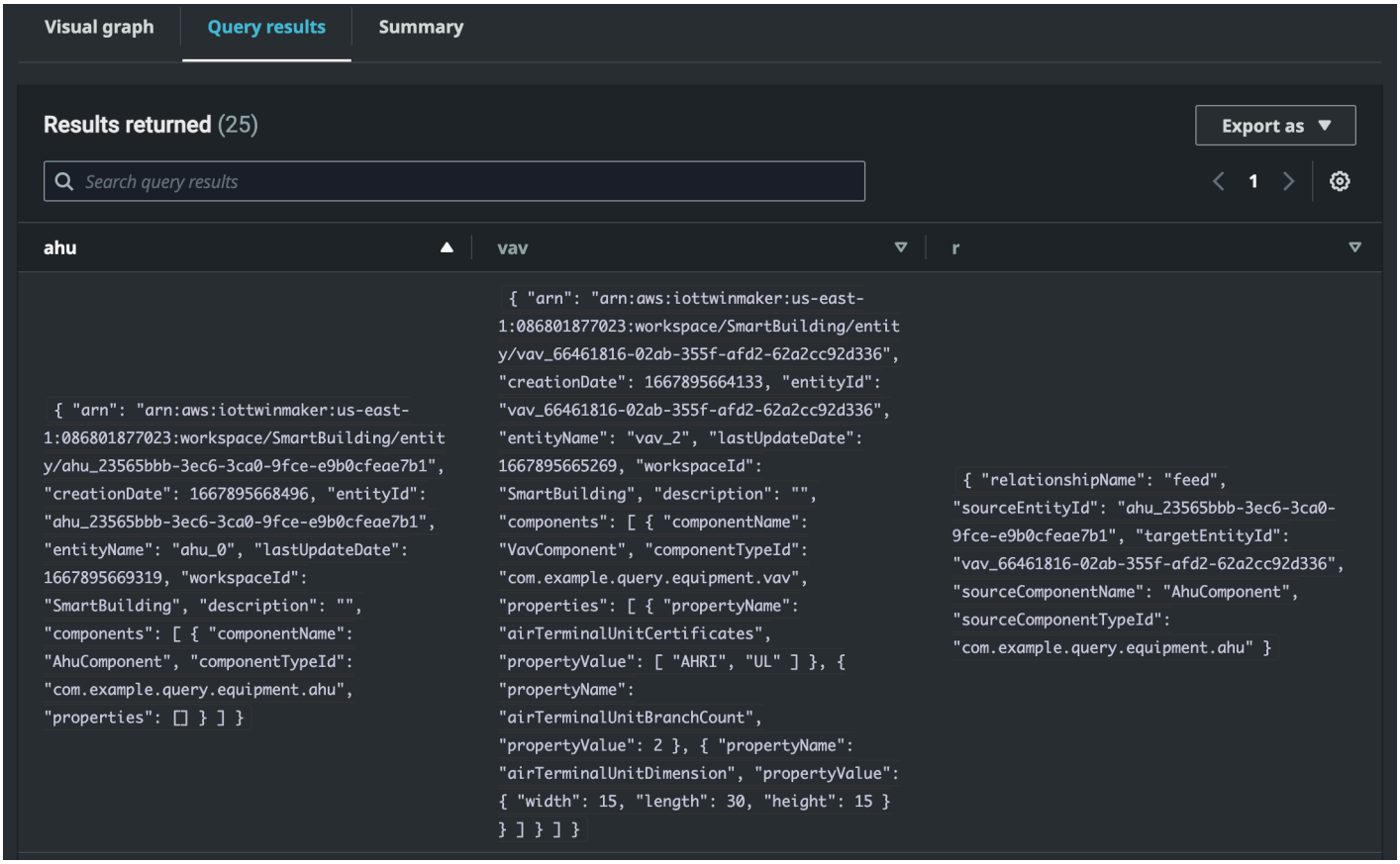
クエリの結果は、次の画像の通り、グラフで視覚化、表で一覧表示、および実行サマリーとして表示、の 3 つの方法で表示されます。

- グラフの視覚化



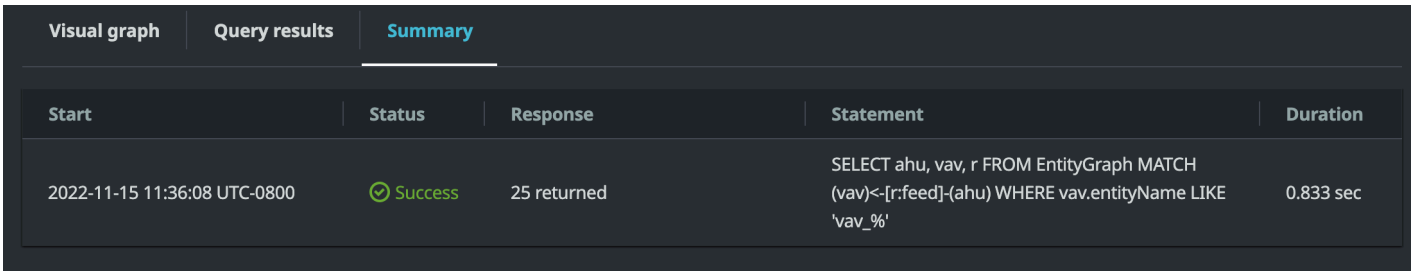
ビジュアルグラフには、結果に少なくとも 1 つのリレーションを含むクエリのデータのみが表示されます。グラフでは、エンティティはノードとして、リレーションシップはグラフ内の有向エッジとして表示されます。

- 表形式のデータ:



表形式のデータには、すべてのクエリのデータ表示されます。テーブルから特定の結果または結果のサブセットを検索できます。データは JSON または CSV 形式でエクスポートできます。

- 実行サマリー



実行サマリーには、クエリとクエリのステータスに関するメタデータが表示されます。

AWS IoT TwinMaker ナレッジグラフの追加リソース

このセクションでは、ナレッジグラフにクエリを記述するために使用されるPartiQL構文の基本的な例と、ナレッジグラフデータモデルに関する情報を提供するPartiQLドキュメントへのリンクを示します。

- PartiQL グラフデータモデルドキュメント
- PartiQL グラフクエリのドキュメント

この一連の例は、基本的なクエリとその応答を示しています。これを参考にして、独自のクエリを作成してください。

基本的なクエリ

- フィルターを使用してすべてのエンティティを取得

```
SELECT entity
FROM EntityGraph MATCH (entity)
WHERE entity.entityName = 'room_0'
```

このクエリは、ワークスペース内の同じ名前のエンティティをすべて返しますroom_0。

FROMclause: EntityGraph は、ワークスペース内のすべてのエンティティとその関係を含むグラフコレクションです。このコレクションは、 AWS IoT TwinMaker ワークスペース内のエンティティに基づいて自動的に作成および管理されます。

MATCH 句: グラフの一部と一致するパターンを指定します。この場合、パターン (entity) はグラフ内のすべてのノードと一致し、エンティティ変数にバインドされます。FROM 句の後には MATCH 句が続く必要があります。

WHEREclause: entityName room_0 値が一致する必要があるノードのフィールドにフィルターを指定します。

SELECTclause: entity エンティティノード全体が返されるように変数を指定します。

レスポンス :

```
{
  "columnDescriptions": [
    {
      "name": "entity",
      "type": "NODE"
    }
  ],
  "rows": [
    {
      "rowData": [
```

```
        {
          "arn": "arn:aws:iottwinmaker:us-east-1: 577476956029: workspace /
 SmartBuilding8292022 / entity / room_18f3ef90 - 7197 - 53 d1 - abab -
 db9c9ad02781 ",
          "creationDate": 1661811123914,
          "entityId": "room_18f3ef90-7197-53d1-abab-db9c9ad02781",
          "entityName": "room_0",
          "lastUpdateDate": 1661811125072,
          "workspaceId": "SmartBuilding8292022",
          "description": "",
          "components": [
            {
              "componentName": "RoomComponent",
              "componentTypeId": "com.example.query.construction.room",
              "properties": [
                {
                  "propertyName": "roomFunction",
                  "propertyValue": "meeting"
                },
                {
                  "propertyName": "roomNumber",
                  "propertyValue": 0
                }
              ]
            }
          ]
        }
      ]
    }
  ]
}
```

は、名前やタイプなど、columnDescriptions列に関するメタデータを返します。返されるタイプは、NODE です。これはノード全体が返されたことを示しています。この型の他の値には、関係を示す値やVALUE、整数や文字列などのスカラー値などがあります。EDGE

rows は行のリストを返します。一致したエンティティは 1 つだけなので、エンティティのすべてのフィールドを含む 1 つの rowData が返されます。

> **Note**
>
> スカラー値しか返せない SQL とは異なり、PartiQL を使用してオブジェクトを (JSON として) 返すことができます。

各ノードには、、などのエンティティレベルのフィールドentityId、、などのコンポーネントレベルのフィールドcomponents、arnおよびなどのプロパティレベルのフィールドがすべてネストされた JSON として格納されますpropertyName。componentName componentTypeId properties propertyValue

- すべてのリレーションシップをフィルターで取得:

```
SELECT relationship
FROM EntityGraph MATCH (e1)-[relationship]->(e2)
WHERE relationship.relationshipName = 'isLocationOf'
```

このクエリは、ワークスペース内のすべてのリレーションシップをリレーション名 isLocationOf で返します。

MATCHこの句:ディレクテッド・エッジ (で示される) で接続され、という変数にバインドされている 2 つのノード (で示される ( )) に一致するパターンを指定します。-[]-> relationship

WHERE句:relationshipName値がのエッジのフィールドにフィルターを指定します。isLocationOf

SELECT 句: エッジノード全体が返されるようリレーションシップ変数を指定します。

レスポンス

```
{
    "columnDescriptions": [{
        "name": "relationship",
        "type": "EDGE"
    }],
    "rows": [{
        "rowData": [{
            "relationshipName": "isLocationOf",
            "sourceEntityId": "floor_83faea7a-ea3b-56b7-8e22-562f0cf90c5a",
            "targetEntityId": "building_4ec7f9e9-e67e-543f-9d1b- 235df7e3f6a8",
```

```
            "sourceComponentName": "FloorComponent",
            "sourceComponentTypeId": "com.example.query.construction.floor"
        }]
    },
        ... //rest of the rows are omitted
    ]
}
```

columnDescriptions内のカラムのタイプはEDGE.

それぞれが、rowDataのようなフィールドを持つエッジを表しますrelationshipName。これはエンティティに定義されているリレーションシッププロパティ名と同じです。sourceEntityId、、には、sourceComponentNameリレーションシッププロパティが定義されているエンティティとコンポーネントに関する情報が表示されます。sourceComponentTypeIdは、targetEntityIdこのリレーションシップがどのエンティティを指しているかを指定します。

- 特定のエンティティと特定の関係を持つすべてのエンティティを取得します。

```
SELECT e2.entityName
      FROM EntityGraph MATCH (e1)-[r]->(e2)
      WHERE relationship.relationshipName = 'isLocationOf'
      AND e1.entityName = 'room_0'
```

このクエリは、isLocationOfroom_0そのエンティティと関係があるすべてのエンティティのすべてのエンティティ名を返します。

MATCH句:ディレクテッドエッジ () を持つ任意の 2 つのノード (e1,e2) と一致するパターンを指定します。r

WHERE 句: リレーションシップ名とソースエンティティ名のフィルターを指定します。

SELECTこの句:entityNamee2ノード内のフィールドを返します。

レスポンス

```
{
  "columnDescriptions": [
    {
       "name": "entityName",
       "type": "VALUE"
```

```
    }
  ],
   "rows": [
    {
       "rowData": [
         "floor_0"
      ]
    }
  ]
}
```

ColumnDescriptionsでは、VALUEカラムのタイプは文字列なのでentityName。

1 つのエンティティfloor_0、が返されます。

MATCH

1 MATCH つの節では以下のパターンがサポートされています。

- ノード 'b' がノード 'a' を指しているものと一致させます。

```
FROM EntityGraph MATCH (a)-[rel]-(b)
```

- ノード 'a' がノード 'b' を指しているものと一致させます。

```
FROM EntityGraph MATCH (a)-[]->(b)
```

リレーションシップにフィルターを指定する必要がないと仮定すると、リレーションシップにバインドされた変数はありません。

- ノード 'a' がノード 'b' を指し、ノード 'b' がノード 'a' を指しているものと一致させます。

```
FROM EntityGraph MATCH (a)-[rel]-(b)
```

これによって 2 つのマッチが返されます。1 つは 'a' から 'b'、もう 1 つは 'b' から 'a' なので、可能な限り有向エッジを使用することをおすすめします。

- リレーションシップ名はプロパティグラフのラベルでもあるためEntityGraph、rel.relationshipNameWHERE句内でフィルターを指定する代わりに、コロン (:) の後にリレーションシップ名を指定するだけで済みます。

```
FROM EntityGraph MATCH (a)-[:isLocationOf]-(b)
```

- チェーン: 複数のリレーションシップに一致するようにパターンを連鎖させることができます。

```
FROM EntityGraph MATCH (a)-[rel1]->(b)-[rel2]-(c)
```

- 変数ホップパターンは、複数のノードや エッジにまたがっていることもあります。

```
FROM EntityGraph MATCH (a)-[]->{1,5}(b)
```

このクエリは、ノード 'a' からの出力エッジが 1 ~ 5 ホップ以内の任意のパターンにマッチします。指定できる格量指定子は次のとおりです。

{m,n} - m 回から n 回の間の繰り返し

{m,} - m 回以上の繰り返し。

FROM:

エンティティノードには、コンポーネントなどのネストされたデータを含めることができ、そのデータ自体にはプロパティなどのさらにネストされたデータが含まれます。これらは、MATCH パターンの結果をネスト解除することでアクセスできます。

```
SELECT e
FROM EntityGraph MATCH (e), e.components AS c, c.properties AS p
WHERE c.componentTypeId = 'com.example.query.construction.room',
AND p.propertyName = 'roomFunction'
AND p.propertyValue = 'meeting'
```

ネストされたフィールドにアクセスするには、変数を . ドットで囲みます。コンマ (,) を使用して、内部のコンポーネントを含むエンティティをネスト解除 (または結合) し、次にそのコンポーネント内のプロパティをネスト解除 (または結合) します。 AS変数をネストされていない変数にバインドして、or 句で使用できるようにするために使用されます。WHERE SELECTこのクエリは、コンポーネントタイプ ID com.example.query.construction.room のコンポーネント内の値 meeting と、roomFunction という名前のプロパティを含むすべてのエンティティを返します。

エンティティ内の複数のコンポーネントなど、1 つのフィールドの複数のネストされたフィールドにアクセスするには、カンマ表記を使用して結合を行います。

```
SELECT e
```

```
FROM EntityGraph MATCH (e), e.components AS c1, e.components AS c2
```

SELECT:

- ノードを返す:

```
SELECT e
FROM EntityGraph MATCH (e)
```

- エッジを返す:

```
SELECT r
FROM EntityGraph MATCH (e1)-[r]->(e2)
```

- スカラー値を返す:

```
SELECT floor.entityName, room.description, p.propertyValue AS roomfunction
FROM EntityGraph MATCH (floor)-[:isLocationOf]-(room),
room.components AS c, c.properties AS p
```

  AS を使用してエイリアシングで出力フィールドの名前をフォーマットします。ここでは、レスポンス内の列名の propertyValue の代わりに、roomfunction が返されます。

- エイリアスを返す:

```
SELECT floor.entityName AS floorName, luminaire.entityName as luminaireName
FROM EntityGraph MATCH (floor)-[:isLocationOf]-(room)-[:hasPart]-
(lightingZone)-[:feed]-(luminaire)
WHERE floor.entityName = 'floor_0'
AND luminaire.entityName like 'lumin%'
```

  明示的に指定し、読みやすくし、クエリ内のあいまいさを避けるために、エイリアスを使用することを強くお勧めします。

WHERE:

- サポートされている論理演算子は、、、およびです。AND NOT OR
- サポートされている比較演算子は、<、<=、>、=>、= および != です。
- OR同じフィールドに複数の条件を指定する場合は、INキーワードを使用してください。
- エンティティ、コンポーネント、またはプロパティフィールドで絞り込みます。

```
FROM EntityGraph MATCH (e), e.components AS c, c.properties AS p
```

```
WHERE e.entityName = 'room_0'
AND c.componentTypeId = 'com.example.query.construction.room',
AND p.propertyName = 'roomFunction'
AND NOT p.propertyValue = 'meeting'
OR p.propertyValue = 'office'
```

- configurationプロパティを絞り込む。unitCelsiusこれが設定マップのキーと値です。

```
WHERE p.definition.configuration.unit = 'Celsius'
```

- マッププロパティに指定されたキーと値が含まれているかどうかを確認します。

```
WHERE p.propertyValue.length = 20.0
```

- マッププロパティに指定されたキーが含まれているかどうかを確認します。

```
WHERE NOT p.propertyValue.length IS MISSING
```

- リストプロパティに指定された値が含まれているかどうかを確認します。

```
WHERE 10.0 IN p.propertyValue
```

- 大文字と小文字を区別しない比較にはこの lower( ) 関数を使用します。デフォルトでは、大文字と小文字を区別した比較が使用されます。

```
WHERE lower(p.propertyValue) = 'meeting'
```

LIKE:

フィールドの正確な値がわからず、指定したフィールドで全文検索を実行できる場合に便利です。% はゼロ以上を表します。

```
WHERE e.entityName LIKE '%room%'
```

- インフィックス検索: %room%
- プレフィックス検索: room%
- サフィックス検索: %room
- 値に '%' が含まれている場合は、にエスケープ文字を入力し、エスケープ文字をで指定しますESCAPE。LIKE

```
WHERE e.entityName LIKE 'room\%' ESCAPE '\'
```

DISTINCT:

```
SELECT DISTINCT c.componentTypeId
FROM EntityGraph MATCH (e), e.components AS c
```

- DISTINCT キーワードは、最終結果から重複を排除します。

  DISTINCT は複雑なデータ型ではサポートされていません。

COUNT

```
SELECT COUNT(e), COUNT(c.componentTypeId)
FROM EntityGraph MATCH (e), e.components AS c
```

- COUNTこのキーワードは、クエリ結果の項目数を計算します。
- COUNTネストされた複合フィールドやグラフパターンフィールドではサポートされていません。
- COUNTDISTINCTおよびネストされたクエリでは集計はサポートされていません。

  たとえば、COUNT(DISTINCT e.entityId) はサポートされません。

パス

パスプロジェクションを使ったクエリでは、以下のパターンプロジェクションがサポートされています。

- 可変ホップクエリ

  ```
  SELECT p FROM EntityGraph MATCH p = (a)-[]->{1, 3}(b)
  ```

  このクエリは、ノード a からの出力エッジが 1 ~ 3 ホップ以内の任意のパターンのノードメタデータを照合して投影します。

- 固定ホップクエリ

  ```
  SELECT p FROM EntityGraph MATCH p = (a)-[]->(b)<-[]-(c)
  ```

  このクエリは、エンティティと入力エッジのメタデータを照合して b に投影します。

- 無向クエリ

```
SELECT p FROM EntityGraph MATCH p = (a)-[]-(b)-[]-(c)
```

このクエリは、a と c を b 経由で接続する 1 ホップパターンでノードのメタデータを照合して投影します。

```
{
    "columnDescriptions": [
        {
            "name": "path",
            "type": "PATH"
        }
    ],
    "rows": [
        {
            "rowData": [
                {
                    "path": [
                        {
                            "entityId": "a",
                            "entityName": "a"
                        },
                        {
                            "relationshipName": "a-to-b-relation",
                            "sourceEntityId": "a",
                            "targetEntityId": "b"
                        },
                        {
                            "entityId": "b",
                            "entityName": "b"
                        }
                    ]
                }
            ]
        },
        {
            "rowData": [
                {
                    "path": [
                        {
                            "entityId": "b",
                            "entityName": "b"
                        },
```

```
                        {
                            "relationshipName": "b-to-c-relation",
                            "sourceEntityId": "b",
                            "targetEntityId": "c"
                        },
                        {
                            "entityId": "c",
                            "entityName": "c"
                        }
                    ]
                }
            ]
        }
    ]
}
```

PATHこのクエリレスポンスは、a と c の間の b 経由の各パス/パターンのすべてのノードとエッジを識別するメタデータのみで構成されます。

リミットとオフセット:

```
SELECT e.entityName
FROM EntityGraph MATCH (e)
WHERE e.entityName LIKE 'room_%'
LIMIT 10
OFFSET 5
```

LIMIT はクエリで返される結果の数を指定し、OFFSET はスキップする結果の数を指定します。

結果の上限値と最大値:

次の例は、合計 500 件の結果を返すクエリを示していますが、1 回に API 呼び出しごとに 50 件しか表示できません。このパターンは、UI に 50 件の結果しか表示できない場合など、表示される結果の量を制限する必要がある場合に使用できます。

```
aws iottwinmaker execute-query \
--workspace-id exampleWorkspace \
--query-statement "SELECT e FROM EntityGraph MATCH (e) LIMIT 500"\
--max-results 50
```

- LIMITキーワードはクエリに影響し、結果の行を制限します。返される結果の総数を制限せずに API 呼び出しごとに返される結果の数を制御する必要がある場合は、を使用してくださいLIMIT。

- `max-results`ExecuteQuery API アクションのオプションパラメータです。 `max-results`API と、上記のクエリの範囲内での結果の読み取り方法にのみ適用されます。

  `max-results`クエリで使用すると、返される結果の実際の数を制限することなく、表示される結果の数を減らすことができます。

以下のクエリは、結果の次のページまで繰り返し処理されます。このクエリは ExecuteQuery API 呼び出しを使用して行 51 ~ 100 を返します。結果の次のページは、で指定されます。この場合、トークンは:です。`next-token` "H7kyGmvK376L"

```
aws iottwinmaker execute-query \
--workspace-id exampleWorkspace \
--query-statement "SELECT e FROM EntityGraph MATCH (e) LIMIT 500"\
--max-results 50
--next-token "H7kyGmvK376L"
```

- `next-token`この文字列は、結果の次のページを指定します。詳細については、ExecuteQueryAPI アクションを参照してください。

AWS IoT TwinMaker ナレッジグラフクエリには以下の制限があります。

制限 の名前	クォータ	調整可能
クエリ実行タイムアウト	10 秒	いいえ
ホップの最大数	10	[Yes (はい)]
セルフの最大数JOIN:%s	20	はい
投影フィールドの最大数	20	はい
条件式の最大数 (AND,OR,NOT)	10	[Yes (はい)]
LIKEエクスプレッションパターンの最大長 (ワイルドカードとエスケープを含む)	20	はい
1 つの句に指定できる項目の最大数 IN	10	[Yes (はい)]

制限 の名前	クォータ	調整可能
の最大値:OFFSET	3000	はい
の最大値:LIMIT	3000	はい
トラバーサルの最大値 (+) OFFSET LIMIT	3000	はい

AWS IoT SiteWiseとのアセット同期

AWS IoT TwinMaker アセットとアセットモデルのアセット同期 (アセット同期) をサポートします。AWS IoT SiteWise アセット同期は、 AWS IoT SiteWise AWS IoT SiteWise コンポーネントタイプを使用して既存のアセットとアセットモデルを取得し、 AWS IoT TwinMaker これらのリソースをエンティティ、コンポーネント、およびコンポーネントタイプに変換します。以下のセクションでは、アセット同期の設定方法と、 AWS IoT SiteWise どのアセットとアセットモデルをワークスペースに同期できるかについて説明します。 AWS IoT TwinMaker

トピック

AWS IoT SiteWiseとのアセット同期の使用

このトピックでは、 AWS IoT SiteWise アセットの同期をオンにして設定する方法を説明します。使用しているワークスペースの種類に応じて、適切な手順に従ってください。

> **Important**
>
> the section called “カスタムワークスペースとデフォルトワークスペースの違い”カスタムワークスペースとデフォルトワークスペースの違いについては、を参照してください。

トピック

カスタムワークスペースを使用する

アセット同期を有効にする前に、次の前提条件を確認します。

前提条件

使用する前に AWS IoT SiteWise、次のことを行っていることを確認してください。

- AWS IoT TwinMaker ワークスペースができました。
- AWS IoT SiteWiseにアセットとアセットモデルをいくつか作成しておきます。モデル作成の詳細については、「アセットモデルの作成」を参照してください。
- 、、ListAssets、 AWS IoT SiteWise ListAssetModelsの各アクションの読み取り権限を持つ IAM ロールが作成されました。DescribeAsset DescribeAssetModel

  IAM ロールには、次の AWS IoT TwinMakerの書き込み許可も必要です: CreateEntity、UpdateEntity、DeleteEntity、CreateComponentType、UpdateComponentTy

