開発者ガイド

AWS IoT FleetWise

AWS IoT FleetWise: 開発者ガイド

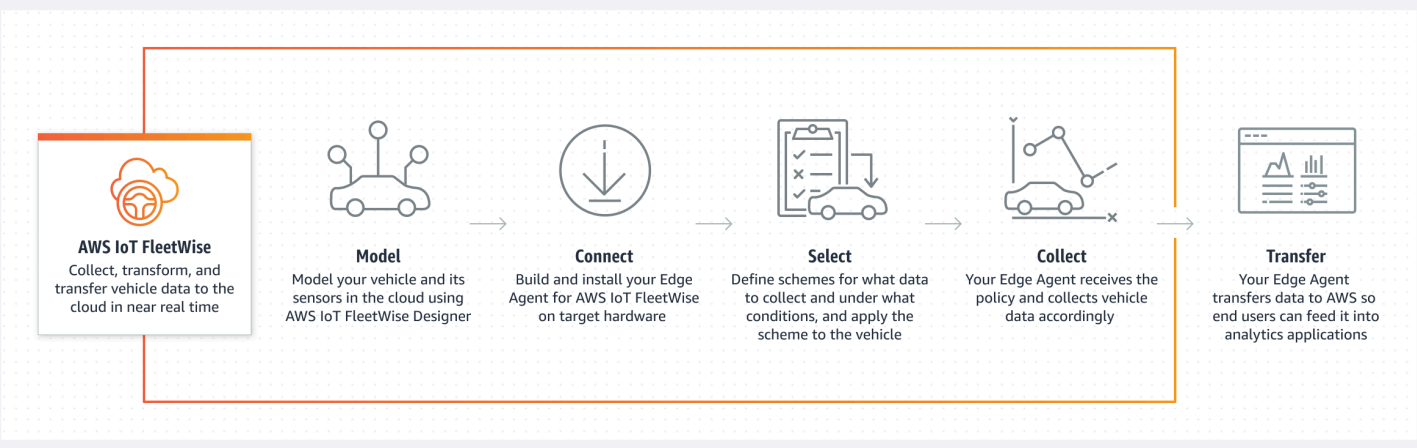
Table of Contents

AWS IoT FleetWise とは

AWS IoT FleetWise は、クラウドで車両データを収集して整理するために使用できるマネージドサービスです。収集したデータを使用して、車両の品質、性能、自律性を改善できます。AWS IoT FleetWise を使用すると、さまざまなプロトコルやデータ形式を使用する車両からデータを収集して整理することができます。AWSIoT FleetWise は、低レベルのメッセージを人間が読める値に変換し、クラウド上のデータ形式をデータ分析用に標準化するために役立ちます。また、データ収集キャンペーンを定義して、収集する車両データと、そのデータをクラウドに転送するタイミングを制御することもできます。

クラウドに配置された車両データは、車両のフリートの状態を分析するアプリケーションで使用できます。このデータは、潜在的なメンテナンス問題の特定、車載インフォテインメントシステムのスマート化、分析と機械学習 (ML) による自動運転や運転支援システムなどの高度なテクノロジーの改良に役立ちます。

AWS IoT FleetWise の基本的なアーキテクチャを次の図に示します。



トピック

利点

AWS IoT FleetWise の主な利点は次のとおりです。

車両データをよりインテリジェントに収集

必要なデータだけをクラウドに送信して分析するインテリジェントなデータ収集により、データの関連性を向上させます。

標準化されたフリート全体のデータを簡単に分析

カスタムのデータ収集システムやログ記録システムを開発しなくても、車両のフリートからの標準化されたデータを分析します。

クラウドでの自動データ同期

標準センサー (テレメトリーデータ) とビジョンシステム (カメラ、レーダー、LIDAR からのデータ) の両方から収集したデータを統合して表示し、クラウド内で自動的に同期します。AWSIoT FleetWise は、構造化および非構造化の両方のビジョンシステムデータ、メタデータ、および標準センサーデータをクラウド内で自動的に同期します。これにより、イベントの全体像を把握してインサイトを得るプロセスが効率化されます。

> **Note**
>
> ビジョンシステムデータはプレビューリリースであり、変更される可能性があります。

ユースケース

AWS IoT FleetWise を使用できるシナリオには、次のようなものがあります。

AI/ML モデルのトレーニング

実稼働車両からデータを収集することで、自動運転支援システムや先進運転支援システムに使用される機械学習モデルを継続的に改善します。

デジタルカスタマーエクスペリエンスの強化

車載インフォテインメントシステムのデータを使用して、オーディオビジュアルコンテンツとアプリ内インサイトの関連性を高めます。

車両フリートのヘルスの維持

フリートデータからのインサイトを使用して、EV バッテリーのヘルスや充電レベルのモニタリング、メンテナンススケジュールの管理、燃料消費量の分析などを行います。

AWS IoT FleetWise を初めて使用する場合

AWS IoT FleetWise を初めて使用する場合は、最初に以下のセクションを読むことをお勧めします。

- AWS IoT FleetWise の仕組み
- AWS IoT FleetWise のセットアップ
- Edge Agent ソフトウェアのデモ
- クラウドへのデータの取り込み

AWS IoT FleetWise へのアクセス

AWS IoT FleetWise にアクセスするには、AWS IoT FleetWise コンソールまたは API を使用します。

AWS IoT FleetWise の料金

各車両は MQTT メッセージを通じてクラウドにデータを送信します。AWS IoT FleetWise で作成した車両の料金は、毎月末にお支払いいただきます。また、車両から収集するメッセージにも料金がかかります。料金に関する最新の情報については、「AWS IoT FleetWise の料金」を参照してください。MQTT メッセージングプロトコルの詳細については、「AWS IoT Core デベロッパーガイド」の「MQTT」を参照してください。

AWS IoT FleetWise の仕組み

以下のセクションでは、AWS IoT FleetWise サービスコンポーネントの概要と、それらがどのように相互に機能するかを示します。

この概要を読んだ後は、「AWS IoT FleetWise のセットアップ」のセクションを参照して、AWS IoT FleetWise のセットアップ方法を学んでください。

トピック

主要なコンセプト

AWS IoT FleetWise は、車両とそのセンサーおよびアクチュエータをクラウドでモデル化するための、車両モデリングフレームワークを提供します。車両とクラウド間の安全な通信を可能にするために、AWS IoT FleetWise には、車両にインストールできるエッジエージェントソフトウェアの開発に役立つリファレンス実装も用意されています。データ収集スキームをクラウドで定義し、車両にデプロイできます。車両で動作するエッジエージェントソフトウェアは、データ収集スキームを使用して、収集するデータとクラウドに転送するタイミングを制御します。

以下は、AWS IoT FleetWise の主な概念です。

シグナル

シグナルは、車両データとそのメタデータを格納するために定義する基本構造です。シグナルには、属性、ブランチ、センサー、アクチュエータがあります。例えば、車内の温度値を受け取るセンサーを作成し、そのメタデータ (センサー名、データ型、単位など) を格納できます。詳細については、「シグナルカタログの作成と管理」を参照してください。

属性

属性は、製造元や製造日など、通常は変更されない静的な情報を表します。

ブランチ

ブランチとは、ネストされた構造内のシグナルを表します。ブランチは、シグナルの階層を明確に示します。例えば、`Vehicle` というブランチに `Powertrain` という子ブランチがあるとします。`Powertrain` ブランチには `combustionEngine` という子ブランチがあります。`combustionEngine` ブランチを特定するには、`Vehicle.Powertrain.combustionEngine` という式を使用します。

センサー

センサーデータは、液面、温度、振動、電圧などの車両の状態について、現在の状態と経時的な変化を報告します。

アクチュエータ

アクチュエータデータは、モーター、ヒーター、ドアロックなど、車両デバイスの状態を報告します。車両デバイスの状態を変更すると、アクチュエータデータが更新される可能性がありま

す。例えば、ヒーターを表すアクチュエータを定義できます。このアクチュエータは、ヒーターをオンまたはオフにしたときに新しいデータを受け取ります。

カスタム構造

カスタム構造 (構造体とも呼ばれる) は、複雑なデータ構造または高次のデータ構造を表します。これにより、同じソースから生成されたデータの論理的なバインドやグループ化が容易になります。構造体は、複雑なデータ型や高次の形状を表すなど、アトミック操作でデータを読み書きする場合に使用します。

構造体型のシグナルは、プリミティブデータ型の代わりに構造体データ型への参照を使用してシグナルカタログで定義します。構造体は、センサー、属性、アクチュエータ、ビジョンシステムデータ型など、あらゆるタイプのシグナルに使用できます。構造体型のシグナルを送受信する場合、AWS IoT FleetWise は、含まれるすべての項目に有効な値があることを期待するため、すべての項目は必須です。例えば、構造体内に項目として Vehicle.Camera.Image.height、Vehicle.Camera.Image.width、Vehicle.Camera.Image.data が含まれている場合、送信されたシグナルには、これらすべての項目の値が含まれていることが期待されます。

> **Note**
>
> ビジョンシステムデータはプレビューリリースであり、変更される可能性があります。

カスタムプロパティ

カスタムプロパティは複雑なデータ構造のメンバーを表します。プロパティのデータ型は、プリミティブまたは別の構造体のいずれかになります。

構造体とカスタムプロパティを使用して高次の形状を表現する場合、意図した高次の形状は常にツリー構造として定義され、視覚化されます。カスタムプロパティはすべてのリーフノードを定義するために使用し、構造体はリーフ以外のすべてのノードを定義するために使用します。

シグナルカタログ

シグナルカタログには、シグナルのコレクションが格納されます。シグナルカタログ内のシグナルを使用して、さまざまなプロトコルやデータ形式を使用する車両をモデル化できます。例えば、異なる自動車メーカーの 2 台の車があるとします。1 台はコントローラーエリアネットワーク (CAN バス) プロトコルを使用し、もう 1 台はオンボードダイアグノーシス (OBD) プロトコルを使用しています。シグナルカタログには、車内の温度値を受信するセンサーを定義すること

ができます。このセンサーを、両方の車の熱電対を表すために使用できます。詳細については、「シグナルカタログの作成と管理」を参照してください。

車両モデル (モデルマニフェスト)

車両モデルとは、車両の形式を標準化し、車両内のシグナル間の関係を定義するために使用できる宣言的な構造です。車両モデルにより、同じタイプの複数の車両に一貫した情報が適用されます。車両モデルを作成するには、シグナルを追加します。詳細については、「車両モデルの作成と管理」を参照してください。

デコーダーマニフェスト

デコーダーマニフェストには、車両モデル内の各シグナルのデコード情報が含まれています。車両内のセンサーやアクチュエータが送信するのは、低レベルのメッセージ (バイナリデータ) です。デコーダーマニフェストを使用することで、AWS IoT FleetWise でバイナリデータを人間が読める値に変換できるようになります。すべてのデコーダーマニフェストは車両モデルに関連付けられます。詳細については、「デコーダーマニフェストの作成と管理」を参照してください。

ネットワークインターフェイス

車載ネットワークで使用されるプロトコルに関する情報が含まれています。AWSIoT FleetWise では、次のプロトコルがサポートされています。

コントローラーエリアネットワーク (CAN バス)

電子制御ユニット (ECU) 間でのデータの通信方法を定義するプロトコル。ECU には、エンジンコントロールユニット、エアバッグ、オーディオシステムなどがあります。

オンボードダイアグノーシス (OBD) II

ECU 間の自己診断データの通信方法を定義する、より進化したプロトコル。車両の問題を特定するために役立つ標準の故障診断コード (DTC) が多数定義されています。

車両ミドルウェア

車両ミドルウェアは、ネットワークインターフェイスの一種として定義されます。車両ミドルウェアの例としては、ロボットオペレーティングシステム (ROS 2) や Scalable service-Oriented MiddlewarE over IP (SOME/IP) が挙げられます。

> **Note**
>
> AWS IoT FleetWise は、ビジョンシステムデータ用の ROS 2 ミドルウェアをサポートしています。

デコーダーシグナル

特定のシグナルについて詳細なデコード情報を提供します。車両モデルに指定されたすべてのシグナルは、デコーダーシグナルとペアになっている必要があります。デコーダーマニフェストに CAN ネットワークインターフェイスが含まれている場合は、CAN デコーダーシグナルも含まれている必要があります。デコーダーマニフェストに OBD ネットワークインターフェイスが含まれている場合は、OBD デコーダーシグナルも含まれている必要があります。

デコーダーマニフェストに車両ミドルウェアインターフェイスも含まれている場合は、メッセージデコーダーシグナルも含める必要があります。

車両

車やトラックなどの物理的な車両を仮想的に表現したものです。車両とは、車両モデルのインスタンスです。同じ車両モデルから作成された車両は、同じシグナルのグループを継承します。各車両は AWS IoT モノに相当します。

フリート

フリートは、車両のグループを表します。車両のフリートを簡単に管理できるようにするには、事前に個々の車両をフリートに関連付ける必要があります。

Campaign

データ収集スキームが含まれています。キャンペーンはクラウドで定義し、車両またはフリートにデプロイします。キャンペーンにより、データをどのように選択して収集し、クラウドに転送するかに関する指示がエッジエージェントソフトウェアに与えられます。

データ収集スキーム

データ収集スキームは、エッジエージェントソフトウェアにデータの収集方法を指示します。現在、AWS IoT FleetWise では、条件ベースの収集スキームと時間ベースの収集スキームがサポートされています。

条件ベースの収集スキーム

論理式を使用して、収集するデータを認識します。エッジエージェントソフトウェアは、条件が満たされたときにデータを収集します。例えば、`$variable.myVehicle.InVehicleTemperature >35.0` という式を使用すると、エッジエージェントソフトウェアは 35.0 より大きい温度値を収集します。

時間ベースの収集スキーム

データ収集の頻度を定義する時間間隔をミリ秒単位で指定します。例えば、時間間隔が 10,000 ミリ秒の場合、エッジエージェントソフトウェアはデータを 10 秒ごとに 1 回収集します。

AWS IoT FleetWise の機能

AWS IoT FleetWise の主な機能は次のとおりです。

車両のモデリング

車両の仮想表現を構築し、車両シグナルを体系化する共通形式を適用します。AWSIoT FleetWise は、車両シグナルの標準化に使用できる Vehicle Signal Specification (VSS) をサポートしています。

スキームベースのデータ収集

価値の高い車両データのみをクラウドに転送するスキームを定義します。条件ベースのスキームを定義すると、収集するデータを制御できます。例えば、車内の温度値データを、値が 40 度を超える場合に収集できます。時間ベースのスキームを定義して、データの収集頻度を制御することもできます。

AWS IoT FleetWise 用エッジエージェントソフトウェア

車両内で実行されるエッジエージェントソフトウェアは、車両とクラウド間の通信を支援するものです。車両がクラウドに接続されている間、エッジエージェントソフトウェアは継続的にデータ収集スキームを受信し、それに従ってデータを収集します。

関連サービス

AWS IoT FleetWise は、クラウドソリューションの可用性とスケーラビリティを高めるために、以下の AWS サービスと統合されます。

- AWS IoT Core - 車両データを AWS IoT FleetWise にアップロードする AWS IoT デバイスを登録して制御します。詳細については、「AWS IoT デベロッパーガイド」の「AWS IoT とは」を参照してください。
- Amazon Timestream - 時系列データベースを使用して、車両データを保存および分析します。詳細については、「Amazon Timestream Developer Guide」の「What is Amazon Timestream」を参照してください。
- Amazon S3 - オブジェクトストレージサービスを使用して、車両データを保存および管理します。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「Amazon S3 とは」を参照してください。

AWS IoT FleetWise のセットアップ

AWS IoT FleetWise を初めて使用する場合は、事前に以下のセクションのステップを完了してください。

トピック

AWS アカウントのセットアップ

以下のタスクを完了して、AWS にサインアップして管理者ユーザーを作成します。

AWS アカウントへのサインアップ

AWS アカウントがない場合は、以下のステップを実行して作成します。

AWS アカウントにサインアップするには

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

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

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

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

管理ユーザーの作成

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

AWS アカウントのルートユーザーをセキュリティで保護する

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

   ルートユーザーを使用してサインインする方法については、「AWS サインイン User Guide」の「Signing in as the root user」を参照してください。
2. ルートユーザーの多要素認証 (MFA) を有効にします。

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

管理ユーザーを作成する

1. IAM Identity Center を有効にする

   手順については、「AWS IAM Identity Centerユーザーガイド」の「AWS IAM Identity Centerの有効化」を参照してください。
2. IAM アイデンティティセンターで、管理ユーザーに管理アクセス権を付与します。

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

管理ユーザーとしてサインインする

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

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

> **Note**
>
> AWS IoT FleetWise では、サービスにリンクされたロールを使用できます。サービスにリンクされたロールは AWS IoT FleetWise によって事前定義され、AWS IoT FleetWise から Amazon CloudWatch にメトリクスを送信するために必要なアクセス許可が含まれています。詳細については、「AWS IoT のサービスにリンクされたロールの使用 FleetWise」を参照してください。

コンソールの開始方法

AWS アカウントにまだサインインしていない場合はサインインし、AWS IoT FleetWise コンソールを開きます。AWS IoT FleetWise の使用を開始するには、車両モデルを作成します。車両モデルは、車両の形式を標準化するものです。

1. AWS IoT FleetWise コンソールに移動します。
2. [AWS IoT FleetWise の使用を開始する] で、[使用を開始] を選択します。

車両モデルの作成の詳細については、「車両モデルの作成 (コンソール)」を参照してください。

設定の構成

AWS IoT FleetWise コンソールまたは API を使用すると、Amazon CloudWatch Logs メトリクスや Amazon CloudWatch Logs の設定を構成し、AWS マネージドキーでデータを暗号化できます。

CloudWatch メトリクスにより、AWS IoT FleetWise やその他の AWS リソースをモニタリングできます。CloudWatch メトリクスは、サービス制限を超過していないかどうかを確認するなどの目的で、メトリクスを収集して追跡するために使用できます。CloudWatch のメトリクスの詳細については、「Amazon CloudWatch による AWS IoT FleetWise のモニターリング」を参照してください。

CloudWatch Logs を使用すると、AWS IoT FleetWise は CloudWatch ロググループにログデータを送信します。そのログデータを問題の特定と軽減に役立てることができます。CloudWatch Logs の詳細については、「AWS IoT FleetWise のログ記録の構成」を参照してください。

データ暗号化では、AWS IoT FleetWise は AWS マネージドキーを使用してデータを暗号化します。AWS KMS でキーの作成と管理を行うこともできます。暗号化の詳細については、「データ暗号化」を参照してください。

設定の構成 (コンソール)

AWS アカウントにまだサインインしていない場合はサインインし、AWS IoT FleetWise コンソールを開きます。

1. AWS IoT FleetWise コンソールに移動します。
2. 左側のペインで、[設定] を選択します。
3. [メトリクス] で、[有効化] を選択します。AWSIoT FleetWise は、サービスにリンクされたロールに自動的に CloudWatch マネージドポリシーをアタッチし、CloudWatch メトリクスを有効にします。
4. [ログ記録] で、[編集] を選択します。
   a. [CloudWatch ログ記録] セクションで、[ロググループ] を入力します。
   b. 変更を保存するには、[送信] を選択します。
5. [暗号化] セクションで、[編集] を選択します。
   a. 使用するキーの種類を選択します。詳細については、「キー管理」を参照してください。
      i. [AWS キーを使用] - AWS IoT FleetWise がキーを所有および管理します。
      ii. [別の AWS Key Management Service キーを選択する] - アカウント内の AWS KMS keys を自分で管理します。
   b. 変更を保存するには、[送信] を選択します。

設定の構成 (AWS CLI)

AWS CLI で、アカウントを登録して設定を構成します。

1. 設定を構成するには、次のコマンドを実行します。

```
aws iotfleetwise register-account
```

2. 設定を確認するには、次のコマンドを実行して登録ステータスを取得します。

> **Note**
>
> サービスにリンクされたロールは、AWS IoT FleetWise メトリクスを CloudWatch に発行するためにのみ使用されます。詳細については、「AWS IoT のサービスにリンクされたロールの使用 FleetWise」を参照してください。

```
aws iotfleetwise get-register-account-status
```

Example レスポンス

```
{
    "accountStatus": "REGISTRATION_SUCCESS",
    "creationTime": "2022-07-28T11:31:22.603000-07:00",
    "customerAccountId": "012345678912",
    "iamRegistrationResponse": {
        "errorMessage": "",
        "registrationStatus": "REGISTRATION_SUCCESS",
        "roleArn": "arn:aws:iam::012345678912:role/AWSIoTFleetwiseServiceRole"
    },
    "lastModificationTime": "2022-07-28T11:31:22.854000-07:00",
    }
}
```

登録ステータスは次のいずれかになります。

- REGISTRATION_SUCCESS - AWS リソースは正常に登録されました。
- REGISTRATION_PENDING - AWS IoT FleetWise は登録リクエストを処理しています。このプロセスの完了には約 5 分かかります。
- REGISTRATION_FAILURE - AWS IoT FleetWise は AWS リソースを登録できません。あとでもう一度試してみてください。

AWS IoT を始める FleetWise

AWS IoT を使用すると FleetWise、車両データを収集、変換、転送できます。このセクションのチュートリアルを使用して、 AWS IoT FleetWise を始めましょう。

AWS IoT の詳細については、以下のトピックを参照してください FleetWise。

- クラウドへのデータの取り込み
- 車両のモデリング
- 車両の作成、プロビジョニング、管理
- フリートの作成と管理
- キャンペーンによるデータの収集と転送

要件

AWS IoT AWS アカウント を使い始めるには、が必要です FleetWise。アカウントをお持ちでない場合は、「AWS IoT FleetWise のセットアップ」を参照してください。

AWS IoT FleetWise が利用可能なリージョンを使用してください。詳しくは、「AWS IoT FleetWise エンドポイントとクォータ」を参照してください。のリージョンセレクターを使用して、これらのリージョンのいずれかに切り替えることができます。 AWS Management Console

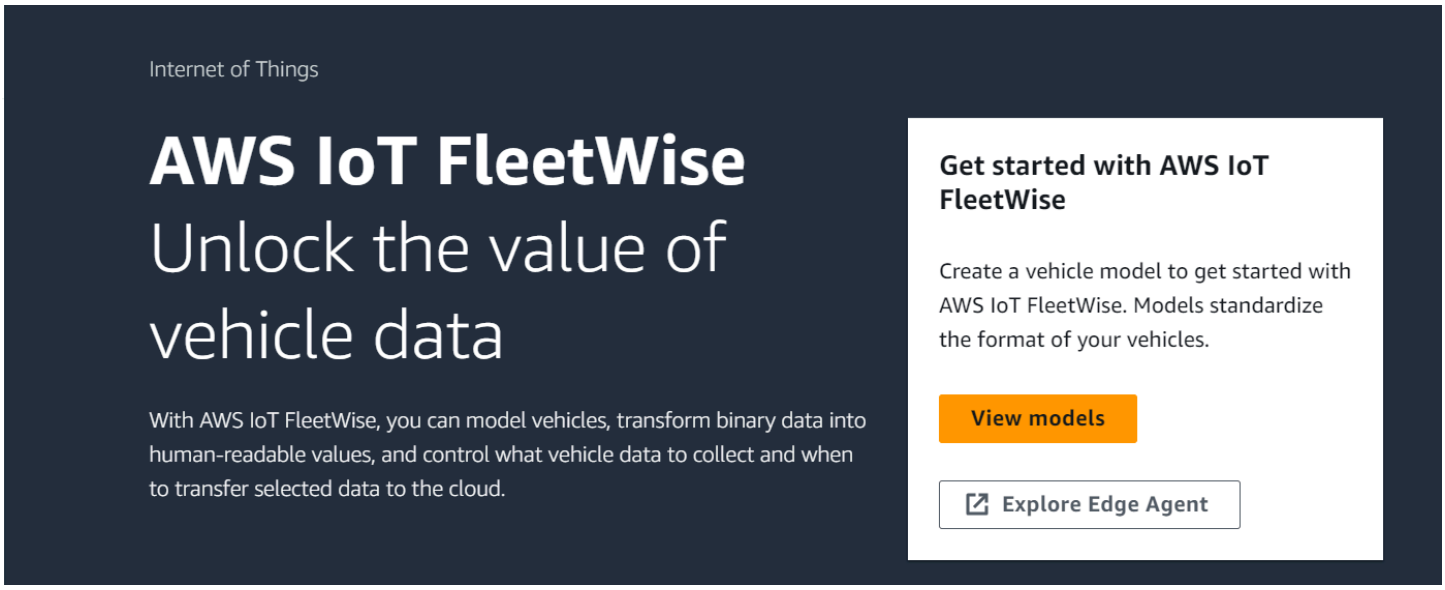
Edge Agent ソフトウェアのデモ

Explore Edge Agent クイックスタートデモを使用して AWS IoT FleetWise を探索し、IoT AWS 用の Edge Agent ソフトウェアを開発する方法を学ぶことができます FleetWise。 AWS CloudFormation このデモではテンプレートを使用します。エッジエージェントのリファレンス実装を詳しく紹介し、エッジエージェントの開発、Amazon EC2 Graviton へのエッジエージェントソフトウェアのデプロイ、サンプル車両データの生成の手順を示します。また、このデモには、シグナルカタログ、車両モデル、デコーダーマニフェスト、車両、フリート、キャンペーンをクラウド内で作成するために使用できるスクリプトも用意されています。クイックスタートデモの詳細については、次の手順を実行して、「エッジエージェントソフトウェアデベロッパーガイド」をダウンロードしてください。

クイックスタートデモをダウンロードするには

1. AWS IoT FleetWise コンソールに移動します。

2. サービスのホームページの「 AWS IoT 入門 FleetWise」セクションで、「Explore Edge Agent」を選択します。



チュートリアル: AWS IoT 入門 FleetWise (コンソール)

AWS IoT FleetWise を使用して、自動運転車から独自のデータ形式をほぼリアルタイムで収集、変換、クラウドに転送します。フリート全体のインサイトにアクセスできます。これにより、車両のヘルスに関する問題の効率的な検出と軽減、価値の高いデータシグナルの転送、問題のリモート診断を、コストを抑えながら行うことができます。

このチュートリアルでは、 AWS IoT を使い始める方法を説明します FleetWise。車両モデル (モデルマニフェスト)、デコーダーマニフェスト、車両、キャンペーンの作成方法を学ぶことができます。

AWS IoT の主要なコンポーネントと概念の詳細については FleetWise、を参照してくださいAWS IoT FleetWise の仕組み。

推定所要時間: 約 45 分。

> **Important**
>
> FleetWise このデモで作成および消費する AWS IoT リソースに対して課金されます。詳細については、AWS IoT FleetWise FleetWise 価格ページの「AWS IoT」を参照してください。

トピック

前提条件

この入門チュートリアルを完了するには、まず以下の準備が必要です。

- と AWS アカウント。がない場合は AWS アカウント、『AWS Account Management リファレンスガイド』 AWS アカウントの「の作成」を参照してください。
- AWS IoT AWS リージョン をサポートするファンへのアクセス FleetWise。現在、 AWS IoT FleetWise は米国東部 (バージニア北部) とヨーロッパ (フランクフルト) でサポートされています。
- Amazon Timestream リソース:
  - Amazon Timestream データベース。詳細については、「Amazon Timestream Developer Guide」の「Create a database」を参照してください。
  - Amazon Timestream で作成された、データを保持するための Amazon Timestream テーブル。詳細については、「Amazon Timestream Developer Guide」の「Create a table」を参照してください。

ステップ 1: AWS IoT 用 Edge エージェントソフトウェアのセットアップ FleetWise

> **Note**
>
> CloudFormation このステップのスタックはテレメトリデータを使用します。CloudFormation ビジョンシステムデータを使用してスタックを作成することもできます。詳細については、「ビジョンシステムデータデベロッパーガイド」を参照してください。

ビジョンシステムデータはプレビューリリースであり、変更される可能性があります。

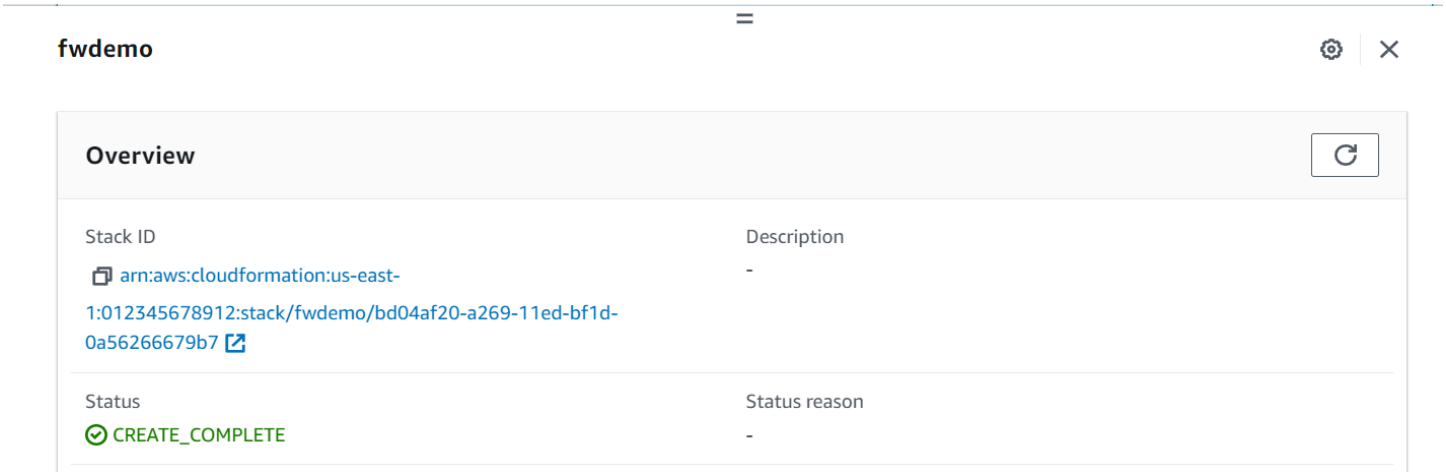
AWS IoT 用のエッジエージェントソフトウェアは、 FleetWise 車両とクラウド間の通信を促進します。クラウドに接続された車両からデータを収集する方法についての指示は、データ収集スキームから受け取ります。

エッジエージェントソフトウェアをセットアップするには、[一般的な情報] で次の操作を行います。

1. CloudFormation 起動テンプレートを開きます。
2. [スタックのクイック作成] ページで、[スタック名] に AWS IoT FleetWise リソースのスタックの名前を入力します。スタックは、 AWS CloudFormation このテンプレートが作成するリソースの名前のプレフィックスとして表示されるわかりやすい名前です。
3. [パラメータ] で、スタックに関連するパラメータのカスタム値を入力します。

   a. [Fleetsize] - Fleetsize パラメータを更新すると、フリート内の車両の数を増やすことができます。

   b. IoT CoreRegion-IoT CoreRegion パラメータを更新することで、 AWS IoT モノが作られるリージョンを指定できます。 AWS IoT FleetWise 車両の作成に使用したのと同じリージョンを使用する必要があります。詳細については AWS リージョン、「リージョンとゾーン- Amazon Elastic Compute Cloud」を参照してください。
4. 機能セクションで、IAM AWS CloudFormation リソースの作成を確認するボックスを選択します。
5. [スタックの作成] を選択し、スタックのステータスに CREATE_COMPLETE が表示されるまで約 15 分待ちます。
6. スタックが作成されたことを確認するには、[スタックの情報] タブを選択し、ビューを更新して、CREATE_COMPLETE を探します。



> **Important**
>
> FleetWise このデモで作成および消費する AWS IoT リソースに対して課金されます。詳細については、AWS IoT FleetWise FleetWise 価格ページの「AWS IoT」を参照してください。

ステップ 2: 車両モデルを作成する

> **Important**
>
> AWS IoT FleetWise コンソールでは、ビジョンシステムのデータ信号を使用して車両モデルを作成することはできません。代わりに、 AWS CLIを使用してください。

車両モデルを使用して、車両の形式を標準化し、作成する車両のシグナル間の関係を定義できます。車両モデルを作成すると、シグナルカタログも作成されます。シグナルカタログは、車両モデルの作成に再利用できる標準化されたシグナルのコレクションです。シグナルは、車両データとそのメタデータを格納するために定義する基本構造です。現時点では、 AWS IoT FleetWise AWS リージョンサービスはアカウントごとに 1 つのシグナルカタログのみをサポートしています。これにより、車両のフリートからの処理データが整合していることを検証できます。

車両モデルを作成するには

1. AWS IoT FleetWise コンソールを開きます。
2. ナビゲーションペインで、[車両モデル] を選択します。
3. [車両モデル] ページで、[車両モデルを作成] を選択します。

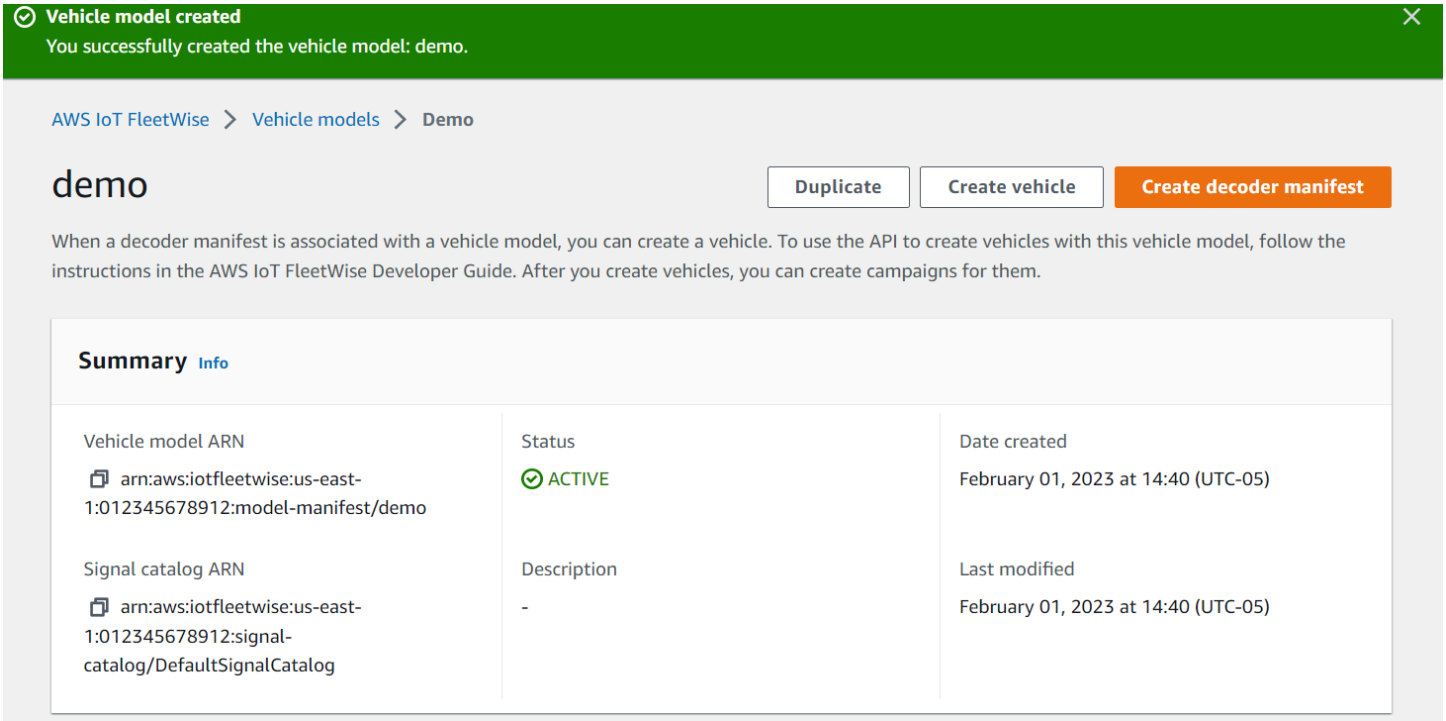
4. [一般的な情報] セクションに、車両モデルの名前 (Vehicle1 など) と、必要に応じて説明を入力します。[次へ] を選択します。
5. シグナルカタログから 1 つ以上のシグナルを選択します。カタログの検索でシグナルを名前でフィルタリングするか、リストからシグナルを選択できます。例えば、タイヤ空気圧とブレーキ圧力のシグナルを選択すると、これらのシグナルに関連するデータを収集できます。[次へ] を選択します。
6. ローカルデバイスから .dbc ファイルを選択してアップロードします。[次へ] を選択します。

   > **Note**
   >
   > このチュートリアルでは、サンプル .dbc ファイルをダウンロードして、このステップ用にアップロードできます。

7. 車両モデルに属性を追加し、[次へ] を選択します。

   a. [名前] - 車両属性の名前を入力します (製造元の名前や製造日など)。
   b. [データ型] - [データ型] メニューで、データ型を選択します。
   c. [単位] - (オプション) 単位の値を入力します (キロメートルや摂氏など)。
   d. [パス] - (オプション) シグナルのパスの名前を入力します (`Vehicle.Engine.Light.` など)。ドット (.) は子シグナルであることを示します。
   e. [デフォルト値] - (オプション) デフォルト値を入力します。
   f. [説明] - (オプション) 属性の説明を入力します。

