개발자 가이드

AWS IoT FleetWise

AWS IoT FleetWise: 개발자 가이드

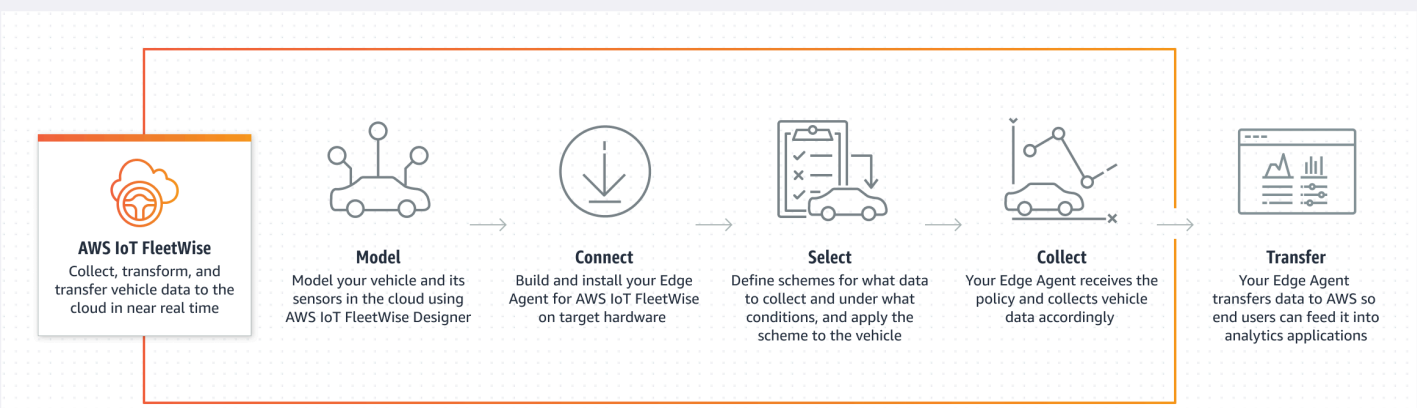
Table of Contents

AWS IoT FleetWise 란 무엇입니까?

AWS IoT FleetWise는 차량 데이터를 수집하여 클라우드에 구성하는 데 사용할 수 있는 관리형 서비스입니다. 수집된 데이터를 사용하여 차량 품질, 성능 및 자율성을 개선할 수 있습니다. AWS IoT FleetWise를 사용하면 다양한 프로토콜과 데이터 형식을 사용하는 차량에서 데이터를 쉽게 수집하고 구성할 수 있습니다. AWS IoT FleetWise는 저수준 메시지를 사람이 읽을 수 있는 값으로 변환하고 데이터 분석을 위해 클라우드에서 데이터 형식을 표준화하는 데 도움을 줍니다. 또한 데이터 수집 캠페인을 정의하여 수집할 차량 데이터와 해당 데이터를 클라우드로 전송할 시기를 제어할 수 있습니다.

차량 데이터가 클라우드에 있으면 플릿 상태를 분석하는 애플리케이션에 사용할 수 있습니다. 이 데이터는 잠재적인 유지보수 문제를 식별하고, 차량 내 인포테인먼트 시스템을 더 스마트하게 만들고, 분석 및 기계 학습(ML)을 통해 자율 주행 및 운전자 지원 시스템과 같은 고급 기술을 개선할 수 있습니다.

다음 다이어그램은 AWS IoT FleetWise의 기본 아키텍처를 보여줍니다.



주제

이점

AWS IoT FleetWise의 주요 이점은 다음과 같습니다.

보다 지능적으로 차량 데이터 수집

분석을 위해 필요한 데이터만 클라우드로 전송하는 지능형 데이터 수집을 통해 데이터 관련성을 개선할 수 있습니다.

표준화된 플릿 전체 데이터를 쉽게 분석

사용자 지정 데이터 수집 또는 로깅 시스템을 개발할 필요 없이 차량 플릿의 표준화된 데이터를 분석할 수 있습니다.

클라우드에서 자동으로 데이터 동기화

표준 센서(텔레메트리 데이터)와 비전 시스템에서 수집한 데이터(카메라, 레이더, 라이다의 데이터)를 통합적으로 확인하고 클라우드에서 자동으로 동기화된 상태로 유지할 수 있습니다. AWS IoT FleetWise는 정형 및 비정형 비전 시스템 데이터, 메타데이터, 표준 센서 데이터를 클라우드에서 자동으로 동기화합니다. 이를 통해 이벤트를 전체적으로 파악하고 인사이트를 얻는 프로세스가 간소화됩니다.

> **Note**
>
> 비전 시스템 데이터는 평가판 릴리스이며 변경될 수 있습니다.

사용 사례

AWS IoT FleetWise를 사용할 수 있는 시나리오는 다음과 같습니다.

AI/ML 모델 훈련

프로덕션 차량에서 데이터를 수집하여 자율 주행 및 첨단 운전자 지원 시스템에 사용되는 기계 학습 모델을 지속적으로 개선합니다.

디지털 고객 경험 향상

인포테인먼트 시스템의 데이터를 사용하여 차량 내 시청각 콘텐츠와 인앱 인사이트를 보다 관련성 있게 만듭니다.

차량 플릿 상태 유지

플릿 데이터에서 얻은 인사이트를 사용하여 EV 배터리 상태 및 충전 수준을 모니터링하고, 유지 보수 일정을 관리하고, 연료 소비를 분석하는 등의 작업을 수행합니다.

AWS IoT FleetWise를 처음 사용하시나요?

AWS IoT FleetWise를 처음 사용하는 경우 먼저 다음 섹션을 읽어보는 것이 좋습니다.

- AWS IoT FleetWise 작동 방식
- AWS IoT FleetWise 설정
- 엣지 에이전트 소프트웨어 데모
- 클라우드로 데이터 수집

AWS IoT FleetWise 액세스

AWS IoT FleetWise 콘솔 또는 API를 사용하여 AWS IoT FleetWise에 액세스할 수 있습니다.

AWS IoT FleetWise 가격

차량은 MQTT 메시지를 통해 데이터를 클라우드로 전송합니다. AWS IoT FleetWise에서 만든 차량에 대한 비용을 매월 말에 지불합니다. 또한 차량에서 수집한 메시지에 대한 비용도 지불합니다. 요금에 대한 최신 정보는 AWS IoT FleetWise 요금 페이지를 참조하세요. MQTT 메시징 프로토콜에 대해 자세히 알아보려면 AWS IoT Core개발자 안내서의 MQTT를 참조하세요.

AWS IoT FleetWise 작동 방식

다음 섹션에서 AWS IoT FleetWise 서비스 구성 요소 및 상호 작용 방식을 간단히 설명합니다.

이 소개를 읽은 후 AWS IoT FleetWise를 설정하는 방법을 알아보려면 AWS IoT FleetWise 설정 섹션을 참조하세요.

주제

- 주요 개념
- AWS IoT FleetWise의 특징

주요 개념

AWS IoT FleetWise는 클라우드에서 차량과 센서 및 액추에이터를 모델링할 수 있는 차량 모델링 프레임워크를 제공합니다. 차량과 클라우드 간의 보안 통신을 지원하기 위해 AWS IoT FleetWise는 차량에 설치할 수 있는 Edge Agent 소프트웨어 개발에 도움이 되는 참조 구현도 제공합니다. 클라우드에서 데이터 수집 체계를 정의하고 이를 차량에 배포할 수 있습니다. 차량에서 실행 중인 Edge Agent 소프트웨어는 데이터 수집 체계를 사용하여 수집할 데이터와 클라우드로 전송할 시기를 제어합니다.

다음은 AWS IoT FleetWise의 핵심 개념입니다.

신호

신호는 차량 데이터와 해당 메타데이터를 포함하도록 정의하는 기본 구조입니다. 신호는 속성, 분기, 센서 또는 액추에이터일 수 있습니다. 예를 들어, 차량 내 온도 값을 수신하고 센서 이름, 데이터 유형 및 단위를 포함한 메타데이터를 저장하는 센서를 생성할 수 있습니다. 자세한 내용은 신호 카탈로그 생성 및 관리 섹션을 참조하세요.

속성

속성은 일반적으로 변경되지 않는 정적 정보(예: 제조업체 및 제조일)를 나타냅니다.

브랜치

브랜치는 신호를 중첩된 구조로 나타냅니다. 브랜치는 신호 계층 구조를 보여줍니다. 예를 들어, `Vehicle` 브랜치에는 하위 브랜치, `Powertrain`이(가) 있습니다. `Powertrain` 브랜치에는 하위 브랜치, `combustionEngine`이(가) 있습니다. `combustionEngine` 브랜치를 찾으려면 `Vehicle.Powertrain.combustionEngine` 표현식을 사용하세요.

센서

센서 데이터는 차량의 현재 상태와 시간에 따른 변화(예: 유체 수준, 온도, 진동, 전압 등)를 보고합니다.

액추에이터

액추에이터 데이터는 모터, 히터, 도어록과 같은 차량 장치의 상태를 보고합니다. 차량 장치의 상태를 변경하면 액추에이터 데이터를 업데이트할 수 있습니다. 예를 들어, 히터를 나타내는 액추에이터를 정의할 수 있습니다. 히터를 켜거나 끌 때 액추에이터가 새 데이터를 수신합니다.

사용자 지정 구조

사용자 지정 구조(구조체라고도 함)는 복합 또는 고차 데이터 구조를 나타냅니다. 이를 통해 동일한 소스에서 생성된 데이터를 쉽게 논리적으로 바인딩하거나 그룹화할 수 있습니다. 구조체는 복합 데

이터 유형 또는 고차 모양을 나타내는 것과 같이 원자성 연산으로 데이터를 읽거나 쓸 때 사용됩니다.

구조체 유형의 신호는 프리미티브 데이터 유형 대신 구조체 데이터 유형에 대한 참조를 사용하여 신호 카탈로그에 정의됩니다. 구조체는 센서, 속성, 액추에이터, 비전 시스템 데이터 유형을 비롯한 모든 유형의 신호에 사용할 수 있습니다. 구조체 유형의 신호가 송신되거나 수신되는 경우 AWS IoT FleetWise는 포함된 모든 항목이 유효한 값을 가질 것으로 예상하므로 모든 항목은 필수입니다. 예를 들어 구조체에 Vehicle.Camera.Image.height, Vehicle.Camera.Image.width 및 Vehicle.Camera.Image.data 항목이 포함된 경우 송신된 신호에 이러한 모든 항목의 값이 포함될 것으로 예상됩니다.

> **Note**
>
> 비전 시스템 데이터는 평가판 릴리스이며 변경될 수 있습니다.

사용자 지정 속성

사용자 지정 속성은 복합 데이터 구조의 멤버를 나타냅니다. 속성의 데이터 유형은 프리미티브이거나 다른 구조체일 수 있습니다.

구조체와 사용자 지정 속성을 사용하여 고차 모양을 표현할 때는 의도한 고차 모양이 항상 트리 구조로 정의되고 표시됩니다. 사용자 지정 속성은 모든 리프 노드를 정의하는 데 사용되고 구조체는 리프가 아닌 모든 노드를 정의하는 데 사용됩니다.

신호 카탈로그

신호 카탈로그에는 신호 컬렉션이 포함되어 있습니다. 신호 카탈로그에 있는 신호는 다양한 프로토콜과 데이터 형식을 사용하는 차량을 모델링할 때 사용할 수 있습니다. 예를 들어, 서로 다른 자동차 제조업체에서 만든 두 대의 자동차가 있습니다. 하나는 제어 영역 네트워크(CAN 버스) 프로토콜을 사용하고 다른 하나는 OBD(온보드 진단) 프로토콜을 사용합니다. 신호 카탈로그에서 센서를 정의하여 차량 내 온도 값을 수신할 수 있습니다. 이 센서는 두 차량의 열전대를 나타내는 데 사용할 수 있습니다. 자세한 내용은 신호 카탈로그 생성 및 관리 섹션을 참조하세요.

차량 모델(모델 매니페스트)

차량 모델은 차량 형식을 표준화하고 차량 내 신호 간의 관계 정의에 사용할 수 있는 선언적 구조입니다. 차량 모델은 동일한 유형의 다중 차량에 일관된 정보를 적용합니다. 신호를 추가하여 차량 모델을 생성합니다. 자세한 내용은 차량 모델 생성 및 관리 섹션을 참조하세요.

디코더 매니페스트

디코더 매니페스트에는 차량 모델의 각 신호에 대한 디코딩 정보가 포함되어 있습니다. 차량의 센서와 액추에이터는 저수준 메시지(바이너리 데이터)를 전송합니다. AWS IoT FleetWise는 디코더 매니페스트를 사용하여 바이너리 데이터를 사람이 읽을 수 있는 값으로 변환할 수 있습니다. 모든 디코더 매니페스트는 차량 모델과 연결됩니다. 자세한 내용은 디코더 매니페스트 생성 및 관리 섹션을 참조하세요.

네트워크 인터페이스

차량 내 네트워크에서 사용하는 프로토콜에 대한 정보가 들어 있습니다. AWS IoT FleetWise는 다음 프로토콜을 지원합니다.

컨트롤러 영역 네트워크(CAN 버스)

전자 제어 장치(ECU) 간 데이터 통신 방식을 정의하는 프로토콜입니다. ECU는 엔진 제어 장치, 에어백 또는 오디오 시스템일 수 있습니다.

온보드 진단(OBD) II

자체 진단 데이터가 ECU 간에 전달되는 방식을 정의하는 추가 개발된 프로토콜입니다. 차량의 문제를 식별하는 데 도움이 되는 여러 표준 진단 문제 코드(DTC)를 제공합니다.

차량 미들웨어

네트워크 인터페이스 유형으로 정의되는 차량 미들웨어입니다. 차량 미들웨어의 예로는 로봇 운영 체제(ROS 2) 및 IP를 통한 확장 가능한 서비스 지향 미들웨어(SOME/IP)가 있습니다.

> **Note**
>
> AWS IoT FleetWise는 비전 시스템 데이터용 ROS 2 미들웨어를 지원합니다.

디코더 신호

특정 신호에 대한 자세한 디코딩 정보를 제공합니다. 차량 모델에 지정된 모든 신호는 디코더 신호와 페어링되어야 합니다. 디코더 매니페스트에 CAN 네트워크 인터페이스가 포함된 경우 CAN 디코더 신호를 포함해야 합니다. 디코더 매니페스트에 OBD 네트워크 인터페이스가 포함된 경우 OBD 디코더 신호를 포함해야 합니다.

디코더 매니페스트에 차량 미들웨어 인터페이스도 포함되어 있는 경우 메시지 디코더 신호가 포함되어야 합니다.

차량

실제 차량(예: 자동차 또는 트럭)을 가상으로 표현한 것입니다. 차량은 차량 모델의 인스턴스입니다. 동일한 차량 모델에서 생성된 차량은 동일한 신호 그룹을 상속받습니다. 각 차량은 AWS IoT 사물에 해당합니다.

플릿

플릿은 차량 그룹을 나타냅니다. 여러 차량을 쉽게 관리하려면 먼저 개별 차량을 플릿에 연결해야 합니다.

Campaign

데이터 수집 체계를 포함합니다. 클라우드에서 캠페인을 정의하고 이를 차량 또는 플릿에 배포합니다. 캠페인은 Edge Agent 소프트웨어 지침에 따라 데이터를 선택하고 수집하고 클라우드로 전송하는 방법을 제공합니다.

데이터 수집 체계

데이터 수집 체계는 Edge Agent 소프트웨어에 데이터 수집 방법에 대한 지침을 제공합니다. 현재 AWS IoT FleetWise는 조건 기반 수집 체계와 시간 기반 수집 체계를 지원합니다.

조건 기반 수집 체계

논리적 표현식을 사용하여 수집할 데이터를 인식합니다. Edge Agent 소프트웨어는 조건이 충족되는 경우 데이터를 수집합니다. 예를 들어, 표현식이 `$variable.myVehicle.InVehicleTemperature >35.0`인 경우 Edge Agent 소프트웨어는 35.0보다 큰 온도 값을 수집합니다.

시간 기반 수집 체계

밀리초 단위로 기간을 지정하여 데이터 수집 주기를 정의합니다. 예를 들어, 기간이 10,000밀리초인 경우 Edge Agent 소프트웨어는 10초마다 한 번씩 데이터를 수집합니다.

AWS IoT FleetWise의 특징

AWS IoT FleetWise의 주요 기능은 다음과 같습니다.

차량 모델링

차량을 가상으로 표현하고 일반적인 형식을 적용하여 차량 신호를 구성합니다. AWS IoT FleetWise는 차량 신호를 표준화하는 데 사용할 수 있는 차량 신호 사양(VSS)을 지원합니다.

체계 기반 데이터 수집

가치가 높은 차량 데이터만 클라우드로 전송하는 방식을 정의하합니다. 40도를 초과하는 차량 내 온도 값 데이터와 같이 수집할 데이터를 제어하는 조건 기반 체계를 정의할 수 있습니다. 데이터 수집 빈도를 제어하는 시간 기반 체계를 정의할 수도 있습니다.

AWS IoT FleetWise 소프트웨어용 Edge Agent

차량에서 실행되는 Edge Agent 소프트웨어는 차량과 클라우드 간의 통신을 용이하게 합니다. 차량이 클라우드에 연결되어 있는 동안 Edge Agent 소프트웨어는 지속적으로 데이터 수집 체계를 수신하고 그에 따라 데이터를 수집합니다.

관련 서비스

AWS IoT FleetWise는 다음 AWS 서비스를 통해 클라우드 솔루션의 가용성 및 확장성을 개선합니다.

- AWS IoT Core – 차량 데이터를 AWS IoT FleetWise에 업로드하는 AWS IoT 장치를 등록하고 제어하세요. 자세한 내용은 AWS IoT 개발자 가이드의 AWS IoT이란 무엇입니까?를 참조하세요.
- Amazon Timestream — 시계열 데이터베이스를 사용하여 차량 데이터를 저장하고 분석합니다. 자세한 내용을 알아보려면 Amazon Timestream 개발자 안내서의 Timestream은 무엇인가를 참조하세요.
- Amazon S3 — 객체 스토리지 서비스를 사용하여 차량 데이터를 저장하고 관리합니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 Amazon S3는 무엇인가를 참조하세요.

AWS IoT FleetWise 설정

AWS IoT FleetWise를 처음 사용한다면 먼저 다음 섹션의 단계를 완료해야 합니다.

주제

AWS 계정 설정

다음 태스크를 완료하여 AWS에 가입하고 관리자를 생성합니다.

AWS 계정에 등록

AWS 계정 항목이 없으면 다음 절차에 따라 생성하세요.

AWS 계정에 가입하려면

1. https://portal.aws.amazon.com/billing/signup을 엽니다.
2. 온라인 지시 사항을 따르세요.

   가입 절차 중 전화를 받고 전화 키패드로 확인 코드를 입력하는 과정이 있습니다.

   AWS 계정에 가입하면 AWS 계정 루트 사용자 항목이 생성됩니다. 루트 사용자에게 계정의 모든 AWS 서비스 및 리소스에 대한 액세스 권한이 있습니다. 보안 모범 사례는 관리 사용자에게 관리자 액세스 권한을 할당하고, 루트 사용자만 루트 사용자 액세스 권한이 필요한 태스크를 수행하는 것입니다.

AWS 항목은 가입 절차 완료된 후 사용자에게 확인 이메일을 전송합니다. 언제든지 https://aws.amazon.com/으로 이동하고 내 계정을 선택하여 현재 계정 활동을 보고 계정을 관리할 수 있습니다.

관리 사용자 생성

AWS 계정에 가입하고, AWS 계정 루트 사용자에 보안 조치를 한 다음, AWS IAM Identity Center를 활성화하여 일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 생성합니다.

귀하의 AWS 계정 루트 사용자 보호

1. 루트 사용자를 선택하고 AWS 계정 이메일 주소를 입력하여 AWS Management Console에 계정 소유자로 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.

   루트 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인 사용 설명서의 Sign in as the root user을 참조하세요.
2. 루트 사용자의 다중 인증(MFA)을 활성화합니다.

   지침은 IAM 사용 설명서의 AWS 계정 루트 사용자용 가상 MFA 디바이스 활성화(콘솔) 섹션을 참조하세요.

관리 사용자 생성

1. IAM Identity Center를 활성화합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 AWS IAM Identity Center 활성화를 참조하세요.
2. IAM Identity Center에서 관리 사용자에게 관리 액세스 권한을 부여합니다.

   IAM Identity Center 디렉터리를 ID 소스로 사용하는 방법에 대한 자습서는 AWS IAM Identity Center 사용 설명서의 기본 IAM Identity Center 디렉터리로 사용자 액세스 구성을 참조하세요.

관리자 사용자로 로그인

- IAM 자격 증명 센터 사용자로 로그인하려면 IAM 자격 증명 센터 사용자를 생성할 때 이메일 주소로 전송된 로그인 URL을 사용합니다.

  IAM Identity Center 사용자로 로그인하는 데 도움이 필요한 경우 AWS 로그인 사용 설명서의 AWS 액세스 포털에 로그인을 참조하세요.

> **Note**
>
> AWS IoT FleetWise와 함께 서비스 연결 역할을 사용할 수 있습니다. 서비스 연결 역할은 AWS IoT FleetWise에 의해 미리 정의되며, AWS IoT FleetWise가 Amazon CloudWatch에 지표를 전송하는 데 필요한 권한을 포함합니다. 자세한 내용은 AWS IoT용 서비스 연결 역할 사용 FleetWise 섹션을 참조하세요.

콘솔에서 시작하기

아직 AWS 계정에 로그인하지 않은 경우 로그인한 다음 AWS IoT FleetWise 콘솔을 엽니다. AWS IoT FleetWise를 시작하려면 차량 모델을 생성합니다. 차량 모델은 차량 형식을 표준화합니다.

1. AWS IoT FleetWise 콘솔로 이동합니다.
2. AWS IoT FleetWise(으)로 시작하기에서, 시작하기를 선택합니다.

차량 모델 생성에 대한 자세한 내용은 차량 모델 생성(콘솔)을(를) 참조하세요.

설정 구성

AWS IoT FleetWise 콘솔 또는 API를 사용하여 Amazon CloudWatch Logs 지표, Amazon CloudWatch Logs에 대한 설정을 구성하고 AWS 관리형 키을(를) 사용하여 데이터를 암호화할 수 있습니다.

CloudWatch 지표를 사용하면 AWS IoT FleetWise와 기타 AWS 리소스를 모니터링할 수 있습니다. CloudWatch 지표를 사용하여 서비스 한도 초과 여부를 확인하는 등 지표를 수집하고 추적할 수 있습니다. CloudWatch 지표에 대한 자세한 내용은 Amazon CloudWatch로 AWS IoT FleetWise 모니터링 섹션을 참조하세요.

CloudWatch Logs를 사용하면 AWS IoT FleetWise가 로그 데이터를 CloudWatch 로그 그룹으로 전송하며, 사용자는 이 로그를 사용하여 문제를 식별하고 완화할 수 있습니다. CloudWatch Logs에 대한 자세한 내용은 AWS IoT FleetWise 로깅 구성 섹션을 참조하세요.

데이터 암호화를 통해 AWS IoT FleetWise는 AWS 관리형 키을(를) 사용하여 데이터를 암호화합니다. AWS KMS을(를) 사용하여 키를 만들고 관리하도록 선택할 수도 있습니다. 암호화에 대한 자세한 내용은 데이터 암호화 섹션을 참조하세요.

설정 구성(콘솔)

아직 AWS 계정에 로그인하지 않은 경우 로그인한 다음 AWS IoT FleetWise 콘솔을 엽니다.

1. AWS IoT FleetWise 콘솔로 이동합니다.
2. 왼쪽 창에서 설정을 선택합니다.
3. 지표에서 활성화를 선택합니다. AWS IoT FleetWise는 CloudWatch 관리형 정책을 서비스 연결 역할에 자동으로 연결하고 CloudWatch 지표를 활성화합니다.
4. 로깅에서 편집을 선택합니다.

a. CloudWatch 로깅섹션에서 로그 그룹을 입력합니다.

b. 변경 사항을 저장하려면, 제출을 선택합니다.

5. 암호화 섹션에서 편집을 선택합니다.

a. 사용하려는 키 타입을 선택합니다. 자세한 내용은 키 관리 섹션을 참조하세요.

i. AWS키 사용 — AWS IoT FleetWise는 키를 소유하고 관리합니다.

ii. 다른 AWS Key Management Service 키 선택 — 계정에 있는 AWS KMS keys을(를) 관리합니다.

b. 변경 사항을 저장하려면, 제출을 선택합니다.

설정 구성(AWS CLI)

AWS CLI에서 계정을 등록하여 설정을 구성합니다.

1. 설정을 구성하려면, 다음 명령을 실행합니다.

```
aws iotfleetwise register-account
```

2. 설정을 확인하려면 다음 명령을 실행하여 등록 상태를 검색합니다.

> **Note**
>
> 서비스 연결 역할은 AWS IoT FleetWise 지표를 CloudWatch에 게시하는 데만 사용됩니다. 자세한 내용은 AWS IoT용 서비스 연결 역할 사용 FleetWise 섹션을 참조하세요.

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

Example 응답

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

등록 상태는 다음 중 하나일 수 있습니다.

- REGISTRATION_SUCCESS – AWS 리소스가 성공적으로 등록되었습니다.
- REGISTRATION_PENDING – AWS IoT FleetWise가 등록 요청을 처리하고 있습니다. 이 프로세스를 완료하는 데 5분가량 소요됩니다.
- REGISTRATION_FAILURE – AWS IoT FleetWise는 AWS 리소스를 등록할 수 없습니다. 나중에 다시 시도해 주세요.

AWS IoT 시작하기 FleetWise

AWS FleetWiseIoT를 사용하면 차량 데이터를 수집, 변환 및 전송할 수 있습니다. 이 섹션의 자습서를 사용하여 AWS FleetWise IoT를 시작하세요.

AWS IoT에 대해 자세히 알아보려면 다음 항목을 참조하십시오 FleetWise.

- 클라우드로 데이터 수집
- 차량 모델링
- 차량 생성, 공급 및 관리
- 플릿 생성 및 관리
- 캠페인을 통한 데이터 수집 및 전송

요구 사항

AWS 계정 AWS IoT를 시작하려면 이 있어야 FleetWise 합니다. 계정이 없는 경우 AWS IoT FleetWise 설정 단원을 참조하십시오.

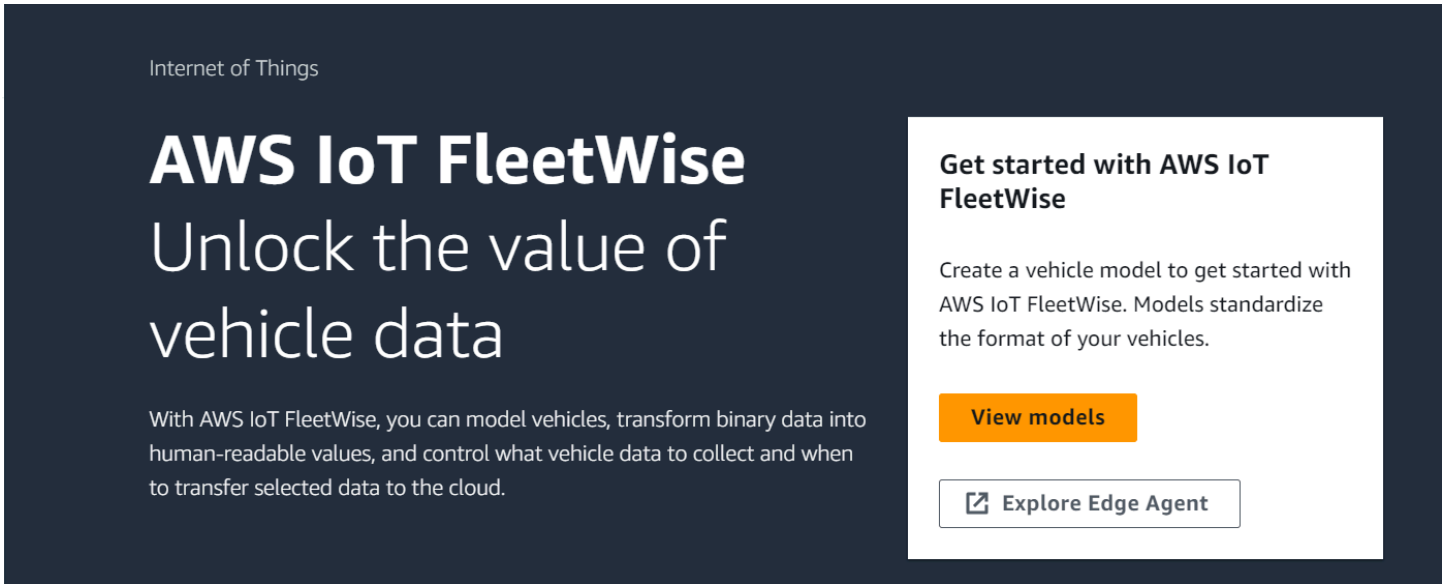
AWS FleetWise IoT를 사용할 수 있는 지역을 사용하십시오. 자세한 내용은 AWS IoT FleetWise 엔드포인트 및 할당량을 참조하십시오. 의 지역 선택기를 사용하여 이러한 지역 중 하나로 AWS Management Console 전환할 수 있습니다.

엣지 에이전트 소프트웨어 데모

Explore Edge Agent 퀵 스타트 데모를 사용하여 IoT를 FleetWise 살펴보고 AWS IoT용 AWS Edge Agent 소프트웨어를 개발하는 방법을 배울 수 FleetWise 있습니다. 이 데모에서는 AWS CloudFormation 템플릿을 사용합니다. Edge Agent 참조 구현을 검토하고, Edge Agent를 개발한 다음, Amazon EC2 Graviton에 Edge Agent 소프트웨어를 배포하고 샘플 차량 데이터를 생성하는 과정을 안내합니다. 또한 데모는 클라우드에서 신호 카탈로그, 차량 모델, 디코더 매니페스트, 차량, 플릿 및 캠페인 생성에 사용할 수 있는 스크립트를 제공합니다. 퀵 스타트 데모에 대한 자세한 내용을 보려면 다음을 수행하여 Edge Agent 소프트웨어 개발자 가이드를 다운로드합니다.