  次の IAM ロールを必要なロールのテンプレートとして使用できます。

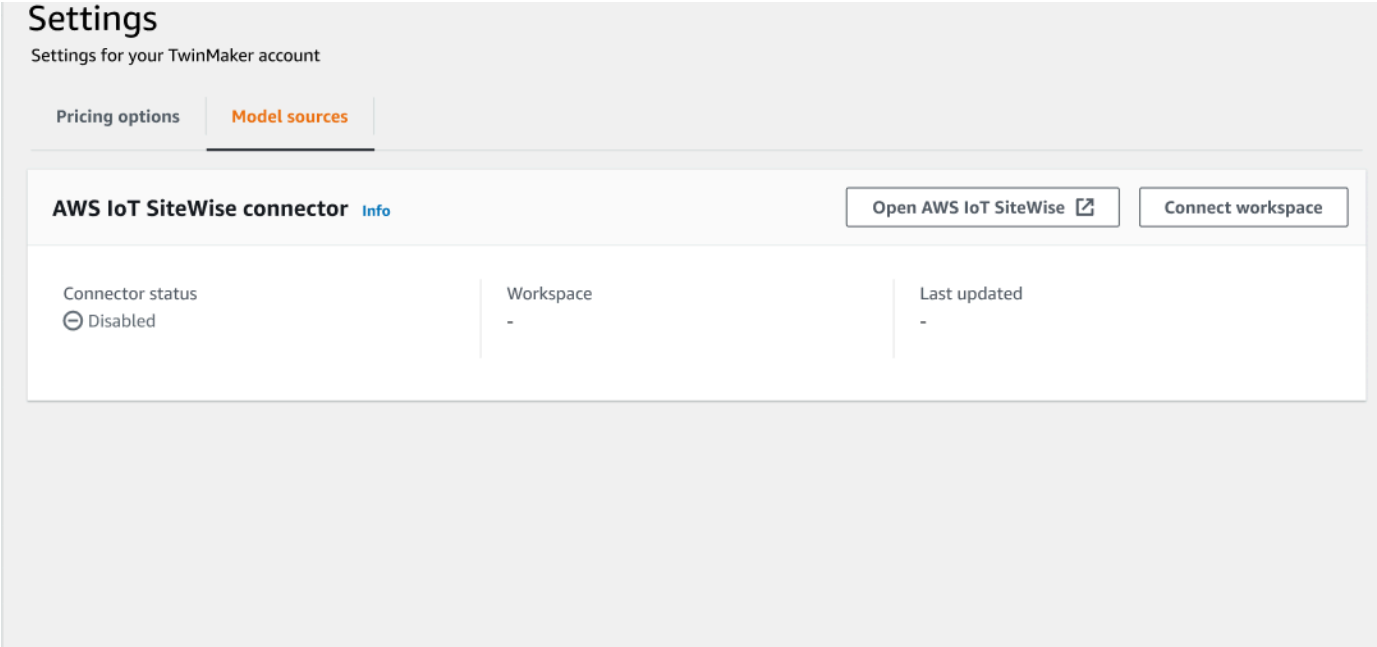
```
// trust relationships
                        {
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                 "Service": [
                    "iottwinmaker.amazonaws.com",
                    "iotsitewise.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}

// permissions
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iottwinmaker:*",
            "Resource": "*"
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "iotsitewise:Describe*",
                "iotsitewise:List*"
            ],
            "Resource": "*"
        }
    ]
}
```

以下の手順を使用して、 AWS IoT SiteWise のアセット同期を有効にします。

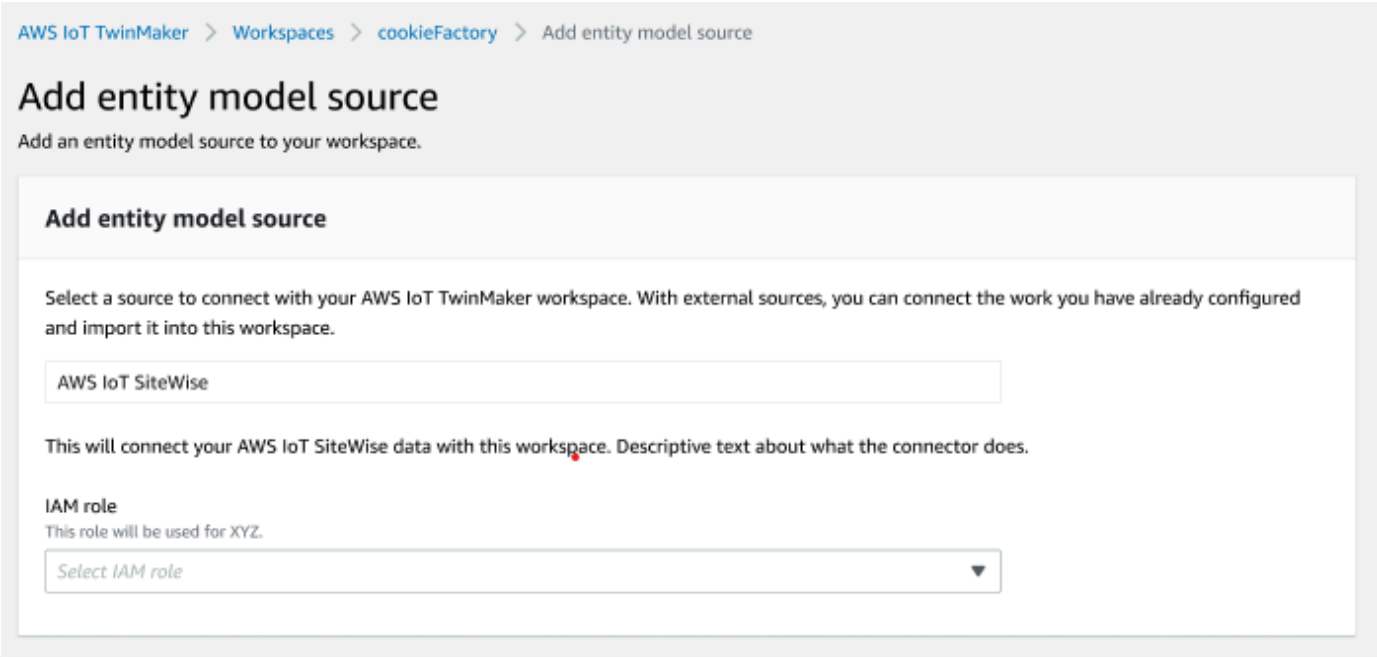
1. 「AWS IoT TwinMaker 」 コンソールで 「設定」 ページに移動します。
2. 「モデルソース」 タブを開きます。



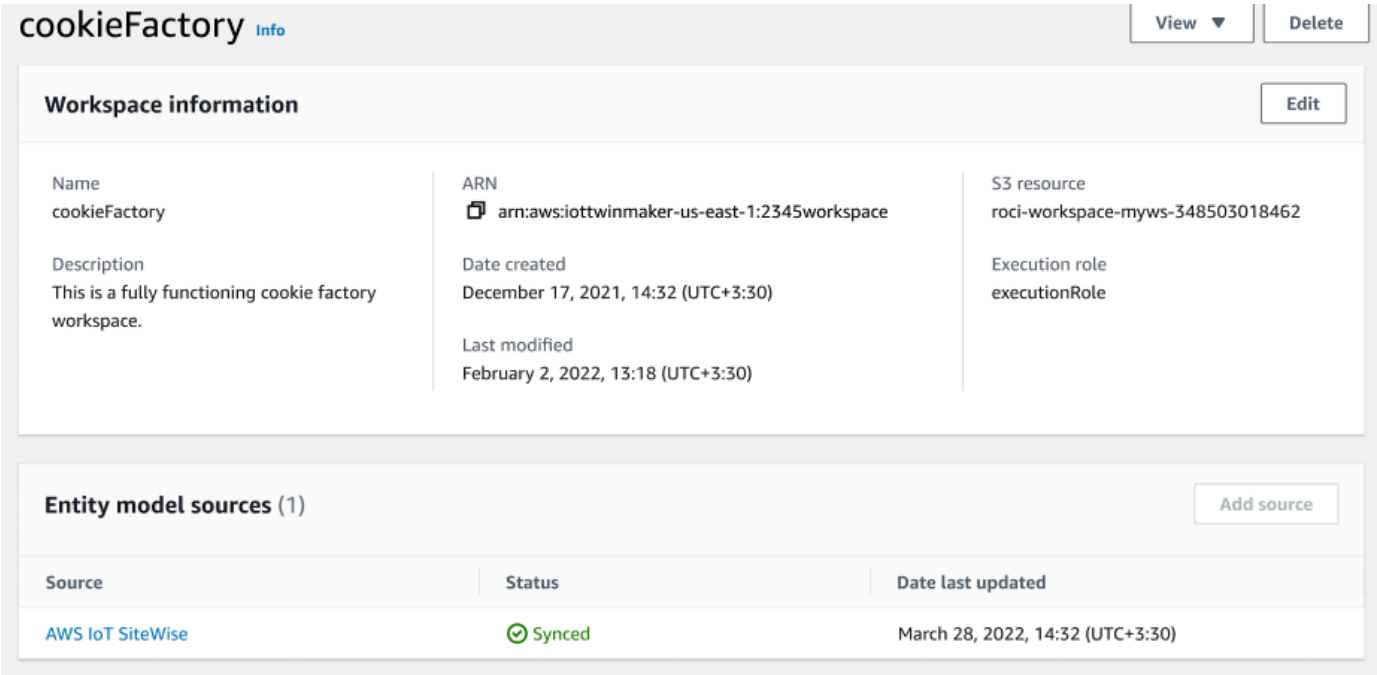
3. 「ワークスペースをConnect」を選択して、 AWS IoT TwinMaker AWS IoT SiteWise ワークスペースをアセットにリンクします。

> **Note**
>
> アセットの同期は 1 AWS IoT TwinMaker つのワークスペースでのみ使用できます。別のワークスペースで同期する場合は、ワークスペースから同期を切断し、別のワークスペースに接続する必要があります。

4. 次に、アセットの同期を使用するワークスペースに移動します。
5. 「ソースを追加」 を選択します。「エンティティモデルソースを追加」 ページが開きます。



6. 「エンティティモデルソースを追加」 ページで、「ソース」 フィールドでAWS IoT SiteWiseが表示されることを確認します。IAM ロールの前提条件として作成した IAM ロールを選択します。
7. これで、 AWS IoT SiteWise アセットの同期が有効になりました。選択した 「ワークスペース」 ページの上部に、アセットの同期がアクティブであることを確認する確認バナーが表示されます。また、「エンティティモデルソース」 セクションに同期ソースが表示されます。



IoT を使用する SiteWiseDefaultWorkspace

AWS IoT SiteWiseAWS IoT TwinMaker インテグレーションにオプトインすると、IoTSiteWiseDefaultWorkspaceという名前のデフォルトワークスペースが作成され、と自動的に同期されます。 AWS IoT SiteWise

AWS IoT TwinMaker CreateWorkspaceAPI IoTSiteWiseDefaultWorkspace を使用してという名前のワークスペースを作成することもできます。

前提条件

作成する前にIoTSiteWiseDefaultWorkspace、次のことを行っていることを確認してください。

- AWS IoT TwinMaker サービスにリンクされたロールを作成します。詳細については、「 のサービスにリンクされたロールの使用 AWS IoT TwinMaker」を参照してください。
- IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。

  ロールまたはユーザーを確認し、権限があることを確認します。iotsitewise:EnableSiteWiseIntegration

  必要に応じて、ロールまたはユーザーに権限を追加します。

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

カスタムワークスペースとデフォルトワークスペースの違い

> **Important**
>
> AWS IoT SiteWise CompositionModelなどの新機能はでのみ使用できます。IoTSiteWiseDefaultWorkspaceカスタムワークスペースではなく、デフォルトのワークスペースを使用することをお勧めします。

を使用する場合IoTSiteWiseDefaultWorkspace、アセット同期機能付きのカスタムワークスペースを使用する場合とはいくつか大きな違いがあります。

- デフォルトワークスペースを作成する場合、Amazon S3 ロケーションと IAM ロールはオプションです。

  > **Note**
  >
  > UpdateWorkspaceを使用して Amazon S3 ロケーションと IAM ロールを指定できます。

- には、IoTSiteWiseDefaultWorkspace AWS IoT SiteWise リソースの同期先となるリソース数の制限はありません。 AWS IoT TwinMaker
- からリソースを同期すると AWS IoT SiteWise、SyncSourceそのリソースも同期されますSITEWISE_MANAGED。Entitiesこれにはとが含まれますComponentTypes。
- AWS IoT SiteWise CompositionModelなどの新機能はにしかありませんIoTSiteWiseDefaultWorkspace。

特有の制限事項は`IoTSiteWiseDefaultWorkspace`、次のとおりです。

- デフォルトのワークスペースは削除できません。
- リソースを削除するには、 AWS IoT SiteWise まずリソースを削除する必要があります。次に、そのリソースに対応するリソースを削除します。 AWS IoT TwinMaker

AWS IoT SiteWiseからの同期のリソース

このトピックでは、 AWS IoT SiteWise AWS IoT TwinMaker どのアセットからワークスペースに同期できるかを一覧表示します。

> **Important**
>
> カスタムワークスペースとデフォルトワークスペースの違いカスタムワークスペースとデフォルトワークスペースの違いについては、を参照してください。

カスタムワークスペースとデフォルトワークスペース

以下のリソースは同期され、カスタムワークスペースとデフォルトワークスペースの両方で利用できます。

アセットモデル

AWS IoT TwinMaker 内のアセットモデルごとに新しいコンポーネントタイプを作成します。 AWS IoT SiteWise

- `TypeId`アセットモデルのコンポーネントは、以下のパターンのいずれかを使用します。
  - カスタムワークスペース- `iotsitewise.assetmodel:`*`assetModelId`*
  - 既定のワークスペース- *`assetModelId`*
- アセットモデル内の各プロパティは、コンポーネントタイプの新しいプロパティで、以下のいずれかの命名パターンがあります。
  - カスタムワークスペース- `Property_`*`propertyId`*
  - 既定のワークスペース- *`propertyId`*

  `displayName`のプロパティ名はプロパティ定義にとして保存されます。 AWS IoT SiteWise

- LISTRELATIONSHIPアセットモデルの各階層は新しいタイプのプロパティで、nestedTypeはコンポーネントタイプに属します。階層は、以下のいずれかのプレフィックスが付いた名前のプロパティにマップされます。
  - カスタムワークスペース- *Hierarchy_hierarchyId*
  - 既定のワークスペース- *hierarchyId*

アセット

AWS IoT TwinMaker 内のアセットごとに新しいエンティティを作成します AWS IoT SiteWise。

- entityIdは assetId in と同じです AWS IoT SiteWise。
- これらのエンティティには sitewiseBase という単一のコンポーネントがあり、そのコンポーネントタイプはこのアセットのアセットモデルに対応します。
- プロパティエイリアスや測定単位の設定など、アセットレベルのオーバーライドはすべて、AWS IoT TwinMakerのエンティティに反映されます。

既定のワークスペースのみ。

以下のアセットは同期され、デフォルトのワークスペースでのみ使用できます。IoTSiteWiseDefaultWorkspace

AssetModelComponents

AWS IoT TwinMaker AssetModelComponents AWS IoT SiteWise各インに新しいコンポーネントタイプを作成します。

- TypeIdアセットモデルのコンポーネントは以下のパターンを使用します:assetModelId.
- アセットモデル内の各プロパティは、コンポーネントタイプの新しいプロパティで、プロパティ名は propertyId です。 AWS IoT SiteWise のプロパティ名は、displayNameプロパティ定義にとして保存されます。
- LISTRELATIONSHIPアセットモデルの各階層は新しいタイプのプロパティで、nestedTypeはコンポーネントタイプに属します。階層は、名前の前に hierarchyId が付いたプロパティにマップされます。

AssetModelCompositeModel

AWS IoT TwinMaker AssetModelCompositeModel各インに対して新しいコンポーネントタイプを作成します AWS IoT SiteWise。

- TypeIdアセットモデルのコンポーネントは以下のパターンを使用します:assetModelId_assetModelCompositeModelId.

- アセットモデル内の各プロパティは、コンポーネントタイプの新しいプロパティで、プロパティ名は `propertyId` です。 AWS IoT SiteWise のプロパティ名は、`displayName`プロパティ定義にとして保存されます。

AssetCompositeModels

AWS IoT TwinMaker `AssetCompositeModel`in ごとに新しい複合コンポーネントを作成します AWS IoT SiteWise。

- `componentName`は `assetModelCompositeModelId` in と同じです AWS IoT SiteWise。

リソースは同期されていません。

次のリソースは同期されません。

同期されていないアセットとアセットモデル

- アラームモデルは CompositeModels として同期されますが、アラームに関連するアセット内の対応するデータは同期されません。
- AWS IoT SiteWise データストリームは同期されません。アセットモデルでモデル化されたプロパティのみが同期されます。
- 属性、測定値、変換、集計、および式やウィンドウなどのメタデータ計算のプロパティ値は同期されません。エイリアス、測定単位、データ型などのプロパティに関するメタデータのみが同期されます。 AWS IoT TwinMaker 値は通常のデータコネクター API を使用してクエリできます。 GetPropertyValueHistory

同期されたエンティティとコンポーネントタイプは以下で使用してください。 AWS IoT TwinMaker

アセットが同期されると AWS IoT SiteWise、同期されたコンポーネントタイプは読み取り専用になります。 AWS IoT TwinMaker更新や削除のアクションはすべてで行う必要があり AWS IoT SiteWise、SyncJob AWS IoT TwinMaker がまだアクティブな場合は変更内容が同期されます。

AWS IoT SiteWise 同期されたエンティティとベースコンポーネントも読み取り専用になります。 AWS IoT TwinMaker説明や `entityName` などのエンティティレベルの属性が更新されていない限り、同期されていないコンポーネントを同期されたエンティティに追加できます。

同期されたエンティティを操作する方法には、いくつかの制限があります。同期されたエンティティの階層内の同期されたエンティティの下に子エンティティを作成することはできません。さらに、同

期されたコンポーネントタイプから派生する同期されていないコンポーネントタイプを作成することはできません。

> **Note**
>
> でアセットが削除されたり、 AWS IoT SiteWise 同期ジョブを削除したりすると、追加のコンポーネントはエンティティとともに削除されます。

これらの同期されたエンティティは Grafana ダッシュボードで使用でき、通常のエンティティと同様にシーンコンポーザーにタグとして追加できます。これらの同期されたエンティティに対してナレッジグラフクエリを発行することもできます。

> **Note**
>
> 変更されていない同期されたエンティティには料金は発生しませんが、 AWS IoT TwinMaker で変更が加えられた場合はそれらのエンティティに対して料金が発生します。たとえば、同期されていないコンポーネントを同期されたエンティティに追加すると、そのエンティティは課金されるようになります。 AWS IoT TwinMaker詳細については、「AWS IoT TwinMaker の料金」を参照してください。

同期ステータスとエラーを分析する

このトピックでは、同期エラーとステータスを分析する方法についてのガイダンスを提供します。

> **Important**
>
> the section called “カスタムワークスペースとデフォルトワークスペースの違い”カスタムワークスペースとデフォルトワークスペースの違いについては、を参照してください。

ジョブステータスを同期する

同期ジョブは、状態に応じて以下のいずれかのステータスになります。

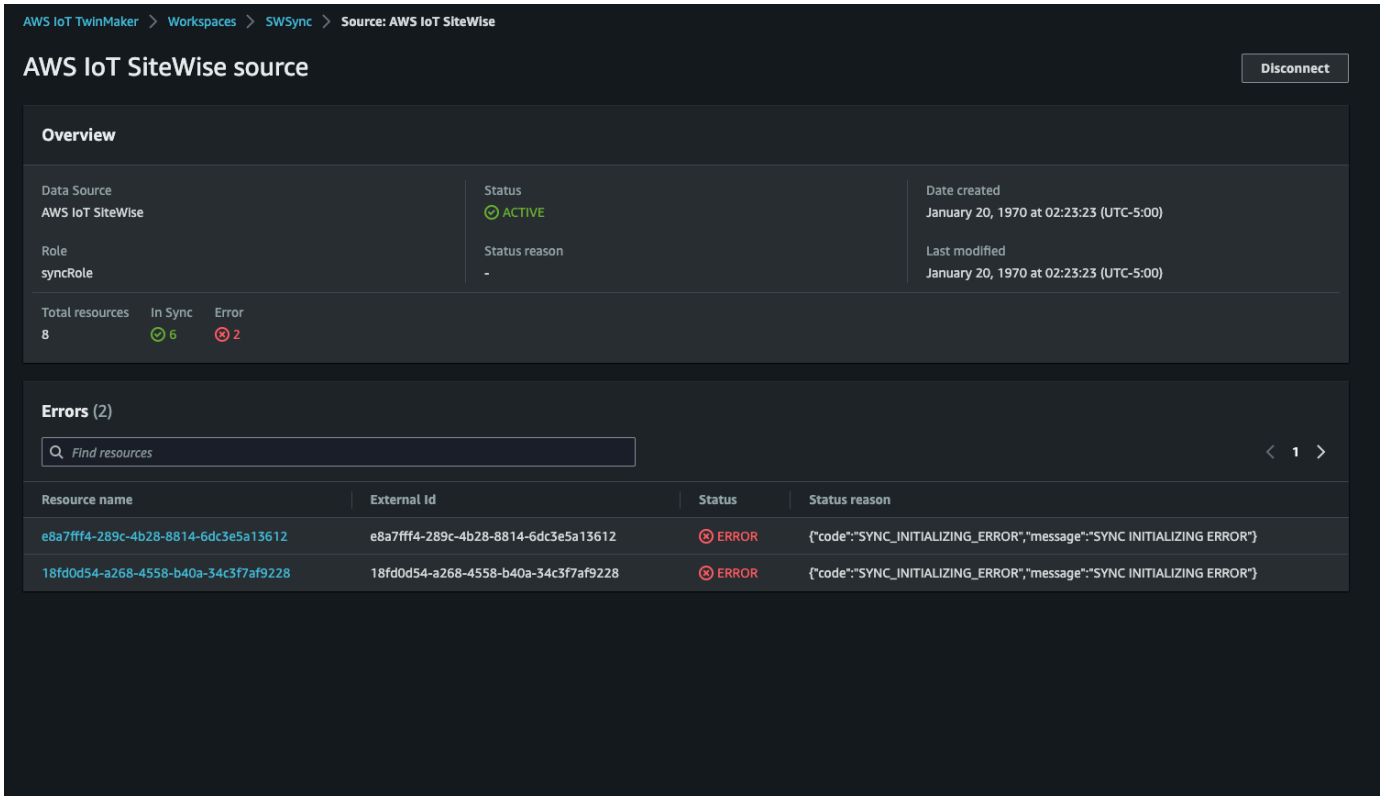
- `CREATING`同期ジョブの状態は、ジョブが権限をチェックし、 AWS IoT SiteWise 同期の準備のためにデータをロードしていることを意味します。

- `INITIALIZING`同期ジョブの状態は、 AWS IoT SiteWise にある既存のリソースがすべて同期されていることを意味します。 AWS IoT TwinMakerユーザーが多数のアセットとアセットモデルが AWS IoT SiteWiseにある場合は、この手順が完了するまでに時間がかかることがあります。同期されたリソースの数は、「AWS IoT TwinMaker 」 コンソールで同期ジョブを確認するか、`ListSyncResources` API を呼び出して監視できます。
- 同期ジョブの `ACTIVE` 状態は、初期化ステップが完了している状態です。これで、ジョブは AWS IoT SiteWiseからの新しい更新を同期する準備ができました。
- 同期ジョブの `ERROR` 状態は、前述のいずれかの状態でのエラーを示しています。エラーメッセージを確認します。IAM ロールの設定に問題がある可能性があります。新しい IAM ロールを使用する場合は、エラーが発生した同期ジョブを削除し、新しいロールで新しい IAM ロールを作成してください。

同期エラーは、ワークスペースのエンティティモデルソーステーブルからアクセスできるモデルソースページに表示されます。モデルソースページには、同期に失敗したリソースのリストが表示されます。ほとんどのエラーは同期ジョブによって自動的に再試行されますが、リソースにアクションが必要な場合は `ERROR` 状態のままになります。 ListSyncResourcesAPI を使用してエラーのリストを取得することもできます。

現在のソースのリストにあるエラーをすべて表示するには、以下の手順を実行します。

1. 「AWS IoT TwinMaker 」 コンソールのワークスペースに移動します。
2. AWS IoT SiteWise エンティティモデルソースモーダルにリストされているソースを選択し、アセット同期の詳細ページを開きます。



3. 前のスクリーンショットに示すように、エラーが続くリソースは エラーテーブルに一覧表示されます。このテーブルを使用して、特定のリソースに関連するエラーを追跡して修正できます。