8. 構成を確認します。準備が完了したら、[作成] を選択します。車両モデルが正常に作成されたことを示す通知が表示されます。



ステップ 3: デコーダーマニフェストを作成する

作成した車両モデルにはデコーダーマニフェストが関連付けられます。これらには、 AWS IoT FleetWise が車両データをバイナリ形式から解読可能な分析可能な値にデコードして変換するのに役立つ情報が含まれています。デコーダーマニフェストの構成に役立つコンポーネントが、ネットワークインターフェイスとデコーダーシグナルです。ネットワークインターフェイスには、車両ネットワークが使用する CAN または OBD プロトコルに関する情報が含まれています。デコーダーシグナルは、特定のシグナルのデコード情報を提供します。

デコーダーマニフェストを作成するには

1. AWS IoT FleetWise コンソールを開きます。
2. ナビゲーションペインで、[車両モデル] を選択します。
3. [車両モデル] セクションで、デコーダーマニフェストの作成に使用する車両モデルを選択します。
4. [デコーダーマニフェストを作成] を選択します。

ステップ 4: デコーダーマニフェストを構成する

デコーダーマニフェストを構成するには

> **Important**
>
> AWS IoT FleetWise コンソールを使用してデコーダーマニフェストにビジョンシステムデータ信号を設定することはできません。代わりに、 AWS CLIを使用してください。詳細については、「デコーダーマニフェストの作成 (AWS CLI)」を参照してください。

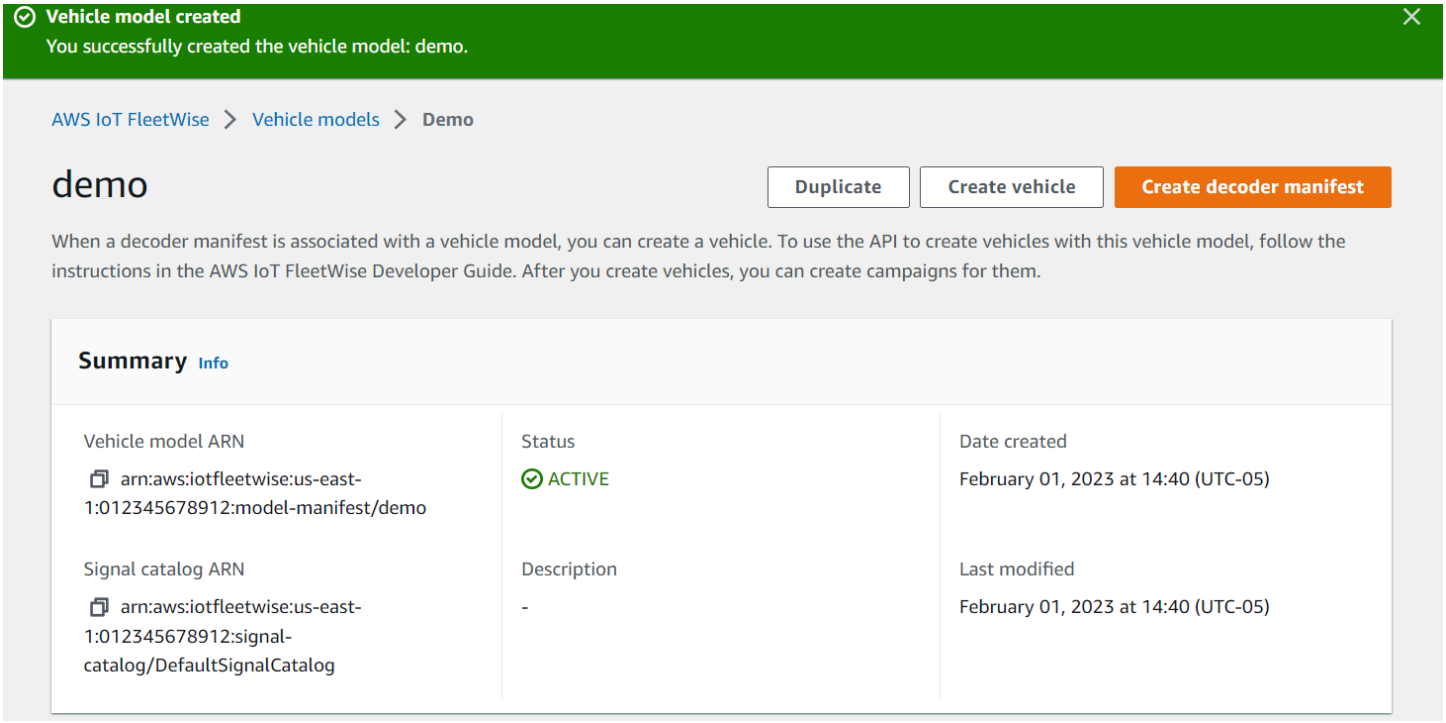
1. デコーダーマニフェストを識別しやすくするために、名前と必要に応じて説明を入力します。[次へ] を選択します。
2. 1 つ以上のネットワークインターフェイスを追加するには、CAN_INTERFACE タイプまたは OBD_INTERFACE タイプを選択します。
   - オンボードダイアグノーシス (OBD) インターフェイス - 電子制御ユニット (ECU) 間での自己診断データの通信方法を定義するプロトコルが必要な場合は、このインターフェイスタイプを選択します。このプロトコルには、車両の問題のトラブルシューティングに役立つ標準の故障診断コード (DTC) が多数定義されています。
   - コントローラーエリアネットワーク (CAN バス) インターフェイス - ECU 間でのデータの通信方法を定義するプロトコルが必要な場合は、このインターフェイスタイプを選択します。ECU には、エンジンコントロールユニット、エアバッグ、オーディオシステムなどがあります。
3. ネットワークインターフェイス名を入力します。
4. ネットワークインターフェイスにシグナルを追加するには、リストから 1 つ以上のシグナルを選択します。
5. 前のステップで追加したシグナル用のデコーダーシグナルを選択します。デコード情報を提供するには、.dbc ファイルをアップロードします。車両モデル内の各シグナルは、リストから選択できるデコーダーシグナルとペアにする必要があります。
6. 別のネットワークインターフェイスを追加するには、[ネットワークインターフェイスを追加] を選択します。ネットワークインターフェイスの追加が完了したら、[次へ] を選択します。
7. 構成を確認し、[作成] を選択します。デコーダーマニフェストが正常に作成されたことを示す通知が表示されます。

ステップ 5: 車両を作成する

AWS IoT では FleetWise、車両は実際の物理的な車両を仮想的に表現したものです。同じ車両モデルから作成したすべての車両は、同じシグナルグループを継承します。作成した各車両は、新しく作成した IoT モノに対応します。すべての車両は、デコーダーマニフェストに関連付ける必要があります。

前提条件

1. 車両モデルとデコーダーマニフェストを作成済みであることを確認します。また、車両モデルのステータスが ACTIVE であることも確認します。
   a. 車両モデルのステータスが ACTIVE であることを確認するには、 AWS IoT FleetWise コンソールを開きます。
   b. ナビゲーションペインで、[車両モデル] を選択します。
   c. [概要] セクションの [ステータス] で、車両のステータスを確認します。



車両を作成するには

1. AWS FleetWise コンソールを開きます。
2. ナビゲーションペインで、[車両] を選択します。

3. [車両を作成] を選択します。
4. 車両のプロパティを定義するには、車両名を入力し、モデルマニフェスト (車両モデル) とデコーダーマニフェストを選択します。
5. (オプション) 車両の属性を定義するには、キーと値のペアを入力し、[属性を追加] を選択します。
6. (オプション) AWS リソースにラベルを付けるには、タグを追加し、[新しいタグを追加] を選択します。
7. [次へ] を選択します。
8. 車両証明書を構成するには、独自の証明書をアップロードするか、[新しい証明書の自動生成] を選択します。セットアップを迅速に行うには、証明書を自動生成することをお勧めします。既に証明書がある場合は、代わりにそれを使用できます。
9. パブリックキーとプライベートキーのファイルをダウンロードし、[次へ] を選択します。
10. 車両証明書にポリシーをアタッチするには、既存のポリシー名を入力するか、新しいポリシーを作成します。新しいポリシーを作成するには、[ポリシーを作成] を選択し、[次へ] を選択します。
11. 構成を確認します。完了したら、[車両を作成] を選択します。

ステップ 6: キャンペーンを作成する

AWS IoT では FleetWise、車両からクラウドへのデータの選択、収集、転送を容易にするためにキャンペーンが使用されます。キャンペーンにはデータ収集スキームが含まれています。データ収集スキームは、条件ベースの収集スキームまたは時間ベースの収集スキームを使用したデータの収集方法をエッジエージェントソフトウェアに指示します。

キャンペーンを作成するには

1. AWS IoT FleetWise コンソールを開きます。
2. ナビゲーションペインで、[キャンペーン] を選択します。
3. [キャンペーンを作成] を選択します。
4. キャンペーンの名前と、必要に応じて説明を入力します。
5. キャンペーンのデータ収集スキームを設定するには、データ収集スキームを手動で定義するか、.json ファイルをローカルデバイスからアップロードできます。.json ファイルをアップロードすると、データ収集スキームが自動的に定義されます。

a. データ収集スキームを手動で定義するには、[データ収集スキームを定義] を選択し、キャンペーンに使用するデータ収集スキームのタイプを選択します。[条件ベース] の収集スキームまたは [時間ベース] の収集スキームを選択できます。

b. [時間ベース] の収集スキームを選択した場合は、キャンペーンで車両データを収集する期間を指定する必要があります。

c. 条件ベースの収集スキームを選択した場合は、収集するデータを認識する式を指定する必要があります。シグナルの名前を表す変数、比較演算子、比較値を指定してください。

d. (オプション) 式の言語バージョンを選択するか、デフォルト値の 1 のままにします。

e. (オプション) 2 つのデータ収集イベント間のトリガー間隔を指定します。

f. データを収集するには、エッジエージェントソフトウェアの [トリガーモード] 条件を選択します。デフォルトでは、Edge Agent for AWS IoT FleetWise ソフトウェアは、条件が満たされるたびに常にデータを収集します。または、[最初のトリガー時] を選択して、条件が初めて満たされたときにのみデータを収集することもできます。

g. (オプション) その他の高度なスキームオプションを選択できます。

6. データ収集スキームでデータを収集するシグナルを指定するには、メニューからシグナルの名前を検索します。

7. (オプション) 最大サンプル数または最小サンプリング間隔を選択できます。シグナルをさらに追加することもできます。

8. [次へ] を選択します。

9. キャンペーンでデータを転送する先のストレージを定義します。Amazon S3 または Amazon Timestream にデータを保存できます。

   a. Amazon S3 — AWS IoT FleetWise へのアクセス権限がある S3 バケットを選択します。

   b. Amazon Timestream - Timestream データベースとテーブル名を選択します。 AWS IoT FleetWise タイムストリームへのデータ送信を許可する IAM ロールを入力します。

10. [次へ] を選択します。

11. 検索ボックスから車両属性または車両名を選択します。

12. 車両に選択した属性または名前に関連する値を入力します。

13. キャンペーンでデータを収集する車両を選択します。[次へ] を選択します。

14. キャンペーンの構成を確認し、[キャンペーンを作成] を選択します。ユーザーまたはユーザーのチームは、このキャンペーンを車両にデプロイする必要があります。

ステップ 7: クリーンアップする

このチュートリアルで使用したリソースに追加料金がかからないように、 AWS CloudFormation スタックとすべてのスタックリソースを削除してください。

AWS CloudFormation スタックを削除するには:

1. AWS CloudFormation コンソールを開きます。
2. スタックのリストから、ステップ 1 で作成したスタックを選択します。
3. [削除] をクリックします。
4. 削除を確認するには、[削除] を選択します。スタックの消去には約 15 分かかります。

次のステップ

1. キャンペーンで収集した車両データを処理して視覚化できます。詳細については、「車両データの処理と視覚化」を参照してください。
2. AWS IoT FleetWise の問題をトラブルシューティングして解決できます。詳細については、「AWS IoT FleetWise のトラブルシューティング」を参照してください。

クラウドへのデータの取り込み

AWS IoT FleetWise 用エッジエージェントソフトウェアは、車両にインストールして実行したときに、車両とクラウド間の安全な通信を支援するように設計されています。

> **Note**
>
> - AWS IoT FleetWise は、重篤な人身事故、死亡事故、環境被害、または物的損害につながる可能性のある危険な環境や重要なシステムの運用に使用したり、それらの運用に関連して使用したりすることを意図したものではありません。AWS IoT FleetWise の使用を通じて収集される車両データは、情報提供のみを目的としています。車両機能を制御または操作するために AWS IoT FleetWise を使用することはできません。
> - AWS IoT FleetWise の使用を通じて収集される車両データは、適用される車両安全規制の下で生じる可能性のあるコンプライアンス義務 (安全監視や報告義務など) を満たす目的を含め、ユースケースに応じて正確性を評価する必要があります。このような評価には、他の業界標準の手段や情報源 (車両の運転者からの報告など) を通じた情報の収集とレビューを含める必要があります。

データをクラウドに取り込むには、次の操作を行います。

1. AWS IoT FleetWise 用エッジエージェントソフトウェアを開発し、車両にインストールします。エッジエージェントソフトウェアの作成方法の詳細については、次の手順を実行して、AWS IoT FleetWise 用エッジエージェントソフトウェアデベロッパーガイドをダウンロードしてください。
   1. AWS IoT FleetWise コンソールに移動します。
   2. サービスのホームページで、[AWS IoT FleetWise の使用を開始する] セクションの [エッジエージェントを調べる] を選択します。
2. 車両モデルの作成に使用するシグナルを含むシグナルカタログを作成またはインポートします。詳細については、シグナルカタログの作成 (AWS CLI) および シグナルカタログのインポート (AWS CLI) を参照してください。

   > **Note**
   >
   > - AWS IoT FleetWise コンソールを使用して最初の車両モデルを作成する場合は、シグナルカタログを手動で作成する必要はありません。最初の車両モデルを作成すると、AWS

IoT FleetWise によって自動的にシグナルカタログが作成されます。詳細については、「車両モデルの作成 (コンソール)」を参照してください。

- 現在、AWS IoT FleetWise は、各AWS リージョンで AWS アカウントごとに 1 つのシグナルカタログをサポートしています。

3. シグナルカタログ内のシグナルを使用して、車両モデルを作成します。詳細については、「車両モデルの作成」を参照してください。

> **Note**
>
> - AWS IoT FleetWise コンソールを使用して車両モデルを作成する場合は、.dbc ファイルをアップロードしてシグナルをインポートできます。.dbc は、コントローラーエリアネットワーク (CAN バス) データベースでサポートされるファイル形式です。車両モデルが作成された後、新しいシグナルが自動的にシグナルカタログに追加されます。詳細については、「車両モデルの作成 (コンソール)」を参照してください。
> - `CreateModelManifest` API オペレーションを使用して車両モデルを作成する場合は、`UpdateModelManifest` API オペレーションを使用して車両モデルをアクティブ化する必要があります。詳細については、「車両モデルの更新 (AWS CLI)」を参照してください。
> - AWS IoT FleetWise コンソールを使用して車両モデルを作成する場合は、AWS IoT FleetWise によって自動的に車両モデルがアクティブ化されます。

4. デコーダーマニフェストを作成します。デコーダーマニフェストには、前のステップで作成した車両モデルに指定されているすべてのシグナルのデコード情報が格納されます。デコーダーマニフェストは、作成した車両モデルに関連付けられます。詳細については、「デコーダーマニフェストの作成と管理」を参照してください。

> **Note**
>
> - `CreateDecoderManifest` API オペレーションを使用してデコーダーマニフェストを作成する場合は、`UpdateDecoderManifest` API オペレーションを使用してデコーダーマニフェストをアクティブ化する必要があります。詳細については、「デコーダーマニフェストの更新 (AWS CLI)」を参照してください。
> - AWS IoT FleetWise コンソールを使用してデコーダーマニフェストを作成する場合は、AWS IoT FleetWise によって自動的にデコーダーマニフェストがアクティブ化されます。

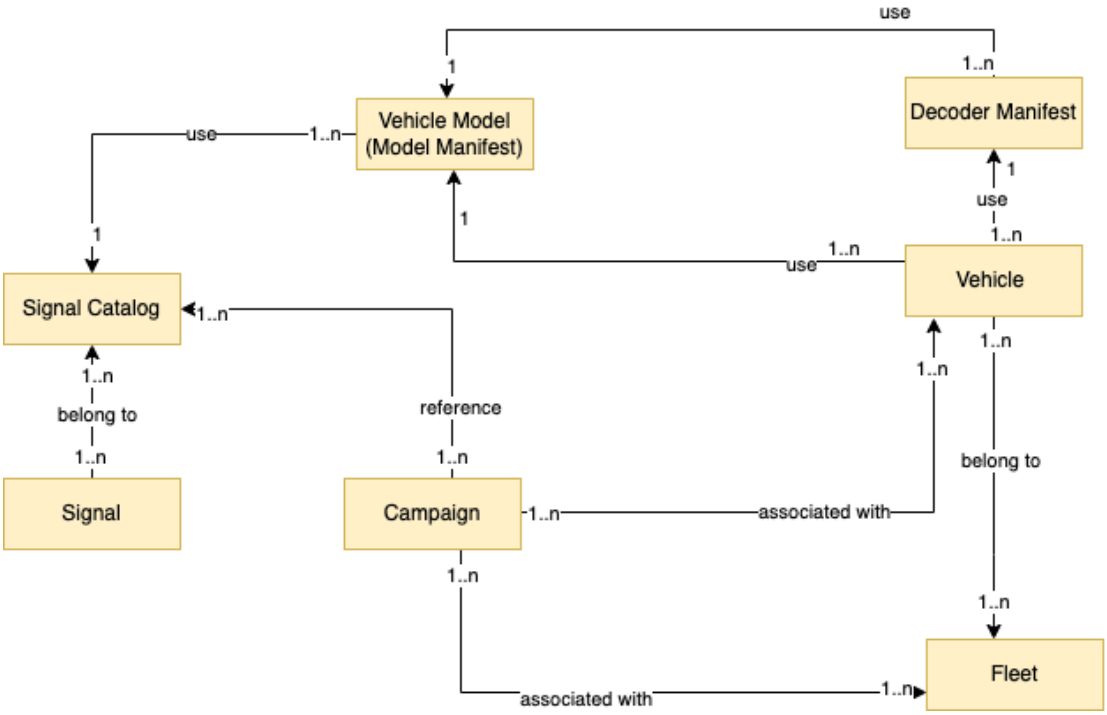
5. 車両モデルから車両を作成します。同じ車両モデルから作成された車両は、同じシグナルのグループを継承します。データをクラウドに取り込む前に、AWS IoT Core を使用して車両をプロビジョニングする必要があります。詳細については、「車両の作成、プロビジョニング、管理」を参照してください。
6. (オプション) 車両のグループを表すフリートを作成し、個々の車両をそのフリートに関連付けます。これにより、複数の車両を同時に管理できます。詳細については、「フリートの作成と管理」を参照してください。
7. キャンペーンを作成します。キャンペーンは、車両または車両のフリートにデプロイされます。キャンペーンにより、データをどのように選択して収集し、クラウドに転送するかに関する指示がエッジエージェントソフトウェアに与えられます。詳細については、「キャンペーンによるデータの収集と転送」を参照してください。

> **Note**
>
> AWS IoT FleetWise でキャンペーンを車両またはフリートにデプロイできるようにするには、事前に `UpdateCampaign` API オペレーションを使用してキャンペーンを承認する必要があります。詳細については、「キャンペーンの更新 (AWS CLI)」を参照してください。

エッジエージェントソフトウェアは、予約済みトピック (`$aws/iotfleetwise/vehicles/`*`vehicleName`*`/signals`) を使用して車両データを AWS IoT Core に転送します。その後、データは AWS IoT FleetWise に送信されます。AWSIoT FleetWise は、そのデータを Timestream テーブルまたは Amazon S3 バケットに配信します。Timestream を使用してデータのクエリを実行したり、Amazon QuickSight または Grafana を使用してデータを視覚化したりできます。詳細については、「車両データの処理と視覚化」を参照してください。

車両のモデリング

AWS IoT FleetWise は、クラウドで車両の仮想表現を構築するために使用できる車両モデリングフレームワークを提供します。車両のモデル化に使用するコアコンポーネントには、シグナル、シグナルカタログ、車両モデル、デコーダーマニフェストがあります。



シグナル

シグナルは、車両データとそのメタデータを格納するために定義する基本構造です。シグナルには、属性、ブランチ、センサー、アクチュエータがあります。例えば、車内の温度値を受け取るセンサーを作成し、そのメタデータ (センサー名、データ型、単位など) を格納できます。詳細については、「シグナルカタログの作成と管理」を参照してください。

シグナルカタログ

シグナルカタログには、シグナルのコレクションが格納されます。シグナルカタログ内のシグナルを使用して、さまざまなプロトコルやデータ形式を使用する車両をモデル化できます。例えば、異なる自動車メーカーの 2 台の車があるとします。1 台はコントローラーエリアネットワーク (CAN バス) プロトコルを使用し、もう 1 台はオンボードダイアグノーシス (OBD) プロトコルを使用しています。シグナルカタログには、車内の温度値を受信するセンサーを定義することができます。このセンサーを、両方の車の熱電対を表すために使用できます。詳細については、「シグナルカタログの作成と管理」を参照してください。

車両モデル (モデルマニフェスト)

車両モデルとは、車両の形式を標準化し、車両内のシグナル間の関係を定義するために使用できる宣言的な構造です。車両モデルにより、同じタイプの複数の車両に一貫した情報が適用されます。車両モデルを作成するには、シグナルを追加します。詳細については、「車両モデルの作成と管理」を参照してください。

デコーダーマニフェスト

デコーダーマニフェストには、車両モデル内の各シグナルのデコード情報が含まれています。車両内のセンサーやアクチュエータが送信するのは、低レベルのメッセージ (バイナリデータ) です。デコーダーマニフェストを使用すると、 AWS IoT FleetWise はバイナリデータを人間が読める値に変換できます。すべてのデコーダーマニフェストは車両モデルに関連付けられます。詳細については、「デコーダーマニフェストの作成と管理」を参照してください。

AWS IoT FleetWise コンソールまたは API を使用して、次のように車両をモデル化できます。

1. 車両モデルの作成に使用するシグナルを含むシグナルカタログを作成またはインポートします。詳細については、シグナルカタログの作成 (AWS CLI)およびシグナルカタログのインポート (AWS CLI)を参照してください。

   > **Note**
   >
   > - AWS IoT FleetWise コンソールを使用して最初の車両モデルを作成する場合、信号カタログを手動で作成する必要はありません。初めて車両モデルを作成すると、 AWS IoT FleetWise が自動的に信号カタログを作成します。詳細については、「車両モデルの作成 (コンソール)」を参照してください。
   > - AWS IoT FleetWise は現在、 AWS アカウントごとにシグナルカタログをサポートしています AWS リージョン。

2. シグナルカタログ内のシグナルを使用して、車両モデルを作成します。詳細については、「車両モデルの作成」を参照してください。

   > **Note**
   >
   > - AWS IoT FleetWise コンソールを使用して車両モデルを作成する場合、.dbc ファイルをアップロードして信号をインポートできます。.dbc は、コントローラーエリアネットワーク (CAN バス) データベースがサポートするファイル形式です。車両モデルが作成

された後、新しいシグナルが自動的にシグナルカタログに追加されます。詳細については、「車両モデルの作成 (コンソール)」を参照してください。

- `CreateModelManifest` API オペレーションを使用して車両モデルを作成する場合は、`UpdateModelManifest` API オペレーションを使用して車両モデルをアクティブ化する必要があります。詳細については、「車両モデルの更新 (AWS CLI)」を参照してください。
- AWS IoT FleetWise コンソールを使用して車両モデルを作成すると、 AWS IoT FleetWise が自動的に車両モデルを起動します。

3. デコーダーマニフェストを作成します。デコーダーマニフェストには、前のステップで作成した車両モデルに指定されているすべてのシグナルのデコード情報が格納されます。デコーダーマニフェストは、作成した車両モデルに関連付けられます。詳細については、「デコーダーマニフェストの作成と管理」を参照してください。

> **Note**
>
> - `CreateDecoderManifest` API オペレーションを使用してデコーダーマニフェストを作成する場合は、`UpdateDecoderManifest` API オペレーションを使用してデコーダーマニフェストをアクティブ化する必要があります。詳細については、「デコーダーマニフェストの更新 (AWS CLI)」を参照してください。
> - AWS IoT FleetWise コンソールを使用してデコーダーマニフェストを作成すると、AWS IoT FleetWise は自動的にデコーダーマニフェストをアクティベートします。

CAN バスデータベースは .dbc ファイル形式をサポートしています。.dbc ファイルをアップロードして、シグナルとデコーダーシグナルをインポートできます。サンプルの .dbc ファイルを取得するには、次の手順を実行します。

.dbc ファイルを取得するには

1. .zip をダウンロードします。EngineSignals
2. `EngineSignals.zip` ファイルをダウンロードしたディレクトリに移動します。
3. ファイルを解凍し、`EngineSignals.dbc` としてローカルに保存します。

トピック

- シグナルカタログの作成と管理

- 車両モデルの作成と管理
- デコーダーマニフェストの作成と管理

シグナルカタログの作成と管理

> **Note**
>
> デモスクリプトをダウンロードして、ROS 2 メッセージを、シグナルカタログと互換性のある VSS JSON ファイルに変換できます。詳細については、「ビジョンシステムデータデベロッパーガイド」を参照してください。

信号カタログは、車両モデルの作成に再利用できる標準化された信号のコレクションです。 AWS IoT は車両信号仕様 (VSS) FleetWise をサポートしており、これに従って信号を定義できます。シグナルには次のタイプがあります。

属性

属性は、製造元や製造日など、通常は変更されない静的な情報を表します。

ブランチ

ブランチとは、ネストされた構造内のシグナルを表します。ブランチは、シグナルの階層を明確に示します。例えば、`Vehicle` というブランチに `Powertrain` という子ブランチがあるとします。`Powertrain` ブランチには `combustionEngine` という子ブランチがあります。`combustionEngine` ブランチを特定するには、`Vehicle.Powertrain.combustionEngine` という式を使用します。

センサー

センサーデータは、液面、温度、振動、電圧などの車両の状態について、現在の状態と経時的な変化を報告します。

アクチュエータ

アクチュエータデータは、モーター、ヒーター、ドアロックなど、車両デバイスの状態を報告します。車両デバイスの状態を変更すると、アクチュエータデータが更新される可能性があります。例えば、ヒーターを表すアクチュエータを定義できます。このアクチュエータは、ヒーターをオンまたはオフにしたときに新しいデータを受け取ります。

カスタム構造

カスタム構造 (構造体とも呼ばれる) は、複雑なデータ構造または高次のデータ構造を表します。これにより、同じソースから生成されたデータの論理的なバインドやグループ化が容易になります。構造体は、複雑なデータ型や高次の形状を表すなど、アトミック操作でデータを読み書きする場合に使用します。

構造体型のシグナルは、プリミティブデータ型の代わりに構造体データ型への参照を使用してシグナルカタログで定義します。構造体は、センサー、属性、アクチュエータ、ビジョンシステムデータ型など、あらゆるタイプのシグナルに使用できます。構造体型のシグナルが送受信される場合、 AWS IoT は、 FleetWise 含まれているすべての項目に有効な値があることを想定しているため、すべての項目が必須です。例えば、構造体内に項目として Vehicle.Camera.Image.height、Vehicle.Camera.Image.width、Vehicle.Camera.Image.data が含まれている場合、送信されたシグナルには、これらすべての項目の値が含まれていることが期待されます。

> **Note**
>
> ビジョンシステムデータはプレビューリリースであり、変更される可能性があります。

カスタムプロパティ

カスタムプロパティは複雑なデータ構造のメンバーを表します。プロパティのデータ型は、プリミティブまたは別の構造体のいずれかになります。

構造体とカスタムプロパティを使用して高次の形状を表現する場合、意図した高次の形状は常にツリー構造として定義され、視覚化されます。カスタムプロパティはすべてのリーフノードを定義するために使用し、構造体はリーフ以外のすべてのノードを定義するために使用します。

> **Note**
>
> - AWS IoT FleetWise コンソールを使用して最初の車両モデルを作成する場合、信号カタログを手動で作成する必要はありません。初めて車両モデルを作成すると、 AWS IoT FleetWise が自動的に信号カタログを作成します。詳細については、「車両モデルの作成 (コンソール)」を参照してください。
> - AWS IoT FleetWise コンソールを使用して車両モデルを作成する場合、.dbc ファイルをアップロードして信号をインポートできます。.dbc は、コントローラーエリアネットワー

ク (CAN バス) データベースがサポートするファイル形式です。車両モデルが作成された後、新しいシグナルが自動的にシグナルカタログに追加されます。詳細については、「車両モデルの作成 (コンソール)」を参照してください。

- AWS IoT FleetWise は現在、 AWS アカウント 地域ごとに信号カタログをサポートしています。

AWS IoT FleetWise には、シグナルカタログの作成と管理に使用できる次の API オペレーションが用意されています。

- CreateSignalCatalog— 新しいシグナルカタログを作成します。
- ImportSignalCatalog— JSON ファイルをアップロードして、シグナルをインポートしてシグナルカタログを作成します。シグナルは VSS に従って定義し、JSON 形式で保存する必要があります。
- UpdateSignalCatalog— シグナルを更新、削除、または追加することにより、既存のシグナルカタログを更新します。
- DeleteSignalCatalog— 既存のシグナルカタログを削除します。
- ListSignalCatalogs— すべてのシグナルカタログの要約をページ分割したリストを取得します。
- ListSignalCatalogNodes— 特定のシグナルカタログ内のすべてのシグナル (ノード) のサマリーをページ分割したリストを取得します。
- GetSignalCatalog— シグナルカタログに関する情報を取得します。

チュートリアル

- シグナルの構成
- シグナルカタログの作成 (AWS CLI)
- シグナルカタログのインポート
- シグナルカタログの更新 (AWS CLI)
- シグナルカタログの削除 (AWS CLI)
- シグナルカタログ情報の取得 (AWS CLI)

シグナルの構成

このセクションでは、ブランチ、属性、センサー、アクチュエータの構成方法を説明します。

トピック

ブランチの構成

ブランチを構成するには、以下の情報を指定します。

- `fullyQualifiedName` - ブランチの完全修飾名。ブランチのパスにブランチの名前を続けたものです。子ブランチを参照するには、ドット (.) を使用します。例えば、`Vehicle.Chassis.SteeringWheel` は `SteeringWheel` ブランチの完全修飾名です。`Vehicle.Chassis.` はこのブランチのパスを示します。

  完全修飾名には最大 150 文字を入力できます。有効な文字は、a～z、A～Z、0～9、コロン (:)、アンダースコア (_) です。
- (オプション) `Description` - ブランチの説明。

  説明には最大 2048 文字を入力できます。有効な文字は、a～z、A～Z、0～9、: (コロン)、_ (アンダースコア)、- (ハイフン) です。
- (オプション) `deprecationMessage` - 移動または削除されるノードやブランチに関する非推奨メッセージ。

  deprecationMessage には最大 2048 文字を入力できます。有効な文字は、a～z、A～Z、0～9、: (コロン)、_ (アンダースコア)、- (ハイフン) です。
- (オプション) `comment` - 説明に追加するコメント。コメントは、ブランチに関する追加情報を提供するために使用できます。例えば、ブランチの意図や、関連するブランチへの参照を示すことができます。

  コメントには最大 2048 文字を入力できます。有効な文字は、a～z、A～Z、0～9、: (コロン)、_ (アンダースコア)、- (ハイフン) です。

属性の構成

属性を構成するには、以下の情報を指定します。

- dataType— アトリビュートのデータ型
  は、INT8、UINT8、INT16、UINT16、INT32、INT64、UINT64、UINT64、BOOLEAN、FLOAT、DOUBL
  のいずれかである必要がありま
  す。、UINT64_ARRAY、BOOLEAN_ARRAY、FLOAT_ARRAY、DOUBLE_ARRAY、STRING_ARRAY、
  またはデータ型ブランチで定義されたカスタム構造体。 fullyQualifiedName
- fullyQualifiedName - 属性の完全修飾名は、属性のパスに属性名を続
  けたものです。子シグナルを参照するには、ドット (.) を使用します。例え
  ば、Vehicle.Chassis.SteeringWheel.Diameter は Diameter 属性の完全修飾名で
  す。Vehicle.Chassis.SteeringWheel. はこの属性のパスを示します。

  完全修飾名には最大 150 文字を入力できます。有効な文字は、a~z、A~Z、0~9、: (コロン)、_
  (アンダースコア) です。
- (オプション) Description - 属性の説明。

  説明には最大 2048 文字を入力できます。有効な文字は、a~z、A~Z、0~9、: (コロン)、_ (アン
  ダースコア)、- (ハイフン) です。
- (オプション) unit - 属性の科学単位 (km、摂氏など)。
- (オプション) min - 属性の最小値。
- (オプション) max - 属性の最大値。
- (オプション) defaultValue - 属性のデフォルト値。
- (オプション) assignedValue - 属性に割り当てられた値。
- (オプション) allowedValues - 属性が受け入れる値のリスト。
- (オプション) deprecationMessage - 移動または削除されるノードやブランチに関する非推奨
  メッセージ。

  deprecationMessage には最大 2048 文字を入力できます。有効な文字は、a~z、A~Z、0~9、:
  (コロン)、_ (アンダースコア)、- (ハイフン) です。
- (オプション) comment - 説明に追加するコメント。コメントは、属性に関する追加情報を提供す
  るために使用できます。例えば、属性の意図や、関連する属性への参照を示すことができます。

  コメントには最大 2048 文字を入力できます。有効な文字は、a~z、A~Z、0~9、: (コロン)、_
  (アンダースコア)、- (ハイフン) です。

センサーまたはアクチュエータの構成

センサーまたはアクチュエータを構成するには、以下の情報を指定します。

- dataType— 信号のデータ型は次のいずれかでなければなりません:INT8、UINT8、INT16、UINT16、INT32、INT64、UINT64、UINT64、BOOLEAN、FLOAT、DOUBLE
  文字
  列、UNIX_TIMESTAMP、INT8_ARRAY、UINT8_ARRAY、UINT16_ARRAY、UINT16_ARRAY、INT32_A
  またはデータ型ブランチで定義されたカスタム構造体。 fullyQualifiedName
- fullyQualifiedName - シグナルの完全修飾名は、シグナルのパスにシグナルの
  名前を続けたものです。子シグナルを参照するには、ドット (.) を使用します。例えば、Vehicle.Chassis.SteeringWheel.HandsOff.HandsOffSteeringState は HandsOffSteeringState アクチュエータの完全修飾名です。Vehicle.Chassis.SteeringWheel.HandsOff. はこのアクチュエータのパスを示します。

  完全修飾名には最大 150 文字を入力できます。有効な文字は、a～z、A～Z、0～9、: (コロン)、_ (アンダースコア) です。
- (オプション) Description - シグナルの説明。

  説明には最大 2048 文字を入力できます。有効な文字は、a～z、A～Z、0～9、: (コロン)、_ (アンダースコア)、- (ハイフン) です。
- (オプション) unit - シグナルの科学単位 (km、摂氏など)。
- (オプション) min - シグナルの最小値。
- (オプション) max - シグナルの最大値。
- (オプション) assignedValue - シグナルに割り当てられた値。
- (オプション) allowedValues - シグナルが受け入れる値のリスト。
- (オプション) deprecationMessage - 移動または削除されるノードやブランチに関する非推奨メッセージ。

  deprecationMessage には最大 2048 文字を入力できます。有効な文字は、a～z、A～Z、0～9、: (コロン)、_ (アンダースコア)、- (ハイフン) です。
- (オプション) comment - 説明に追加するコメント。コメントは、センサーやアクチュエータに関する追加情報を提供するために使用できます。例えば、それらの意図や、関連するセンサーまたはアクチュエータへの参照を示すことができます。

  コメントには最大 2048 文字を入力できます。有効な文字は、a～z、A～Z、0～9、: (コロン)、_ (アンダースコア)、- (ハイフン) です。

複雑なデータ型の設定

複雑なデータ型は、ビジョンシステムをモデル化するときに使用します。これらのデータ型は、ブランチに加えて、構造 (構造体とも呼ばれる) とプロパティで構成されます。構造体は、画像のように、複数の値で記述されるシグナルです。プロパティは、プリミティブデータ型 (UINT8 など) や別の構造体 (タイムスタンプなど) など、構造体のメンバーを表します。例えば、Vehicle.Cameras.Front はブランチを表し、Vehicle.Cameras.Front.Image は構造体を表し、Vehicle.Cameras.Timestamp はプロパティを表します。