퀵 스타트 데모를 다운로드하려면

1. AWS IoT FleetWise 콘솔로 이동합니다.
2. 서비스 홈 페이지의 AWS IoT 시작하기 FleetWise 섹션에서 Edge Agent 탐색을 선택합니다.



튜토리얼: AWS IoT 시작하기 FleetWise (콘솔)

AWS FleetWise IoT를 사용하여 자율주행차의 고유한 데이터 형식을 거의 실시간으로 수집, 변환하고 클라우드로 전송합니다. 플릿 전반에 걸친 인사이트에 액세스할 수 있습니다. 이를 통해 차량 상태 문제를 효율적으로 감지하고 해결하며, 고부가가치 데이터 신호를 전송하고 원격으로 문제를 진단할 수 있으며, 동시에 이 모든 과정에서 비용을 절감할 수 있습니다.

이 자습서에서는 AWS IoT를 시작하는 방법을 보여줍니다 FleetWise. 차량 모델(모델 매니페스트), 디코더 매니페스트, 차량 및 캠페인을 만드는 방법을 배우게 됩니다.

AWS FleetWiseIoT의 주요 구성 요소 및 개념에 대한 자세한 내용은 을 참조하십시오AWS IoT FleetWise 작동 방식.

예상 소요 시간: 45분.

> **Important**
>
> 이 데모에서 생성하고 소비하는 AWS IoT FleetWise 리소스에 대한 요금이 부과됩니다. 자세한 내용은 AWS IoT FleetWise FleetWise 요금 페이지에서AWS IoT를 참조하십시오.

주제

- 필수 조건
- 1단계: AWS IoT용 에지 에이전트 소프트웨어 설정 FleetWise

필수 조건

이 시작하기 튜토리얼을 완료하려면 다음이 필요합니다.

- An AWS 계정. 계정이 AWS 계정없는 경우 AWS Account Management 참조 안내서의 만들기를 참조하십시오. AWS 계정
- AWS IoT를 AWS 리전 지원하는 네트워크에 FleetWise 액세스하십시오. 현재 AWS FleetWise IoT는 미국 동부 (버지니아 북부) 와 유럽 (프랑크푸르트) 에서 지원됩니다.
- Amazon Timestream 리소스:
  - Amazon Timestream 데이터베이스. 자세한 내용은 Amazon Timestream 개발자 안내서의 데이터베이스 생성을 참조하세요.
  - Amazon Timestream에서 생성한 Amazon Timestream 테이블로, 데이터를 보관합니다. 자세한 내용은 Amazon Timestream 개발자 가이드의 테이블 생성을 참조하세요.

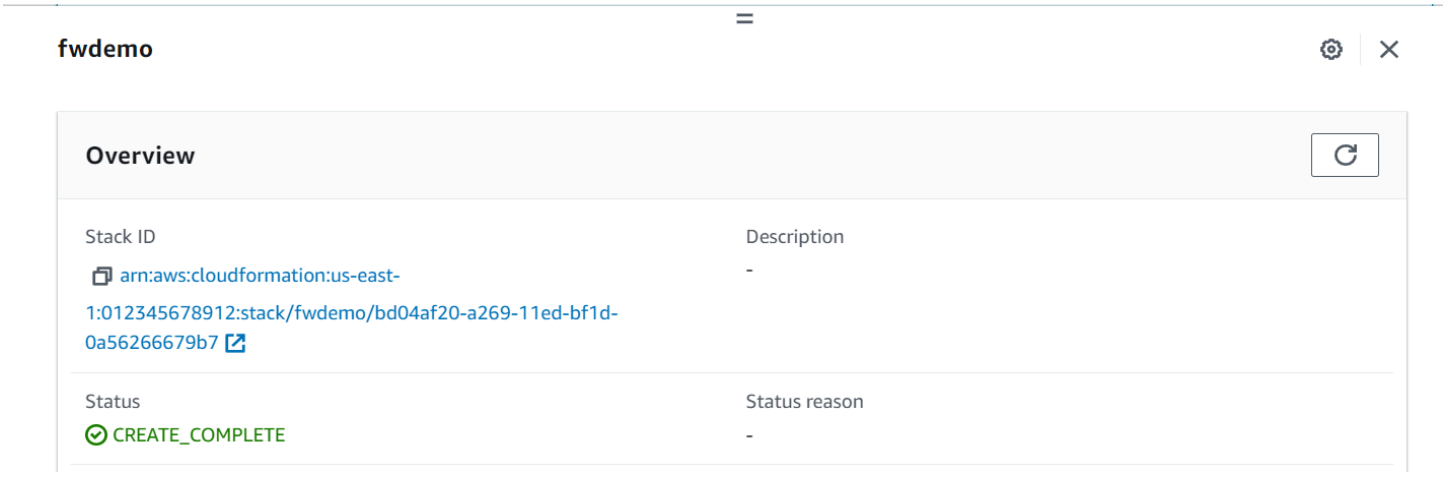
1단계: AWS IoT용 에지 에이전트 소프트웨어 설정 FleetWise

> **Note**
>
> 이 단계의 CloudFormation 스택은 텔레메트리 데이터를 사용합니다. 비전 시스템 데이터를 사용하여 CloudFormation 스택을 만들 수도 있습니다. 자세한 내용은 Vision System Data Developer Guide를 참조하세요.
> 비전 시스템 데이터는 평가판 릴리스이며 변경될 수 있습니다.

AWS IoT용 Edge Agent 소프트웨어는 차량과 클라우드 간의 통신을 FleetWise 용이하게 합니다. 데이터 수집 체계로부터 클라우드 연결 차량에서 데이터를 수집하는 방법에 대한 지침을 받습니다.

Edge Agent 소프트웨어를 설정하려면 일반 정보에서 다음을 수행하세요.

1. 시작 CloudFormation 템플릿을 엽니다.
2. 빠른 스택 생성 페이지에서 스택 이름에 AWS IoT FleetWise 리소스 스택의 이름을 입력합니다. 스택은 AWS CloudFormation 템플릿이 생성하는 리소스 이름에 접두사로 표시되는 친숙한 이름입니다.
3. 매개 변수에서 스택과 관련된 매개 변수의 사용자 지정 값을 입력합니다.
   a. Fleetsize - Fleetsize 파라미터를 업데이트하여 플릿의 차량 수를 늘릴 수 있습니다.
   b. IoT CoreRegion - IoT CoreRegion 매개변수를 업데이트하여 AWS IoT 사물이 생성되는 지역을 지정할 수 있습니다. AWS IoT FleetWise 차량을 만들 때 사용한 것과 동일한 지역을 사용해야 합니다. 에 대한 AWS 리전자세한 내용은 지역 및 영역 - Amazon Elastic Compute 클라우드를 참조하십시오.
4. 기능 섹션에서 IAM 리소스 생성을 확인하는 AWS CloudFormation 상자를 선택합니다.
5. 스택 생성을 선택한 다음 스택 상태가 CREATE_COMPLETE로 표시될 때까지 약 15분 정도 기다립니다.
6. 스택이 생성되었는지 확인하려면 스택 정보 탭을 선택하고 뷰를 새로 고친 다음 CREATE_COMPLETE를 찾으세요.



Important

이 데모에서 생성하고 소비하는 AWS IoT FleetWise 리소스에 대한 요금이 부과됩니다. 자세한 내용은 AWS IoT FleetWise FleetWise 요금 페이지에서AWS IoT를 참조하십시오.

2단계: 차량 모델 생성

> **Important**
>
> AWS IoT FleetWise 콘솔에서는 비전 시스템 데이터 신호를 사용하여 차량 모델을 만들 수 없습니다. 그 대신 AWS CLI을 사용합니다.

차량 모델을 사용하여 차량의 형식을 표준화할 수 있으며 생성되는 차량 신호 간의 관계를 정의할 수 있습니다. 신호 카탈로그는 차량 모델을 생성할 때도 생성됩니다. 차량 모델 생성에 재사용할 수 있는 표준화된 신호 컬렉션을 생성합니다. 신호는 차량 데이터와 해당 메타데이터를 포함하도록 정의하는 기본 구조입니다. 현재 AWS IoT FleetWise 서비스는 AWS 리전 계정당 하나의 신호 카탈로그만 지원합니다. 이를 통해 차량 플릿에서 처리된 데이터를 일관되게 유지할 수 있습니다.

차량를 생성하는 방법

1. AWS IoT FleetWise 콘솔을 엽니다.
2. 기본 탐색 창에서 차량 모델을 선택합니다.
3. 모델 페이지에서 모델 생성을 선택합니다.
4. 일반 정보 섹션에서 차량 모델 이름(예: Vehicle1)과 선택적 설명을 입력합니다. 다음을 선택합니다.
5. 신호 카탈로그에서 하나 이상의 신호를 선택합니다. 검색 카탈로그에서 이름을 기준으로 신호를 필터링하거나 목록에서 신호를 선택할 수 있습니다. 예를 들어, 타이어 압력과 브레이크 압력에 대한 신호를 선택하여 이러한 신호와 관련된 데이터를 수집할 수 있습니다. 다음을 선택합니다.
6. .dbc 파일을 선택하고 로컬 장치에서 업로드하세요. 다음을 선택합니다.

   > **Note**
   >
   > 이 자습서에서는 이 단계에서 업로드할 샘플.dbc 파일을 다운로드할 수 있습니다.

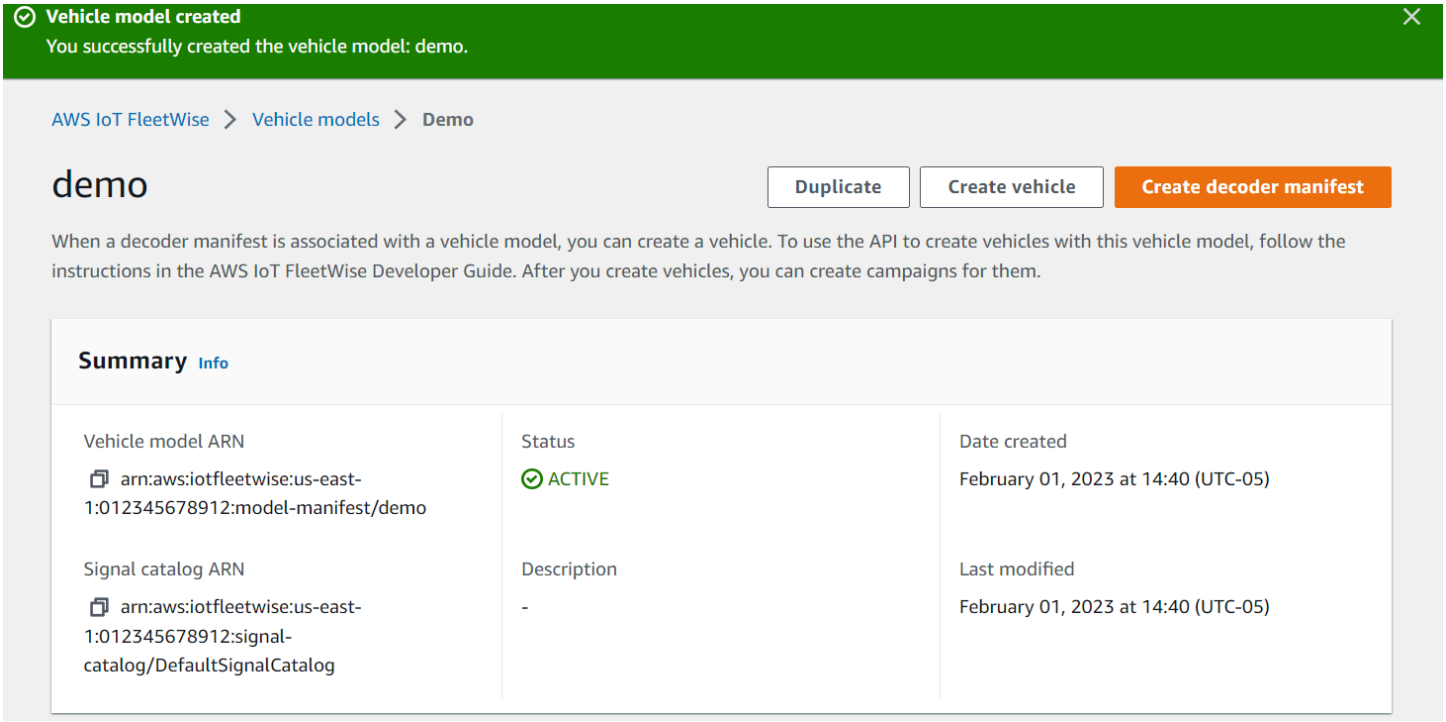
7. 차량 모델에 속성을 추가한 후 다음을 선택합니다.

   a. 이름 - 제조업체 이름 또는 제조 날짜와 같은 차량 속성의 이름을 입력합니다.
   b. 데이터 유형 - 데이터 유형 메뉴에서 데이터 유형을 선택합니다.
   c. 단위 - (선택 사항) 킬로미터 또는 섭씨와 같은 단위 값을 입력합니다.
   d. 경로 - (선택 사항) 신호 경로 이름(예: `Vehicle.Engine.Light.`)을 입력합니다. 점(.)은 신호가 하위 신호임을 나타냅니다.

e. 기본값 - (선택 사항) 기본값을 입력합니다.

f. 설명 - (선택 사항) 속성에 대한 설명을 입력합니다.

8. 구성을 검토합니다. 준비가 되었으면 생성을 선택합니다. 차량 모델이 성공적으로 생성되었다는 알림이 나타납니다.



3단계: 디코더 매니페스트 생성

디코더 매니페스트는 생성한 차량 모델과 연결됩니다. 여기에는 AWS IoT가 바이너리 형식의 차량 데이터를 FleetWise 디코딩하고 분석할 수 있는 사람이 읽을 수 있는 값으로 변환하는 데 도움이 되는 정보가 포함되어 있습니다. 네트워크 인터페이스와 디코더 신호는 디코더 매니페스트를 구성하는 데 도움이 되는 구성 요소입니다. 네트워크 인터페이스에는 차량 네트워크에서 사용하는 CAN 또는 OBD 프로토콜에 대한 정보가 포함됩니다. 디코더 신호는 특정 신호에 대한 디코딩 정보를 제공합니다.

디코더 매니페스트를 생성하려는 경우

1. AWS IoT FleetWise 콘솔을 엽니다.
2. 기본 탐색 창에서 차량 모델을 선택합니다.
3. 차량 모델 섹션에서 디코더 매니페스트 생성에 사용할 차량 모델을 선택합니다.
4. 디코더 매니페스트 생성을 선택합니다.

4단계: 디코더 매니페스트 구성

디코더 매니페스트를 구성하려는 경우

> **Important**
>
> AWS IoT FleetWise 콘솔을 사용하여 디코더 매니페스트의 비전 시스템 데이터 신호를 구성할 수 없습니다. 그 대신 AWS CLI을 사용합니다. 자세한 정보는 디코더 매니페스트 만들기(AWS CLI)을 참조하세요.

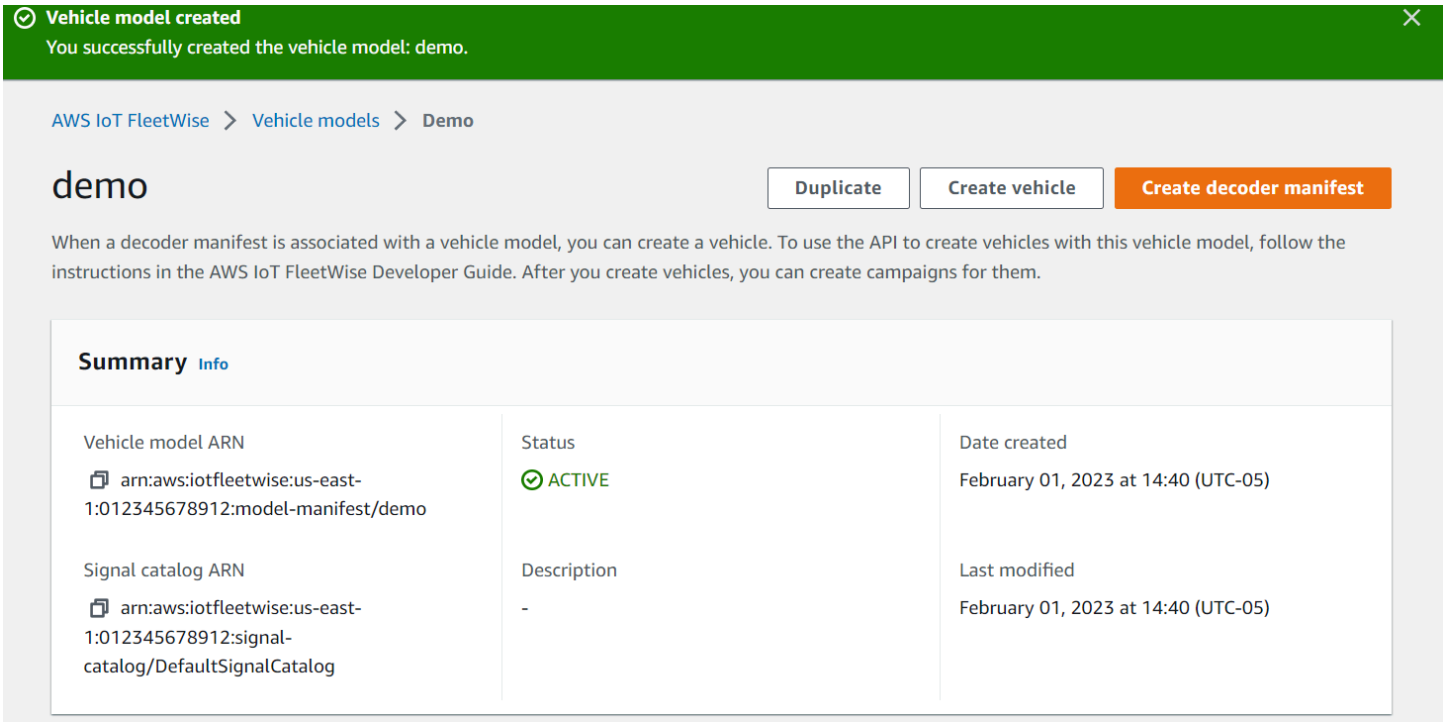
1. 디코더 매니페스트를 식별하는 데 도움이 되도록 이름과 설명(선택 사항)을 입력합니다. 그리고 다음을 선택합니다.
2. 네트워크 인터페이스를 하나 이상 추가하려면 CAN_INTERFACE 또는 OBD_INTERFACE 유형을 선택합니다.
   - 온보드 진단(OBD) 인터페이스 - 전자 제어 장치(ECU) 간에 자가 진단 데이터가 통신되는 방식을 정의하는 프로토콜이 필요한 경우 이 인터페이스 유형을 선택합니다. 이 프로토콜은 차량 문제 해결에 도움이 되는 여러 표준 진단 문제 코드(DTC)를 제공합니다.
   - 컨트롤러 영역 네트워크(CAN 버스) 인터페이스 - ECU 간 데이터 통신 방식을 정의하는 프로토콜을 원하는 경우 이 인터페이스 유형을 선택하세요. ECU는 엔진 제어 장치, 에어백 또는 오디오 시스템일 수 있습니다.
3. 네트워크 인터페이스 이름을 입력합니다.
4. 네트워크 인터페이스에 신호를 추가하려면 목록에서 하나 이상의 신호를 선택합니다.
5. 이전 단계에서 추가한 신호에 사용할 디코더 신호를 선택합니다. 디코딩 정보를 제공하려면.dbc 파일을 업로드하세요. 차량 모델의 각 신호는 목록에서 선택할 수 있는 디코더 신호와 페어링되어야 합니다.
6. 보조 네트워크 인터페이스를 추가하려면 네트워크 인터페이스 추가를 선택합니다. 네트워크 인터페이스를 모두 추가했으면 다음을 선택합니다.
7. 구성을 살펴본 후 생성을 선택합니다. 디코더 매니페스트가 성공적으로 생성되었다는 알림이 나타납니다.

5단계: 차량 생성

AWS FleetWiseIoT에서 차량은 실제 차량을 가상으로 표현한 것입니다. 동일한 차량 모델에서 생성된 모든 차량은 동일한 신호 그룹을 상속하며, 생성한 각 차량은 새로 생성된 IoT 사물에 해당합니다. 모든 차량을 디코더 매니페스트에 연결해야 합니다.

필수 조건

1. 차량 모델 및 디코더 매니페스트를 이미 생성했는지 확인하세요. 또한 차량 모델의 상태가 ACTIVE인지 확인하세요.
   a. 차량 모델의 상태가 ACTIVE인지 확인하려면 AWS IoT FleetWise 콘솔을 엽니다.
   b. 기본 탐색 창에서 차량 모델을 선택합니다.
   c. 요약 섹션의 상태에서 차량 상태를 확인합니다.



차량을 생성하려는 경우

1. AWS FleetWise 콘솔을 엽니다.
2. 탐색 창에서 차량을 선택합니다.
3. 차량 생성을 선택합니다.

4. 차량 속성을 정의하려면 차량 이름을 입력한 다음 모델 매니페스트(차량 모델)와 디코더 매니페스트를 선택합니다.
5. (선택 사항) 차량 속성을 정의하려면 키-값 쌍을 입력한 다음 속성 추가를 선택합니다.
6. (선택 사항) AWS 리소스에 레이블을 지정하려면 태그를 추가한 다음 새 태그 추가를 선택합니다.
7. 다음을 선택합니다.
8. 차량 인증서를 구성하려면 자체 인증서를 업로드하거나 신규 인증서 자동 생성을 선택할 수 있습니다. 더 빠르게 설정하려면 인증서를 자동 생성하는 것이 좋습니다. 이미 인증서가 있는 경우 해당 인증서를 대신 사용하도록 선택할 수 있습니다.
9. 공개 및 개인 키 파일을 다운로드한 후 다음을 선택합니다.
10. 정책을 차량 인증서에 첨부하려면 기존 정책 이름을 입력하거나 새 정책을 생성할 수 있습니다. 새 정책을 생성하려면 정책 생성을 선택한 후 다음을 선택합니다.
11. 구성을 검토합니다. 완료했으면 차량 생성을 선택합니다.

6단계: 캠페인 생성

AWS FleetWiseIoT에서 캠페인은 차량에서 클라우드로 데이터를 쉽게 선택, 수집 및 전송하는 데 사용됩니다. 캠페인에는 데이터 수집 체계가 포함되어 있으며 이는 Edge Agent 소프트웨어에 조건 기반 수집 체계 또는 시간 기반 수집 체계로 데이터를 수집하는 방법에 대한 지침을 제공합니다.

캠페인을 생성하려는 경우

1. AWS IoT FleetWise 콘솔을 엽니다.
2. 탐색 창에서 캠페인을 선택합니다.
3. 캠페인 생성을 선택합니다.
4. 캠페인에 대한 이름과 선택 사항 설명을 입력합니다.
5. 캠페인의 데이터 수집 체계를 구성하려면 데이터 수집 체계를 수동으로 정의하거나 로컬 디바이스에서 .json 파일을 업로드합니다. .json 파일을 업로드하면 데이터 수집 체계가 자동으로 정의됩니다.

    a. 데이터 수집 체계를 수동으로 정의하려면 데이터 수집 체계 정의를 선택하고 캠페인에 사용할 데이터 수집 체계 유형을 선택합니다. 조건 기반 수집 체계 또는 시간 기반 수집 체계를 선택할 수 있습니다.

    b. 시간 기반 수집 체계를 선택하는 경우 캠페인에서 차량 데이터를 수집하는 기간을 지정해야 합니다.

c. 조건 기반 수집 체계를 선택하는 경우 수집할 데이터를 인식하기 위한 표현식을 지정해야 합니다. 신호 이름을 변수, 비교 연산자 및 비교 값으로 지정해야 합니다.

d. (선택 사항) 표현식의 언어 버전을 선택하거나 기본값인 1로 유지합니다.

e. (선택 사항) 두 데이터 수집 이벤트 사이의 트리거 간격을 지정합니다.

f. 데이터를 수집하려면 Edge Agent 소프트웨어의 트리거 모드 조건을 선택합니다. 기본적으로 AWS IoT용 Edge Agent FleetWise 소프트웨어는 조건이 충족될 때마다 항상 데이터를 수집합니다. 또는 조건이 처음으로 충족되는 경우, 즉 첫 번째 트리거 시에만 데이터를 수집할 수 있습니다.

g. (선택 사항) 고급 체계 옵션을 더 선택할 수 있습니다.

6. 데이터 수집 체계에서 데이터를 수집할 신호를 지정하려면 메뉴에서 신호 이름을 검색합니다.
7. (선택 사항) 최대 샘플 수 또는 최소 샘플링 간격을 선택할 수 있습니다. 또한 더 많은 신호를 추가할 수 있습니다.
8. 다음을 선택합니다.
9. 캠페인에서 데이터를 전송할 저장 대상을 정의합니다. 데이터를 Amazon S3 또는 Amazon Timestream에 저장할 수 있습니다.

   a. Amazon S3 — AWS IoT FleetWise 권한이 있는 S3 버킷을 선택합니다.

   b. Amazon Timestream — Timestream 데이터베이스 및 테이블 이름을 선택합니다. 타임스트림으로 데이터를 AWS IoT FleetWise 전송할 수 있는 IAM 역할을 입력합니다.

10. 다음을 선택합니다.
11. 검색 상자에서 차량 속성 또는 차량 이름을 선택합니다.
12. 차량에 대해 선택한 속성 또는 이름과 관련된 값을 입력합니다.
13. 캠페인에서 데이터를 수집할 차량을 선택합니다. 그리고 다음을 선택합니다.
14. 캠페인 구성을 검토한 다음 캠페인 생성을 선택합니다. 사용자 또는 팀이 차량에 캠페인을 배포해야 합니다.

7단계: 정리

이 자습서에서 사용한 리소스에 대한 추가 요금이 부과되지 않도록 하려면 AWS CloudFormation 스택과 모든 스택 리소스를 삭제하십시오.

AWS CloudFormation 스택을 삭제하려면

1. AWS CloudFormation 콘솔을 엽니다.

2. 스택 목록에서 1단계에서 만든 스택을 선택합니다.
3. 삭제를 선택합니다.
4. 삭제를 확인하려면 삭제를 선택합니다. 스택을 삭제하는 데 약 15분이 걸립니다.

다음 단계

1. 캠페인에서 수집하는 차량 데이터를 처리하고 시각화할 수 있습니다. 자세한 정보는 차량 데이터 처리 및 시각화을 참조하세요.
2. AWS FleetWiseIoT와 관련된 문제를 해결하고 해결할 수 있습니다. 자세한 내용은 AWS IoT FleetWise 문제 해결을(를) 참조하세요.

클라우드로 데이터 수집

AWS IoT FleetWise 소프트웨어용 Edge Agent는 차량에 설치 및 실행 시 차량과 클라우드 간의 보안 통신을 촉진하도록 설계되었습니다.

> **Note**
>
> - AWS IoT FleetWise는 심각한 신체적 부상이나 사망을 초래하거나 환경 또는 재산상의 피해를 야기할 수 있는 위험한 환경 또는 중요 시스템의 작동에 사용하거나 이와 연계하여 사용하도록 설계되지 않았습니다. AWS IoT FleetWise를 사용하여 수집된 차량 데이터는 정보 제공의 목적으로만 사용되며, 차량 기능을 제어하거나 운영하기 위해 AWS IoT FleetWise를 사용할 수 없습니다.
> - AWS IoT FleetWise를 사용하여 수집된 차량 데이터는 해당 차량 안전 규정에 따른 규정 준수 의무(예: 안전 모니터링 및 보고 의무)를 충족하기 위한 목적을 포함하여 사용 사례에 적합한 정확성을 평가해야 합니다. 이러한 평가에는 다른 업계 표준 수단 및 출처(예: 차량 운전자 보고서)를 통한 정보 수집 및 검토가 포함되어야 합니다.

클라우드로 데이터를 인제스트하려면 다음을 수행합니다.

1. AWS IoT FleetWise 소프트웨어용 Edge Agent를 개발하고 차량에 설치합니다. Edge Agent 소프트웨어를 사용하는 방법에 대한 자세한 내용은 AWS IoT FleetWise용 Edge Agent 소프트웨어 개발자 안내서를 다운로드하여 다음을 수행합니다.
   1. AWS IoT FleetWise 콘솔로 이동합니다.
   2. 서비스 홈 페이지의 AWS IoT FleetWise와 시작하기 섹션에서 Edge Agent 탐색을 선택합니다.
2. 차량 모델 생성에 사용할 신호를 포함하고 있는 신호 카탈로그를 생성하거나 가져오세요. 자세한 정보는 신호 카탈로그 생성(AWS CLI) 및 신호 카탈로그 가져오기(AWS CLI) 섹션을 참조하십시오.

> **Note**
>
> - AWS IoT FleetWise 콘솔을 사용하여 첫 번째 차량 모델을 만드는 경우 신호 카탈로그를 수동으로 생성할 필요가 없습니다. 첫 번째 차량 모델을 만들면 AWS IoT FleetWise가 자동으로 신호 카탈로그를 생성합니다. 자세한 내용은 차량 모델 생성(콘솔) 섹션을 참조하세요.

- AWS IoT FleetWise는 현재 AWS 리전별 각 AWS 계정에 대한 신호 카탈로그를 지원합니다.

3. 신호 카탈로그의 신호를 사용하여 차량 모델을 생성하세요. 자세한 내용은 차량용 모델 생성 섹션을 참조하세요.