次のようなエラーが考えられます。

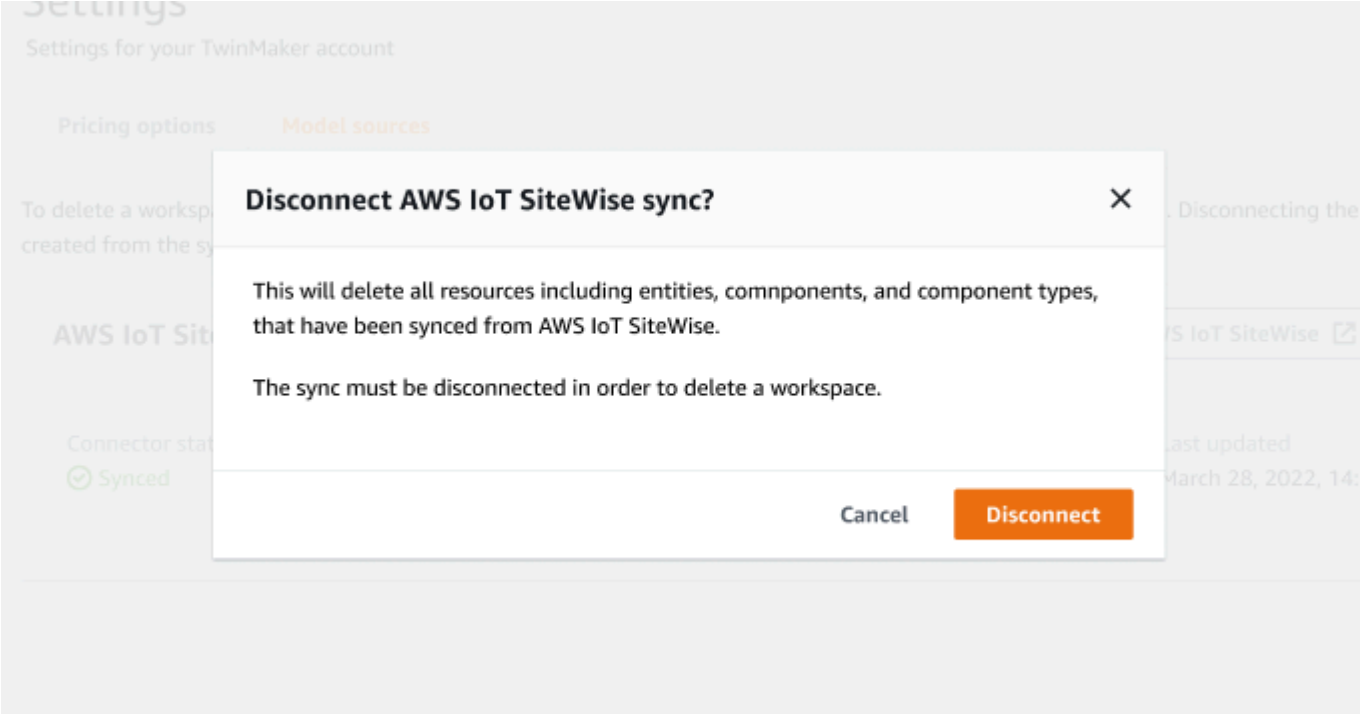
- AWS IoT SiteWise 重複するアセット名はサポートしますが、同じ親エンティティではサポートされず、 AWS IoT TwinMaker R00T同じレベルでのみサポートされます。の親エンティティの下に同じ名前のアセットが 2 つあると AWS IoT SiteWise、そのうちの 1 つは同期に失敗します。このエラーを修正するには、 AWS IoT SiteWise 同期する前に 1 つのアセットを削除するか、別の親アセットの下に移動してください。
- AWS IoT SiteWise アセット ID と同じ ID のエンティティが既にある場合、そのアセットは既存のエンティティを削除するまで同期されません。

同期ジョブを削除する

以下の手順に従って、同期ジョブを削除します。

> **Important**
>
> the section called “カスタムワークスペースとデフォルトワークスペースの違い”カスタムワークスペースとデフォルトワークスペースの違いについては、を参照してください。

1. AWS IoT TwinMaker コンソールに移動します。
2. 同期ジョブを削除するワークスペースを開きます。
3. 「エンティティモデルソース」 で、「 AWS IoT SiteWise ソース」 を選択してソースの詳細ページを開きます。
4. 同期ジョブを停止するには、「切断」 を選択します。同期ジョブを完全に削除するかどうかの選択を確定します。



同期ジョブが削除されると、同じワークスペースまたは別のワークスペースで同期ジョブを再作成できます。

ワークスペースに同期ジョブがある場合、そのワークスペースは削除できません。ワークスペースを削除する前に、まず同期ジョブを削除してください。

同期ジョブの削除中にエラーが発生した場合、同期ジョブは DELETING 状態のままになり、自動的に再試行されます。リソースの削除に関連するエラーが発生した場合に、同期されたエンティティまたはコンポーネントタイプを手動で削除できるようになりました。

> **Note**
>
> AWS IoT SiteWise 同期元のリソースが最初に削除され、次に同期ジョブ自体が削除されます。

アセット同期の上限

> **Important**
>
> the section called “カスタムワークスペースとデフォルトワークスペースの違い”カスタムワークスペースとデフォルトワークスペースの違いについては、を参照してください。

AWS IoT SiteWise クォータはデフォルトのAWS IoT TwinMaker クォータよりも高いため、 AWS IoT SiteWiseからの同期のエンティティとコンポーネントタイプの以下の制限を引き上げています。

- 同期できるのは1000のアセットモデルだけなので、1つのワークスペースで同期できるコンポーネントタイプは1000種類です。 AWS IoT SiteWise
- 同期できるのは 100,000 個のアセットのみなので、ワークスペース内の 100,000 個の同期済みエンティティ。 AWS IoT SiteWise
- 親エンティティあたりの子エンティティは最大 2000 です。1 つの親アセットにつき 2000 の子アセットを同期します。

  > **Note**
  >
  > GetEntityAPI は階層プロパティの最初の 50 個の子エンティティのみを返しますが、GetPropertyValueこの API を使用してすべての子エンティティのリストをページ分割して取得できます。

- 同期されるコンポーネントごとに 600 のプロパティ。これにより AWS IoT SiteWise、合計 600 のプロパティと階層を含むアセットモデルを同期できます。

> **Note**
>
> これらの制限は同期されたエンティティにのみ適用されます。同期されていないリソースの制限を増やす必要がある場合は、クォータ引き上げをリクエストしてください。

AWS IoT TwinMaker Grafana ダッシュボードの統合

AWS IoT TwinMaker はアプリケーションプラグインによる Grafana 統合をサポートします。Grafana バージョン 8.2.0 以降を使用して、デジタルツインアプリケーションを操作できます。AWS IoT TwinMaker プラグインは、デジタルツインデータに接続するためのカスタムパネル、ダッシュボードテンプレート、およびデータソースを提供します。

Grafana のオンボーディングとダッシュボードのアクセス許可の設定方法の詳細については、次のトピックを参照してください。

トピック

- Grafana シーンビューアーの CORS の設定
- Grafana 環境の設定
- ダッシュボード IAM ロールの作成
- AWS IoT TwinMaker ビデオプレーヤーポリシーの作成

> **Note**
>
> Amazon S3 バケットの CORS (クロスオリジンリソース共有) の設定を変更して、Grafana ユーザーインターフェイスがバケットからリソースをロードできるようにする必要があります。手順については、「Grafana シーンビューアーの CORS の設定」を参照してください。

AWS IoT TwinMaker Grafana プラグインの詳細については、「AWS IoT TwinMakerアプリケーションドキュメント」を参照してください。

Grafana プラグインの主要コンポーネントについては、以下を参照してください。

- AWS IoT TwinMaker データソース
- ダッシュボードテンプレート
- シーンビューアーパネル
- ビデオプレーヤーパネル

Grafana シーンビューアーの CORS の設定

AWS IoT TwinMaker Grafana プラグインには CORS (クロスオリジンリソース共有) の設定が必要です。これにより、Grafana ユーザーインターフェイスが Amazon S3 バケットからリソースをロードできるようになります。CORS の設定がないと、Grafana ドメインは Amazon S3 バケット内のリソースにアクセスできないため、シーンビューアーに「ネットワーク障害により 3D シーンのロードに失敗しました」というエラーメッセージが表示されます。

Amazon S3 バケットに CORS を設定するには、次の手順を実行します。

1. 「IAM」 コンソールにサインインし 「Amazon S3」 コンソールを開きます。
2. 「バケット」 リストで、AWS IoT TwinMaker ワークスペースのリソースバケットとして使用するバケットの名前を選択します。
3. 「アクセス許可」 を選択します。
4. 「クロスオリジンリソース共有」 セクションで 「編集」 を選択して CORS エディタを開きます。
5. 「CORS 設定エディタ」 のテキストボックスで、Grafana ワークスペースドメイン *GRAFANA-WORKSPACE-DOMAIN* を自分のドメインに置き換えて、次の JSON CORS 設定を入力するか、コピーして貼り付けます。

> **Note**
>
> "AllowedOrigins": JSON 要素の先頭にあるアスタリスク * 文字はそのままにしておく必要があります。

```
    [
{
    "AllowedHeaders": [
        "*"
    ],
    "AllowedMethods": [
        "GET",
        "PUT",
        "POST",
        "DELETE",
        "HEAD"
```

```
        ],
        "AllowedOrigins": [
            "*GRAFANA-WORKSPACE-DOMAIN"
        ],
        "ExposeHeaders": [
            "ETag"
        ]
    }
]
```

6. 変更の保存を選択して、CORS 設定を完了します。

Amazon S3 バケットでの CORS の詳細については、「Cross-Origin Resource Sharing (CORS) の使用」を参照してください。

Grafana 環境の設定

Amazon Managed Grafana を使用してフルマネージド型のサービスを使用することも、自分で管理する Grafana 環境を設定することもできます。Amazon Managed Grafana を使用すると、オープンソースの Grafana をニーズに合わせて迅速にデプロイ、運用、スケールできます。または、Grafana サーバーを管理する独自のインフラストラクチャを設定することもできます。

Grafana 環境オプションの詳細については、次のトピックを参照してください。

- Amazon Managed Grafana
- セルフマネージド型 Grafana

Amazon Managed Grafana

Amazon Managed Grafana では AWS IoT TwinMaker プラグインが提供されるため、AWS IoT TwinMaker を Grafana とすばやく統合できます。Amazon Managed Grafana が Grafana サーバーを管理するため、お客様はハードウェアやその他の Grafana インフラストラクチャを構築、パッケージ化、デプロイすることなく、データを視覚化することができます。Amazon Managed Grafana の詳細については、「Amazon Managed Grafana とは」を参照してください。

> **Note**
>
> Amazon Managed Grafana は現在、バージョン 1.3.1 の AWS IoT TwinMaker Grafana プラグインをサポートしています。

Amazon Managed Grafana の前提条件

AWS IoT TwinMaker を Amazon Managed Grafana ダッシュボードで使用するには、まず以下の前提条件を満たす必要があります。

- AWS IoT TwinMaker ワークスペースを作成します。ワークスペースの作成の詳細については、「AWS IoT TwinMaker の使用開始」を参照してください。

> **Note**
>
> AWS マネジメントコンソールで Amazon Managed Grafana ワークスペースを初めて作成したときは、AWS IoT TwinMaker は表示されません。ただし、プラグインはすべてのワークスペースにすでにインストールされています。AWS IoT TwinMaker プラグインはオープンソースの 「Grafana プラグイン」 リストで確認できます。AWS IoT TwinMaker データソースは、「データソース」 ページで 「データソースを追加」 を選択することで確認できます。

Amazon Managed Grafana ワークスペースを作成すると、Grafana インスタンスのアクセス許可を管理するための IAM ロールが自動的に作成されます。これはワークスペース IAM ロールと呼ばれます。これは、Grafana のすべての AWS IoT TwinMaker データソースを設定するために使用する認証プロバイダーオプションです。Amazon Managed Grafana は AWS IoT TwinMaker のアクセス許可の自動追加をサポートしていないため、これらのアクセス許可は手動で設定する必要があります。手動によるアクセス許可の詳細については、「ダッシュボード IAM ロールの作成」を参照してください。

セルフマネージド型 Grafana

Grafana を実行する独自のインフラストラクチャをホストすることを選択できます。Grafana をマシン上でローカルで実行する方法については、「Grafana のインストール」を参照してください。AWS IoT TwinMaker プラグインは、公開中の Grafana カタログで利用可能です。このプラグインを Grafana 環境にインストールする方法については、「AWS IoT TwinMaker アプリケーション」を参照してください。

Grafana をローカルで実行する場合、ダッシュボードを簡単に共有したり、複数のユーザーにアクセスを提供したりすることはできません。ローカル Grafana を使ったダッシュボードの共有に関するスクリプト形式のクイックスタートガイドについては、「AWS IoT TwinMaker サンプルリポジトリ」をご覧ください。このリソースでは、Grafana 環境を Cloud9 で、Amazon EC2 をパブリックエンドポイントでホストする手順を説明します。

TwinMaker データソースの設定に使用する認証プロバイダーを決定する必要があります。環境の認証情報は、デフォルトの認証情報チェーンに基づいて設定します (「デフォルトの認証情報プロバイダーチェーンの使用」を参照)。デフォルト認証情報は、どのユーザーまたはロールの永久認証情報でもかまいません。たとえば、Amazon EC2 で Grafana を実行している場合、デフォルトの認証情報チェーンは Amazon EC2 実行ロールにアクセスでき、これが認証プロバイダーになります。ダッシュボード IAM ロールの作成 の手順では、認証プロバイダーの IAM Amazon リソースネーム (ARN) が必要です。

ダッシュボード IAM ロールの作成

AWS IoT TwinMaker を使用すると、Grafana ダッシュボードのデータアクセスを制御できます。Grafana ダッシュボードのユーザーは、データを表示したり、場合によってはデータを書き込んだりするために、さまざまなアクセス許可の範囲を持つ必要があります。たとえば、アラームオペレーターには動画を視聴するアクセス許可がない場合がありますが、管理者にはすべてのリソースに対するアクセス許可があります。Grafana は、認証情報と IAM ロールが提供されるデータソースを通じてアクセス許可を定義します。AWS IoT TwinMaker データソースは、そのロールのアクセス許可を持つ AWS の認証情報を取得します。IAM ロールが提供されていない場合、Grafana は認証情報の範囲を使用しますが、これは AWS IoT TwinMaker によって減らすことはできません。

AWS IoT TwinMaker のダッシュボードを Grafana で使用するには、IAM ロールを作成してポリシーをアタッチします。次のテンプレートを使用して、これらのポリシーを作成できます。

IAM ポリシーを作成する

IAM コンソールで *YourWorkspaceId*DashboardPolicy と呼ばれる IAM ポリシーを作成します。このポリシーは、ワークスペースに Amazon S3 バケットと AWS IoT TwinMaker リソースへのアクセスを許可します。Amazon Kinesis Video Stream用 AWS IoT Greengrass エッジコネクタ を使用することもできます。これには、キネシス　ビデオ　ストリーム のアクセス許可と、コンポーネントに設定された AWS IoT SiteWise アセットが必要です。ユースケースに合わせて、次のいずれかのポリシーテンプレートを選択します。

1. 動画へのアクセス許可なしポリシー

Grafana のビデオプレーヤーパネルを使用しない場合は、次のテンプレートを使用してポリシーを作成します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::bucketName/*",
        "arn:aws:s3:::bucketName"
        ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iottwinmaker:Get*",
        "iottwinmaker:List*"
      ],
      "Resource": [
        "arn:aws:iottwinmaker:region:accountId:workspace/workspaceId",
        "arn:aws:iottwinmaker:region:accountId:workspace/workspaceId/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iottwinmaker:ListWorkspaces",
      "Resource": "*"
    }
  ]
}
```

Amazon S3 の各バケットは、ワークスペースごとに作成されます。ダッシュボードに表示できる 3D モデルとシーンが含まれています。SceneViewer パネルは、このバケットから項目をロードします。

2。動画へのアクセス許可範囲絞り込みポリシー

Grafana のビデオプレーヤーパネルへのアクセスを制限するには、Amazon Kinesis Video Stream 用 AWS IoT Greengrass エッジコネクタのリソースをタグ別にグループ化します。動画リソースへのアクセス許可範囲を絞り込む方法の詳細については、「AWS IoT TwinMaker ビデオプレーヤーポリシーの作成」を参照してください。

3。すべての動画へのアクセス許可

動画をグループ化しない場合は、Grafana ビデオプレーヤーからすべての動画にアクセスできるようにすることができます。Grafana ワークスペースにアクセスできる人なら誰でも、アカウント内のどのストリームでも動画を再生でき、どの AWS IoT SiteWise アセットにも読み取り専用アクセス許可があります。これには、今後作成されるすべてのリソースが含まれます。

次のテンプレートを使用してポリシーを作成します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::bucketName/*",
        "arn:aws:s3:::bucketName"
        ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iottwinmaker:Get*",
        "iottwinmaker:List*"
      ],
      "Resource": [
        "arn:aws:iottwinmaker:region:accountId:workspace/workspaceId",
        "arn:aws:iottwinmaker:region:accountId:workspace/workspaceId/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iottwinmaker:ListWorkspaces",
```

```
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "kinesisvideo:GetDataEndpoint",
        "kinesisvideo:GetHLSStreamingSessionURL"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iotsitewise:GetAssetPropertyValue",
        "iotsitewise:GetInterpolatedAssetPropertyValues"
      ],
      "Resource": "*"
    },
    {
       "Effect": "Allow",
       "Action": [
        "iotsitewise:BatchPutAssetPropertyValue"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "aws:ResourceTag/EdgeConnectorForKVS": "*workspaceId*"
        }
      }
    }
  ]
}
```

このポリシーテンプレートで、次のアクセス許可が付与されます。

- シーンをロードするための S3 バケットへの読み取り専用アクセス。
- ワークスペース内のすべてのエンティティとコンポーネントの AWS IoT TwinMaker への読み取り専用アクセス。
- アカウント内のすべての キネシス　ビデオ　ストリーム 動画をストリーミングするための読み取り専用アクセス。
- アカウント内のすべての AWS IoT SiteWise アセットの資産価値履歴への読み取り専用アクセス。

- キー EdgeConnectorForKVS と値 workspaceId がタグ付けされた AWS IoT SiteWise アセットの任意のプロパティへのデータインジェスト。

エッジからのカメラ AWS IoT SiteWise アセットリクエスト動画アップロードのタグ付け

Grafana のビデオプレーヤーを使用すると、ユーザーは動画をエッジキャッシュから キネシス　ビデオ　ストリーム にアップロードするよう手動でリクエストできます。この機能は、Amazon Kinesis Video Stream用 AWS IoT Greengrass エッジコネクタに関連付けられていて、キー EdgeConnectorForKVS でタグ付けされているすべてのAWS IoT SiteWiseアセットで有効にできます。

タグ値には、以下の文字のいずれかで区切られた WorkspaceID のリストを使用できます: . : + = @ _ / -。たとえば、Amazon Kinesis Video Stream用 AWS IoT Greengrass エッジコネクタに関連付けられた AWS IoT SiteWise アセットを AWS IoT TwinMaker ワークスペース全体で使用する場合は、次のパターンに従うタグを使用できます: WorkspaceA/WorkspaceB/WorkspaceC。Grafana プラグインは、AWS IoT TwinMaker WorkspaceID を使用して AWS IoT SiteWise アセットデータインジェストのグループ化を強制します。

ダッシュボードポリシーにさらにアクセス許可を追加する

AWS IoT TwinMaker Grafana プラグインは、認証プロバイダーを使用して、作成したダッシュボードロール AssumeRole で を呼び出します。内部的には、プラグインは、 AssumeRole 呼び出しでセッションポリシーを使用して、アクセスできるアクセス許可の最大範囲を制限します。セッションポリシーの詳細については、「セッションポリシー」を参照してください。

AWS IoT TwinMaker ワークスペースのダッシュボードロールに設定できる最大許容ポリシーは次のとおりです。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::bucketName/*",
```

```
        "arn:aws:s3:::bucketName"
        ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iottwinmaker:Get*",
        "iottwinmaker:List*"
      ],
      "Resource": [
        "arn:aws:iottwinmaker:region:accountId:workspace/workspaceId",
        "arn:aws:iottwinmaker:region:accountId:workspace/workspaceId/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iottwinmaker:ListWorkspaces",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "kinesisvideo:GetDataEndpoint",
        "kinesisvideo:GetHLSStreamingSessionURL"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iotsitewise:GetAssetPropertyValue",
        "iotsitewise:GetInterpolatedAssetPropertyValues"
      ],
      "Resource": "*"
    },
    {
       "Effect": "Allow",
       "Action": [
        "iotsitewise:BatchPutAssetPropertyValue"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "aws:ResourceTag/EdgeConnectorForKVS": "*workspaceId*"
```

```
        }
      }
    }
  ]
}
```

Allowより多くのアクセス許可を持つステートメントを追加すると、AWS IoT TwinMaker プラグインでは機能しなくなります。これは、プラグインが必要最小限のアクセス許可を使用するための設計によるものです。

ただし、アクセス許可の範囲をさらに絞り込むこともできます。詳細については、「AWS IoT TwinMaker ビデオプレーヤーポリシーの作成」を参照してください。

Grafana ダッシュボード IAM ロールの作成

IAM コンソールを使用して *YourWorkspaceId*DashboardRole と呼ばれる IAM ロールを作成します。*YourWorkspaceId*DashboardPolicy をロールにアタッチします。

ダッシュボードロールの信頼ポリシーを編集するには、Grafana 認証プロバイダーに AssumeRole をダッシュボードロールに呼び出すためのアクセス許可を付与する必要があります。次のテンプレートを使用して信頼ポリシーを更新します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "ARN of Grafana authentication provider"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Grafana 環境の作成と認証プロバイダーの検索の詳細については、「Grafana 環境の設定」を参照してください。

AWS IoT TwinMaker ビデオプレーヤーポリシーの作成

以下は、Grafana の AWS IoT TwinMaker プラグインに必要なすべての動画へのアクセス許可を含むポリシーのテンプレートです。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::bucketName/*",
        "arn:aws:s3:::bucketName"
        ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iottwinmaker:Get*",
        "iottwinmaker:List*"
      ],
      "Resource": [
        "arn:aws:iottwinmaker:region:accountId:workspace/workspaceId",
        "arn:aws:iottwinmaker:region:accountId:workspace/workspaceId/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iottwinmaker:ListWorkspaces",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "kinesisvideo:GetDataEndpoint",
        "kinesisvideo:GetHLSStreamingSessionURL"
      ],
      "Resource": "*"
    },
    {
```

```
      "Effect": "Allow",
      "Action": [
        "iotsitewise:GetAssetPropertyValue",
        "iotsitewise:GetInterpolatedAssetPropertyValues"
      ],
      "Resource": "*"
    },
    {
       "Effect": "Allow",
       "Action": [
        "iotsitewise:BatchPutAssetPropertyValue"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "aws:ResourceTag/EdgeConnectorForKVS": "*workspaceId*"
        }
      }
    }
  ]
}
```

ポリシー全体についての詳細については、IAM ポリシーを作成する トピックの「すべての動画へのアクセス許可ポリシーのテンプレート」を参照してください。

リソースへのアクセス範囲の絞り込み

Grafana の Video Player パネルは、Kinesis Video Streams と IoT SiteWise を直接呼び出して、完全な動画再生エクスペリエンスを提供します。AWS IoT TwinMaker ワークスペースに関連付けられていないリソースへの不正アクセスを防ぐには、ワークスペースダッシュボードロールの IAM ポリシーに条件を追加します。

GET アクセス許可の範囲の絞り込み

リソースにタグを付けることで、Amazon Kinesis Video Streamと AWS IoT SiteWise アセットへのアクセスを絞り込むことができます。動画のアップロードリクエスト機能を有効にするために、すでに AWS IoT TwinMaker WorkspaceID に基づいて AWS IoT SiteWise カメラアセットにタグを付けている場合があります。「エッジから動画をアップロード」のトピックを参照してください。同じタグキーと値のペアを使用して AWS IoT SiteWise アセットへの GET アクセスを制限したり、同じ方法で キネシス　ビデオ　ストリーム にタグを付けることができます。

その後、*YourWorkspaceId*DashboardPolicy 内の kinesisvideo ステートメントと iotsitewise ステートメントにこの条件を追加できます。

```
"Condition": {
  "StringLike": {
    "aws:ResourceTag/EdgeConnectorForKVS": "*workspaceId*"
  }
}
```

実際の使用事例: カメラのグループ化

このシナリオには、工場でクッキーを焼くプロセスを監視するカメラが多数あります。クッキー生地の束は生地部屋で作られ、生地は冷凍部屋で冷凍され、クッキーは焼成部屋で焼かれます。これらの各部屋にはカメラがあり、異なるオペレーターチームが各プロセスを個別に監視しています。各グループのオペレーターに、それぞれの部屋の権限を与えたいと考えています。クッキー工場のデジタルツインを構築する場合、使用するワークスペースは 1 つだけですが、カメラのアクセス許可範囲は部屋ごとに設定する必要があります。

このようなアクセス許可の分離は、GroupingID に基づいてカメラのグループにタグを付けることで実現できます。このシナリオでは、groupingIds は BatterRoom、 FreezerRoom、および で す BakingRoom。各部屋のカメラは キネシス　ビデオ　ストリーム に接続されており、Key = EdgeConnectorForKVS、Value = BatterRoom のタグが付いている必要があります。値には、次のいずれかの文字で区切られたグループのリストを指定できます: `.` `:` `+` `=` `@` `_` `/` `-`。

*YourWorkspaceId*DashboardPolicy を修正するには、以下のポリシーステートメントを使用します。

```
...,
{
  "Effect": "Allow",
  "Action": [
    "kinesisvideo:GetDataEndpoint",
    "kinesisvideo:GetHLSStreamingSessionURL"
  ],
  "Resource": "*",
  "Condition": {
    "StringLike": {
      "aws:ResourceTag/EdgeConnectorForKVS": "*groupingId*"
    }
  }
```

```
},
{
  "Effect": "Allow",
  "Action": [
    "iotsitewise:GetAssetPropertyValue",
    "iotsitewise:GetInterpolatedAssetPropertyValues"
  ],
  "Resource": "*",
  "Condition": {
    "StringLike": {
      "aws:ResourceTag/EdgeConnectorForKVS": "*groupingId*"
    }
  }
},
...
```

これらのステートメントは、グループ内の特定のリソースへのストリーミングビデオの再生と AWS IoT SiteWise プロパティ履歴へのアクセスを制限します。*groupingId* はユースケースによって定義されます。このシナリオでは、roomId になります。

スコープダウンAWS IoT SiteWise BatchPutAssetPropertyValue アクセス許可

このアクセス許可を与えると、ビデオプレーヤーの動画アップロードリクエスト機能が有効になります。動画をアップロードするときは、時間範囲を指定し 「Grafana ダッシュボード」 パネルで 「送信」 を選択してリクエストを送信できます。

iotsitewise:BatchPutAssetPropertyValue permissions を付与するには、デフォルトのポリシーを使用します。

```
...,
{
  "Effect": "Allow",
  "Action": [
    "iotsitewise:BatchPutAssetPropertyValue"
  ],
  "Resource": "*",
  "Condition": {
    "StringLike": {
      "aws:ResourceTag/EdgeConnectorForKVS": "*workspaceId*"
    }
```