次の複雑なデータ型の例は、シグナルとデータ型を 1 つの JSON ファイルにエクスポートする方法を示しています。

Example 複雑なデータ型

```
{
  "Vehicle": {
    "type": "branch"
    // Signal tree
  },
  "ComplexDataTypes": {
    "VehicleDataTypes": {
      // complex data type tree
      "children": {
        "branch": {
          "children": {
            "Struct": {
              "children": {
                "Property": {
                  "type": "property",
                  "datatype": "Data type",
                  "description": "Description",
                  //                  ...
                }
              },
              "description": "Description",
              "type": "struct"
            }
          }
          "description": "Description",
          "type": "branch"
        }
      }
    }
```

```
  }
}
```

> **Note**
>
> デモスクリプトをダウンロードして、ROS 2 メッセージを、シグナルカタログと互換性のある VSS JSON ファイルに変換できます。詳細については、「ビジョンシステムデータデベロッパーガイド」を参照してください。
> ビジョンシステムデータはプレビューリリースであり、変更される可能性があります。

構造体の設定

カスタム構造 (または構造体) を設定するには、以下の情報を指定します。

- `fullyQualifiedName` — カスタム構造の完全修飾名。例えば、カスタム構造の完全修飾名は `ComplexDataTypes.VehicleDataTypes.SVMCamera` です。

  完全修飾名には最大 150 文字を入力できます。有効な文字は、a～z、A～Z、0～9、: (コロン)、_ (アンダースコア) です。
- (オプション) `Description` - シグナルの説明。

  説明には最大 2048 文字を入力できます。有効な文字は、a～z、A～Z、0～9、: (コロン)、_ (アンダースコア)、- (ハイフン) です。
- (オプション) `deprecationMessage` - 移動または削除されるノードやブランチに関する非推奨メッセージ。

  deprecationMessage には最大 2048 文字を入力できます。有効な文字は、a～z、A～Z、0～9、: (コロン)、_ (アンダースコア)、- (ハイフン) です。
- (オプション) `comment` - 説明に追加するコメント。コメントは、センサーやアクチュエータに関する追加情報を提供するために使用できます。例えば、それらの意図や、関連するセンサーまたはアクチュエータへの参照を示すことができます。

  コメントには最大 2048 文字を入力できます。有効な文字は、a～z、A～Z、0～9、: (コロン)、_ (アンダースコア)、- (ハイフン) です。

プロパティの設定

カスタムプロパティを設定するには、以下の情報を指定します。

- dataType - シグナルのデータ型
  は、INT8、UINT8、INT16、UINT16、INT32、UINT32、INT64、UINT64、BOOLEAN、FLOAT、DOUBL
  のいずれかである必要があります。
- fullyQualifiedName — カスタムプロパティの完全修飾名。例えば、カスタムプロパティの完全修飾名は ComplexDataTypes.VehicleDataTypes.SVMCamera.FPS です。

  完全修飾名には最大 150 文字を入力できます。有効な文字は、a ~ z、A ~ Z、0 ~ 9、: (コロン)、_ (アンダースコア) です。
- (オプション) Description - シグナルの説明。

  説明には最大 2048 文字を入力できます。有効な文字は、a ~ z、A ~ Z、0 ~ 9、: (コロン)、_ (アンダースコア)、- (ハイフン) です。
- (オプション) deprecationMessage - 移動または削除されるノードやブランチに関する非推奨メッセージ。

  deprecationMessage には最大 2048 文字を入力できます。有効な文字は、a ~ z、A ~ Z、0 ~ 9、: (コロン)、_ (アンダースコア)、- (ハイフン) です。
- (オプション) comment - 説明に追加するコメント。コメントは、センサーやアクチュエータに関する追加情報を提供するために使用できます。例えば、それらの意図や、関連するセンサーまたはアクチュエータへの参照を示すことができます。

  コメントには最大 2048 文字を入力できます。有効な文字は、a ~ z、A ~ Z、0 ~ 9、: (コロン)、_ (アンダースコア)、- (ハイフン) です。
- (オプション) dataEncoding — プロパティがバイナリデータかどうかを示します。カスタムプロパティのデータエンコーディングは、BINARY または TYPED のいずれかである必要があります。
- (オプション) structFullyQualifiedName — カスタムプロパティのデータ型が Struct または の場合、カスタムプロパティの構造 (struct) ノードの完全修飾名 StructArray。

  完全修飾名には最大 150 文字を入力できます。有効な文字は、a ~ z、A ~ Z、0 ~ 9、: (コロン)、_ (アンダースコア) です。

シグナルカタログの作成 (AWS CLI)

CreateSignalCatalogAPI オペレーションを使用してシグナルカタログを作成できます。以下の例ではを使用しています AWS CLI。

シグナルカタログを作成するには、次のコマンドを実行します。

設定を含む JSON *signal-catalog-configuration*ファイルの名前に置き換えます。

```
aws iotfleetwise create-signal-catalog --cli-input-json file://signal-catalog-configuration.json
```

- *signal-catalog-name*作成するシグナルカタログの名前に置き換えてください。
- (オプション) *description* は、シグナルカタログの識別に役立つ説明に置き換えます。

ブランチ、属性、センサー、アクチュエータの構成方法の詳細については、「シグナルの構成」を参照してください。

```
{
    "name": "signal-catalog-name",
    "description": "description",
    "nodes": [
  {
    "branch": {
      "fullyQualifiedName": "Types"
    }
  },
  {
    "struct": {
      "fullyQualifiedName": "Types.sensor_msgs_msg_CompressedImage"
    }
  },
  {
    "struct": {
      "fullyQualifiedName": "Types.std_msgs_Header"
    }
  },
  {
    "struct": {
      "fullyQualifiedName": "Types.builtin_interfaces_Time"
    }
  },
  {
    "property": {
      "fullyQualifiedName": "Types.builtin_interfaces_Time.sec",
      "dataType": "INT32",
      "dataEncoding": "TYPED"
    }
```

```
  },
  {
    "property": {
      "fullyQualifiedName": "Types.builtin_interfaces_Time.nanosec",
      "dataType": "UINT32",
      "dataEncoding": "TYPED"
    }
  },
  {
    "property": {
      "fullyQualifiedName": "Types.std_msgs_Header.stamp",
      "dataType": "STRUCT",
      "structFullyQualifiedName": "Types.builtin_interfaces_Time"
    }
  },
  {
    "property": {
      "fullyQualifiedName": "Types.std_msgs_Header.frame_id",
      "dataType": "STRING",
      "dataEncoding": "TYPED"
    }
  },
  {
    "property": {
      "fullyQualifiedName": "Types.sensor_msgs_msg_CompressedImage.header",
      "dataType": "STRUCT",
      "structFullyQualifiedName": "Types.std_msgs_Header"
    }
  },
  {
    "property": {
      "fullyQualifiedName": "Types.sensor_msgs_msg_CompressedImage.format",
      "dataType": "STRING",
      "dataEncoding": "TYPED"
    }
  },
  {
    "property": {
      "fullyQualifiedName": "Types.sensor_msgs_msg_CompressedImage.data",
      "dataType": "UINT8_ARRAY",
      "dataEncoding": "BINARY"
    }
  },
  {
```

```
    "branch": {
      "fullyQualifiedName": "Vehicle",
      "description": "Vehicle"
    }
  },
  {
    "branch": {
      "fullyQualifiedName": "Vehicle.Cameras"
    }
  },
  {
    "branch": {
      "fullyQualifiedName": "Vehicle.Cameras.Front"
    }
  },
  {
    "sensor": {
      "fullyQualifiedName": "Vehicle.Cameras.Front.Image",
      "dataType": "STRUCT",
      "structFullyQualifiedName": "Types.sensor_msgs_msg_CompressedImage"
    }
  },
  {
    "struct": {
      "fullyQualifiedName": "Types.std_msgs_msg_Float64"
    }
  },
  {
    "property": {
      "fullyQualifiedName": "Types.std_msgs_msg_Float64.data",
      "dataType": "DOUBLE",
      "dataEncoding": "TYPED"
    }
  },
  {
    "sensor": {
      "fullyQualifiedName": "Vehicle.Velocity",
      "dataType": "STRUCT",
      "structFullyQualifiedName": "Types.std_msgs_msg_Float64"
    }
  },
  {
    "struct": {
      "fullyQualifiedName": "Types.sensor_msgs_msg_RegionOfInterest"
```

```
    }
  },
  {
    "property": {
      "fullyQualifiedName": "Types.sensor_msgs_msg_RegionOfInterest.x_offset",
      "dataType": "UINT32",
      "dataEncoding": "TYPED"
    }
  },
  {
    "property": {
      "fullyQualifiedName": "Types.sensor_msgs_msg_RegionOfInterest.y_offset",
      "dataType": "UINT32",
      "dataEncoding": "TYPED"
    }
  },
  {
    "property": {
      "fullyQualifiedName": "Types.sensor_msgs_msg_RegionOfInterest.height",
      "dataType": "UINT32",
      "dataEncoding": "TYPED"
    }
  },
  {
    "property": {
      "fullyQualifiedName": "Types.sensor_msgs_msg_RegionOfInterest.width",
      "dataType": "UINT32",
      "dataEncoding": "TYPED"
    }
  },
  {
    "property": {
      "fullyQualifiedName": "Types.sensor_msgs_msg_RegionOfInterest.do_rectify",
      "dataType": "BOOLEAN",
      "dataEncoding": "TYPED"
    }
  },
  {
    "branch": {
      "fullyQualifiedName": "Vehicle.Perception"
    }
  },
  {
    "sensor": {
```

```
      "fullyQualifiedName": "Vehicle.Perception.Obstacle",
      "dataType": "STRUCT",
      "structFullyQualifiedName": "Types.sensor_msgs_msg_RegionOfInterest"
    }
  }
]
}
```

> **Note**
>
> デモスクリプトをダウンロードして、ROS 2 メッセージを、シグナルカタログと互換性のある VSS JSON ファイルに変換できます。詳細については、「ビジョンシステムデータデベロッパーガイド」を参照してください。
> ビジョンシステムデータはプレビューリリースであり、変更される可能性があります。

シグナルカタログのインポート

AWS IoT FleetWise コンソールまたは API を使用して信号カタログをインポートできます。

トピック

- シグナルカタログのインポート (コンソール)
- シグナルカタログのインポート (AWS CLI)

シグナルカタログのインポート (コンソール)

AWS IoT FleetWise コンソールを使用して信号カタログをインポートできます。

> **Important**
>
> 使用できるシグナルカタログは最大 1 つです。シグナルカタログが既に存在する場合、コンソールにはシグナルカタログをインポートするオプションは表示されません。

シグナルカタログをインポートするには

1. AWS IoT FleetWise コンソールを開きます。
2. ナビゲーションペインで、[シグナルカタログ] を選択します。

3. シグナルカタログの概要ページで、[シグナルカタログをインポート] を選択します。
4. シグナルを含むファイルをインポートします。
   - S3 バケットからファイルをアップロードするには:

     a. [S3 からインポート] を選択します。

     b. [S3 を参照] を選択します。

     c. [バケット] で、バケット名またはオブジェクトを入力し、リストから選択します。次に、リストからファイルを選択します。[ファイルを選択] ボタンを選択します。

     または、[S3 URI] に Amazon Simple Storage Service の URI を入力します。詳細については、「Amazon S3 ユーザーガイド」の「バケットへのアクセス方法」を参照してください。
   - コンピュータからファイルをアップロードするには:

     a. [ファイルをインポート] を選択します。

     b. Vehicle Signal Specification (VSS) 形式の .json ファイルをアップロードします。
5. シグナルカタログを確認し、[ファイルをインポート] を選択します。

シグナルカタログのインポート (AWS CLI)

ImportSignalCatalogAPI オペレーションを使用して、シグナルカタログの作成に役立つ JSON ファイルをアップロードできます。JSON ファイルにシグナルを保存するには、Vehicle Signal Specification (VSS) に従う必要があります。以下の例ではを使用しています AWS CLI。

シグナルカタログをインポートするには、次のコマンドを実行します。

- *signal-catalog-name*作成するシグナルカタログの名前に置き換えます。
- (オプション) *description* は、シグナルカタログの識別に役立つ説明に置き換えます。
- VSS *signal-catalog-configuration-vss*で定義されたシグナルを含む JSON 文字列ファイルの名前に置き換えます。

ブランチ、属性、センサー、アクチュエータの構成方法の詳細については、「シグナルの構成」を参照してください。

```
aws iotfleetwise import-signal-catalog \
                --name signal-catalog-name \
```

```
                --description  description \
                --vss file://signal-catalog-configuration-vss.json
```

JSON は、文字列化して vssJson フィールドに渡す必要があります。以下は、VSS で定義されたシグナルの例です。

```
{
 "Vehicle": {
  "type": "branch",
  "children": {
   "Chassis": {
    "type": "branch",
    "description": "All data concerning steering, suspension, wheels, and brakes.",
    "children": {
     "SteeringWheel": {
      "type": "branch",
      "description": "Steering wheel signals",
      "children": {
       "Diameter": {
        "type": "attribute",
        "description": "The diameter of the steering wheel",
        "datatype": "float",
        "unit": "cm",
        "min": 1,
        "max": 50
       },
       "HandsOff": {
        "type": "branch",
        "children": {
         "HandsOffSteeringState": {
          "type": "actuator",
          "description": "HndsOffStrWhlDtSt. Hands Off Steering State",
          "datatype": "boolean"
         },
         "HandsOffSteeringMode": {
          "type": "actuator",
          "description": "HndsOffStrWhlDtMd. Hands Off Steering Mode",
          "datatype": "int8",
          "min": 0,
          "max": 2
         }
        }
       }
```

```
      }
     },
     "Accelerator": {
      "type": "branch",
      "description": "",
      "children": {
       "AcceleratorPedalPosition": {
        "type": "sensor",
        "description": "Throttle__Position. Accelerator pedal position as percent. 0 =
 Not depressed. 100 = Fully depressed.",
        "datatype": "uint8",
        "unit": "%",
        "min": 0,
        "max": 100.000035
       }
      }
     }
    }
   },
   "Powertrain": {
    "type": "branch",
    "description": "Powertrain data for battery management, etc.",
    "children": {
     "Transmission": {
      "type": "branch",
      "description": "Transmission-specific data, stopping at the drive shafts.",
      "children": {
       "VehicleOdometer": {
        "type": "sensor",
        "description": "Vehicle_Odometer",
        "datatype": "float",
        "unit": "km",
        "min": 0,
        "max": 67108863.984375
       }
      }
     },
     "CombustionEngine": {
      "type": "branch",
      "description": "Engine-specific data, stopping at the bell housing.",
      "children": {
       "Engine": {
        "type": "branch",
        "description": "Engine description",
```

```
        "children": {
         "timing": {
          "type": "branch",
          "description": "timing description",
          "children": {
           "run_time": {
            "type": "sensor",
            "description": "Engine run time",
            "datatype": "int16",
            "unit": "ms",
            "min": 0,
            "max": 10000
           },
           "idle_time": {
            "type": "sensor",
            "description": "Engine idle time",
            "datatype": "int16",
            "min": 0,
            "unit": "ms",
            "max": 10000
           }
          }
         }
        }
       }
      }
     }
    },
    "Axle": {
     "type": "branch",
     "description": "Axle signals",
     "children": {
      "TireRRPrs": {
       "type": "sensor",
       "description": "TireRRPrs. Right rear Tire pressure in kilo-Pascal",
       "datatype": "float",
       "unit": "kPaG",
       "min": 0,
       "max": 1020
      }
     }
    }
   }
```

```
},
"Cameras": {
 "type": "branch",
 "description": "Branch to aggregate all cameras in the vehicle",
 "children": {
  "FrontViewCamera": {
   "type": "sensor",
   "datatype": "VehicleDataTypes.SVMCamera",
   "description": "Front view camera"
  },
  "RearViewCamera": {
   "type": "sensor",
   "datatype": "VehicleDataTypes.SVMCamera",
   "description": "Rear view camera"
  },
  "LeftSideViewCamera": {
   "type": "sensor",
   "datatype": "VehicleDataTypes.SVMCamera",
   "description": "Left side view camera"
  },
  "RightSideViewCamera": {
   "type": "sensor",
   "datatype": "VehicleDataTypes.SVMCamera",
   "description": "Right side view camera"
  }
 }
},
"ComplexDataTypes": {
 "VehicleDataTypes": {
  "type": "branch",
  "description": "Branch to aggregate all camera related higher order data types",
  "children": {
   "SVMCamera": {
    "type": "struct",
    "description": "This data type represents Surround View Monitor (SVM) camera
system in a vehicle",
    "comment": "Test comment",
    "deprecation": "Test deprecation message",
    "children": {
     "Make": {
      "type": "property",
      "description": "Make of the SVM camera",
      "datatype": "string",
      "comment": "Test comment",
```

```
        "deprecation": "Test deprecation message"
      },
      "Description": {
        "type": "property",
        "description": "Description of the SVM camera",
        "datatype": "string",
        "comment": "Test comment",
        "deprecation": "Test deprecation message"
      },
      "FPS": {
        "type": "property",
        "description": "FPS of the SVM camera",
        "datatype": "double",
        "comment": "Test comment",
        "deprecation": "Test deprecation message"
      },
      "Orientation": {
        "type": "property",
        "description": "Orientation of the SVM camera",
        "datatype": "VehicleDataTypes.Orientation",
        "comment": "Test comment",
        "deprecation": "Test deprecation message"
      },
      "Range": {
        "type": "property",
        "description": "Range of the SVM camera",
        "datatype": "VehicleDataTypes.Range",
        "comment": "Test comment",
        "deprecation": "Test deprecation message"
      },
      "RawData": {
        "type": "property",
        "description": "Represents binary data of the SVM camera",
        "datatype": "uint8[]",
        "dataencoding": "binary",
        "comment": "Test comment",
        "deprecation": "Test deprecation message"
      },
      "CapturedFrames": {
        "type": "property",
        "description": "Represents selected frames captured by the SVM camera",
        "datatype": "VehicleDataTypes.Frame[]",
        "dataencoding": "typed",
        "comment": "Test comment",
```

```
       "deprecation": "Test deprecation message"
      }
     }
    },
    "Range": {
     "type": "struct",
     "description": "Range of a camera in centimeters",
     "comment": "Test comment",
     "deprecation": "Test deprecation message",
     "children": {
      "Min": {
       "type": "property",
       "description": "Minimum range of a camera in centimeters",
       "datatype": "uint32",
       "comment": "Test comment",
       "deprecation": "Test deprecation message"
      },
      "Max": {
       "type": "property",
       "description": "Maximum range of a camera in centimeters",
       "datatype": "uint32",
       "comment": "Test comment",
       "deprecation": "Test deprecation message"
      }
     }
    },
    "Orientation": {
     "type": "struct",
     "description": "Orientation of a camera",
     "comment": "Test comment",
     "deprecation": "Test deprecation message",
     "children": {
      "Front": {
       "type": "property",
       "description": "Indicates whether the camera is oriented to the front of the
vehicle",
       "datatype": "boolean",
       "comment": "Test comment",
       "deprecation": "Test deprecation message"
      },
      "Rear": {
       "type": "property",
       "description": "Indicates whether the camera is oriented to the rear of the
vehicle",
```

```
       "datatype": "boolean",
       "comment": "Test comment",
       "deprecation": "Test deprecation message"
      },
      "Side": {
       "type": "property",
       "description": "Indicates whether the camera is oriented to the side of the
 vehicle",
       "datatype": "boolean",
       "comment": "Test comment",
       "deprecation": "Test deprecation message"
      }
     }
    },
    "Frame": {
     "type": "struct",
     "description": "Represents a camera frame",
     "comment": "Test comment",
     "deprecation": "Test deprecation message",
     "children": {
      "Data": {
       "type": "property",
       "datatype": "string",
       "dataencoding": "binary",
       "comment": "Test comment",
       "deprecation": "Test deprecation message"
      }
     }
    }
   }
  }
 }

}
```

次の例は、VSS で定義されたものと同じシグナルを JSON 文字列で示しています。

```
{
   "vssJson": "{\"Vehicle\":{\"type\":\"branch\",\"children\":{\"Chassis\":{\"type
\":\"branch\",\"description\":\"All data concerning steering, suspension, wheels,
 and brakes.\",\"children\":{\"SteeringWheel\":{\"type\":\"branch\",\"description
\":\"Steering wheel signals\",\"children\":{\"Diameter\":{\"type\":\"attribute\",
\"description\":\"The diameter of the steering wheel\",\"datatype\":\"float\",\"unit
```

```
\":\"cm\",\"min\":1,\"max\":50},\"HandsOff\":{\"type\":\"branch\",\"children\":
{\"HandsOffSteeringState\":{\"type\":\"actuator\",\"description\":\"HndsOffStrWhlDtSt.
 Hands Off Steering State\",\"datatype\":\"boolean\"},\"HandsOffSteeringMode\":
{\"type\":\"actuator\",\"description\":\"HndsOffStrWhlDtMd. Hands Off Steering Mode
\",\"datatype\":\"int8\",\"min\":0,\"max\":2}}}},\"Accelerator\":{\"type\":\"branch
\",\"description\":\"\",\"children\":{\"AcceleratorPedalPosition\":{\"type\":\"sensor
\",\"description\":\"Throttle__Position. Accelerator pedal position as percent. 0
 = Not depressed. 100 = Fully depressed.\",\"datatype\":\"uint8\",\"unit\":\"%\",
\"min\":0,\"max\":100.000035}}}},\"Powertrain\":{\"type\":\"branch\",\"description
\":\"Powertrain data for battery management, etc.\",\"children\":{\"Transmission\":
{\"type\":\"branch\",\"description\":\"Transmission-specific data, stopping at the
 drive shafts.\",\"children\":{\"VehicleOdometer\":{\"type\":\"sensor\",\"description
\":\"Vehicle_Odometer\",\"datatype\":\"float\",\"unit\":\"km\",\"min\":0,\"max
\":67108863.984375}}},\"CombustionEngine\":{\"type\":\"branch\",\"description\":
\"Engine-specific data, stopping at the bell housing.\",\"children\":{\"Engine\":
{\"type\":\"branch\",\"description\":\"Engine description\",\"children\":{\"timing\":
{\"type\":\"branch\",\"description\":\"timing description\",\"children\":{\"run_time\":
{\"type\":\"sensor\",\"description\":\"Engine run time\",\"datatype\":\"int16\",\"unit
\":\"ms\",\"min\":0,\"max\":10000},\"idle_time\":{\"type\":\"sensor\",\"description
\":\"Engine idle time\",\"datatype\":\"int16\",\"min\":0,\"unit\":\"ms\",\"max
\":10000}}}}}}}},\"Axle\":{\"type\":\"branch\",\"description\":\"Axle signals\",
\"children\":{\"TireRRPrs\":{\"type\":\"sensor\",\"description\":\"TireRRPrs. Right
 rear Tire pressure in kilo-Pascal\",\"datatype\":\"float\",\"unit\":\"kPaG\",\"min
\":0,\"max\":1020}}}}}}"
}
```


> **Note**
>
> デモスクリプトをダウンロードして、ROS 2 メッセージを、シグナルカタログと互換性のある VSS JSON ファイルに変換できます。詳細については、「ビジョンシステムデータデベロッパーガイド」を参照してください。
> ビジョンシステムデータはプレビューリリースであり、変更される可能性があります。

シグナルカタログの更新 (AWS CLI)

UpdateSignalCatalogAPI オペレーションを使用して、既存のシグナルカタログを更新できます。以下の例ではを使用しています AWS CLI。

既存のシグナルカタログを更新するには、次のコマンドを実行します。

設定を含む JSON *signal-catalog-configuration*ファイルの名前に置き換えます。

```
aws iotfleetwise update-signal-catalog --cli-input-json file://signal-catalog-configuration.json
```

*signal-catalog-name*更新するシグナルカタログの名前に置き換えてください。

ブランチ、属性、センサー、アクチュエータの構成方法の詳細については、「シグナルの構成」を参照してください。

> **Important**
>
> カスタム構造はイミュータブルです。プロパティを既存のカスタム構造 (構造体) に並べ替えたり、挿入したりする必要がある場合は、その構造を削除して、目的のプロパティの順序で新しい構造を作成します。
> カスタム構造を削除するには、構造の完全修飾名を nodesToRemove に追加します。シグナルが参照している構造は、削除できません。構造を参照するシグナル (そのデータ型はターゲット構造として定義されます) は、シグナルカタログの更新リクエストの前に、更新または削除する必要があります。

```
{
     "name": "signal-catalog-name",
     "nodesToAdd": [{
       "branch": {
        "description": "Front left of vehicle specific data.",
        "fullyQualifiedName": "Vehicle.Front.Left"
       }
      },
      {
       "branch": {
        "description": "Door-specific data for the front left of vehicle.",
        "fullyQualifiedName": "Vehicle.Front.Left.Door"
       }
      },
      {
       "actuator": {
        "fullyQualifiedName": "Vehicle.Front.Left.Door.Lock",
        "description": "Whether the front left door is locked.",
        "dataType": "BOOLEAN"
       }
      },
```

```
      {
       "branch": {
        "fullyQualifiedName": "Vehicle.Camera"
       }
      },
      {
       "struct": {
        "fullyQualifiedName": "Vehicle.Camera.SVMCamera"
       }
      },
      {
       "property": {
        "fullyQualifiedName": "Vehicle.Camera.SVMCamera.ISO",
        "dataType": "STRING"
       }
      }
     ],
     "nodesToRemove": ["Vehicle.Chassis.SteeringWheel.HandsOffSteeringState"],
     "nodesToUpdate": [{
      "attribute": {
       "dataType": "FLOAT",
       "fullyQualifiedName": "Vehicle.Chassis.SteeringWheel.Diameter",
       "max": 55
      }
     }]
    }
```

シグナルカタログの削除 (AWS CLI)

DeleteSignalCatalogAPI オペレーションを使用してシグナルカタログを削除できます。以下の例ではを使用しています AWS CLI。

> **Important**
>
> シグナルカタログを削除する前に、関連付けられている車両モデル、デコーダーマニフェスト、車両、フリート、キャンペーンがないことを確認してください。手順については、以下を参照してください。
>
> - 車両モデルの削除
> - デコーダーマニフェストの削除
> - 車両の削除

- フリートの削除 (AWS CLI)
- キャンペーンの削除

既存のシグナルカタログを削除するには、次のコマンドを実行します。*signal-catalog-name*削除するシグナルカタログの名前に置き換えます。

```
aws iotfleetwise delete-signal-catalog --name signal-catalog-name
```

> **Note**
>
> このコマンドは出力を生成しません。

シグナルカタログ情報の取得 (AWS CLI)

ListSignalCatalogsAPI オペレーションを使用して、シグナルカタログが削除されたかどうかを確認できます。以下の例ではを使用しています AWS CLI。

すべてのシグナルカタログの概要をページ分割されたリストとして取得するには、次のコマンドを実行します。

```
aws iotfleetwise list-signal-catalogs
```

ListSignalCatalogNodesAPI オペレーションを使用して、シグナルカタログが更新されたかどうかを確認できます。以下の例ではを使用しています AWS CLI。

特定のシグナルカタログに含まれているすべてのシグナル (ノード) の概要をページ分割されたリストとして取得するには、次のコマンドを実行します。

*signal-catalog-name*チェックするシグナルカタログの名前に置き換えてください。

```
aws iotfleetwise list-signal-catalog-nodes --name signal-catalog-name
```

GetSignalCatalogAPI オペレーションを使用してシグナルカタログ情報を取得できます。以下の例ではを使用しています AWS CLI。

シグナルカタログに関する情報を取得するには、次のコマンドを実行します。

*signal-catalog-name*取得したいシグナルカタログの名前に置き換えます。

```
aws iotfleetwise get-signal-catalog --name signal-catalog-name
```

> **Note**
>
> このオペレーションは結果整合性があります。言い換えると、シグナルカタログへの変更はすぐには反映されない場合があります。

車両モデルの作成と管理

車両の形式を標準化するために役立つ車両モデルを作成するには、シグナルを使用します。車両モデルにより、同じタイプの複数の車両に一貫した情報が適用され、車両のフリートからのデータを処理できるようになります。同じ車両モデルから作成された車両は、同じシグナルのグループを継承します。詳細については、「車両の作成、プロビジョニング、管理」を参照してください。

各車両モデルには、車両モデルの状態を含むステータスフィールドがあります。ステータスは以下のいずれかの値になります。

- ACTIVE - 車両モデルはアクティブです。
- DRAFT - 車両モデルの構成が保存されています。

> **Important**
>
> - CreateModelManifest API オペレーションを使用して最初の車両モデルを作成する場合は、まずシグナルカタログを作成する必要があります。詳細については、「シグナルカタログの作成 (AWS CLI)」を参照してください。
> - AWS IoT FleetWise コンソールを使用して車両モデルを作成すると、 AWS IoT FleetWise が自動的に車両モデルを起動します。
> - CreateModelManifest API オペレーションを使用して車両モデルを作成する場合は、車両モデルは DRAFT 状態のままになります。
> - DRAFT 状態の車両モデルから車両を作成することはできません。車両モデルを ACTIVE 状態に変更するには、UpdateModelManifest API オペレーションを使用します。
> - ACTIVE 状態の車両モデルは編集できません。

トピック

車両モデルの作成

AWS IoT FleetWise コンソールまたは API を使用して車両モデルを作成できます。

> **Important**
>
> `CreateModelManifest` API オペレーションを使用して車両モデルを作成する場合は、事前にシグナルカタログを用意する必要があります。

トピック

車両モデルの作成 (コンソール)

AWS IoT FleetWise コンソールでは、以下の方法で車両モデルを作成できます。

- が提供するテンプレートを使用してください。 AWS
- 車両モデルの手動作成
- 車両モデルの複製

が提供するテンプレートを使用してください。 AWS

AWS IoT FleetWise には、信号カタログ、車両モデル、デコーダーマニフェストを自動的に作成する車載診断 (OBD) II、J1979 テンプレートが用意されています。このテンプレートでは、デコーダーマニフェストに OBD ネットワークインターフェイスも追加されます。詳細については、「デコーダーマニフェストの作成と管理」を参照してください。

テンプレートを使用して車両モデルを作成するには

1. AWS IoT FleetWise コンソールに移動します。
2. ナビゲーションペインで、[車両モデル] を選択します。
3. [車両モデル] ページで、[提供されたテンプレートを追加] を選択します。
4. [オンボードダイアグノーシス (OBD) II] を選択します。
5. AWS IoT FleetWise が作成している OBD ネットワークインターフェースの名前を入力します。
6. [追加] を選択します。

車両モデルの手動作成

シグナルカタログからシグナルを追加したり、1 つ以上の .dbc ファイルをアップロードしてシグナルをインポートしたりできます。.dbc ファイルは、コントローラーエリアネットワーク (CAN バス) データベースでサポートされるファイル形式です。

> **Important**
>
> AWS IoT FleetWise コンソールを使用してビジョンシステムのデータ信号を含む車両モデルを作成することはできません。代わりに、 AWS CLI を使用して車両モデルを作成してください。
> ビジョンシステムデータはプレビューリリースであり、変更される可能性があります。

車両モデルを手動で作成するには

1. AWS IoT FleetWise コンソールに移動します。
2. ナビゲーションペインで、[車両モデル] を選択します。
3. [車両モデル] ページで、[車両モデルを作成] を選択し、次の操作を行います。

トピック

- ステップ 1: 車両モデルを構成する
- ステップ 2: シグナルを追加する
- ステップ 3: シグナルをインポートする
- (オプション) ステップ 4: 属性を追加する
- ステップ 5: 確認して作成する

ステップ 1: 車両モデルを構成する

[一般的な情報] セクションで、次の操作を行います。

1. 車両モデルの名前を入力します。
2. (オプション) 説明を入力します。
3. [次へ] を選択します。

ステップ 2: シグナルを追加する

> **Note**
>
> - AWS IoT を初めて使用する場合 FleetWise、このステップはシグナルカタログが作成されるまで実行できません。最初の車両モデルが作成されると、 AWS IoT FleetWise は最初の車両モデルに追加された信号を含む信号カタログを自動的に作成します。
> - AWS IoT の経験があれば FleetWise、信号カタログから信号を選択するか、.dbc ファイルをアップロードして信号をインポートすることで、車両モデルに信号を追加できます。
> - 車両モデルを作成するには、1 つ以上のシグナルが必要です。

シグナルを追加するには

1. シグナルカタログから、車両モデルに追加するシグナルを 1 つ以上選択します。選択したシグナルは右側のペインで確認できます。

   > **Note**
   >
   > 選択したシグナルだけが車両モデルに追加されます。

2. [次へ] を選択します。

ステップ 3: シグナルをインポートする

> **Note**
>
> - AWS IoT を初めて使用する場合は FleetWise、信号をインポートするために少なくとも 1 つの.dbc ファイルをアップロードする必要があります。

- AWS IoT の経験があれば FleetWise、信号カタログから信号を選択するか、.dbc ファイルをアップロードして信号をインポートすることで、車両モデルに信号を追加できます。
- 車両モデルを作成するには、1 つ以上のシグナルが必要です。

シグナルをインポートするには

1. [ファイルを選択] を選択します。
2. ダイアログボックスで、シグナルを含む .dbc ファイルを選択します。複数の .dbc ファイルをアップロードできます。
3. AWS IoT は.dbc FleetWise ファイルを解析して信号を取得します。

   [シグナル] セクションで、シグナルごとに以下のメタデータを指定します。

   - [名前] - シグナルの名前。

     シグナル名は一意である必要があります。シグナルの名前とパスには、合わせて最大 150 文字を入力できます。有効な文字は、a ~ z、A ~ Z、0 ~ 9、: (コロン)、_ (アンダースコア) です。

   - [データ型] - シグナルのデータ型は、INT8、UINT8、INT16、UINT16、INT32、UINT32、INT64、UINT64、BOOLEAN、FLOAT、DO のいずれかである必要があります。
   - [シグナルタイプ] - シグナルのタイプ。[センサー] または [アクチュエータ]. を指定できます。
   - (オプション) [単位] - シグナルの科学単位 (km、摂氏など)。
   - (オプション) [パス] - シグナルのパス。JSONPath と同様に、子シグナルを参照するにはドット (.) を使用します。例えば、**`Vehicle.Engine.Light`** のように指定します。

     シグナルの名前とパスには、合わせて最大 150 文字を入力できます。有効な文字は、a ~ z、A ~ Z、0 ~ 9、: (コロン)、_ (アンダースコア) です。

   - (オプション) [最小値] - シグナルの最小値。
   - (オプション) [最大値] - シグナルの最大値。
   - (オプション) [説明] - シグナルの説明。

     説明には最大 2048 文字を入力できます。有効な文字は、a ~ z、A ~ Z、0 ~ 9、: (コロン)、_ (アンダースコア)、- (ハイフン) です。

4. [次へ] を選択します。

(オプション) ステップ 4: 属性を追加する

シグナルカタログには、既存の属性と合わせて最大 100 個の属性を追加できます。

属性を追加するには

1. [属性を追加] で、属性ごとに以下のメタデータを指定します。

   - [名前] - 属性の名前。

     シグナル名は一意である必要があります。シグナルの名前とパスには、合わせて最大 150 文字を入力できます。有効な文字は、a ~ z、A ~ Z、0 ~ 9、: (コロン)、_ (アンダースコア) です。

   - [データ型] - 属性のデータ型は、INT8、UINT8、INT16、UINT16、INT32、UINT32、INT64、UINT64、BOOLEAN、FLOAT、DO のいずれかである必要があります。
   - (オプション) [単位] - 属性の科学単位 (km、摂氏など)。
   - (オプション) [パス] - シグナルのパス。JSONPath と同様に、子シグナルを参照するにはドット (.) を使用します。例えば、**`Vehicle.Engine.Light`** のように指定します。

     シグナルの名前とパスには、合わせて最大 150 文字を入力できます。有効な文字は、a ~ z、A ~ Z、0 ~ 9、: (コロン)、_ (アンダースコア) です。

   - (オプション) [最小値] - 属性の最小値。
   - (オプション) [最大値] - 属性の最大値。
   - (オプション) [説明] - 属性の説明。

     説明には最大 2048 文字を入力できます。有効な文字は、a ~ z、A ~ Z、0 ~ 9、: (コロン)、_ (アンダースコア)、- (ハイフン) です。

2. [次へ] を選択します。

ステップ 5: 確認して作成する

車両モデルの構成を確認し、[作成] を選択します。

車両モデルの複製

AWS IoT FleetWise は既存の車両モデルの構成をコピーして新しいモデルを作成できます。選択した車両モデルで指定されているシグナルが、新しい車両モデルにコピーされます。

車両モデルを複製するには

1. AWS IoT FleetWise コンソールに移動します。
2. ナビゲーションペインで、[車両モデル] を選択します。
3. 車両モデルの一覧からモデルを選択し、[モデルの複製] を選択します。

車両モデルを構成するには、「車両モデルの手動作成」チュートリアルに従います。

FleetWise 車両モデルを作成するリクエストを AWS IoT が処理するまでに数分かかることがあります。車両モデルが正常に作成されると、[車両モデル] ページの [ステータス] 列に [ACTIVE] と表示されます。車両モデルがアクティブになると、編集することはできなくなります。

車両モデルの作成 (AWS CLI)

CreateModelManifestAPI オペレーションを使用して車両モデル (モデルマニフェスト) を作成できます。次の例では AWS CLIを使用しています。