> **Note**
>
> - AWS IoT FleetWise 콘솔을 사용하여 차량 모델을 생성하는 경우, .dbc 파일을 업로드하여 신호를 가져올 수 있습니다. .dbc는 컨트롤러 영역 네트워크(CAN 버스) 데이터베이스가 지원하는 파일 형식입니다. 차량 모델이 생성되면 새 신호가 신호 카탈로그에 자동으로 추가됩니다. 자세한 내용은 차량 모델 생성(콘솔) 섹션을 참조하세요.
> - `CreateModelManifest`API 작업을 사용하여 차량 모델을 생성하는 경우 `UpdateModelManifest` API 작업을 사용하여 차량 모델을 활성화해야 합니다. 자세한 내용은 차량 모델 업데이트(AWS CLI) 섹션을 참조하세요.
> - AWS IoT FleetWise 콘솔을 사용하여 차량 모델을 생성하는 경우, AWS IoT FleetWise가 자동으로 차량 모델을 활성화합니다.

4. 디코더 매니페스트를 생성합니다. 디코더 매니페스트에는 차량 모델(이전 단계에서 생성)에 명시된 모든 신호의 디코딩 정보가 포함되어 있습니다. 디코더 매니페스트는 생성한 차량 모델과 연결됩니다. 자세한 내용은 디코더 매니페스트 생성 및 관리 섹션을 참조하세요.

> **Note**
>
> - `CreateDecoderManifest` API 작업을 사용하여 디코더 매니페스트를 만드는 경우 `UpdateDecoderManifest` API 작업을 사용하여 디코더 매니페스트를 활성화해야 합니다. 자세한 내용은 디코더 매니페스트 업데이트(AWS CLI) 섹션을 참조하세요.
> - AWS IoT FleetWise 콘솔을 사용하여 디코더 매니페스트를 생성하면 AWS IoT FleetWise가 자동으로 디코더 매니페스트를 활성화합니다.

5. 차량 모델에서 차량을 만듭니다. 동일한 차량 모델에서 생성된 차량은 동일한 신호 그룹을 상속받습니다. 데이터를 클라우드로 인제스트하려면 먼저 차량을 프로비저닝하기 위해 AWS IoT Core을(를) 사용해야 합니다. 자세한 내용은 차량 생성, 공급 및 관리 섹션을 참조하세요.
6. (선택 사항) 차량 그룹을 나타내는 플릿을 생성한 다음 개별 차량을 플릿과 연결합니다. 이를 통해 동시에 여러 차량을 관리할 수 있습니다. 자세한 내용은 플릿 생성 및 관리 섹션을 참조하세요.

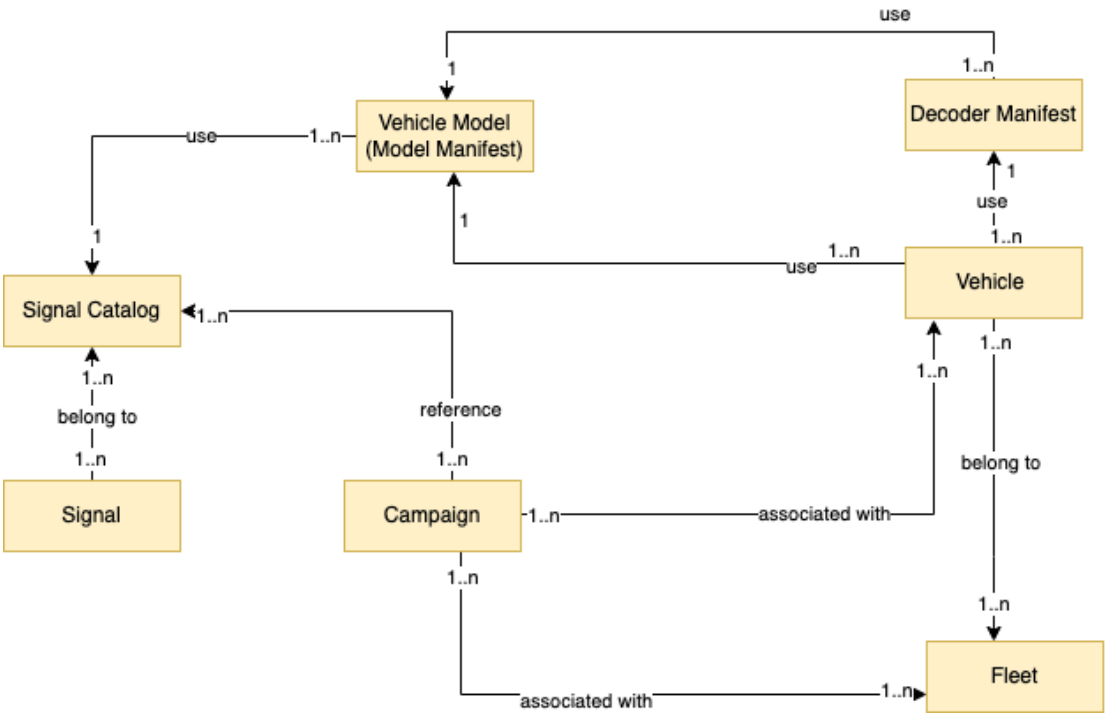
7. 캠페인을 생성합니다. 캠페인은 차량 또는 여러 차량의 플릿에 배포됩니다. 캠페인은 Edge Agent 소프트웨어에서 데이터를 선택하고 수집하고 클라우드로 전송하는 방법에 대한 지침을 제공합니다. 자세한 내용은 캠페인을 통한 데이터 수집 및 전송 섹션을 참조하세요.

> **Note**
>
> AWS IoT FleetWise가 차량 또는 플릿에 캠페인을 배포하려면 먼저 `UpdateCampaign` API 작업을 사용하여 캠페인을 승인해야 합니다. 자세한 내용은 캠페인 업데이트 (AWS CLI) 섹션을 참조하세요.

Edge Agent 소프트웨어는 예약된 주제 `$aws/iotfleetwise/vehicles/vehicleName/signals`을(를) 사용하여 차량 데이터를 AWS IoT Core(으)로 전송하고, 이는 AWS IoT FleetWise로 데이터를 전송합니다. AWS 그런 다음 IoT FleetWise는 데이터를 Timestream 테이블 또는 Amazon S3 버킷으로 전송합니다. Timestream을 사용하여 데이터를 쿼리하고 Amazon QuickSight 또는 Grafana를 사용하여 데이터를 시각화할 수 있습니다. 자세한 내용은 차량 데이터 처리 및 시각화 섹션을 참조하세요.

차량 모델링

AWS IoT는 클라우드에서 차량을 가상으로 표현하는 데 사용할 수 있는 차량 모델링 프레임워크를 FleetWise 제공합니다. 신호, 신호 카탈로그, 차량 모델 및 디코더 매니페스트는 차량 모델링에 사용하는 핵심 구성 요소입니다.



신호

신호는 차량 데이터와 해당 메타데이터를 포함하도록 정의하는 기본 구조입니다. 신호는 속성, 분기, 센서 또는 액추에이터일 수 있습니다. 예를 들어, 차량 내 온도 값을 수신하고 센서 이름, 데이터 유형 및 단위를 포함한 메타데이터를 저장하는 센서를 생성할 수 있습니다. 자세한 정보는 신호 카탈로그 생성 및 관리을 참조하세요.

신호 카탈로그

신호 카탈로그에는 신호 컬렉션이 포함되어 있습니다. 신호 카탈로그에 있는 신호는 다양한 프로토콜과 데이터 형식을 사용하는 차량을 모델링할 때 사용할 수 있습니다. 예를 들어, 서로 다른 자동차 제조업체에서 만든 두 대의 자동차가 있습니다. 하나는 제어 영역 네트워크(CAN 버스) 프로토콜을 사용하고 다른 하나는 OBD(온보드 진단) 프로토콜을 사용합니다. 신호 카탈로그에서 센서를 정의하여 차량 내 온도 값을 수신할 수 있습니다. 이 센서는 두 차량의 열전대를 나타내는 데 사용할 수 있습니다. 자세한 정보는 신호 카탈로그 생성 및 관리을 참조하세요.

차량 모델(모델 매니페스트)

차량 모델은 차량 형식을 표준화하고 차량 내 신호 간의 관계 정의에 사용할 수 있는 선언적 구조입니다. 차량 모델은 동일한 유형의 다중 차량에 일관된 정보를 적용합니다. 신호를 추가하여 차량 모델을 생성합니다. 자세한 정보는 차량 모델 생성 및 관리을 참조하세요.

디코더 매니페스트

디코더 매니페스트에는 차량 모델의 각 신호에 대한 디코딩 정보가 포함되어 있습니다. 차량의 센서와 액추에이터는 저수준 메시지(바이너리 데이터)를 전송합니다. AWS FleetWise IoT는 디코더 매니페스트를 통해 바이너리 데이터를 사람이 읽을 수 있는 값으로 변환할 수 있습니다. 모든 디코더 매니페스트는 차량 모델과 연결됩니다. 자세한 정보는 디코더 매니페스트 생성 및 관리을 참조하세요.

AWS IoT FleetWise 콘솔 또는 API를 사용하여 다음과 같은 방식으로 차량을 모델링할 수 있습니다.

1. 차량 모델을 만드는 데 사용할 신호가 포함된 신호 카탈로그를 생성하거나 가져옵니다. 자세한 내용은 신호 카탈로그 생성(AWS CLI) 및 신호 카탈로그 가져오기(AWS CLI) 섹션을 참조하세요.

   > **Note**
   >
   > - AWS IoT FleetWise 콘솔을 사용하여 첫 번째 차량 모델을 생성하는 경우 신호 카탈로그를 수동으로 생성할 필요가 없습니다. 첫 번째 차량 모델을 만들면 AWS IoT가 FleetWise 자동으로 신호 카탈로그를 생성합니다. 자세한 정보는 차량 모델 생성(콘솔)을 참조하세요.
   > - AWS IoT는 FleetWise 현재 AWS 계정별로 신호 카탈로그를 지원합니다 AWS 리전.

2. 신호 카탈로그의 신호를 사용하여 차량 모델을 생성하세요. 자세한 정보는 차량용 모델 생성을 참조하세요.

   > **Note**
   >
   > - AWS IoT FleetWise 콘솔을 사용하여 차량 모델을 생성하는 경우.dbc 파일을 업로드하여 신호를 가져올 수 있습니다.. dbc는 컨트롤러 영역 네트워크 (CAN 버스) 데이터베이스가 지원하는 파일 형식입니다. 차량 모델이 생성되면 새 신호가 신호 카탈로그에 자동으로 추가됩니다. 자세한 정보는 차량 모델 생성(콘솔)을 참조하세요.
   > - `CreateModelManifest`API 작업을 사용하여 차량 모델을 생성하는 경우 `UpdateModelManifest` API 작업을 사용하여 차량 모델을 활성화해야 합니다. 자세한 정보는 차량 모델 업데이트(AWS CLI)을 참조하세요.

- AWS IoT FleetWise 콘솔을 사용하여 차량 모델을 생성하면 AWS IoT가 FleetWise 자동으로 차량 모델을 활성화합니다.

3. 디코더 매니페스트를 생성합니다. 디코더 매니페스트에는 차량 모델(이전 단계에서 생성)에 명시된 모든 신호의 디코딩 정보가 포함되어 있습니다. 디코더 매니페스트는 생성한 차량 모델과 연결됩니다. 자세한 정보는 디코더 매니페스트 생성 및 관리을 참조하세요.

> **Note**
>
> - `CreateDecoderManifest` API 작업을 사용하여 디코더 매니페스트를 만드는 경우 `UpdateDecoderManifest` API 작업을 사용하여 디코더 매니페스트를 활성화해야 합니다. 자세한 정보는 디코더 매니페스트 업데이트(AWS CLI)을 참조하세요.
> - AWS IoT FleetWise 콘솔을 사용하여 디코더 매니페스트를 생성하면 AWS IoT가 FleetWise 자동으로 디코더 매니페스트를 활성화합니다.

CAN 버스 데이터베이스는.dbc 파일 형식을 지원합니다. .dbc 파일을 업로드하여 신호 및 디코더 신호를 가져올 수 있습니다. .dbc 파일 예제를 가져오려면 다음을 수행합니다.

.dbc 파일을 가져오려는 경우

1. .zip을 EngineSignals 다운로드하세요.
2. `EngineSignals.zip` 파일을 다운로드한 디렉터리로 이동합니다.
3. 콘텐츠의 압축을 풀고 `EngineSignals.dbc`(으)로 로컬로 저장합니다.

주제

- 신호 카탈로그 생성 및 관리
- 차량 모델 생성 및 관리
- 디코더 매니페스트 생성 및 관리

신호 카탈로그 생성 및 관리

> **Note**
>
> 데모 스크립트를 다운로드하여 ROS 2 메시지를 신호 카탈로그와 호환되는 VSS JSON 파일로 변환할 수 있습니다. 자세한 내용은 Vision System Data Developer Guide를 참조하세요.

신호 카탈로그는 차량 모델을 만드는 데 재사용할 수 있는 표준화된 신호 모음입니다. AWS IoT는 신호를 정의할 때 따를 수 있는 차량 신호 사양 (VSS) 을 FleetWise 지원합니다. 신호는 다음 유형 중 하나일 수 있습니다.

속성

속성은 일반적으로 변경되지 않는 정적 정보(예: 제조업체 및 제조일)를 나타냅니다.

브랜치

브랜치는 신호를 중첩된 구조로 나타냅니다. 브랜치는 신호 계층 구조를 보여줍니다. 예를 들어, `Vehicle` 브랜치에는 하위 브랜치, `Powertrain`이(가) 있습니다. `Powertrain` 브랜치에는 하위 브랜치, `combustionEngine`이(가) 있습니다. `combustionEngine` 브랜치를 찾으려면 `Vehicle.Powertrain.combustionEngine` 표현식을 사용하세요.

센서

센서 데이터는 차량의 현재 상태와 시간에 따른 변화(예: 유체 수준, 온도, 진동, 전압 등)를 보고합니다.

액추에이터

액추에이터 데이터는 모터, 히터, 도어록과 같은 차량 장치의 상태를 보고합니다. 차량 장치의 상태를 변경하면 액추에이터 데이터를 업데이트할 수 있습니다. 예를 들어, 히터를 나타내는 액추에이터를 정의할 수 있습니다. 히터를 켜거나 끌 때 액추에이터가 새 데이터를 수신합니다.

사용자 지정 구조

사용자 지정 구조(구조체라고도 함)는 복합 또는 고차 데이터 구조를 나타냅니다. 이를 통해 동일한 소스에서 생성된 데이터를 쉽게 논리적으로 바인딩하거나 그룹화할 수 있습니다. 구조체는 복합 데이터 유형 또는 고차 모양을 나타내는 것과 같이 원자성 연산으로 데이터를 읽거나 쓸 때 사용됩니다.

구조체 유형의 신호는 프리미티브 데이터 유형 대신 구조체 데이터 유형에 대한 참조를 사용하여 신호 카탈로그에 정의됩니다. 구조체는 센서, 속성, 액추에이터, 비전 시스템 데이터 유형을 비

롯한 모든 유형의 신호에 사용할 수 있습니다. 구조체 유형의 신호가 전송되거나 수신되는 경우 AWS IoT는 포함된 모든 항목이 유효한 값을 가질 FleetWise 것으로 예상하므로 모든 항목은 필수입니다. 예를 들어 구조체에 Vehicle.Camera.Image.height, Vehicle.Camera.Image.width 및 Vehicle.Camera.Image.data 항목이 포함된 경우 송신된 신호에 이러한 모든 항목의 값이 포함될 것으로 예상됩니다.

> **Note**
>
> 비전 시스템 데이터는 평가판 릴리스이며 변경될 수 있습니다.

사용자 지정 속성

사용자 지정 속성은 복합 데이터 구조의 멤버를 나타냅니다. 속성의 데이터 유형은 프리미티브이거나 다른 구조체일 수 있습니다.

구조체와 사용자 지정 속성을 사용하여 고차 모양을 표현할 때는 의도한 고차 모양이 항상 트리 구조로 정의되고 표시됩니다. 사용자 지정 속성은 모든 리프 노드를 정의하는 데 사용되고 구조체는 리프가 아닌 모든 노드를 정의하는 데 사용됩니다.

> **Note**
>
> - AWS IoT FleetWise 콘솔을 사용하여 첫 번째 차량 모델을 생성하는 경우 신호 카탈로그를 수동으로 생성할 필요가 없습니다. 첫 번째 차량 모델을 만들면 AWS IoT가 FleetWise 자동으로 신호 카탈로그를 생성합니다. 자세한 정보는 차량 모델 생성(콘솔)을 참조하세요.
> - AWS IoT FleetWise 콘솔을 사용하여 차량 모델을 생성하는 경우.dbc 파일을 업로드하여 신호를 가져올 수 있습니다.. dbc는 컨트롤러 영역 네트워크 (CAN 버스) 데이터베이스가 지원하는 파일 형식입니다. 차량 모델이 생성되면 새 신호가 신호 카탈로그에 자동으로 추가됩니다. 자세한 정보는 차량 모델 생성(콘솔)을 참조하세요.
> - AWS IoT는 FleetWise 현재 각 AWS 계정 지역에 대한 신호 카탈로그를 지원합니다.

AWS IoT는 신호 카탈로그를 만들고 관리하는 데 사용할 수 있는 다음과 같은 API 작업을 FleetWise 제공합니다.

- CreateSignalCatalog— 새 신호 카탈로그를 생성합니다.

- ImportSignalCatalog— JSON 파일을 업로드하여 신호를 가져와 신호 카탈로그를 생성합니다. 신호는 VSS에 따라 정의하고 JSON 형식으로 저장해야 합니다.
- UpdateSignalCatalog— 신호를 업데이트, 제거 또는 추가하여 기존 신호 카탈로그를 업데이트합니다.
- DeleteSignalCatalog— 기존 신호 카탈로그를 삭제합니다.
- ListSignalCatalogs— 모든 신호 카탈로그의 요약이 페이지별로 구분된 목록을 검색합니다.
- ListSignalCatalogNodes— 지정된 신호 카탈로그에 있는 모든 신호 (노드) 의 요약 페이지를 매긴 목록을 검색합니다.
- GetSignalCatalog— 신호 카탈로그에 대한 정보를 검색합니다.

튜토리얼

- 신호 구성
- 신호 카탈로그 생성(AWS CLI)
- 신호 카탈로그 가져오기
- 신호 카탈로그 업데이트(AWS CLI)
- 신호 카탈로그 삭제(AWS CLI)
- 신호 카탈로그 정보 가져오기(AWS CLI)

신호 구성

이 섹션에서는 분기, 속성, 센서 및 액추에이터를 구성하는 방법을 보여줍니다.

주제

- 브랜치 구성
- 구성 속성
- 센서 또는 액추에이터를 구성합니다
- 복합 데이터 유형 구성

브랜치 구성

새 연결을 추가하려면 다음 정보를 지정합니다.

- `fullyQualifiedName` – 브랜치의 완전히 정규화된 이름은 브랜치 경로에 브랜치 이름을 더한 것입니다. 자식 브랜치를 가리키려면 점(.)을 사용합니다. 예를 들어

`Vehicle.Chassis.SteeringWheel`은()는 `SteeringWheel` 브랜치의 완전히 정규화된 이름입니다. `Vehicle.Chassis.`이(가) 브랜치의 경로입니다.

정규화된 이름은 최대 150자까지 가능합니다. 유효한 문자: a–z, A–Z, 0–9, 콜론(:) 및 밑줄(_)

- (선택 사항) `Description` – 브랜치에 대한 설명입니다.

  설명은 최대 2048자입니다. 유효한 문자: a–z, A–Z, 0–9, :(콜론), _(밑줄), 및 -(하이픈)

- (선택 사항) `deprecationMessage` - 이동 또는 삭제 중인 노드 또는 분기에 대한 지원 중단 메시지입니다.

  deprecationMessage는 최대 2048자까지 쓸 수 있습니다. 유효한 문자: a–z, A–Z, 0–9, :(콜론), _(밑줄), 및 -(하이픈)

- (선택 사항) `comment` — 설명 외에 코멘트를 추가합니다. 코멘트는 브랜치에 대한 이론적 근거나 관련 브랜치에 대한 참조 등 브랜치에 대한 추가 정보를 제공하는 데 사용될 수 있습니다.

  코멘트는 최대 2048자까지 쓸 수 있습니다. 유효한 문자: a–z, A–Z, 0–9, :(콜론), _(밑줄), 및 -(하이픈)

구성 속성

속성을 구성하려면 다음 정보를 지정하세요.

- `dataType`— 속성의 데이터 유형은 INT8, UINT8, INT16, UINT16, INT32, UINT32, INT64, UINT64, 부울, 플로트, 더블, 문자열, UNIX_타임스탬프, INT8_ARRAY, UINT8_ARRAY, INT16_ARRAY, INT32_ARRAY, INT32_ARRAY, INT64_ARRAY, BOOLEAN_ARRAY 중 하나여야 합니다., FLOAT_ARRAY, DOUBLE_ARRAY, STRING_ARRAY, UNIX_TIMESTAMP_ARRAY, UNNOWN 또는 데이터 유형 브랜치에 정의된 사용자 지정 구조체 fullyQualifiedName
- `fullyQualifiedName` – 속성의 완전히 정규화된 이름은 속성 경로에 속성 이름을 더한 값입니다. 자식 신호를 나타내려면 점(.)을 사용합니다. 예를 들어 `Vehicle.Chassis.SteeringWheel.Diameter`은(는) `Diameter` 속성의 완전히 정규화된 이름입니다. `Vehicle.Chassis.SteeringWheel.`은(는) 속성의 경로입니다.

  완전히 정규화된 이름은 최대 150자까지 가능합니다. 유효한 문자: a–z, A–Z, 0–9, : (콜론), 및 _ (밑줄).

- (선택 사항) `Description` — 속성에 대한 설명입니다.

  설명은 최대 2048자입니다. 유효한 문자: a–z, A–Z, 0–9, :(콜론), _(밑줄), 및 -(하이픈)

- (선택 사항) `unit` — 속성의 과학 단위 (예: km 또는 섭씨).
- (선택 사항) `min` — 속성의 최소값입니다.
- (선택 사항) `max` — 속성의 최대값입니다.
- (선택 사항) `defaultValue` — 속성의 기본값입니다.
- (선택 사항) `assignedValue` — 속성에 할당된 값입니다.
- (선택 사항) `allowedValues` — 속성이 허용하는 값 목록입니다.
- (선택 사항) `deprecationMessage` — 이동 또는 삭제되는 노드 또는 분기에 대한 지원 중단 메시지입니다.

  deprecationMessage는 최대 2048자까지 쓸 수 있습니다. 유효한 문자: a–z, A–Z, 0–9, :(콜론), _(밑줄), 및 -(하이픈)
- (선택 사항) `comment` — 설명 외에 코멘트를 추가합니다. 코멘트를 사용하여 속성에 대한 이론적 근거나 관련 속성에 대한 참조 등 속성에 대한 추가 정보를 제공할 수 있습니다.

  코멘트는 최대 2048자까지 쓸 수 있습니다. 유효한 문자: a–z, A–Z, 0–9, :(콜론), _(밑줄), 및 -(하이픈)

센서 또는 액추에이터를 구성합니다

센서 또는 액추에이터를 구성하려면 다음 정보를 지정합니다.

- `dataType`— 신호의 데이터 유형은 INT8, UINT8 INT16 UINT16 INT32 UINT32 INT64 UINT64, BOOLEAN_ARRAY, FLOAT, DOUBLE, STRING, UNIX_TIMESTAMP, INT8_ARRAY, INT16_ARRAY, INT32_ARRAY, INT32_ARRAY, INT64_ARRAY, BOOLEAN_ARRAY 중 하나여야 합니다., FLOAT_ARRAY, DOUBLE_ARRAY, STRING_ARRAY, UNIX_TIMESTAMP_ARRAY fullyQualifiedName, UNNOWN 또는 데이터 유형 브랜치에 정의된 사용자 지정 구조체
- `fullyQualifiedName`— 신호의 완전히 정규화된 이름은 신호 경로에 신호 이름을 더한 것입니다. 하위 신호를 나타내려면 점(.)을 사용합니다. 예를 들어 `Vehicle.Chassis.SteeringWheel.HandsOff.HandsOffSteeringState`은 (는) `HandsOffSteeringState` 액추에이터의 완전히 정규화된 이름입니다. `Vehicle.Chassis.SteeringWheel.HandsOff.`이(가) 액추에이터의 경로입니다.

  완전히 정규화된 이름은 최대 150자까지 가능합니다. 유효한 문자: a–z, A–Z, 0–9, : (콜론), 및 _ (밑줄).
- (선택 사항) `Description` — 신호에 대한 설명입니다.

설명은 최대 2048자입니다. 유효한 문자: a–z, A–Z, 0–9, :(콜론), _(밑줄), 및 -(하이픈)

- (선택 사항) `unit` — 신호의 과학 단위 (예: km 또는 섭씨).
- (선택 사항) `min` — 신호의 최소값입니다.
- (선택 사항) `max` — 신호의 최대값입니다.
- (선택 사항) `assignedValue` — 신호에 할당된 값입니다.
- (선택 사항) `allowedValues` — 신호가 받아들이는 값 목록입니다.
- (선택 사항) `deprecationMessage` — 이동 또는 삭제되는 노드 또는 분기에 대한 지원 중단 메시지입니다.

  deprecationMessage는 최대 2048자까지 쓸 수 있습니다. 유효한 문자: a–z, A–Z, 0–9, :(콜론), _(밑줄), 및 -(하이픈)
- (선택 사항) `comment` — 설명 외에 코멘트를 추가합니다. 코멘트를 사용하여 센서 또는 액추에이터에 대한 추가 정보 (예: 이론적 근거 또는 관련 센서 또는 액추에이터에 대한 참조) 를 제공할 수 있습니다.

  코멘트는 최대 2048자까지 쓸 수 있습니다. 유효한 문자: a–z, A–Z, 0–9, :(콜론), _(밑줄), 및 -(하이픈)

복합 데이터 유형 구성

복합 데이터 유형은 비전 시스템을 모델링할 때 사용됩니다. 분기 외에도 이러한 데이터 유형은 구조(구조체라고도 함)와 속성으로 구성됩니다. 구조체는 이미지와 같이 여러 값으로 설명되는 신호입니다. 속성은 프리미티브 데이터 유형(예: UINT8) 또는 다른 구조체(예: 타임스탬프)와 같은 구조체의 멤버를 나타냅니다. 예를 들어 Vehicle.Cameras.Front는 분기를 나타내고 Vehicle.Cameras.Front.Image는 구조체를 나타내며 Vehicle.Cameras.Timestamp는 속성을 나타냅니다.

다음 복합 데이터 유형 예시는 신호 및 데이터 유형을 단일 JSON 파일로 내보내는 방법을 보여줍니다.

Example 복합 데이터 유형

```
{
  "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**
>
> 데모 스크립트를 다운로드하여 ROS 2 메시지를 신호 카탈로그와 호환되는 VSS JSON 파일로 변환할 수 있습니다. 자세한 내용은 Vision System Data Developer Guide를 참조하세요. 비전 시스템 데이터는 평가판 릴리스이며 변경될 수 있습니다.

구조체 구성

사용자 지정 구조(또는 구조체)를 구성하려면 다음 정보를 지정합니다.

- `fullyQualifiedName` - 사용자 지정 구조의 정규화된 이름입니다. 예를 들어, 사용자 지정 구조의 정규화된 이름은 `ComplexDataTypes.VehicleDataTypes.SVMCamera`일 수 있습니다.

  완전히 정규화된 이름은 최대 150자까지 가능합니다. 유효한 문자: a–z, A–Z, 0–9, : (콜론), 및 _ (밑줄).

- (선택 사항) Description — 신호에 대한 설명입니다.

  설명은 최대 2048자입니다. 유효한 문자: a–z, A–Z, 0–9, :(콜론), _(밑줄), 및 -(하이픈)

- (선택 사항) deprecationMessage — 이동 또는 삭제되는 노드 또는 분기에 대한 지원 중단 메시지입니다.

  deprecationMessage는 최대 2048자까지 쓸 수 있습니다. 유효한 문자: a–z, A–Z, 0–9, :(콜론), _(밑줄), 및 -(하이픈)

- (선택 사항) comment — 설명 외에 코멘트를 추가합니다. 코멘트를 사용하여 센서 또는 액추에이터에 대한 추가 정보 (예: 이론적 근거 또는 관련 센서 또는 액추에이터에 대한 참조) 를 제공할 수 있습니다.

  코멘트는 최대 2048자까지 쓸 수 있습니다. 유효한 문자: a–z, A–Z, 0–9, :(콜론), _(밑줄), 및 -(하이픈)

속성 구성

사용자 지정 속성을 구성하려면 다음 정보를 지정합니다.

- dataType - 신호의 데이터 유형은 다음 중 하나여야 합니다. INT8, UINT8, INT16, UINT16, INT32, UINT32, INT64, UINT64, BOOLEAN, FLOAT, DOUBLE, STRING, UNIX_TIMESTAMP, INT8_ARRAY, UINT8_ARRAY, INT16_ARRAY, UINT16_ARRAY, INT32_ARRAY, UINT32_ARRAY, INT64_ARRAY, UINT64_ARRAY, BOOLEAN_ARRAY, FLOAT_ARRAY, DOUBLE_ARRAY, STRING_ARRAY, UNIX_TIMESTAMP_ARRAY 또는 UNKNOWN
- fullyQualifiedName - 사용자 지정 속성의 정규화된 이름입니다. 예를 들어, 사용자 지정 속성의 정규화된 이름은 ComplexDataTypes.VehicleDataTypes.SVMCamera.FPS일 수 있습니다.

  완전히 정규화된 이름은 최대 150자까지 가능합니다. 유효한 문자: a–z, A–Z, 0–9, : (콜론), 및 _ (밑줄)

- (선택 사항) Description — 신호에 대한 설명입니다.

  설명은 최대 2048자입니다. 유효한 문자: a–z, A–Z, 0–9, :(콜론), _(밑줄), 및 -(하이픈)

- (선택 사항) deprecationMessage — 이동 또는 삭제되는 노드 또는 분기에 대한 지원 중단 메시지입니다.

  deprecationMessage는 최대 2048자까지 쓸 수 있습니다. 유효한 문자: a–z, A–Z, 0–9, :(콜론), _(밑줄), 및 -(하이픈)

- (선택 사항) comment — 설명 외에 코멘트를 추가합니다. 코멘트를 사용하여 센서 또는 액추에이터에 대한 추가 정보 (예: 이론적 근거 또는 관련 센서 또는 액추에이터에 대한 참조) 를 제공할 수 있습니다.

  코멘트는 최대 2048자까지 쓸 수 있습니다. 유효한 문자: a–z, A–Z, 0–9, :(콜론), _(밑줄), 및 -(하이픈)
- (선택 사항) dataEncoding - 속성이 바이너리 데이터인지 여부를 나타냅니다. 사용자 지정 속성의 데이터 인코딩은 BINARY 또는 TYPED 중 하나여야 합니다.
- (선택 사항) structFullyQualifiedName — 사용자 지정 속성의 데이터 유형이 Struct 또는 인 경우 사용자 지정 속성에 대한 구조체 (struct) 노드의 정규화된 이름입니다 StructArray.

  완전히 정규화된 이름은 최대 150자까지 가능합니다. 유효한 문자: a–z, A–Z, 0–9, : (콜론), 및 _ (밑줄).

신호 카탈로그 생성(AWS CLI)

CreateSignalCatalogAPI 작업을 사용하여 신호 카탈로그를 만들 수 있습니다. 다음 예제에서는 를 사용합니다 AWS CLI.

신호 카탈로그를 만들려면 다음 명령을 실행합니다.

구성이 포함된 JSON 파일 이름으로 *signal-catalog-configuration*바꾸십시오.

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

- 생성 중인 신호 카탈로그의 *signal-catalog-name*이름으로 바꾸십시오.
- (선택 사항) *description*을 신호 카탈로그를 식별하는 데 도움이 되는 설명으로 바꿉니다.

분기, 속성, 센서 및 액추에이터를 구성하는 방법에 대한 자세한 내용은 신호 구성 섹션을 참조하세요.