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

このポリシーを使用すると、ユーザーはAWS IoT SiteWiseカメラアセット上の任意のプロパティ BatchPutAssetPropertyValue に対して を呼び出すことができます。ステートメントの条件で指定することで、特定の PropertyId の認証を制限できます。

```
{
  ...
  "Condition": {
    "StringEquals": {
      "iotsitewise:propertyId": "propertyId"
    }
  }
  ...
}
```

Grafana の Video Player パネルは、 という名前の測定プロパティにデータを取り込み VideoUploadRequest、エッジキャッシュから Kinesis Video Streams へのビデオのアップロードを開始します。「AWS IoT SiteWise」 コンソールでこのプロパティの PropertyId を検索します。*YourWorkspaceId*DashboardPolicy を修正するには、次のポリシーステートメントを使用します。

```
...,
{
  "Effect": "Allow",
  "Action": [
    "iotsitewise:BatchPutAssetPropertyValue"
  ],
  "Resource": "*",
  "Condition": {
    "StringLike": {
      "aws:ResourceTag/EdgeConnectorForKVS": "*workspaceId*"
    },
    "StringEquals": {
      "iotsitewise:propertyId": "VideoUploadRequestPropertyId"
    }
  }
},
...
```

このステートメントは、データの取り込みをタグ付けされた AWS IoT SiteWise カメラアセットの特定のプロパティに制限します。詳細については、「AWS IoT SiteWise と IAM の連携」を参照してください。

AWS IoT SiteWise アラームを AWS IoT TwinMaker Grafana ダッシュボードにConnect

> **Note**
>
> この機能はプレビュー公開リリースであり、変更される可能性があります。

AWS IoT TwinMaker AWS IoT SiteWise AWS IoT TwinMaker イベントアラームをコンポーネントにインポートできます。これにより、 AWS IoT SiteWise データ移行用のカスタムデータコネクタを実装しなくても、アラームのステータスを照会したり、アラームのしきい値を設定したりできます。AWS IoT TwinMaker Grafana プラグインを使用すると、アラームに対して API 呼び出しを行ったり、 AWS IoT TwinMaker アラームを直接操作したりすることなく、Grafana でアラームのステータスを視覚化し、アラームのしきい値を設定できます。 AWS IoT SiteWise

AWS IoT SiteWise アラーム設定の前提条件

アラームを作成して Grafana ダッシュボードに統合する前に、以下の前提条件を確認します。

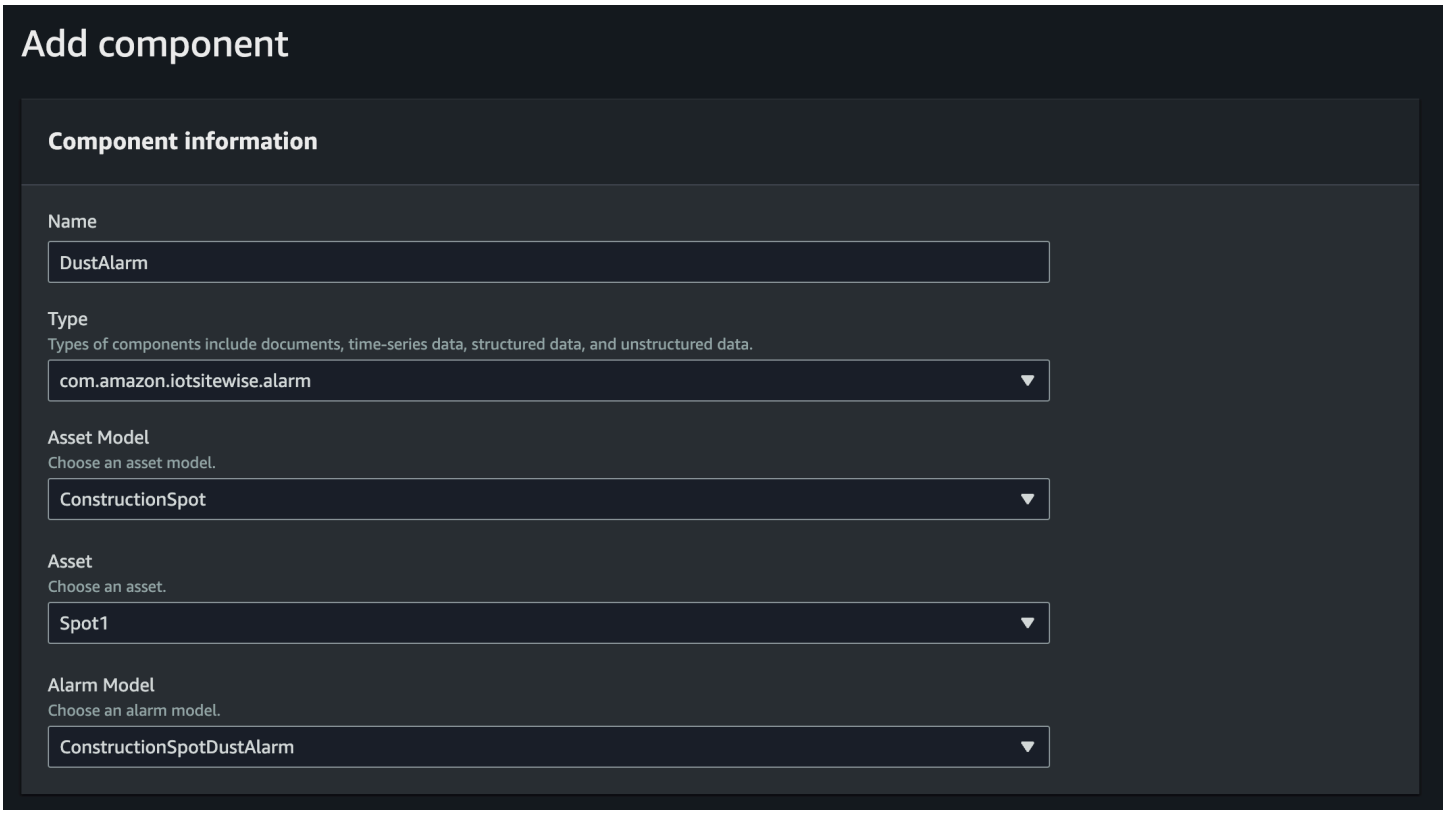
- AWS IoT SiteWiseのモデルと資産システムに慣れてください。詳細については、『ユーザーガイド』の「アセットモデルの作成」と「アセットの作成」を参照してください。
- IoT Events アラームモデルと、 AWS IoT SiteWise それらをモデルにアタッチする方法をよく理解してください。詳細については、『ユーザーガイド』の「AWS IoT イベントアラームの定義」を参照してください。
- Grafana AWS IoT TwinMaker と統合することで、Grafana AWS IoT TwinMaker 内のリソースにアクセスできるようになります。詳細については、AWS IoT TwinMaker Grafana ダッシュボードの統合 を参照してください。

アラームコンポーネントの IAM ロールを定義します。 AWS IoT SiteWise

AWS IoT TwinMaker ワークスペース IAM ロールを使用して、Grafana のアラームしきい値をクエリして設定します。Grafana AWS IoT SiteWise でアラームを操作するには、 AWS IoT TwinMaker ワークスペースロールに以下の権限が必要です。

```
{
"Effect": "Allow",
    "Action": [
        "iotevents:DescribeAlarmModel",
    ],
    "Resource": ["{IoTEventsAlarmModelArn}"]
},{
"Effect": "Allow",
    "Action": [
        "iotsitewise:BatchPutAssetPropertyValue"
    ],
    "Resource": ["{IoTSitewiseAssetArn}"]
}
```

AWS IoT TwinMaker コンソールで、アセットを表すエンティティを作成します。 AWS IoT SiteWise com.amazon.iotsitewise.alarmコンポーネントタイプとしてを使用してそのエンティティ用のコンポーネントを追加し、対応するアセットモデルとアラームモデルを選択してください。



上のスクリーンショットは、このエンティティをタイプで作成した例です。com.amazon.iotsitewise.alarm

このコンポーネントを作成すると、 AWS IoT TwinMaker AWS IoT SiteWise 関連するアラームプロパティがおよびから自動的にインポートされます AWS IoT Events。このアラームコンポーネントタイプのパターンを繰り返して、ワークスペースに必要なすべてのアセットのアラームコンポーネントを作成できます。

AWS IoT TwinMaker API を通じてクエリと更新を行います。

アラームコンポーネントを作成したら、 AWS IoT TwinMaker API を使用してアラームのステータス、しきい値をクエリし、アラームのしきい値を更新できます。

以下は、アラームステータスをクエリするリクエストの例です。

```
aws iottwinmaker get-property-value-history --cli-input-json \
'{
    "workspaceId": "{workspaceId}",
    "entityId": "{entityId}",
    "componentName": "{componentName}",
    "selectedProperties": ["alarm_status"],
    "startTime": "{startTimeIsoString}",
    "endTime": "{endTimeIsoString}"
}'
```

以下は、アラームのしきい値をクエリするリクエストの例です。

```
aws iottwinmaker get-property-value-history --cli-input-json \
'{
    "workspaceId": "{workspaceId}",
    "entityId": "{entityId}",
    "componentName": "{componentName}",
    "selectedProperties": ["alarm_threshold"],
    "startTime": "{startTimeIsoString}",
    "endTime": "{endTimeIsoString}"
}'
```

以下は、アラームのしきい値を更新するリクエストの例です。

```
aws iottwinmaker batch-put-property-values --cli-input-json \
'{
    "workspaceId": "{workspaceId}",
    "entries": [
        {
```

```
            "entityPropertyReference": {
                "entityId": "{entityId}",
                "componentName": "{componentName}",
                "propertyName": "alarm_threshold"
            },
            "propertyValues": [
                {
                    "value": {
                        "doubleValue": "{newThreshold}"
                    },
                    "time": "{effectiveTimeIsoString}"
                }
            ]
        }
    ]
}'
```

Grafana ダッシュボードをアラーム用に設定する

書き込み可能なダッシュボード IAM ロールを 2 つ作成する必要があります。これは通常のロールですが、iottwinmaker:BatchPutPropertyValues TwinMaker 以下の例のようにアクションをワークスペース arn に追加する権限があります。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iottwinmaker:Get*",
               "iottwinmaker:List*",
                "iottwinmaker:BatchPutPropertyValues"
            ],
            "Resource": [
                "{workspaceArn}",
                "{workspaceArn}/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iottwinmaker:ListWorkspaces",
            "Resource": "*"
```

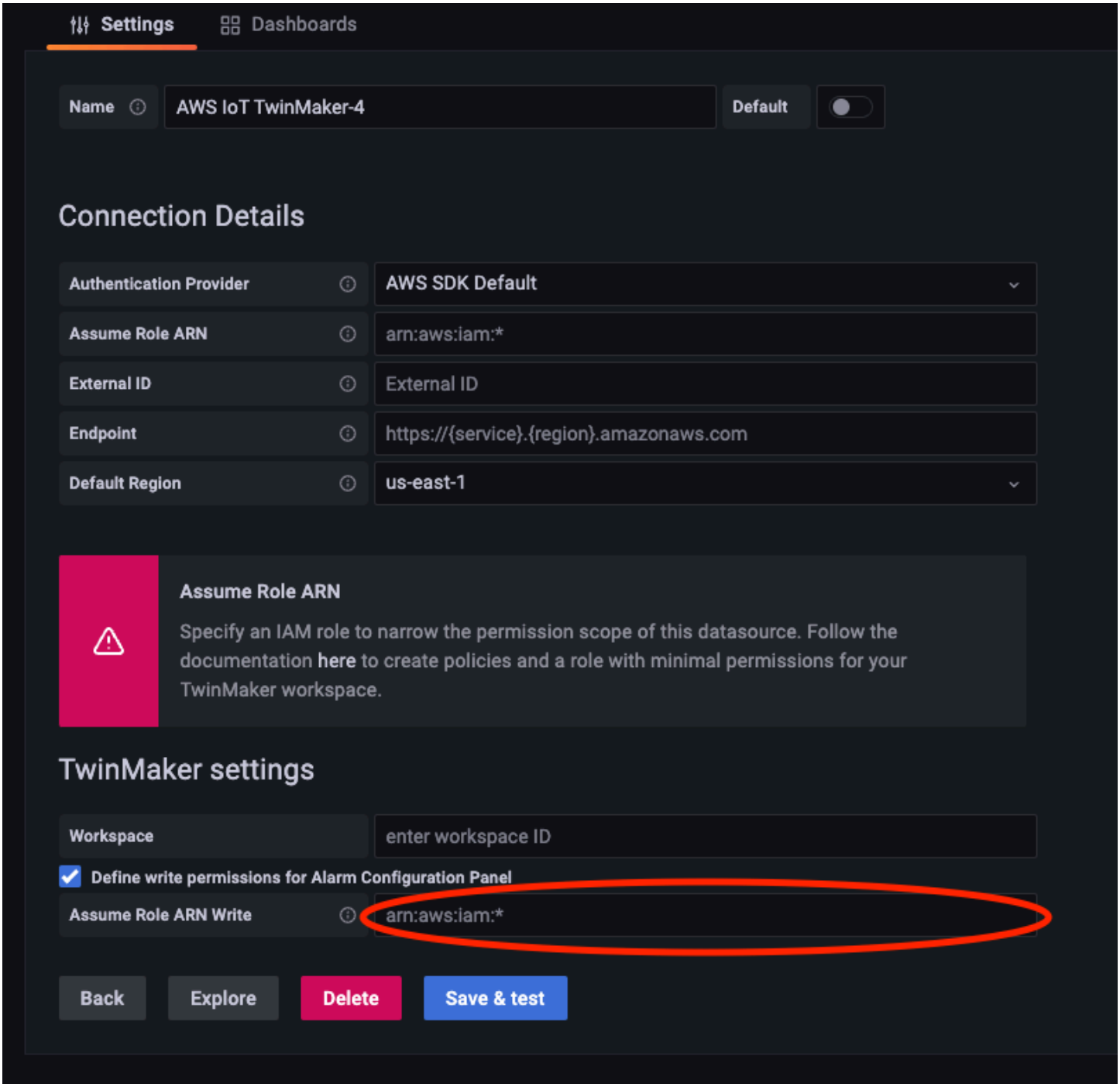
```
        }
    ]
}
```

代わりに、IAM ロールの最後にこのステートメントを追加することもできます。

```
{
    "Effect": "Allow",
    "Action": [
        "iottwinmaker:BatchPutPropertyValues"
    ],
    "Resource": [
        "{workspaceArn}",
        "{workspaceArn}/*"
    ]
}
```

データソースには、作成したダッシュボード書き込みロールに write arn が設定されている必要があります。

IAM ロールを変更したら、Grafana ダッシュボードにログインして、更新されたロール arn を引き継ぎます。「アラーム設定パネルの書き込み権限を定義」のチェックボックスを選択し、書き込みロールの arn をコピーします。



Grafana ダッシュボードを使用してアラームを視覚化する

以下の手順に従って、ダッシュボードにアラーム設定パネルを追加して設定します。

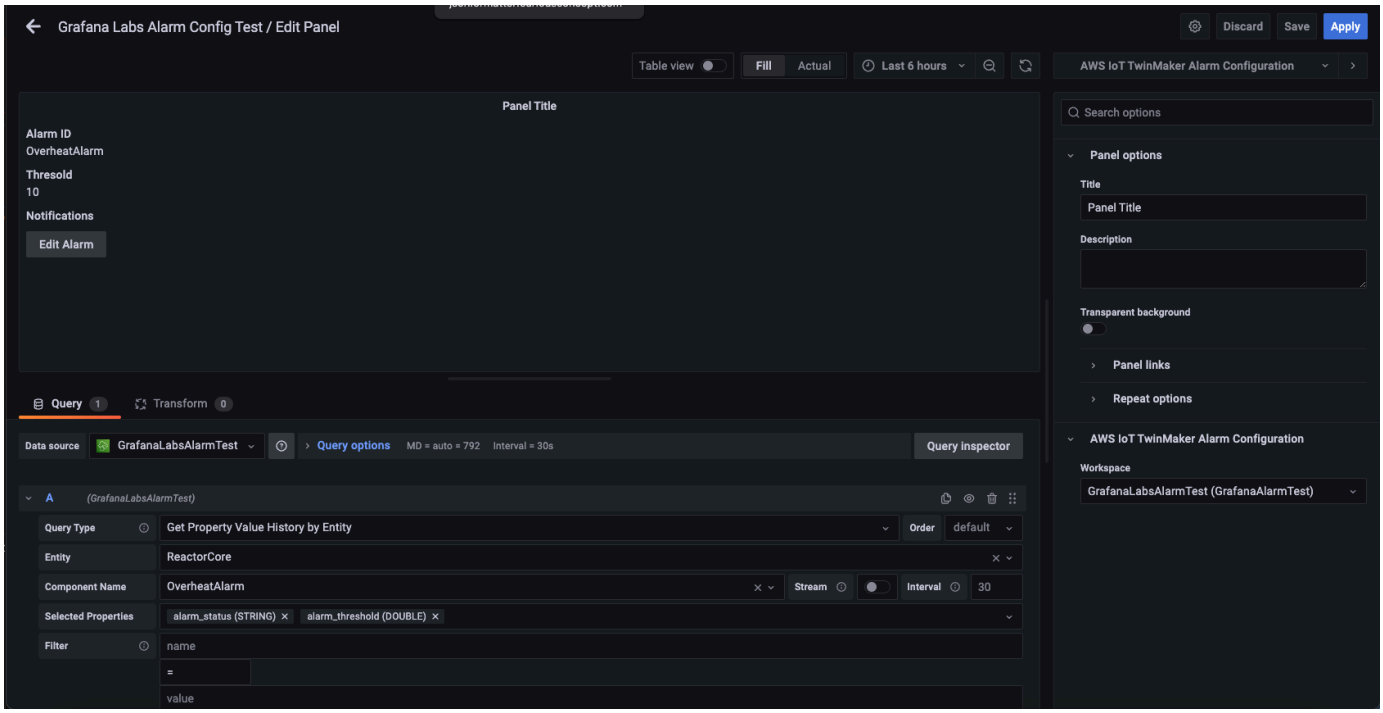
1. パネルオプションでワークスペースを選択します。
2. クエリ設定でデータソースを設定します。

3. 次のクエリタイプを使用します: `Get Property Value History by Entity`。
4. アラームを追加したいエンティティまたはエンティティ変数を選択します。
5. エンティティを選択したら、プロパティを適用するコンポーネントまたはコンポーネント変数を選択します。
6. プロパティには、`alarm_status` と `alarm_threshold` を選択します。

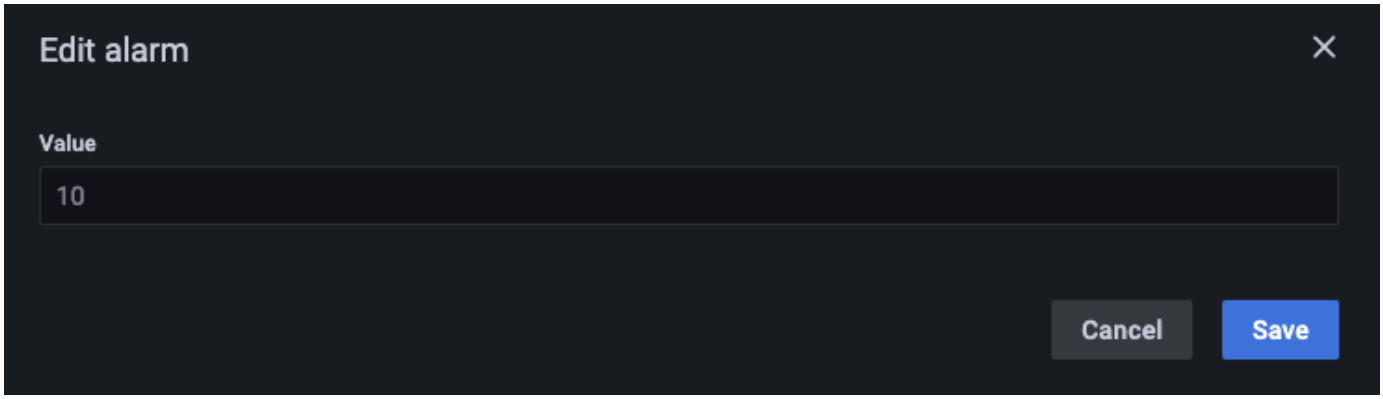
   接続されると、アラーム ID の ID と現在のしきい値が表示されます。

> **Note**
>
> プレビュー公開では、通知は表示されません。アラームのステータスとしきい値を確認して、プロパティが正しく適用されていることを確認する必要があります。

7. 最新の値が表示されるように、デフォルトのクエリの昇順を使用する必要があります。
8. クエリのフィルターセクションは空のままでかまいません。設定全体を下図に示します。



9. 「アラームの編集」 ボタンを使用すると、現在のアラームのしきい値を変更するダイアログを表示できます。
10. 「保存」 を選択して新しいしきい値を設定します。



> **Note**
>
> このパネルは、現在を含むリアルタイムの時間範囲でのみ使用してください。過去の終了および開始時間範囲を使用すると、アラームのしきい値を常に現在のしきい値として編集する際に、予期しない値が表示されることがあります。

AWS IoT TwinMaker マターポート統合

Matterporには、現実の環境をスキャンし、Matterportデジタルツインとも呼ばれる没入型3Dモデルを作成するためのさまざまなキャプチャオプションが用意されています。これらのモデルはMatterportスペースと呼ばれます。 AWS IoT TwinMaker はMatterportインテグレーションをサポートしているため、Matterportデジタルツインを AWS IoT TwinMaker のシーンにインポートできます。Matterport デジタルツインをと組み合わせることで AWS IoT TwinMaker、デジタルツインシステムを仮想環境で視覚化して監視できます。

Matterportの使用方法について詳しくは、AWS IoT TwinMaker Matterportページにあ るMatterportのドキュメントを参照してください。

インテグレーションに関するトピック

- インテグレーションの概要
- Matterportインテグレーションの前提条件
- Matterportの認証情報を生成して記録してください
- Matterport の認証情報は以下の場所に保管してください。 AWS Secrets Manager
- Matterport スペースをシーンにインポートします。 AWS IoT TwinMaker
- Grafana ダッシュボードのマターポートスペースを使用してください AWS IoT TwinMaker

- ウェブアプリケーションで Matterport スペースを使用してください。 AWS IoT TwinMaker

インテグレーションの概要

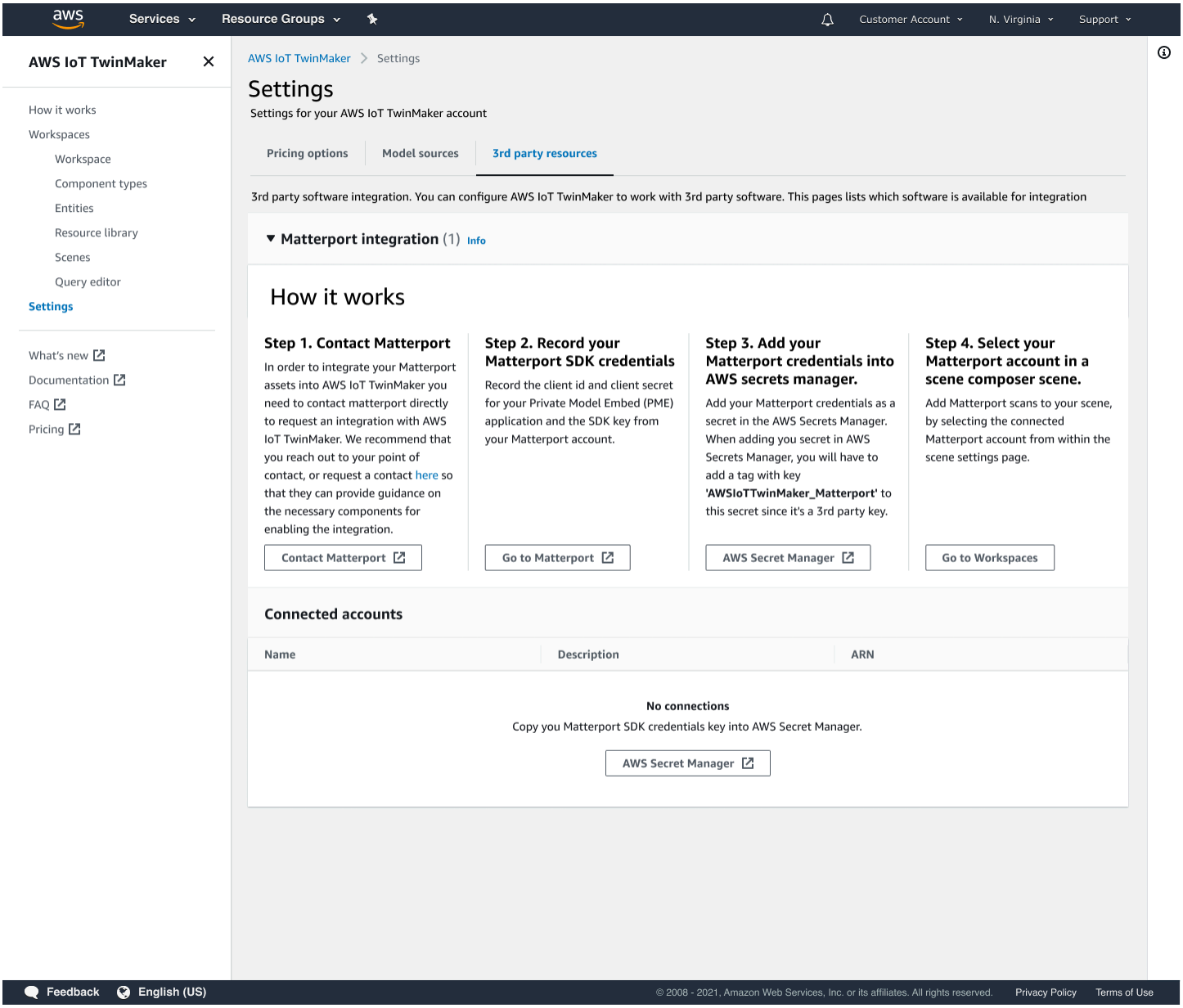
このインテグレーションでは、以下の操作を行うことができます。

- アプリキットの Matterport タグとスペースを使用してください。 AWS IoT TwinMaker
- インポートしたマターポートデータを AWS IoT TwinMaker Grafana ダッシュボードに表示します。 AWS IoT TwinMaker と Grafana の使用方法の詳細については、Grafana ダッシュボード統合ドキュメントをご覧ください。
- Matterport スペースをシーンにインポートします。 AWS IoT TwinMaker
- シーン内のデータにバインドしたい Matterport タグを選択してインポートします。 AWS IoT TwinMaker
- AWS IoT TwinMaker シーン内の Matterport スペースとタグの変更を自動的に表示し、どちらを同期するかを承認します。

インテグレーションプロセスは3つの重要なステップで構成されています。

1. Matterportの認証情報を生成して記録してください
2. Matterport の認証情報は以下の場所に保管してください。 AWS Secrets Manager
3. Matterport スペースをシーンにインポートします。 AWS IoT TwinMaker

AWS IoT TwinMaker コンソールでインテグレーションを開始します。コンソールの「設定」ページで、「サードパーティリソース」の下にある「Matterportインテグレーション」を開き、インテグレーションに必要なさまざまなリソース間を移動します。



Matterportインテグレーションの前提条件

Matterport を統合する前に、 AWS IoT TwinMaker 以下の前提条件を満たしていることを確認してください。

- エンタープライズレベルの Matterport アカウントと、統合に必要な Matterport 製品を購入しました。 AWS IoT TwinMaker
- AWS IoT TwinMaker ワークスペースができました。詳細については、「はじめに」を参照してください AWS IoT TwinMaker。

- AWS IoT TwinMaker ワークスペースのロールが更新されました。ワークスペースロールの作成の詳細については、詳しくは、「AWS IoT TwinMakerのサービスロールの作成と管理」を参照してください。

  次のコードをワークスペースロールに追加します。