> **Important**
>
> AWS IoT FleetWise API を使用して最初の車両モデルを作成する場合は、まず信号カタログを作成する必要があります。シグナルカタログの作成方法の詳細については、「シグナルカタログの作成 (AWS CLI)」を参照してください。

車両モデルを作成するには、次のコマンドを実行します。

構成を含む JSON *vehicle-model-configuration*ファイルの名前に置き換えます。

```
aws iotfleetwise create-model-manifest --cli-input-json file://vehicle-model-
configuration.json
```

- *vehicle-model-name*作成する車両モデルの名前に置き換えてください。
- *signal-catalog-ARN* は、シグナルカタログの Amazon リソースネーム (ARN) に置き換えます。
- (オプション) *description* は、車両モデルの識別に役立つ説明に置き換えます。

ブランチ、属性、センサー、アクチュエータの構成方法の詳細については、「シグナルの構成」を参照してください。

```
{
    "name": "vehicle-model-name",
    "signalCatalogArn": "signal-catalog-ARN",
    "description": "description",
    "nodes": ["Vehicle.Chassis"]
}
```

車両モデルの更新 (AWS CLI)

UpdateModelManifestAPI オペレーションを使用して、既存の車両モデル (モデルマニフェスト) を更新できます。次の例では AWS CLIを使用しています。

既存の車両モデルを更新するには、次のコマンドを実行します。

構成を含む JSON *update-vehicle-model-configuration*ファイルの名前に置き換えます。

```
aws iotfleetwise update-model-manifest --cli-input-json file://update-vehicle-model-configuration.json
```

- *vehicle-model-name*更新する車両モデルの名前に置き換えてください。
- (オプション) 車両モデルを有効化するには、*vehicle-model-status*に置き換えますACTIVE。

> **Important**
>
> 車両モデルがアクティブになると、その車両モデルは変更できなくなります。

- (オプション) *description* は、車両モデルの識別に役立つ、更新された説明に置き換えます。

```
{
    "name": "vehicle-model-name",
    "status": "vehicle-model-status",
    "description": "description",
    "nodesToAdd": ["Vehicle.Front.Left"],
    "nodesToRemove": ["Vehicle.Chassis.SteeringWheel"],
}
```

車両モデルの削除

AWS IoT FleetWise コンソールまたは API を使用して車両モデルを削除できます。

> **Important**
>
> まず、車両モデルに関連付けられている車両とデコーダーマニフェストを削除する必要があります。詳細については、車両の削除およびデコーダーマニフェストの削除を参照してください。

車両モデルの削除 (コンソール)

車両モデルを削除するには、 AWS IoT FleetWise コンソールを使用します。

車両モデルを削除するには

1. AWS IoT FleetWise コンソールに移動します。
2. ナビゲーションペインで、[車両モデル] を選択します。
3. [車両モデル] ページで、ターゲットの車両モデルを選択します。
4. [削除] を選択します。
5. [**vehicle-model-name** を削除しますか?] で、削除する車両モデルの名前を入力し、[確認] を選択します。

車両モデルの削除 (AWS CLI)

DeleteModelManifestAPI オペレーションを使用して、既存の車両モデル (モデルマニフェスト) を削除できます。次の例では AWS CLIを使用しています。

車両モデルを削除するには、次のコマンドを実行します。

*model-manifest-name*削除する車両モデルの名前に置き換えます。

```
aws iotfleetwise delete-model-manifest --name model-manifest-name
```

> **Note**
>
> このコマンドは出力を生成しません。

車両モデル情報の取得 (AWS CLI)

ListModelManifestsAPI オペレーションを使用して、車両モデルが削除されたかどうかを確認できます。以下の例ではを使用しています AWS CLI。

すべての車両モデルの概要をページ分割されたリストとして取得するには、次のコマンドを実行します。

```
aws iotfleetwise list-model-manifests
```

ListModelManifestNodesAPI オペレーションを使用して、車両モデルが更新されたかどうかを確認できます。以下の例ではを使用しています AWS CLI。

特定の車両モデルに含まれているすべてのシグナル (ノード) の概要をページ分割されたリストとして取得するには、次のコマンドを実行します。

*vehicle-model-name*確認する車両モデルの名前に置き換えます。

```
aws iotfleetwise list-model-manifest-nodes /
                --name vehicle-model-name
```

車両モデルに関する情報を取得するには、次のコマンドを実行します。

vehicle-model は、取得する車両モデルの名前に置き換えます。

```
aws iotfleetwise get-model-manifest --name vehicle-model
```

> **Note**
>
> このオペレーションは結果整合性があります。言い換えると、車両モデルへの変更はすぐには反映されない場合があります。

デコーダーマニフェストの作成と管理

デコーダーマニフェストには、 AWS IoT が車両データ (バイナリデータ) を人間が読める値に変換したり、 FleetWise データ分析用にデータを準備したりするために使用するデコード情報が含まれています。デコーダーマニフェストの構成に使用するコアコンポーネントが、ネットワークインターフェイスとデコーダーシグナルです。

ネットワークインターフェイス

車載ネットワークが使用するプロトコルに関する情報が含まれています。 AWS IoT FleetWise は以下のプロトコルをサポートしています。

コントローラーエリアネットワーク (CAN バス)

電子制御ユニット (ECU) 間でのデータの通信方法を定義するプロトコル。ECU には、エンジンコントロールユニット、エアバッグ、オーディオシステムなどがあります。

オンボードダイアグノーシス (OBD) II

ECU 間の自己診断データの通信方法を定義する、より進化したプロトコル。車両の問題を特定するために役立つ標準の故障診断コード (DTC) が多数定義されています。

車両ミドルウェア

車両ミドルウェアは、ネットワークインターフェイスの一種として定義します。車両ミドルウェアの例としては、ロボットオペレーティングシステム (ROS 2) や Scalable service-Oriented MiddlewarE over IP (SOME/IP) が挙げられます。

> **Note**
>
> AWS IoT FleetWise は、ビジョンシステムデータ用の ROS 2 ミドルウェアをサポートしています。

デコーダーシグナル

特定のシグナルについて詳細なデコード情報を提供します。車両モデルに指定されたすべてのシグナルは、デコーダーシグナルとペアになっている必要があります。デコーダーマニフェストに CAN ネットワークインターフェイスが含まれている場合は、CAN デコーダーシグナルも含まれている必要があります。デコーダーマニフェストに OBD ネットワークインターフェイスが含まれている場合は、OBD デコーダーシグナルも含まれている必要があります。

デコーダーマニフェストに車両ミドルウェアインターフェイスも含まれている場合は、メッセージデコーダーシグナルも含める必要があります。

各デコーダーマニフェストは車両モデルに関連付けられている必要があります。 AWS IoT FleetWise は、関連するデコーダーマニフェストを使用して、車両モデルに基づいて作成された車両からのデータをデコードします。

各デコーダーマニフェストには、デコーダーマニフェストの状態を含むステータスフィールドがあります。ステータスは以下のいずれかの値になります。

- ACTIVE - デコーダーマニフェストはアクティブです。
- DRAFT - デコーダーマニフェストの構成は保存されていません。
- VALIDATING — デコーダーマニフェストの適格性が検証中です。これは、少なくとも 1 つのビジョンシステムデータシグナルを含むデコーダーマニフェストにのみ適用されます。
- INVALID — デコーダーマニフェストが検証に失敗したため、まだアクティブ化できません。これは、少なくとも 1 つのビジョンシステムデータシグナルを含むデコーダーマニフェストにのみ適用されます。 ListDecoderManifests と GetDecoderManifest API を使用して、検証に失敗した理由を確認できます。

> **Important**
>
> - AWS IoT FleetWise コンソールを使用してデコーダーマニフェストを作成すると、 AWS IoT FleetWise は自動的にデコーダーマニフェストをアクティベートします。
> - CreateDecoderManifest API オペレーションを使用してデコーダーマニフェストを作成する場合は、デコーダーマニフェストは DRAFT 状態のままになります。
> - DRAFT のデコーダーマニフェストに関連付けられている車両モデルから車両を作成することはできません。デコーダーマニフェストを ACTIVE 状態に変更するには、UpdateDecoderManifest API オペレーションを使用します。
> - ACTIVE 状態のデコーダーマニフェストは編集できません。

トピック

ネットワークインターフェイスとデコーダーシグナルの構成

すべてのデコーダーマニフェストには、少なくとも 1 つのネットワークインターフェイスと、関連付けられた車両モデルで指定されているシグナルとペアになるデコーダーシグナルが含まれています。

デコーダーマニフェストに CAN ネットワークインターフェイスが含まれている場合は、CAN デコーダーシグナルも含まれている必要があります。デコーダーマニフェストに OBD ネットワークインターフェイスが含まれている場合は、OBD デコーダーシグナルも含まれている必要があります。

トピック

ネットワークインターフェイスの構成

CAN ネットワークインターフェイスを構成するには、以下の情報を指定します。

- `name` - CAN インターフェイスの名前。

  インターフェイス名は一意でなければならず、1 ~ 100 文字を使用できます。
- (オプション) `protocolName` - プロトコルの名前。

  有効な値: `CAN-FD` および `CAN`
- (オプション) `protocolVersion` — AWS IoT FleetWise は現在 CAN-FD と CAN 2.0b をサポートしています。

  有効な値: `1.0` および `2.0b`

OBD ネットワークインターフェイスを構成するには、以下の情報を指定します。

- `name` - OBD インターフェイスの名前。

  インターフェイス名は一意でなければならず、1 ~ 100 文字を使用できます。
- `requestMessageId` - 車両データをリクエストするメッセージの ID。
- (オプション) `dtcRequestIntervalSeconds` - 車両に故障診断コード (DTC) をリクエストする頻度 (秒単位)。例えば、指定した値が 120 の場合、エッジエージェントソフトウェアは保存された DTC を 2 分に 1 回収集します。

- (オプション) hasTransmissionEcu - 車両にトランスミッションコントロールモジュール (TCM) が搭載されているかどうか。

  有効な値: true および false

- (オプション) obdStandard — AWS IoT FleetWise がサポートする OBD 規格。 AWS FleetWise IoTは現在、ワールド・ワイド・ハーモナイゼーション・オンボード診断 (WWH-OBD) ISO15765-4規格をサポートしています。
- (オプション) pidRequestIntervalSeconds - 車両に OBD II PID をリクエストする頻度。例えば、指定した値が 120 の場合、エッジエージェントソフトウェアは OBD II PID を 2 分に 1 回収集します。
- (オプション) useExtendedIds - メッセージに拡張 ID を使用するかどうか。

  有効な値: true および false

車両ミドルウェアネットワークインターフェイスを設定するには、以下の情報を指定します。

- name — 車両ミドルウェアインターフェイスの名前。

  インターフェイス名は一意でなければならず、1 ~ 100 文字を使用できます。

- protocolName — プロトコル名。

  有効な値: ROS_2

デコーダーシグナルの構成

CAN デコーダーシグナルを構成するには、以下の情報を指定します。

- factor - メッセージのデコードに使用される乗数。
- isBigEndian - メッセージのバイト順がビッグエンディアンかどうか。ビッグエンディアンの場合、シーケンス内の最上位の値が最初 (最も低いストレージアドレス) に格納されます。
- isSigned - メッセージが符号付きかどうか。符号付きの場合、メッセージは正の数値と負数の両方を表すことができます。
- length - メッセージの長さ (バイト単位)。
- messageId - メッセージの ID。
- offset - シグナル値の計算に使用されるオフセット。factor と組み合わせて、計算は value = raw_value * factor + offset となります。

- `startBit` - メッセージの先頭ビットの位置を示します。
- (オプション) `name` - シグナルの名前。

OBD デコーダーシグナルを構成するには、以下の情報を指定します。

- `byteLength` - メッセージの長さ (バイト単位)。
- `offset` - シグナル値の計算に使用されるオフセット。scaling と組み合わせて、計算は `value = raw_value * scaling + offset` となります。
- `pid` - このシグナルについて車両にメッセージをリクエストする際に使用する診断コード。
- `pidResponseLength` - リクエストされたメッセージの長さ。
- `scaling` - メッセージのデコードに使用される乗数。
- `serviceMode` - メッセージ内のオペレーション (診断サービス) のモード。
- `startByte` - メッセージの先頭を示します。
- (オプション) `bitMaskLength` - メッセージ内のマスクされたビット数。
- (オプション) `bitRightShift` - 右にシフトされた位置の数。

メッセージデコーダーシグナルを設定するには、以下の情報を指定します。

- `topicName` — メッセージシグナルのトピック名。この名前は、ROS 2 のトピックに対応しています。構造化メッセージオブジェクトの詳細については、を参照してください。StructuredMessage
- `structuredMessage` — メッセージシグナルの構造化メッセージ。定義を使用して定義することも primitiveMessageDefinition、 structuredMessageList structuredMessageDefinition 再帰的に定義することもできます。

デコーダーマニフェストの作成

AWS IoT FleetWise コンソールまたは API を使用して、車両モデルのデコーダーマニフェストを作成できます。

> **Important**
>
> デコーダーマニフェストを作成する前に車両モデルが必要です。すべてのデコーダーマニフェストは、車両モデルに関連付ける必要があります。詳細については、「車両モデルの作成と管理」を参照してください。

トピック

- デコーダーマニフェストの作成 (コンソール)
- デコーダーマニフェストの作成 (AWS CLI)

デコーダーマニフェストの作成 (コンソール)

AWS IoT FleetWise コンソールを使用して、車両モデルに関連するデコーダーマニフェストを作成できます。

> **Important**
>
> AWS IoT FleetWise コンソールを使用してデコーダーマニフェストにビジョンシステムデータ信号を設定することはできません。代わりに、 AWS CLIを使用してください。ビジョンシステムデータはプレビューリリースであり、変更される可能性があります。

デコーダーマニフェストを作成するには

1. AWS IoT FleetWise コンソールに移動します。
2. ナビゲーションペインで、[車両モデル] を選択します。
3. ターゲットの車両モデルを選択します。
4. 車両モデルの概要ページで、[デコーダーマニフェストを作成] を選択し、次の操作を行います。

トピック

- ステップ 1: デコーダーマニフェストを構成する
- ステップ 2: ネットワークインターフェイスを追加する
- ステップ 3: 確認して作成する

ステップ 1: デコーダーマニフェストを構成する

[一般的な情報] セクションで、次の操作を行います。

1. デコーダーマニフェストの一意の名前を入力します。
2. (オプション) 説明を入力します。
3. [次へ] を選択します。

ステップ 2: ネットワークインターフェイスを追加する

各デコーダーマニフェストには、少なくとも 1 つのネットワークインターフェイスが必要です。複数のネットワークインターフェイスをデコーダーマニフェストに追加できます。

ネットワークインターフェイスを作成するには

- [ネットワークインターフェイス] で、次の操作を行います。
  a. [ネットワークインターフェイスのタイプ] で、[CAN_INTERFACE] または [OBD_INTERFACE] を選択します。
  b. ネットワークインターフェイスの一意の名前を入力します。
  c. 一意のネットワークインターフェイス ID を入力します。 AWS IoT が生成した ID を使用できます FleetWise。
  d. 車両モデルに指定されている 1 つ以上のシグナルを選択して、デコーダーシグナルとペアリングします。
  e. デコード情報を提供するには、.dbc ファイルをアップロードします。 AWS IoT は.dbc FleetWise ファイルを解析してデコーダ信号を取得します。
  f. [ペアリングされているシグナル] セクションで、すべてのシグナルがデコーダーシグナルとペアになっていることを確認します。
  g. [次へ] を選択します。

> **Note**
>
> - 各ネットワークインターフェイスにアップロードできる .dbc ファイルは 1 つだけです。
> - 車両モデルに指定されているすべてのシグナルが、デコーダーシグナルとペアになっていることを確認してください。

- 別のネットワークインターフェイスを追加すると、編集中のネットワークインターフェイスは編集できなくなります。既存のネットワークインターフェイスは削除することができます。

ステップ 3: 確認して作成する

デコーダーマニフェストの構成を確認し、[作成] を選択します。

デコーダーマニフェストの作成 (AWS CLI)

CreateDecoderManifestAPI オペレーションを使用してデコーダマニフェストを作成できます。次の例では AWS CLIを使用しています。

> **Important**
> デコーダーマニフェストを作成する前に、まず車両モデルを作成してください。詳細については、「車両モデルの作成」を参照してください。

デコーダーマニフェストを作成するには、次のコマンドを実行します。

設定を含む JSON *decoder-manifest-configuration*ファイルの名前に置き換えます。

```
aws iotfleetwise create-decoder-manifest --cli-input-json file://decoder-manifest-configuration.json
```

- *decoder-manifest-name*作成するデコーダーマニフェストの名前に置き換えます。
- *vehicle-model-ARN* は、車両モデルの Amazon リソースネーム (ARN) に置き換えます。
- (オプション) *description* は、デコーダーマニフェストの識別に役立つ説明に置き換えます。

ブランチ、属性、センサー、アクチュエータの構成方法の詳細については、「ネットワークインターフェイスとデコーダーシグナルの構成」を参照してください。

```
{
    "name": "decoder-manifest-name",
    "modelManifestArn": "vehicle-model-arn",
    "description": "description",
    "networkInterfaces": [
```

```
        {
            "canInterface": {
                "name": "myNetworkInterface",
                "protocolName": "CAN",
                "protocolVersion": "2.0b"
            },
            "interfaceId": "Qq1acaenByOB3sSM39SYm",
            "type": "CAN_INTERFACE"
        }
    ],
    "signalDecoders": [
        {
            "canSignal": {
                "name": "Engine_Idle_Time",
                "factor": 1,
                "isBigEndian": true,
                "isSigned": false,
                "length": 24,
                "messageId": 271343712,
                "offset": 0,
                "startBit": 16
            },
            "fullyQualifiedName": "Vehicle.EngineIdleTime",
            "interfaceId": "Qq1acaenByOB3sSM39SYm",
            "type": "CAN_SIGNAL"
        },
        {
            "canSignal": {
                "name": "Engine_Run_Time",
                "factor": 1,
                "isBigEndian": true,
                "isSigned": false,
                "length": 24,
                "messageId": 271343712,
                "offset": 0,
                "startBit": 40
            },
            "fullyQualifiedName": "Vehicle.EngineRunTime",
            "interfaceId": "Qq1acaenByOB3sSM39SYm",
            "type": "CAN_SIGNAL"
        }
    ]
}
```

- *decoder-manifest-name*作成するデコーダーマニフェストの名前に置き換えてください。
- *vehicle-model-ARN* は、車両モデルの Amazon リソースネーム (ARN) に置き換えます。
- (オプション) *description* は、デコーダーマニフェストの識別に役立つ説明に置き換えます。

構造 (構造体) 内のプロパティノードの順序は、シグナルカタログと車両モデル (モデルマニフェスト) で定義されている順序と一致している必要があります。ブランチ、属性、センサー、アクチュエータの構成方法の詳細については、「ネットワークインターフェイスとデコーダーシグナルの構成」を参照してください。

```
{
 "name": "decoder-manifest-name",
 "modelManifestArn": "vehicle-model-arn",
 "description": "description",
 "networkInterfaces": [{
  "canInterface": {
   "name": "myNetworkInterface",
   "protocolName": "CAN",
   "protocolVersion": "2.0b"
  },
  "interfaceId": "Qq1acaenByOB3sSM39SYm",
  "type": "CAN_INTERFACE"
 }, {
  "type": "VEHICLE_MIDDLEWARE",
  "interfaceId": "G1KzxkdnmV5Hn7wkV3ZL9",
  "vehicleMiddleware": {
   "name": "ROS2_test",
   "protocolName": "ROS_2"
  }
 }],
 "signalDecoders": [{
   "canSignal": {
    "name": "Engine_Idle_Time",
    "factor": 1,
    "isBigEndian": true,
    "isSigned": false,
    "length": 24,
    "messageId": 271343712,
    "offset": 0,
    "startBit": 16
   },
   "fullyQualifiedName": "Vehicle.EngineIdleTime",
```

```
   "interfaceId": "Qq1acaenByOB3sSM39SYm",
   "type": "CAN_SIGNAL"
  },
  {
   "canSignal": {
    "name": "Engine_Run_Time",
    "factor": 1,
    "isBigEndian": true,
    "isSigned": false,
    "length": 24,
    "messageId": 271343712,
    "offset": 0,
    "startBit": 40
   },
   "fullyQualifiedName": "Vehicle.EngineRunTime",
   "interfaceId": "Qq1acaenByOB3sSM39SYm",
   "type": "CAN_SIGNAL"
  },
  {
   "fullyQualifiedName": "Vehicle.CompressedImageTopic",
   "type": "MESSAGE_SIGNAL",
   "interfaceId": "G1KzxkdnmV5Hn7wkV3ZL9",
   "messageSignal": {
    "topicName": "CompressedImageTopic:sensor_msgs/msg/CompressedImage",
    "structuredMessage": {
     "structuredMessageDefinition": [{
       "fieldName": "header",
       "dataType": {
        "structuredMessageDefinition": [{
          "fieldName": "stamp",
          "dataType": {
           "structuredMessageDefinition": [{
             "fieldName": "sec",
             "dataType": {
              "primitiveMessageDefinition": {
               "ros2PrimitiveMessageDefinition": {
                "primitiveType": "INT32"
               }
              }
             }
            },
            {
             "fieldName": "nanosec",
             "dataType": {
```

```
            "primitiveMessageDefinition": {
             "ros2PrimitiveMessageDefinition": {
              "primitiveType": "UINT32"
             }
            }
           }
          }
         ]
        }
       },
       {
        "fieldName": "frame_id",
        "dataType": {
         "primitiveMessageDefinition": {
          "ros2PrimitiveMessageDefinition": {
           "primitiveType": "STRING"
          }
         }
        }
       }
      ]
     }
    },
    {
     "fieldName": "format",
     "dataType": {
      "primitiveMessageDefinition": {
       "ros2PrimitiveMessageDefinition": {
        "primitiveType": "STRING"
       }
      }
     }
    },
    {
     "fieldName": "data",
     "dataType": {
      "structuredMessageListDefinition": {
       "name": "listType",
       "memberType": {
        "primitiveMessageDefinition": {
         "ros2PrimitiveMessageDefinition": {
          "primitiveType": "UINT8"
         }
        }
```

```
          },
          "capacity": 0,
          "listType": "DYNAMIC_UNBOUNDED_CAPACITY"
        }
       }
      }
     ]
    }
   }
  }
 ]
}
```

> **Note**
>
> デモスクリプトをダウンロードして、ビジョンシステムシグナルを含むデコーダーマニフェストを作成できます。詳細については、「ビジョンシステムデータデベロッパーガイド」を参照してください。
> ビジョンシステムデータはプレビューリリースであり、変更される可能性があります。

デコーダーマニフェストの更新 (AWS CLI)

UpdateDecoderManifestAPI オペレーションを使用してデコーダーマニフェストを更新できます。ネットワークインターフェイスとシグナルデコーダーの追加、削除、更新が可能です。デコーダーマニフェストのステータスを変更することもできます。次の例では AWS CLIを使用しています。

デコーダーマニフェストを更新するには、次のコマンドを実行します。

*decoder-manifest-name*更新するデコーダーマニフェストの名前に置き換えてください。

```
aws iotfleetwise update-decoder-manifest /
                --name decoder-manifest-name /
                --status ACTIVE
```

> **Important**
>
> デコーダーマニフェストをアクティブ化すると、編集することはできなくなります。

デコーダーマニフェストの削除

AWS IoT FleetWise コンソールまたは API を使用してデコーダマニフェストを削除できます。

> **Important**
>
> まず、デコーダーマニフェストに関連付けられている車両を削除する必要があります。詳細については、「車両の削除」を参照してください。

トピック

- デコーダーマニフェストの削除 (コンソール)
- デコーダーマニフェストの削除 (AWS CLI)

デコーダーマニフェストの削除 (コンソール)

AWS IoT FleetWise コンソールを使用してデコーダマニフェストを削除できます。

デコーダーマニフェストを削除するには

1. AWS IoT FleetWise コンソールに移動します。
2. ナビゲーションペインで、[車両モデル] を選択します。
3. ターゲットの車両モデルを選択します。
4. 車両モデルの概要ページで、[デコーダーマニフェスト] タブを選択します。
5. ターゲットのデコーダーマニフェストを選択し、[削除] を選択します。
6. [**decoder-manifest-name** を削除しますか?] で、削除するデコーダーマニフェストの名前を入力し、[確認]. を選択します。

デコーダーマニフェストの削除 (AWS CLI)

DeleteDecoderManifestAPI オペレーションを使用してデコーダーマニフェストを削除できます。以下の例ではを使用しています。 AWS CLI

> **Important**
>
> デコーダーマニフェストを削除する前に、まず関連付けられている車両を削除してください。詳細については、「車両の削除」を参照してください。

デコーダーマニフェストを削除するには、次のコマンドを実行します。

*decoder-manifest-name*削除するデコーダーマニフェストの名前に置き換えます。

```
aws iotfleetwise delete-decoder-manifest --name decoder-manifest-name
```

デコーダーマニフェスト情報の取得 (AWS CLI)

ListDecoderManifestsAPI オペレーションを使用して、デコーダマニフェストが削除されたかどうかを確認できます。以下の例ではを使用しています。 AWS CLI

すべてのデコーダーマニフェストの概要をページ分割されたリストとして取得するには、次のコマンドを実行します。

```
aws iotfleetwise list-decoder-manifests
```

ListDecoderManifestSignalsAPI オペレーションを使用して、デコーダーマニフェストのデコーダー信号が更新されているかどうかを確認できます。以下の例ではを使用しています。 AWS CLI

特定のデコーダーマニフェストに含まれているすべてのデコーダーシグナル (ノード) の概要をページ分割されたリストとして取得するには、次のコマンドを実行します。

*decoder-manifest-name*チェックしているデコーダーマニフェストの名前に置き換えます。

```
aws iotfleetwise list-decoder-manifest-signals /
                --name decoder-manifest-name
```

ListDecoderManifestNetworkInterfacesAPI オペレーションを使用して、デコーダマニフェストのネットワークインターフェースが更新されているかどうかを確認できます。次の例では AWS CLIを使用しています。

特定のデコーダーマニフェストに含まれているすべてのネットワークインターフェイスの概要をページ分割されたリストとして取得するには、次のコマンドを実行します。

*decoder-manifest-name*確認するデコーダーマニフェストの名前に置き換えてください。

```
aws iotfleetwise list-decoder-manifest-network-interfaces /
                --name decoder-manifest-name
```

GetDecoderManifestAPI オペレーションを使用して、デコーダーマニフェスト内のネットワークインターフェースとデコーダー信号が更新されているかどうかを確認できます。以下の例ではを使用しています。 AWS CLI

デコーダーマニフェストに関する情報を取得するには、次のコマンドを実行します。

decoder-manifest は、取得するデコーダーマニフェストの名前に置き換えます。

```
aws iotfleetwise get-decoder-manifest --name decoder-manifest
```

> **Note**
>
> このオペレーションは結果整合性があります。言い換えると、デコーダーマニフェストへの変更はすぐには反映されない場合があります。

車両の作成、プロビジョニング、管理

車両とは、車両モデルのインスタンスです。車両は車両モデルから作成し、デコーダーマニフェストに関連付ける必要があります。車両は 1 つ以上のデータストリームをクラウドにアップロードします。例えば、車両は走行距離、エンジン温度、ヒーターの状態に関するデータをクラウドに送信できます。すべての車両には、以下の情報が含まれています。

`vehicleName`

車両を識別する ID。

車両名には、個人を特定できる情報 (PII) や、その他の機密情報または重要情報を追加しないでください。車両名には、Amazon AWS など他のサービスからもアクセスできます CloudWatch。車両名でプライベートデータや機密データを使用することは想定されていません。

`modelManifestARN`

車両モデル (モデルマニフェスト) の Amazon リソースネーム (ARN)。すべての車両は車両モデルから作成されます。同じ車両モデルから作成された車両は、その車両モデルから継承された同じシグナルのグループで構成されます。これらのシグナルはシグナルカタログで定義され、標準化されます。

`decoderManifestArn`

デコーダーマニフェストの ARN。デコーダーマニフェストは、 AWS IoT が未加工の信号データ (バイナリデータ) FleetWise を人間が読める値に変換するために使用できるデコード情報を提供します。デコーダーマニフェストは車両モデルに関連付けられている必要があります。 AWS IoT FleetWise は同じデコーダーマニフェストを使用して、同じ車両モデルに基づいて作成された車両の未加工データをデコードします。

`attributes`

属性は、静的な情報を含むキーと値のペアです。車両には、車両モデルから継承された属性を含めることができます。同じ車両モデルから作成された他の車両と区別するために、個々の車両に追加の属性を設定することもできます。例えば、黒い車があるとすると、属性として `{"color": "black"}` という値を指定できます。

> **Important**
>
> 個々の車両に属性を追加できるようにするには、関連付けられている車両モデルで、それらの属性を事前に定義する必要があります。

車両モデル、デコーダーマニフェスト、属性の詳細については、「車両のモデリング」を参照してください。

AWS IoT FleetWise には、車両の作成と管理に使用できる次の API オペレーションが用意されています。

- CreateVehicle— 新しい車両を作成します。
- BatchCreateVehicle— 1 台以上の新しいビークルを作成します。
- UpdateVehicle— 既存の車両を更新します。
- BatchUpdateVehicle— 1 台以上の既存の車両を更新します。
- DeleteVehicle— 既存の車両を削除します。
- ListVehicles— 全車両の概要をページ分割したリストを取得します。
- GetVehicle— 車両に関する情報を取得します。

チュートリアル

- 車両のプロビジョニング
- 予約済みトピック
- 車両の作成
- 車両の更新 (AWS CLI)
- 複数の車両の更新 (AWS CLI)
- 車両の削除
- 車両情報の取得 (AWS CLI)

車両のプロビジョニング

車内で稼働する Edge Agent for AWS IoT FleetWise ソフトウェアは、データを収集してクラウドに転送します。 AWS IoT FleetWise はと統合され AWS IoT Core 、MQTT を介したエッジエージェントソフトウェアとクラウド間の安全な通信をサポートします。各車両は何にでも対応しています AWS IoT 。 AWS IoT 既存のモノを使って車両を作ることも、 AWS FleetWise AWS IoT IoTを設定して車両用のモノを自動で作成することもできます。詳細については、「車両の作成 (AWS CLI)」を参照してください。

AWS IoT Core AWS IoT FleetWise リソースへのアクセスを安全に制御するのに役立つ認証と承認をサポートします。車両は X.509 証明書を使用して認証 (サインイン) を受け、 AWS IoT FleetWise を

使用したり、 AWS IoT Core ポリシーを使用して特定のアクションを実行する権限 (権限を持つ) を取得したりできます。

車両の認証

AWS IoT Core 車両を認証するポリシーを作成できます。

車両を認証するには

- AWS IoT Core ポリシーを作成するには、以下のコマンドを実行します。
  - *policy-name* は、作成するポリシーの名前に置き換えます。
  - *file-name* は、ポリシーを含む JSON ファイルの名前に置き換えます。 AWS IoT Core

```
aws iot create-policy --policy-name policy-name --policy-document file://file-
name.json
```

ポリシーの例を使用する前に、次の作業を行ってください。

- ###### AWS IoT AWS FleetWise リソースを作成したリージョンに置き換えます。
- *AWSAccount AWS #########* ID に置き換えてください。

この例には、 AWS IoT が予約したトピックが含まれています FleetWise。これらのトピックをポリシーに追加する必要があります。詳細については、「予約済みトピック」を参照してください。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:region:awsAccount:client/
${iot:Connection.Thing.ThingName}"
            ]
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:region:awsAccount:topic/$aws/iotfleetwise/vehicles/
${iot:Connection.Thing.ThingName}/checkins",
                "arn:aws:iot:region:awsAccount:topic/$aws/iotfleetwise/vehicles/
${iot:Connection.Thing.ThingName}/signals"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:region:awsAccount:topicfilter/$aws/iotfleetwise/
vehicles/${iot:Connection.Thing.ThingName}/collection_schemes",
                "arn:aws:iot:region:awsAccount:topicfilter/$aws/iotfleetwise/
vehicles/${iot:Connection.Thing.ThingName}/decoder_manifests"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:region:awsAccount:topic/$aws/iotfleetwise/vehicles/
${iot:Connection.Thing.ThingName}/collection_schemes",
                "arn:aws:iot:region:awsAccount:topic/$aws/iotfleetwise/vehicles/
${iot:Connection.Thing.ThingName}/decoder_manifests"
            ]
        }
    ]
}
```

車両の認可

車両を認可する X.509 証明書を作成できます。

車両を認可するには

> **Important**
>
> 車両ごとに新しい証明書を作成することをお勧めします。

1. RSA キーペアを作成して X.509 証明書を発行するには、次のコマンドを実行します。

   - *cert* は、コマンドから出力される certificatePem の内容を保存するファイルの名前に置き換えます。
   - *public-key* を KeyPair のコマンド出力内容を保存するファイルの名前に置き換えます。PublicKey。
   - *private-key* を KeyPair のコマンド出力内容を保存するファイルの名前に置き換えます。PrivateKey。

   ```
   aws iot create-keys-and-certificate \
       --set-as-active \
       --certificate-pem-outfile cert.pem \
       --public-key-outfile public-key.key" \
       --private-key-outfile  private-key.key"
   ```

2. 出力から、証明書の Amazon リソースネーム (ARN) をコピーします。
3. 証明書にポリシーをアタッチするには、次のコマンドを実行します。

   - *policy-name* は、 AWS IoT Core 作成したポリシーの名前に置き換えてください。
   - *certificate-arn* は、コピーした証明書の ARN に置き換えます。

   ```
   aws iot attach-policy \
       --policy-name policy-name\
       --target "certificate-arn"
   ```

4. 証明書をモノにアタッチするには、次のコマンドを実行します。

   - *thing-name* は、 AWS IoT 自分のモノの名前または車両の ID に置き換えてください。
   - *certificate-arn* は、コピーした証明書の ARN に置き換えます。

```
aws iot attach-thing-principal \
    --thing-name thing-name \
    --principal "certificate-arn"
```

予約済みトピック

AWS IoT FleetWise では以下のトピックの使用を留保しています。予約済みトピックで許可されている場合は、そのトピックにサブスクライブまたは発行することができます。ただし、ドル記号 ($) で始まる新しいトピックを作成することはできません。予約済みトピックでサポートされていない発行オペレーションやサブスクライブオペレーションを使用すると、接続が終了することがあります。

トピック	許可されているクライアントオペレーション	説明	
$aws/iotfleetwise/vehicles/*vehicleName*/checkins	公開	エッジエージェントソフトウェアは、このトピックに車両のステータス情報を発行します。 車両のステータス情報はプロトコルバッファ (protobuf) 形式で交換されます。詳細については、『Edge Agent for AWS IoT FleetWise ソフトウェア開発者ガイド』を参照してください。	
$aws/iotfleetwise/vehicles/	公開	エッジエージェントソフトウェアは、このトピックにシグナルを発行します。	

トピック	許可されているクライアントオペレーション	説明	
vehicleName /signals		シグナル情報はプロトコルバッファ (protobuf) 形式で交換されます。詳細については、『Edge Agent for AWS IoT FleetWise ソフトウェア開発者ガイド』を参照してください。	
$aws/iotfleetwise/vehicles/*vehicleName*/collection_schemes	Subscribe	AWS IoT FleetWise はこのトピックにデータ収集スキームを公開しています。車両はこれらのデータ収集スキームを使用します。	
$aws/iotfleetwise/vehicles/*vehicleName*/decoder_manifests	Subscribe	AWS IoT FleetWise はこのトピックのデコーダマニフェストを公開しています。車両はこれらのデコーダーマニフェストを使用します。	

車両の作成

AWS IoT FleetWise コンソールまたは API を使用して車両を作成できます。

> **Important**
>
> 開始する前に、次の点を確認してください。

- 車両モデルが用意されていて、その車両モデルのステータスが ACTIVE になっている必要があります。詳細については、「車両モデルの作成と管理」を参照してください。
- 車両モデルがデコーダーマニフェストに関連付けられていて、そのデコーダーマニフェストのステータスが ACTIVE になっている必要があります。詳細については、「デコーダーマニフェストの作成と管理」を参照してください。

トピック

車両の作成 (コンソール)

AWS IoT FleetWise コンソールを使用して車両を作成できます。

Important

開始する前に、次の点を確認してください。

- 車両モデルが用意されていて、その車両モデルのステータスが ACTIVE になっている必要があります。詳細については、「車両モデルの作成と管理」を参照してください。
- 車両モデルがデコーダーマニフェストに関連付けられていて、そのデコーダーマニフェストのステータスが ACTIVE になっている必要があります。詳細については、「デコーダーマニフェストの作成と管理」を参照してください。

車両を作成するには

1. AWS IoT FleetWise コンソールを開きます。
2. ナビゲーションペインで、[車両] を選択します。
3. 車両の概要ページで、[車両を作成] を選択し、以下の手順を実行します。

トピック

ステップ 1: 車両のプロパティを定義する

このステップでは、車両に名前を付け、モデルマニフェストとデコーダーマニフェストに関連付けます。

1. 車両の一意の名前を入力します。

   > **Important**
   >
   > AWS IoT 車両は何にでも対応します。同じ名前のモノが既に存在する場合は、[車両を IoT モノに関連付ける] を選択すると、その車両でモノが更新されます。または、別の車両名を選択すると、 AWS FleetWise IoTが自動的にその車両用の新しいものを作成します。

2. リストから車両モデル (モデルマニフェスト) を選択します。
3. リストからデコーダーマニフェストを選択します。デコーダーマニフェストが車両モデルに関連付けられます。
4. (オプション) 車両の属性を関連付けるには、[属性を追加] を選択します。このステップを省略した場合、車両をキャンペーンにデプロイできるようにするには、車両の作成後に属性を追加する必要があります。
5. (オプション) 車両にタグを関連付けるには、[新しいタグを追加] を選択します。車両の作成後にタグを追加することもできます。
6. [次へ] を選択します。

ステップ 2: 車両証明書を構成する

車両を AWS IoT Thing として使用するには、ポリシーを添付した車両証明書を設定する必要があります。このステップを省略した場合、車両をキャンペーンにデプロイできるようにするには、車両の作成後に証明書を構成する必要があります。

1. [新しい証明書の自動生成 (推奨)] を選択します。
2. [次へ] を選択します。

ステップ 3: 証明書にポリシーをアタッチする

前のステップで構成した証明書にポリシーをアタッチします。

1. [ポリシー] に、既存のポリシー名を入力します。新しいポリシーを作成するには、[ポリシーを作成] を選択します。
2. [次へ] を選択します。

ステップ 4: 確認して作成する

車両の構成を確認し、[車両を作成] を選択します。

> **Important**
>
> 車両が作成されたら、証明書とキーをダウンロードする必要があります。証明書とプライベートキーを使用して、Edge Agent for AWS IoT FleetWise ソフトウェアで車両を接続します。

車両の作成 (AWS CLI)

車両を作成するときは、デコーダーマニフェストに関連付けられた車両モデルを使用する必要があります。CreateVehicleAPI オペレーションを使用して車両を作成できます。次の例では AWS CLIを使用しています。

> **Important**
>
> 開始する前に、次の点を確認してください。
>
> - 車両モデルが用意されていて、その車両モデルのステータスが ACTIVE になっている必要があります。詳細については、「車両モデルの作成と管理」を参照してください。
> - 車両モデルがデコーダーマニフェストに関連付けられていて、そのデコーダーマニフェストのステータスが ACTIVE になっている必要があります。詳細については、「デコーダーマニフェストの作成と管理」を参照してください。

車両を作成するには、次のコマンドを実行します。

file-name は、車両の構成を含む JSON ファイルの名前に置き換えます。