```
{
    "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**
>
> 데모 스크립트를 다운로드하여 ROS 2 메시지를 신호 카탈로그와 호환되는 VSS JSON 파일로 변환할 수 있습니다. 자세한 내용은 Vision System Data Developer Guide를 참조하세요. 비전 시스템 데이터는 평가판 릴리스이며 변경될 수 있습니다.

신호 카탈로그 가져오기

AWS IoT FleetWise 콘솔 또는 API를 사용하여 신호 카탈로그를 가져올 수 있습니다.

주제

신호 카탈로그 가져오기(콘솔)

AWS IoT FleetWise 콘솔을 사용하여 신호 카탈로그를 가져올 수 있습니다.

> **Important**
>
> 최대 1개의 신호 카탈로그를 가질 수 있습니다. 신호 카탈로그가 이미 있는 경우 콘솔에 신호 카탈로그를 가져오는 옵션이 표시되지 않습니다.

신호 카탈로그를 가져오려는 경우

1. AWS IoT FleetWise 콘솔을 엽니다.
2. 탐색 창에서 신호 카탈로그(Signal catalog)를 선택합니다.
3. 신호 카탈로그 요약 페이지에서 신호 카탈로그 가져오기를 선택합니다.
4. 신호가 포함된 파일을 가져옵니다.
   - S3 버킷으로부터 파일을 업로드하려면
     a. S3에서 가져오기(Import from S3)를 선택합니다.
     b. S3 찾아보기(Browse S3)를 선택합니다.
     c. 버킷의 경우 버킷 이름 또는 객체를 입력하고 목록에서 선택한 다음 목록에서 파일을 선택합니다. 파일 선택 버튼을 선택합니다.

     또는 S3 URI의 경우 Amazon Simple Storage Service URI를 입력합니다. 자세한 내용은 Amazon S3 사용 설명서의 버킷에 액세스하는 방법을 참조하세요.
   - 컴퓨터에서 파일을 업로드하려는 경우
     a. Import file(파일 가져오기)을 선택합니다.
     b. .json 파일을 차량 신호 사양(VSS)형식으로.json 파일을 업로드합니다.
5. 신호 카탈로그를 확인한 다음 파일 가져오기를 선택합니다.

신호 카탈로그 가져오기(AWS CLI)

ImportSignalCatalogAPI 작업을 사용하여 신호 카탈로그를 생성하는 데 도움이 되는 JSON 파일을 업로드할 수 있습니다. 신호를 JSON 파일에 저장하려면 차량 신호 사양(VSS) 을 따라야 합니다. 다음 예제에서는 를 사용합니다 AWS CLI.

신호 카탈로그를 가져오려면 다음 명령을 실행합니다.

- 만들고 *signal-catalog-name*있는 신호 카탈로그의 이름으로 바꾸십시오.
- (선택 사항) 설명을 신호 카탈로그를 식별하는 데 도움이 되는 *##*으로 바꾸세요.
- *signal-catalog-configuration-vss*VSS에 정의된 신호가 포함된 JSON 문자열 파일의 이름으로 바꾸십시오.

분기, 속성, 센서 및 액추에이터를 구성하는 방법에 대한 자세한 내용은 신호 구성 섹션을 참조하세요.

```
aws iotfleetwise import-signal-catalog \
                --name signal-catalog-name \
                --description  description \
                --vss file://signal-catalog-configuration-vss.json
```

JSON을 문자열화하여 vssJson 필드에 전달해야 합니다. 다음은 VSS에 정의된 신호의 예제입니다.

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

}
```

아래 예제에서는 VSS에 정의된 것과 동일한 신호를 JSON 문자열로 보여줍니다.

```
{
   "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**
>
> 데모 스크립트를 다운로드하여 ROS 2 메시지를 신호 카탈로그와 호환되는 VSS JSON 파일로 변환할 수 있습니다. 자세한 내용은 Vision System Data Developer Guide를 참조하세요.
> 비전 시스템 데이터는 평가판 릴리스이며 변경될 수 있습니다.

신호 카탈로그 업데이트(AWS CLI)

UpdateSignalCatalogAPI 작업을 사용하여 기존 신호 카탈로그를 업데이트할 수 있습니다. 다음 예제에서는 를 사용합니다 AWS CLI.

기존 신호 카탈로그를 업데이트하려면 다음 명령을 실행합니다.

구성이 포함된 JSON 파일 이름으로 *signal-catalog-configuration*바꾸십시오.

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

업데이트하려는 신호 카탈로그의 *signal-catalog-name*이름으로 바꾸십시오.

분기, 속성, 센서 및 액추에이터를 구성하는 방법에 대한 자세한 내용은 신호 구성 섹션을 참조하세요.

> **Important**
>
> 사용자 지정 구조는 변경할 수 없습니다. 기존 사용자 지정 구조(구조체)의 순서를 변경하거나 속성을 삽입해야 하는 경우 구조를 삭제하고 원하는 속성 순서로 완전히 새로운 구조를 생성합니다.
> 사용자 지정 구조를 삭제하려면 nodesToRemove에서 해당 구조의 정규화된 이름을 추가합니다. 신호에서 참조되는 구조는 삭제할 수 없습니다. 구조를 참조하는 모든 신호(해당 데이터 유형이 대상 구조로 정의됨)는 신호 카탈로그 업데이트를 요청하기 전에 업데이트하거나 삭제해야 합니다.

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

신호 카탈로그 삭제(AWS CLI)

DeleteSignalCatalogAPI 작업을 사용하여 신호 카탈로그를 삭제할 수 있습니다. 다음 예제에서는 를 사용합니다 AWS CLI.

> **Important**
>
> 신호 카탈로그를 삭제하기 전에 관련 차량 모델, 디코더 매니페스트, 차량, 플릿 또는 캠페인이 없는지 확인하세요. 지침은 다음을 참조하세요.
>
> - 차량 모델 삭제
> - 디코더 매니페스트 삭제
> - 차량 삭제
> - 플릿 삭제(AWS CLI)
> - 캠페인 삭제

기존 신호 카탈로그를 삭제하려면 다음 명령을 실행합니다. 삭제하려는 신호 카탈로그의 *signal-catalog-name*이름으로 바꾸십시오.

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

> **Note**
>
> 이 명령은 출력을 생성하지 않습니다.

신호 카탈로그 정보 가져오기(AWS CLI)

ListSignalCatalogsAPI 작업을 사용하여 신호 카탈로그가 삭제되었는지 확인할 수 있습니다. 다음 예제에서는 를 사용합니다 AWS CLI.

모든 신호 카탈로그의 요약 목록을 페이지별로 검색하려면 다음 명령을 실행합니다.

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

ListSignalCatalogNodesAPI 작업을 사용하여 신호 카탈로그가 업데이트되었는지 확인할 수 있습니다. 다음 예제에서는 를 사용합니다 AWS CLI.

지정된 신호 카탈로그에 있는 모든 신호(노드)의 요약 목록을 페이지별로 구분하여 검색하려면 다음 명령을 실행합니다.

확인 중인 신호 카탈로그의 *signal-catalog-name*이름으로 바꾸십시오.

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

GetSignalCatalogAPI 작업을 사용하여 신호 카탈로그 정보를 검색할 수 있습니다. 다음 예제에서는 를 사용합니다 AWS CLI.

신호 카탈로그에 대한 정보를 검색하려면 다음 명령을 실행합니다.

검색하려는 신호 카탈로그의 *signal-catalog-name*이름으로 바꾸십시오.

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

> **Note**
>
> 이 작업은 결과적 일관성을 갖습니다. 즉, 신호 카탈로그의 변경 사항이 즉시 반영되지 않을 수도 있습니다.

차량 모델 생성 및 관리

신호를 사용하여 차량 형식 표준화에 도움이 되는 차량 모델을 생성합니다. 차량 모델은 동일한 유형의 여러 차량에 일관된 정보를 적용하므로 여러 차량의 데이터를 처리할 수 있습니다. 동일한 차량 모델에서 생성된 차량은 동일한 신호 그룹을 상속받습니다. 자세한 정보는 차량 생성, 공급 및 관리을 참조하세요.

각 차량 모델에는 차량 모델의 상태가 포함된 상태 필드가 있습니다. 상태는 다음 값 중 하나일 수 있습니다.

- `ACTIVE`— 차량 모델이 활성 상태입니다.
- `DRAFT`— 차량 모델의 구성이 저장됩니다.

Important

- `CreateModelManifest` API 작업을 사용하여 첫 번째 차량 모델을 생성하려면 먼저 신호 카탈로그를 생성해야 합니다. 자세한 정보는 신호 카탈로그 생성(AWS CLI)을 참조하세요.
- AWS IoT FleetWise 콘솔을 사용하여 차량 모델을 생성하면 AWS IoT가 FleetWise 자동으로 차량 모델을 활성화합니다.
- `CreateModelManifest` API 작업을 사용하여 차량 모델을 생성하는 경우 차량 모델은 `DRAFT` 상태를 유지합니다.
- `DRAFT` 상태에 있는 차량 모델로는 차량을 생성할 수 없습니다. `UpdateModelManifest` API 작업을 사용하여 차량 모델을 `ACTIVE` 상태로 변경합니다.
- `ACTIVE` 상태에 있는 차량 모델은 편집할 수 없습니다.

주제

차량용 모델 생성

AWS IoT FleetWise 콘솔 또는 API를 사용하여 차량 모델을 생성할 수 있습니다.

Important

`CreateModelManifest` API 작업을 사용하여 차량 모델을 생성하려면 먼저 신호 카탈로그가 있어야 합니다.

주제

차량 모델 생성(콘솔)

AWS IoT FleetWise 콘솔에서 다음과 같은 방법으로 차량 모델을 생성할 수 있습니다.

- 에서 제공한 템플릿을 사용하십시오. AWS
- 수동으로 차량 모델 생성
- 차량 모델 복제

에서 제공한 템플릿을 사용하십시오. AWS

AWS IoT는 신호 카탈로그, 차량 모델 및 디코더 매니페스트를 자동으로 생성하는 온보드 진단 (OBD) II, J1979 템플릿을 FleetWise 제공합니다. 또한 템플릿은 OBD 네트워크 인터페이스를 디코더 매니페스트에 추가합니다. 자세한 정보는 디코더 매니페스트 생성 및 관리을 참조하세요.

템플릿을 사용하여 차량 모델을 만들려는 경우

1. AWS IoT FleetWise 콘솔로 이동합니다.
2. 탐색 창에서 차량 모델을 선택합니다.
3. 차량 모델 페이지에서 제공된 템플릿 추가를 선택합니다.
4. 온보드 진단(OBD) II를 선택합니다.
5. AWS FleetWise IoT가 생성하는 OBD 네트워크 인터페이스의 이름을 입력합니다.
6. 추가를 선택합니다.

수동으로 차량 모델 생성

하나 이상의.dbc 파일을 업로드하여 신호 카탈로그의 신호를 추가하거나 신호를 가져올 수 있습니다. .dbc 파일은 CAN 버스(컨트롤러 영역 네트워크) 데이터베이스에서 지원하는 파일 형식입니다.

> **Important**
>
> AWS IoT FleetWise 콘솔을 사용하여 비전 시스템 데이터 신호가 포함된 차량 모델을 만들 수 없습니다. 대신 AWS CLI 를 사용하여 차량 모델을 생성하십시오.
> 비전 시스템 데이터는 평가판 릴리스이며 변경될 수 있습니다.

차량 모델을 수동으로 생성하려면

1. AWS IoT FleetWise 콘솔로 이동합니다.

2. 기본 탐색 창에서 차량 모델을 선택합니다.
3. 차량 모델 페이지에서 차량 모델 생성을 선택하고 다음을 수행합니다.

주제

1단계: 차량 모델 구성

일반 정보에서 다음을 수행합니다.

1. 차량 모델 이름을 입력합니다.
2. (선택 사항) 설명을 입력합니다.
3. 다음을 선택합니다.

2단계: 신호 추가

> **Note**
>
> - AWS FleetWiseIoT를 처음 사용하는 경우 신호 카탈로그가 있어야 이 단계를 수행할 수 있습니다. 첫 번째 차량 모델이 생성되면 AWS IoT는 첫 번째 차량 모델에 추가된 신호가 포함된 신호 카탈로그를 FleetWise 자동으로 생성합니다.
> - AWS FleetWiseIoT에 익숙하다면 신호 카탈로그에서 신호를 선택하거나.dbc 파일을 업로드하여 신호를 가져오는 방식으로 차량 모델에 신호를 추가할 수 있습니다.
> - 차량 모델을 생성하려면 신호가 하나 이상 있어야 합니다.

신호를 추가하려는 경우

1. 차량 모델에 추가할 신호 카탈로그에서 하나 이상의 신호를 선택합니다. 오른쪽 창에서 선택한 신호를 리뷰할 수 있습니다.

> **Note**
>
> 선택한 신호만 차량 모델에 추가됩니다.

2. 다음을 선택합니다.

3단계: 신호 가져오기

> **Note**
>
> - AWS FleetWiseIoT를 처음 사용하는 경우 신호를 가져오려면 최소한 하나 이상의.dbc 파일을 업로드해야 합니다.
> - AWS FleetWiseIoT에 익숙하다면 신호 카탈로그에서 신호를 선택하거나.dbc 파일을 업로드하여 신호를 가져오는 방식으로 차량 모델에 신호를 추가할 수 있습니다.
> - 차량 모델을 생성하려면 신호가 하나 이상 있어야 합니다.

신호를 가져오려는 경우

1. 파일 선택을 선택합니다.
2. 대화 상자에서 신호가 포함된.dbc 파일을 선택합니다. 여러 개의 .dbc 파일을 업로드할 수 있습니다.
3. AWS IoT는.dbc 파일을 FleetWise 파싱하여 신호를 검색합니다.

   신호 섹션에서 각 신호에 대해 다음 메타데이터를 지정합니다.

   - 이름 - 신호 이름.

     신호는 고유해야 합니다. 신호 이름과 경로는 최대 150자까지 가능합니다. 유효한 문자: a–z, A–Z, 0–9, : (콜론), 및 _ (밑줄).
   - 데이터 유형 — 신호의 데이터 유형은 다음 중 하나여야 합니다: INT8, UINT8, INT16, UINT16, INT32, UINT32, INT64, UINT64, BOOLEAN, FLOAT, DOUBLE, STRING, UNIX_TIMESTAMP, INT8_ARRAY, UINT8_ARRAY, INT16_ARRAY, UINT16_ARRAY, INT32_ARRAY, UINT32_ARRAY, INT64_ARRAY, UINT64_ARRAY, BOOLEAN_ARRAY, FLOAT_ARRAY, DOUBLE_ARRAY, STRING_ARRAY, UNIX_TIMESTAMP_ARRAY, 또는 UNKNOWN.
   - 신호 유형 — 신호 유형으로, 센서 또는 액추에이터일 수 있습니다.

- (선택 사항) 단위 — 신호의 과학 단위(예: km 또는 섭씨).
- (선택 사항) 경로 — 신호 경로. JSONPath와 마찬가지로 점(.)을 사용하여 하위 신호를 나타냅니다. 예를 들어 **`Vehicle.Engine.Light`**입니다.

  신호 이름과 경로는 최대 150자까지 가능합니다. 유효한 문자: a–z, A–Z, 0–9, : (콜론), 및 _ (밑줄).
- (선택 사항) 최소 — 신호의 최소값입니다.
- (선택 사항) 최대 — 신호의 최대값입니다.
- (선택 사항) 설명 – 신호에 대한 설명입니다.

  설명은 최대 2048자입니다. 유효한 문자: a–z, A–Z, 0–9, :(콜론), _(밑줄), 및 -(하이픈)

4. 다음을 선택합니다.

(선택 사항) 4단계: 속성 추가

신호 카탈로그의 기존 속성을 포함하여 최대 100개의 속성을 추가할 수 있습니다.

속성을 추가하려는 경우

1. 속성 추가에서 각 속성에 대해 다음 메타데이터를 지정합니다.

   - 이름 – 속성의 이름.

     신호 이름은 고유해야 합니다. 신호 이름과 경로는 최대 150자까지 가능합니다. 유효한 문자: a–z, A–Z, 0–9, : (콜론), 및 _ (밑줄)
   - 데이터 유형 — 속성의 데이터 유형은 다음 중 하나여야 합니다: INT8, UINT8, INT16, UINT16, INT32, UINT32, INT64, UINT64, BOOLEAN, FLOAT, DOUBLE, STRING, UNIX_TIMESTAMP, INT8_ARRAY, UINT8_ARRAY, INT16_ARRAY, UINT16_ARRAY, INT32_ARRAY, UINT32_ARRAY, INT64_ARRAY, UINT64_ARRAY, BOOLEAN_ARRAY, FLOAT_ARRAY, DOUBLE_ARRAY, STRING_ARRAY, UNIX_TIMESTAMP_ARRAY 또는 UNKNOWN
   - (선택 사항) 단위 — 속성의 과학 단위 (예: km 또는 섭씨).
   - (선택 사항) 경로 — 신호 경로. JSONPath와 마찬가지로 점(.)을 사용하여 하위 신호를 나타냅니다. 예를 들어 **`Vehicle.Engine.Light`**입니다.

     신호 이름과 경로는 최대 150자까지 입력할 수 있습니다. 유효한 문자: a–z, A–Z, 0–9, : (콜론), 및 _ (밑줄)
   - (선택 사항) 최소 — 속성의 최솟값입니다.

- (선택 사항) 최대 — 속성의 최대값입니다.
- (선택 사항) 설명 — 속성에 대한 설명입니다.

  설명은 최대 2048자입니다. 유효한 문자: a–z, A–Z, 0–9, :(콜론), _(밑줄), 및 -(하이픈)

2. 다음을 선택합니다.

5단계: 검토 및 생성

차량 모델의 구성을 확인한 다음 생성을 선택합니다.

차량 모델 복제

AWS IoT는 기존 차량 모델의 구성을 복사하여 새 모델을 만들 FleetWise 수 있습니다. 선택한 차량 모델에 지정된 신호가 새 차량 모델에 복사됩니다.

차량 모델을 복제하려는 경우

1. AWS IoT FleetWise 콘솔로 이동합니다.
2. 탐색 창에서 차량 모델을 선택합니다.
3. 차량 모델 목록에서 모델을 선택한 다음 모델 복제를 선택합니다.

차량 모델을 구성하려면 수동으로 차량 모델 생성 튜토리얼을 따르세요.

AWS IoT가 차량 모델 생성 요청을 처리하는 FleetWise 데 몇 분 정도 걸릴 수 있습니다. 차량 모델이 성공적으로 생성되면 차량 모델 페이지의 상태 열에 ACTIVE가 표시됩니다. 차량 모델이 활성화되면 편집할 수 없습니다.

차량 모델 생성(AWS CLI)

CreateModelManifestAPI 작업을 사용하여 차량 모델 (모델 매니페스트) 을 생성할 수 있습니다. 다음 예제에서는 AWS CLI을(를) 사용합니다.

> **Important**
>
> AWS IoT FleetWise API를 사용하여 첫 번째 차량 모델을 만들려면 먼저 신호 카탈로그를 생성해야 합니다. 신호 카탈로그 생성 방법에 대한 자세한 내용은 신호 카탈로그 생성(AWS CLI) 섹션을 참조하세요.

차량를 생성하려면 다음 명령을 실행합니다.

구성이 포함된 JSON 파일 이름으로 *vehicle-model-configuration*바꾸십시오.

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

- 만들고 *vehicle-model-name*있는 차량 모델의 이름으로 바꾸십시오.
- *signal-catalog-ARN*을 신호 카탈로그의 Amazon 리소스 이름(ARN)으로 바꿉니다.
- (선택 사항) 차량 모델을 식별하는 데 도움이 되는 *##*으로 바꿉니다.

분기, 속성, 센서 및 액추에이터를 구성하는 방법에 대한 자세한 내용은 신호 구성 섹션을 참조하세요.

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

차량 모델 업데이트(AWS CLI)

UpdateModelManifestAPI 작업을 사용하여 기존 차량 모델 (모델 매니페스트) 을 업데이트할 수 있습니다. 다음 예제에서는 AWS CLI을(를) 사용합니다.

기존 차량 모델을 업데이트하려면 다음 명령을 실행합니다.

구성이 *update-vehicle-model-configuration*포함된 JSON 파일 이름으로 바꾸십시오.

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

- 업데이트하려는 차량 모델 *vehicle-model-name*이름으로 바꾸십시오.
- (선택 사항) 차량 모델을 활성화하려면 *vehicle-model-status*로 교체하십시오ACTIVE.

> **Important**
> 차량 모델을 활성화시킨 후에는 차량 모델을 변경할 수 없습니다.

- *(## ##) ##*을 차량 모델을 쉽게 식별할 수 있도록 업데이트된 설명으로 교체합니다.

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

차량 모델 삭제

AWS IoT FleetWise 콘솔 또는 API를 사용하여 차량 모델을 삭제할 수 있습니다.

> **Important**
>
> 차량 모델과 관련된 차량 및 디코더 매니페스트를 먼저 삭제해야 합니다. 자세한 내용은 차량 삭제 및 디코더 매니페스트 삭제 섹션을 참조하세요.

차량 모델 삭제(콘솔)

차량 모델을 삭제하려면 AWS IoT FleetWise 콘솔을 사용하세요.

차량 모델 삭제하려는 경우

1. AWS IoT FleetWise 콘솔로 이동합니다.
2. 탐색 창에서 차량 모델을 선택합니다.
3. 차량 모델 페이지에서 대상 차량 모델을 선택합니다.
4. 삭제를 선택합니다.
5. 삭제하시겠습니까 **vehicle-model-name**?에서 삭제할 차량 모델 이름을 입력한 다음 확인을 선택합니다.

차량 모델 삭제(AWS CLI)

DeleteModelManifestAPI 작업을 사용하여 기존 차량 모델 (모델 매니페스트) 을 삭제할 수 있습니다. 다음 예제에서는 AWS CLI을(를) 사용합니다.

차량 모델을 삭제하려면, 다음 명령을 실행합니다.

삭제하려는 차량 모델 *model-manifest-name*이름으로 바꾸십시오.

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

> **Note**
>
> 이 명령은 출력을 생성하지 않습니다.

차량 모델 정보 가져오기(AWS CLI)

ListModelManifestsAPI 작업을 사용하여 차량 모델이 삭제되었는지 확인할 수 있습니다. 다음 예제에서는 를 사용합니다 AWS CLI.

모든 차량 모델의 페이지별 요약 목록을 검색하려면 다음 명령을 실행합니다.

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

ListModelManifestNodesAPI 작업을 사용하여 차량 모델이 업데이트되었는지 확인할 수 있습니다. 다음 예제에서는 를 사용합니다 AWS CLI.

지정된 차량 모델에 있는 모든 신호(노드)의 요약 목록을 페이지별로 구분하여 검색하려면 다음 명령을 실행합니다.

확인 중인 차량 모델 *vehicle-model-name*이름으로 바꾸십시오.

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

차량 모델에 대한 정보를 검색하려면 다음 명령을 실행합니다.

*vehicle-model*을 검색하려는 차량 모델 이름으로 교체합니다.

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

> **Note**
>
> 이 작업은 결과적 일관성을 갖습니다. 다시 말해서 차량 모델을 변경하더라도 바로 반영되지 않을 수도 있습니다.

디코더 매니페스트 생성 및 관리

디코더 매니페스트에는 AWS IoT가 차량 데이터 (바이너리 데이터) 를 사람이 읽을 수 있는 값으로 변환하고 데이터 분석을 위한 데이터를 준비하는 데 FleetWise 사용하는 디코딩 정보가 포함되어 있습니다. 네트워크 인터페이스 및 디코더 신호는 디코더 매니페스트 구성에 사용하는 핵심 구성 요소입니다.

네트워크 인터페이스

차량 내 네트워크에서 사용하는 프로토콜에 대한 정보가 들어 있습니다. AWS IoT는 다음 프로토콜을 FleetWise 지원합니다.

컨트롤러 영역 네트워크(CAN 버스)

전자 제어 장치(ECU) 간 데이터 통신 방식을 정의하는 프로토콜입니다. ECU는 엔진 제어 장치, 에어백 또는 오디오 시스템일 수 있습니다.

온보드 진단(OBD) II

자체 진단 데이터가 ECU 간에 전달되는 방식을 정의하는 추가 개발된 프로토콜입니다. 차량의 문제를 식별하는 데 도움이 되는 여러 표준 진단 문제 코드(DTC)를 제공합니다.

차량 미들웨어

네트워크 인터페이스 유형으로 정의되는 차량 미들웨어입니다. 차량 미들웨어의 예로는 로봇 운영 체제(ROS 2) 및 IP를 통한 확장 가능한 서비스 지향 미들웨어(SOME/IP)가 있습니다.

> **Note**
>
> AWS IoT는 비전 시스템 데이터를 위한 ROS 2 미들웨어를 FleetWise 지원합니다.

디코더 신호

특정 신호에 대한 자세한 디코딩 정보를 제공합니다. 차량 모델에 지정된 모든 신호는 디코더 신호와 페어링되어야 합니다. 디코더 매니페스트에 CAN 네트워크 인터페이스가 포함된 경우 CAN

디코더 신호를 포함해야 합니다. 디코더 매니페스트에 OBD 네트워크 인터페이스가 포함된 경우 OBD 디코더 신호를 포함해야 합니다.

디코더 매니페스트에 차량 미들웨어 인터페이스도 포함되어 있는 경우 메시지 디코더 신호가 포함되어야 합니다.

각 디코더 매니페스트는 차량 모델과 연결되어야 합니다. AWS IoT는 관련 디코더 매니페스트를 FleetWise 사용하여 차량 모델을 기반으로 생성된 차량의 데이터를 디코딩합니다.

각 디코더 매니페스트에는 디코더 매니페스트의 상태가 포함된 상태 필드가 있습니다. 상태는 다음 값 중 하나일 수 있습니다.

- `ACTIVE`— 디코더 매니페스트가 활성 상태입니다.
- `DRAFT`— 디코더 매니페스트의 컨피그레이션이 저장되지 않습니다.
- `VALIDATING` - 디코더 매니페스트가 적격성을 검증하고 있습니다. 이는 하나 이상의 비전 시스템 데이터 신호가 포함된 디코더 매니페스트에만 적용됩니다.
- `INVALID` - 디코더 매니페스트가 검증에 실패하여 아직 활성화할 수 없습니다. 이는 하나 이상의 비전 시스템 데이터 신호가 포함된 디코더 매니페스트에만 적용됩니다. ListDecoderManifests 및 GetDecoderManifest API를 사용하여 검증 실패 이유를 확인할 수 있습니다.

> **Important**
>
> - AWS IoT FleetWise 콘솔을 사용하여 디코더 매니페스트를 생성하면 AWS IoT가 FleetWise 자동으로 디코더 매니페스트를 활성화합니다.
> - `CreateDecoderManifest` API 작업을 사용하여 디코더 매니페스트를 생성하는 경우 디코더 매니페스트는 `DRAFT` 상태를 유지합니다.
> - `DRAFT` 디코더 매니페스트와 연결된 차량 모델로는 차량을 만들 수 없습니다. `UpdateDecoderManifest` API 작업을 사용하여 디코더 매니페스트를 `ACTIVE` 상태로 변경합니다.
> - `ACTIVE` 상태에 있는 디코더 매니페스트는 편집할 수 없습니다.

주제

- 네트워크 인터페이스 및 디코더 신호를 구성합니다
- 디코더 매니페스트 생성

- 디코더 매니페스트 업데이트(AWS CLI)
- 디코더 매니페스트 삭제
- 디코더 매니페스트 정보 가져오기 (AWS CLI)

네트워크 인터페이스 및 디코더 신호를 구성합니다

모든 디코더 매니페스트에는 적어도 하나의 네트워크 인터페이스와 디코더 신호가 있으며 이는 관련된 차량 모델에 지정된 신호와 짝을 이룹니다.