```
{
  "Effect": "Allow",
  "Action": "secretsmanager:GetSecretValue",
  "Resource": [
   "AWS Secrets Manager secret ARN"
  ]
}
```

- インテグレーションを有効にするために必要なライセンスを設定するには、Matterportに連絡する必要があります。Matterportはまた、インテグレーションのためのプライベートモデルエンベッド（PME）も可能にします。

  既にMatterportのアカウントマネージャーがいる場合は、直接担当者に連絡してください。

  Matterportの担当者がいない場合は、以下の手順でMatterportに連絡し、インテグレーションをリクエストしてください。

  1. Matterportと AWS IoT TwinMakerページを開きます。
  2. 「お問い合わせ」ボタンを押して、お問い合わせフォームを開きます。
  3. 必要な情報をフォームに入力します。
  4. 準備ができたら、「こんにちは」を選択してMatterportにリクエストを送信してください。

  インテグレーションをリクエストすると、インテグレーションプロセスを続行するために必要なMatterport SDKとプライベートモデルエンベッド（PME）認証情報を生成できます。

> **Note**
>
> これには、新しい製品やサービスの購入に手数料がかかる場合があります。

Matterportの認証情報を生成して記録してください

Matterport をと統合するには AWS IoT TwinMaker、Matterport AWS Secrets Manager の認証情報を提供する必要があります。次の手順に従って、Matterport SDK認証情報を生成します。

1. Matterportアカウントにログインします。
2. アカウントの設定ページに移動します。
3. 設定ページに移動したら、「開発者ツール」オプションを選択します。
4. 「開発者ツール」ページで、「SDKキー管理」セクションに移動します。
5. 「SDKキー管理」セクションに移動したら、新しいSDKキーを追加するオプションを選択します。
6. Matterport SDK キーを取得したら、Grafana AWS IoT TwinMaker サーバーのキーにドメインを追加します。 AWS IoT TwinMaker アプリキットを使用している場合は、必ずカスタムドメインも追加してください。
7. 次に、「アプリケーションインテグレーション管理」セクションを見つけると、「PMEアプリケーションの一覧」が表示されているはずです。以下の情報を記録してください。

   - クライアントID
   - クライアントシークレット

   > **Note**
   >
   > クライアントシークレットは一度しか表示されないため、クライアントシークレットを記録することを強くお勧めします。Matterportインテグレーションを続行するには、AWS Secrets Manager コンソールで「クライアントシークレット」を提示する必要があります。

   これらの認証情報は、必要なコンポーネントを購入し、アカウントのPMEがMatterportによって有効化された時点で自動的に作成されます。これらの認証情報が表示されない場合は、Matterportにお問い合わせください。お問い合わせは、Matterportおよび AWS IoT TwinMakerお問い合わせフォームをご覧ください。

Matterport SDK認証情報について詳しくは、Matterportの公式SDKドキュメントSDKドキュメントの概要をご覧ください。

Matterport の認証情報は以下の場所に保管してください。 AWS Secrets Manager

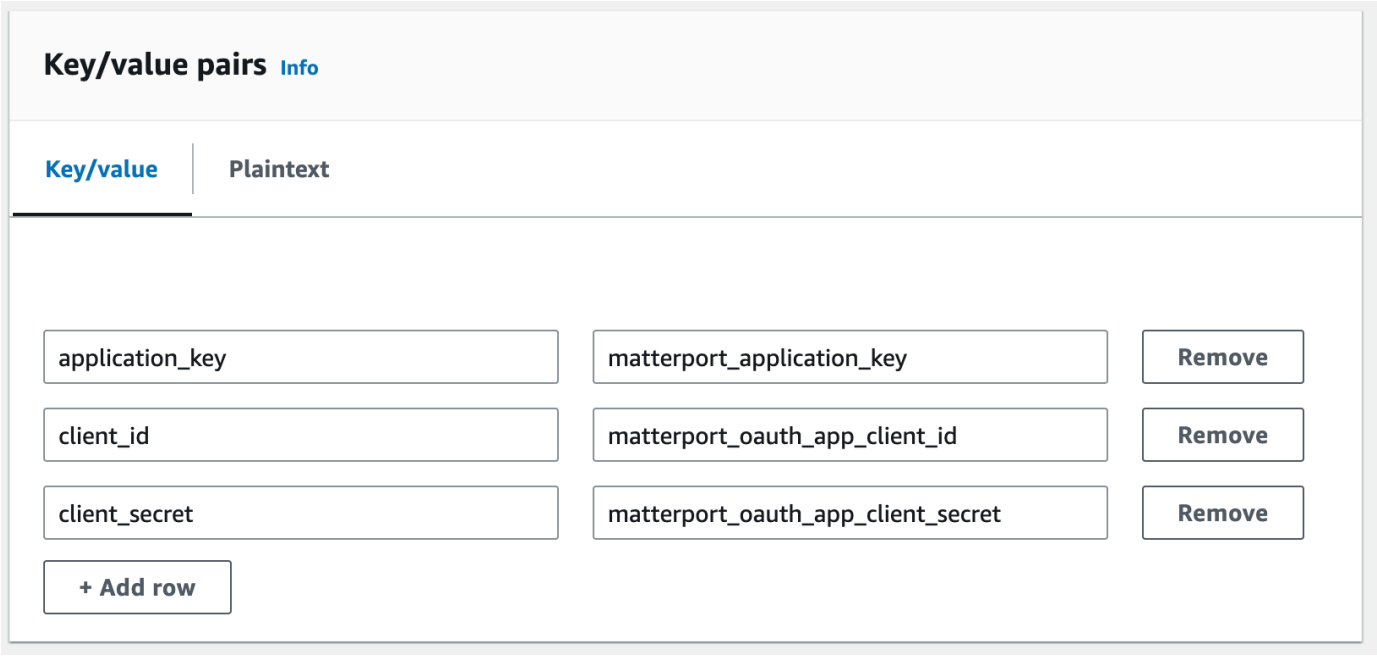
以下の手順に従って Matterport 認証情報をに保存してください。 AWS Secrets Manager

> **Note**
>
> Matterportインテグレーションを続行するには、Matterportの認証情報を生成して記録してくださいトピックの手順で作成した「クライアントシークレット」が必要です。

1. コンソールにログインします。 AWS Secrets Manager
2. 「シークレット」ページに移動し、「新しいシークレットを保存」を選択します。
3. 「シークレットタイプの選択」で、「他の種類のシークレット」を選択します。
4. 「キー/値ペア」セクションで、Matterportの認証情報を値として、以下のキーと値のペアを追加します。

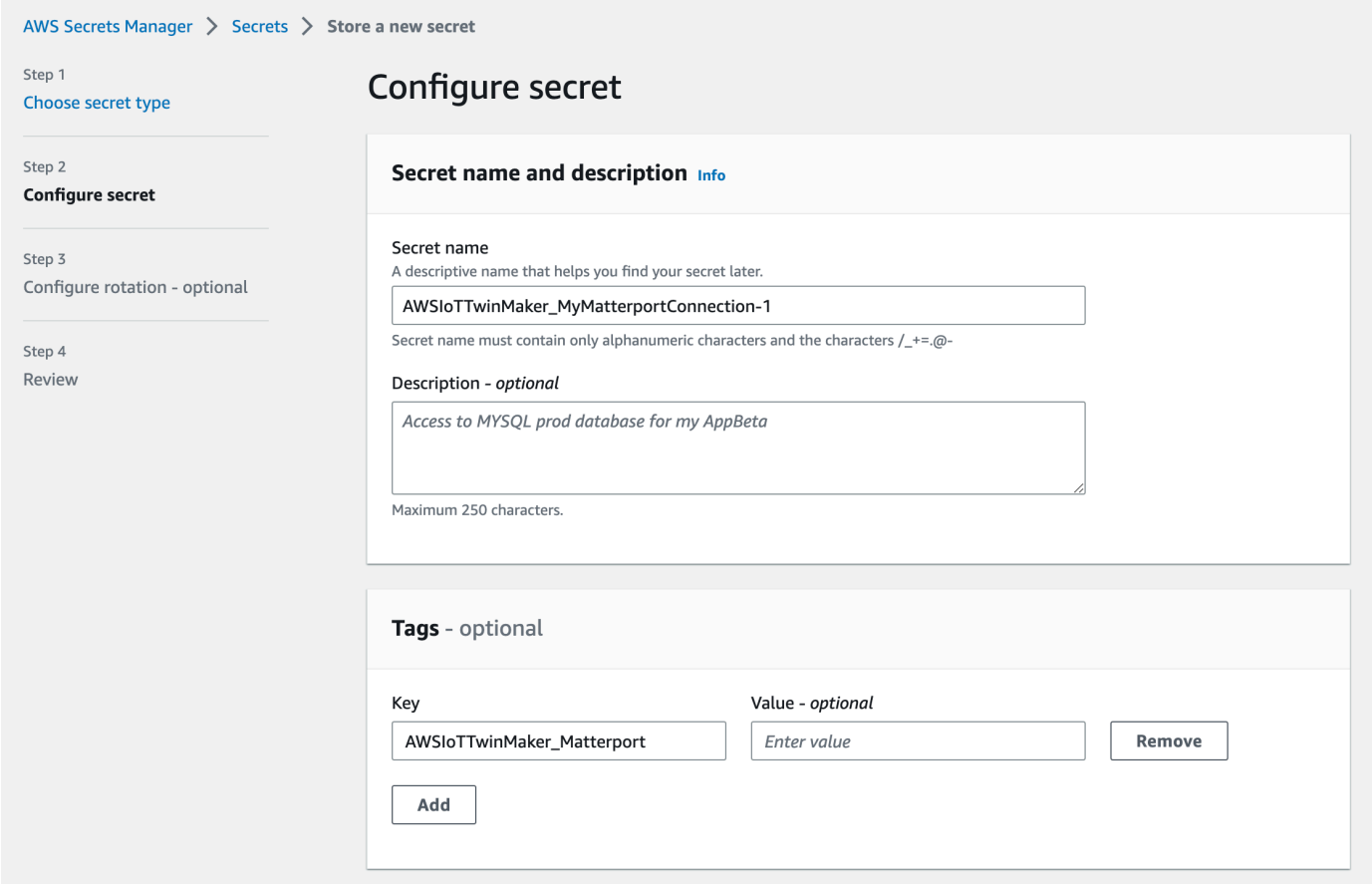
   - キー：`application_key`と値：*`<Matterport####>`*を使用してキーと値のペアを作成します。
   - キー：`client_id`と値：*`<Matterport####>`*を使用してキーと値のペアを作成します。
   - キー：`client_secret`と値：*`<Matterport####>`*を使用してキーと値のペアを作成します。

   完了したら、以下の例のような設定になっているはずです。



5. 「暗号化キー」については、デフォルトの暗号化キーaws/secretsmanagerを選択したままにしておくことができます。
6. 「次へ」を選択して「シークレットの設定」ページに進みます。
7. 「シークレット名」と「説明」のフィールドに入力します。
8. 「タグ」セクションで、このシークレットにタグを追加します。

   タグを作成するときは、AWSIoTTwinMaker_Matterport次のスクリーンショットのようにキーを割り当てます。



> **Note**
>
> タグを追加する必要があります。タグはオプションとして記載されていますが、サードパーティシークレットを AWS Secrets Managerに追加する場合はタグが必要です。

「値」フィールドはオプションです。キーを入力したら、「追加」を選択して次のステップに進むことができます。

9. 次へを選択して、ローテーションの設定ページに進みます。シークレットローテーションの設定は任意です。シークレットの追加を完了したいが、ローテーションは必要ない場合は、もう一度「次へ」を選択してください。シークレットローテーションについて詳しくは、「シークレットローテーション」を参照してください。 AWS Secrets Manager
10. レビューページでシークレットの設定を確認します。シークレットを追加する準備ができたら、「保存」を選択します。

の使用について詳しくは AWS Secrets Manager、 AWS Secrets Manager 以下のドキュメントを参照してください。

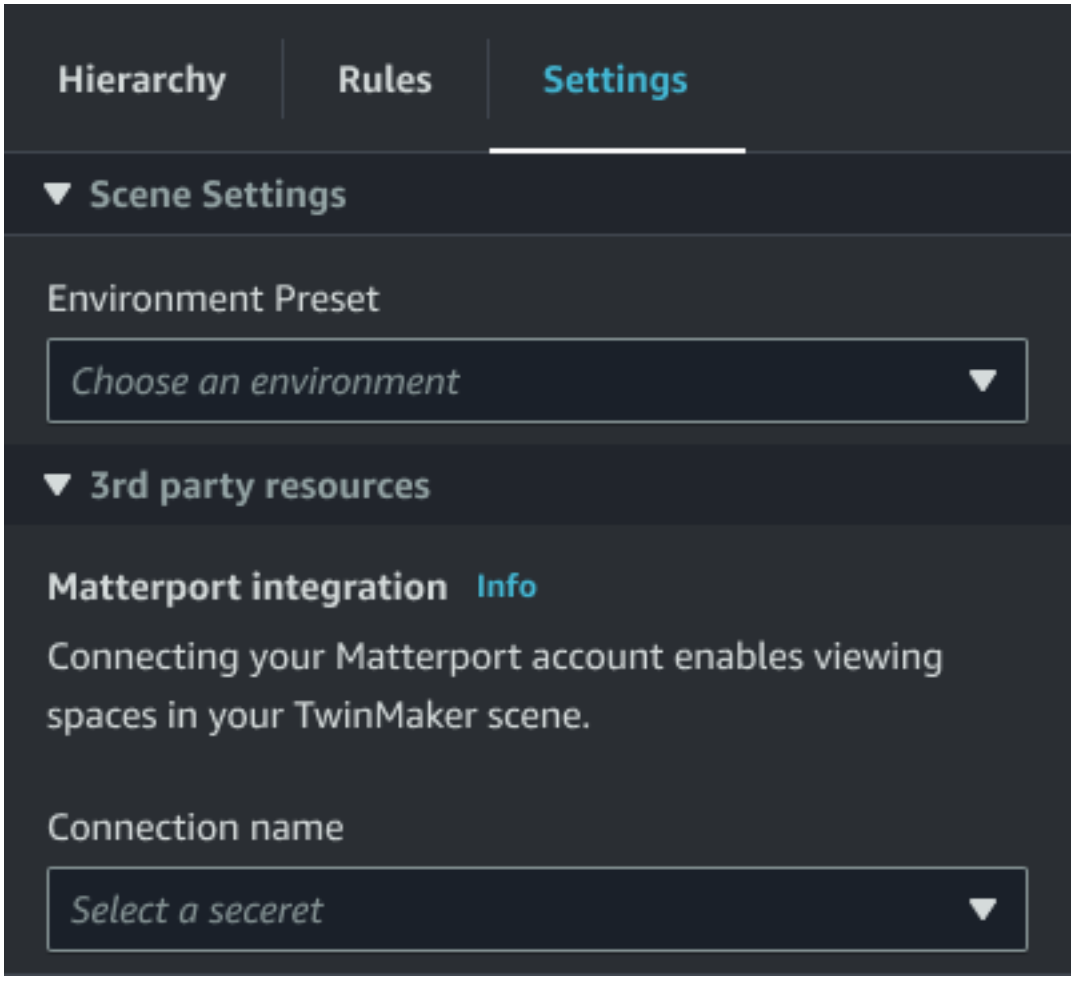
- シークレットの作成と管理には以下を使用します。 AWS Secrets Manager
- とは何ですか AWS Secrets Manager?
- AWS Secrets Manager シークレットをローテーションする

これで Matterport AWS IoT TwinMaker アセットをシーンにインポートする準備ができました。次のセクションMatterport スペースをシーンにインポートします。 AWS IoT TwinMakerの手順を参照してください。

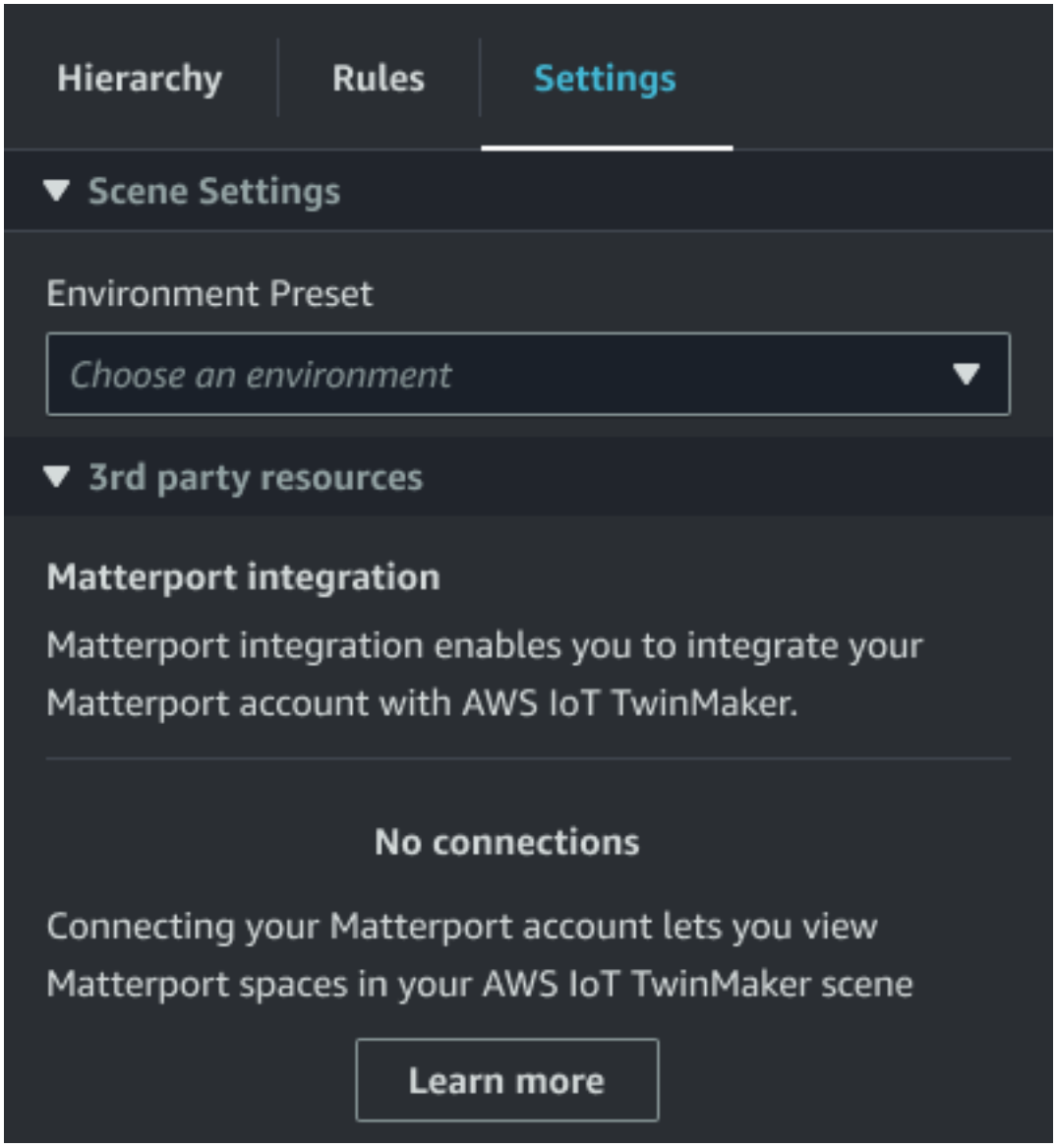
Matterport スペースをシーンにインポートします。 AWS IoT TwinMaker

シーン設定ページから接続されたMatterportアカウントを選択し、シーンにMatterportスキャンを追加してくます。次の手順に従って、Matterportのスキャンとタグをインポートします。

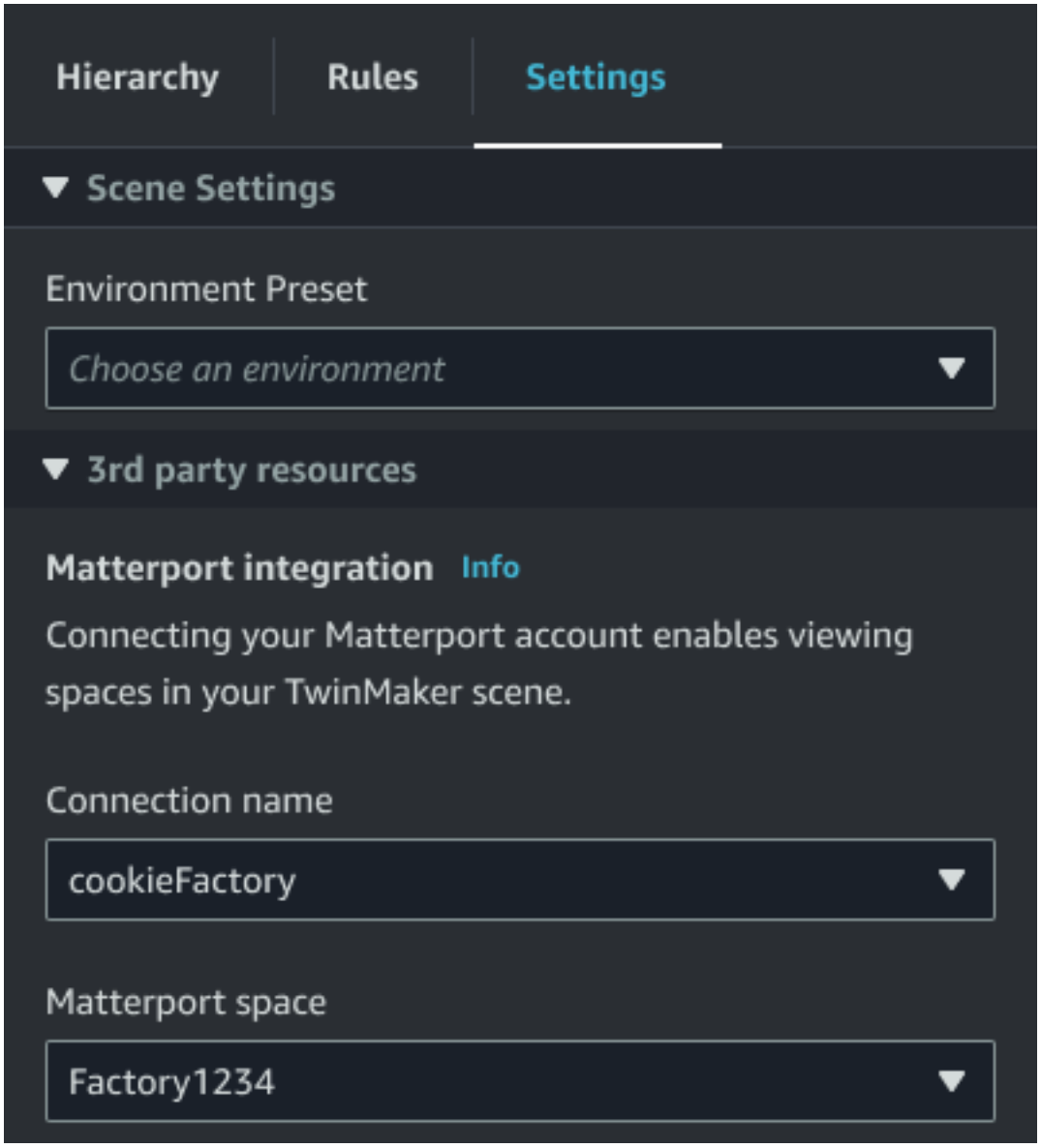
1. AWS IoT TwinMaker コンソールにログインします。
2. Matterport AWS IoT TwinMaker スペースを使用したいシーンを作成するか、既存のシーンを開きます。
3. シーンが開いたら、「設定」タブに移動します。
4. 「設定」の「サードパーティリソース」で、「接続名」を探し、Matterport の認証情報は以下の場所に保管してください。 AWS Secrets Managerの手順で作成したシークレットを入力します。



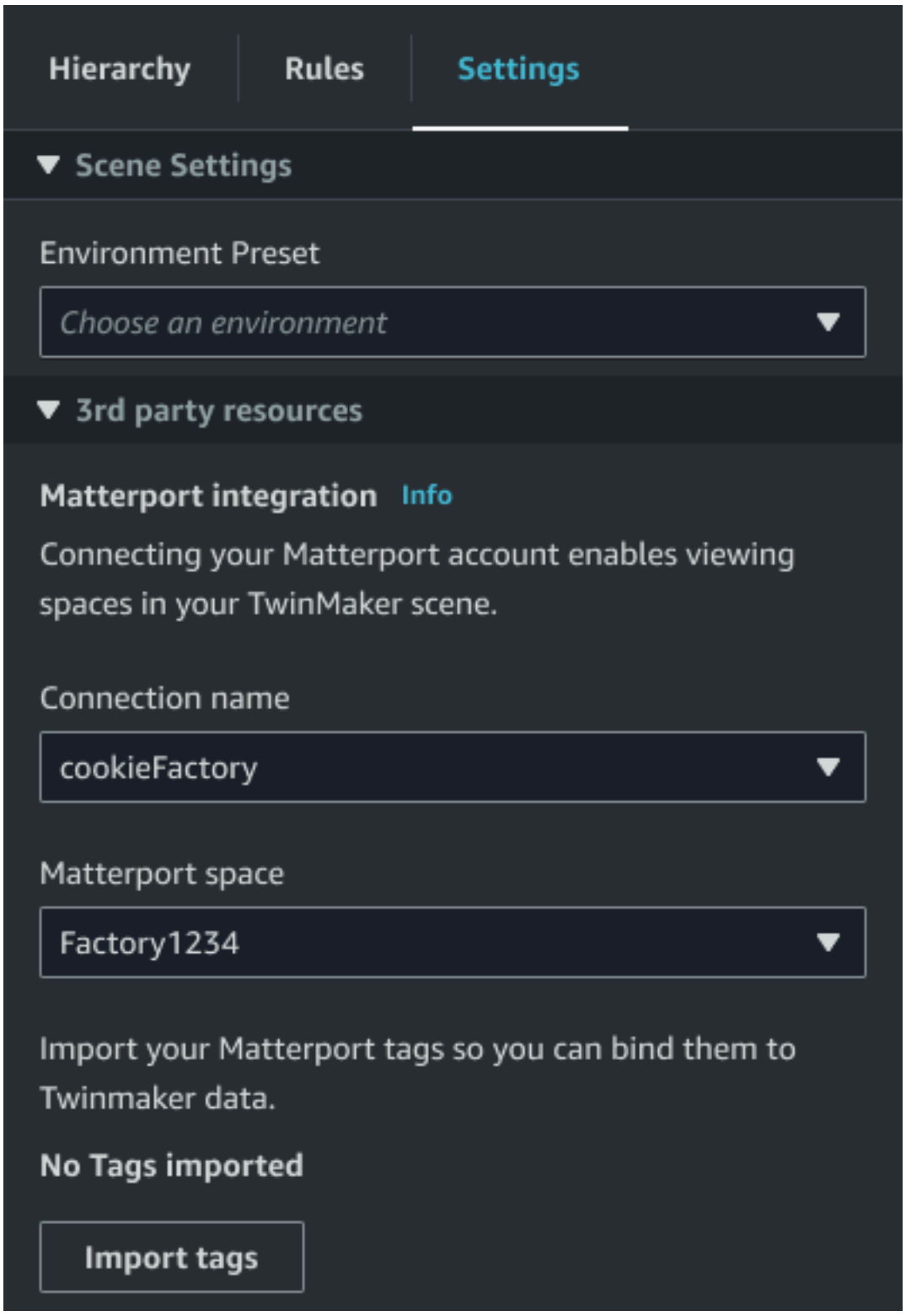
> **Note**
>
> 「接続なし」というメッセージが表示された場合は、AWS IoT TwinMaker コンソール設定ページに移動してMatterportインテグレーションのプロセスを開始してください。



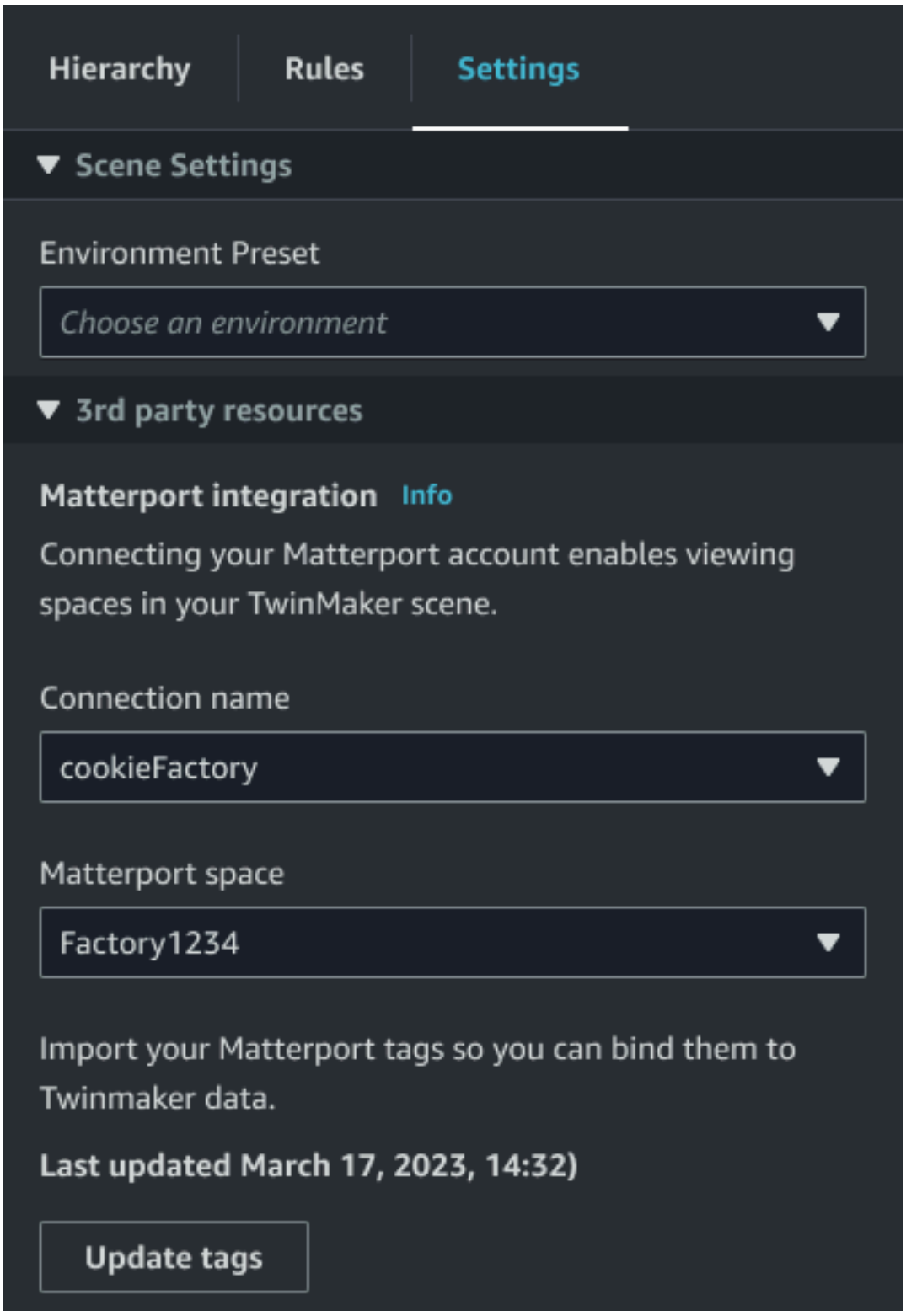
5. 次に、シーンで使用したいMatterportスペースを「Matterportスペース」ドロップダウンから選択します。



6. スペースを選択したら、Matterport タグをインポートし、[タグのインポート] AWS IoT TwinMaker ボタンを押すことでシーンタグに変換できます。



Matterportタグをインポートすると、ボタンは「タグの更新」ボタンに置き換わります。Matterport タグは、Matterport AWS IoT TwinMaker アカウントの最新の変更が常に反映されるように、継続的に更新できます。



7. Matterport AWS IoT TwinMaker との統合が完了し、 AWS IoT TwinMaker シーンにインポートした Matterport スペースとタグの両方が含まれるようになりました。このシーン内では、他のシーンと同じように作業できます。 AWS IoT TwinMaker

シーンの操作について詳しくは、「 AWS IoT TwinMaker シーンの作成と編集 AWS IoT TwinMaker」を参照してください。

Grafana ダッシュボードのマターポートスペースを使用してください AWS IoT TwinMaker

Matterport スペースをシーンにインポートすると、 AWS IoT TwinMaker そのシーンを Grafana ダッシュボードの Matterport スペースで表示できます。ですでに Grafana を設定している場合は AWS IoT TwinMaker、Grafana ダッシュボードを開くだけで、インポートされた Matterport スペースでシーンを表示できます。

まだ Grafana AWS IoT TwinMaker で設定していない場合は、まず Grafana 統合プロセスを完了してください。Grafana AWS IoT TwinMaker と統合する場合、2 つの選択肢があります。セルフマネージドGrafanaインスタンスを使用することも、Amazon Managed Grafanaを使用することもできます。

Grafanaのオプションとインテグレーションプロセスの詳細については、以下のドキュメントを参照してください。

- AWS IoT TwinMaker Grafana ダッシュボードの統合。
- Amazon Managed Grafana。
- セルフマネージドGrafana。

ウェブアプリケーションで Matterport スペースを使用してください。 AWS IoT TwinMaker

Matterport AWS IoT TwinMaker スペースをシーンにインポートすると、そのシーンをアプリキットの Web アプリケーションの Matterport スペースで表示できます。 AWS IoT

アプリケーションキットの使用方法について詳しくは、以下のドキュメントを参照してください。 AWS IoT

- AWS IoT TwinMaker AWS IoT アプリケーションキットでの使用について詳しくは、を参照してくださいAWS IoT TwinMaker UI コンポーネントを使用してカスタマイズされたウェブアプリケーションを作成する。
- アプリケーションキットの使用方法について詳しくは、 AWS IoT アプリケーションキット Github ページをご覧くださいAWS IoT 。

- AWS IoT アプリケーションキットを使用して新しい Web アプリケーションを起動する方法については、公式の IoT App Kit ドキュメントページをご覧ください。

AWS IoT TwinMakerビデオインテグレーション

ビデオカメラはデジタルツインシミュレーションの好機です。AWS IoT TwinMakerを使用して、カメラの位置や物理的状態をシミュレートできます。現場のカメラ用にAWS IoT TwinMakerで エンティティを作成し、ビデオコンポーネントを使用して、ライブビデオとメタデータをサイトから AWS IoT TwinMakerシーンまたはGrafanaダッシュボードにストリーミングします。

AWS IoT TwinMakerは、エッジデバイスから2つの方法でビデオをキャプチャできます。Kinesis Video Streams用のエッジコネクタを使用してエッジデバイスからビデオをストリーミングすることも、エッジデバイスにビデオを保存してMQTTメッセージでビデオのアップロードを開始することもできます。AWS IoTサービスで使用するために、デバイスからビデオデータをストリーミングするには、このコンポーネントを使用します。必要なリソースを生成し、Kinesis Video Streams のエッジコネクタをデプロイするには、 の「Kinesis Video Streams のエッジコネクタの開始方法」を参照してください GitHub。AWS IoT Greengrassコンポーネントの詳細については、Kinesis Video Streams のエッジコネクタに関するAWS IoT Greengrassドキュメントを参照してください。

必要AWS IoT SiteWiseなモデルを作成し、Kinesis Video Streams Greengrassコンポーネントを設定したら、AWS IoT TwinMakerコンソールのデジタルツインアプリケーションにエッジでビデオをストリーミングまたは録画できます。デバイスからのライブストリームとメタデータをGrafanaダッシュボードで表示することもできます。GrafanaとAWS IoT TwinMakerのインテグレーションについての詳細は、AWS IoT TwinMaker Grafana ダッシュボードの統合を参照してください。

AWS IoT TwinMakerでビデオをストリーミングするには、Kinesis Video Streams用のエッジコネクタを使用

Kinesis Video Streams用のエッジコネクタを使用すると、AWS IoT TwinMakerシーン内のエンティティにビデオとデータをストリーミングできます。これにはビデオコンポーネントを使用します。シーンで使用するビデオコンポーネントを作成するには、次の手順を実行します。

前提条件

AWS IoT TwinMakerシーンにビデオコンポーネントを作成する前に、次の前提条件を満たしていることを確認してください。

- Kinesis Video Streamsのエッジコネクタに必要なAWS IoT SiteWiseモデルとアセットを作成しました。コネクタのAWS IoT SiteWiseアセット作成の詳細については、「Kinesis Video Streams用エッジコネクタを始める」を参照してください。

- KinesisAWS IoT GreengrassVideo Streamsエッジコネクタをデバイスにデプロイ。Kinesis Video Streamsエッジコネクタコンポーネントのデプロイについて詳しくは、デプロイREADMEを参照してください。

AWS IoT TwinMakerシーン用のビデオコンポーネントの作成

次の手順を実行して、シーンのKinesis Video Streamsコンポーネントのエッジコネクタを作成します。

1. AWS IoT TwinMakerコンソールで、ビデオコンポーネントを追加するシーンを開きます。
2. シーンが開いたら、既存のエンティティを選択するか、コンポーネントを追加するエンティティを作成して、「コンポーネントの追加」を選択します。
3. 「コンポーネントの追加」ペインでコンポーネントの名前を入力し、「タイプ」にcom.amazon.iotsitewise.connector.edgevideoを選択します。
4. 作成したAWS IoT SiteWiseカメラモデルの名前を選択して、アセットモデルを選択します。この名前は以下の形式でなければなりません：`EdgeConnectorForKVSCameraModel-0abc`、末尾のアルファベットと数字の文字列は、あなたのアセット名と一致すべきです。
5. アセットでは、AWS IoT SiteWiseビデオをストリーミングしたいカメラアセットを選択します。小さなウィンドウが開き、現在のビデオストリームのプレビューが表示されます。

   > **Note**
   >
   > ビデオストリーミングをテストするには、「テスト」を選択します。このテストでは、MQTTイベントを送信して動画ライブストリーミングを開始します。プレーヤーに動画が表示されるまでしばらくお待ちください。

6. エンティティにビデオコンポーネントを追加するには、「コンポーネントの追加」を選択します。

Kinesis Video StreamsからGrafanaダッシュボードにビデオとメタデータを追加

AWS IoT TwinMakerシーン内のエンティティのビデオコンポーネントを作成したら、ライブストリームを表示するようにGrafanaのビデオパネルを設定できます。AWS IoT TwinMakerとGrafana

が適切にインテグレートされていることを確認してください。詳細については、「AWS IoT TwinMaker Grafana ダッシュボードの統合」を参照してください。

> **Important**
>
> Grafana ダッシュボードで動画を表示するには、Grafana データソースに適切な IAM アクセス許可があることを確認する必要があります。必要なロールとポリシーを作成するには、ダッシュボード IAM ロールの作成を参照してください。

次の手順を実行して、Kinesis Video StreamsとメタデータをGrafanaダッシュボードに表示します。

1. AWS IoT TwinMakerダッシュボードを開きます。
2. 「パネルを追加」を選択し、「空のパネルを追加」を選択します。
3. パネルリストから、AWS IoT TwinMakerビデオプレーヤーパネルを選択します。
4. AWS IoT TwinMaker ビデオプレーヤーパネルで、ビデオのストリーミング元の Kinesis ビデオストリームの名前KinesisVideoStreamNameで、 のストリーム名を入力します。

   > **Note**
   >
   > Grafanaビデオパネルにメタデータをストリーミングするには、まずビデオストリーミングコンポーネントを含むエンティティを作成する必要があります。

5. オプション：AWS IoT SiteWiseアセットからビデオプレーヤーにメタデータをストリーミングするには、エンティティで、AWS IoT TwinMakerシーンで作成したAWS IoT TwinMakerエンティティを選択します。コンポーネント名には、AWS IoT TwinMakerシーン内のエンティティ用に作成したビデオコンポーネントを選択します。

AWS IoT TwinMakerFlinkライブラリを使用する

AWS IoT TwinMakerは、デジタルツインで使用する外部データストアにデータを読み書きするために使用できるFlinkライブラリを提供します。

AWS IoT TwinMakerFlinkライブラリは、Managed Service for Apache Flinkのカスタムコネクタとしてインストールし、Managed Service for Apache FlinkのZeppelinノートブックでFlink SQLクエリを実行することで使用します。このノートブックは、継続的に実行されるストリーム処理アプリケーションに昇格できます。ライブラリはAWS IoT TwinMakerコンポーネントを利用してワークスペースからデータを取得します。

AWS IoT TwinMakerFlinkライブラリには以下が必要です。

前提条件

1. シーンとコンポーネントが完全に配置されたワークスペース。AWSサービス（AWS IoT SiteWiseおよびKinesis Video Streams）からのデータには、組み込みコンポーネントタイプを使用します。サードパーティーのソースからのデータ用にカスタムコンポーネントタイプを作成します。詳細については、「???」を参照してください。
2. Managed Service for Apache Flink for Apache Flinkを使用したStudioノートブックの理解。これらのノートブックはApache Zeppelinを搭載し、Apache Flinkフレームワークを使用しています。詳細は、「Managed Service for Apache FlinkでStudioノートブックを使用する」を参照してください。

ライブラリの使用方法については、AWS IoT TwinMakerFlinkライブラリユーザーガイドを参照してください。

AWS IoT TwinMakerサンプルのクイックスタートでAWS IoT TwinMakerをセットアップする手順については、サンプルinsightsアプリケーションのREADMEファイルを参照してください。

AWS IoT TwinMaker でのログ記録とモニタリング

モニタリングは、AWS IoT TwinMaker およびその他の AWS ソリューションの信頼性、可用性、およびパフォーマンスを維持する上で重要な部分です。AWS IoT TwinMaker は、サービスをモニタリングしたり、問題が発生したときに報告したり、必要に応じて自動アクションを実行したりするために以下のモニタリングツールをサポートしています。

- Amazon CloudWatch は、AWS リソースおよび AWS で実行しているアプリケーションをリアルタイムでモニタリングします。メトリクスの収集と追跡、カスタマイズしたダッシュボードの作成、および指定したメトリクスが指定したしきい値に達したときに通知またはアクションを実行するアラームの設定を行うことができます。例えば、CloudWatch で Amazon EC2 インスタンスの CPU 使用率などのメトリクスを追跡し、必要に応じて新しいインスタンスを自動的に起動できます。詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。
- Amazon CloudWatch Logsは、AWS IoT TwinMaker ゲートウェイ、CloudTrail、またはその他のソースのログファイルのモニタリング、保存、アクセス許可を行います。CloudWatch Logs は、ログファイル内の情報をモニタリングし、特定のしきい値が満たされたときに通知します。高い耐久性を備えたストレージにログデータをアーカイブすることも可能です。詳細については、「Amazon CloudWatch Logs ユーザーガイド」を参照してください。
- 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。

トピック

- Amazon CloudWatch メトリクスによる AWS IoT TwinMaker のモニタリング
- AWS CloudTrail による AWS IoT TwinMaker API コールのログ記録

Amazon CloudWatch メトリクスによる AWS IoT TwinMaker のモニタリング

raw データを収集して読み取り可能なほぼリアルタイムのメトリクスに処理する CloudWatch を使用して AWS IoT TwinMaker をモニタリングできます。これらの統計は 15 か月間保持されるため、履歴情報にアクセスし、ウェブアプリケーションまたはサービスの動作をより的確に把握できます。また、特定のしきい値を監視するアラームを設定し、これらのしきい値に達したときに通知を送信した

りアクションを実行したりできます。詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。

AWS IoT TwinMaker は、以下のセクションにリストされているメトリクスとディメンションを AWS/IoTTwinMaker 名前空間に公開します。

> **Tip**
>
> AWS IoT TwinMaker は、メトリクスを 1 分間隔で公開します。CloudWatch コンソールでこれらのメトリクスをグラフで表示する場合は、「期間」 を 「1 分」 にすることをお勧めします。

目次

メトリクス

AWS IoT TwinMaker は以下のメトリクスを公開します。

メトリクス

メトリクス	説明
ComponentTypeCreationFailure	このメトリクスは、コンポーネントタイプの作成が成功したかどうかを報告します。 このメトリクスは、コンポーネントタイプがCREATING の状態になった場合に公開されます。これは、スキーマイニシャライザーで必要なプロパティを使用してコンポーネントタイプが作成され、これらのプロパティがデフォルト値でインスタンス化された場合に発生します。 メトリクス値は 0 の成功または 1 の失敗のどちらでもかまいません。 ディメンション: ComponentTypeId、WorkspaceId。

メトリクス	説明
	単位: カウント
ComponentTypeUpdateFailure	このメトリクスは、コンポーネントタイプの更新が成功したかどうかを報告します。 このメトリクスは、コンポーネントタイプが UPDATING の状態になった場合に公開されます。これは、コンポーネントタイプがスキーマイニシャライザーで必要なプロパティで更新され、これらのプロパティがデフォルト値でインスタンス化された場合に発生します。 メトリクス値は 0 の成功または 1 の失敗のどちらでもかまいません。 ディメンション: ComponentTypeId、WorkspaceId。 単位: カウント
EntityCreationFailure	このメトリクスは、エンティティの作成が成功したかどうかを報告します。このメトリクスは、エンティティが CREATING の状態になった場合に公開されます。これは、エンティティがコンポーネントで作成された場合に発生します。 メトリクス値は 0 の成功または 1 の失敗のどちらでもかまいません。 ディメンション: EntityName、EntityId、WorkspaceId。 単位: カウント

メトリクス	説明
EntityUpdateFailure	このメトリクスは、エンティティの更新が成功したかどうかを報告します。このメトリクスは、エンティティが UPDATING の状態になった場合に公開されます。これはエンティティが更新された場合に発生します。 メトリクス値は 0 の成功または 1 の失敗のどちらでもかまいません。 ディメンション: EntityName、EntityId、WorkspaceId。 単位: カウント
EntityDeletionFailure	このメトリクスは、エンティティの削除が成功したかどうかを報告します。このメトリクスは、エンティティが DELETING の状態になった場合に公開されます。これはエンティティが削除された場合に発生します。 メトリクス値は 0 の成功または 1 の失敗のどちらでもかまいません。 ディメンション: EntityName、EntityId、WorkspaceId。 単位: カウント

> **Tip**
>
> すべてのメトリクスは、CloudWatch の AWS/IoTTwinMaker 名前空間に公開されます。

AWS CloudTrail による AWS IoT TwinMaker API コールのログ記録

AWS IoT TwinMaker は AWS CloudTrail と統合されています。このサービスは、ユーザーやロール、または AWS IoT TwinMaker の AWS の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す。CloudTrail は、AWS IoT TwinMaker の API コールをイベントとしてキャプチャします。キャプチャされた呼び出しには、AWS IoT TwinMaker コンソールの呼び出しと、AWS IoT TwinMaker 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、AWS IoT TwinMaker のイベントなど、Amazon S3 バケットへの CloudTrail イベントの継続的な配信を有効にすることができます 証跡を設定しない場合でも、「CloudTrail」 コンソールの 「イベント履歴」 で最新のイベントを表示できます。CloudTrail によって収集された情報を使用して、AWS IoT TwinMaker に対して行われた要求、要求が行われた IP アドレス、要求を行った人、要求が行われた日時、および追加の詳細を判別できます。

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

CloudTrail での AWS IoT TwinMaker 情報