```
aws iotfleetwise create-vehicle --cli-input-json file://file-name.json
```

Example 車両の構成

- (オプション) associationBehavior の値には次のいずれかを指定できます。
  - CreateIotThing—車両が作成されると、 AWS FleetWise AWS IoT IoTは自動的に車両の車両IDの名前でモノを作成します。
  - ValidateIotThingExists - 既存の AWS IoT モノを使用して車両を作成します。

    AWS IoT モノを作成するには、以下のコマンドを実行します。*thing-name* は、作成するモノの名前に置き換えます。

    ```
    aws iot create-thing --thing-name thing-name
    ```

  指定されていない場合、 AWS FleetWise AWS IoT IoTは車両用のモノを自動的に作成します。

  > **Important**
  >
  > 車両が作成された後に AWS IoT Thing がプロビジョニングされていることを確認してください。詳細については、「車両のプロビジョニング」を参照してください。

- *vehicle-name* は、次のいずれかに置き換えます。
  - AWS IoT associationBehaviorValidateIotThingExistsモノの名前がに設定されている。
  - associationBehavior が CreateIotThing に設定されている場合は、作成する車両の ID。

    車両 ID は 1～100 文字で指定できます。有効な文字は、a～z、A～Z、0～9、ダッシュ (-)、アンダースコア (_)、コロン (:) です。
- *model-manifest-ARN* は、車両モデル (モデルマニフェスト) の ARN に置き換えます。
- *decoder-manifest-ARN* は、指定した車両モデルに関連付けられているデコーダーマニフェストの ARN に置き換えます。
- (オプション) 同じ車両モデルから作成された他の車両と区別するために、この車両に追加の属性を設定できます。例えば、電気自動車があるとすると、属性として {"fuelType": "electric"} という値を指定できます。

> **Important**
>
> 個々の車両に属性を追加できるようにするには、関連付けられている車両モデルで、それらの属性を事前に定義する必要があります。

```
{
    "associationBehavior": "associationBehavior",
    "vehicleName": "vehicle-name",
    "modelManifestArn": "model-manifest-ARN",
    "decoderManifestArn": "decoder-manifest-ARN",
    "attributes": {
        "key": "value"
    }
}
```

複数の車両の作成 (AWS CLI)

BatchCreateVehicleAPI オペレーションを使用して、複数の車両を一度に作成できます。次の例ではAWS CLIを使用しています。

複数の車両を作成するには、次のコマンドを実行します。

file-name は、複数の車両の構成を含む JSON ファイルの名前に置き換えます。

```
aws iotfleetwise batch-create-vehicle --cli-input-json file://file-name.json
```

Example 車両の構成

```
{
    "vehicles": [
        {
                "associationBehavior": "associationBehavior",
                "vehicleName": "vehicle-name",
                "modelManifestArn": "model-manifest-ARN",
                "decoderManifestArn": "decoder-manifest-ARN",
                "attributes": {
                    "key": "value"
                }
        },
```

```
        {
                "associationBehavior": "associationBehavior",
                "vehicleName": "vehicle-name",
                "modelManifestArn": "model-manifest-ARN",
                "decoderManifestArn": "decoder-manifest-ARN",
                "attributes": {
                    "key": "value"
                }
        }
    ]
}
```

1 回のバッチオペレーションで最大 10 台の車両を作成できます。車両の構成の詳細については、「車両の作成 (AWS CLI)」を参照してください。

車両の更新 (AWS CLI)

UpdateVehicleAPI オペレーションを使用して既存の車両を更新できます。次の例では AWS CLIを使用しています。

車両を更新するには、次のコマンドを実行します。

file-name は、車両の構成を含む JSON ファイルの名前に置き換えます。

```
aws iotfleetwise update-vehicle --cli-input-json file://file-name.json
```

Example 車両の構成

- *vehicle-name* は、更新する車両の ID に置き換えます。
- (オプション) *model-manifest-ARN* は、使用中の車両モデルに代えて使用する車両モデル (モデルマニフェスト) の ARN に置き換えます。
- (オプション) *decoder-manifest-ARN* は、指定した新しい車両モデルに関連付けられているデコーダーマニフェストの ARN に置き換えます。
- (オプション) *attribute-update-mode*車両属性に置き換えます。
  - `Merge` - 新しい属性を既存の属性にマージします。既存の属性は新しい値で更新され、存在しない属性は新しく追加されます。

    例えば、車両に `{"color": "black", "fuelType": "electric"}` という属性が設定されている場合、この車両を `{"color": "", "fuelType": "gasoline", "model": "x"}`

という属性で更新すると、更新後の車両の属性は {"fuelType": "gasoline", "model": "x"} になります。

- Overwrite - 既存の属性を新しい属性に置き換えます。

  例えば、車両に {"color": "black", "fuelType": "electric"} という属性が設定されている場合、この車両を {"model": "x"} という属性で更新すると、更新後の車両の属性は {"model": "x"} になります。

入力に属性が含まれている場合は必須です。

- (オプション) 新しい属性を追加したり、既存の属性を新しい値で更新したりするには、attributes を構成します。例えば、電気自動車があるとすると、属性として {"fuelType": "electric"} という値を指定できます。

  属性を削除するには、attributeUpdateMode を Merge に設定します。

> **Important**
>
> 個々の車両に属性を追加できるようにするには、関連付けられている車両モデルで、それらの属性を事前に定義する必要があります。

```
{
        "vehicleName": "vehicle-name",
        "modelManifestArn": "model-manifest-arn",
        "decoderManifestArn": "decoder-manifest-arn",
        "attributeUpdateMode": "attribute-update-mode"
        }
}
```

複数の車両の更新 (AWS CLI)

BatchUpdateVehicleAPI オペレーションを使用して、複数の既存の車両を一度に更新できます。次の例では AWS CLIを使用しています。

複数の車両を更新するには、次のコマンドを実行します。

file-name は、複数の車両の構成を含む JSON ファイルの名前に置き換えます。

```
aws iotfleetwise batch-update-vehicle --cli-input-json file://file-name.json
```

Example 車両の構成

```
{
   "vehicles": [
      {
         "vehicleName": "vehicle-name",
         "modelManifestArn": "model-manifest-arn",
         "decoderManifestArn": "decoder-manifest-arn",
         "mergeAttributes": true,
         "attributes": {
         "key": "value"
         }
      },
      {
         "vehicleName": "vehicle-name",
         "modelManifestArn": "model-manifest-arn",
         "decoderManifestArn": "decoder-manifest-arn",
         "mergeAttributes": true,
         "attributes": {
         "key": "value"
         }
      }
   ]
}
```

1 回のバッチオペレーションで最大 10 台の車両を更新できます。各車両の構成の詳細については、「車両の更新 (AWS CLI)」を参照してください。

車両の削除

AWS IoT FleetWise コンソールまたは API を使用して車両を削除できます。

> **Important**
>
> 車両が削除されると、 AWS IoT FleetWise はその車両を関連する車両群やキャンペーンから自動的に削除します。詳細については、フリートの作成と管理およびキャンペーンによるデータの収集と転送を参照してください。ただし、車両はモノとして存在するか、モノと関

連付けられたままになります。 AWS IoT Coreモノを削除する手順については、「AWS IoT Core デベロッパーガイド」の「モノの削除」を参照してください。

車両の削除 (コンソール)

AWS IoT FleetWise コンソールを使用して車両を削除できます。

車両を削除するには

1. AWS IoT FleetWise コンソールに移動します。
2. ナビゲーションペインで、[車両] を選択します。
3. [車両] ページで、削除する車両の横にあるボタンを選択します。
4. [削除] を選択します。
5. [削除]**vehicle-name** で、車両の名前を入力し、[削除] を選択します。

車両の削除 (AWS CLI)

DeleteVehicleAPI オペレーションを使用して車両を削除できます。以下の例ではを使用しています AWS CLI。

車両を削除するには、次のコマンドを実行します。

vehicle-name は、削除する車両の ID に置き換えます。

```
aws iotfleetwise delete-vehicle --vehicle-name vehicle-name
```

車両情報の取得 (AWS CLI)

ListVehiclesAPI オペレーションを使用して、車両が削除されたかどうかを確認できます。次の例では AWS CLIを使用しています。

すべての車両の概要をページ分割されたリストとして取得するには、次のコマンドを実行します。

```
aws iotfleetwise list-vehicles
```

GetVehicleAPI オペレーションを使用して車両情報を取得できます。次の例では AWS CLIを使用しています。

車両のメタデータを取得するには、次のコマンドを実行します。

vehicle-name は、取得する車両の ID に置き換えます。

```
aws iotfleetwise get-vehicle --vehicle-name vehicle-name
```

> **Note**
>
> このオペレーションは結果整合性があります。言い換えると、車両への変更はすぐには反映されない場合があります。

フリートの作成と管理

フリートは、車両のグループを表します。車両が関連付けられていないフリートは空のエンティティです。フリートを使用して複数の車両を同時に管理するには、事前に車両をフリートに関連付ける必要があります。車両は複数のフリートに所属することができます。キャンペーンをデプロイすると、車両のフリートから収集するデータと、データを収集するタイミングを制御できます。詳細については、「キャンペーンによるデータの収集と転送」を参照してください。

フリートには、以下の情報が含まれています。

`fleetId`

フリートの ID。

(オプション) `description`

フリートを見つけるために役立つ説明。

`signalCatalogArn`

シグナルカタログの Amazon リソースネーム (ARN)。

AWS IoT FleetWise には、フリートの作成と管理に使用できる以下の API オペレーションが用意されています。

- CreateFleet - 同じシグナルのグループを含む車両のグループを作成します。
- AssociateVehicleFleet - 車両をフリートに関連付けます。
- DisassociateVehicleFleet - 車両とフリートの関連付けを解除します。
- UpdateFleet - 既存のフリートの説明を更新します。
- DeleteFleet - 既存のフリートを削除します。
- ListFleets - すべてのフリートの概要をページ分割されたリストとして取得します。
- ListFleetsForVehicle - 車両が所属しているすべてのフリートの ID をページ分割されたリストとして取得します。
- ListVehiclesInFleet - フリート内のすべての車両の概要をページ分割されたリストとして取得します。
- GetFleet - フリートに関する情報を取得します。

トピック

フリートの作成 (AWS CLI)

CreateFleet API オペレーションを使用すると、フリートを作成できます。次の例では AWS CLI を使用しています。

> **Important**
>
> フリートを作成する前に、シグナルカタログを用意する必要があります。詳細については、「シグナルカタログの作成 (AWS CLI)」を参照してください。

フリートを作成するには、次のコマンドを実行します。

- *fleet-id* は、作成するフリートの ID に置き換えます。

  フリート ID は一意でなければならず、1 ~ 100 文字にする必要があります。有効な文字は、英字 (A ~ Z と a ~ z)、数字 (0 ~ 9)、コロン (:)、ダッシュ (-)、アンダースコア (_) です。
- (オプション) *description* は、説明に置き換えます。

  説明は 1 ~ 2048 文字で入力できます。
- *signal-catalog-arn* は、シグナルカタログの ARN に置き換えます。

```
aws iotfleetwise create-fleet \
   --fleet-id fleet-id \
   --description description \
   --signal-catalog-arn signal-catalog-arn
```

フリートへの車両の関連付け (AWS CLI)

AssociateVehicleFleet API オペレーションを使用すると、車両をフリートに関連付けることができます。次の例では AWS CLI を使用しています。

> **Important**
>
> - 車両をフリートに関連付ける前に、車両とフリートを用意する必要があります。詳細については、「車両の作成、プロビジョニング、管理」を参照してください。
> - キャンペーンのターゲットとなっているフリートに車両を関連付けると、AWS IoT FleetWise によってキャンペーンが自動的に車両にデプロイされます。

車両をフリートに関連付けるには、次のコマンドを実行します。

- *fleet-id* は、フリートの ID に置き換えます。
- *vehicle-name* は、車両の ID に置き換えます。

```
aws iotfleetwise associate-vehicle-fleet --fleet-id fleet-id --vehicle-name vehicle-name
```

車両とフリートの関連付けの解除 (AWS CLI)

DisassociateVehicleFleet API オペレーションを使用すると、車両とフリートの関連付けを解除できます。次の例では AWS CLI を使用しています。

車両とフリートの関連付けを解除するには、次のコマンドを実行します。

- *fleet-id* は、フリートの ID に置き換えます。
- *vehicle-name* は、車両の ID に置き換えます。

```
aws iotfleetwise disassociate-vehicle-fleet --fleet-id fleet-id --vehicle-name vehicle-name
```

フリートの更新 (AWS CLI)

UpdateFleet API オペレーションを使用すると、フリートの説明を更新できます。次の例では AWS CLI を使用しています。

フリートを更新するには、次のコマンドを実行します。

- *fleet-id* は、更新するフリートの ID に置き換えます。
- *description* は、新しい説明に置き換えます。

  説明は 1～2048 文字で入力できます。

```
aws iotfleetwise update-fleet --fleet-id fleet-id --description description
```

フリートの削除 (AWS CLI)

DeleteFleet API オペレーションを使用すると、フリートを削除できます。次の例では AWS CLI を使用しています。

> **Important**
>
> フリートを削除する前に、関連付けられている車両がないことを確認してください。車両とフリートの関連付けを解除する方法については、「車両とフリートの関連付けの解除 (AWS CLI)」を参照してください。

フリートを削除するには、次のコマンドを実行します。

fleet-id は、削除するフリートの ID に置き換えます。

```
aws iotfleetwise delete-fleet --fleet-id fleet-id
```

フリート情報の取得 (AWS CLI)

ListFleets API オペレーションを使用すると、フリートが削除されたかどうかを確認できます。次の例では AWS CLI を使用しています。

すべてのフリートの概要をページ分割されたリストとして取得するには、次のコマンドを実行します。

```
aws iotfleetwise list-fleets
```

ListFleetsForVehicle API オペレーションを使用すると、車両が所属しているすべてのフリートの ID をページ分割されたリストとして取得できます。次の例では AWS CLI を使用しています。

車両が所属しているすべてのフリートの ID をページ分割されたリストとして取得するには、次のコマンドを実行します。

vehicle-name は、車両の ID に置き換えます。

```
aws iotfleetwise list-fleets-for-vehicle \
            --vehicle-name vehicle-name
```

ListVehiclesInFleet API オペレーションを使用すると、フリート内のすべての車両の概要をページ分割されたリストとして取得できます。次の例では AWS CLI を使用しています。

フリート内のすべての車両の概要をページ分割されたリストとして取得するには、次のコマンドを実行します。

fleet-id は、フリートの ID に置き換えます。

```
aws iotfleetwise list-vehicles-in-fleet \
            --fleet-id fleet-id
```

GetFleet API オペレーションを使用すると、フリート情報を取得できます。次の例では AWS CLI を使用しています。

フリートのメタデータを取得するには、次のコマンドを実行します。

fleet-id は、フリートの ID に置き換えます。

```
aws iotfleetwise get-fleet \
            --fleet-id fleet-id
```

> **Note**
>
> このオペレーションは結果整合性があります。言い換えると、フリートへの変更はすぐには反映されない場合があります。

キャンペーンによるデータの収集と転送

キャンペーンとは、データ収集ルールのオーケストレーションです。キャンペーンにより、データをどのように選択して収集し、クラウドに転送するかに関する指示が AWS IoT FleetWise 用エッジエージェントソフトウェアに与えられます。

キャンペーンはクラウドで作成します。ユーザーまたはユーザーのチームがキャンペーンを承認すると、AWS IoT FleetWise はそれを自動的に車両にデプロイします。キャンペーンを 1 台の車両にデプロイするか、車両のフリートにデプロイするかを選択できます。エッジエージェントソフトウェアは、実施中のキャンペーンが車両にデプロイされるまでデータの収集を開始しません。

> **Note**
>
> キャンペーンは、次の条件が満たされるまで機能しません。
>
> - エッジエージェントソフトウェアが車両内で実行されている。エッジエージェントソフトウェアを開発、インストール、使用する方法の詳細を確認するには、以下の操作を行います。
>
>   1. AWS IoT FleetWise コンソールに移動します。
>   2. サービスのホームページで、[AWS IoT FleetWise の使用を開始する] セクションの [エッジエージェントを調べる] を選択します。
> - AWS IoT Core のセットアップが完了し、車両がプロビジョニングされている。詳細については、「車両のプロビジョニング」を参照してください。

各キャンペーンには、以下の情報が含まれています。

`signalCatalogArn`

キャンペーンに関連付けられているシグナルカタログの Amazon リソースネーム (ARN)。

(オプション) `tags`

タグは、キャンペーンの管理に使用できるメタデータです。さまざまなサービスのリソースに同じタグを割り当てて、それらのリソースが関連していることを示すことができます。

`TargetArn`

キャンペーンのデプロイ先となる車両またはフリートの ARN。

`name`

キャンペーンを識別するために役立つ一意の名前。

`collectionScheme`

データ収集スキームは、どのようなデータをいつ収集するかに関する指示をエッジエージェントソフトウェアに与えます。AWSIoT FleetWise では、条件ベースの収集スキームと時間ベースの収集スキームがサポートされています。

`conditionBasedCollectionScheme`

条件ベースの収集スキームでは、収集するデータを認識するために論理式が使用されます。エッジエージェントソフトウェアは、条件が満たされたときにデータを収集します。

`expression`

収集するデータを認識するために使用される論理式。例えば、`$variable.`myVehicle.InVehicleTemperature` > 50.0` という式を指定すると、エッジエージェントソフトウェアは 50.0 より大きい温度値を収集します。式の書き方の手順については、「キャンペーンの論理式」を参照してください。

(オプション) `triggerMode` には次のいずれかの値を指定できます。

- `RISING_EDGE` - エッジエージェントソフトウェアは、条件が初めて満たされたときにのみデータを収集します。例えば、`$variable.`myVehicle.AirBagDeployed` == true` です。
- `ALWAYS` - エッジエージェントソフトウェアは、条件が満たされるたびにデータを収集します。

(オプション) `minimumTriggerIntervalMs`

2 つのデータ収集イベント間の最小時間 (ミリ秒単位)。シグナルが頻繁に変化する場合は、データの収集速度を遅くすることができます。

(オプション) `conditionLanguageVersion`

条件式言語のバージョン。

`timeBasedCollectionScheme`

時間ベースの収集スキームを定義するときは、時間間隔をミリ秒単位で指定します。エッジエージェントソフトウェアは、その時間間隔を使用してデータの収集頻度を決定します。例えば、時間間隔が 120,000 ミリ秒の場合、エッジエージェントソフトウェアはデータを 2 分ごとに 1 回収集します。

(オプション) `compression`

ワイヤレス帯域幅を節約し、ネットワークトラフィックを減らすために、SNAPPY を指定して車両内のデータを圧縮できます。

デフォルト (OFF) では、エッジエージェントソフトウェアはデータを圧縮しません。

`dataDestinationConfigs`

キャンペーンで車両データを転送する宛先を選択します。Amazon S3 または Amazon Timestream にデータを保存できます。

S3 は、耐久性の高いデータ管理機能とダウンストリームデータサービスを提供する、コスト効率に優れたデータストレージメカニズムです。S3 は、運転動作や長期メンテナンスの分析に関連するデータに使用できます。

Timestream は、傾向やパターンをほぼリアルタイムで特定するために役立つデータ永続化メカニズムです。Timestream は、車両の速度やブレーキの履歴の傾向を分析する場合など、時系列データに使用できます。

(オプション) `dataExtraDimensions`

1 つ以上の属性を追加して、シグナルに追加情報を提供できます。

(オプション) `description`

キャンペーンの目的を識別するために役立つ説明を追加できます。

(オプション) `diagnosticsMode`

診断モードを `SEND_ACTIVE_DTCS` に設定すると、キャンペーンは、保存された標準の故障診断コード (DTC) を送信するようになります。これは車両の問題を特定するために役立ちます。例えば、P0097 は、エンジンコントロールモジュール (ECM) によって、空気温度センサー 2 (IAT2) の入力が通常のセンサー範囲よりも低いと判断されたことを示します。

デフォルト (OFF) では、エッジエージェントソフトウェアは診断コードを送信しません。

(オプション) `expiryTime`

キャンペーンの有効期限を定義できます。キャンペーンの有効期限が切れると、エッジエージェントソフトウェアはこのキャンペーンで指定されたデータの収集を停止します。車両に複数のキャンペーンがデプロイされている場合、エッジエージェントソフトウェアは他のキャンペーンを使用してデータを収集します。

デフォルト値: `253402243200` (9999 年 12 月 31 日 00:00:00 UTC)

(オプション) `postTriggerCollectionDuration`

スキームが呼び出された後、エッジエージェントソフトウェアで一定期間データを収集し続けるように、トリガー後の収集期間を定義できます。例えば、`$variable.`myVehicle.Engine.RPM` > 7000.0` という式が指定された条件ベースの収集スキームが呼び出された場合に、エッジエージェントソフトウェアでエンジンの 1 分あたりの回転数 (RPM) 値を引き続き収集できます。RPM が一度 7000 を超えただけでも、機械的な問題を示している可能性があります。この場合、エッジエージェントソフトウェアでデータの収集を継続して状況をモニタリングできます。

デフォルト値: `0`

(オプション) `priority`

キャンペーンの優先度を示す整数を指定できます。数値が小さいキャンペーンほど優先度が高くなります。1 つの車両に複数のキャンペーンをデプロイする場合、優先度の高いキャンペーンが最初に開始されます。

デフォルト値: `0`

(オプション) `signalsToCollect`

データ収集スキームが呼び出されたときにデータが収集されるシグナルのリスト。

> **Important**
> 条件ベースの収集スキームの式で使用されるシグナルは、このフィールドに指定する必要があります。

`name`

データ収集スキームが呼び出されたときにデータが収集されるシグナルの名前。

(オプション) `maxSampleCount`

データ収集スキームが呼び出されたときにエッジエージェントソフトウェアが収集してクラウドに転送するデータサンプルの最大数。

(オプション) `minimumSamplingIntervalMs`

2 つのデータサンプル収集イベント間の最小時間 (ミリ秒単位)。シグナルが頻繁に変化する場合は、このパラメータを使用してデータの収集速度を遅くすることができます。

有効な範囲: 0 ~ 4294967295

(オプション) spoolingMode

spoolingMode が TO_DISK に設定されている場合、エッジエージェントソフトウェアは、車両がクラウドに接続されていないときにデータを一時的にローカルに保存します。接続が再確立されると、ローカルに保存されたデータが自動的にクラウドに転送されます。

デフォルト値: OFF

(オプション) startTime

承認されたキャンペーンは開始時刻に有効になります。

デフォルト値: 0

キャンペーンのステータスは次のいずれかの値になります。

- CREATING - AWS IoT FleetWise はキャンペーンの作成リクエストを処理しています。
- WAITING_FOR_APPROVAL - 作成後のキャンペーンは WAITING_FOR_APPROVAL 状態になります。キャンペーンを承認するには、UpdateCampaign API オペレーションを使用します。キャンペーンが承認されると、AWS IoT FleetWise によってキャンペーンが自動的にターゲットの車両またはフリートにデプロイされます。詳細については、「キャンペーンの更新 (AWS CLI)」を参照してください。
- RUNNING - キャンペーンはアクティブです。
- SUSPENDED - キャンペーンは停止しています。キャンペーンを再開するには、UpdateCampaign API オペレーションを使用します。

AWS IoT FleetWise には、キャンペーンの作成と管理に使用できる以下の API オペレーションが用意されています。

- CreateCampaign - 新しいキャンペーンを作成します。
- UpdateCampaign - 既存のキャンペーンを更新します。キャンペーンを作成したら、この API オペレーションを使用してキャンペーンを承認する必要があります。
- DeleteCampaign - 既存のキャンペーンを削除します。
- ListCampaigns - すべてのキャンペーンの概要をページ分割されたリストとして取得します。
- GetCampaign - キャンペーンに関する情報を取得します。

チュートリアル

- キャンペーンの作成
- キャンペーンの更新 (AWS CLI)
- キャンペーンの削除
- キャンペーン情報の取得 (AWS CLI)

キャンペーンの作成

車両データを収集するキャンペーンを作成するには、AWS IoT FleetWise コンソールまたは API を使用できます。

> **Important**
>
> キャンペーンが機能するためには、次の条件が満たされている必要があります。
>
> - エッジエージェントソフトウェアが車両内で実行されている。エッジエージェントソフトウェアを開発、インストール、使用する方法の詳細を確認するには、以下の操作を行います。
>
> 1. AWS IoT FleetWise コンソールに移動します。
> 2. サービスのホームページで、[AWS IoT FleetWise の使用を開始する] セクションの [エッジエージェントを調べる] を選択します。
>
> - AWS IoT Core のセットアップが完了し、車両がプロビジョニングされている。詳細については、「車両のプロビジョニング」を参照してください。

トピック

- キャンペーンの作成 (コンソール)
- キャンペーンの作成 (AWS CLI)
- キャンペーンの論理式

キャンペーンの作成 (コンソール)

AWS IoT FleetWise コンソールを使用して、車両データを選択して収集し、クラウドに転送するキャンペーンを作成できます。

キャンペーンを作成するには

1. AWS IoT FleetWise コンソールに移動します。
2. ナビゲーションペインで、[キャンペーン] を選択します。
3. [キャンペーン] ページで、[キャンペーンを作成] を選択し、以下のトピックの手順を完了します。

トピック

> **Important**
>
> - キャンペーンを作成する前に、シグナルカタログと車両を用意する必要があります。詳細については、シグナルカタログの作成と管理 および 車両の作成、プロビジョニング、管理 を参照してください。
> - キャンペーンが作成されたら、そのキャンペーンを承認する必要があります。詳細については、「ステップ 5: キャンペーンをデプロイする」を参照してください。

ステップ 1: キャンペーンを構成する

[一般的な情報] セクションで、次の操作を行います。

1. キャンペーンの名前を入力します。
2. (オプション) 説明を入力します。

キャンペーンのデータ収集スキームを構成します。データ収集スキームは、どのようなデータをいつ収集するかに関する指示をエッジエージェントソフトウェアに与えます。AWS IoT FleetWise コンソールでは、次の方法でデータ収集スキームを構成できます。

- データ収集スキームを手動で定義します。

- データ収集スキームを自動的に定義するためのファイルをアップロードします。

[設定オプション] で、次のいずれかを選択します。

- 手動でデータ収集スキームのタイプを指定し、オプションを定義してスキームをカスタマイズするには、[データ収集スキームを定義] を選択します。

  手動でデータ収集スキームのタイプを指定し、オプションを定義してスキームをカスタマイズします。

  1. [データ収集スキームの詳細] セクションで、このキャンペーンで使用するデータ収集スキームのタイプを選択します。収集する車両データを認識するために論理式を使用するには、[条件ベース] を選択します。特定の時間間隔を使用して車両データの収集頻度を決定するには、[時間ベース] を選択します。
  2. キャンペーンでデータを収集する期間を定義します。

     > **Note**
     >
     > デフォルトでは、承認されたキャンペーンはすぐにアクティブ化され、終了時間は設定されません。追加料金が発生しないようにするには、時間範囲を指定する必要があります。

  3. 条件ベースのデータ収集スキームを指定した場合は、収集するデータを認識する論理式を定義する必要があります。AWSIoT FleetWise は、条件ベースの収集スキームで収集するデータを認識するために論理式を使用します。この式では、シグナルの完全修飾名を表す変数、比較演算子、および比較値を指定する必要があります。

     例えば、`$variable.`myVehicle.InVehicleTemperature` > 50.0` という式を指定すると、AWS IoT FleetWise は 50.0 より大きい温度値を収集します。式の書き方の手順については、「キャンペーンの論理式」を参照してください。

     収集するデータを認識するために使用される論理式を入力します。
  4. (オプション) 条件式の言語バージョンを指定できます。デフォルト値は 1 です。
  5. (オプション) 最小トリガー間隔を指定できます。これは、2 つのデータ収集イベント間の最小時間を表します。例えば、シグナルが頻繁に変化する場合は、データの収集速度を遅くすることができます。

6. エッジエージェントソフトウェアでデータを収集するための [トリガーモード] の条件を指定します。デフォルトの [常時] では、AWS IoT FleetWise 用エッジエージェントソフトウェアは条件が満たされるたびにデータを収集します。または、[最初のトリガー時] を選択して、条件が初めて満たされたときにのみデータを収集することもできます。
7. 時間ベースのデータ収集スキームを指定した場合は、[期間] を 10,000～60,000 ミリ秒で指定する必要があります。エッジエージェントソフトウェアは、その時間間隔を使用してデータの収集頻度を決定します。
8. (オプション) [高度なスキームオプション] で、スキームの高度なオプションを編集できます。

   a. データを圧縮することでワイヤレス帯域幅を節約し、ネットワークトラフィックを減らすには、SNAPPY を選択します。

   b. (オプション) データ収集イベントの後にデータを収集し続ける期間をミリ秒単位で定義するには、[トリガー後の収集期間] を指定します。

   c. (オプション) キャンペーンの優先度を指定するには、キャンペーンの [優先度] を指定します。優先度の数値が小さいキャンペーンほど優先度が高いと見なされ、最初にデプロイされます。

   d. エッジエージェントソフトウェアは、車両がクラウドに接続されていないときにデータを一時的にローカルに保存できます。接続が再確立されると、ローカルに保存されたデータが自動的にクラウドに転送されます。[データのローカル保存] で、接続の切断時にエッジエージェントでデータをローカルに保存するかどうかを指定します。

   e. (オプション) シグナルの追加情報を提供するには、[追加のデータディメンション] として最大 5 個の属性を追加します。

- データ収集スキームを定義するファイルをアップロードするには、[ローカルデバイスから .json ファイルをアップロード] を選択します。AWSIoT FleetWise は、ファイルに定義されているオプションを自動的に定義します。選択されたオプションを確認して更新できます。

  データ収集スキームに関する詳細が記述された .json ファイルをアップロードします。

  1. データ収集スキームの情報をインポートするには、[ファイルを選択] を選択します。必要なファイル形式の詳細については、API ドキュメントの「CreateCampaign」を参照してください。

     > **Note**
     >
     > AWS IoT FleetWise では現在、.json ファイル形式の拡張子がサポートされています。

2. AWS IoT FleetWise は、ファイル内の情報に基づいてデータ収集スキームを自動的に定義します。AWS IoT FleetWise によって選択されたオプションを確認します。必要に応じてオプションを更新できます。

シグナルの指定

データ収集スキームが呼び出されたときにデータを収集するシグナルを指定します。

> **Important**
> 条件ベースの収集スキームの式で使用されるシグナルは、このフィールドに指定する必要があります。

データを収集するシグナルを指定するには

1. [名前] で、シグナルの完全修飾名を検索します。

   > **Note**
   > シグナルの完全修飾名は、シグナルのパスにシグナルの名前を続けたものです。子シグナルを参照するには、ドット (.) を使用します。
   > 例えば、`Vehicle.Chassis.SteeringWheel.HandsOff.HandsOffSteeringState` は `HandsOffSteeringState` アクチュエータの完全修飾名です。`Vehicle.Chassis.SteeringWheel.HandsOff.` はこのアクチュエータのパスを示します。

2. (オプション) [最大サンプル数] に、データ収集スキームが呼び出されたときにエッジエージェントソフトウェアが収集してクラウドに転送するデータサンプルの最大数を入力します。
3. (オプション) [最小サンプリング間隔] に、2 つのデータサンプル収集イベント間の最小時間をミリ秒単位で入力します。シグナルが頻繁に変化する場合は、このパラメータを使用してデータの収集速度を遅くすることができます。
4. 別のシグナルを追加するには、[シグナルをさらに追加] を選択します。最大 999 個のシグナルを追加できます。
5. [次へ] を選択します。

ステップ 2: 保存先を定義する

> **Note**
>
> キャンペーンにビジョンシステムデータシグナルが含まれている場合のみ、車両データを Amazon S3 に転送できます。
> ビジョンシステムデータはプレビューリリースであり、変更される可能性があります。

キャンペーンで収集したデータを保存する先を選択します。Amazon S3 または Amazon Timestream に車両データを転送できます。

[送信先の設定] で、次の操作を行います。

- ドロップダウンリストから S3 または Timestream を選択します。

車両データを S3 バケットに保存する場合は、[Amazon S3] を選択します。S3 は、データをオブジェクトとしてバケットに保存するオブジェクトストレージサービスです。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「Amazon S3 バケットの作成、設定、操作」を参照してください。

S3 は、データストレージのコストを最適化し、データレイク、一元化されたデータストレージ、データ処理パイプライン、分析など、車両データを利用するための追加メカニズムを提供します。S3 を使用すると、データを保存してバッチ処理や分析を行うことができます。例えば、機械学習 (ML) モデル用に急ブレーキイベントのレポートを作成できます。受信した車両データは、配信前に 10 分間バッファリングされます。

Amazon S3

> **Important**
>
> AWS IoT FleetWise に S3 バケットへの書き込みアクセス許可がある場合のみ、S3 にデータを転送できます。アクセス権の付与の詳細については、「AWS IoT FleetWise によるアクセス制御」を参照してください。

[S3 destination settings] で、次の操作を行います。

1. [S3 bucket] で、AWS IoT FleetWise にアクセス許可があるバケットを選択します。

2. (オプション) S3 バケットに保存されているデータを体系化するために使用できるカスタムプレフィックスを入力します。
3. 出力形式を選択します。これは、S3 バケットに保存されるファイルの形式です。
4. S3 バケットに保存されたデータを .gzip ファイルとして圧縮するかどうかを選択します。ストレージコストが最小限に抑えられるため、データを圧縮することをお勧めします。
5. [S3 送信先の設定] で選択したオプションに応じて、[S3 オブジェクト URI の例] が変更されます。これは、S3 に保存されるファイルの例を示すものです。

車両データを Timestream テーブルに保存するには、[Amazon Timestream] を選択します。Timestream を使用すると、車両データにクエリを実行して傾向やパターンを特定できます。例えば、Timestream を使用して車両の燃料レベルのアラームを作成できます。受信した車両データは、ほぼリアルタイムに Timestream に転送されます。詳細については、「Amazon Timestream Developer Guide」の「What is Amazon Timestream?」を参照してください。

Amazon Timestream

> **Important**
>
> AWS IoT FleetWise に Timestream へのデータの書き込みアクセス許可がある場合のみ、テーブルにデータを転送できます。アクセス権の付与の詳細については、「AWS IoT FleetWise によるアクセス制御」を参照してください。

[Timestream テーブルの設定] で、次の操作を行います。

1. [Timestream データベース名] で、ドロップダウンリストから Timestream データベースの名前を選択します。
2. [Timestream テーブル名] で、ドロップダウンリストから Timestream テーブルの名前を選択します。

[Timestream のサービスアクセス] で、次の操作を行います。

- ドロップダウンリストから IAM ロールを選択します。
- [Next] (次へ) をクリックします。

ステップ 3: 車両を追加する

キャンペーンをデプロイする車両を選択するには、車両のリストで目的の車両を選択します。車両の作成時に追加した属性や値、または車両名で検索して、車両をフィルタリングします。

[車両をフィルタリング] で、次の操作を行います。

1. 検索ボックスで属性または車両名を検索し、リストから選択します。

   > **Note**
   >
   > 各属性は 1 回だけ使用できます。

2. キャンペーンをデプロイする属性の値または車両名を入力します。例えば、属性の完全修飾名が `fuelType` の場合は、その値として `gasoline` を入力します。
3. 別の車両属性を検索するには、前のステップを繰り返します。車両属性は最大 5 つまで、車両名はいくつでも検索できます。
4. [車両名] に、検索条件に一致する車両のリストが表示されます。キャンペーンをデプロイする先の車両を選択します。

   > **Note**
   >
   > 検索結果には最大 100 台の車両が表示されます。[すべて選択] を選択すると、すべての車両がキャンペーンに追加されます。

5. [Next] (次へ) をクリックします。

ステップ 4: 確認して作成する

キャンペーンの構成を確認し、[キャンペーンを作成] を選択します。

> **Note**
>
> キャンペーンが作成されたら、ユーザーまたはユーザーのチームがキャンペーンを車両にデプロイする必要があります。

ステップ 5: キャンペーンをデプロイする

キャンペーンを作成したら、ユーザーまたはユーザーのチームがキャンペーンを車両にデプロイする必要があります。

キャンペーンをデプロイするには

1. [キャンペーンの概要] ページで、[デプロイ] を選択します。
2. デプロイを開始してキャンペーンに接続された車両からデータ収集を開始することを確認します。
3. [デプロイ] を選択します。

キャンペーンに接続されている車両からのデータ収集を一時停止する場合は、[キャンペーンの概要] ページで [停止] を選択します。キャンペーンに接続されている車両からのデータ収集を再開するには、[再開] を選択します。

キャンペーンの作成 (AWS CLI)

CreateCampaign API オペレーションを使用すると、キャンペーンを作成できます。次の例では AWS CLI を使用しています。

キャンペーンを作成するときは、車両から収集されたデータを Amazon S3 (S3) と Amazon Timestream のいずれかに保存できます。ほぼリアルタイムの処理を必要とするデータを保存する場合など、高速でスケーラブルなサーバーレス時系列データベースが必要な場合は、Timestream を選択します。業界をリードするスケーラビリティ、データ可用性、セキュリティ、パフォーマンスを提供するオブジェクトストレージが必要な場合は、S3 を選択します。

> **Important**
>
> AWS IoT FleetWise に S3 または Timestream へのデータ書き込みアクセス許可がある場合のみ、車両データを転送できます。アクセス権の付与の詳細については、「AWS IoT FleetWise によるアクセス制御」を参照してください。

キャンペーンの作成

> **Important**
>
> - キャンペーンを作成する前に、シグナルカタログと車両またはフリートが必要です。詳細については、「シグナルカタログの作成と管理」、「車両の作成、プロビジョニング、管理」、および「フリートの作成と管理」を参照してください。
> - キャンペーンが作成されたら、UpdateCampaign API オペレーションを使用してキャンペーンを承認する必要があります。詳細については、「キャンペーンの更新 (AWS CLI)」を参照してください。

キャンペーンを作成するには、次のコマンドを実行します。

file-name は、キャンペーンの構成を含む JSON ファイルの名前に置き換えます。