디코더 매니페스트에 CAN 네트워크 인터페이스가 포함된 경우 CAN 디코더 신호를 포함해야 합니다. 디코더 매니페스트에 OBD 네트워크 인터페이스가 포함된 경우 OBD 디코더 신호를 포함해야 합니다.

주제

- 네트워크 인터페이스 구성
- 디코더 신호 구성

네트워크 인터페이스 구성

CAN 네트워크 인터페이스를 구성하려면 다음 정보를 지정합니다.

- `name`— CAN 인터페이스의 이름.

  인터페이스 이름은 고유해야 하며 1~100자를 포함할 수 있습니다.
- (선택 사항) `protocolName` — 프로토콜 이름.

  유효한 값: `CAN-FD` 및 `CAN`
- (선택 사항) `protocolVersion` — AWS IoT는 FleetWise 현재 CAN-FD 및 CAN 2.0b를 지원합니다.

  유효한 값: `1.0` 및 `2.0b`

OBD 네트워크 인터페이스를 구성하려면 다음 정보를 지정합니다.

- `name`— OBD 인터페이스의 이름.

  인터페이스 이름은 고유해야 하며 1~100자를 포함할 수 있습니다.
- `requestMessageId` – 차량 데이터를 요청하는 메시지의 ID입니다.

- (선택 사항) dtcRequestIntervalSeconds - 차량에서 진단 문제 코드(DTC)를 요청하는 빈도를 초 단위로 나타냅니다. 예를 들어 지정된 값이 120인 경우 Edge Agent 소프트웨어는 2분에 한 번씩 저장된 DTC를 수집합니다.
- (선택 사항) hasTransmissionEcu - 차량에 변속기 제어 모듈(TCM)이 있는지 여부입니다.

  유효한 값: true 및 false
- (선택 사항) obdStandard — AWS IoT가 FleetWise 지원하는 OBD 표준입니다. AWS IoT는 FleetWise 현재 월드 와이드 하모나이제이션 온보드 진단 (WWH-OBD) ISO15765-4 표준을 지원합니다.
- (선택 사항) pidRequestIntervalSeconds - 차량에서 OBD II PID를 요청하는 빈도. 예를 들어, 지정된 값이 120인 경우 Edge Agent 소프트웨어는 2분에 한 번씩 OBD II PID를 수집합니다.
- (Optional) useExtendedIds – 메시지에 확장 ID를 사용할지 여부입니다.

  유효한 값: true 및 false

차량 미들웨어 네트워크 인터페이스를 구성하려면 다음 정보를 지정합니다.

- name - 차량 미들웨어 인터페이스의 이름입니다.

  인터페이스 이름은 고유해야 하며 1~100자를 포함할 수 있습니다.
- protocolName - 프로토콜 이름입니다.

  유효값: ROS_2

디코더 신호 구성

CAN 디코더 신호를 구성하려면 다음 정보를 지정합니다.

- factor – 메시지를 디코딩하는 데 사용되는 승수입니다.
- isBigEndian – CAN 메시지의 바이트 순서가 빅엔디언인지 여부입니다. 빅엔디안인 경우 시퀀스에서 가장 중요한 값이 가장 낮은 저장소 주소에 먼저 저장됩니다.
- isSigned— 메시지 서명 여부. 서명된 메시지는 양수와 음수를 모두 표시할 수 있습니다.
- length – 메시지의 바이트 길이.
- messageId – 메시지의 ID입니다.
- offset— 신호 값을 계산하는 데 사용되는 오프셋입니다. 팩터와 함께 계산하면 계산은 value = raw_value * factor + offset와(과) 같습니다.

- `startBit`— 메시지의 첫 번째 비트 위치를 나타냅니다.
- (선택 사항) `name` — 신호의 이름입니다.

OBD 디코더 신호를 구성하려면 다음 정보를 지정합니다.

- `byteLength` – 바이트 단위의 메시지 길이입니다.
- `offset`— 신호 값을 계산하는 데 사용되는 오프셋입니다. 스케일링과 함께 계산하면 `value = raw_value * scaling + offset`와(과) 같습니다.
- `pid` – 이 신호에 대해 차량에서 데이터 요청에 사용되는 진단 코드입니다.
- `pidResponseLength`— 요청된 메시지의 길이입니다.
- `scaling` – 메시지를 디코딩하는 데 사용되는 승수입니다.
- `serviceMode` – 메시지의 작업 모드(진단 서비스)입니다.
- `startByte` – 메시지의 시작을 나타냅니다.
- (선택 사항) `bitMaskLength` — 메시지에서 마스킹되는 비트 수입니다.
- (선택 사항) `bitRightShift` — 오른쪽으로 이동한 위치 수입니다.

메시지 디코더 신호를 구성하려면 다음 정보를 지정합니다.

- `topicName` - 메시지 신호의 주제 이름입니다. ROS 2의 주제에 해당합니다. 구조화된 메시지 객체에 대한 자세한 내용은 을 참조하십시오. StructuredMessage
- `structuredMessage` - 메시지 신호에 대한 구조화된 메시지입니다. a primitiveMessageDefinition, structuredMessageList Definition을 사용하여 정의하거나 structuredMessageDefinition 재귀적으로 정의할 수 있습니다.

디코더 매니페스트 생성

AWS IoT FleetWise 콘솔 또는 API를 사용하여 차량 모델의 디코더 매니페스트를 생성할 수 있습니다.

> **Important**
>
> 디코더 매니페스트를 생성하려면 우선 차량 모델을 보유해야 합니다. 모든 디코더 매니페스트는 차량 모델과 연결되어야 합니다. 자세한 정보는 차량 모델 생성 및 관리을 참조하세요.

주제

- 디코더 매니페스트 생성(콘솔)
- 디코더 매니페스트 만들기(AWS CLI)

디코더 매니페스트 생성(콘솔)

AWS IoT FleetWise 콘솔을 사용하여 차량 모델과 관련된 디코더 매니페스트를 만들 수 있습니다.

> **Important**
>
> AWS IoT FleetWise 콘솔을 사용하여 디코더 매니페스트의 비전 시스템 데이터 신호를 구성할 수 없습니다. 그 대신 AWS CLI을 사용합니다. 비전 시스템 데이터는 평가판 릴리스이며 변경될 수 있습니다.

디코더 매니페스트를 생성하려는 경우

1. AWS IoT FleetWise 콘솔로 이동합니다.
2. 기본 탐색 창에서 차량 모델을 선택합니다.
3. 대상 차량 모델을 선택합니다.
4. 차량 모델 요약 페이지에서 디코더 매니페스트 생성을 선택하고 다음을 수행합니다.

주제

- 1단계: 디코더 매니페스트 구성
- 2단계: 네트워크 인터페이스 추가
- 3단계: 검토 및 생성

1단계: 디코더 매니페스트 구성

일반 정보에서 다음을 수행합니다.

1. 디코더 매니페스트에 고유한 이름을 입력합니다.
2. (선택 사항) 설명을 입력합니다.
3. 다음을 선택합니다.

2단계: 네트워크 인터페이스 추가

각 디코더 매니페스트에는 네트워크 인터페이스가 하나 이상 있어야 합니다. 여러 개의 네트워크 인터페이스를 디코더 매니페스트에 추가할 수 있습니다.

네트워크 인터페이스 생성

- 네트워크 인터페이스에서 다음을 수행합니다.
  a. 네트워크 인터페이스 유형에서 CAN_INTERFACE 또는 OBD_INTERFACE를 선택합니다.
  b. 네트워크 인터페이스의 고유한 이름을 입력합니다.
  c. 고유한 네트워크 인터페이스 ID를 입력합니다. AWS IoT에서 생성된 ID를 사용할 수 FleetWise 있습니다.
  d. 차량 모델에 지정된 하나 이상의 신호를 선택하여 디코더 신호와 페어링합니다.
  e. 디코딩 정보를 제공하려면.dbc 파일을 업로드하세요. AWS IoT는.dbc 파일을 FleetWise 파싱하여 디코더 신호를 검색합니다.
  f. 페어링 신호 섹션에서 모든 신호가 디코더 신호와 페어링되었는지 확인합니다.
  g. 다음을 선택합니다.

> **Note**
>
> - 각 네트워크 인터페이스마다 한개의 .dbc 파일만 업로드할 수 있습니다.
> - 차량 모델에 지정된 모든 신호가 디코더 신호와 페어링되었는지 확인합니다.
> - 다른 네트워크 인터페이스를 추가하기로 선택한 후에는 편집 중인 인터페이스를 편집할 수 없습니다. 기존 네트워크 인터페이스를 모두 삭제할 수 있습니다.

3단계: 검토 및 생성

디코더 매니페스트의 구성을 확인한 다음 생성을 선택합니다.

디코더 매니페스트 만들기(AWS CLI)

CreateDecoderManifestAPI 작업을 사용하여 디코더 매니페스트를 만들 수 있습니다. 다음 예제에서는 AWS CLI을(를) 사용합니다.

> **Important**
>
> 디코더 매니페스트를 생성하기 전에 먼저 차량 모델을 생성합니다. 자세한 정보는 차량용 모델 생성을 참조하세요.

디코더 매니페스트를 만들려면 다음 명령을 실행합니다.

구성이 *decoder-manifest-configuration*포함된 JSON 파일 이름으로 바꾸십시오.

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

- 만들고 *decoder-manifest-name*있는 디코더 매니페스트의 이름으로 바꾸십시오.
- *vehicle-model-ARN*을 차량 모델의 Amazon 리소스 이름(ARN)으로 교체합니다.
- (선택 사항) *##*을 디코더 매니페스트를 식별하는 데 도움이 되는 설명으로 교체합니다.

분기, 속성, 센서 및 액추에이터를 구성하는 방법에 대한 자세한 내용은 네트워크 인터페이스 및 디코더 신호를 구성합니다 섹션을 참조하세요.

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

- 만들고 *decoder-manifest-name*있는 디코더 매니페스트의 이름으로 바꾸십시오.
- *vehicle-model-ARN*을 차량 모델의 Amazon 리소스 이름(ARN)으로 교체합니다.
- (선택 사항) *##*을 디코더 매니페스트를 식별하는 데 도움이 되는 설명으로 교체합니다.

구조(구조체) 내 속성 노드의 순서는 신호 카탈로그 및 차량 모델(모델 매니페스트)에 정의된 것과 동일하게 유지되어야 합니다. 분기, 속성, 센서 및 액추에이터를 구성하는 방법에 대한 자세한 내용은 네트워크 인터페이스 및 디코더 신호를 구성합니다 섹션을 참조하세요.

```
{
 "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**
>
> 데모 스크립트를 다운로드하여 비전 시스템 신호가 포함된 디코더 매니페스트를 생성할 수 있습니다. 자세한 내용은 Vision System Data Developer Guide를 참조하세요.
> 비전 시스템 데이터는 평가판 릴리스이며 변경될 수 있습니다.

디코더 매니페스트 업데이트(AWS CLI)

UpdateDecoderManifestAPI 작업을 사용하여 디코더 매니페스트를 업데이트할 수 있습니다. 네트워크 인터페이스와 신호 디코더를 추가, 제거 및 업데이트할 수 있습니다. 디코더 매니페스트의 상태를 변경할 수도 있습니다. 다음 예제에서는 AWS CLI을(를) 사용합니다.

디코더 매니페스트를 업데이트하려면 다음 명령을 실행합니다.

업데이트하려는 디코더 매니페스트의 *decoder-manifest-name*이름으로 바꾸십시오.

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

> **Important**
>
> 디코더 매니페스트를 활성화한 후에는 편집할 수 없습니다.

디코더 매니페스트 삭제

AWS IoT FleetWise 콘솔 또는 API를 사용하여 디코더 매니페스트를 삭제할 수 있습니다.

> **Important**
>
> 디코더 매니페스트와 연결된 차량을 먼저 삭제해야 합니다. 자세한 정보는 차량 삭제을 참조하세요.

주제

- 디코더 매니페스트 삭제(콘솔)

- 디코더 매니페스트 삭제(AWS CLI)

디코더 매니페스트 삭제(콘솔)

AWS IoT FleetWise 콘솔을 사용하여 디코더 매니페스트를 삭제할 수 있습니다.

디코더 매니페스트를 삭제하려는 경우

1. AWS IoT FleetWise 콘솔로 이동합니다.
2. 기본 탐색 창에서 차량 모델을 선택합니다.
3. 대상 차량 모델을 선택합니다.
4. 차량 모델 요약 페이지에서 디코더 매니페스트 탭을 선택합니다.
5. 대상 디코더 매니페스트를 선택한 다음 삭제를 선택합니다.
6. 삭제하시겠습니까 **decoder-manifest-name**?에서 삭제할 디코더 매니페스트의 이름을 입력한 다음 확인을 선택합니다.

디코더 매니페스트 삭제(AWS CLI)

DeleteDecoderManifestAPI 작업을 사용하여 디코더 매니페스트를 삭제할 수 있습니다. 다음 예제에서는 를 사용합니다. AWS CLI

> **Important**
>
> 디코더 매니페스트를 삭제하기 전에 먼저 관련 차량을 삭제합니다. 자세한 정보는 차량 삭제을 참조하세요.

디코더 매니페스트를 삭제하려면 다음 명령을 실행합니다.

삭제하려는 디코더 매니페스트의 *decoder-manifest-name*이름으로 바꾸십시오.

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

디코더 매니페스트 정보 가져오기 (AWS CLI)

ListDecoderManifestsAPI 작업을 사용하여 디코더 매니페스트가 삭제되었는지 확인할 수 있습니다. 다음 예제에서는 를 사용합니다. AWS CLI

모든 디코더 매니페스트의 요약 목록을 페이지 단위로 검색하려면 다음 명령을 실행합니다.

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

ListDecoderManifestSignalsAPI 작업을 사용하여 디코더 매니페스트의 디코더 신호가 업데이트되었는지 확인할 수 있습니다. 다음 예제에서는 를 사용합니다. AWS CLI

지정된 디코더 매니페스트에 있는 모든 디코더 신호(노드)의 요약 목록을 페이지별로 구분하여 검색하려면 다음 명령어를 실행합니다.

확인 중인 디코더 매니페스트의 *decoder-manifest-name*이름으로 바꾸십시오.

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

ListDecoderManifestNetworkInterfacesAPI 작업을 사용하여 디코더 매니페스트의 네트워크 인터페이스가 업데이트되었는지 확인할 수 있습니다. 다음 예에는 AWS CLI이(가) 사용됩니다.

지정된 디코더 매니페스트에서 모든 네트워크 인터페이스의 요약 목록을 페이지별로 분류하여 검색하려면 다음 명령을 실행합니다.

확인 중인 디코더 매니페스트의 *decoder-manifest-name*이름으로 바꾸십시오.

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

GetDecoderManifestAPI 작업을 사용하여 디코더 매니페스트의 네트워크 인터페이스 및 디코더 신호가 업데이트되었는지 확인할 수 있습니다. 다음 예제에서는 를 사용합니다. AWS CLI

디코더 매니페스트에 대한 정보를 검색하려면 다음 명령을 실행합니다.

*decoder-manifest*를 검색하려는 디코더 매니페스트의 이름으로 교체합니다.

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

> **Note**
>
> 이 작업은 결과적 일관성을 갖습니다. 다시 말해서 디코더 매니페스트를 변경하더라도 바로 반영되지 않을 수도 있습니다.

차량 생성, 공급 및 관리

차량은 차량 모델의 인스턴스입니다. 차량은 차량 모델에서 생성되고 디코더 매니페스트와 연결되어야 합니다. 차량은 하나 이상의 데이터 스트림을 클라우드에 업로드합니다. 예를 들어 차량은 주행 거리, 엔진 온도, 히터 상태 데이터를 클라우드로 전송할 수 있습니다. 모든 차량에는 다음 정보가 포함되어 있습니다.

`vehicleName`

차량을 식별하는 ID입니다.

개인 식별 정보(PII)나 기타 기밀 정보 또는 민감한 정보를 차량 이름에 추가하지 마십시오. 차량 이름은 Amazon을 비롯한 다른 AWS 서비스에서 액세스할 수 CloudWatch 있습니다. 차량 이름은 개인 데이터나 민감한 데이터에 사용하기 위한 것이 아닙니다.

`modelManifestARN`

차량 모델(모델 매니페스트)의 Amazon 리소스 이름(ARN)입니다. 모든 차량은 차량 모델을 기반으로 생성됩니다. 동일한 차량 모델에서 생성된 차량은 차량 모델에서 상속된 동일한 신호 그룹으로 구성됩니다. 이러한 신호는 신호 카탈로그에서 정의되고 표준화됩니다.

`decoderManifestArn`

디코더 매니페스트의 ARN. 디코더 매니페스트는 AWS IoT가 원시 신호 데이터 (바이너리 데이터)를 사람이 읽을 FleetWise 수 있는 값으로 변환하는 데 사용할 수 있는 디코딩 정보를 제공합니다. 디코더 매니페스트는 차량 모델과 연결되어야 합니다. AWS IoT는 동일한 디코더 매니페스트를 FleetWise 사용하여 동일한 차량 모델을 기반으로 생성된 차량의 원시 데이터를 디코딩합니다.

`attributes`

속성은 정적 정보가 포함된 키-값 페어입니다. 차량에는 차량 모델에서 상속된 속성이 포함될 수 있습니다. 속성을 추가하여 개별 차량을 동일한 차량 모델에서 생성된 다른 차량과 구별할 수 있습니다. 예를 들어 검은색 자동차가 있는 경우 속성에 다음 값을 지정할 수 있습니다: `{"color": "black"}`.

> **Important**
>
> 속성을 개별 차량에 추가하려면 먼저 관련 차량 모델에서 속성을 정의해야 합니다.

차량 모델, 디코더 매니페스트 및 속성에 대한 자세한 내용은 차량 모델링을(를) 참조하세요.

AWS IoT는 차량을 생성하고 관리하는 데 사용할 수 있는 다음과 같은 API 작업을 FleetWise 제공합니다.

- CreateVehicle— 새 차량을 만듭니다.
- BatchCreateVehicle— 한 대 이상의 새 차량을 만듭니다.
- UpdateVehicle— 기존 차량을 업데이트합니다.
- BatchUpdateVehicle— 하나 이상의 기존 차량을 업데이트합니다.
- DeleteVehicle— 기존 차량을 삭제합니다.
- ListVehicles— 모든 차량의 요약 페이지를 매긴 목록을 검색합니다.
- GetVehicle— 차량에 대한 정보를 검색합니다.

튜토리얼

- 차량 공급
- 예약된 주제
- 차량 생성
- 차량 업데이트 (AWS CLI)
- 여러 차량 업데이트 (AWS CLI)
- 차량 삭제
- 차량 정보 가져오기(AWS CLI)

차량 공급

차량에서 실행되는 AWS IoT용 Edge Agent FleetWise 소프트웨어는 데이터를 수집하여 클라우드로 전송합니다. AWS IoT는 MQTT와 FleetWise AWS IoT Core 통합되어 Edge Agent 소프트웨어와 클라우드 간의 보안 통신을 지원합니다. 각 차량은 AWS IoT 사물에 해당합니다. 기존 AWS IoT 사물을 사용하여 차량을 만들거나 AWS IoT를 FleetWise 설정하여 차량용 AWS IoT 사물을 자동으로 생성할 수 있습니다. 자세한 정보는 차량 생성(AWS CLI)을 참조하세요.

AWS IoT Core AWS IoT FleetWise 리소스에 대한 액세스를 안전하게 제어하는 데 도움이 되는 인증 및 권한 부여를 지원합니다. 차량은 X.509 인증서를 사용하여 AWS IoT FleetWise 및 AWS IoT Core 정책을 사용하여 인증 (로그인) 을 받아 지정된 작업을 수행할 수 있는 권한을 부여 (권한 보유) 할 수 있습니다.

차량 인증

AWS IoT Core 정책을 생성하여 차량을 인증할 수 있습니다.

차량을 인증하려는 경우

- AWS IoT Core 정책을 생성하려면 다음 명령을 실행합니다.
  - *policy-name*을 생성하려는 정책 이름으로 바꿉니다.
  - *## ### ###* 포함된 JSON 파일 이름으로 바꾸십시오. AWS IoT Core

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

예제 정책을 사용하기 전에 다음을 수행합니다.

- *###* AWS IoT FleetWise 리소스를 생성한 AWS 지역으로 바꾸십시오.
- *AWS ### ## ID#* 바꾸십시오. AWS

이 예에는 AWS IoT에서 예약한 주제가 포함되어 FleetWise 있습니다. 정책에 주제를 추가해야 합니다. 자세한 정보는 예약된 주제을 참조하세요.

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

차량 권한 부여

X.509 인증서를 생성하여 차량을 인증할 수 있습니다.

차량 인증하기

> **Important**
> 각 차량에 대한 새 인증서를 생성하는 것이 좋습니다.

1. RSA 키 쌍을 생성하고 X.509 인증서를 발급하려면 다음 명령을 실행합니다.

   - *cert*를 CertitificatePEM의 명령 출력 내용을 저장하는 파일 이름으로 교체합니다.
   - *## ## KeyPair#* 명령 출력 내용을 저장하는 파일 이름으로 바꾸십시오. PublicKey.
   - *private-key# KeyPair#* 명령 출력 내용을 저장하는 파일 이름으로 바꾸십시오. 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. 출력에서 인증서의 Amazon 리소스 이름(ARN)을 복사합니다..
3. 인증서에 정책을 첨부하려면 다음 명령을 실행합니다.

   - *policy-name#* 생성한 AWS IoT Core 정책 이름으로 바꾸십시오.
   - *certificate-arn*을 복사한 인증서의 ARN으로 교체합니다.

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

4. 인증서를 사물에 연결하려면 다음 명령을 실행합니다.

   - *## ###* AWS IoT 사물 이름 또는 차량 ID로 바꾸십시오.
   - *certificate-arn*을 복사한 인증서의 ARN으로 교체합니다.

   ```
   aws iot attach-thing-principal \
       --thing-name thing-name \
   ```

```
    --principal "certificate-arn"
```

예약된 주제

AWS IoT는 다음 주제를 사용하도록 FleetWise 예약합니다. 예약된 주제가 허용하는 경우 해당 주제를 구독하거나 게시할 수 있습니다. 그러나 달러 기호로 시작하는 새 주제를 생성할 수는 없습니다. 예약된 주제와 함께 지원되지 않는 게시 또는 구독 작업을 사용하면 연결이 종료될 수 있습니다.

주제	허용된 클라이언트 작업	설명	
`$aws/iotfleetwise/vehicles/`*vehicleName*`/checkins`	게시	Edge Agent 소프트웨어는 차량 상태 정보를 이 주제에 게시합니다. 차량 상태 정보는 프로토콜 버퍼(protobuf) 형식으로 교환됩니다. 자세한 내용은 AWS IoT용 Edge Agent FleetWise 소프트웨어 개발자 안내서를 참조하십시오.	
`$aws/iotfleetwise/vehicles/`*vehicleName*`/signals`	Publish	엣지 에이전트 소프트웨어는 이 주제에 신호를 게시합니다. 신호 정보는 프로토콜 버퍼(protobuf) 형식으로 교환됩니다. 자세한 내용은 AWS IoT용 Edge Agent FleetWise 소프트웨어 개발자 안내서를 참조하십시오.	

주제	허용된 클라이언트 작업	설명	
`$aws/iotfleetwise/vehicles/vehicleName/collection_schemes`	Subscribe	AWS IoT는 이 FleetWise 주제에 대한 데이터 수집 체계를 게시합니다. 차량은 이러한 데이터 수집 체계를 사용합니다.	
`$aws/iotfleetwise/vehicles/vehicleName/decoder_manifests`	Subscribe	AWS IoT는 이 주제에 대한 디코더 매니페스트를 FleetWise 게시합니다. 차량은 이러한 디코더 매니페스트를 사용합니다.	

차량 생성

AWS IoT FleetWise 콘솔 또는 API를 사용하여 차량을 만들 수 있습니다.

> **Important**
>
> 시작하기 전에 다음 사항에 유의하세요.
>
> - 차량 모델이 있어야 하고 차량 모델이 `ACTIVE` 상태여야 합니다. 자세한 정보는 차량 모델 생성 및 관리을 참조하세요.
> - 차량 모델은 디코더 매니페스트와 연결되어야 하고 디코더 매니페스트는 `ACTIVE` 상태여야 합니다. 자세한 정보는 디코더 매니페스트 생성 및 관리을 참조하세요.

주제

- 차량 생성 (콘솔)
- 차량 생성(AWS CLI)
- 여러 차량 생성 (AWS CLI)

차량 생성 (콘솔)

AWS IoT FleetWise 콘솔을 사용하여 차량을 만들 수 있습니다.

> **Important**
>
> 시작하기 전에 다음 사항에 유의하세요.
>
> - 차량 모델이 있어야 하고 차량 모델이 ACTIVE 상태여야 합니다. 자세한 정보는 차량 모델 생성 및 관리을 참조하세요.
> - 차량 모델은 디코더 매니페스트와 연결되어야 하고 디코더 매니페스트는 ACTIVE 상태여야 합니다. 자세한 정보는 디코더 매니페스트 생성 및 관리을 참조하세요.

차량을 생성하려는 경우

1. AWS IoT FleetWise 콘솔을 엽니다.
2. 탐색 창에서 차량을 선택합니다.
3. 차량 요약 페이지에서 차량 생성을 선택하고 다음 단계를 수행합니다.

주제

1단계: 차량 속성 정의

이 단계에서는 차량 이름을 지정하고 모델 매니페스트 및 디코더 매니페스트와 연결합니다.

1. 차량의 고유한 이름을 입력합니다.

> **Important**
>
> 차량은 사물에 AWS IoT 해당합니다. 해당 이름의 사물이 이미 있는 경우 차량을 IoT 사물에 연결을 선택하여 헤딩 차량으로 사물을 업데이트합니다. 또는 다른 차량 이름을 선택하면 AWS FleetWise IoT가 자동으로 차량에 대한 새 이름을 생성합니다.

2. 목록에서 차량 모델(모델 매니페스트)을 선택합니다.
3. 목록에서 디코더 매니페스트를 선택합니다. 디코더 매니페스트는 차량 모델과 연결됩니다.
4. (선택 사항) 차량 속성을 연결하려면 속성 추가를 선택합니다. 이 단계를 건너뛰면 차량이 생성된 후 속성을 추가해야 캠페인에 배포할 수 있습니다.
5. (선택 사항) 태그를 차량에 연결하려면 새 태그 추가를 선택합니다. 차량을 만든 후에도 태그를 추가할 수 있습니다.
6. 다음을 선택합니다.

2단계: 차량 인증서 구성

차량을 AWS IoT 사물로 사용하려면 정책이 첨부된 차량 인증서를 구성해야 합니다. 이 단계를 건너뛰면 차량이 생성된 후 인증서를 구성해야 캠페인에 배포할 수 있습니다.

1. 신규 인증서 자동 생성(권장)을 선택합니다.
2. 다음을 선택합니다.

3단계: 인증서에 정책 연결

이전 단계에서 구성한 인증서에 정책을 연결합니다.

1. 정책의 경우 기존 정책 이름을 입력합니다. 새 정책을 생성하려면, 정책 생성을 선택합니다.
2. 다음을 선택합니다.

4단계: 검토 및 생성

차량 구성을 확인한 다음 차량 생성을 선택합니다.

> **Important**
>
> 차량을 생성한 후에는 인증서와 키를 다운로드해야 합니다. 인증서와 개인 키를 사용하여 AWS IoT용 Edge Agent FleetWise 소프트웨어에서 차량을 연결합니다.

차량 생성(AWS CLI)

차량을 만들 때는 디코더 매니페스트와 연결된 차량 모델을 사용해야 합니다. CreateVehicleAPI 작업을 사용하여 차량을 생성할 수 있습니다. 다음 예제에서는 AWS CLI을(를) 사용합니다.

> **Important**
>
> 시작하기 전에 다음 사항에 유의하세요.
>
> - 차량 모델이 있어야 하고 차량 모델이 `ACTIVE` 상태여야 합니다. 자세한 정보는 차량 모델 생성 및 관리을 참조하세요.
> - 차량 모델은 디코더 매니페스트와 연결되어야 하고 디코더 매니페스트는 `ACTIVE` 상태여야 합니다. 자세한 정보는 디코더 매니페스트 생성 및 관리을 참조하세요.

차량을 생성하려면 다음 명령을 실행합니다.

*file-name*을 차량 구성이 포함된 JSON 파일 이름으로 교체합니다.

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

Example 차량 구성

- (선택 사항) `associationBehavior` 값은 다음 중 하나일 수 있습니다.
  - `CreateIotThing`— 차량이 생성되면 AWS IoT가 AWS IoT 차량의 차량 ID 이름으로 사물을 FleetWise 자동으로 생성합니다.
  - `ValidateIotThingExists`— 기존 사물을 사용하여 AWS IoT 차량을 만들 수 있습니다.

    사물을 AWS IoT 생성하려면 다음 명령을 실행합니다. *## ##*을 만들고 싶은 사물의 이름으로 바꾸세요.

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

지정하지 않으면 AWS IoT가 FleetWise 자동으로 차량용 AWS IoT 사물을 생성합니다.

> **Important**
>
> 차량을 만든 후에 AWS IoT 사물이 프로비저닝되었는지 확인하세요. 자세한 정보는 차량 공급을 참조하세요.

- *vehicle-name*을 다음 중 하나로 바꾸세요.
  - 사물의 이름 (AWS IoT 구성된 `associationBehavior` 경우). `ValidateIotThingExists`
  - `associationBehavior`이(가) `CreateIotThing`(으)로 구성된 경우 생성할 차량 ID입니다.

    차량 ID는 1~100자일 수 있습니다. 유효한 문자: a–z, A–Z, 0–9, 대쉬 (-), 밑줄 (_), 및 콜론 (:).
- *model-manifest-ARN*을 차량 모델(모델 매니페스트)의 ARN으로 교체합니다.
- *decoder-manifest-ARN*을 지정된 차량 모델과 관련된 디코더 매니페스트의 ARN으로 교체합니다.
- (선택 사항) 속성을 추가하여 이 차량을 동일한 차량 모델에서 만든 다른 차량과 구별할 수 있습니다. 예를 들어 전기 자동차를 사용하는 경우 속성에 다음 값을 지정할 수 있습니다: `{"fuelType": "electric"}`.

> **Important**
>
> 속성을 개별 차량에 추가하려면 먼저 관련 차량 모델에서 속성을 정의해야 합니다.

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

여러 차량 생성 (AWS CLI)

BatchCreateVehicleAPI 작업을 사용하여 한 번에 여러 차량을 생성할 수 있습니다. 다음 예제에서는 AWS CLI을(를) 사용합니다.

차량을 여러 대 생성하려면 다음 명령을 실행합니다.

*file-name*을 여러 차량의 구성이 포함된 JSON 파일 이름으로 교체하세요.

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

Example 차량 구성

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

각 배치 작업에 대해 최대 10대의 차량을 생성할 수 있습니다. 구성 파일에 대한 자세한 내용은 차량 생성(AWS CLI) 섹션을 참조하세요.

차량 업데이트 (AWS CLI)

UpdateVehicleAPI 작업을 사용하여 기존 차량을 업데이트할 수 있습니다. 다음 예제에서는 AWS CLI을(를) 사용합니다.

차량을 업데이트하려면 다음 명령을 실행합니다.

*file-name*을 차량 구성이 포함된 JSON 파일 이름으로 교체하세요.

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

Example 차량 구성

- *vehicle-name*을 업데이트하려는 차량의 ID로 교체하세요.
- (선택 사항) *model-manifest-ARN*을 사용 중인 차량 모델을 교체하는 데 사용하는 차량 모델(모델 매니페스트)의 ARN으로 교체합니다.
- (선택 사항) *decoder-manifest-ARN*을 지정한 새 차량 모델과 연결된 디코더 매니페스트의 ARN으로 교체하세요.
- (선택 사항) 차량 *attribute-update-mode*속성으로 교체하십시오.
  - `Merge`— 기존 속성을 새 값으로 업데이트하고 새 속성이 없으면 추가하여 새 속성을 기존 속성에 병합합니다.

    예를 들어, 차량에 `{"color": "black", "fuelType": "electric"}`와(과) 같은 속성이 있고 차량을 `{"color": "", "fuelType": "gasoline", "model": "x"}` 속성으로 업데이트하면 업데이트된 차량의 속성은 다음과 같습니다: `{"fuelType": "gasoline", "model": "x"}`.
  - `Overwrite`— 기존 속성을 새 속성으로 대체합니다.

    예를 들어, 차량에 `{"color": "black", "fuelType": "electric"}`와(과) 같은 속성이 있는 경우 차량을 해당 `{"model": "x"}` 속성으로 업데이트하면 업데이트된 차량에도 해당 `{"model": "x"}` 속성이 있습니다.

  입력에 속성이 있는 경우 이는 필수입니다.
- (선택 사항) 새 속성을 추가하거나 기존 속성을 새 값으로 업데이트하려면 `attributes`을(를) 구성하세요. 예를 들어 전기 자동차를 사용하는 경우 속성에 다음 값을 지정할 수 있습니다: `{"fuelType": "electric"}`.

  속성을 삭제하려면 `attributeUpdateMode`을(를) `Merge`(으)로 구성하세요.

> **Important**
>
> 속성을 개별 차량에 추가하려면 먼저 관련 차량 모델에서 속성을 정의해야 합니다.

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

여러 차량 업데이트 (AWS CLI)

BatchUpdateVehicleAPI 작업을 사용하여 기존 차량 여러 대를 한 번에 업데이트할 수 있습니다. 다음 예제에서는 AWS CLI을(를) 사용합니다.

여러 차량을 업데이트하려면 다음 명령을 실행합니다.

*file-name*을 여러 차량의 구성이 포함된 JSON 파일 이름으로 교체하세요.

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

Example 차량 구성

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

각 일괄 작업에 대해 최대 10대의 차량을 업데이트할 수 있습니다. 이런 종류의 구성에 대한 자세한 정보는 차량 업데이트 (AWS CLI) 섹션을 참조하세요.

차량 삭제

AWS IoT FleetWise 콘솔 또는 API를 사용하여 차량을 삭제할 수 있습니다.

> **Important**
>
> 차량이 삭제되면 AWS IoT는 관련 차량 및 캠페인에서 차량을 FleetWise 자동으로 제거합니다. 자세한 내용은 플릿 생성 및 관리 및 캠페인을 통한 데이터 수집 및 전송 섹션을 참조하세요. 하지만 차량은 여전히 사물로 존재하거나 사물과 연관되어 있습니다. AWS IoT Core사물 삭제에 대한 지침은 AWS IoT Core 개발자 가이드의 사물 삭제를 참조하세요.

차량 삭제(콘솔)

AWS IoT FleetWise 콘솔을 사용하여 차량을 삭제할 수 있습니다.

차량을 삭제하려는 경우

1. AWS IoT FleetWise 콘솔로 이동합니다.
2. 탐색 창에서 차량을 선택합니다.
3. 차량 페이지에서 삭제하려는 차량 옆에 있는 버튼을 선택합니다.
4. 삭제를 선택합니다.
5. 삭제**vehicle-name**에서, 차량 이름을 입력한 다음 삭제를 선택합니다.

차량 삭제(AWS CLI)

DeleteVehicleAPI 작업을 사용하여 차량을 삭제할 수 있습니다. 다음 예에서는 를 사용합니다 AWS CLI.

차량을 삭제하려면 다음 명령을 실행합니다.

*vehicle-name*을 삭제하려는 차량의 ID로 교체합니다.

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

차량 정보 가져오기(AWS CLI)

ListVehiclesAPI 작업을 사용하여 차량이 삭제되었는지 확인할 수 있습니다. 다음 예제에서는 AWS CLI을(를) 사용합니다.

모든 차량에 대한 페이지 매겨진 요약 목록을 검색하려면 다음 명령을 실행합니다.