CloudTrail は AWS アカウントの作成時に自動的に有効になります。サポートするイベントアクティビティが AWS IoT TwinMaker で発生すると、CloudTrail はイベント履歴の他の AWS サービスのイベントと共にそのイベントアクティビティを記録します。AWS アカウントで最近のイベントを表示、検索、ダウンロードできます。詳細については、「CloudTrail イベント履歴でのイベントの表示」をご参照ください。

AWS のイベントなど、AWS IoT TwinMaker アカウントのイベントの継続的なレコードについては、追跡を作成します。証跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、コンソールで証跡を作成するときに、証跡がすべての AWS リージョンに適用されます。CloudTrail 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 ログで収集したイベントデータをより詳細に分析し、それに基づく対応するためにその他の AWS のサービスを設定できます。詳細については、次を参照してください。

- 追跡を作成するための概要
- CloudTrail がサポートされているサービスと統合
- CloudTrail の Amazon SNS 通知の設定
- 複数のリージョンからの CloudTrail ログファイルの受け取りと複数のアカウントからの CloudTrail ログファイルの受け取り

ほとんどの AWS IoT TwinMaker オペレーションは CloudTrail によって口グに記録され、AWS IoT TwinMaker API リファレンスに記録されます。

次のデータプレーンオペレーションは、CloudTrail によって口グ記録されません。

- GetPropertyValue
- GetPropertyValueHistory
- BatchPutPropertyValues

各イベントまたはログエントリには、リクエストの生成者に関する情報が含まれます。ID 情報は次の判断に役立ちます。

- リクエストが、ルートとユーザー認証情報のどちらを使用して送信されたか。
- リクエストが、ロールとフェデレーションユーザーの一時的なセキュリティ認証情報のどちらを使用して送信されたか。
- リクエストが、別の AWS の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。

のセキュリティ AWS IoT TwinMaker

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

セキュリティは、 AWS とユーザーの間で共有される責任です。責任共有モデルではこれを、クラウドのセキュリティ、およびクラウド内でのセキュリティと説明しています:

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

このドキュメントは、 の使用時に責任共有モデルを適用する方法を理解するのに役立ちます AWS IoT TwinMaker。以下のトピックでは、セキュリティおよびコンプライアンスの目的を達成するために AWS IoT TwinMaker を設定する方法を示します。また、 AWS IoT TwinMaker リソースのモニタリングや保護に役立つ他の AWS のサービスの使用方法についても説明します。

トピック

- でのデータ保護 AWS IoT TwinMaker
- の Identity and Access Management AWS IoT TwinMaker
- AWS IoT TwinMaker およびインターフェイス VPC エンドポイント (AWS PrivateLink)
- のコンプライアンス検証 AWS IoT TwinMaker
- の耐障害性 AWS IoT TwinMaker
- のインフラストラクチャセキュリティ AWS IoT TwinMaker

でのデータ保護 AWS IoT TwinMaker

責任 AWS 共有モデル、 でのデータ保護に適用されます AWS IoT TwinMaker。このモデルで説明されているように、 AWS はすべての を実行するグローバルインフラストラクチャを保護する責任があります AWS クラウド。お客様は、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。また、使用する AWS のサービス のセキュリティ設定と管理タスクもユーザーの責任となります。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記事「AWS 責任共有モデルおよび GDPR」を参照してください。

データ保護の目的で、 認証情報を保護し AWS アカウント 、 AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 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 は必須であり TLS 1.3 がお勧めです。
- を使用して API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。
- AWS 暗号化ソリューションと、 内のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
- 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 AWS を介して にアクセスするときに FIPS 140-2 検証済みの暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-2」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報は、タグ、または名前フィールドなどの自由形式のテキストフィールドに配置しないことを強くお勧めします。これは、コンソール、API、AWS IoT TwinMaker または SDK を使用して AWS CLIまたは他の AWS のサービス を操作する場合も同様です。 AWS SDKs 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。外部サーバーへの URL を提供する場合は、そのサーバーへのリクエストを検証するための認証情報を URL に含めないように強くお勧めします。

保管中の暗号化

AWS IoT TwinMaker は、必要に応じて、サービスが作成する Amazon S3 バケットにワークスペース情報を保存します。サービスが自動的に作成するバケットでは、サーバー側の暗号化がデフォルトで有効になっています。新しいワークスペースを作成するときに独自の Amazon S3 バケットを使用する場合は、デフォルトのサーバー側の暗号化の有効化をお勧めします。Amazon S3 でのデフォルトの暗号化の詳細については、「Amazon S3 バケットのデフォルトのサーバー側の暗号化動作の設定」を参照してください。

転送中の暗号化

に送信されるすべてのデータは AWS IoT TwinMaker 、HTTPS プロトコルを使用して TLS 接続を介して送信されるため、転送中はデフォルトで保護されます。

> **Note**
>
> が Amazon S3 バケットと AWS IoT TwinMaker やり取りするときに転送中の暗号化を強制するコントロールとして、Amazon S3 バケットアドレスで HTTPS を使用することをお勧めします。Amazon S3 バケットの詳細については、「Amazon S3 バケットの作成、設定、操作」を参照してください。

の Identity and Access Management AWS IoT TwinMaker

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 AWS のサービス するのに役立つ です。IAM 管理者は、誰を認証 (サインイン) し、誰に AWS IoT TwinMaker リソースの使用を承認する (アクセス許可を付与する) かを制御します。IAM は、追加料金なしで AWS のサービス 使用できる です。

トピック

- 対象者
- アイデンティティを使用した認証
- ポリシーを使用したアクセスの管理
- が IAM と AWS IoT TwinMaker 連携する方法
- のアイデンティティベースのポリシーの例 AWS IoT TwinMaker

- AWS IoT TwinMaker ID とアクセスのトラブルシューティング
- のサービスにリンクされたロールの使用 AWS IoT TwinMaker
- AWS の マネージドポリシー AWS IoT TwinMaker

対象者

AWS Identity and Access Management (IAM) の使用方法は、 で行う作業によって異なります AWS IoT TwinMaker。

サービスユーザー – AWS IoT TwinMaker サービスを使用してジョブを実行する場合、管理者から必要な認証情報とアクセス許可が与えられます。さらに多くの AWS IoT TwinMaker 機能を使用して作業を行う場合は、追加のアクセス許可が必要になることがあります。アクセスの管理方法を理解すると、管理者から適切な権限をリクエストするのに役に立ちます。 AWS IoT TwinMaker機能にアクセスできない場合は、「AWS IoT TwinMaker ID 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 AWS IoT TwinMaker リソースを担当している場合は、通常、 へのフルアクセスがあります AWS IoT TwinMaker。サービスユーザーがどの AWS IoT TwinMaker 機能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、IAM 管理者にリクエストを送信して、サービスユーザーの権限を変更する必要があります。このページの情報を点検して、IAM の基本概念を理解してください。会社で IAM を で使用する方法の詳細については、 AWS IoT TwinMaker「」を参照してくださいが IAM と AWS IoT TwinMaker 連携する方法。

IAM 管理者 - 管理者は、 AWS IoT TwinMakerへのアクセスを管理するポリシーの書き込み方法の詳細について確認する場合があります。IAM で使用できる AWS IoT TwinMaker 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、「」を参照してくださいのアイデンティティベースのポリシーの例 AWS IoT TwinMaker。

アイデンティティを使用した認証

認証とは、ID 認証情報 AWS を使用して にサインインする方法です。として、IAM ユーザーとして AWS アカウントのルートユーザー、または IAM ロールを引き受けて認証（ にサインイン AWS) される必要があります。

ID ソースを介して提供された認証情報を使用して、フェデレーティッド ID AWS として にサインインできます。 AWS IAM Identity Center（ IAM Identity Center) ユーザー、会社のシングルサインオン認証、Google または Facebook の認証情報は、フェデレーティッド ID の例です。フェデレーティッ

ドアイデンティティとしてサインインする場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。フェデレーション AWS を使用して にアクセスすると、間接的にロールを引き受けることになります。

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。へのサインインの詳細については AWS、「 ユーザーガイド」の「 へのサインイン AWS アカウント方法AWS サインイン 」を参照してください。

AWS プログラムで にアクセスする場合、 は Software Development Kit (SDK) とコマンドラインインターフェイス (CLI) AWS を提供し、認証情報を使用して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リクエストを自分で署名する方法の詳細については、IAM ユーザーガイドの API AWS 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、追加セキュリティ情報の提供をリクエストさ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向上させることをお勧めします。詳細については、『AWS IAM Identity Center ユーザーガイド』の「Multi-factor authentication」(多要素認証) および『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 ルートユーザー

を作成するときは AWS アカウント、アカウント内のすべての およびリソースへの AWS のサービス 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 ID は AWS アカウント 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でアクセスできます。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「ルートユーザー認証情報が必要なタスク」を参照してください。

フェデレーティ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、一時的な認証情報を使用して にアクセスするための ID プロバイダーとのフェデレーションの使用を要求 AWS のサービス します。

フェデレーティッド ID 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、Identity Center ディレクトリのユーザー、または ID ソースを通じて提供さ

れた認証情報 AWS のサービス を使用して にアクセスするユーザーです。フェデレーティッド ID がにアクセスすると AWS アカウント、ロールを引き受け、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 Identity Center でユーザーとグループを作成することも、独自の ID ソース内のユーザーとグループのセットに接続して同期して、すべての AWS アカウント とアプリケーションで使用することもできます。IAM Identity Center の詳細については、『AWS IAM Identity Center ユーザーガイド』の「What is IAM Identity Center?」(IAM Identity Center 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、単一のユーザーまたはアプリケーションに対して特定のアクセス許可 AWS アカウント を持つ 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、IAM 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。例えば、IAMAdmins 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権限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、特定のアクセス許可 AWS アカウント を持つ 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 を切り替える AWS Management Console ことで、 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、 または AWS API AWS CLI オペレーションを呼び出すか、カスタム URL を使用します。ロールを使用する方法の詳細については、「IAM ユーザーガイド」の「IAM ロールの使用」を参照してください。

IAM 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アクセス – フェデレーティッド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。フェデレーティッド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が付与されます。フェデレーションの詳細については、『IAM ユーザーガイド』の「サードパーティーアイデンティティプロバイダー向けロールの作成」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、権限セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity Center 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 ユーザー権限 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる権限を一時的に IAM ロールで引き受けることができます。
- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースにアクセスすることを、別のアカウントの人物 (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 では AWS のサービス、(ロールをプロキシとして使用する代わりに) ポリシーをリソースに直接アタッチできます。クロスアカウントアクセスにおける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との相違点」を参照してください。
- クロスサービスアクセス — 一部の は、他の の機能 AWS のサービス を使用します AWS のサービス。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスでは、呼び出し元プリンシパルの権限、サービスロール、またはサービスにリンクされたロールを使用してこれを行う場合があります。
  - 転送アクセスセッション (FAS) – IAM ユーザーまたはロールを使用して でアクションを実行する場合 AWS、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、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、 を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストリクエストリクエストと組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。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。
  - サービスロール - 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 のサービスにアクセス許可を委任するロールの作成」を参照してください。

- サービスにリンクされたロール – サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールの権限を表示できますが、編集することはできません。

- Amazon EC2 で実行されてい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、インスタンスにアタッチされたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、『IAM ユーザーガイド』の「Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して権限を付与する」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については、『IAM ユーザーガイド』の「(IAM ユーザーではなく) IAM ロールをいつ作成したら良いのか?」を参照してください。

ポリシーを使用したアクセスの管理

でアクセスを制御する AWS には、ポリシーを作成し、 AWS ID またはリソースにアタッチします。ポリシーは AWS 、アイデンティティまたはリソースに関連付けられているときにアクセス許可を定義する のオブジェクトです。 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きに、これらのポリシー AWS 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。ほとんどのポリシーは JSON ドキュメント AWS として に保存されます。JSON ポリシードキュメントの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。

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、リソースで必要なアクションを実行するための権限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継ぐことができます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの権限を定義します。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。そのポリシーを持つユーザーは、 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のポリシー

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、および条件をコントロールします。アイデンティティベースのポリシーを作成する方法については、「IAM ユーザーガイド」の「IAM ポリシーの作成」を参照してください。

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ています。管理ポリシーは、 内の複数のユーザー、グループ、ロールにアタッチできるスタンドアロンポリシーです AWS アカウント。管理ポリシーには、 AWS 管理ポリシーとカスタマー管理ポリシーが含まれます。マネージドポリシーまたはインラインポリシーのいずれか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比較」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあげられます。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、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

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は、IAM の AWS マネージドポリシーを使用できません。

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。

Amazon S3、 AWS WAF、および Amazon VPC は、ACLs。ACL の詳細については、『Amazon Simple Storage Service デベロッパーガイド』の「アクセスコントロールリスト (ACL) の概要」を参照してください。

その他のポリシータイプ

AWS は、一般的ではない追加のポリシータイプをサポートします。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) に付与できる権限の上限を設定する高度な機能です。エンティティにアクセス許可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アクセス許可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、アクセス許可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、権限は無効になります。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 (SCPs) – SCPs は、 の組織または組織単位 (OU) に対する最大アクセス許可を指定する JSON ポリシーです AWS Organizations。 AWS Organizations は、 AWS アカウント ビジネスが所有する複数の をグループ化して一元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、各 を含むメンバーアカウントのエンティティのアクセス許可を制限します AWS アカウントのルートユーザー。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与する場合にリクエストを許可するかどうか AWS を決定する方法については、IAM ユーザーガイドの「ポリシー評価ロジック」を参照してください。

が IAM と AWS IoT TwinMaker 連携する方法

IAM を使用して へのアクセスを管理する前に AWS IoT TwinMaker、 で使用できる IAM 機能について学びます AWS IoT TwinMaker。

で使用できる IAM の機能 AWS IoT TwinMaker

IAM 機能	AWS IoT TwinMaker サポート
アイデンティティベースのポリシー	Yes
リソースベースのポリシー	No
ポリシーアクション	Yes
ポリシーリソース	Yes
ポリシー条件キー	Yes
ACL	No
ABAC (ポリシー内のタグ)	部分的
一時的な認証情報	Yes
プリンシパル権限	Yes
サービスロール	あり
サービスリンクロール	いいえ

AWS IoT TwinMaker およびその他の AWS のサービスがほとんどの IAM 機能と連携する方法の概要を把握するには、「 AWS IAM Identity Center ユーザーガイド」のAWS 「IAM と連携する のサービス」を参照してください。

のアイデンティティベースのポリシー AWS IoT TwinMaker

アイデンティティベースポリシーをサポートする	Yes

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、および条件をコントロールします。アイデンティティベースのポリシーを作成する方法については、『IAM ユーザーガイド』の「IAM ポリシーの作成」を参照してください。

IAM アイデンティティベースのポリシーでは、許可または拒否するアクションとリソース、およびアクションを許可または拒否す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 ポリシーの要素のリファレンス」を参照してください。

のアイデンティティベースのポリシーの例 AWS IoT TwinMaker

AWS IoT TwinMaker 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、「」を参照してくださいのアイデンティティベースのポリシーの例 AWS IoT TwinMaker。

内のリソースベースのポリシー AWS IoT TwinMaker

リソースベースのポリシーのサポート	No

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には例として、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあげられます。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、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

クロスアカウントアクセスを有効にするには、アカウント全体、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 にある場合 AWS アカウント、信頼されたアカウントの IAM 管理者は、プリンシパルエンティティ (ユーザーまたはロール) にリソースへのアクセス許可も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追加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。

のポリシーアクション AWS IoT TwinMaker

ポリシーアクションに対するサポート	はい

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、関連付けられた 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。

AWS IoT TwinMaker アクションのリストを確認するには、「サービス認証リファレンス」の「 で定義されるアクション AWS IoT TwinMaker」を参照してください。

のポリシーアクションは、アクションの前に次のプレフィックス AWS IoT TwinMaker を使用します。