```
aws iotfleetwise create-campaign --cli-input-json file://file-name.json
```

- *campaign-name* は、作成するキャンペーンの名前に置き換えます。
- *signal-catalog-arn* は、シグナルカタログの Amazon リソースネーム (ARN) に置き換えます。
- *target-arn* は、作成したフリートまたは車両の ARN に置き換えます。
- *bucket-arn* は、S3 バケットの ARN に置き換えます。

```
{
    "name": "campaign-name",
    "targetArn": "target-arn",
    "signalCatalogArn": "signal-catalog-arn",
    "collectionScheme": {
        "conditionBasedCollectionScheme": {
            "conditionLanguageVersion": 1,
            "expression": "$variable.`Vehicle.DemoBrakePedalPressure` > 7000",
            "minimumTriggerIntervalMs": 1000,
            "triggerMode": "ALWAYS"
        }
    },
    "compression": "SNAPPY",
```

```
    "diagnosticsMode": "OFF",
    "postTriggerCollectionDuration": 1000,
    "priority": 0,
    "signalsToCollect": [
        {
         "maxSampleCount": 100,
         "minimumSamplingIntervalMs": 0,
         "name": "Vehicle.DemoEngineTorque"
        },
        {
         "maxSampleCount": 100,
         "minimumSamplingIntervalMs": 0,
         "name": "Vehicle.DemoBrakePedalPressure"
        }
    ],
    "spoolingMode": "TO_DISK",
    "dataDestinationConfigs": [
        {
         "s3Config": {
             "bucketArn": "bucket-arn",
             "dataFormat": "PARQUET",
             "prefix": "campaign-name",
              "storageCompressionFormat": "GZIP"
      }
    }
  ]
}
```

- *campaign-name* は、作成するキャンペーンの名前に置き換えます。
- *signal-catalog-arn* は、シグナルカタログの Amazon リソースネーム (ARN) に置き換えます。
- *target-arn* は、作成したフリートまたは車両の ARN に置き換えます。
- *role-arn* は、Timestream テーブルにデータを配信するためのアクセス許可を AWS IoT FleetWise に付与するタスク実行ロールの ARN に置き換えます。
- *table-arn* は、Timestream テーブルの ARN に置き換えます。

```
{
  "name": "campaign-name",
  "targetArn": "target-arn",
```

```
  "signalCatalogArn": "signal-catalog-arn",
  "collectionScheme": {
    "conditionBasedCollectionScheme": {
      "conditionLanguageVersion": 1,
      "expression": "$variable.`Vehicle.DemoBrakePedalPressure` > 7000",
      "minimumTriggerIntervalMs": 1000,
      "triggerMode": "ALWAYS"
    }
  },
  "compression": "SNAPPY",
  "diagnosticsMode": "OFF",
  "postTriggerCollectionDuration": 1000,
  "priority": 0,
  "signalsToCollect": [
    {
      "maxSampleCount": 100,
      "minimumSamplingIntervalMs": 0,
      "name": "Vehicle.DemoEngineTorque"
    },
    {
      "maxSampleCount": 100,
      "minimumSamplingIntervalMs": 0,
      "name": "Vehicle.DemoBrakePedalPressure"
    }
  ],
  "spoolingMode": "TO_DISK",
  "dataDestinationConfigs": [
    {
      "timestreamConfig": {
        "executionRoleArn": "role-arn",
        "timestreamTableArn": "table-arn"
      }
    }
  ]
}
```

キャンペーンの論理式

AWS IoT FleetWise は、論理式を使用して、キャンペーンの一部として収集するデータを認識します。式の詳細については、「AWS IoT Events デベロッパーガイド」の「式」を参照してください。

式変数は、収集するデータの種類に関する規則に準拠するように構成する必要があります。テレメトリーシステムデータの場合、式変数はシグナルの完全修飾名でなければなりません。ビジョンシステ

ムデータの場合、式はシグナルの完全修飾名と、シグナルのデータ型からそのプロパティの 1 つに至るパスを組み合わせたものになります。

シグナルカタログに次のノードが含まれている場合の例:

```
{
    myVehicle.ADAS.Camera:
    type: sensor
    datatype: Vehicle.ADAS.CameraStruct
    description: "A camera sensor"

    myVehicle.ADAS.CameraStruct:
    type: struct
    description: "An obstacle detection camera output struct"
}
```

ノードが ROS 2 の定義に従っている場合の例:

```
{
    Vehicle.ADAS.CameraStruct.msg:
    boolean obstaclesExists
    uint8[] image
    Obstacle[30] obstacles
}
{
    Vehicle.ADAS.Obstacle.msg:
    float32: probability
    uint8 o_type
    float32: distance
}
```

すべての可能なイベント式変数は以下のとおりです。

```
{
...
    $variable.`myVehicle.ADAS.Camera.obstaclesExists`
    $variable.`myVehicle.ADAS.Camera.Obstacle[0].probability`
    $variable.`myVehicle.ADAS.Camera.Obstacle[1].probability`
...
    $variable.`myVehicle.ADAS.Camera.Obstacle[29].probability`
    $variable.`myVehicle.ADAS.Camera.Obstacle[0].o_type`
    $variable.`myVehicle.ADAS.Camera.Obstacle[1].o_type`
...
```

```
    $variable.`myVehicle.ADAS.Camera.Obstacle[29].o_type`
    $variable.`myVehicle.ADAS.Camera.Obstacle[0].distance`
    $variable.`myVehicle.ADAS.Camera.Obstacle[1].distance`
...
    $variable.`myVehicle.ADAS.Camera.Obstacle[29].distance`
}
```

キャンペーンの更新 (AWS CLI)

UpdateCampaign API オペレーションを使用すると、既存のキャンペーンを更新できます。次のコマンドは AWS CLI を使用します。

- *campaign-name* は、更新するキャンペーンの名前に置き換えます。
- *action* は、次のいずれかに置き換えます。
  - APPROVE - キャンペーンを承認して、AWS IoT FleetWise が車両またはフリートにデプロイできるようにします。
  - SUSPEND - キャンペーンを停止します。キャンペーンが車両から削除され、停止中のキャンペーン内のすべての車両でデータ送信が停止されます。
  - RESUME - SUSPEND で停止したキャンペーンを再開します。キャンペーンがすべての車両に再デプロイされ、車両でデータ送信が再開されます。
  - UPDATE - 属性を定義してシグナルに関連付けることで、キャンペーンを更新します。

```
aws iotfleetwise update-campaign \
            --name campaign-name \
            --action action
```

キャンペーンの削除

キャンペーンを削除するには、AWS IoT FleetWise コンソールまたは API を使用できます。

キャンペーンの削除 (コンソール)

キャンペーンを削除するには、AWS IoT FleetWise コンソールを使用できます。

キャンペーンを削除するには

1. AWS IoT FleetWise コンソールに移動します。

2. ナビゲーションペインで、[キャンペーン] を選択します。
3. [キャンペーン] ページで、ターゲットキャンペーンを選択します。
4. [削除] を選択します。
5. [**campaign-name** を削除しますか?] で、削除するキャンペーンの名前を入力し、[確認] を選択します。

キャンペーンの削除 (AWS CLI)

DeleteCampaign API オペレーションを使用すると、キャンペーンを削除できます。次の例では AWS CLI を使用しています。

キャンペーンを削除するには、次のコマンドを実行します。

campaign-name は、削除する車両の名前に置き換えます。

```
aws iotfleetwise delete-campaign --name campaign-name
```

キャンペーン情報の取得 (AWS CLI)

ListCampaigns API オペレーションを使用すると、キャンペーンが削除されたかどうかを確認できます。次の例では AWS CLI を使用しています。

すべてのキャンペーンの概要をページ分割されたリストとして取得するには、次のコマンドを実行します。

```
aws iotfleetwise list-campaigns
```

GetCampaign API オペレーションを使用すると、車両情報を取得できます。次の例では AWS CLI を使用しています。

キャンペーンのメタデータを取得するには、次のコマンドを実行します。

campaign-name は、取得するキャンペーンの名前に置き換えます。

```
aws iotfleetwise get-campaign --name campaign-name
```

> **Note**
>
> このオペレーションは結果整合性があります。言い換えると、キャンペーンへの変更はすぐには反映されない場合があります。

車両データの処理と視覚化

AWS IoT FleetWise 用エッジエージェントソフトウェアは、選択された車両データを Amazon Timestream または Amazon Simple Storage Service (Amazon S3) に転送します。データが送信先に到着したら、他の AWSのサービスを使用して、データを視覚化して共有できます。

Timestream での車両データの処理

Timestream は、1 日あたりに何兆もの時系列データポイントを保存して分析できる、フルマネージド型の時系列データベースです。データはカスタマー管理の Timestream テーブルに保存されます。Timestream を使用すると、車両データにクエリを実行して、車両に関するインサイトを得ることができます。詳細については、「What is Amazon Timestream?」を参照してください。

Timestream に転送されるデータのデフォルトのスキーマには、次のフィールドが含まれています。

フィールド名	データタイプ	説明
`eventId`	varchar	データ収集イベントの ID。
`vehicleName`	varchar	データが収集された車両の ID。
`name`	varchar	エッジエージェントソフトウェアがデータ収集に使用したキャンペーンの名前。
`time`	タイムスタンプ	データポイントのタイムスタンプ。
`measure_name`	varchar	シグナルの名前。
`measure_value::bigint`	bigint	Integer 型のシグナル値。
`measure_value::double`	double	Double 型のシグナル値。

フィールド名	データタイプ	説明	
`measure_value::boolean`	boolean	Boolean 型のシグナル値。	

Timestream に保存された車両データの視覚化

車両データが Timestream に転送されたら、次の AWSのサービスを使用して、データの視覚化、モニタリング、分析、共有を行うことができます。

- Grafana または Amazon Managed Grafana を使用して、ダッシュボードでデータを視覚化してモニタリングします。複数の AWS ソース (Amazon CloudWatch、Amazon Timestream など) やその他のデータソースからのデータを、1 つの Grafana ダッシュボードで視覚化することが可能です。
- Amazon QuickSight を使用して、ダッシュボードでデータを分析して視覚化します。

S3 での車両データの処理

Amazon S3 は、任意の量のデータを保存して保護するオブジェクトストレージサービスです。S3 は、データレイク、バックアップと復元、アーカイブ、エンタープライズアプリケーション、AWS IoT デバイス、ビッグデータ分析など、さまざまなユースケースに使用できます。データは、バケット内のオブジェクトとして S3 に保存されます。詳細については、「Amazon S3 とは」を参照してください。

Amazon S3 に転送されるデータのデフォルトのスキーマには、次のフィールドが含まれています。

フィールド名	データタイプ	説明	
`eventId`	varchar	データ収集イベントの ID。	
`vehicleName`	varchar	データが収集された車両の ID。	
`name`	varchar	エッジエージェントソフトウェアがデー	

フィールド名	データタイプ	説明
		タ収集に使用したキャンペーンの名前。
time	タイムスタンプ	データポイントのタイムスタンプ。
measure_name	varchar	シグナルの名前。
measure_value_BIGINT	bigint	Integer 型のシグナル値。
measure_value_DOUBLE	double	Double 型のシグナル値。
measure_value_BOOLEAN	boolean	Boolean 型のシグナル値。
measure_value_STRUCT	struct	Struct 型のシグナル値。

S3 オブジェクトの形式

AWS IoT FleetWise は、車両データを S3 に転送し、オブジェクトとして保存します。データを一意に識別するオブジェクト URI を使用して、キャンペーンのデータを検索できます。S3 オブジェクト URI 形式は、収集データが非構造化データか処理済みデータかによって異なります。

Unstructured data (非構造化データ)

非構造化データは、事前に定義されていない方法で S3 に保存されます。画像や動画など、さまざまな形式で保存できます。

Amazon Ion ファイルのシグナルデータと共に AWS IoT FleetWise に渡した車両メッセージはデコードされ、オブジェクトとして S3 に転送されます。S3 オブジェクトは各シグナルを表し、バイナリエンコーディングされます。

非構造化データの S3 オブジェクト URI は、次の形式を使用します。

```
s3://bucket-name/prefix/unstructured-data/random-ID-yyyy-MM-dd-HH-mm-ss-SSS-
vehicleName-signalName-fieldName
```

処理済みデータ

処理済みデータは S3 に保存され、メッセージを検証、強化、変換する処理ステップを経ます。処理済みデータの例としては、オブジェクトリストや速度があります。

S3 に転送されたデータは、約 10 分間バッファリングされたレコードを表すオブジェクトとして保存されます。デフォルトで AWS は、オブジェクトを S3 に書き込む前に、year=YYYY/month=MM/date=DD/hour=HH という形式の UTC 時間のプレフィックスを追加します。このプレフィックスにより、バケットに論理階層が作成されます。各スラッシュ (/) は階層のレベルを形成します。処理済みデータには、非構造化データへの S3 オブジェクト URI も含まれます。

処理済みデータの S3 オブジェクト URI は、次の形式を使用します。

```
s3://bucket-name/prefix/processed-data/year=YYYY/month=MM/day=DD/hour=HH/
part-0000-random-ID.gz.parquet
```

raw データ

raw データは、プライマリデータとも呼ばれ、Amazon Ion ファイルから収集されたデータです。raw データを使用して、問題のトラブルシューティングを行ったり、エラーの根本原因を特定したりできます。

raw データの S3 オブジェクト URI は、次の形式を使用します。

```
s3://bucket-name/prefix/raw-data/vehicle-name/eventID-timestamp.10n
```

S3 に保存した車両データの分析

車両データを S3 に転送したら、以下の AWS のサービスを使用して、データのモニタリング、分析、共有を行うことができます。

Amazon SageMaker を使用して、ダウンストリームのラベリングや機械学習 (ML) ワークフロー用のデータを抽出して分析します。

詳細については、「Amazon SageMaker デベロッパーガイド」の以下のトピックを参照してください。

- データを処理する
- 機械学習モデルをトレーニングする
- イメージにラベル付けする

AWS Glue クローラー を使用してデータをカタログ化し、Amazon Athena で分析します。デフォルトでは、S3 に書き込まれたオブジェクトは Apache Hive スタイルのタイムパーティションに分割され、データパスには等号で接続されたキーと値のペアが含まれます。

詳細については、「Amazon Athena ユーザーガイド」の次のトピックを参照してください。

- Athena でのデータのパーティション化
- AWS Glue を使用した Amazon S3 内のデータソースへの接続
- Athena で AWS Glue を使用するときのベストプラクティス

Amazon QuickSight を使用してデータを視覚化します。これを行うには、Athena テーブルを読み取るか、S3 バケットを直接読み取ります。

> **Tip**
>
> Amazon QuickSight は Apache Parquet 形式をサポートしていないため、S3 から直接読み取る場合は、車両データが JSON 形式であることを確認してください。

詳細については、「Amazon QuickSight ユーザーガイド」の次のトピックを参照してください。

- サポートされているデータソース
- データソースの作成

AWS CLI と AWS SDK

このセクションでは、AWS IoT FleetWise API リクエストを作成する方法について説明します。AWS IoT FleetWise のオペレーションとデータ型の詳細については、AWS IoT FleetWise の API リファレンスを参照してください。

さまざまなプログラミング言語で AWS IoT FleetWise を使用するには、AWS SDK を使用します。この SDK には、以下の自動機能が含まれています。

- サービスリクエストに暗号署名する
- リクエストを再試行する
- エラーレスポンスの処理をする

コマンドラインからのアクセスには、AWS CLI で AWS IoT FleetWise を使用します。AWS IoT FleetWise やその他のサービスをコマンドラインから制御したり、スクリプトを通じて自動化したりできます。

AWS IoT FleetWise のトラブルシューティング

このセクションのトラブルシューティング情報と解決策を使用して、AWS IoT FleetWise の問題を解決してください。

以下の情報は、AWS IoT FleetWise での一般的な問題のトラブルシューティングに役立ちます。

トピック

デコーダーマニフェストの問題

デコーダーマニフェストの問題のトラブルシューティングを行います。

デコーダーマニフェスト API コールの診断

エラー	トラブルシューティングのガイドライン
`UpdateOperationFailure.ConflictingDecoderUpdate`	同じデコーダーマニフェストに複数の更新リクエストがあります。しばらく待ってから、もう一度試してください。
`UpdateOperationFailure.InternalFailure`	InternalFailure はカプセル化された例外として起動されます。問題自体は、カプセル化された例外によって異なります。
`UpdateOperationFailure.ActiveDecoderUpdate`	デコーダーマニフェストは `Active` 状態であるため、更新できません。デコーダマニフェストの状態を `DRAFT` に変更してから、もう一度試してください。
`UpdateOperationFailure.ConflictingModelUpdate`	AWS IoT FleetWise は、他の人が変更中の車両モデル (モデルマニフェスト) を検証しようとしています。しばらく待ってから、もう一度試してください。

エラー	トラブルシューティングのガイドライン
UpdateOperationFailure.ModelManifestValidationResponse : FailureReason.MODEL_DATA_ENTRIES_NOT_FOUND	車両モデルには、シグナルが関連付けられていません。車両モデルにシグナルを追加し、シグナルが関連するシグナルカタログにあることを確認してください。
UpdateOperationFailure.ModelManifestValidationResponse : FailureReason.MODEL_NOT_ACTIVE	車両モデルを更新して ACTIVE 状態にしてから、もう一度試してください。
UpdateOperationFailure.ModelManifestValidationResponse : FailureReason.MODEL_NOT_FOUND	AWS IoT FleetWise は、デコーダーマニフェストに関連付けられた車両モデルを見つけることができません。車両モデルの Amazon リソースネーム (ARN) を確認して、もう一度試してください。
UpdateOperationFailure.ModelManifestValidationResponse (FailureReason.MODEL_DATA_ENTRIES_READ_FAILURE	車両モデルのシグナル名がシグナルカタログに見つからなかったため、車両モデルの検証に失敗しました。車両モデル内のすべてのシグナルが、関連するシグナルカタログに含まれていることを確認してください。
UpdateOperationFailure.ValidationFailure	デコーダーマニフェストの更新リクエストに、無効なシグナルまたはネットワークインターフェイスが見つかりました。例外によって返されたすべてのシグナルとネットワークインターフェイスが存在すること、使用されているすべてのシグナルが使用可能なインターフェイスに関連付けられていること、およびシグナルと関連付けられているインターフェイスを削除しないことを確認してください。
UpdateOperationFailure.KmsKeyAccessDenied	オペレーションに使用している AWS Key Management Service (AWS KMS) キーにアクセス許可の問題があります。使用しているロールがキーにアクセスできることを確認し、もう一度試してください。

エラー	トラブルシューティングのガイドライン
UpdateOperationFailure.DecoderDoesNotExist	デコーダーマニフェストが存在しません。デコーダーマニフェスト名を確認して、もう一度試してください。

SIGNAL_DECODER_INCOMPATIBLE_WITH_SIGNAL_CATALOG を原因とするビジョンシステムデータエラーメッセージには、リクエストが失敗した原因に関する情報がヒントとして応答に含まれます。このヒントを使用して、どのトラブルシューティングガイドラインに従うべきかを判断できます。

> **Note**
>
> ビジョンシステムデータはプレビューリリースであり、変更される可能性があります。

デコーダーマニフェストのビジョンシステムデータ検証の診断

エラー	トラブルシューティングのガイドライン
InvalidSignalDecoder.withReason(SignalDecoderFailureReason.NO_SIGNAL_IN_CATALOG_FOR_DECODER_SIGNAL)	AWS IoT FleetWise は、シグナルデコーダーで使用されているルートシグナル構造をシグナルカタログで見つけられませんでした。構造のルートシグナルがシグナルカタログで正しく定義されていることを確認してください。
InvalidSignalDecoder.withReason(SignalDecoderFailureReason.SIGNAL_DECODER_TYPE_INCOMPATIBLE_WITH_MESSAGE_SIGNAL_TYPE)	シグナルカタログのプリミティブメッセージが、デコーダーマニフェストの更新リクエストと同じデータ型で定義されていませんでした。リクエストで定義されているプリミティブメッセージが、対応するシグナルカタログ定義と一致することを確認してください。
InvalidSignalDecoder.withReason(SignalDecoderFailureReason.STRUCT_SIZE_MISMATCH)	シグナルカタログ内の構造体に定義されているプロパティの数が、デコーダーマニフェストでデコードしようとしているプロパティの数と一致しません。シグナルカタログで定義されてい

エラー	トラブルシューティングのガイドライン
	るシグナルと比較して、デコードするシグナルの数が正しいことを確認してください。
`InvalidSignalDecoder.withReason(SignalDecoderFailureReason.SIGNAL_DECODER_INCOMPATIBLE_WITH_SIGNAL_CATALOG)`	AWS IoT FleetWise は、デコーダマニフェストリクエストに StructuredMessageDefinition の定義がないシグナルが、シグナルカタログで STRUCT として定義されていることを検出しました。デコーダーマニフェストの更新リクエストで、各構造体が StructuredMessageDefinition として定義されていることを確認してください。
`InvalidSignalDecoder.withReason(SignalDecoderFailureReason.SIGNAL_DECODER_INCOMPATIBLE_WITH_SIGNAL_CATALOG)`	デコーダーマニフェストで使用されている構造体のルートシグナルが、シグナルカタログで構造体として正しく定義されていません。デコーダーマニフェストで使用するルートシグナル構造体には、StructFullyQualifiedName フィールドが定義されている必要があります。また、その fullyQualifiedName を持つ STRUCT ノードも必要です。
`InvalidSignalDecoder.withReason(SignalDecoderFailureReason.SIGNAL_DECODER_INCOMPATIBLE_WITH_SIGNAL_CATALOG)`	デコーダーマニフェストリクエストで使用されるリーフメッセージの 1 つがプリミティブメッセージとして定義されていません。リクエスト内のすべてのリーフオブジェクトがプリミティブメッセージとして定義されていることを確認してください。
`InvalidSignalDecoder.withReason(SignalDecoderFailureReason.SIGNAL_DECODER_INCOMPATIBLE_WITH_SIGNAL_CATALOG)`	シグナルカタログの配列オブジェクトが、デコーダーマニフェストの更新リクエストで StructuredMessageListDefinition として定義されていませんでした。デコーダーマニフェストの更新リクエストで、すべての配列プロパティが StructuredMessageListDefinition として定義されていることを確認してください。

AWS IoT FleetWise 用エッジエージェントソフトウェアの問題

エッジエージェントソフトウェアの問題のトラブルシューティングを行います。

問題

問題: エッジエージェントソフトウェアが起動しない。

エッジエージェントソフトウェアが起動しない場合、次のエラーが表示されることがあります。

- ```
  Error from reader: * Line 1, Column 1
  Syntax error: value, object or array expected.
  ```

  解決策: AWS IoT FleetWise 用エッジエージェントソフトウェアの構成ファイルが、有効な JSON 形式であることを確認します。例えば、カンマが正しく使用されていることを確認してください。構成ファイルの詳細については、次の手順を実行して、AWS IoT FleetWise 用エッジエージェントソフトウェアデベロッパーガイドをダウンロードしてください。

  1. AWS IoT FleetWise コンソールに移動します。
  2. サービスのホームページで、[AWS IoT FleetWise の使用を開始する] セクションの [エッジエージェントを調べる] を選択します。

- ```
  [ERROR] [SocketCANBusChannel::connect]: [ SocketCan with name xxx is not accessible]
  [ERROR] [IoTFleetWiseEngine::connect]: [ Failed to Bind Consumers to Producers ]
  ```

  解決策: このエラーは、エッジエージェントソフトウェアが、構成ファイルに定義されているネットワークインターフェイスとのソケット通信を確立できない場合に表示されることがあります。

構成で定義されているすべてのネットワークインターフェイスが使用可能であることを確認するには、次のコマンドを実行します。

```
ip link show
```

ネットワークインターフェイスをオンラインにするには、次のコマンドを実行します。*network-interface-id* は、ネットワークインターフェイスの ID に置き換えます。

```
sudo ip link set network-interface-id up
```

-
```
[ERROR] [AwsIotConnectivityModule::connect]: [Connection failed with error]
[WARN] [AwsIotChannel::send]: [No alive MQTT Connection.]
# or
[WARN] [AwsIotChannel::send]: [aws-c-common: AWS_ERROR_FILE_INVALID_PATH]
```

解決策: このエラーは、エッジエージェントソフトウェアが AWS IoT Core への MQTT 接続を確立できない場合に表示されることがあります。以下が正しく構成されていることを確認し、エッジエージェントソフトウェアを再起動します。

- `mqttConnection::endpointUrl` - AWS アカウントの IoT デバイスエンドポイント。
- `mqttConnection::clientID` - エッジエージェントソフトウェアが実行されている車両の ID。
- `mqttConnection::certificateFilename` - 車両証明書ファイルのパス。
- `mqttConnection::privateKeyFilename` - 車両のプライベートキーファイルのパス。
- AWS IoT Core を使用して車両がプロビジョニングされていること。詳細については、「車両のプロビジョニング」を参照してください。

トラブルシューティングの詳細については、AWS IoT Device SDK for C++ の「Frequently Asked Questions」を参照してください。

問題: [ERROR] [IoTFleetWiseEngine::connect]: [Failed to init persistency library]

解決策: このエラーは、エッジエージェントソフトウェアが永続化ストレージを検出できない場合に表示されることがあります。以下が正しく構成されていることを確認し、エッジエージェントソフトウェアを再起動します。

persistency:persistencyPath - 収集スキーム、デコーダーマニフェスト、データスナップショットの永続化に使用されるローカルパス。

問題: エッジエージェントソフトウェアがオンボードダイアグノーシス (OBD) II の PID と故障診断コード (DTC) を収集しない。

解決策: このエラーは、obdInterface:pidRequestIntervalSeconds または obdInterface:dtcRequestIntervalSeconds が 0 に設定されている場合に表示されることがあります。

エッジエージェントソフトウェアがオートマチックトランスミッションの車両で実行されている場合は、obdInterface:hasTransmissionEcu が true に設定されていることを確認してください。

車両が拡張コントローラーエリアネットワーク (CAN バス) のアービトレーション ID をサポートしている場合は、obdInterface:useExtendedIds が true に設定されていることを確認してください。

問題: AWS IoT FleetWise 用エッジエージェントソフトウェアがネットワークからデータを収集しない、またはデータ検査ルールを適用できない。

解決策: このエラーは、デフォルトのクォータを超過した場合に表示されることがあります。

[リソース]	クォータ	調整可能	注意
シグナル ID の値。	シグナル ID は 50,000 以下にする必要があります	はい	エッジエージェントソフトウェアは、50,000 より大きい ID を持つシグナルからはデータを収集しません。このクォータを変更する前に、シグナルカタログに含まれているシグナル数を確認することをお勧めします。

[リソース]	クォータ	調整可能	注意
車両あたりのアクティブなデータ収集スキームの数	256	はい	このクォータを変更する前に、クラウドで作成したキャンペーンの数と、各キャンペーンに含まれているスキームの数を確認することをお勧めします。
シグナル履歴バッファのサイズ	20 MB	はい	クォータを超過した場合、エッジエージェントソフトウェアは新しいデータの収集を停止します。

問題: [ERROR] [AwsIotConnectivityModule::connect]: [Connection failed with error] または [WARN] [AwsIotChannel::send]: [No alive MQTT Connection.]

解決策: このエラーは、エッジエージェントソフトウェアがクラウドに接続されていない場合に表示されることがあります。デフォルトでは、エッジエージェントソフトウェアは 1 分ごとに AWS IoT Core に ping リクエストを送信し、3 分間待機します。応答がない場合、エッジエージェントソフトウェアは自動的にクラウドへの接続を再確立します。

AWS IoT におけるセキュリティ FleetWise

AWS クラウドセキュリティは最優先事項です。 AWS 顧客は、最もセキュリティに敏感な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の恩恵を受けることができます。

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

- クラウドのセキュリティ — AWS AWS AWS クラウド内でサービスを実行するインフラストラクチャを保護する責任があります。 AWS また、安全に使用できるサービスも提供します。第三者監査人は、AWS、当社のセキュリティの有効性を定期的にテストおよび検証しています。 AWS IoT に適用されるコンプライアンスプログラムについては FleetWise、「コンプライアンスプログラムによる AWS 対象範囲内サービス」「コンプライアンスプログラムによる 」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、 AWS 使用する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、 AWS IoT を使用する際に責任分担モデルを適用する方法を理解するのに役立ちます FleetWise。セキュリティとコンプライアンスの目標を満たすように AWS IoT FleetWise を設定する方法を示します。また、 AWS IoT AWS FleetWise リソースの監視と保護に役立つ他のサービスの使用方法についても学びます。

コンテンツ

AWS IoT におけるデータ保護 FleetWise

AWS https://aws.amazon.com/compliance/shared-responsibility-model/、 AWS IoTのデータ保護に適用されます FleetWise。このモデルで説明したように、 AWS は、 AWS クラウドすべてを稼働させ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。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「AWS 責任共有モデルおよび GDPR」のブログ記事を参照してください。

データ保護のため、 AWS アカウント 認証情報を保護し、 AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。こうすると、それぞれのジョブを遂行するために必要なアクセス許可のみを各ユーザーに付与できます。また、以下の方法でデータを保護することをお勧めします。

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用してリソースと通信します。 AWS TLS 1.2、できれば TLS 1.3 が必要です。
- を使用して API とユーザーアクティビティの口ギングを設定します。 AWS CloudTrail
- AWS 暗号化ソリューションと、 AWS のサービスその中に含まれるデフォルトのセキュリティコントロールをすべて使用してください。
- 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 を介してアクセスするときに FIPS 140-2 で検証された暗号モジュールが必要な場合は、FIPS エンドポイントを使用してください。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-2」を参照してください。

お客様の E メールアドレスなどの機密情報やセンシティブ情報は、タグや名前フィールドなどの自由形式のフィールドに配置しないことを強くお勧めします。これには、コンソール、API AWS CLI、FleetWise または AWS SDK AWS のサービス を使用して AWS IoT やその他のものを操作する場合も含まれます。名前に使用する自由記述のテキストフィールドやタグに入力したデータは、課金や診断ログに使用される場合があります。外部サーバーへの 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。

AWS IoT FleetWise は、 AWS 車両データをクラウドに送信するために、開発してサポートされている車両ハードウェアにインストールするエッジエージェントと組み合わせて使用することを目的としています。法管轄区域によっては、車両からデータを抽出することがデータプライバシー規制の対象となる場合があります。 AWS IoT FleetWise を使用してエッジエージェントをインストールする前に、適用法に基づくコンプライアンス義務を確認することを強くお勧めします。これには、法的に適切なプライバシー通知を提示し、車両データの抽出に必要な同意を得るために適用される法的要件が含まれます。

保管中の暗号化

車両から収集されたデータは、MQTT AWS IoT Core メッセージプロトコルを使用してメッセージを介してクラウドに送信されます。 AWS IoT FleetWise はデータを Amazon タイムストリームデータベースに配信します。Timestream 内のデータは暗号化されます。デフォルトでは、 AWS のサービス 保存中のデータはすべて暗号化されます。

保存時の暗号化は AWS Key Management Service (AWS KMS) と統合され、データの暗号化に使用される暗号化キーを管理します。顧客管理キーを使用して、 AWS FleetWise IoT によって収集されたデータを暗号化することを選択できます。 AWS KMS暗号化キーはを通じて作成、管理、表示できます。詳細については、「What is AWS Key Management Service?」を参照してください。 『AWS Key Management Service 開発者ガイド』の。

転送中の暗号化

AWS IoT サービスと交換されるすべてのデータは、転送中にトランスポート層セキュリティ (TLS) を使用して暗号化されます。詳細については、「AWS IoT デベロッパーガイド」の「トランスポートセキュリティ」を参照してください。

また、 AWS IoT Core 認証と承認をサポートしているため、 AWS IoT FleetWise リソースへのアクセスを安全に制御できます。車両は X.509 証明書を使用して認証 (サインイン) を受け、 AWS IoT FleetWise を使用したり、 AWS IoT Core ポリシーを使用して特定のアクションを実行する権限 (権限を持つ) を取得したりできます。詳細については、「the section called “車両のプロビジョニング”」を参照してください。

データ暗号化

データ暗号化とは FleetWise、転送中 (AWS IoT との間やゲートウェイとサーバー間の送受信中) と保存時 (ローカルデバイスまたは内部に保存されている間) のデータを保護することです。 AWS のサービス保管中のデータの保護には、クライアント側の暗号化を使用できます。

> **Note**
>
> AWS IoT FleetWise エッジ処理は、 AWS IoT FleetWise ゲートウェイ内でホストされ、ローカルネットワーク経由でアクセス可能な API を公開します。これらの API は、 AWS IoT FleetWise Edge コネクタが所有するサーバー証明書に基づく TLS 接続を介して公開されます。これらの API では、クライアント認証にアクセス制御パスワードが使用されます。サーバー証明書の秘密鍵とアクセス制御パスワードはどちらもディスクに保存されます。 AWS IoT FleetWise エッジ処理では、保存中のこれらの認証情報のセキュリティをファイルシステムの暗号化に依存しています。

サーバー側の暗号化とクライアント側の暗号化の詳細については、以下のトピックを参照してください。

コンテンツ

保管中の暗号化

AWS IoT FleetWise AWS はデータをクラウドとゲートウェイに保存します。

保存中のデータはクラウドにあります。 AWS

AWS IoT AWS のサービス は、デフォルトで保存中のデータを暗号化する other FleetWise にデータを保存します。保管時の暗号化は AWS Key Management Service (AWS KMS) と統合され、IoT で資産プロパティ値の暗号化と値の集計に使用される暗号化キーを管理します。 AWS FleetWise顧客管理キーを使用して、 AWS FleetWise IoT の資産プロパティ値を暗号化し、値を集計することを選択できます。 AWS KMS暗号化キーはを通じて作成、管理、表示できます。

データを暗号化するには、 AWS 所有のキー またはお客様が管理するキーを選択できます。

仕組み

保存時の暗号化は、 AWS KMS データの暗号化に使用される暗号化キーの管理と統合されています。

- AWS 所有のキー — デフォルトの暗号化キー。 AWS この鍵は IoT FleetWise が所有しています。このキーを表示または管理したり、 AWS アカウントで使用したりすることはできません。また、

AWS CloudTrail キーに対する操作は口グには表示されません。このキーは追加料金なしで使用することができます。

- カスタマーマネージドキー - キーは、お客様が作成、所有、管理するアカウントに保存されます。ユーザーは、KMS キーに関する完全なコントロール権を持ちます。 AWS KMS 追加料金がかかります。

AWS 所有のキー

AWS 所有のキー アカウントには保存されません。 AWS これらは複数で使用できるように所有、管理する KMS キーのコレクションの一部です。 AWS アカウント AWS のサービス AWS 所有のキーデータを保護するために使用できます。

データの閲覧、管理、使用 AWS 所有のキー、使用状況の監査はできません。ただし、データを暗号化するキーを保護するために何らかの操作を行ったり、プログラムを変更したりする必要はありません。

使用しても料金は発生せず AWS 所有のキー、 AWS KMS アカウントのクォータにもカウントされません。

カスタマーマネージドキー

カスタマーマネージドキーは、お客様が作成、所有、管理するアカウント内の KMS キーです。これらの KMS キーはお客様が完全に制御でき、次のような操作が可能です。

- キーポリシー、IAM ポリシー、グラントの確立と維持
- 有効化と無効化
- 暗号化マテリアルのローテーション
- タグの追加
- それらを参照するエイリアスの作成
- 削除のスケジュール設定

Amazon CloudWatch Logs を使用して CloudTrail 、ユーザーに代わって AWS IoT FleetWise AWS KMS が送信するリクエストを追跡することもできます。

顧客管理キーを使用している場合は、アカウントに保存されている KMS キーへの AWS IoT FleetWise アクセスを許可する必要があります。 AWS IoT FleetWise はエンベロープ暗号化とキー階層を使用してデータを暗号化します。 AWS KMS 暗号化キーは、このキー階層のルートキーを暗号

化するために使用されます。詳細については、「AWS Key Management Service デベロッパーガイド」の「エンベロープ暗号化」を参照してください。

以下のポリシー例では、ユーザーに代わってカスタマー管理キーの作成に AWS IoT FleetWise 権限を付与します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1603902045292",
      "Action": [
        "kms:GenerateDataKey*",
        "kms:Decrypt",
        "kms:DescribeKey",
        "kms:CreateGrant",
        "kms:RetireGrant",
        "kms:RevokeGrant"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Important

KMS キーポリシーに新しいセクションを追加するときは、ポリシー内の既存のセクションを変更しないでください。 AWS IoT FleetWise で暗号化が有効になっていて、次のいずれかに該当する場合、 AWS IoT FleetWise はデータに対して操作を実行できません。

- KMS キーが無効または削除されている。
- サービスに対して KMS キーポリシーが正しく構成されていない。

ビジョンシステムデータに対する保管中の暗号化の使用

Note

ビジョンシステムデータはプレビューリリースであり、変更される可能性があります。

AWS IoT AWS KMS FleetWise アカウントでキーによる顧客管理暗号化を有効にしていて、ビジョンシステムのデータを使用したい場合は、複雑なデータタイプに対応するように暗号化設定をリセットしてください。これにより FleetWise 、 AWS IoT はビジョンシステムデータに必要な追加の権限を確立できます。

> **Note**
>
> ビジョンシステムデータの暗号化設定をリセットしていないと、デコーダーマニフェストが検証中のままになる可能性があります。

1. GetEncryptionConfigurationAPI オペレーションを使用して、 AWS KMS 暗号化が有効になっているかどうかを確認します。暗号化タイプが FLEETWISE_DEFAULT_ENCRYPTION の場合、追加のアクションは必要ありません。
2. 暗号化タイプがの場合KMS_BASED_ENCRYPTION、PutEncryptionConfigurationAPI FLEETWISE_DEFAULT_ENCRYPTION オペレーションを使用して暗号化タイプをにリセットします。