```
aws iotfleetwise list-vehicles
```

GetVehicleAPI 작업을 사용하여 차량 정보를 검색할 수 있습니다. 다음 예제에서는 AWS CLI을(를) 사용합니다.

차량의 메타데이터를 검색하려면 다음 명령을 실행합니다.

*vehicle-name*을 검색하려는 차량의 ID로 교체하세요.

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

> **Note**
>
> 이 작업은 결과적 일관성을 갖습니다. 다시 말해서 차량을 변경하더라도 바로 반영되지 않을 수도 있습니다.

플릿 생성 및 관리

플릿은 차량 그룹을 나타냅니다. 관련 차량이 없는 플릿은 공허한 존재입니다. 플릿을 사용하여 여러 차량을 동시에 관리하려면 먼저 차량을 플릿과 연결해야 합니다. 한 대의 차량이 다중 플릿에 속할 수 있습니다. 캠페인을 배포하여 여러 차량의 플릿에서 어떤 데이터를 수집할지, 언제 데이터를 수집할지 제어할 수 있습니다. 자세한 내용은 캠페인을 통한 데이터 수집 및 전송 섹션을 참조하세요.

플릿에는 다음 정보가 포함됩니다.

`fleetId`

플릿의 ID입니다.

(선택 사항) `description`

플릿을 찾는 데 도움이 되는 설명.

`signalCatalogArn`

신호 카탈로그의 Amazon 리소스 이름(ARN)입니다.

AWS IoT FleetWise는 플릿을 생성하고 관리하는 데 사용할 수 있는 다음과 같은 API 작업을 제공합니다.

- CreateFleet — 동일한 신호 그룹을 포함하는 차량 그룹을 만듭니다.
- AssociateVehicleFleet — 차량을 플릿에 연결합니다.
- DisassociateVehicleFleet – 차량과 플릿의 연결을 끊습니다.
- UpdateFleet — 기존 플릿에 대한 설명을 업데이트합니다.
- DeleteFleet — 기존 플릿을 삭제합니다.
- ListFleets — 모든 플릿의 요약 페이지를 매긴 목록을 검색합니다.
- ListFleetsForVehicle — 차량이 속한 모든 플릿의 ID 목록을 페이지별로 구분하여 검색합니다.
- ListVehiclesInFleet — 플릿 내 모든 차량의 요약 목록을 페이지별로 구분하여 검색합니다.
- GetFleet — 플릿에 대한 정보를 검색합니다.

주제

- 플릿 만들기(AWS CLI)
- 차량을 플릿과 연결(AWS CLI)

- 차량과 플릿 연결 해제(AWS CLI)
- 플릿 업데이트(AWS CLI)
- 플릿 삭제(AWS CLI)
- 플릿 정보 얻기(AWS CLI)

플릿 만들기(AWS CLI)

CreateFleet API 작업을 사용하여 차량 플릿을 생성할 수 있습니다. 다음 예에는 AWS CLI이(가) 사용됩니다.

> **Important**
>
> 플릿을 생성하려면 먼저 신호 카탈로그를 생성해야 합니다. 자세한 내용은 신호 카탈로그 생성(AWS CLI) 섹션을 참조하세요.

플릿을 생성하려면 다음 명령을 실행합니다.

- *fleet-id*를 생성 중인 플릿의 ID로 교체하세요.

  플릿 ID는 고유해야 하며 1~100자여야 합니다. 유효한 문자: 문자 (A-Z 및 a~z), 숫자 (0-9), 콜론 (:), 대시 (-), 및 밑줄 (_).
- (선택 사항) *description*을 설명으로 대체합니다.

  설명은 1~2048자까지 입력할 수 있습니다.
- *signal-catalog-arn*을 신호 카탈로그의 ARN으로 교체하세요.

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

차량을 플릿과 연결(AWS CLI)

AssociateVehicleFleet API 작업을 사용하여 차량을 플릿과 연결할 수 있습니다. 다음 예에는 AWS CLI이(가) 사용됩니다.

> **Important**
>
> - 차량을 플릿과 연결하려면 먼저 차량과 플릿이 있어야 합니다. 자세한 내용은 차량 생성, 공급 및 관리 섹션을 참조하세요.
> - 차량을 캠페인 대상 플릿과 연결하면 AWS IoT FleetWise가 자동으로 캠페인을 차량에 배포합니다.

차량을 플릿과 연결하려면 다음 명령을 실행합니다.

- *fleet-id*를 플릿의 ID로 교체하세요.
- *vehicle-name*을 차량 ID로 교체하세요.

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

차량과 플릿 연결 해제(AWS CLI)

DisassociateVehicleFleet API 작업을 사용하여 차량과 플릿의 연결을 끊을 수 있습니다. 다음 예에는 AWS CLI이(가) 사용됩니다.

차량과 플릿의 연결을 해제하려면 다음 명령을 실행합니다.

- *fleet-id*를 플릿의 ID로 교체하세요.
- *vehicle-name*을 차량 ID로 교체하세요.

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

플릿 업데이트(AWS CLI)

UpdateFleet API 작업을 사용하여 플릿에 대한 설명을 업데이트할 수 있습니다. 다음 예에는 AWS CLI이(가) 사용됩니다.

플릿을 업데이트하려면 다음 명령을 실행합니다.

- *fleet-id*를 업데이트하려는 플릿의 ID로 교체하세요.
- *description*을 새 설명으로 교체하세요.

  설명은 1~2048자까지 입력할 수 있습니다.

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

플릿 삭제(AWS CLI)

DeleteFleet API 작업을 사용하여 플릿을 삭제할 수 있습니다. 다음 예에는 AWS CLI이(가) 사용됩니다.

> **Important**
>
> 플릿을 삭제하기 전에 연결된 차량이 없는지 확인하세요. 차량과 플릿의 연결을 해제하는 방법에 대한 자세한 내용은 차량과 플릿 연결 해제(AWS CLI)을(를) 참조하세요.

플릿을 삭제하려면 다음 명령을 실행합니다.

*fleet-id*를 삭제하려는 플릿의 ID로 교체하세요.

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

플릿 정보 얻기(AWS CLI)

ListFleets API 작업을 사용하여 플릿이 삭제되었는지 확인할 수 있습니다. 다음 예제에서는 AWS CLI을(를) 사용합니다.

모든 플릿의 요약이 페이지별로 구분된 목록을 검색하려면 다음 명령을 실행합니다.

```
aws iotfleetwise list-fleets
```

ListFleetsForVehicle API 작업을 사용하여 차량이 속한 모든 플릿의 페이지 매김 ID 목록을 검색할 수 있습니다. 다음 예제에서는 AWS CLI을(를) 사용합니다.

차량이 속한 모든 플릿의 ID 목록을 페이지 단위로 검색하려면 다음 명령을 실행합니다.

*vehicle-name*을 차량 ID로 교체하세요.

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

ListVehiclesInFleet API 작업을 사용하여 플릿 내 모든 차량의 페이지를 매긴 요약 목록을 검색할 수 있습니다. 다음 예제에서는 AWS CLI을(를) 사용합니다.

플릿의 모든 차량에 대한 요약 목록을 페이지 단위로 검색하려면 다음 명령을 실행합니다.

*fleet-id*를 플릿의 ID로 교체하세요.

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

GetFleet API 작업을 사용하여 플릿 정보를 검색할 수 있습니다. 다음 예제에서는 AWS CLI을(를) 사용합니다.

플릿의 메타데이터를 검색하려면 다음 명령을 실행합니다.

*fleet-id*를 플릿의 ID로 교체하세요.

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

> **Note**
>
> 이 작업은 결과적 일관성을 갖습니다. 다시 말해서 플릿을 변경하더라도 바로 반영되지 않을 수도 있습니다.

캠페인을 통한 데이터 수집 및 전송

캠페인은 데이터 수집 규칙의 오케스트레이션입니다. 캠페인은 Edge Agent for AWS IoT FleetWise 소프트웨어에서 데이터를 선택, 수집 및 클라우드로 전송하는 방법에 대한 지침을 제공합니다.

클라우드에 캠페인을 생성합니다. 사용자 또는 사용자의 팀이 캠페인을 승인하면 AWS IoT FleetWise가 자동으로 캠페인을 차량에 배포합니다. 캠페인을 차량 또는 여러 차량의 플릿에 배포하도록 선택할 수 있습니다. Edge Agent 소프트웨어는 실행 중인 캠페인이 차량에 배포될 때까지 데이터 수집을 시작하지 않습니다.

> **Note**
>
> 캠페인은 다음과 같은 상황이 발생하기 전까지는 작동하지 않습니다.
>
> - Edge Agent 소프트웨어가 차량에서 실행되고 있습니다. Edge Agent 소프트웨어를 개발, 설치 및 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.
>
> 1. AWS IoT FleetWise 콘솔로 이동합니다.
> 2. 서비스 홈 페이지의 AWS IoT FleetWise 시작하기 섹션에서 Edge Agent 탐색을 선택합니다.
>
> - 차량 프로비저닝을 위해 AWS IoT Core을(를) 설정했습니다. 자세한 내용은 차량 공급 섹션을 참조하세요.

각 캠페인에는 다음 정보가 포함되어 있습니다.

`signalCatalogArn`

캠페인과 연결된 신호 카탈로그의 Amazon 리소스 이름(ARN)입니다.

(선택 사항) `tags`

태그는 캠페인을 관리하는 데 사용할 수 있는 메타데이터입니다. 서로 다른 서비스의 리소스에 동일한 태그를 지정하여 리소스가 서로 연관되어 있음을 나타낼 수 있습니다.

`TargetArn`

캠페인이 배포되는 차량 또는 플릿의 ARN입니다.

`name`

캠페인을 식별하는 데 도움이 되는 고유한 이름.

`collectionScheme`

데이터 수집 체계는 수집할 데이터나 수집 시기에 대한 Edge Agent 소프트웨어 지침을 제공합니다. AWS IoT FleetWise는 현재 조건 기반 수집 체계와 시간 기반 수집 체계를 지원합니다.

`conditionBasedCollectionScheme`

조건 기반 수집 체계는 수집할 데이터를 인식하기 위한 논리적 표현식을 사용합니다. Edge Agent 소프트웨어는 조건이 충족되는 경우 데이터를 수집합니다.

`expression`

수집할 데이터를 인식하는 데 사용되는 논리적 표현식입니다. 예를 들어 `` $variable.`myVehicle.InVehicleTemperature` > 50.0 `` 표현식이 지정된 경우 Edge Agent 소프트웨어는 50.0보다 큰 온도 값을 수집합니다. 표현식을 작성하는 방법에 대한 지침은 캠페인의 논리적 표현식 섹션을 참조하세요.

(선택 사항) `triggerMode`은(는) 다음 값 중 하나일 수 있습니다.

- `RISING_EDGE`— Edge Agent 소프트웨어는 조건이 처음으로 충족되는 경우에만 데이터를 수집합니다. 예: `` $variable.`myVehicle.AirBagDeployed` == true ``.
- `ALWAYS`— Edge Agent 소프트웨어는 조건이 충족될 때마다 데이터를 수집합니다.

(선택 사항) `minimumTriggerIntervalMs`

두 데이터 수집 이벤트 사이의 최소 기간(밀리초)입니다. 신호가 자주 바뀌는 경우 더 느린 속도로 데이터를 수집할 수 있습니다.

(선택 사항) `conditionLanguageVersion`

조건부 표현식 언어의 버전.

`timeBasedCollectionScheme`

시간 기반 수집 체계를 정의할 때는 기간을 밀리초 단위로 지정합니다. Edge Agent 소프트웨어는 기간을 사용하여 데이터 수집 빈도를 결정합니다. 예를 들어, 기간이 120,000밀리초인 경우, Edge Agent 소프트웨어는 2분에 한 번씩 데이터를 수집합니다.

(선택 사항) `compression`

무선 대역폭을 절약하고 네트워크 트래픽을 줄이기 위해 SNAPPY를 지정하여 차량의 데이터를 압축할 수 있습니다.

기본적으로(`OFF`), Edge Agent 소프트웨어는 데이터를 압축하지 않습니다.

`dataDestinationConfigs`

캠페인에서 차량 데이터를 전송할 목적지를 선택합니다. 데이터를 Amazon S3 또는 Amazon Timestream에 저장하도록 선택할 수 있습니다.

S3는 내구성이 뛰어난 데이터 관리 기능과 다운스트림 데이터 서비스를 제공하는 비용 효율적인 데이터 스토리지 메커니즘입니다. S3를 운전 습관과 관련된 데이터나 장기 유지 관리 분석에 사용할 수 있습니다.

Timestream은 추세와 패턴을 거의 실시간으로 식별하는 데 도움이 되는 데이터 지속성 메커니즘입니다. Timestream을 사용하여 차량 속도 또는 제동의 과거 추세를 분석하는 것과 같은 시계열 데이터에 사용할 수 있습니다.

(선택 사항) `dataExtraDimensions`

신호에 대한 추가 정보를 제공하기 위한 속성을 하나 이상 추가할 수 있습니다.

(선택 사항) `description`

캠페인을 식별할 수 있는 설명을 추가할 수 있습니다.

(선택 사항) `diagnosticsMode`

진단 모드를 사용하도록 `SEND_ACTIVE_DTCS`을(를) 구성하면 캠페인에서 차량에 어떤 문제가 있는지 식별할 때 도움이 되는 저장된 표준 진단 문제 코드(DTC)를 전송합니다. 예를 들어, P0097 신호는 엔진 제어 모듈(ECM)이 흡기 온도 센서 2(IAT2) 입력이 정상 센서 범위보다 낮다고 판단했음을 나타냅니다.

기본적으로(`OFF`), Edge Agent 소프트웨어는 진단 코드를 전송하지 않습니다.

(선택 사항) `expiryTime`

캠페인의 만료일을 정의할 수 있습니다. 캠페인이 만료되면 Edge Agent 소프트웨어는 이 캠페인에 지정된 대로 데이터 수집을 중지합니다. 차량에 여러 캠페인을 배포한 경우 Edge Agent 소프트웨어는 다른 캠페인을 사용하여 데이터를 수집합니다.

기본값: `253402243200` (9999년 12월 31일, 00:00:00 UTC)

(선택 사항) `postTriggerCollectionDuration`

사후 트리거 수집 기간을 정의하여 스키마가 간접적으로 호출된 후 Edge Agent 소프트웨어가 지정된 기간 동안 데이터를 계속 수집하도록 할 수 있습니다. 예를 들어, 다음 표현식이 포함된 조건 기반 수집 체계가 간접적으로 호출되는 경우: `$variable.`myVehicle.Engine.RPM` > 7000.0`인 경우, Edge Agent 소프트웨어는 엔진의 분당 회전 수(RPM) 값을 계속 수집합니다.

RPM이 7000보다 한 번만 높아지더라도 기계적 문제가 있음을 의미할 수 있습니다. 이 경우 Edge Agent 소프트웨어에서 상태를 모니터링하는 데 도움이 되는 데이터를 계속 수집하는 것이 좋습니다.

기본 값: `0`

(선택 사항) `priority`

캠페인의 우선순위를 나타내는 정수를 지정할 수 있습니다. 수가 적은 캠페인일수록 우선 순위가 높습니다. 차량에 여러 캠페인을 배포하는 경우 우선 순위가 높은 캠페인이 먼저 시작됩니다.

기본 값: `0`

(선택 사항) `signalsToCollect`

데이터 수집 체계를 실행할 때 데이터가 수집되는 신호 목록입니다.

> **Important**
> 조건 기반 수집 체계의 표현식에 사용되는 신호를 이 필드에 지정해야 합니다.

`name`

데이터 수집 체계를 호출할 때 데이터가 수집되는 신호의 이름입니다.

(선택 사항) `maxSampleCount`

데이터 수집 체계를 호출할 때 Edge Agent 소프트웨어가 수집하여 클라우드로 전송하는 최대 데이터 샘플 수입니다.

(선택 사항) `minimumSamplingIntervalMs`

두 데이터 샘플 수집 이벤트 사이의 최소 기간(밀리초)입니다. 신호가 자주 바뀌는 경우 이 파라미터를 사용하여 더 느린 속도로 데이터를 수집할 수 있습니다.

유효 범위: 0-4294967295

(선택 사항) `spoolingMode`

`spoolingMode`이(가) `TO_DISK`(으)로 구성된 경우 Edge Agent 소프트웨어는 차량이 클라우드에 연결되어 있지 않을 때 데이터를 로컬에 임시로 저장합니다. 연결이 다시 설정되면 로컬에 저장된 데이터가 클라우드로 자동 전송됩니다.

기본 값: `OFF`

(선택 사항) `startTime`

승인된 캠페인은 시작 시 활성화됩니다.

기본 값: `0`

캠페인의 상태는 다음 값 중 하나일 수 있습니다.

- `CREATING`— AWS IoT FleetWise가 캠페인 생성 요청을 처리하고 있습니다.
- `WAITING_FOR_APPROVAL`— 캠페인이 생성되면, `WAITING_FOR_APPROVAL` 상태가 됩니다. 캠페인을 승인하려면 `UpdateCampaign` API 작업을 사용하세요. 캠페인이 승인되면 AWS IoT FleetWise는 캠페인을 대상 차량 또는 차량에 자동으로 배포합니다. 자세한 내용은 캠페인 업데이트 (AWS CLI) 섹션을 참조하세요.
- `RUNNING` — 캠페인이 활성화되었습니다.
- `SUSPENDED`— 캠페인이 일시 중지되었습니다. 캠페인을 재개하려면 `UpdateCampaign` API 작업을 사용하세요.

AWS IoT FleetWise는 캠페인 생성 및 관리에 사용할 수 있는 다음의 API 작업을 제공합니다.

- CreateCampaign — 새 캠페인을 생성합니다.
- UpdateCampaign — 기존 캠페인을 업데이트합니다. 캠페인을 생성한 후에는 이 API 작업을 사용하여 캠페인을 승인해야 합니다.
- DeleteCampaign — 기존 캠페인을 삭제합니다.
- ListCampaigns - 모든 캠페인에 대해 페이지별로 구분된 요약 목록을 검색합니다.
- GetCampaign — 캠페인에 대한 정보를 검색합니다.

튜토리얼

- 캠페인 생성
- 캠페인 업데이트 (AWS CLI)
- 캠페인 삭제
- 캠페인 정보 가져오기(AWS CLI)

캠페인 생성

AWS IoT FleetWise 콘솔 또는 API를 사용하여 차량 데이터를 수집하는 캠페인을 만들 수 있습니다.

Important

캠페인을 제대로 수행하려면 다음이 필요합니다.

- Edge Agent 소프트웨어가 차량에서 실행되고 있습니다. Edge Agent 소프트웨어를 개발, 설치 및 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.
  1. AWS IoT FleetWise 콘솔로 이동합니다.
  2. 서비스 홈 페이지의 AWS IoT FleetWise 시작하기 섹션에서 Edge Agent 탐색을 선택합니다.
- 차량 프로비저닝을 위해 AWS IoT Core을(를) 설정했습니다. 자세한 내용은 차량 공급 섹션을 참조하세요.

주제

캠페인 생성하기(콘솔)

AWS IoT FleetWise 콘솔을 사용하여 캠페인을 생성하고 차량 데이터를 선택, 수집하고 클라우드로 전송할 수 있습니다.

캠페인을 생성하려는 경우

1. AWS IoT FleetWise 콘솔로 이동합니다.
2. 탐색 창에서 캠페인을 선택합니다.
3. 캠페인 페이지에서 캠페인 생성을 선택하고 다음 항목의 단계를 완료하세요.

주제

- 2단계: 스토리지 대상 정의
- 3단계: 차량 추가
- 4단계: 검토 및 생성
- 5단계: 캠페인을 배포합니다

> **Important**
>
> - 캠페인을 생성하려면 먼저 신호 카탈로그와 차량이 있어야 합니다. 자세한 정보는 신호 카탈로그 생성 및 관리 및 차량 생성, 공급 및 관리 섹션을 참조하십시오.
> - 캠페인을 생성한 후에는 캠페인을 승인해야 합니다. 자세한 내용은 5단계: 캠페인을 배포합니다 섹션을 참조하세요.

1단계: 캠페인을 구성합니다

일반 정보 섹션에서 다음을 수행합니다.

1. 캠페인 이름을 입력합니다.
2. (선택 사항) 설명을 입력합니다.

캠페인의 데이터 수집 체계를 구성합니다. 데이터 수집 체계는 수집할 데이터나 수집 시기에 대한 Edge Agent 소프트웨어 지침을 제공합니다. AWS IoT FleetWise 콘솔에서 다음과 같은 방법으로 데이터 수집 체계를 구성할 수 있습니다.

- 데이터 수집 체계를 수동으로 정의합니다.
- 파일을 업로드하여 데이터 수집 체계를 자동으로 정의합니다.

구성 옵션에서 다음 옵션 중 하나를 선택합니다.

- 데이터 수집 체계 유형을 수동으로 지정하고 구성표를 사용자 지정하는 옵션을 정의하려면 데이터 수집 체계 정의를 선택합니다.

  데이터 수집 체계의 유형을 수동으로 지정하고 체계를 사용자 지정하는 옵션을 정의합니다.

1. 데이터 수집 체계 세부 정보 섹션에서 이 캠페인에서 사용할 데이터 수집 체계 유형을 선택합니다. 논리적 표현식을 사용하여 수집할 차량 데이터를 인식하려면 상태 기반을 선택합니다. 특정 기간을 사용하여 차량 데이터 수집 빈도를 결정하려면 시간 기반을 선택합니다.
2. 캠페인에서 데이터를 수집하는 기간을 정의합니다.

   > **Note**
   >
   > 기본적으로 승인된 캠페인은 즉시 활성화되며 종료 시간이 설정되지 않습니다. 추가 요금을 피하려면 시간 범위를 지정해야 합니다.

3. 조건 기반 데이터 수집 체계를 지정한 경우 수집할 데이터를 인식하기 위한 논리적 표현식을 정의해야 합니다. AWS IoT FleetWise는 논리적 표현식을 사용하여 조건 기반 체계에서 수집할 데이터를 인식합니다. 표현식에는 신호의 완전히 정규화된 이름을 변수, 비교 연산자 및 비교 값으로 지정해야 합니다.

   예를 들어 `$variable.`myVehicle.InVehicleTemperature` > 50.0` 표현식을 지정하는 경우 AWS IoT FleetWise는 50.0보다 큰 온도 값을 수집합니다. 표현식을 작성하는 방법에 대한 지침은 캠페인의 논리적 표현식 섹션을 참조하세요.

   수집할 데이터 인식에 사용되는 논리 표현식을 입력합니다.
4. (선택 사항) 조건부 표현식 언어의 버전을 지정합니다. 기본값은 1입니다.
5. (선택 사항) 두 데이터 수집 이벤트 사이의 최소 기간인 최소 트리거 간격을 지정할 수 있습니다. 예를 들어, 신호가 자주 바뀌는 경우 더 느린 속도로 데이터를 수집할 수 있습니다.
6. Edge Agent 소프트웨어가 데이터를 수집할 수 있도록 트리거 모드 조건을 지정합니다. 기본적으로 AWS IoT FleetWise용 Edge Agent 소프트웨어는 조건이 충족될 때마다 항상 데이터를 수집합니다. 또는 조건이 처음으로 충족되는 경우, 즉 첫 번째 트리거 시에만 데이터를 수집할 수 있습니다.
7. 시간 기반 데이터 수집 체계를 지정한 경우 10,000~60,000밀리초 범위의 기간을 밀리초 단위로 지정해야 합니다. Edge Agent 소프트웨어는 기간을 사용하여 데이터 수집 빈도를 결정합니다.
8. (선택 사항) 구성표의 고급 구성표 옵션을 편집할 수 있습니다.

   a. 데이터를 압축하여 무선 대역폭을 절약하고 네트워크 트래픽을 줄이려면 Snappy를 선택합니다.

   b. (선택 사항) 데이터 수집 이벤트 후 데이터 수집을 계속하는 시간 (밀리초) 을 정의하려면 사후 트리거 수집 기간을 지정할 수 있습니다.

c. (선택 사항) 캠페인의 우선 순위 수준을 나타내려면 캠페인 우선 순위를 지정할 수 있습니다. 우선 순위가 적은 캠페인이 먼저 배포되며 우선 순위가 높은 것으로 간주됩니다.

d. Edge Agent 소프트웨어는 차량이 클라우드에 연결되어 있지 않을 때 데이터를 로컬에 임시로 저장할 수 있습니다. 연결이 다시 설정되면 로컬에 저장된 데이터가 클라우드로 자동 전송됩니다. 연결이 끊긴 경우 Edge Agent가 로컬에서 데이터 저장 여부를 지정합니다.

e. (선택 사항) 신호에 대한 추가 정보를 제공하려면 최대 5개의 속성을 추가 데이터 차원으로 추가합니다.

- 파일을 업로드하여 데이터 수집 체계를 정의하려면 로컬 장치에서.json 파일 업로드를 선택합니다. AWS IoT FleetWise는 파일에 정의할 수 있는 옵션을 자동으로 정의합니다. 선택한 옵션을 검토하고 업데이트할 수 있습니다.

  데이터 수집 체계에 대한 세부 정보가 포함된.json 파일을 업로드합니다.

  1. 데이터 수집 체계에 대한 정보를 가져오려면 파일 선택을 선택합니다. 필수 파일 형식에 대한 자세한 내용은 CreateMampaign API 설명서를 참조하세요.

     > **Note**
     >
     > AWS IoT FleetWise는 현재.json 파일 형식 확장자를 지원합니다.

  2. AWS IoT FleetWise는 파일의 정보를 기반으로 데이터 수집 체계를 자동으로 정의합니다. AWS IoT FleetWise가 선택한 옵션을 검토하십시오. 필요한 경우 옵션을 업데이트할 수 있습니다.

신호 지정

데이터 수집 체계가 호출될 때 데이터를 수집할 신호를 지정할 수 있습니다.

> **Important**
>
> 조건 기반 수집 체계의 표현식에 사용되는 신호를 이 필드에 지정해야 합니다.

데이터를 수집할 신호를 지정하는 경우

1. 신호의 완전히 정규화된 이름을 검색합니다.

> **Note**
>
> 신호의 완전히 정규화된 이름은 신호 경로에 신호 이름을 더한 것입니다. 하위 신호를 나타내려면 점(.)을 사용합니다.
> 예를 들어 `Vehicle.Chassis.SteeringWheel.HandsOff.HandsOffSteeringState`은(는) `HandsOffSteeringState` 액추에이터의 완전한 자격 이름입니다. `Vehicle.Chassis.SteeringWheel.HandsOff.`이(가) 액추에이터의 경로입니다.

2. (선택 사항) 최대 샘플 수에는 데이터 수집 체계가 간접적으로 호출될 때 Edge Agent 소프트웨어가 수집하여 클라우드로 전송하는 최대 데이터 샘플 수를 입력합니다.
3. (선택 사항) 최소 샘플링 간격에 두 데이터 샘플 수집 이벤트 사이의 최소 시간(밀리초)을 입력합니다. 신호가 자주 바뀌는 경우 이 파라미터를 사용하여 더 느린 속도로 데이터를 수집할 수 있습니다.
4. 다른 신호를 추가하려면 신호 추가를 선택합니다. 최대 999개의 신호를 추가할 수 있습니다.
5. 다음을 선택합니다.

2단계: 스토리지 대상 정의

> **Note**
>
> 캠페인에 비전 시스템 데이터 신호가 포함된 경우에만 차량 데이터를 Amazon S3로 전송할 수 있습니다.
> 비전 시스템 데이터는 평가판 릴리스이며 변경될 수 있습니다.

캠페인에서 수집한 데이터를 저장할 대상을 선택합니다. Amazon S3 또는 Amazon Timestream으로 차량 데이터를 전송할 수 있습니다.

목적지 설정에서 다음을 수행합니다.

- 드롭다운 목록에서 S3 또는 Timestream을 선택합니다.

차량 데이터를 S3 버킷에 저장하려면 Amazon S3를 선택합니다. S3는 데이터를 버킷 내의 객체로 저장하는 객체 스토리지 서비스입니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 Amazon S3 버킷 만들기, 구성 및 작업을 참조하세요.

S3는 데이터 스토리지 비용을 최적화하고 데이터 레이크, 중앙 집중식 데이터 스토리지, 데이터 처리 파이프라인, 분석과 같은 차량 데이터를 사용하기 위한 추가 메커니즘을 제공합니다. S3를 사용하여 일괄 처리 및 분석을 위한 데이터를 저장할 수 있습니다. 예를 들어, 기계 학습(ML) 모델을 위한 하드 브레이킹 이벤트 보고서를 생성할 수 있습니다. 수신 차량 데이터는 배송 전 10분 동안 버퍼링됩니다.

Amazon S3

> **Important**
>
> AWS IoT FleetWise에 S3 버킷에 쓸 권한이 있는 경우에만 데이터를 S3로 전송할 수 있습니다. 액세스 권한 부여에 대한 자세한 내용은 AWS IoT FleetWise를 사용한 액세스 제어를 참조하세요.

S3 대상 설정에서 다음을 수행하세요.

1. S3 버킷의 경우 AWS IoT FleetWise 권한이 있는 버킷을 선택하세요.
2. (선택 사항) S3 버킷에 저장된 데이터 구성에 사용할 수 있는 사용자 지정 접두사를 입력합니다.
3. 출력 형식을 선택합니다. 출력 형식은 S3 버킷에 저장되는 형식 파일입니다.
4. S3 버킷에 저장된 데이터를.gzip 파일로 압축할지 여부를 선택합니다. 데이터를 압축하면 스토리지 비용이 최소화되므로 압축하는 것이 좋습니다.
5. S3 대상 설정에서 선택한 옵션에 따라 예제 S3 객체 URI가 변경됩니다. 다음은 S3에 어떤 파일로 저장되는지의 예입니다.

타임스트림 테이블에 차량 데이터를 저장하려면 Amazon Timestream을 선택합니다. Timestream을 사용하여 차량 데이터를 쿼리하여 추세와 패턴을 식별할 수 있습니다. 예를 들어 Timestream을 사용하여 차량 연료 수준에 대한 알람을 만들 수 있습니다. 들어오는 차량 데이터는 거의 실시간으로 Timestream으로 전송됩니다. 자세한 내용은 Amazon Timestream 개발자 가이드에서 Amazon Timestream이란 무엇인가요?를 참조하세요.

Amazon Timestream

> **Important**
>
> AWS IoT FleetWise에 Timestream에 데이터를 쓸 권한이 있는 경우에만 데이터를 테이블로 전송할 수 있습니다. 액세스 권한 부여에 대한 자세한 내용은 AWS IoT FleetWise를 사용한 액세스 제어를 참조하세요.

타임스트림 테이블 설정에서 다음을 수행합니다.

1. 타임스트림 데이터베이스 이름의 경우 드롭다운 목록에서 타임스트림 데이터베이스의 이름을 선택합니다.
2. 타임스트림 테이블 이름의 경우 드롭다운 목록에서 타임스트림 테이블의 이름을 선택합니다.

타임스트림용 서비스 액세스에서 다음을 수행합니다.

- 드롭다운 목록에서 IAM 역할을 선택합니다.

- 다음을 선택합니다.

3단계: 차량 추가

캠페인을 전개할 차량을 선택하려면 차량 목록에서 해당 차량을 선택합니다. 차량을 만들 때 추가한 속성과 값을 검색하거나 차량 이름을 기준으로 차량을 필터링할 수 있습니다.

차량 필터링에서 다음을 수행합니다.

1. 검색 상자에서 속성 또는 차량 이름을 찾아 목록에서 선택합니다.

   > **Note**
   >
   > 각 속성은 한 번만 사용할 수 있습니다.

2. 캠페인을 배포할 대상 차량 이름 또는 속성의 값을 입력합니다. 예를 들어 속성의 완전히 정규화된 이름이 `fuelType`인 경우 해당값으로 `gasoline`을(를) 입력합니다.
3. 다른 차량 속성을 검색하려면 이전 단계를 반복합니다. 최대 5개의 차량 속성과 무제한의 차량 이름을 검색할 수 있습니다.
4. 검색과 일치하는 차량이 차량 이름 아래에 나열됩니다. 캠페인을 배포할 차량을 선택합니다.

   > **Note**
   >
   > 검색 결과에는 최대 100대의 차량이 표시됩니다. 캠페인에 모든 차량을 추가하려면 모두 선택을 선택합니다.

5. 다음을 선택합니다.

4단계: 검토 및 생성

캠페인 구성을 확인한 다음 캠페인 생성을 선택합니다.

> **Note**
>
> 캠페인을 만든 후에는 사용자 또는 팀이 캠페인을 차량에 배포해야 합니다.

5단계: 캠페인을 배포합니다

캠페인을 만든 후에는 사용자 또는 팀이 캠페인을 차량에 배포해야 합니다.

캠페인을 배포하려는 경우

1. 캠페인 요약 페이지에서 배포를 선택합니다.
2. 배포를 시작하고 캠페인에 연결된 차량으로부터 데이터 수집을 시작할지 검토하고 확인합니다.
3. [배포]를 선택합니다.

캠페인에 연결된 차량의 데이터 수집을 일시 중지하려면 캠페인 요약 페이지에서 일시 중지를 선택합니다. 캠페인에 연결된 차량에서 데이터 수집을 재개하려면 재개를 선택합니다.

캠페인 생성(AWS CLI)

CreateWorkGroup API 작업을 사용하여 작업 그룹을 생성할 수도 있습니다. 다음 예제에서는 AWS CLI을(를) 사용합니다.

캠페인을 생성할 때 차량에서 수집한 데이터를 Amazon S3(S3) 또는 Amazon Timestream에 저장할 수 있습니다. Timestream을 선택하면 거의 실시간 처리가 필요한 데이터를 저장하는 등 빠르고 확장 가능하며 서버가 필요 없는 시계열 데이터베이스를 사용할 수 있습니다. 업계 최고의 확장성, 데이터 가용성, 보안, 성능을 갖춘 객체 스토리지로 S3를 선택하세요.

> **Important**
>
> AWS IoT FleetWise에 S3 또는 Timestream에 데이터를 쓸 권한이 있는 경우에만 차량 데이터를 전송할 수 있습니다. 액세스 권한 부여에 대한 자세한 내용은 AWS IoT FleetWise를 사용한 액세스 제어를 참조하세요.

캠페인 생성

> **Important**
>
> - 캠페인을 만들기 전에 신호 카탈로그와 차량 또는 플릿이 있어야 합니다. 자세한 내용은 신호 카탈로그 생성 및 관리, 차량 생성, 공급 및 관리, 플릿 생성 및 관리 단원을 참조하세요.
> - 캠페인을 생성한 후에는 UpdateCampaign API 작업을 사용하여 캠페인을 승인해야 합니다. 자세한 내용은 캠페인 업데이트 (AWS CLI) 단원을 참조하십시오.

캠페인을 생성하려면 다음 명령을 실행합니다.

*file-name*을 캠페인 구성이 포함된 JSON 파일 이름으로 교체합니다.

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

- 캠페인 이름을 만들고 있는 *campaign-name*으로 교체합니다.
- *signal-catalog-arn*을 신호 카탈로그의 Amazon 리소스 이름(ARN)으로 교체합니다.
- *target-arn*을 생성한 플릿 또는 차량의 ARN으로 교체합니다.
- *bucket-arn*을 S3 버킷의 ARN으로 바꿉니다.

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

- 캠페인 이름을 만들고 있는 *campaign-name*으로 교체합니다.
- *signal-catalog-arn*을 신호 카탈로그의 Amazon 리소스 이름(ARN)으로 교체합니다.
- *target-arn*을 생성한 플릿 또는 차량의 ARN으로 교체합니다.
- *role-arn*을 AWS IoT FleetWise에 타임스트림 테이블로 데이터를 전달할 수 있는 권한을 부여하는 작업 실행 역할의 ARN으로 교체합니다.
- *table-arn*을 타임스트림 테이블의 ARN으로 교체합니다.

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

캠페인의 논리적 표현식

AWS는 논리적 표현식을 사용하여 캠페인의 일환으로 수집할 데이터를 인식합니다. 표현식에 대한 자세한 내용은 AWS IoT Events 개발자 안내서의 표현식을 참조하세요.

표현식 변수는 수집되는 데이터 유형에 대한 규칙을 준수하도록 구성되어야 합니다. 텔레메트리 시스템 데이터의 경우 표현식 변수는 신호의 정규화된 이름이어야 합니다. 비전 시스템 데이터의 경우 표현식은 신호의 정규화된 이름을 신호의 데이터 유형에서 속성 중 하나로 이어지는 경로와 결합합니다.

예를 들어 신호 카탈로그에 다음 노드가 포함된 경우