```
iottwinmaker
```

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": [
   "iottwinmaker:action1",
   "iottwinmaker:action2"
         ]
```

AWS IoT TwinMaker 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、「」を参照してくださいのアイデンティティベースのポリシーの例 AWS IoT TwinMaker。

のポリシーリソース AWS IoT TwinMaker

ポリシーリソースに対するサポート	はい

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どのような条件でアクションを実行できるかということです。

Resource JSON ポリシー要素は、アクションが適用されるオブジェクトを指定します。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": "*"
```

AWS IoT TwinMaker リソースタイプとその ARNs」の「 で定義されるリソース AWS IoT TwinMaker」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、「AWS IoT TwinMakerで定義されるアクション」を参照してください。

AWS IoT TwinMaker 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、「」を参照してくださいのアイデンティティベースのポリシーの例 AWS IoT TwinMaker。

のポリシー条件キー AWS IoT TwinMaker

サービス固有のポリシー条件キーのサポート	はい

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な条件を指定できます。Condition 要素はオプションです。イコールや未満などの 条件演算子 を使用して条件式を作成することで、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を評価します。1 つの条件キーに複数の値を指定すると、 は論理ORオペレーションを使用して条件 AWS を評価します。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 ポリシーの要素: 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。

AWS IoT TwinMaker 条件キーのリストを確認するには、「サービス認証リファレンス」の「 の条件キー AWS IoT TwinMaker」を参照してください。条件キーを使用できるアクションとリソースについては、「 で定義されるアクション AWS IoT TwinMaker」を参照してください。

AWS IoT TwinMaker 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、「」を参照してくださいのアイデンティティベースのポリシーの例 AWS IoT TwinMaker。

AWS IoT TwinMakerのアクセスコントロールリスト (ACL)

ACL のサポート	No

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。

を使用した属性ベースのアクセスコントロール (ABAC) AWS IoT TwinMaker

ABAC (ポリシー内のタグ) のサポート	部分的

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。では AWS、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) および多くの 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、ポリシーの条件要素でタグ情報を提供します。

サービスがすべてのリソースタイプに対して 3 つの条件キーすべてをサポートする場合、そのサービスの値ははいです。サービスが一部のリソースタイプに対してのみ 3 つの条件キーのすべてをサポートする場合、値は「部分的」になります。

ABAC の詳細については、『IAM ユーザーガイド』の「ABAC 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。

での一時的な認証情報の使用 AWS IoT TwinMaker

一時的な認証情報のサポート	はい

一部の は、一時的な認証情報を使用してサインインすると機能 AWS のサービス しません。一時的な認証情報 AWS のサービス を使用する などの詳細については、IAM ユーザーガイドのAWS のサービス「IAM と連携する 」を参照してください。

ユーザー名とパスワード以外の AWS Management Console 方法で にサインインする場合、一時的な認証情報を使用します。例えば、会社の Single Sign-On (SSO) リンク AWS を使用して にアクセスすると、そのプロセスによって一時的な認証情報が自動的に作成されます。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。

一時的な認証情報は、 AWS CLI または AWS API を使用して手動で作成できます。その後、これらの一時的な認証情報を使用して . AWS recommends にアクセスできます AWS。この際、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 の一時的セキュリティ認証情報」を参照してください。

のクロスサービスプリンシパル許可 AWS IoT TwinMaker

フォワードアクセスセッション (FAS) をサポート	はい

IAM ユーザーまたはロールを使用して でアクションを実行すると AWS、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、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、 を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストリクエストリクエストと組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。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。

AWS IoT TwinMakerのサービスロール

サービスロールに対するサポート	あり

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 のサービスにアクセス許可を委任するロールの作成」を参照してください。

> **Warning**
>
> サービスロールのアクセス許可を変更すると、 AWS IoT TwinMaker 機能が破損する可能性があります。が指示する場合以外 AWS IoT TwinMaker は、サービスロールを編集しないでください。

のサービスにリンクされたロール AWS IoT TwinMaker

サービスにリンクされたロールのサポート	いいえ

サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールの権限を表示できますが、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 と提携するAWS のサービス」を参照してください。表の中から、[Service-linked role] (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、[Yes] リンクを選択します。

のアイデンティティベースのポリシーの例 AWS IoT TwinMaker

デフォルトでは、 ユーザーおよびロールには、 AWS IoT TwinMaker リソースを作成または変更する権限はありません。また、、 AWS Command Line Interface （ AWS CLI ） AWS Management Console、または AWS API を使用してタスクを実行することはできません。IAM 管理者は、リソースで必要なアクションを実行するための権限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。

これらサンプルの JSON ポリシードキュメントを使用して、IAM アイデンティティベースのポリシーを作成する方法については、『IAM ユーザーガイド』の「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 の形式など AWS IoT TwinMaker、 で定義される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「 のアクション、リソース、および条件キー AWS IoT TwinMaker」を参照してください。 ARNs

トピック

- ポリシーのベストプラクティス
- AWS IoT TwinMaker コンソールを使用する
- 自分の権限の表示をユーザーに許可する

ポリシーのベストプラクティス

ID ベースのポリシーは、ユーザーのアカウントで誰かが AWS IoT TwinMaker リソースを作成、アクセス、または削除できる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 管理ポリシーを開始し、最小特権のアクセス許可に移行する – ユーザーとワークロードにアクセス許可を付与するには、多くの一般的なユースケースにアクセス許可を付与する AWS 管理ポリシーを使用します。これらは で使用できます AWS アカウント。ユースケースに固有の AWS カスタマー管理ポリシーを定義して、アクセス許可をさらに減らすことをお勧めします。詳細については、IAM ユーザーガイドの「AWS マネージドポリシー」または「AWS ジョブ機能の管理ポリシー」を参照してください。
- 最小特権を適用する – IAM ポリシーで権限を設定するときは、タスクの実行に必要な権限のみを付与します。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、最小特権権限とも呼ばれています。IAM を使用して権限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 でのポリシーと権限」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。条件を使用して、 などの特定の を介してサービスアクションが使用される場合に AWS のサービス、サービスアクションへのアクセスを許可することもできます AWS CloudFormation。詳細については、IAM ユーザーガイドの [IAM JSON policy elements: Condition] (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) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、「IAM ユーザーガイド」の「IAM Access Analyzer ポリシーの検証」を参照してください。
- 多要素認証 (MFA) を要求する – で IAM ユーザーまたはルートユーザーを必要とするシナリオがある場合は AWS アカウント、セキュリティを強化するために MFA を有効にします。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。

AWS IoT TwinMaker コンソールを使用する

AWS IoT TwinMaker コンソールにアクセスするには、最小限のアクセス許可のセットが必要です。これらのアクセス許可により、 の AWS IoT TwinMaker リソースの詳細を一覧表示および表示できます AWS アカウント。最小限必要な許可よりも制限が厳しいアイデンティティベースのポリシーを作

成すると、そのポリシーを持つエンティティ (ユーザーまたはロール) に対してコンソールが意図したとおりに機能しません。

AWS CLI または AWS API のみを呼び出すユーザーには、最小限のコンソールアクセス許可を付与する必要はありません。代わりに、実行しようとしている API オペレーションに一致するアクションのみへのアクセスが許可されます。

ユーザーとロールが AWS IoT TwinMaker 引き続きコンソールを使用できるようにするには、エンティティに AWS IoT TwinMaker `ConsoleAccess`または `ReadOnly` AWS 管理ポリシーもアタッチします。詳細については、「AWS IAM Identity Center ユーザーガイド」の「ユーザーへのアクセス許可の追加」を参照してください。

自分の権限の表示をユーザーに許可する

この例では、ユーザーアイデンティティにアタッチされたインラインおよびマネージドポリシーの表示を IAM ユーザーに許可するポリシーの作成方法を示します。このポリシーには、コンソールで、または AWS CLI または AWS API を使用してプログラムでこのアクションを実行するアクセス許可が含まれています。

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

```
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS IoT TwinMaker ID とアクセスのトラブルシューティング

次の情報は、 および 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 AWS IoT TwinMaker と修正に役立ちます。

トピック

- でアクションを実行する権限がない AWS IoT TwinMaker
- iam を実行する権限がありません。PassRole
- 自分の 以外のユーザーに自分の AWS IoT TwinMaker リソース AWS アカウント へのアクセスを許可したい

でアクションを実行する権限がない AWS IoT TwinMaker

「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。

次のエラー例は、mateojackson IAM ユーザーがコンソールを使用して、ある *my-example-widget* リソースに関する詳細情報を表示しようとしたことを想定して、その際に必要なiottwinmaker:*GetWidget* アクセス許可を持っていない場合に発生するものです。

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

この場合、iottwinmaker:*GetWidget* アクションを使用して *my-example-widget*リソースへのアクセスを許可するように、mateojackson ユーザーのポリシーを更新する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン資格情報を提供した担当者が管理者です。

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 AWS IoT TwinMakerにロールを渡すことができるようにする必要があります。

一部の AWS のサービス では、新しいサービスロールまたはサービスにリンクされたロールを作成する代わりに、そのサービスに既存のロールを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 AWS IoT TwinMaker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メアリーに `iam:PassRole` アクションの実行を許可する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン資格情報を提供した担当者が管理者です。

自分の 以外のユーザーに自分の AWS IoT TwinMaker リソース AWS アカウント へのアクセスを許可したい

他のアカウントのユーザーや組織外の人が、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセスコントロールリスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください:

- がこれらの機能 AWS IoT TwinMaker をサポートしているかどうかを確認するには、「」を参照してくださいが IAM と AWS IoT TwinMaker 連携する方法。
- 所有 AWS アカウント している のリソースへのアクセスを提供する方法については、IAM ユーザーガイドの「所有 AWS アカウント している別の の IAM ユーザーへのアクセスを提供する」を参照してください。

- リソースへのアクセスをサードパーティー に提供する方法については AWS アカウント、IAM ユーザーガイドの「サードパーティー AWS アカウント が所有する へのアクセスを提供する」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) へのアクセス権限」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。

のサービスにリンクされたロールの使用 AWS IoT TwinMaker

AWS IoT TwinMaker は AWS Identity and Access Management 、 (IAM) サービスにリンクされたロール を使用します。サービスにリンクされたロールは、 に直接リンクされた一意のタイプの IAM ロールです AWS IoT TwinMaker。サービスにリンクされたロールは によって事前定義 AWS IoT TwinMaker されており、ユーザーに代わってサービスから他の AWS のサービスを呼び出すために必要なすべてのアクセス許可が含まれています。

サービスにリンクされたロールを使用すると、必要なアクセス許可を手動で追加する必要がなくなるため、 の設定 AWS IoT TwinMaker が簡単になります。 は、サービスにリンクされたロールのアクセス許可 AWS IoT TwinMaker を定義し、特に定義されている場合を除き、 のみがそのロールを引き受け AWS IoT TwinMaker ることができます。定義したアクセス許可には、信頼ポリシーと許可ポリシーが含まれます。この許可ポリシーを他の IAM エンティティにアタッチすることはできません。