```
{
    aws iotfleetwise put-encryption-configuration --encryption-type
      FLEETWISE_DEFAULT_ENCRYPTION
 }
```

3. PutEncryptionConfigurationAPI オペレーションを使用して暗号化タイプを再度有効にします。KMS_BASED_ENCRYPTION

```
{
    aws iotfleetwise put-encryption-configuration \
        --encryption-type "KMS_BASED_ENCRYPTION"
        --kms-key-id kms_key_id
 }
```

暗号化の有効化の詳細については、「キー管理」を参照してください。

キー管理

AWS IoT FleetWise クラウドキー管理

デフォルトでは、 AWS IoT FleetWise AWS マネージドキー はを使用して内のデータを保護します AWS クラウド。カスタマー管理キーを使用して AWS FleetWise IoT のデータを暗号化するように設定を更新できます。 AWS Key Management Service (AWS KMS) を使用して暗号化キーを作成、管理、表示できます。

AWS IoT FleetWise は、 AWS KMS 顧客が管理するキーをに格納してサーバー側の暗号化をサポートし、以下のリソースのデータを暗号化します。

AWS IoT FleetWise リソース	データタイプ	保管中にカスタマーマネージドキーで暗号化されるフィールド
シグナルカタログ		説明
	属性	description、allowedValues、defaultValue、min、max
	アクチュエータ	description、allowedValues、min、max
	センサー	description、allowedValues、min、max
車両モデル (モデルマニフェスト)		説明
デコーダーマニフェスト		説明
	CanInterface	protocolName、protocolVersion
	ObdInterface	requestMessageId、 dtcRequestInterval秒、 hasTransmissionEcu、OBD スタンダード、秒、pidRequestInterval useExtendedIds

AWS IoT FleetWise リソース	データタイプ	保管中にカスタマーマネージドキーで暗号化されるフィールド
	CanSignal	係数 isBigEndian、IsSigned、長さ、messageId、オフセット、開始ビット
	ObdSignal	バイト長、オフセット、PID、スケーリング、サービスモード、開始バイト、 pidResponseLength bitMaskLength bitRightShift
車両		attributes
Campaign		説明
	conditionBasedCollectionスキーム	エクスプレッション condition LanguageVersion、 minimumTriggerInterval Ms、トリガーモード
	TimeBasedCollectionScheme	periodMs

> **Note**
>
> その他のデータやリソースは、 AWS IoT が管理するキーによるデフォルトの暗号化を使用して暗号化されます FleetWise。このキーは AWS IoT FleetWise アカウントに作成され、保存されます。

詳細については、「What is AWS Key Management Service?」を参照してください。 『AWS Key Management Service 開発者ガイド』の。

KMS キーによる暗号化の有効化 (コンソール)

カスタマーマネージドキーを AWS IoT で使用するには FleetWise、 AWS IoT FleetWise 設定を更新する必要があります。

KMS キーによる暗号化を有効にするには (コンソール)

1. AWS IoT FleetWise コンソールを開きます。
2. [設定] に移動します。
3. [暗号化] で、[編集] を選択して [暗号化の編集] ページを開きます。
4. [暗号化キータイプ] で [AWS KMS 別のキーを選択] を選択します。これにより、 AWS KMSに保存されたカスタマーマネージドキーによる暗号化が有効になります。

   > **Note**
   >
   > AWS IoT FleetWise リソースには、顧客管理のキー暗号化のみを使用できます。これには、シグナルカタログ、車両モデル (モデルマニフェスト)、デコーダーマニフェスト、車両、フリート、キャンペーンが含まれます。

5. 次のいずれかのオプションを使用して、KMS キーを選択します。

   - 既存の KMS キーを使用するには - リストから KMS キーエイリアスを選択します。
   - 新しい KMS キーを作成するには — [AWS KMS キーを作成] を選択します。

   > **Note**
   >
   > AWS KMS コンソールが開きます。KMS キーの作成の詳細については、「AWS Key Management Service デベロッパーガイド」の「キーの作成」を参照してください。

6. [保存] を選択して、設定を更新します。

KMS キーによる暗号化の有効化 (AWS CLI)

PutEncryptionConfigurationAPI オペレーションを使用して AWS IoT FleetWise アカウントの暗号化を有効にできます。以下の例ではを使用しています AWS CLI。

暗号化を有効にするには、次のコマンドを実行します。

- *KMS key id* は、KMS キーの ID に置き換えます。

```
aws iotfleetwise put-encryption-configuration –kms-key-id KMS key id –encryption-type
 KMS_BASED_ENCRYPTION
```

Example レスポンス

```
{
 "kmsKeyId": "customer_kms_key_id",
 "encryptionStatus": "PENDING",
 "encryptionType": "KMS_BASED_ENCRYPTION"
}
```

KMS キーポリシー

KMS キーを作成したら、少なくとも KMS キーポリシーに次のステートメントを追加して IoT AWS と連携させる必要があります。 FleetWise

```
{
  "Sid": "Allow FleetWise to encrypt and decrypt data when customer managed KMS key
 based encryption is enabled",
  "Effect": "Allow",
  "Principal": {
    "Service": "iotfleetwise.amazonaws.com"
  },
  "Action": [
    "kms:GenerateDataKey*",
    "kms:Decrypt",
    "kms:DescribeKey",
    "kms:CreateGrant",
    "kms:RetireGrant",
    "kms:RevokeGrant"
  ],
  "Resource": "*"
}
```

AWS IoT で使用する KMS キーポリシーの編集について詳しくは FleetWise、『AWS Key Management Service 開発者ガイド』の「キーポリシーの変更」を参照してください。

> **⚠ Important**
>
> KMS キーポリシーに新しいセクションを追加するときは、ポリシー内の既存のセクションを変更しないでください。 AWS IoT FleetWise で暗号化が有効になっていて、次のいずれかに該当する場合、 AWS IoT FleetWise はデータに対して操作を実行できません。
>
> - KMS キーが無効または削除されている。

- サービスに対して KMS キーポリシーが正しく構成されていない。

によるアクセス制御 AWS IoT FleetWise

以下のセクションでは、 AWS IoT FleetWise リソースへのアクセスとリソースからのアクセスを制御する方法について説明します。対象となる情報には、 AWS FleetWise キャンペーン中にIoTが車両データを転送できるようにアプリケーションにアクセス権を付与する方法が含まれます。また、Amazon S3 (S3) バケットまたは Amazon Timestream AWS IoT FleetWise データベースとデータを保存するテーブルへのアクセスを許可する方法についても説明しています。

これらすべてのアクセス形態を管理するテクノロジーは AWS Identity and Access Management (IAM) です。IAM の詳細については、「IAM とは?」を参照してください。

コンテンツ

Amazon S3 AWS IoT FleetWise デスティネーションへのアクセスを許可する

Amazon S3 デスティネーションを使用する場合、車両データを S3 AWS IoT FleetWise バケットに配信します。オプションで、 AWS KMS お客様が所有するキーをデータ暗号化に使用することもできます。エラーロギングが有効になっている場合は、 AWS IoT FleetWise CloudWatch データ配信エラーもロググループとストリームに送信されます。配信ストリームを作成するときは、IAM ロールが必要です。

AWS IoT FleetWise S3 宛先のサービスプリンシパルを含むバケットポリシーを使用します。バケットポリシーの追加方法の詳細については、「Amazon Simple Storage Service ユーザーガイド」の「Amazon S3 コンソールを使用したバケットポリシーの追加」を参照してください。

以下のアクセスポリシーを使用して S3 AWS IoT FleetWise バケットへのアクセスを有効にします。S3 バケットを所有していない場合、Amazon S3 アクションのリストに `s3:PutObjectAcl` を追加します。これにより、によって配信されるオブジェクトへのフルアクセスがバケット所有者に付与されます AWS IoT FleetWise。バケット内のオブジェクトへのアクセスを保護する方法の詳細については、「Amazon Simple Storage Service ユーザーガイド」の「バケットポリシーの例」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "iotfleetwise.amazonaws.com"
        ]
      },
      "Action": [
        "s3:ListBucket"
      ],
      "Resource": "arn:aws:s3:::bucket-name"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "iotfleetwise.amazonaws.com"
        ]
      },
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::bucket-name/*",
      "Condition": {
        "StringEquals": {
          "aws:SourceArn": "campaign-arn",
          "aws:SourceAccount": "account-id"
        }
      }
    }
  ]
}
```

以下のバケットポリシーは、 AWS リージョン内のアカウントのすべてのキャンペーンを対象としています。

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "iotfleetwise.amazonaws.com"
        ]
      },
      "Action": [
        "s3:ListBucket"
      ],
      "Resource": "arn:aws:s3:::bucket-name"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "iotfleetwise.amazonaws.com"
        ]
      },
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::bucket-name/*",
      "Condition": {
        "StringLike": {
          "aws:SourceArn": "arn:aws:iotfleetwise:region:account-id:campaign/*",
          "aws:SourceAccount": "account-id"
        }
      }
    }
  ]
}
```

S3 バケットに KMS キーがアタッチされている場合、そのキーには以下のポリシーが必要です。キー管理については、Amazon Simple Storage Service ユーザーガイドの「AWS Key Management Service キーによるサーバー側の暗号化 (SSE-KMS) によるデータの保護」を参照してください。

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

```
  "Principal": {
    "Service": "iotfleetwise.amazonaws.com"
  },
  "Action": [
    "kms:GenerateDataKey",
    "kms:Decrypt"
   ],
  "Resource": "key-arn"
}
```

> **Important**
>
> バケットを作成すると、S3 は、デフォルトのアクセスコントロールリスト (ACL) を作成し、リソースに対する完全なコントロールをリソース所有者に付与します。 AWS IoT が S3 FleetWise にデータを配信できない場合は、S3 バケットの ACL を必ず無効にしてください。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「すべての新しいバケットの ACL を無効にし、オブジェクト所有権を執行します。」を参照してください。

Amazon Timestream AWS IoT FleetWise 送信先へのアクセスを許可する

Timestream 宛先を使用すると、車両データが Timestream AWS IoT FleetWise テーブルに配信されます。Timestream AWS IoT FleetWise にデータを送信できるようにするには、ポリシーを IAM ロールにアタッチする必要があります。

コンソールを使用してキャンペーンを作成すると、 AWS IoT FleetWise は必要なポリシーをロールに自動的にアタッチします。

開始する前に、次の点を確認してください。

> **Important**
>
> - AWS IoT FleetWise のタイムストリームリソースを作成するときは、 AWS 同じリージョンを使用する必要があります。 AWS リージョンを切り替えると、Timestream リソースへのアクセスで問題が発生する可能性があります。
> - AWS IoT FleetWise は米国東部 (バージニア北部) とヨーロッパ (フランクフルト) で利用できます。

- サポートされているリージョンのリストについては、「AWS 全般のリファレンス」の「Timestream エンドポイントとクォータ」を参照してください。

- Timestream データベースが必要です。チュートリアルについては、「Amazon Timestream Developer Guide」の「Create a database」を参照してください。
- 指定の Timestream データベースにテーブルが作成されている必要があります。チュートリアルについては、「Amazon Timestream Developer Guide」の「Create a table」を参照してください。

を使用すると、 AWS CLI Timestream の信頼ポリシーを含む IAM ロールを作成できます。IAM ロールを作成するには、次のコマンドを実行します。

信頼ポリシーを持つ IAM ロールを作成するには

- *TimestreamExecutionRole*作成するロールの名前に置き換えてください。
- *trust-policy* は、信頼ポリシーを含む JSON ファイルに置き換えます。

```
aws iam create-role --role-name TimestreamExecutionRole --assume-role-policy-document
 file://trust-policy.json
```

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "timestreamTrustPolicy",
      "Effect": "Allow",
      "Principal": {
        "Service": "iotfleetwise.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
           "aws:SourceArn": [
            "arn:aws:iotfleetwise:region:account-id:campaign/campaign-name"
           ],
           "aws:SourceAccount": [
            "account-id"
          ]
```

```
        }
      }
    }
  ]
}
```

Timestream FleetWise にデータを書き込むための権限を AWS IoT に付与する権限ポリシーを作成します。アクセス許可を作成するには、次のコマンドを実行します。

アクセス許可ポリシーを作成するには

- *AWSIoTFleetwiseAccessTimestreamPermissionsPolicy*作成するポリシーの名前に置き換えてください。
- *permissions-policy* は、アクセス許可ポリシーを含む JSON ファイルの名前に置き換えます。

```
aws iam create-policy --policy-name AWSIoTFleetwiseAccessTimestreamPermissionsPolicy --
policy-document file://permissions-policy.json
```

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "timestreamIngestion",
      "Effect": "Allow",
      "Action": [
        "timestream:WriteRecords",
        "timestream:Select",
        "timestream:DescribeTable"
      ],
      "Resource": "table-arn"
    },
    {
      "Sid": "timestreamDescribeEndpoint",
      "Effect": "Allow",
      "Action": [
        "timestream:DescribeEndpoints"
      ],
      "Resource": "*"
    }
```

```
  ]
}
```

アクセス許可ポリシーを IAM ロールにアタッチするには

1. 出力から、アクセス許可ポリシーの Amazon リソースネーム (ARN) をコピーします。
2. IAM アクセス許可ポリシーを IAM ロールにアタッチするには、次のコマンドを実行します。
   - *permissions-policy-arn*前のステップでコピーした ARN に置き換えます。
   - 作成した IAM *TimestreamExecutionRole*ロールの名前に置き換えます。

```
aws iam attach-role-policy --policy-arn permissions-policy-arn --role-
name TimestreamExecutionRole
```

詳細については、「IAM ユーザーガイド」の「AWS リソースのアクセス管理」を参照してください。

AWS IoT 向けIdentity and Access Management FleetWise

AWS Identity and Access Management (IAM) は、 AWS のサービス AWS 管理者がリソースへのアクセスを安全に制御できるようにするものです。IAM 管理者は、誰が AWS IoT FleetWise リソースを使用するかを認証 (サインイン) および許可 (権限の付与) できるユーザーを制御します。IAM AWS のサービス は追加料金なしで使用できるアプリです。

トピック

対象者

AWS Identity and Access Management (IAM) の使い方は、 AWS IoT FleetWise で行う作業によって異なります。

サービスユーザー — AWS IoT FleetWise サービスを使用して業務を行う場合、管理者は必要な認証情報と権限を提供します。作業に多くの AWS IoT FleetWise 機能を使用するようになると、追加の権限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しておくと、管理者に適切な許可をリクエストするうえで役立ちます。 AWS IoT の機能にアクセスできない場合は FleetWise、を参照してくださいAWS IoT FleetWise ID とアクセスのトラブルシューティング。

サービス管理者 — 会社で AWS IoT FleetWise リソースを担当していれば、おそらく AWS IoT にフルアクセスできるでしょう FleetWise。サービスユーザーがどの AWS IoT FleetWise 機能やリソースにアクセスすべきかを判断するのはあなたの仕事です。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。会社が IAM を AWS IoT でどのように活用できるかについての詳細は FleetWise、を参照してくださいAWS IoT と IAM FleetWise の連携の仕組み。

IAM 管理者 — IAM 管理者の場合は、IoT へのアクセスを管理するためのポリシーを作成する方法の詳細を知りたいと思うかもしれません。 AWS FleetWiseIAM で使用できる AWS IoT FleetWise アイデンティティベースのポリシーの例については、を参照してください。IoT のアイデンティティベースのポリシーの例 AWS FleetWise

アイデンティティを使用した認証

認証とは、ID AWS 認証情報を使用してサインインする方法です。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 (サインイン AWS) する必要があります。

ID ソースを通じて提供された認証情報を使用して、フェデレーション ID AWS としてサインインできます。 AWS IAM Identity Center フェデレーテッド ID の例としては、(IAM Identity Center) ユーザー、会社のシングルサインオン認証、Google や Facebook の認証情報などがあります。フェデレーション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。 AWS フェデレーションを使用してアクセスすると、間接的にロールを引き継ぐことになります。

ユーザーのタイプによっては、 AWS Management Console AWS またはアクセスポータルにサインインできます。へのサインインについて詳しくは AWS、『AWS サインイン ユーザーガイド』の「AWS アカウントにサインインする方法」を参照してください。

AWS プログラムからアクセスする場合は、認証情報を使用してリクエストに暗号署名するためのソフトウェア開発キット (SDK) とコマンドラインインターフェイス (CLI) AWS を提供します。 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。推奨方法を使用して自分でリクエストに署名する方法の詳細については、IAM ユーザーガイドの「 AWS API リクエストへの署名」を参照してください。

使用する認証方法を問わず、セキュリティ情報の提供を追加でリクエストされる場合もあります。たとえば、アカウントのセキュリティを強化するために多要素認証 (MFA) AWS を使用することを推奨しています。詳細については、「AWS IAM Identity Center ユーザーガイド」の「多要素認証」および「IAM ユーザーガイド」の「AWSでの多要素認証 (MFA) の使用」を参照してください。

AWS アカウント root ユーザー

を作成するときは AWS アカウント、 AWS のサービス アカウント内のすべてのリソースに完全にアクセスできる 1 つのサインイン ID から始めます。この ID は AWS アカウント root ユーザーと呼ばれ、アカウントの作成に使用した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されます。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を保護し、それらを使用してルートユーザーのみが実行できるタスクを実行してください。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。

フェデレーション ID

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、ID AWS のサービス プロバイダーとのフェデレーションを使用して一時的な認証情報を使用してアクセスするように要求します。

フェデレーテッド ID とは、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、Identity Center ディレクトリのユーザー、または ID AWS のサービス ソースを通じて提供された認証情報を使用してアクセスする任意のユーザーです。 AWS Directory Serviceフェデレーテッド ID がアクセスすると AWS アカウント、そのユーザーがロールを引き受け、そのロールが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したり、独自のアイデンティティソース内のユーザーやグループに接続して同期したりして、すべてのアプリケーションで使用することができます。 AWS アカウント IAM Identity Center の詳細については、「AWS IAM Identity Center ユーザーガイド」の「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、1 人のユーザーまたはアプリケーションに対して特定の権限を持つ社内の AWS アカウント ID です。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を保有する IAM ユーザーを作成する代わりに、一時的な認証情報を使用することをお勧めします。ただし、IAM ユーザーでの長期的な認証情報が必要な特定のユースケースがある場合は、アクセスキーをローテーションすることをお勧めします。詳細については、IAM ユーザーガイドの「長期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションする」を参照してください。

IAM グループ は、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許可を与えることができます。

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

IAM ロール

IAM ロールは、 AWS アカウント 特定の権限を持つ社内の ID です。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。 AWS Management Console ロールを切り替えることで、の IAM ロールを一時的に引き受けることができます。 AWS CLI または AWS API オペレーションを呼び出すか、カスタム URL を使用してロールを引き受けることができます。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。

一時的な認証情報を持った IAM ロールは、以下の状況で役立ちます。

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

ス許可セットの詳細については、「AWS IAM Identity Center ユーザーガイド」の「アクセス許可セット」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、ロールをプロキシとして使用する代わりに AWS のサービス、ポリシーをリソースに直接アタッチできるものもあります。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 — AWS のサービス AWS のサービス他の機能を使用するものもあります。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許可、サービスロール、またはサービスリンク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アクションを実行する場合 AWS、あなた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、 AWS のサービス を呼び出したプリンシパルの権限をリクエスト元と組み合わせて使用して AWS のサービス、ダウンストリームサービスにリクエストを行います。FASリクエストは、AWS のサービス サービスが他のユーザーとのやりとりやリソースとのやり取りを必要とするリクエストを受信したときにのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。
  - 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、にリンクされている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。 AWS アカウント サービスにリンクされたロールはに表示され、そのサービスが所有します。IAM 管理者は、サービスリンクロールの許可を表示できますが、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 API AWS CLI リクエストを行うアプリケーションの一時的な認証情報を管理できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。EC2 AWS インスタンスにロールを割り当て、そのロールを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る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 ロールを使用して許可を付与する」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

AWS ポリシーを作成して AWS ID またはリソースにアタッチすることで、アクセスを制御します。ポリシーとは、ID またはリソースに関連付けると権限を定義するオブジェクトです。 AWS AWS プリンシパル (ユーザー、ルートユーザー、またはロールセッション) がリクエストを行うと、これらのポリシーを評価します。ポリシーでの権限により、リクエストが許可されるか拒否されるかが決まります。ほとんどのポリシーは JSON AWS ドキュメントとして保存されます。JSON ポリシードキュメントの構造と内容の詳細については、「IAM ユーザーガイド」の「JSON ポリシー概要」を参照してください。

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

デフォルトでは、ユーザーやロールに権限はありません。IAM 管理者は、リソース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、ユーザーグループ、ロールなど、アイデンティティにアタッチできる JSON 許可ポリシードキュメントです。これらのポリシーは、ユーザー

とロールが実行できるアクション、リソース、および条件をコントロールします。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のポリシーは、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、およびロールにアタッチできるスタンドアロンポリシーです。 AWS アカウント管理ポリシーには、 AWS 管理ポリシーと顧客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。ポリシーが添付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、ロール、フェデレーティッドユーザ、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。IAM AWS の管理ポリシーをリソースベースのポリシーで使用することはできません。

アクセスコントロールリスト (ACL)

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための権限を持つかをコントロールします。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

ACL をサポートするサービスの例としては AWS WAF、Amazon S3、、および Amazon VPC があります。ACL の詳細については、「Amazon Simple Storage Service デベロッパーガイド」の「アクセスコントロールリスト (ACL) の概要」を参照してください。

その他のポリシータイプ

AWS あまり一般的ではないポリシータイプもサポートしています。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- アクセス許可の境界 - アクセス許可の境界は、アイデンティティベースのポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる許可の上限を設定する高度な機能です。エンティティに権限の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとその権限の境界の共通部分になります。`Principal` フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、権限の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、許可は無効になります。アクセス許可の境界の詳細については、「IAM ユーザーガイド」の「IAM エンティティのアクセス許可の境界」を参照してください。
- サービスコントロールポリシー (SCP) — SCP は、組織または組織単位 (OU) の最大権限を指定する JSON ポリシーです。 AWS Organizations AWS Organizations は、 AWS アカウント 企業が所有する複数のものをグループ化して一元管理するためのサービスです。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP は、メンバーアカウントのエンティティ (各エンティティを含む) の権限を制限します。 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ティッド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される範囲は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポリシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合もあります。これらのポリシーのいずれかを明示的に拒否した場合、許可は無効になります。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。 AWS 複数のポリシータイプが関係している場合にリクエストを許可するかどうかを決定する方法については、IAM ユーザーガイドの「ポリシー評価ロジック」を参照してください。

AWS IoT と IAM FleetWise の連携の仕組み

IAM を使用して AWS IoT へのアクセスを管理する前に FleetWise、IoT で使用できる IAM 機能について学んでください。 AWS FleetWise

IoT で使用できる IAM 機能 AWS FleetWise

IAM 機能	AWS IoT FleetWise 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	いいえ
ポリシーアクション	Yes
ポリシーリソース	はい
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	いいえ
サービスリンクロール	No

AWS IoT FleetWise AWS やその他のサービスがほとんどの IAM 機能でどのように機能するかを大まかに把握するには、IAM ユーザーガイドの「IAM AWS と連携するサービス」を参照してください。

IoT のアイデンティティベースのポリシー AWS FleetWise

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、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件を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学ぶには、「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。

IoT のアイデンティティベースのポリシーの例 AWS FleetWise

AWS IoT FleetWise ID 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。IoT のアイデンティティベースのポリシーの例 AWS FleetWise

IoT 内のリソースベースのポリシー AWS FleetWise

リソースベースのポリシーのサポート	なし

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、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。ポリシーが添付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、ロール、フェデレーティッドユーザ、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権限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。

AWS IoT に関するポリシーアクション FleetWise

ポリシーアクションに対するサポート	Yes

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

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には通常、関連する AWS API オペレーションと同じ名前が付けられます。一致する API オペレーションのない 許可のみのアクション 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アクセス許可を付与するポリシーで使用されます。

AWS IoT FleetWise アクションのリストについては、『サービス認証リファレンス』の「AWS IoT FleetWise によって定義されるアクション」を参照してください。

AWS IoT のポリシーアクションは、 FleetWise アクションの前に次のプレフィックスを使用します。

```
iotfleetwise
```

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": [
      "iotfleetwise:action1",
      "iotfleetwise:action2"
]
```

ワイルドカード (*) を使用して複数アクションを指定できます。例えば、`List` という単語で始まるすべてのアクションを指定するには、次のアクションを含めます。

```
"Action": "iotfleetwise:List*"
```

AWS IoT FleetWise ID 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。IoT のアイデンティティベースのポリシーの例 AWS FleetWise

AWS IoT のポリシーリソース FleetWise

ポリシーリソースに対するサポート	Yes

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

`Resource` JSON ポリシーの要素は、オブジェクトあるい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は、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、 リソースレベルの許可 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、リソースレベルのアクセス許可をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

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

AWS IoT FleetWise リソースタイプとその ARN のリストについては、『サービス認証リファレンス』の「AWS IoT FleetWise によって定義されるリソース」を参照してください。各リソースの ARN を指定できるアクションについては、「AWS IoT FleetWise で定義されるアクション」を参照してください。

AWS IoT FleetWise ID 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。IoT のアイデンティティベースのポリシーの例 AWS FleetWise

AWS IoT のポリシー条件キー FleetWise

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 がどの リソース に対してどのような条件下 で アクション を実行できるかということです。

Condition 要素 (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要素を指定する場合、または 1 つの Condition要素に複数のキーを指定する場合、 AWS では 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、 AWS OR論理演算を使用して条件を評価します。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 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。 AWS すべての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 FleetWise 条件キーのリストについては、『サービス認証リファレンス』の「 AWS IoT FleetWise の条件キー」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、「AWS IoT で定義されるアクション」を参照してください FleetWise 。

AWS IoT FleetWise ID 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。IoT のアイデンティティベースのポリシーの例 AWS FleetWise

IoT のアクセス制御リスト (ACL) AWS FleetWise

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許可を持つかをコントロールします。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

IoT による属性ベースのアクセス制御 (ABAC) AWS FleetWise

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。IAM エンティティ (ユーザーまたはロール) AWS や多くのリソースにタグを付けることができます。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。次に、プリンシパルのタグがアクセスを試行するリソースのタグと一致したときにオペレーションを許可するよう、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 Yes です。サービスが一部のリソースタイプに対してのみ 3 つの条件キーすべてをサポートする場合、値は Partial です。

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。

> **Note**
>
> AWS IoT は `CreateCampaign` API FleetWise 操作に必要なサポートのみをサポートしますiam:PassRole。

AWS IoT での一時的な認証情報の使用 FleetWise

一時的な認証情報のサポート	Yes

AWS のサービス 一時的な認証情報を使用してサインインすると機能しないものもあります。 AWS のサービス 一時的な認証情報で機能するものなど、追加情報については、『IAM ユーザーガイド』の「IAM と連携する」を参照してくださいAWS のサービス 。

ユーザー名とパスワード以外の方法でにサインインすると、 AWS Management Console 一時的な認証情報が使用されることになります。たとえば、会社のシングルサインオン (SSO) AWS リンクを使用してアクセスすると、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。

または API を使用して一時的な認証情報を手動で作成できます。 AWS CLI AWS その後、その一時的な認証情報を使用してアクセスできます AWS。 AWS 長期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すすめします。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。

IoT のクロスサービスプリンシパル権限 AWS FleetWise

フォワードアクセスセッション (FAS) をサポート	Yes

IAM ユーザーまたはロールを使用してアクションを実行する場合 AWS、その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、を呼び出したプリンシパルの権限と AWS のサービス、 AWS のサービス ダウンストリームサービスにリクエストを行うリクエストを組み合わせて使用します。FASリクエストは、 AWS のサービス サービスが他のユーザーとのやりとりやリソースとのやり取りを必要とするリクエストを受信したときにのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。

AWS IoT のサービスロール FleetWise

サービスロールのサポート	いいえ

サービスロールとは、サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、「IAM ユーザーガイド」の「AWS のサービスに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**
>
> サービスロールの権限を変更すると、 AWS IoT FleetWise の機能が損なわれる可能性があります。サービスロールを編集するのは、 AWS IoT FleetWise がガイダンスを提供する場合に限ります。

IoT のサービスにリンクされた役割 AWS FleetWise

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、にリンクされている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。AWS アカウント サービスにリンクされたロールはに表示され、そのサービスが所有します。IAM 管理者は、サービスリンク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。

サービスリンク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、サービスにリンクされたロール 列に Yesと記載されたサービスを見つけます。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[はい] リンクを選択します。

AWS IoT のサービスにリンクされたロールの使用 FleetWise

AWS IoT FleetWise 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、AWS IoT に直接リンクされているユニークなタイプの IAM ロールです。 FleetWiseサービスにリンクされたロールは AWS IoT FleetWise によって事前定義されており、AWS IoT FleetWise がメトリクスをAmazonに送信するために必要な権限が含まれています。 CloudWatch詳細については、「Amazon CloudWatch による AWS IoT FleetWise のモニターリング」を参照してください。

サービスにリンクされたロールを使用すると、必要な権限を手動で追加する必要がないため、AWS IoT FleetWise をより迅速にセットアップできます。AWS IoT FleetWise はサービスにリンクされたロールの権限を定義し、特に定義されていない限り、AWS IoT FleetWise のみがその役割を引き受けることができます。定義された許可には、信頼ポリシーと許可ポリシーが含まれます。このアクセス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、FleetWise リソースにアクセスする権限を誤って削除することがなくなるため、AWS IoT リソースが保護されます。

サービスにリンクされた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、「サービスにリンクされたロール」列が「はい」になっているサービスを確認してください。そのサービスの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参照するには、「はい」のリンクを選択します。

AWS IoT のサービスにリンクされたロール権限 FleetWise

AWS IoT FleetWise は、AWS AWS IoT T out-of-the-box のすべての権限に使用される AWS AWSServiceRoleForIoTFleetWise管理ポリシー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。 FleetWise

AWSServiceRoleForIoTFleetWise サービスにリンクされたロールは、以下のサービスを信頼してロールを引き受けます。

- `IoTFleetWise`

AWSIoTFleetwiseServiceRolePolicy という名前のロールアクセス権限ポリシーにより、AWS IoT FleetWise は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ます。

- アクション: リソース * での `cloudwatch:PutMetricData`

サービスリンクロールの作成、編集、削除をIAM エンティティ (ユーザー、グループ、ロールなど) に許可するには、許可を設定する必要があります。詳細については、「IAM User Guide」(IAM ユーザーガイド) の「Service-linked role permissions」(サービスリンクロールのアクセス権限) を参照してください。

AWS IoT のサービスにリンクされたロールの作成 FleetWise

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 IoT FleetWise コンソール、、または AWS API でアカウントを登録すると AWS CLI、AWS IoT FleetWise はサービスにリンクされたロールを自動的に作成します。詳細については、「設定の構成」を参照してください。

AWS IoT FleetWise (コンソール) でのサービスにリンクされたロールの作成

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 IoT FleetWise コンソール、 AWS CLI、または AWS API でアカウントを登録すると、AWS IoT FleetWise はサービスにリンクされたロールを自動的に作成します。

AWS IoT のサービスにリンクされたロールの編集 FleetWise

AWS IoT AWSServiceRoleForIoTFleetWise のサービスにリンクされたロールは編集できません。FleetWise作成済みのサービスにリンクされたロールは、さまざまなエンティティによって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 を使用してロールの説明を編集することはできます。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。

サービスリンクロールのクリーンアップ

IAM を使用してサービスリンクロールを削除するには、最初に、そのロールで使用されているリソースをすべて削除する必要があります。

> **Note**
>
> AWS IoT FleetWise がロールを使用してリソースを削除しようとすると、削除が失敗する可能性があります。失敗した場合は、数分待ってから操作を再試行してください。コンソール、 AWS CLI、または AWS API service-linked-role を使用してを削除する方法については、IAM ユーザーガイドの「サービスにリンクされたロールの使用」を参照してください。

このサービスにリンクされたロールを削除してから再度作成する必要がある場合は、AWS IoT にアカウントを登録できます。 FleetWise FleetWise その後、AWS IoT はサービスにリンクされたロールを再度作成します。

IoT のアイデンティティベースのポリシーの例 AWS FleetWise

デフォルトでは、ユーザーとロールには AWS IoT FleetWise リソースを作成または変更する権限がありません。また、 AWS Management Console、 AWS Command Line Interface (AWS CLI)、 AWS API を使用してタスクを実行することもできません。IAM 管理者は、リソース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 FleetWise、 AWS IoT によって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「AWS IoT FleetWise のアクション、リソース、および条件キー」を参照してください。