```
{
    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"
}
```

노드가 ROS 2 정의를 따르는 경우

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

가능한 모든 이벤트 표현식 변수는 다음과 같습니다.

```
{
...
    $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`
}
```

캠페인 업데이트 (AWS CLI)

UpdateCampaign API 작업을 사용하여 기존 캠페인을 업데이트할 수 있습니다. 다음 명령은 AWS CLI을(를) 사용합니다.

- *campaign name*을 업데이트하려는 캠페인의 이름으로 교체합니다.
- *action*을 다음 중 하나로 대체합니다.
  - APPROVE— AWS IoT FleetWise가 이를 차량 또는 플릿에 배포할 수 있도록 하는 캠페인을 승인합니다.
  - SUSPEND— 캠페인을 일시 중단합니다. 캠페인이 차량에서 삭제되고 일시 중단된 캠페인에 포함된 모든 차량은 데이터 전송을 중단합니다.
  - RESUME— SUSPEND 캠페인을 다시 활성화합니다. 캠페인이 모든 차량에 재배치되고 차량이 데이터를 다시 전송합니다.
  - UPDATE— 속성을 정의하고 신호와 연결하여 캠페인을 업데이트합니다.

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

캠페인 삭제

AWS IoT FleetWise 콘솔 또는 API를 사용하여 캠페인을 삭제할 수 있습니다.

캠페인 삭제 (콘솔)

캠페인을 삭제하려면 AWS IoT FleetWise 콘솔을 사용합니다.

캠페인을 삭제하려는 경우

1. AWS IoT FleetWise 콘솔로 이동합니다.
2. 탐색 창에서 캠페인을 선택합니다.
3. 캠페인 페이지에서 대상 캠페인을 선택합니다.
4. Delete을 선택합니다.
5. 삭제 중인가요 **campaign-name**?을(를) 삭제할 캠페인을 입력한 다음 확인을 선택합니다.

캠페인 삭제(AWS CLI)

DeleteCampaign API 작업을 사용하여 캠페인을 삭제할 수 있습니다. 다음 예에는 AWS CLI이(가) 사용됩니다.

캠페인을 삭제하려면 다음 명령을 실행합니다.

*campaign-name*을 삭제하려는 차량의 이름으로 교체합니다.

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

캠페인 정보 가져오기(AWS CLI)

ListCampaign의 API 작업을 사용하여 캠페인이 삭제되었는지 확인할 수 있습니다. 다음 예제에서는 AWS CLI을(를) 사용합니다.

모든 캠페인을 대상으로 페이지가 매겨진 요약 목록을 검색하려면 다음 명령을 실행합니다.

```
aws iotfleetwise list-campaigns
```

GetCampaign API 작업을 사용하여 차량 정보를 검색할 수 있습니다. 다음 예제에서는 AWS CLI을(를) 사용합니다.

캠페인을 검색하려면 다음 명령을 실행합니다.

*campaign-name*을 검색하려는 캠페인 이름으로 교체합니다.

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

> **Note**
>
> 이 작업은 결과적 일관성을 갖습니다. 다시 말해서 캠페인을 변경하더라도 바로 반영되지 않을 수도 있습니다.

차량 데이터 처리 및 시각화

AWS IoT FleetWise용 Edge Agent는 선택된 차량 데이터를 Amazon Timestream 또는 Amazon Simple Storage Service(S3)로 전송합니다. 데이터가 데이터 목적지에 도착하면 다른 AWS 서비스를 사용하여 데이터를 시각화하고 공유할 수 있습니다.

Timestream에서 차량 데이터 처리

Timestream은 하루에 수조 개의 시계열 데이터 포인트를 저장하고 분석할 수 있는 완전 관리형 시계열 데이터베이스입니다. 데이터는 고객이 관리하는 타임스트림 테이블에 저장됩니다. Timestream을 사용하여 차량 데이터를 쿼리하여 차량에 대한 통찰력을 얻을 수 있습니다. 자세한 내용은 Amazon Timestream이란 무엇입니까?를 참조하세요

Timestream으로 전송되는 기본 데이터 스키마에는 다음 필드가 포함됩니다.

필드 이름	데이터 유형	Description
`eventId`	varchar	데이터 수집 이벤트의 ID.
`vehicleName`	varchar	데이터가 수집된 차량의 ID.
`name`	varchar	Edge Agent 소프트웨어가 데이터를 수집하는 데 사용하는 캠페인의 이름.
`time`	타임스탬프	데이터 포인트의 타임스탬프입니다.
`measure_name`	varchar	신호의 이름입니다.
`measure_value::bigint`	bigint	정수 유형의 신호 값.
`measure_value::double`	double	더블 유형의 신호 값.

필드 이름	데이터 유형	Description	
`measure_value::boolean`	boolean	부울 유형의 신호 값.	

Timestream에 저장된 차량 데이터 시각화

차량 데이터가 Timestream으로 전송되면 다음 AWS 서비스를 사용하여 데이터를 시각화, 모니터링, 분석 및 공유할 수 있습니다.

- Grafana 또는 Amazon Managed Grafana를 사용하여 대시보드의 데이터를 시각화하고 모니터링할 수 있습니다. 단일 Grafana 대시보드를 사용하여 여러 AWS 소스(예: Amazon CloudWatch 및 Timestream) 및 기타 데이터 소스의 데이터를 시각화할 수 있습니다.
- Amazon QuickSight를 사용하여 대시보드의 데이터를 분석하고 시각화합니다.

S3에서 차량 데이터 처리

Amazon S3는 원하는 양의 데이터를 저장하고 보호하는 객체 스토리지 서비스입니다. 데이터 레이크, 백업 및 복원, 아카이브, 엔터프라이즈 애플리케이션, AWS IoT 디바이스, 빅 데이터 분석 등 다양한 사용 사례에서 S3를 사용할 수 있습니다. 데이터는 S3에 버킷의 객체로 저장됩니다. 자세한 내용은 Amazon S3란 무엇인가?를 참조하세요

Amazon S3로 전송되는 기본 데이터 스키마에는 다음 필드가 포함됩니다.

필드 이름	데이터 유형	Description	
`eventId`	varchar	데이터 수집 이벤트의 ID.	
`vehicleName`	varchar	데이터가 수집된 차량의 ID.	
`name`	varchar	Edge Agent 소프트웨어가 데이터를 수집하는 데 사용하는 캠페인의 이름.	

필드 이름	데이터 유형	Description
time	타임스탬프	데이터 포인트의 타임스탬프입니다.
measure_name	varchar	신호의 이름입니다.
measure_value_BIGINT	bigint	정수 유형의 신호 값.
measure_value_DOUBLE	double	더블 유형의 신호 값.
measure_value_BOOLEAN	boolean	부울 유형의 신호 값.
measure_value_STRUCT	struct	구조체 유형의 신호 값.

S3 객체 형식

AWS IoT FleetWise는 차량 데이터를 S3로 전송하며 S3에 데이터가 객체로 저장됩니다. 데이터를 고유하게 식별하는 객체 URI를 사용하여 캠페인에서 데이터를 찾을 수 있습니다. S3 객체 URI 형식은 수집된 데이터가 비정형 데이터인지 또는 처리된 데이터인지에 따라 달라집니다.

비정형 데이터

비정형 데이터는 미리 정의되지 않은 방식으로 S3에 저장됩니다. 이미지 또는 비디오와 같은 다양한 형식일 수 있습니다.

Amazon Ion 파일의 신호 데이터와 함께 AWS IoT FleetWise로 전달된 차량 메시지는 디코딩되어 객체로 S3에 전송됩니다. S3 객체는 각 신호를 나타내며 바이너리로 인코딩됩니다.

비정형 데이터 S3 객체 URI는 다음 형식을 사용합니다.

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

처리된 데이터

처리된 데이터는 S3에 저장되며 메시지를 검증, 보강, 변환하는 처리 단계를 거칩니다. 객체 목록과 속도는 처리된 데이터의 예입니다.

S3로 전송된 데이터는 약 10분 동안 버퍼링된 레코드를 나타내는 객체로 저장됩니다. 기본적으로, AWS IoT FleetWise는 S3에 개체를 쓰기 전에 UTC 시간 접두사를 `year=YYYY/month=MM/date=DD/hour=HH` 형식으로 추가합니다. 이 접두사는 버킷에 논리적 계층 구조를 생성하는데, 계층 구조 내에서 슬래시(/) 하나당 한 계층을 생성합니다. 처리된 데이터에는 비정형 데이터에 대한 S3 객체 URI도 포함됩니다.

처리된 데이터 S3 객체 URI는 다음 형식을 사용합니다.

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

원시 데이터

프리미티브 데이터라고도 하는 원시 데이터는 Amazon Ion 파일에서 수집된 데이터입니다. 원시 데이터를 사용하여 문제를 해결하거나 오류의 근본 원인을 파악할 수 있습니다.

원시 데이터 S3 객체 URI는 다음 형식을 사용합니다.

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

S3에 저장된 차량 데이터 분석

차량 데이터가 S3로 전송된 후에는 다음 AWS 서비스를 사용하여 데이터를 모니터링, 분석 및 공유할 수 있습니다.

Amazon SageMaker를 사용하여 다운스트림 레이블 지정 및 기계 학습(ML) 워크플로를 위해 데이터를 추출 및 분석합니다.

자세한 내용은 Amazon SageMaker 개발자 안내서에서 다음 주제를 참조하세요.

- 데이터 처리
- 기계 학습 모델 훈련
- 이미지 레이블 지정

Amazon Athena를 사용하여 데이터를 AWS Glue 크롤러 카탈로그화하고 분석하십시오. 기본적으로 S3에 기록된 객체에는 키-값 쌍이 등호로 연결된 데이터 경로를 포함하는 Apache Hive 스타일 시간 파티션이 있습니다.

자세한 내용은 Amazon Athena 사용 설명서에서 다음 주제를 참조하세요.

- Athena에서 데이터 분할
- AWS Glue을(를) 사용하여 Amazon S3의 데이터 소스에 연결
- AWS Glue와(과) 함께 Athena를 사용할 때의 모범 사례

Amazon QuickSight를 사용하여 Athena 테이블 또는 S3 버킷을 직접 읽어 데이터를 시각화할 수 있습니다.

> **Tip**
>
> Amazon QuickSight는 Apache Parquet 형식을 지원하지 않으므로 S3에서 직접 읽는 경우 차량 데이터가 JSON 형식인지 확인합니다.

자세한 내용은 Amazon QuickSight 사용 설명서에서 다음 주제를 참조하세요.

- 지원되는 데이터 소스
- 데이터 소스 생성

AWS CLI 및 AWS SDK

이 섹션에서는 AWS IoT FleetWise API 요청 작성에 대해 설명합니다. AWS IoT FleetWise 작업 및 데이터 유형에 대한 자세한 내용은 AWS IoT FleetWise API 참조를 참조하세요.

다양한 프로그래밍 언어를 사용하여 AWS IoT FleetWise를 액세스하려면 다음과 같은 자동 기능을 제공하는 AWS SDK를 사용해도 됩니다.

- 서비스 요청에 대한 암호화 서명
- 요청 재시도
- 오류 응답 처리

명령줄 액세스의 경우 AWS IoT FleetWise를 AWS CLI와(과) 함께 사용합니다. 명령줄에서 AWS IoT FleetWise와 기타 서비스를 관리하고 스크립트를 통해 자동화할 수 있습니다.

AWS IoT FleetWise 문제 해결

이 섹션의 문제 해결 정보 및 솔루션을 사용하여 AWS IoT FleetWise 관련 문제를 해결합니다.

다음은 AWS IoT FleetWise에서 일반적으로 발생하는 문제를 해결하는 데 유용한 정보입니다.

주제

디코더 매니페스트 문제

디코더 매니페스트 문제를 해결합니다.

디코더 매니페스트 API 직접 호출 진단

오류	문제 해결 지침
`UpdateOperationFailure.ConflictingDecoderUpdate`	동일한 디코더 매니페스트에 여러 업데이트 요청이 있습니다. 잠시 기다렸다가 다시 시도하세요.
`UpdateOperationFailure.InternalFailure`	InternalFailure는 캡슐화된 예외로 시작됩니다. 문제 자체는 캡슐화된 예외에 따라 달라집니다.
`UpdateOperationFailure.ActiveDecoderUpdate`	디코더 매니페스트가 `Active` 상태이므로 업데이트할 수 없습니다. 디코더 매니페스트 상태를 `DRAFT`로 변경한 후 다시 시도하세요.
`UpdateOperationFailure.ConflictingModelUpdate`	AWS IoT FleetWise가 다른 사람이 수정 중인 차량 모델(모델 매니페스트)에 대해 검증을 시도하고 있습니다. 잠시 기다렸다가 다시 시도하세요.
`UpdateOperationFailure.ModelManifestValidationResponse : FailureReason.MODEL_DATA_ENTRIES_NOT_FOUND`	차량 모델에 연결된 신호가 없습니다. 차량 모델에 신호를 추가하고 연결된 신호 카탈로그에서 해당 신호를 찾을 수 있는지 확인합니다.

오류	문제 해결 지침
UpdateOperationFailure.ModelManifestValidationResponse : FailureReason.MODEL_NOT_ACTIVE	차량 모델을 업데이트하여 ACTIVE 상태가 되도록 한 후 다시 시도하세요.
UpdateOperationFailure.ModelManifestValidationResponse : FailureReason.MODEL_NOT_FOUND	AWS IoT FleetWise가 디코더 매니페스트와 연결된 차량 모델을 찾을 수 없습니다. 차량 모델의 Amazon 리소스 이름(ARN)을 확인한 후 다시 시도하세요.
UpdateOperationFailure.ModelManifestValidationResponse (FailureReason.MODEL_DATA_ENTRIES_READ_FAILURE	차량 모델의 신호 이름을 신호 카탈로그에서 찾을 수 없기 때문에 차량 모델 검증에 실패했습니다. 차량 모델의 신호가 연결된 신호 카탈로그에 모두 포함되어 있는지 확인하세요.
UpdateOperationFailure.ValidationFailure	디코더 매니페스트 업데이트 요청에서 유효하지 않은 신호 또는 네트워크 인터페이스가 발견되었습니다. 예외에서 반환된 모든 신호 및 네트워크 인터페이스가 존재하는지, 사용된 모든 신호가 사용 가능한 인터페이스와 연결되어 있는지, 연결된 신호가 있는 인터페이스를 제거하지 않을지 확인합니다.
UpdateOperationFailure.KmsKeyAccessDenied	작업에 사용된 AWS Key Management Service(AWS KMS) 키에 권한 문제가 있습니다. 키에 액세스할 수 있는 역할을 사용하고 있는지 확인하고 다시 시도하세요.
UpdateOperationFailure.DecoderDoesNotExist	디코더 매니페스트가 존재하지 않습니다. 디코더 매니페스트 이름을 확인한 후 다시 시도하세요.

SIGNAL_DECODER_INCOMPATIBLE_WITH_SIGNAL_CATALOG 이유가 있는 비전 시스템 데이터 오류 메시지에는 요청이 실패한 이유에 대한 정보를 제공하는 힌트가 응답에 포함됩니다. 힌트를 통해 따라야 할 문제 해결 지침을 결정할 수 있습니다.

> **Note**
>
> 비전 시스템 데이터는 평가판 릴리스이며 변경될 수 있습니다.

디코더 매니페스트 비전 시스템 데이터 검증 진단

오류	문제 해결 지침
`InvalidSignalDecoder.withReason(SignalDecoderFailureReason.NO_SIGNAL_IN_CATALOG_FOR_DECODER_SIGNAL)`	AWS IoT FleetWise가 신호 카탈로그에서 신호 디코더에 사용된 루트 신호 구조를 찾지 못했습니다. 구조의 루트 신호가 신호 카탈로그에 제대로 정의되어 있는지 확인하세요.
`InvalidSignalDecoder.withReason(SignalDecoderFailureReason.SIGNAL_DECODER_TYPE_INCOMPATIBLE_WITH_MESSAGE_SIGNAL_TYPE)`	신호 카탈로그의 프리미티브 메시지가 디코더 매니페스트 업데이트 요청에서 동일한 데이터 유형으로 정의되지 않았습니다. 요청에 정의된 프리미티브 메시지가 해당 신호 카탈로그 정의와 일치하는지 확인하세요.
`InvalidSignalDecoder.withReason(SignalDecoderFailureReason.STRUCT_SIZE_MISMATCH)`	신호 카탈로그의 구조체에 정의된 속성 수가 디코더 매니페스트에서 디코딩하려는 속성의 수와 일치하지 않습니다. 신호 카탈로그에 정의된 신호와 비교하여 디코딩할 신호 수가 정확한지 확인하세요.
`InvalidSignalDecoder.withReason(SignalDecoderFailureReason.SIGNAL_DECODER_INCOMPATIBLE_WITH_SIGNAL_CATALOG)`	AWS IoT FleetWise가 디코더 매니페스트 요청에 정의된 structuredMessageDefinition 없이 신호 카탈로그에 STRUCT로 정의된 신호를 발견했습니다. 디코더 매니페스트 업데이트 요청에서 각 구조체가 StructuredMessageDefinition으로 정의되었는지 확인하세요.
`InvalidSignalDecoder.withReason(SignalDecoderFailureReason.SIGNAL_DECODER_INCOMPATIBLE_WITH_SIGNAL_CATALOG)`	디코더 매니페스트에 사용된 구조의 루트 신호가 신호 카탈로그에서 구조로 제대로 정의되지 않았습니다. 디코더 매니페스트에 사용되는 루트 신호 구조에는 해당 StructFullyQualifiedName 필드가 정의되어 있어야 합니다. 또한

오류	문제 해결 지침
	fullyQualifiedName이 있는 STRUCT 노드가 필요합니다.
`InvalidSignalDecoder.withReason(SignalDecoderFailureReason.SIGNAL_DECODER_INCOMPATIBLE_WITH_SIGNAL_CATALOG)`	디코더 매니페스트 요청에 사용된 리프 메시지 중 하나가 프리미티브 메시지로 정의되지 않았습니다. 요청의 모든 리프 객체가 프리미티브 메시지로 정의되었는지 확인하세요.
`InvalidSignalDecoder.withReason(SignalDecoderFailureReason.SIGNAL_DECODER_INCOMPATIBLE_WITH_SIGNAL_CATALOG)`	신호 카탈로그의 배열 객체가 디코더 매니페스트 업데이트 요청에서 structuredMessageListDefinition으로 정의되지 않았습니다. 디코더 매니페스트 업데이트 요청에서 모든 배열 속성이 structuredMessageListDefinition으로 정의되었는지 확인하세요.

AWS IoT용 FleetWise 소프트웨어 Edge Agent 문제

Edge Agent 소프트웨어 문제를 해결합니다.

문제

- 문제: Edge Agent 소프트웨어가 시작되지 않습니다.
- 문제: [ERROR] [IoT FleetwiseEngine::connect]: [지속성 라이브러리 초기화 실패]
- 문제: Edge Agent 소프트웨어가 온보드 진단(OBD) II PID 및 진단 문제 코드(DTC)를 수집하지 않습니다.
- 문제: AWS IoT FleetWise용 Edge Agent 소프트웨어가 네트워크에서 데이터를 수집하지 않거나 데이터 검사 규칙을 적용할 수 없습니다.
- 문제: [ERROR] [AwsIotConnectivityModule::connect]: [오류로 인한 연결 실패] 또는 [WARN] [AwsIotChannel::send]: [활성 MQTT 연결이 없습니다.]

문제: Edge Agent 소프트웨어가 시작되지 않습니다.

Edge Agent 소프트웨어가 시작되지 않을 경우 다음 오류가 표시될 수 있습니다.