サービスリンクロールは、まずその関連リソースを削除しなければ削除できません。これにより、AWS IoT TwinMaker リソースにアクセスするためのアクセス許可を誤って削除することがないため、リソースが保護されます。

サービスリンクロールをサポートする他のサービスについては、「IAM と連動するAWS のサービス」を参照し、[Service-linked role (サービスリンクロール)] の列内で [Yes (はい)] と表記されたサービスを確認してください。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。

のサービスにリンクされたロールのアクセス許可 AWS IoT TwinMaker

AWS IoT TwinMaker は、 という名前のサービスにリンクされたロールを使用しますAWSServiceRoleForIoTTwinMaker。 AWS IoT TwinMaker がユーザーに代わって他の AWS サービスを呼び出し、それらのリソースを同期できるようにします。

AWSServiceRoleForIoTTwinMaker サービスにリンクされたロールは、次のサービスを信頼してロールを引き受けます。

- `iottwinmaker.amazonaws.com`

という名前のロールアクセス許可ポリシー AWSIoTTwinMakerServiceRolePolicy は AWS IoT TwinMaker 、 が指定されたリソースに対して次のアクションを実行できるようにします。

- アクション: `all your iotsitewise asset and asset-model resources` 上で `iotsitewise:DescribeAsset, iotsitewise:ListAssets, iotsitewise:DescribeAssetModel, and iotsitewise:ListAssetModels, iottwinmaker:GetEntity, iottwinmaker:CreateEntity, iottwinmaker:UpdateEntity, iottwinmaker:DeleteEntity, iottwinmaker:ListEntities, iottwinmaker:GetComponentType, iottwinmaker:CreateComponentType, iottwinmaker:UpdateComponentType, iottwinmaker:DeleteComponentType, iottwinmaker:ListComponentTypes`

ユーザー、グループ、ロールなどがサービスにリンクされたロールを作成、編集、削除できるようにするには、アクセス権限を設定する必要があります。詳細については、 IAM ユーザーガイド の「サービスリンクロールのアクセス許可」を参照してください。

のサービスにリンクされたロールの作成 AWS IoT TwinMaker

サービスリンクロールを手動で作成する必要はありません。 AWS Management Console、、 AWS CLIまたは AWS API で AWS IoT SiteWise アセットとアセットモデルを同期 (アセット同期) すると、 によってサービスにリンクされたロールが自動的に AWS IoT TwinMaker 作成されます。

このサービスリンクロールを削除した後で再度作成する必要が生じた場合は、同じ方法でアカウントにロールを再作成できます。 AWS IoT SiteWise アセットとアセットモデルを同期 (アセット同期) すると、 によってサービスにリンクされたロールが再度 AWS IoT TwinMaker 作成されます。

IAM コンソールを使用して、IoT TwinMaker - マネージドロール」ユースケースでサービスにリンクされたロールを作成することもできます。 AWS CLI または AWS API で、サービス名を使用して`iottwinmaker.amazonaws.com`サービスにリンクされたロールを作成します。詳細については、『IAM ユーザーガイド』の「サービスにリンクされたロールの作成」を参照してください。このサービスリンクロールを削除しても、同じ方法でロールを再作成できます。

のサービスにリンクされたロールの編集 AWS IoT TwinMaker

AWS IoT TwinMaker では、 AWSServiceRoleForIoTTwinMaker サービスにリンクされたロールを編集することはできません。サービスリンクロールを作成した後は、多くのエンティティによってロールが参照される可能性があるため、ロール名を変更することはできません。ただし、IAM を使用したロールの説明の編集はできます。詳細については、「IAM ユーザーガイド」の「サービスにリンクされたロールの編集」を参照してください。

のサービスにリンクされたロールの削除 AWS IoT TwinMaker

サービスリンクロールが必要な機能またはサービスが不要になった場合には、そのロールを削除することをお勧めします。そうすることで、モニタリングや保守が積極的に行われていない未使用のエンティティを排除できます。ただし、serviceLinkedロールを手動で削除する前に、まだサービスにリンクされたロールを使用している serviceLinked ワークスペースをクリーンアップする必要があります。

> **Note**
>
> リソースを削除しようとしたときに AWS IoT TwinMaker サービスがロールを使用している場合、削除が失敗する可能性があります。その場合は、数分待ってからオペレーションを再試行してください。

IAM を使用してサービスリンクロールを手動で削除するには

IAM コンソール、、または AWS API を使用して AWS CLI、 AWSServiceRoleForIoTTwinMaker サービスにリンクされたロールを削除します。詳細については、「IAM ユーザーガイド」の「サービスにリンクされたロールの削除」を参照してください。

AWS IoT TwinMaker サービスにリンクされたロールでサポートされているリージョン

AWS IoT TwinMaker は、サービスが利用可能なすべてのリージョンで、サービスにリンクされたロールの使用をサポートします。詳細については、「AWS リージョンとエンドポイント」を参照してください。

AWS の マネージドポリシー AWS IoT TwinMaker

ユーザー、グループ、ロールにアクセス許可を追加するには、自分でポリシーを記述するよりも、AWS 管理ポリシーを使用する方が簡単です。チームに必要な許可のみを提供する IAM カスタマーマネージドポリシーを作成するには、時間と専門知識が必要です。すぐに開始するには、 AWS マネージドポリシーを使用できます。これらのポリシーは、一般的なユースケースをターゲット範囲に含めており、 AWS アカウントで利用できます。 AWS 管理ポリシーの詳細については、「IAM ユーザーガイド」の「 AWS 管理ポリシー」を参照してください。

AWS サービスは、 AWS マネージドポリシーを維持および更新します。 AWS 管理ポリシーのアクセス許可は変更できません。サービスでは、新しい機能を利用できるようにするために、 AWS マネージドポリシーに権限が追加されることがあります。この種類の更新は、ポリシーがアタッチされている、すべてのアイデンティティ (ユーザー、グループおよびロール) に影響を与えます。新しい機能が立ち上げられた場合や、新しいオペレーションが使用可能になった場合に、各サービスが AWS マネージドポリシーを更新する可能性が最も高くなります。サービスは AWS マネージドポリシーからアクセス許可を削除しないため、ポリシーの更新によって既存のアクセス許可が中断されることはありません。

さらに、 は、複数の サービスにまたがる職務機能の マネージドポリシー AWS をサポートします。例えば、 ReadOnlyアクセス AWS 管理ポリシーは、すべての AWS サービスとリソースへの読み取り専用アクセスを提供します。サービスが新機能を起動すると、 は新しいオペレーションとリソースの読み取り専用アクセス許可 AWS を追加します。ジョブ機能のポリシーの一覧および詳細については、「IAM ユーザーガイド」の「AWS のジョブ機能のマネージドポリシー」を参照してください。

AWS マネージドポリシー： AWSIoTTwinMakerServiceRolePolicy

IAM エンティティ AWSIoTTwinMakerServiceRolePolicy に をアタッチすることはできません。このポリシーは、ユーザーに代わって がアクションを実行することを許可する、サービスにリンクされたロールにアタッチされます。詳細については、「のサービスにリンクされたロールのアクセス許可 AWS IoT TwinMaker」を参照してください。

という名前のロールアクセス許可ポリシー AWSIoTTwinMakerServiceRolePolicy は AWS IoT TwinMaker 、 が指定されたリソースに対して次のアクションを実行できるようにします。

- アクション: all your iotsitewise asset and asset-model resources 上で iotsitewise:DescribeAsset, iotsitewise:ListAssets, iotsitewise:DescribeAssetModel, and iotsitewise:ListAssetModels, iottwinmaker:GetEntity, iottwinmaker:CreateEntity, iottwinmaker:UpdateEntity, iottwinmaker:DeleteEntity, iottwinmaker:ListEntities, iottwinmaker:GetComponentType, iottwinmaker:CreateComponentType, iottwinmaker:UpdateComponentType, iottwinmaker:DeleteComponentType, iottwinmaker:ListComponentTypes

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Sid": "SiteWiseAssetReadAccess",
            "Effect": "Allow",
            "Action": [
                "iotsitewise:DescribeAsset"
            ],
            "Resource": [
                "arn:aws:iotsitewise:*:*:asset/*"
            ]
        },
        {
            "Sid": "SiteWiseAssetModelReadAccess",
            "Effect": "Allow",
            "Action": [
                "iotsitewise:DescribeAssetModel"
            ],
            "Resource": [
                "arn:aws:iotsitewise:*:*:asset-model/*"
            ]
        },
        {
            "Sid": "SiteWiseAssetModelAndAssetListAccess",
            "Effect": "Allow",
            "Action": [
                "iotsitewise:ListAssets",
                "iotsitewise:ListAssetModels"
            ],
```

```
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "TwinMakerAccess",
            "Effect": "Allow",
            "Action": [
                "iottwinmaker:GetEntity",
                "iottwinmaker:CreateEntity",
                "iottwinmaker:UpdateEntity",
                "iottwinmaker:DeleteEntity",
                "iottwinmaker:ListEntities",
                "iottwinmaker:GetComponentType",
                "iottwinmaker:CreateComponentType",
                "iottwinmaker:UpdateComponentType",
                "iottwinmaker:DeleteComponentType",
                "iottwinmaker:ListComponentTypes"
            ],
            "Resource": [
                "arn:aws:iottwinmaker:*:*:workspace/*"
            ],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iottwinmaker:linkedServices": [
                        "IOTSITEWISE"
                    ]
                }
            }
        }
    ]
}
```

AWS IoT TwinMakerAWS 管理ポリシーの更新

このサービスがこれらの変更の追跡を開始した以降の の AWS マネージドポリシーの更新に関する詳細を表示します。このページの変更に関する自動通知については、「 ドキュメン履歴ページ」ページで RSS フィードをサブスクライブしてください。

変更	説明	日付
AWSIoTTwinMakerServiceRolePolicy – ポリシーを追加しました	AWS IoT TwinMaker は、が指定されたリソースに対して次のアクションを実行できるようにする という名前 AWSIoTTwinMakerServiceRolePolicy AWS IoT TwinMaker のロール許可ポリシーを追加しました。 • アクション: `all your iotsitewise asset and asset-model resources 上で iotsitewise:DescribeAsset, iotsitewise:ListAssets, iotsitewise:DescribeAssetModel, and iotsitewise:ListAssetModels, iottwinmaker:GetEntity, iottwinmaker:CreateEntity, iottwinmaker:UpdateEntity, iottwinmaker:DeleteEntity, iottwinmaker:ListEntities, iottwinmaker:GetComponentType, iottwinmaker:CreateComponentType, iottwinmaker:UpdateComponentType, iottwinma`	2023 年 11 月 17 日

変更	説明	日付
	ker:DeleteComponentType, iottwinmaker:ListComponentTypes 詳細については、「のサービスにリンクされたロールのアクセス許可 AWS IoT TwinMaker」を参照してください。	
が変更の追跡を開始しました	が AWS マネージドポリシーの変更の追跡を開始しました。	2022 年 5 月 11 日

AWS IoT TwinMaker およびインターフェイス VPC エンドポイント (AWS PrivateLink)

仮想プライベートクラウド (VPC) と AWS IoT TwinMaker 間のプライベート接続は、インターフェイス VPC エンドポイントを作成することで確立できます。インターフェイスエンドポイントは を利用しておりAWS PrivateLink、インターネットゲートウェイ、ネットワークアドレス変換 (NAT) デバイス、VPN 接続、または AWS Direct Connect 接続なしで AWS IoT TwinMaker APIs にプライベートにアクセスできます。VPC 内のインスタンスは、パブリック IP アドレスがなくても AWS IoT TwinMaker APIs。VPC と 間のトラフィック AWS IoT TwinMaker は Amazon ネットワークを離れません。

各インターフェースエンドポイントは、サブネット内の 1 つ以上の Elastic Network Interface によって表されます。

詳細については、「Amazon VPC ユーザーガイド」の「インターフェイス VPC エンドポイント (AWS PrivateLink)」を参照してください。

AWS IoT TwinMaker VPC エンドポイントに関する考慮事項

のインターフェイス VPC エンドポイントを設定する前に AWS IoT TwinMaker、「Amazon VPC ユーザーガイド」の「インターフェイスエンドポイントのプロパティと制限」を参照してください。

AWS IoT TwinMaker は、VPC からのすべての API アクションの呼び出しをサポートします。

- データプレーン API の操作には、次のエンドポイントを使用します。

  ```
  data.iottwinmaker.region.amazonaws.com
  ```

  データプレーン API には、以下のオペレーションが含まれます。

  - GetProperty値
  - GetPropertyValueHistory
  - BatchPutPropertyValues
- コントロールプレーン API のオペレーションには、次のエンドポイントを使用します。

  ```
  api.iottwinmaker.region.amazonaws.com
  ```

  サポートされているコントロールプレーン API オペレーションには以下が含まれます。

  - CreateComponentタイプ
  - CreateEntity
  - CreateScene
  - CreateWorkspace
  - DeleteComponentタイプ
  - DeleteEntity
  - DeleteScene
  - DeleteWorkspace
  - GetComponentタイプ
  - GetEntity
  - GetScene
  - GetWorkspace
  - ListComponentタイプ

- ListComponentタイプ
- ListEntities
- ListScenes
- ListTagsForResource
- ListWorkspaces
- TagResource
- UntagResource
- UpdateComponentタイプ
- UpdateEntity
- UpdateScene
- UpdateWorkspace

AWS IoT TwinMakerのインターフェイス VPC エンドポイントの作成

Amazon VPC コンソールまたは AWS Command Line Interface () を使用して、 AWS IoT TwinMaker サービスの VPC エンドポイントを作成できますAWS CLI。詳細については、 Amazon VPC ユーザーガイド のインターフェイスエンドポイントの作成を参照してください。

次のサービス名 AWS IoT TwinMaker を使用する の VPC エンドポイントを作成します。

- データプレーン API の操作には、次のサービス名を使用します。

```
com.amazonaws.region.iottwinmaker.data
```

- コントロールプレーン API の操作には、次のサービス名を使用します。

```
com.amazonaws.region.iottwinmaker.api
```

エンドポイントのプライベート DNS を有効にする場合、 などのリージョンのデフォルトの DNS 名 AWS IoT TwinMaker を使用して、 に API リクエストを実行できますiottwinmaker.us-east-1.amazonaws.com。

詳細については、「Amazon VPC ユーザーガイド」の「インターフェイスエンドポイントを介したサービスへのアクセス」を参照してください。

AWS IoT TwinMaker PrivateLink は、以下のリージョンでサポートされています。

- us-east-1

  この ControlPlane サービスは、、use1-az1、use1-az2および のアベイラビリティーゾーンでサポートされていますuse1-az6。

  この DataPlane サービスは、、use1-az1、use1-az2および のアベイラビリティーゾーンでサポートされていますuse1-az4。

- us-west-2

  ControlPlane および DataPlane サービスは、、usw2-az1、usw2-az2および のアベイラビリティーゾーンでサポートされていますusw2-az3。

- eu-west-1
- eu-central-1
- ap-southeast-1
- ap-southeast-2

アベイラビリティーゾーンの詳細については、IDs - AWS リソースアクセスマネージャー AWS」を参照してください。

インターフェイス VPC エンドポイント AWS IoT TwinMaker を介したアクセス

インターフェイスエンドポイントを作成すると、 は、 との通信に使用できるエンドポイント固有の DNS ホスト名 AWS IoT TwinMaker を生成します AWS IoT TwinMaker。プライベート DNS のオプションはデフォルトで有効になっています。詳細については、「Amazon VPCユーザーガイド」の「プライベートホストゾーンの使用」を参照してください。

エンドポイントのプライベート DNS を有効にすると、次の VPC エンドポイントのいずれかを介して AWS IoT TwinMaker への API リクエストを行うことができます。

- データプレーン API の操作には、次のエンドポイントを使用します。####### はお客様の AWS リージョンに置き換えてください。」

```
data.iottwinmaker.region.amazonaws.com
```

- コントロールプレーン API の操作には、次のエンドポイントを使用する。####### はお客様の AWS リージョンに置き換えてください。」

```
api.iottwinmaker.region.amazonaws.com
```

エンドポイントのプライベート DNS を無効にした場合、エンドポイントを経由して AWS IoT TwinMaker にアクセスするには、次の操作が必要です。

- API リクエストで VPC エンドポイント URL を指定します。
  - データプレーン API の操作には、次のエンドポイント URL を使用します。*#vpc-endpoint-id#* と *#######* は、VPCエンドポイント ID とリージョンに置き換えてください。

    ```
    vpc-endpoint-id.data.iottwinmaker.region.vpce.amazonaws.com
    ```

  - コントロールプレーン API の操作には、次のエンドポイント URL を使用します。*#vpc-endpoint-id#* と *#######* は、VPCエンドポイント ID とリージョンに置き換えてください。

    ```
    vpc-endpoint-id.api.iottwinmaker.region.vpce.amazonaws.com
    ```

- ホストプレフィックスインジェクションを無効にする。 AWS CLI および AWS SDKs、各 API オペレーションを呼び出すときに、サービスエンドポイントの前にさまざまなホストプレフィックスを付加します。これにより、VPC エンドポイントを指定する AWS IoT TwinMaker と、 AWS CLI および AWS SDKs は に対して無効な URLs を生成します。

  > **Important**
  >
  > AWS CLI 、 AWS Tools for PowerShellでは、ホストプレフィックスインジェクションを無効化することはできません。つまり、プライベート DNS を無効にした場合、VPC エンドポイント AWS IoT TwinMaker を介して または AWS Tools for PowerShell にアクセスすることはできません AWS CLI 。これらのツールを使用してエンドポイント AWS IoT TwinMaker 経由で にアクセスする場合は、プライベート DNS を有効にします。

  AWS SDK でホストプレフィックスインジェクションを無効にする方法については、各 SDK の次のドキュメントセクションを参照してください。

  - AWS SDK for C++
  - AWS SDK for Go
  - AWS SDK for Go v2

- AWS SDK for Java
- AWS SDK for Java 2.x
- AWS SDK for JavaScript
- AWS SDK for .NET
- AWS SDK for PHP
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby

詳細については、「Amazon VPC ユーザーガイド」の「インターフェイスエンドポイントを介したサービスへのアクセス」を参照してください。

の VPC エンドポイントポリシーの作成 AWS IoT TwinMaker

VPC エンドポイントには、 AWS IoT TwinMakerへのアクセスを制御するエンドポイントポリシーをアタッチできます。このポリシーでは、以下の情報を指定します。

- アクションを実行できるプリンシパル。
- 実行可能なアクション。
- このアクションを実行できるリソース。

詳細については、「Amazon VPC ユーザーガイド」の「VPC エンドポイントでサービスへのアクセスを制御する」を参照してください。

例: AWS IoT TwinMaker アクションの VPC エンドポイントポリシー

のエンドポイントポリシーの例を次に示します AWS IoT TwinMaker。このポリシーは、エンドポイントにアタッチされると、123456789012すべてのリソースの AWS アカウントiottwinmakeradmin内の IAM ユーザーに、リストされた AWS IoT TwinMaker アクションへのアクセスを許可します。

```
{
   "Statement":[
      {
        "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/role"
                },
```

```
            "Resource": "*",
            "Effect":"Allow",
            "Action":[
               "iottwinmaker:CreateEntity",
               "iottwinmaker:GetScene",
               "iottwinmaker:ListEntities"
            ]
           }
    ]
}
```

のコンプライアンス検証 AWS IoT TwinMaker

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AWS のサービス による対象範囲内のコンプライアンスプログラムを参照し、関心のあるコンプライアンスプログラムを選択します。一般的な情報については、AWS 「コンプライアンスプログラム」を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、「 でのレポートのダウンロード AWS Artifact」の」を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS のサービス は、お客様のデータの機密性、貴社のコンプライアンス目的、適用される法律および規制によって決まります。 は、コンプライアンスに役立つ以下のリソース AWS 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 AWS を にデプロイする手順について説明します。
- アマゾン ウェブ サービスにおける HIPAA 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 – このホワイトペーパーでは、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について説明します。

  > **Note**
  >
  > すべて AWS のサービス HIPAA の対象とな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 コンプライアンスリソース – このワークブックとガイドのコレクションは、お客様の業界や地域に適用される場合があります。

- AWS カスタマーコンプライアンスガイド – コンプライアンスの観点から責任共有モデルを理解します。このガイドでは、ガイダンスを保護し AWS のサービス 、複数のフレームワーク (米国国立標準技術研究所 (NIST)、Payment Card Industry Security Standards Council (PCI)、国際標準化機構 (ISO) を含む) のセキュリティコントロールにマッピングするためのベストプラクティスをまとめています。
- 「 デベロッパーガイド」の「 ルールによるリソースの評価」 – この AWS Config サービスは、リソース設定が社内プラクティス、業界ガイドライン、および規制にどの程度準拠しているかを評価します。 AWS Config
- AWS Security Hub – これにより AWS のサービス 、 内のセキュリティ状態を包括的に把握できます AWS。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。
- Amazon GuardDuty – これにより AWS アカウント、疑わしいアクティビティや悪意のあるアクティビティがないか環境を監視することで、、、ワークロード、コンテナ、データに対する潜在的な脅威 AWS のサービス を検出します。 GuardDuty は、特定のコンプライアンスフレームワークで義務付けられている侵入検知要件を満たすことで、PCI DSS などのさまざまなコンプライアンス要件への対応に役立ちます。
- AWS Audit Manager – これにより AWS のサービス 、 AWS 使用状況を継続的に監査し、リスクの管理方法と規制や業界標準への準拠を簡素化できます。

の耐障害性 AWS IoT TwinMaker

AWS グローバルインフラストラクチャは、 AWS リージョン およびアベイラビリティーゾーンを中心に構築されています。物理的に分離および分離された複数のアベイラビリティーゾーン AWS リージョン を提供し、低レイテンシー、高スループット、高冗長ネットワークで接続されます。アベイラビリティーゾーンでは、ゾーン間で中断することなく自動的にフェイルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性が高く、フォールトトレラントで、スケーラブルです。

AWS リージョン およびアベイラビリティーゾーンの詳細については、AWS 「 グローバルインフラストラクチャ」を参照してください。

グローバル AWS インフラストラクチャに加えて、 AWS IoT TwinMaker では、データの耐障害性とバックアップのニーズに対応できるように、いくつかの機能を提供しています。

のインフラストラクチャセキュリティ AWS IoT TwinMaker

マネージドサービスである AWS IoT TwinMaker は、ホワイトペーパー「Amazon Web Services: セキュリティプロセスの概要」に記載されている AWS グローバルネットワークセキュリティの手順で保護されています。

が AWS 公開している API コールを使用して、ネットワーク AWS IoT TwinMaker 経由で にアクセスします。クライアントは、Transport Layer Security (TLS) 1.2 以降をサポートする必要があります。TLS 1.3 以降が推奨されます。また、一時的ディフィー・ヘルマン Ephemeral Diffie-Hellman (DHE) や Elliptic Curve Ephemeral Diffie-Hellman (ECDHE) などの Perfect Forward Secrecy (PFS) を使用した暗号スイートもクライアントでサポートされている必要があります。これらのモードは、Java 7 以降など、最近のほとんどのシステムでサポートされています。

また、リクエストには、アクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、AWS Security Token ServiceAWS STS を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

エンドポイントとクォータ

AWS IoT TwinMaker エンドポイントとクォータ

AWS IoT TwinMaker エンドポイントとクォータについては、「 AWS 全般のリファレンス」を参照してください。

- サービスエンドポイントの詳細については、「AWS IoT TwinMaker サービスエンドポイント」を参照してください。
- クォータの詳細については、AWS IoT TwinMaker Service Quotasを参照してください。
- APIスロットリング制限について詳しくは、「AWS IoT TwinMaker APIスロットリング制限」を参照してください。

AWS IoT TwinMaker エンドポイントに関する追加情報

プログラムで に接続するには AWS IoT TwinMaker、 エンドポイントを使用します。HTTPクライアントを使用する場合は、次のようにコントロールプレーンAPIとデータプレーンAPIにプレフィックスを付ける必要があります。ただし、 AWS SDK および AWS Command Line Interface コマンドには必要なプレフィックスが自動的に追加されるため、プレフィックスを追加する必要はありません。

- コントロールプレーンAPIにはapiプレフィックスを使用します。例えば、`api.iottwinmaker.us-west-1.amazonaws.com`です。
- データプレーンAPIにはdataプレフィックスを使用します。例えば、`data.iottwinmaker.us-west-1.amazonaws.com`です。

AWS IoT TwinMaker ユーザーガイドのドキュメント履歴

AWS IoT TwinMaker ドキュメントのリリースの説明は、次の表のとおりです。

変更	説明	日付
新しいサービスにリンクされたロールと新しい IAM ポリシー	AWS IoT TwinMaker は、 と呼ばれる新しいサービスにリンクされたロールを追加しましたAWSServiceRoleForIoTTwinMaker。 は、 AWS IoT TwinMakerがユーザーに代わって他の のサービスを呼び出し、そのリソースを同期できるように、この新しいAWSサービスにリンクされたロールAWS IoT TwinMakerを追加しました。新しい AWSIoTTwinMakerServiceRolePolicy IAM ポリシーがこのロールにアタッチされ、このポリシーは、ユーザーに代わって他の AWSサービスをAWS IoT TwinMaker呼び出し、そのリソースを同期するアクセス許可を付与します。	2023 年 11 月 17 日
初回リリース	AWS IoT TwinMaker ユーザーガイドの初回リリース	2021 年 11 月 30 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。