トピック

- ポリシーのベストプラクティス
- AWS IoT FleetWise コンソールの使用
- 自分の許可の表示をユーザーに許可する
- Amazon Timestream 内のリソースへのアクセス

ポリシーのベストプラクティス

ID ベースのポリシーは、アカウント内の AWS IoT FleetWise リソースを誰かが作成、アクセス、または削除できるかどうかを決定します。これらのアクションを実行すると、 AWS アカウントに料金が発生する可能性があります。アイデンティティベースの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 管理ポリシーから始めて、最小権限の権限に移行する — ユーザーとワークロードへの権限の付与を開始するには、AWS 多くの一般的なユースケースで権限を付与する管理ポリシーを使用してください。これらのポリシーは、で利用できます。 AWS アカウント AWS ユースケースに固有のカスタマー管理ポリシーを定義して、権限をさらに減らすことをお勧めします。詳細については、「IAM ユーザーガイド」の「AWS マネージドポリシー」または「AWS ジョブ機能の管理ポリシー」を参照してください。
- 最小特権を適用する – IAM ポリシーで許可を設定するときは、タスクの実行に必要な許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、 最小特権アクセス許可 とも呼ばれています。IAM を使用して許可を適用する方法の詳細については、「IAM ユーザーガイド」の「IAM でのポリシーとアクセス許可」を参照してください。
- IAM ポリシーで条件を使用してアクセスをさらに制限する – ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。例えば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。サービスアクションがなどの特定の用途で使用された場合は AWS のサービス、条件を使用してサービスアクションへのアクセスを許可することもできます 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) が必要 — IAM ユーザーまたは root 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 をオンにしてください。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 FleetWise コンソールの使用

AWS IoT FleetWise コンソールにアクセスするには、最低限の権限が必要です。これらの権限により、内の AWS IoT FleetWise リソースに関する詳細を一覧表示および表示できる必要があります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 のみを呼び出しているユーザーには、最低限のコンソール権限を付与する必要はありません。代わりに、実行しようとしている API オペレーションに一致するアクションのみへのアクセスが許可されます。

ユーザーとロールが引き続き AWS IoT FleetWise コンソールを使用できるようにするには、 AWS IoT FleetWise `ConsoleAccess` `ReadOnly` AWS または管理ポリシーもエンティティにアタッチします。詳細については、『IAM ユーザーガイド』の「ユーザーへの権限の追加」を参照してください。

自分の許可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添付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を作成する方法を示します。このポリシーには、コンソールで、またはまたは API を使用してこのアクションをプログラム的に実行するための権限が含まれています。 AWS CLI AWS

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
```

```
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon Timestream 内のリソースへのアクセス

AWS IoT を使用する前に FleetWise、 AWS アカウント、IAM、Amazon Timestream リソースを登録して、 AWS クラウド お客様に代わって車両データを送信する AWS IoT FleetWise 権限を付与する必要があります。登録するには以下が必要です。

- Amazon Timestream データベース。
- 指定の Amazon Timestream データベースに作成されたテーブル。
- AWS IoT が Amazon Timestream FleetWise にデータを送信できるようにする IAM ロール。

手順やポリシーの例を含む詳細については、「設定の構成」を参照してください。

AWS IoT FleetWise ID とアクセスのトラブルシューティング

以下の情報を参考にして、 AWS IoT や IAM FleetWise を使用する際に発生する可能性のある一般的な問題の診断と修正に役立ててください。

トピック

- AWS IoT でアクションを実行する権限がありません FleetWise
- 私には IAM を実行する権限がありません:PassRole

- 自分以外の人にも自分の AWS IoT AWS アカウント FleetWise リソースへのアクセスを許可したい

AWS IoT でアクションを実行する権限がありません FleetWise

AWS Management Console アクションを実行する権限がないと表示された場合は、管理者に連絡して支援を受ける必要があります。管理者とは、サインイン認証情報を提供した担当者です。

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

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 iotfleetwise:GetVehicleStatus on resource: myVehicle
```

この場合、Mateo は、iotfleetwise:*GetVehicleStatus* アクションを使用して *myVehicle* リソースにアクセスできるように、管理者にポリシーの更新を依頼します。

私には 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、 AWS IoT にロールを渡せるようにポリシーを更新する必要があります FleetWise。

新しいサービスロールやサービスにリンクされたロールを作成する代わりに、 AWS のサービス 既存のロールをそのサービスに渡すことができるものもあります。そのためには、サービスにロールを渡すアクセス許可が必要です。

次のエラー例は、という名前の IAM ユーザーがコンソールを使用して AWS IoT marymajor FleetWise でアクションを実行しようとしたときに発生します。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、メアリー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 管理者に問い合わせてください。サインイン資格情報を提供した担当者が管理者です。

自分以外の人にも自分の AWS IoT AWS アカウント FleetWise リソースへのアクセスを許可したい

他のアカウントのユーザーや組織外の人が、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセス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:

- AWS IoT FleetWise がこれらの機能をサポートするかどうかについては、を参照してくださいAWS IoT と IAM FleetWise の連携の仕組み。
- AWS アカウント 所有しているリソース全体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「AWS アカウント 所有する別の 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第三者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「AWS アカウント 第三者が所有するリソースへのアクセスの提供」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、「IAM ユーザーガイド」の「外部で認証されたユーザー (ID フェデレーション) へのアクセスの許可」を参照してください。
- クロスアカウントアクセスでのロールとリソースベースのポリシーの使用の違いの詳細については、「IAM ユーザーガイド」の「IAM ロールとリソースベースのポリシーとの相違点」を参照してください。

AWS IoT のコンプライアンス検証 FleetWise

> **Note**
>
> AWS FleetWiseIoT AWS はコンプライアンスプログラムの対象外です。

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、「AWS のサービス コンプライアンスプログラム別の範囲」の「」を参照して、関心のあるコンプライアンスプログラムを選択してください。一般的な情報については、「AWS」を参照してください。

サードパーティの監査レポートはを使用してダウンロードできます AWS Artifact。詳細については、の「レポートのダウンロード」の「 AWS Artifact」を参照してください AWS Artifact。

AWS のサービス を使用する際のコンプライアンス責任は、データの機密性、会社のコンプライアンス目標、および適用される法律と規制によって決まります。 AWS コンプライアンスに役立つ以下のリソースを提供しています。

- セキュリティとコンプライアンスのクイックスタートガイド — これらの導入ガイドでは、アーキテクチャ上の考慮事項について説明し、 AWS セキュリティとコンプライアンスに重点を置いたベースライン環境をデプロイする手順を説明しています。
- Amazon Web Services での 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 HIPAA 対応アプリケーションを作成する方法について説明しています。 AWS

  > **Note**
  >
  > すべての企業が AWS のサービス HIPAA に適格という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 — この一連のワークブックとガイドは、お客様の業界や地域に当てはまる場合があります。
- 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見た責任分担モデルを理解してください。このガイドでは、 AWS のサービス セキュリティを確保するためのベストプラクティスをまとめ、複数のフレームワーク (米国標準技術研究所 (NIST)、ペイメントカード業界セキュリティ標準評議会 (PCI)、国際標準化機構 (ISO) など) にわたるセキュリティ管理にガイダンスをまとめています。
- AWS Config 開発者ガイドのルールによるリソースの評価 — AWS Config このサービスでは、リソース構成が社内慣行、業界ガイドライン、規制にどの程度準拠しているかを評価します。
- AWS Security Hub— AWS のサービス これにより、内部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。
- AWS Audit Manager— AWS のサービス これにより、 AWS 使用状況を継続的に監査して、リスクの管理や規制や業界標準への準拠を簡素化できます。

IoT AWS におけるレジリエンス FleetWise

AWS AWS グローバルインフラストラクチャはリージョンとアベイラビリティーゾーンを中心に構築されています。リージョンには、低レイテンシー、高いスループット、そして高度の冗長ネット

ワークで接続されている複数の物理的に独立および隔離されたアベイラビリティーゾーンがあります。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 リージョンと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。

> **Note**
>
> AWS IoT FleetWise によって処理されたデータは Amazon Timestream データベースに保存されます。Timestream AWS は他のアベイラビリティーゾーンまたはリージョンへのバックアップをサポートします。Timestream SDK を使用して、データにクエリを実行し、任意の宛先に保存する独自のアプリケーションを作成することもできます。
> Amazon Timestream の詳細については、「Amazon Timestream Developer Guide」を参照してください。

AWS IoT におけるインフラストラクチャーセキュリティ FleetWise

マネージドサービスとして、 AWS FleetWise AWS IoTはグローバルネットワークセキュリティによって保護されています。 AWS AWS セキュリティサービスとインフラストラクチャの保護方法については、「AWS クラウドセキュリティ」を参照してください。 AWS インフラストラクチャセキュリティのベストプラクティスを使用して環境を設計するには、「Security Pillar AWS Well-Architected Framework におけるインフラストラクチャ保護」を参照してください。

AWS 公開されている API 呼び出しを使用して、 FleetWise ネットワーク経由で AWS IoT にアクセスします。クライアントは以下をサポートする必要があります:

- Transport Layer Security (TLS)。TLS 1.2、できれば TLS 1.3 が必要です。
- DHE (Ephemeral Diffie-Hellman) や ECDHE (Elliptic Curve Ephemeral Diffie-Hellman) などの Perfect Forward Secrecy (PFS) を使用した暗号スイート。これらのモードは、Java 7 以降など、ほとんどの最新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 Service (AWS

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。

これらの API 操作はどのネットワークロケーションからでも呼び出すことができますが、 AWS IoT FleetWise はリソースベースのアクセスポリシーをサポートしており、これにはソース IP アドレスに基づく制限が含まれる場合があります。 AWS IoT FleetWise ポリシーを使用して、特定の Amazon Virtual Private Cloud (Amazon VPC) エンドポイントまたは特定の VPC からのアクセスを制御することもできます。これにより、特定の AWS IoT FleetWise リソースへのネットワークアクセスが、ネットワーク内の特定の VPC からのみ効果的に分離されます。 AWS

トピック

- インターフェイス VPC AWS FleetWise エンドポイントを介した IoT への接続

インターフェイス VPC AWS FleetWise エンドポイントを介した IoT への接続

インターネット経由で接続する代わりに、仮想プライベートクラウド (VPC AWS PrivateLink) のインターフェイス VPC エンドポイント () を使用して AWS IoT FleetWise に直接接続できます。インターフェイス VPC エンドポイントを使用する場合、VPC と AWS IoT FleetWise AWS 間の通信はすべてネットワーク内で行われます。各 VPC エンドポイントは、VPC サブネット内のプライベート IP アドレスを持つ 1 つ以上の Elastic Network Interface (ENI) で表されます。

インターフェイス VPC エンドポイントは、インターネットゲートウェイ、NAT デバイス、VPN 接続、 FleetWise AWS Direct Connect または接続なしで VPC を AWS IoT に直接接続します。VPC 内のインスタンスは AWS IoT FleetWise API と通信するためにパブリック IP アドレスを必要としません。

VPC FleetWise 経由で AWS IoT を使用するには、VPC 内のインスタンスから接続するか、 AWS Virtual Private Network (VPN) またはを使用してプライベートネットワークを VPC に接続する必要があります。 AWS Direct Connect Amazon VPN については、「Amazon Virtual Private Cloud ユーザーガイド」の「VPN 接続」を参照してください。詳細については AWS Direct Connect、『ユーザーガイド』の「接続の作成」を参照してください。AWS Direct Connect

AWS コンソールまたは AWS Command Line Interface (AWS CLI) コマンドを使用して、 AWS IoT FleetWise に接続するためのインターフェイス VPC エンドポイントを作成できます。詳細については、「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。

インターフェイス VPC エンドポイントを作成した後、エンドポイントのプライベート DNS ホスト名を有効にすると、デフォルトの AWS IoT エンドポイントは VPC FleetWise エンドポイントに解決されます。 AWS IoT FleetWise のデフォルトのサービス名エンドポイントは次の形式です。

```
iotfleetwise.Region.amazonaws.com
```

プライベート DNS ホスト名を有効にしない場合、Amazon VPC は次の形式で使用できる DNS エンドポイント名を提供します。

```
VPCE_ID.iotfleetwise.Region.vpce.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 VPC エンドポイント (AWS PrivateLink)」を参照してください。

AWS IoT FleetWise は VPC 内のすべての API アクションへの呼び出しをサポートしています。

VPC エンドポイントポリシーを VPC エンドポイントにアタッチして、IAM プリンシパルのアクセスを制御できます。また、セキュリティグループを VPC エンドポイントに関連付けて、ネットワークトラフィックの送信元と送信先 (IP アドレスの範囲など) に基づいてインバウンドとアウトバウンドのアクセスを制御することもできます。詳細については、「VPC エンドポイントによるサービスのアクセスコントロール」を参照してください。

IoT 用 AWS VPC エンドポイントポリシーの作成 FleetWise

AWS IoT 用 Amazon VPC FleetWise エンドポイントのポリシーを作成して、以下を指定できます。

- アクションを実行できるプリンシパルまたは実行できないプリンシパル
- 実行できるアクションまたは実行できないアクション

詳細については、「Amazon VPC ユーザーガイド」の「VPC エンドポイントによるサービスのアクセスコントロール」を参照してください。

Example — 指定したアカウントからのすべてのアクセスを拒否する VPC エンドポイントポリシー AWS

次の VPC エンドポイントポリシーは、 AWS アカウント *123456789012* をエンドポイントを使用するすべての API 呼び出しを拒否します。

```
{
```

```
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        },
        {
            "Action": "*",
            "Effect": "Deny",
            "Resource": "*",
            "Principal": {
                "AWS": [
                    "123456789012"
                ]
            }
        }
    ]
}
```

Example - 指定した IAM プリンシパル (ユーザー) への VPC アクセスのみを許可する VPC エンドポイントポリシー

VPC ################## 123456789012 ##### lijuan ############### AWS 他のすべての IAM プリンシパルによるエンドポイントへのアクセスは拒否されます。

```
{
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::123456789012:user/lijuan"
                ]
            }
        }]
}
```

Example — AWS IoT FleetWise アクションの VPC エンドポイントポリシー

以下は、 AWS IoT FleetWise のエンドポイントポリシーの例です。*########################## 123456789012 # IAM #### FleetWise ######### AWS IoT FleetWise ############ ##### AWS #####*

```
{
    "Statement": [
        {
             "Principal": {
                "AWS": [
                    "arn:aws:iam::123456789012:user/fleetWise"
                },
            "Resource": "*",
            "Effect": "Allow",
            "Action": [
                "iotfleetwise:ListFleets",
                "iotfleetwise:ListCampaigns",
                "iotfleetwise:CreateVehicle",
            ]
           }
    ]
}
```

AWS IoT における構成と脆弱性の分析 FleetWise

IoT 環境は、多様な機能を持ち、存続期間が長く、地理的に分散される多数のデバイスで設定されることがあります。このような特性によってデバイスのセットアップが複雑になり、エラーを起こしやすくなります。また、デバイスの計算能力、メモリ、ストレージの機能には制約があることが多いため、デバイスでの暗号化や他の形式のセキュリティの使用は制限されます。多く場合、デバイスは既知の脆弱性を持つソフトウェアを使用しています。これらの要因により、 AWS IoT用のデータを収集する車両を含むIoTデバイスはハッカーにとって魅力的な標的となり FleetWise、それらを継続的に保護することが困難になっています。

構成と IT 制御は、 AWS お客様とお客様との間で共有される責任です。詳細については、「 AWS 責任分担モデル」を参照してください。

AWS IoT のセキュリティベストプラクティス FleetWise

AWS IoT FleetWise には、独自のセキュリティポリシーを策定して実装する際に考慮すべきセキュリティ機能が数多く用意されています。以下のベストプラクティスは一般的なガイドラインであり、完全なセキュリティソリューションを提供するものではありません。これらのベストプラクティスはお客様の環境に必ずしも適切または十分でない可能性があるため、処方箋ではなく、あくまで有用な検討事項とお考えください。

セキュリティについて学ぶには、『AWS IoT 開発者ガイド』 AWS IoT Coreの AWS IoT 「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。

最小限のアクセス許可を付与する

IAM ロールの最小限のアクセス許可セットを使用して、最小特権の原則に従います。IAM ポリシーの `Action` プロパティおよび `Resource` プロパティに対する `*` ワイルドカードの使用を制限します。代わりに、可能な場合はアクションとリソースの有限セットを宣言します。最小特権およびその他のポリシーのベストプラクティスの詳細については、「the section called “ポリシーのベストプラクティス”」を参照してください。

機密情報を記録しない

認証情報やその他の個人を特定できる情報 (PII) のログを記録しないようにしてください。次の安全対策を実施することをお勧めします。

- デバイス名に機密情報を使用しない。
- AWS IoT FleetWise リソースの名前や ID に機密情報を使用しないでください。たとえば、キャンペーン、デコーダーマニフェスト、車両モデル、信号カタログの名前、車両や車両の ID などです。

API AWS CloudTrail 呼び出し履歴の表示に使用します。

セキュリティ分析や運用上のトラブルシューティングを目的として、アカウントで行われた AWS IoT FleetWise API 呼び出しの履歴を表示できます。アカウントで行われた AWS IoT FleetWise API 呼び出しの履歴を受信するには、 CloudTrail をオンにするだけです AWS Management Console。詳細については、「the section called “CloudTrail ログ”」を参照してください。

デバイスのクロックを同期させる

デバイスの時刻を正確に保つことが重要です。X.509 証明書には有効期限の日時があります。デバイスのクロックは、サーバー証明書が現在も有効であることを確認するために使用されます。時間の経過とともにデバイスのクロックがドリフトしたり、バッテリが放電したりする可能性があります。

詳細については、「AWS IoT Core デベロッパーガイド」の「[Keep your device's clock in sync] (デバイスのクロックを同期させる)」ベストプラクティスを参照してください。

AWS IoT FleetWise のモニタリング

モニタリングは、AWS IoT FleetWise やその他の AWS ソリューションの信頼性、可用性、パフォーマンスを維持するうえで重要な要素です。AWS には、AWS IoT FleetWise をモニタリングし、問題が発生したときに報告し、必要に応じて自動アクションを実行する以下のモニタリングツールが用意されています。

- Amazon CloudWatch は、AWS のリソースおよび 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、および指定したしきい値にメトリクスが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 を使用すると、Amazon EC2 インスタンス、CloudTrail、その他のソースからのログファイルをモニタリングおよび保存し、アクセスすることができます。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 アカウントによって行われた、またはそのアカウントに代わって実行された API コールと関連イベントをキャプチャします。次に、ユーザー指定の 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。

Amazon CloudWatch による AWS IoT FleetWise のモニターリング

Amazon CloudWatch のメトリックスは、AWS のリソースとそのパフォーマンスをモニタリングする手段となるものです。AWSIoT FleetWise は、メトリクスを CloudWatch に送信します。AWS Management Console、AWS CLI、または API を使用すると、AWS IoT FleetWise から CloudWatch に送信されるメトリクスのリストを取得できます。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。

> **Important**
>
> AWS IoT FleetWise から CloudWatch にメトリクスを送信できるように、設定を構成する必要があります。詳細については、「設定の構成」を参照してください。

`AWS/IoTFleetWise` 名前空間には、次のメトリクスが含まれます。

シグナルに関するメトリクス

メトリクス	説明
IllegalMessageFromEdge	車両から送信され、AWS IoT FleetWise で受信したメッセージが、必要な形式と一致しませんでした。 単位: カウント ディメンション: VehicleName 有効な統計: Sum
MessageThrottled	車両から AWS IoT FleetWise に送信されたメッセージがスロットリングされました。これは、このアカウントの現在のリージョンにおけるサービス制限を超えているためです。 単位: カウント ディメンション: VehicleName 有効な統計: Sum
ModelingError	車両から送信され、AWS IoT FleetWise で受信したメッセージに、車両モデルに対する検証に失敗するシグナルが含まれています。 単位: カウント ディメンション: ModelManifestName

メトリクス	説明
	有効な統計: Sum
DecodingError	車両から送信され、AWS IoT FleetWise で受信したメッセージに、車両のデコーダーマニフェストに対するデコードに失敗するシグナルが含まれています。 単位: カウント ディメンション: DecoderName 有効な統計: Sum

キャンペーンに関するメトリクス

メトリクス	説明
VehicleNotFound	車両から送信され、AWS IoT FleetWise で受信したメッセージに、不明な車両が含まれています。 単位: カウント ディメンション: VehicleName 有効な統計: Sum
CampaignInvalid	車両から送信され、AWS IoT FleetWise で受信したメッセージに、無効なキャンペーンが含まれています。 単位: カウント ディメンション: CampaignName 有効な統計: Sum

メトリクス	説明
CampaignNotFound	車両から送信され、AWS IoT FleetWise で受信したメッセージに、不明なキャンペーンが含まれています。 単位: カウント ディメンション: CampaignName 有効な統計: Sum

キャンペーンデータ送信先メトリクス

メトリクス	説明
TimestreamWriteError	AWS IoT FleetWise は、車両から Amazon Timestream テーブルにメッセージを書き込むことができませんでした。 単位: カウント ディメンション: DatabaseName、TableName 有効な統計: Sum
S3WriteError	AWSIoT FleetWise は、車両から Amazon Simple Storage Service (Amazon S3) バケットにメッセージを書き込むことができませんでした。 単位: カウント ディメンション: BucketName 有効な統計: Sum
S3ReadError	AWS IoT FleetWise は、Amazon Simple Storage Service (Amazon S3) バケットで車両のオブジェクトキーを読み取れませんでした。

メトリクス	説明
	単位: カウント ディメンション: BucketName 有効な統計: Sum

カスタマーマネージド AWS KMS キーに関するメトリクス

メトリクス	説明
KMSKeyAccessDenied	AWS KMS キーへのアクセス拒否エラーにより、AWS IoT FleetWise は、車両から Timestream テーブルまたは Amazon S3 バケットにメッセージを書き込むことができませんでした。 単位: カウント ディメンション: KMSKeyId 有効な統計: Sum

Amazon CloudWatch Logs による AWS IoT FleetWise のモニタリング

Amazon CloudWatch Logs は、リソースで発生するイベントをモニタリングし、問題がある場合にアラートを発行します。アラートを受け取った場合は、ログファイルにアクセスして、その特定のイベントに関する情報を取得できます。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。

CloudWatch コンソールでの AWS IoT FleetWise ログの表示

> **Important**
>
> CloudWatch コンソールで AWS IoT FleetWise ロググループを表示するには、次の条件が満たされている必要があります。

- AWS IoT FleetWise でログ記録が有効になっている。ログ記録の詳細については、「AWS IoT FleetWise のログ記録の構成」を参照してください。
- AWS IoT オペレーションによって書き込まれたログエントリが既に存在する。

CloudWatch コンソールでAWS IoT FleetWise ログを表示するには

1. CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[ログ]、[ロググループ] の順に選択します。
3. ロググループを選択します。
4. [ロググループの検索] を選択します。アカウントに対して生成されたログイベントの完全なリストが表示されます。
5. 展開アイコンを選択して個々のストリームを確認し、ログレベルが ERROR のログをすべて見つけます。

   [イベントをフィルター] テキストボックスにクエリを入力することもできます。例えば、次のクエリを実行できます。

   ```
   { $.logLevel = "ERROR" }
   ```

フィルター式の詳細については、「Amazon CloudWatch Logs ユーザーガイド」の「フィルターパターン構文」を参照してください。

Example ログエントリ

```
{
  "accountId": "123456789012",
  "vehicleName": "test-vehicle",
  "message": "Unrecognized signal ID",
  "eventType": "MODELING_ERROR",
  "logLevel": "ERROR",
  "timestamp": 1685743214239,
  "campaignName": "test-campaign",
  "signalCatalogName": "test-catalog",
  "signalId": 10242
}
```

シグナルに関するイベントタイプ

イベントタイプ	説明
MODELING_ERROR	車両から送信され、AWS IoT FleetWise で受信したメッセージに、車両モデルに対する検証に失敗するシグナルが含まれています。 属性: vehicleName、campainName、signalCatalogName、signalId、signalValue、signalValueRangeMin、signalValueRangeMax、modelManifestName
ILLEGAL_MESSAGE_FROM_EDGE	車両から送信され、AWS IoT FleetWise で受信したメッセージが、必要な形式と一致しませんでした。 属性: vehicleName、campaignName、signalCatalogName
DECODING_ERROR	車両から送信され、AWS IoT FleetWise で受信したメッセージに、車両のデコーダーマニフェストに対するデコードに失敗するシグナルが含まれています。 属性: campaignName、signalCatalogName、decoderManifestName、(オプション) signalName、(オプション) s3URI

キャンペーンに関するイベントタイプ

イベントタイプ	説明
VEHICLE_NOT_FOUND	車両から送信され、AWS IoT FleetWise で受信したメッセージに、不明な車両が含まれています。 属性: vehicleName、campaignName

イベントタイプ	説明
CAMPAIGN_NOT_FOUND	車両から送信され、AWS IoT FleetWise で受信したメッセージに、不明なキャンペーンが含まれています。 属性: vehicleName (オプション)、campaignName
CAMPAIGN_INVALID	車両から送信され、AWS IoT FleetWise で受信したメッセージに、無効なキャンペーンが含まれています。 属性: vehicleName (オプション)、campaignName

キャンペーンデータ送信先イベントタイプ

イベントタイプ	説明
TIMESTREAM_WRITE_ERROR	AWS IoT FleetWise は、車両から Amazon Timestream テーブルにメッセージを書き込むことができませんでした。 属性: vehicleName、campaignName、timestreamDatabaseName、timestreamTableName
S3_WRITE_ERROR	AWSIoT FleetWise は、車両から Amazon Simple Storage Service (Amazon S3) バケットにメッセージを書き込むことができませんでした。 属性: campaignName、destinationName
S3_READ_ERROR	AWS IoT FleetWise は、Amazon Simple Storage Service (Amazon S3) バケットで車両のオブジェクトキーを読み取れませんでした。

イベントタイプ	説明
	属性: campaignName、destinationName

カスタマーマネージド AWS KMS キーに関するイベントタイプ

イベントタイプ	説明
KMS_KEY_ACCESS_DENIED	AWS KMS キーへのアクセス拒否エラーにより、AWS IoT FleetWise は、車両から Timestream テーブルまたは Amazon S3 バケットにメッセージを書き込むことができませんでした。

属性

すべての CloudWatch Logs エントリには、以下の属性が含まれます。

accountId

自分の AWS アカウント ID。

eventType

ログが生成されたイベントタイプ。イベントタイプの値は、ログエントリが生成される原因となったイベントによって異なります。各ログエントリの説明には、その ログエントリの `eventType` の値が含まれます。

logLevel

使用されているログレベル。詳細については、「AWS IoT Core デベロッパーガイド」の「ログレベル」を参照してください。

message

ログに関する具体的な詳細が含まれています。

タイムスタンプ

AWS IoT FleetWise がログを処理したときのエポックミリ秒のタイムスタンプ。

オプションの属性

CloudWatch Logs エントリには、`eventType` に応じてオプションで以下の属性が含まれます。

decoderManifestName

シグナルを含むデコーダーマニフェストの名前。

destinationName

車両データの送信先の名前。例えば、Amazon S3 バケット名を示します。

campaignName

キャンペーンの名前。

signalCatalogName

シグナルが含まれているシグナルカタログの名前。

signalId

エラーシグナルの ID。

signalIds

エラーシグナル ID のリスト。

signalName

シグナルの名前。

signalTimestampEpochMs

エラーシグナルのタイムスタンプ。

signalValue

エラーシグナルの値。

signalValueRangeMax

エラーシグナルの最大範囲。

signalValueRangeMin

エラーシグナルの最小範囲。

s3URI

車両メッセージに含まれる Amazon Ion ファイルの Amazon S3 固有の識別子。

timestreamDatabaseName

Timestream データベースの名前。

timestreamTableName

Timestream テーブルの名前。

vehicleName

車両モデルの名前。

AWS IoT FleetWise のログ記録の構成

AWS IoT FleetWise のログデータを CloudWatch ロググループに送信できます。CloudWatch Logs によって可視性が提供され、AWS IoT FleetWise が車両からのメッセージの処理に失敗した場合に備えることができます。そのような状況は、例えば、構成の誤りやその他のクライアントエラーが原因で発生する可能性があります。何らかのエラーがある場合は通知されるため、問題を特定して軽減できます。

CloudWatch にログを送信するには、事前に CloudWatch ロググループを作成する必要があります。ロググループは、AWS IoT FleetWise で使用したものと同じアカウントおよび同じリージョンで構成してください。AWS IoT FleetWise でログ記録を有効にするときは、ロググループ名を指定します。ログ記録が有効になると、AWS IoT FleetWise はログストリームの CloudWatch ロググループにログを配信します。

AWS IoT FleetWise から送信されたログデータは、CloudWatch コンソールで表示できます。CloudWatch ロググループの構成の詳細については、「ロググループとログストリームの操作」を参照してください。

CloudWatch にログを発行するためのアクセス許可

CloudWatch ロググループのログ記録を構成するには、このセクションで説明するアクセス許可設定が必要です。アクセス許可の管理については、「IAM ユーザーガイド」の「AWS リソースのアクセス管理」を参照してください。

これらのアクセス許可があると、ログ記録の構成の変更、CloudWatch のログ配信の構成、ロググループに関する情報の取得が可能になります。

```
{
```

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

            "iotfleetwise:PutLoggingOptions",
            "iotfleetwise:GetLoggingOptions"
         ],
         "Resource":[
            "*"
         ],
         "Effect":"Allow",
         "Sid":"IoTFleetwiseLoggingOptionsAPI"
      }
      {
         "Sid":"IoTFleetwiseLoggingCWL",
         "Action":[
            "logs:CreateLogDelivery",
            "logs:GetLogDelivery"
            "logs:UpdateLogDelivery",
            "logs:DeleteLogDelivery",
            "logs:ListLogDeliveries",
            "logs:PutResourcePolicy",
            "logs:DescribeResourcePolicies",
            "logs:DescribeLogGroups"
         ],
         "Resource":[
            "*"
         ],
         "Effect":"Allow"
      }
   ]
}
```

すべての AWS リソースでアクションが許可される場合、ポリシーでは "Resource" 設定が "*" として示されます。これは、各アクションがサポートするすべての AWS リソースでアクションが許可されることを意味します。

AWS IoT FleetWise でのログ記録の構成 (コンソール)

ここでは、AWS IoT FleetWise コンソールを使用してすべてのログ記録を構成する方法について説明します。

AWS IoT FleetWise コンソールを使用してログ記録を構成するには

1. AWS IoT FleetWise コンソールを開きます。
2. 左側のペインで、[設定] を選択します。
3. [設定] ページの [ログ] セクションで、[編集] を選択します。
4. [CloudWatch ログ記録] セクションで、[ロググループ] を入力します。
5. 変更を保存するには、[送信] を選択します。

ログ記録を有効にしたら、CloudWatch コンソールでログデータを表示できます。

AWS でのデフォルトのログ記録の構成 (CLI)

ここでは、CLI を使用して AWS IoT FleetWise のログ記録を構成する方法について説明します。

この手順は、ここに示す CLI コマンドに対応する AWS API のメソッドを使用することにより、API で行うこともできます。GetLoggingOptions API オペレーションを使用して現在の構成を取得し、PutLoggingOptions API オペレーションを使用して構成を変更できます。

CLI を使用して AWS IoT FleetWise のデフォルトのログ記録を構成するには

1. アカウントのログ記録オプションを取得するには、get-logging-options コマンドを使用します。

   ```
   aws iotfleetwise get-logging-options
   ```

2. ログ記録を有効にするには、put-logging-options コマンドを使用します。

   ```
   aws iotfleetwise put-logging-options --cloud-watch-log-delivery
    logType=ERROR,logGroupName=MyLogGroup
   ```

   各パラメータの意味は次のとおりです。

   logType

   CloudWatch Logs にデータを送信するログのタイプ。ログ記録を無効にするには、値を OFF に変更します。

   logGroupName

   このオペレーションでデータを送信する先の CloudWatch Logs グループ。AWS IoT FleetWise のログ記録を有効にする前に、必ずロググループ名を作成してください。

ログ記録を有効にしたら、「AWS CLI を使用したログエントリの検索」を参照してください。

AWS CloudTrail を使用した AWS IoT FleetWise API コールのログ記録

AWS IoT FleetWise は AWS CloudTrail と統合されています。これは、AWS IoT FleetWise のユーザー、ロール、AWS サービスによって実行されたアクションを記録するサービスです。CloudTrail は、AWS IoT FleetWise のすべての API コールをイベントとしてキャプチャします。キャプチャされるコールには、AWS IoT FleetWise コンソールからの呼び出しと、コードからの AWS IoT FleetWise API オペレーションの呼び出しが含まれます。証跡を作成すると、AWS IoT FleetWise のイベントなど、CloudTrail イベントの Amazon S3 バケットへの継続的な配信を有効にすることができます。証跡を構成しない場合でも、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 で収集された情報を使用して、AWS IoT FleetWise に対するリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

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 FleetWise 情報

AWS アカウントを作成すると、そのアカウントに対して CloudTrail が有効になります。AWS IoT FleetWise でアクティビティが発生すると、そのアクティビティは他の AWS サービスのイベントと共に、CloudTrail イベントの [イベント履歴] に記録されます。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」を参照してください。

AWS IoT FleetWise のイベントなど、AWS アカウントのイベントを継続的に記録するには、証跡を作成します。証跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、コンソールで作成した追跡がすべての AWS リージョンに適用されます。追跡は、AWS パーティションのすべてのリージョンからのイベントを口グに記録し、指定した Simple Storage Service (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。

- 証跡の作成に関する概要
- 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
- CloudTrail ログファイルの複数のリージョンからの受け取り

- 複数のアカウントから CloudTrail ログファイルを受け取る

すべての AWS IoT FleetWise アクションは、CloudTrail によってログに記録されます。これらのアクションは「AWS IoT FleetWise API Reference」で説明されています。例えば、`CreateCampaign`、`AssociateVehicleFleet`、`GetModelManifest` の各アクションを呼び出すと、CloudTrail ログファイルにエントリが生成されます。

各イベントまたはログエントリには、リクエストの生成者に関する情報が含まれます。同一性情報は次の判断に役立ちます。

- リクエストが、ルートと IAM 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または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を使用して行われたかどうか。
- 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 エレメント」を参照してください。

AWS IoT FleetWise ログファイルエントリについて

証跡とは、指定した Amazon S3 バケットにイベントをログファイルとして配信できるようにする構成です。CloudTrail ログファイルには、1 つ以上のログエントリがあります。イベントは任意のソースからの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエストのパラメータなどの情報が含まれます。CloudTrail ログファイルは、パブリック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

`AssociateVehicleFleet` オペレーションを示す CloudTrail ログエントリの例は、次のとおりです。

```
{
      "eventVersion": "1.05",
      "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::111122223333:assumed-role/NikkiWolf",
        "accountId": "111122223333",
        "accessKeyId": "access-key-id",
        "userName": "NikkiWolf"
      },
      "eventTime": "2021-11-30T09:56:35Z",
```

```
    "eventSource": "iotfleetwise.amazonaws.com",
    "eventName": "AssociateVehicleFleet",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.21",
    "userAgent": "aws-cli/2.3.2 Python/3.8.8 Darwin/18.7.0 botocore/2.0.0",
    "requestParameters": {
        "fleetId": "f1234567890",
        "vehicleId": "v0213456789"
     },
    "responseElements": {
    },
    "requestID": "9f861429-11e3-11e8-9eea-0781b5c0ac21",
    "eventID": "17385819-4927-41ee-a6a5-29ml0br812v4",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
  }
```

AWS IoT FleetWise デベロッパーガイドのドキュメント履歴

次の表は、AWS IoT FleetWise のドキュメントリリースをまとめたものです。

変更	説明	日付
ビジョンシステムデータのプレビュー	AWS IoT FleetWise のビジョンシステムデータのプレビューを使用して、カメラ、レーダー、LIDAR などの車両ビジョンシステムからデータを収集して整理できます。構造化および非構造化の両方のビジョンシステムデータ、メタデータ (イベント ID、キャンペーン、車両)、および標準センサー (テレメトリーデータ) をクラウド内で自動的に同期します。	2023 年 11 月 26 日
AWS KMS カスタマーマネージドキー	AWS IoT FleetWise が AWS KMS カスタマーマネージドキーをサポートするようになりました。KMS キーを使用して、AWS クラウドに保存されている AWS IoT FleetWise リソース (シグナルカタログ、車両モデル、デコーダーマニフェスト、車両、およびデータ収集キャンペーン設定) に関連するサーバー側のデータを暗号化できます。	2023 年 10 月 16 日
Amazon S3 のオブジェクトストレージ	AWS IoT FleetWise は、Amazon Simple Storage Service (Amazon S3) へのデータの保存をサポートする	2023 年 6 月 1 日

	ようになりました。キャンペーン中に収集したデータは、Amazon Timestream に加えて Amazon S3 にも保存できます。	
一般提供	これは AWS IoT FleetWise の一般リリースです。	2022 年 9 月 27 日
初回リリース	これは AWS IoT FleetWise デベロッパーガイドのプレビューリリースです。	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。