- ```
  Error from reader: * Line 1, Column 1
  ```
  ```

```
Syntax error: value, object or array expected.
```

해결 방법: AWS IoT용 Edge Agent FleetWise의 소프트웨어 구성 파일이 유효한 JSON 형식을 사용하고 있는지 확인합니다. 예를 들어, 쉼표가 올바르게 사용되었는지 확인합니다. 구성 파일에 대한 자세한 내용을 보려면 다음을 수행하여 AWSIoT 소프트웨어용 Edge Agent 개발자 가이드를 다운로드합니다.

1. AWS IoT FleetWise 콘솔로 이동합니다.
2. 서비스 홈 페이지의 AWS IoT FleetWise 시작하기 섹션에서 Edge Agent 탐색을 선택합니다.

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

해결 방법: Edge Agent 소프트웨어가 구성 파일에 정의된 네트워크 인터페이스와의 소켓 통신을 설정하지 못할 경우 이 오류가 표시될 수 있습니다.

구성에 정의된 모든 네트워크 인터페이스를 사용할 수 있는지 확인하려면 다음 명령을 실행합니다.

```
ip link show
```

네트워크 인터페이스를 온라인 상태로 전환하려면 다음 명령을 실행합니다. *network-interface-id*를 네트워크 인터페이스의 ID로 교체하세요.

```
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]
  ```

해결 방법: Edge Agent 소프트웨어에서 AWS IoT Core(으)로 MQTT 연결을 설정하지 못하면 이 오류가 표시될 수 있습니다. 다음이 올바르게 구성되었는지 확인하고 Edge Agent 소프트웨어를 다시 시작합니다.

- `mqttConnection::endpointUrl`— AWS 계정의 IoT 디바이스 엔드포인트.
- `mqttConnection::clientID`— Edge Agent 소프트웨어가 실행 중인 차량의 ID.
- `mqttConnection::certificateFilename`— 차량 인증서 파일의 경로입니다.
- `mqttConnection::privateKeyFilename`— 차량 개인 키 파일의 경로.

- 차량을 프로비저닝하는 데 AWS IoT Core을(를) 사용했습니다. 자세한 내용은 차량 공급 섹션을 참조하세요.

자세한 내용은 AWS IoT Device SDK for C++FAQ를 참조하세요.

문제: [ERROR] [IoT FleetwiseEngine::connect]: [지속성 라이브러리 초기화 실패]

해결 방법: Edge Agent 소프트웨어가 지속성 저장소를 찾지 못할 경우 이 오류가 표시될 수 있습니다. 다음이 올바르게 구성되었는지 확인하고 Edge Agent 소프트웨어를 다시 시작합니다.

`persistency:persistencyPath`— 수집 체계, 디코더 매니페스트 및 데이터 스냅샷을 유지하는 데 사용되는 로컬 경로입니다.

문제: Edge Agent 소프트웨어가 온보드 진단(OBD) II PID 및 진단 문제 코드(DTC)를 수집하지 않습니다.

해결 방법: `obdInterface:pidRequestIntervalSeconds` 또는 `obdInterface:dtcRequestIntervalSeconds`이(가) 0(으)로 구성된 경우 이 오류가 표시될 수 있습니다.

Edge Agent 소프트웨어가 자동 변속기 차량에서 실행 중인 경우 `obdInterface:hasTransmissionEcu`이(가) `true`(으)로 구성되어 있는지 확인합니다.

차량이 확장된 컨트롤러 영역 네트워크(CAN 버스) 중재 ID를 지원하는 경우 지원하도록 `obdInterface:useExtendedIds`이(가) `true`(으)로 구성되어 있는지 확인합니다.

문제: AWS IoT FleetWise용 Edge Agent 소프트웨어가 네트워크에서 데이터를 수집하지 않거나 데이터 검사 규칙을 적용할 수 없습니다.

해결 방법: 기본 할당량을 위반한 경우 이 오류가 표시될 수 있습니다.

Resource	할당량	조정 가능	참고
신호의 ID의 값입니다	신호의 ID는 50,000보다 작거나 같아야 합니다	예	Edge Agent 소프트웨어는 ID가 50,000을 초과하는 신호에서 데이

Resource	할당량	조정 가능	참고
			터를 수집하지 않습니다. 이 할당량을 변경하기 전에 신호 카탈로그에 포함된 신호 수를 확인하는 것이 좋습니다.
차량당 활성 데이터 수집 체계의 수	256	예	이 할당량을 변경하기 전에 클라우드에서 만든 캠페인 수와 각 캠페인에 포함된 스키마 수를 확인하는 것이 좋습니다.
신호의 히스토리 버퍼의 크기입니다	20MB	예	할당량이 초과되면 Edge Agent 소프트웨어는 새 데이터 수집을 중단합니다.

문제: [ERROR] [AwsIotConnectivityModule::connect]: [오류로 인한 연결 실패] 또는 [WARN] [AwsIotChannel::send]: [활성 MQTT 연결이 없습니다.]

해결 방법: Edge Agent 소프트웨어가 클라우드에 연결되지 않은 경우 이 오류가 표시될 수 있습니다. 기본적으로 Edge Agent 소프트웨어는 1분마다 핑 요청을 AWS IoT Core에게 보내고 3분 동안 기다립니다. 응답이 없으면 Edge Agent 소프트웨어가 자동으로 클라우드 연결을 다시 설정합니다.

AWS IoT에서의 보안 FleetWise

클라우드 AWS 보안이 최우선 과제입니다. AWS 고객은 가장 보안에 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 혜택을 누릴 수 있습니다.

보안은 기업과 기업 간의 AWS 공동 책임입니다. 공동 책임 모델은 클라우드 보안 및 클라우드의 보안으로 그 책임을 설명합니다.

- 클라우드 보안 - AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호하는 역할을 합니다. AWS AWS 또한 안전하게 사용할 수 있는 서비스를 제공합니다. AWS FleetWiseIoT에 적용되는 규정 준수 프로그램에 대해 자세히 알아보려면 규정 준수 프로그램 제공 AWS 범위 내 서비스 규정 준수 프로그램이 참조하십시오.
- 클라우드에서의 보안 — 귀하의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 여러분은 데이터의 민감도, 회사 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다

이 설명서는 AWS IoT를 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 FleetWise 됩니다. 보안 및 규정 준수 목표를 FleetWise 충족하도록 AWS IoT를 구성하는 방법을 보여줍니다. 또한 AWS IoT FleetWise 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법도 알아봅니다.

내용

AWS IoT에서의 데이터 보호 FleetWise

AWS 공동 책임 모델 AWS IoT의 데이터 보호에 적용됩니다 FleetWise. 이 모델에 설명된 대로 AWS 는 모든 것을 실행하는 글로벌 인프라를 보호하는 역할을 AWS 클라우드합니다. 이 인프라에서 호스팅

되는 콘텐츠에 대한 제어를 유지하는 것은 사용자의 책임입니다. 사용하는 AWS 서비스 의 보안 구성과 관리 작업에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

데이터 보호를 위해 AWS 계정 자격 증명을 보호하고 AWS IAM Identity Center OR AWS Identity and Access Management (IAM) 을 사용하여 개별 사용자를 설정하는 것이 좋습니다. 이 방식을 사용하면 각 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용합니다.
- SSL/TLS를 사용하여 리소스와 통신하세요. AWS TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다. AWS CloudTrail
- 포함된 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용하십시오 AWS 서비스.
- Amazon Macie와 같은 고급 관리형 보안 서비스를 사용하여 Amazon S3에 저장된 민감한 데이터를 검색하고 보호합니다.
- 명령줄 인터페이스 또는 API를 AWS 통해 액세스할 때 FIPS 140-2로 검증된 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용하십시오. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 Federal Information Processing Standard(FIPS) 140-2를 참조하십시오.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 양식 필드에 입력하지 않는 것이 좋습니다. 여기에는 콘솔 AWS CLI, API FleetWise 또는 AWS 서비스 AWS SDK를 사용하여 AWS IoT 또는 기타 작업을 하는 경우가 포함됩니다. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 보안 인증 정보를 URL에 포함시켜서는 안 됩니다.

AWS FleetWise IoT는 차량 데이터를 AWS 클라우드로 전송하기 위해 지원되는 차량 하드웨어에 개발하고 설치하는 Edge Agent와 함께 사용하기 위한 것입니다. 차량에서 데이터를 추출하는 경우 특정 관할 지역에서는 데이터 개인 정보 보호 규정의 적용 대상일 수 있습니다. AWS FleetWise IoT를 사용하고 Edge Agent를 설치하기 전에 관련 법률에 따른 규정 준수 의무를 평가하는 것이 좋습니다. 여기에는 모든 관련 법적 요건을 포함하고 있으며 이는 법적으로 적절한 개인정보 고지를 제공하고 차량 데이터 추출에 필요한 법적 동의를 획득하기 위한 용도로 사용됩니다.

저장 중 암호화

차량에서 수집된 데이터는 MQTT 메시지 프로토콜이 포함된 AWS IoT Core 메시지를 통해 클라우드로 전송됩니다. AWS FleetWise IoT는 Amazon Timestream 데이터베이스로 데이터를 전송합니다. Timestream 에서는 데이터가 암호화됩니다. 모든 데이터는 기본적으로 유휴 데이터를 AWS 서비스 암호화합니다.

저장 중 암호화는 AWS Key Management Service (AWS KMS) 와 통합되어 데이터를 암호화하는 데 사용되는 암호화 키를 관리합니다. 고객 관리 키를 사용하여 AWS FleetWise IoT에서 수집한 데이터를 암호화하도록 선택할 수 있습니다. 를 통해 AWS KMS암호화 키를 만들고, 관리하고, 볼 수 있습니다. 자세한 내용은 What is AWS Key Management Service? 를 참조하십시오. AWS Key Management Service 개발자 안내서에서.

전송 중 암호화

AWS IoT 서비스와 교환되는 모든 데이터는 전송 계층 보안 (TLS) 을 사용하여 전송 중에 암호화됩니다. 자세한 정보는 AWS IoT 개발자 가이드에서 전송 보안을 참조하세요.

또한 AWS IoT FleetWise 리소스에 대한 액세스를 안전하게 제어할 수 있도록 인증 및 권한 부여를 AWS IoT Core 지원합니다. 차량은 X.509 인증서를 사용하여 AWS IoT를 사용하도록 인증 (로그인) FleetWise 하고 AWS IoT Core 정책을 사용하여 지정된 작업을 수행할 수 있는 권한 (권한 보유) 을 받을 수 있습니다. 자세한 정보는 the section called “차량 공급”을 참조하세요.

데이터 암호화

데이터 암호화란 전송 중 (AWS IoT 송수신 FleetWise, 게이트웨이와 서버 간 이동) 및 저장 중 (로컬 장치 또는 내부에 저장된 데이터) 을 보호하는 것을 말합니다. AWS 서비스클라이언트측 암호화를 사용하여 저장 데이터를 보호할 수 있습니다.

> **Note**
>
> AWS IoT FleetWise 엣지 프로세싱은 AWS IoT FleetWise 게이트웨이 내에 호스팅되고 로컬 네트워크를 통해 액세스할 수 있는 API를 노출합니다. 이러한 API는 IoT AWS Edge 커넥터가 소유한 서버 인증서를 기반으로 하는 TLS 연결을 통해 노출됩니다. FleetWise 클라이언트 인증의 경우 이러한 API는 액세스 제어 암호를 사용합니다. 서버 인증서 개인 키와 액세스 제어 암호는 모두 디스크에 저장됩니다. AWS IoT FleetWise Edge 프로세싱은 저장된 이러한 자격 증명의 보안을 위해 파일 시스템 암호화를 사용합니다.

서버 측 암호화 및 클라이언트측 암호화에 대한 자세한 내용은 다음 주제를 검토하십시오.

내용

저장 중 암호화

AWS IoT는 데이터를 AWS 클라우드와 게이트웨이에 FleetWise 저장합니다.

클라우드에 AWS 저장된 데이터

AWS IoT는 기본적으로 저장된 데이터를 AWS 서비스 암호화하는 다른 곳에 데이터를 FleetWise 저장합니다. 저장 중 암호화는 IoT에서 자산 자산 자산 및 집계 값을 암호화하는 데 사용되는 암호화 키를 관리하기 위해 AWS Key Management Service (AWS KMS) 와 통합됩니다. AWS FleetWise 고객 관리 키를 사용하여 AWS IoT에서 자산 자산 자산 값 및 집계 값을 암호화하도록 선택할 수 있습니다. FleetWise 를 통해 암호화 키를 만들고, 관리하고, 볼 수 있습니다. AWS KMS

데이터를 암호화할 키 AWS 소유 키 또는 고객 관리 키를 선택할 수 있습니다.

작동 방식

저장 중 암호화는 데이터 AWS KMS 암호화에 사용되는 암호화 키 관리 기능과 통합됩니다.

- AWS 소유 키 — 기본 암호화 키. AWS FleetWise IoT가 이 키를 소유합니다. AWS 계정의 키를 확인, 관리 또는 사용할 수 없습니다. 또한 AWS CloudTrail 로그에서 키에 대한 작업을 볼 수 없습니다. 추가 비용 없이 이 키를 사용할 수 있습니다.
- 고객 관리형 키 - 사용자의 계정에 키가 저장되며 사용자가 생성, 소유, 관리하는 유형입니다. KMS 키에 대해 사용자가 모든 것을 제어합니다. 추가 AWS KMS 요금이 적용됩니다.

AWS 소유 키

AWS 소유 키 계정에 저장되지 않습니다. 여러 키에서 사용할 수 있도록 AWS 소유하고 관리하는 KMS 키 컬렉션의 일부입니다. AWS 계정 AWS 서비스 데이터를 보호하는 AWS 소유 키 데 사용할 수 있습니다.

데이터 사용을 AWS 소유 키보거나, 관리하거나, 사용하거나 감사할 수 없습니다. 하지만 데이터를 암호화하는 키를 보호하기 위해 어떤 작업을 수행하거나 어떤 프로그램을 변경할 필요가 없습니다.

사용하면 AWS 소유 키수수료가 부과되지 않으며 계정 AWS KMS 할당량에도 포함되지 않습니다.

고객 관리형 키

고객 관리형 키는 사용자가 생성, 소유 및 관리하는 계정의 KMS 키입니다. 다음과 같이 이러한 KMS 키를 완전히 제어할 수 있습니다.

- 키 정책, IAM 정책 및 권한 부여의 수립 및 유지
- 활성화 및 비활성화
- 암호화 자료 교체
- 태그 추가
- 이를 참조하는 별칭 생성
- KMS 키 삭제 예약

또한 Amazon CloudWatch Logs를 사용하여 CloudTrail AWS IoT가 사용자를 대신하여 FleetWise 보내는 요청을 추적할 AWS KMS 수 있습니다.

고객 관리 키를 사용하는 경우 계정에 저장된 KMS 키에 AWS IoT FleetWise 액세스 권한을 부여해야 합니다. AWS IoT는 봉투 암호화와 키 계층 구조를 FleetWise 사용하여 데이터를 암호화합니다. AWS KMS 암호화 키는 이 키 계층 구조의 루트 키를 암호화하는 데 사용됩니다. 자세한 내용은 AWS Key Management Service 개발자 가이드의 봉투 암호화를 참조하세요.

다음 예제 정책은 사용자를 대신하여 고객 관리 키를 생성할 수 있는 AWS IoT FleetWise 권한을 부여합니다.

```
{
  "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 키 정책에 새 섹션을 추가할 때 정책의 기존 섹션을 변경하지 마세요. AWS IoT에 암호화가 활성화되어 FleetWise 있고 다음 중 하나에 해당하는 경우 AWS IoT는 데이터에 대한 작업을 수행할 FleetWise 수 없습니다.
>
> - KMS 키가 비활성화되었거나 삭제되었습니다.
> - KMS 키 정책이 서비스에 제대로 구성되어 있지 않습니다.

비전 시스템 데이터에 저장 시 암호화 사용

> **Note**
>
> 비전 시스템 데이터는 평가판 릴리스이며 변경될 수 있습니다.

AWS IoT FleetWise 계정에서 AWS KMS 키가 활성화된 고객 관리형 암호화가 있고 비전 시스템 데이터를 사용하려는 경우 복잡한 데이터 유형과 호환되도록 암호화 설정을 재설정하십시오. 이를 통해 AWS IoT는 FleetWise 비전 시스템 데이터에 필요한 추가 권한을 설정할 수 있습니다.

> **Note**
>
> 비전 시스템 데이터에 대한 암호화 설정을 재설정하지 않은 경우 디코더 매니페스트가 검증 중 상태로 멈출 수 있습니다.

1. GetEncryptionConfigurationAPI 작업을 사용하여 AWS KMS 암호화가 활성화되었는지 확인하십시오. 암호화 유형이 `FLEETWISE_DEFAULT_ENCRYPTION`이면 추가 작업이 필요하지 않습니다.
2. 암호화 유형이 `KMS_BASED_ENCRYPTION` 인 경우 PutEncryptionConfigurationAPI 작업을 사용하여 암호화 유형을 로 재설정합니다`FLEETWISE_DEFAULT_ENCRYPTION`.

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

```
}
```

3. PutEncryptionConfigurationAPI 작업을 사용하여 암호화 유형을 로 KMS_BASED_ENCRYPTION 다시 활성화합니다.

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

암호화 활성화에 대한 자세한 내용은 키 관리 섹션을 참조하세요.

키 관리

AWS IoT FleetWise 클라우드 키 관리

기본적으로 AWS IoT는 에서 데이터를 보호하는 AWS 관리형 키 데 FleetWise 사용합니다 AWS 클라우드. 고객 관리 키를 사용하여 AWS FleetWise IoT에서 데이터를 암호화하도록 설정을 업데이트할 수 있습니다. AWS Key Management Service (AWS KMS) 를 통해 암호화 키를 만들고, 관리하고, 볼 수 있습니다.

AWS IoT는 다음 리소스의 데이터를 암호화하기 AWS KMS 위해 고객 관리 키를 저장하여 서버 측 암호화를 FleetWise 지원합니다.

AWS IoT FleetWise 리소스	데이터 유형	고객 관리 키로 유휴 상태에서 암호화된 필드
신호 카탈로그		설명
	속성	description, allowedValues, defaultValue, min, max
	액추에이터	description, allowedValues, min, max
	센서	description, allowedValues, min, max
차량 모델(모델 매니페스트)		설명

AWS IoT FleetWise 리소스	데이터 유형	고객 관리 키로 유휴 상태에서 암호화된 필드
디코더 매니페스트		설명
	CanInterface	protocolName, protocolVersion
	ObdInterface	requestMessageId, dtcRequestInterval 세컨즈, hasTransmissionEcu, OBD 스탠다드, 세컨즈, pidRequestInterval useExtendedIds
	CanSignal	계수 isBigEndian, 부호, 길이, 메시지 ID, 오프셋, 시작 비트
	ObdSignal	바이트 길이, 오프셋, pid, 스케일링, 서비스 모드, 시작 바이트, pidResponseLength bitMaskLength bitRightShift
차량		attributes
Campaign		설명
	conditionBasedCollection스킴	익스프레션 conditionLanguageVersion, minimumTriggerInterval MS, 트리거 모드
	TimeBasedCollectionScheme	periodMs

> **Note**
>
> 기타 데이터 및 리소스는 AWS IoT에서 관리하는 키를 사용한 기본 암호화를 사용하여 암호화됩니다 FleetWise. 이 키는 AWS IoT FleetWise 계정에 생성되어 저장됩니다.

자세한 내용은 AWS Key Management Service무엇입니까를 참조하십시오. AWS Key Management Service 개발자 안내서에서.

KMS 키를 사용한 암호화 활성화 (콘솔)

고객 관리 키를 IoT에서 사용하려면 AWS AWS IoT FleetWise FleetWise 설정을 업데이트해야 합니다.

KMS 키를 사용하여 암호화를 활성화하려는 경우 (콘솔)

1. AWS IoT FleetWise 콘솔을 엽니다.
2. 설정으로 이동합니다.
3. 암호화에서 편집을 선택하여 암호화 편집 페이지를 엽니다.
4. 암호화 키 유형에서 다른 AWS KMS 키 선택을 선택합니다. 이렇게 하면 AWS KMS에 저장된 고객 관리 키로 암호화할 수 있습니다.

> **Note**
>
> 고객 관리형 키 암호화는 AWS IoT FleetWise 리소스에만 사용할 수 있습니다. 여기에는 신호 카탈로그, 차량 모델(모델 매니페스트), 디코더 매니페스트, 차량, 플릿 및 캠페인이 포함됩니다.

5. 다음 옵션 중 하나를 선택하여 KMS 키를 사용합니다.
   - 기존 KMS 키를 사용하려면 — 목록에서 KMS 키별칭을 선택합니다.
   - 새 KMS 키를 만들려면 — 키 생성을 선택합니다. AWS KMS

   > **Note**
   >
   > 그러면 AWS KMS 콘솔이 열립니다. KMS 키 생성에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 키 생성을 참조하세요.

6. 설정을 저장하려면 저장을 선택합니다.

KMS 키를 사용한 암호화 활성화(AWS CLI)

PutEncryptionConfigurationAPI 작업을 사용하여 AWS IoT FleetWise 계정의 암호화를 활성화할 수 있습니다. 다음 예에서는 를 사용합니다 AWS CLI.

암호화를 활성화하려면 다음 명령을 실행합니다.

- *KMS # idj*를 KMS 키의 ID로 교체합니다.

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

Example 응답

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

KMS 키 정책

KMS 키를 만든 후에는 최소한 다음 설명을 KMS 키 정책에 추가해야 AWS IoT에서 작동할 수 있습니다. 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 FleetWiseIoT와 함께 사용할 KMS 키 정책을 편집하는 방법에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 키 정책 변경을 참조하십시오.

> **Important**
>
> KMS 키 정책에 새 섹션을 추가할 때는 정책의 기존 섹션을 변경하지 마십시오. AWS IoT에 암호화가 활성화되어 FleetWise 있고 다음 중 하나에 해당하는 경우 AWS IoT는 데이터에 대한 작업을 수행할 FleetWise 수 없습니다.
>
> - KMS 키가 비활성화되었거나 삭제되었습니다.
> - KMS 키 정책이 서비스에 제대로 구성되어 있지 않습니다.

다음을 통한 AWS IoT FleetWise액세스 제어

다음 섹션에서는 리소스에 대한 액세스 및 AWS IoT FleetWise 리소스 액세스를 제어하는 방법을 다룹니다. 다루는 정보에는 캠페인 중에 AWS IoT가 차량 데이터를 전송할 FleetWise 수 있도록 애플리케이션 액세스 권한을 부여하는 방법이 포함됩니다. 또한 데이터를 저장하기 위해 Amazon S3 (S3) 버킷 또는 Amazon Timestream 데이터베이스 및 테이블에 AWS IoT FleetWise 대한 액세스 권한을 부여하는 방법도 설명합니다.

이러한 모든 형태의 액세스를 관리하는 기술은 AWS Identity and Access Management (IAM) 입니다. IAM에 대한 자세한 내용은 IAM이란? 섹션을 참조하세요.

내용

- Amazon S3 대상에 AWS IoT FleetWise 대한 액세스 권한 부여
- Amazon Timestream 대상에 AWS IoT FleetWise 대한 액세스 권한 부여

Amazon S3 대상에 AWS IoT FleetWise 대한 액세스 권한 부여

Amazon S3 대상을 사용하면 차량 데이터를 S3 버킷으로 전송하고 선택적으로 데이터 암호화를 위해 소유한 AWS KMS 키를 사용할 수 있습니다. AWS IoT FleetWise 오류 로깅이 활성화된 경우 CloudWatch 로그 그룹 및 스트림에 데이터 전송 AWS IoT FleetWise 오류도 전송합니다. 전송 스트림을 생성할 때 IAM 역할을 보유하고 있어야 합니다.

AWS IoT FleetWise S3 대상의 서비스 보안 주체와 함께 버킷 정책을 사용합니다. 버킷 정책 추가에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 Amazon S3 콘솔을 사용하여 버킷 정책 추가를 참조하세요.

다음 액세스 정책을 사용하여 S3 AWS IoT FleetWise 버킷에 액세스할 수 있도록 설정합니다. S3 버킷을 소유하지 않은 경우 Amazon S3 작업 목록에 `s3:PutObjectAcl`을(를) 추가합니다. 이렇게 하면

버킷 소유자에게 에서 제공한 객체에 대한 전체 액세스 권한이 부여됩니다 AWS IoT FleetWise. 버킷의 객체 내 액세스 보호 방법에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 버킷 정책 예제를 참조하세요.

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

다음 버킷 정책은 AWS 지역 내 계정의 모든 캠페인에 적용됩니다.

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

KMS 키가 S3 버킷에 연결되어 있는 경우 KMS 키에 다음 정책이 필요합니다. 키 관리에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 AWS Key Management Service 키를 사용한 서버 측 암호화 (SSE-KMS) 를 사용한 데이터 보호를 참조하십시오.

```
{
```

```
  "Version": "2012-10-17",
  "Effect": "Allow",
  "Principal": {
    "Service": "iotfleetwise.amazonaws.com"
  },
  "Action": [
    "kms:GenerateDataKey",
    "kms:Decrypt"
   ],
  "Resource": "key-arn"
}
```

Important

버킷을 생성하면 S3가 리소스에 대한 모든 권한을 리소스 소유자에게 부여하는 기본 액세스 제어 목록(ACL)을 생성합니다. AWS IoT가 S3에 데이터를 전송할 FleetWise 수 없는 경우 S3 버킷에서 ACL을 비활성화해야 합니다. 자세한 내용은 Amazon Simple Storage Service(S3) 사용 설명서의 모든 새 버킷에 대해 ACL 사용 중지 및 객체 소유권 시행을 참조하세요.

Amazon Timestream 대상에 AWS IoT FleetWise 대한 액세스 권한 부여

타임스트림 목적지를 사용하면 차량 데이터를 타임스트림 테이블로 전송합니다. AWS IoT FleetWise Timestream으로 데이터를 AWS IoT FleetWise 전송하려면 정책을 IAM 역할에 연결해야 합니다.

콘솔을 사용하여 캠페인을 만들면 AWS IoT가 필요한 정책을 역할에 FleetWise 자동으로 연결합니다.

시작하기 전에 다음 사항에 유의하세요.

Important

- AWS IoT용 FleetWise Timestream 리소스를 생성할 때는 동일한 AWS 지역을 사용해야 합니다. AWS 지역을 전환하면 Timestream 리소스에 액세스하는 데 문제가 있을 수 있습니다.
- AWS FleetWise IoT는 미국 동부 (버지니아 북부) 와 유럽 (프랑크푸르트) 에서 사용할 수 있습니다.
- 지원되는 리전의 전체 목록은 AWS 일반 참조에서 Timestream 엔드포인트 및 할당량을 참조하세요.

- Timestream 데이터베이스가 있어야 합니다. 튜토리알의 경우, Amazon Timestream 개발자 안내서의 데이터베이스 생성을 참조하세요.
- 지정된 Timestream 데이터베이스에 테이블을 생성해야 합니다. 튜토리알의 경우, Amazon Timestream 개발자 가이드의 테이블 생성을 참조하세요.

를 사용하여 AWS CLI Timestream에 대한 신뢰 정책이 포함된 IAM 역할을 생성할 수 있습니다. IAM 역할을 생성하는 경우, 다음 명령을 실행합니다.

신뢰 정책으로 IAM 역할을 생성하려는 경우

- 생성 중인 역할의 *TimestreamExecutionRole*이름으로 바꾸십시오.
- *trust-policy*를 신뢰 정책을 포함한 JSON 파일로 교체합니다.

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

AWS IoT에 Timestream에 데이터를 쓸 수 있는 FleetWise 권한을 부여하는 권한 정책을 생성합니다. 서비스 역할 권한이 있는 정책을 만들려면 다음 명령을 실행합니다.

권한 정책 생성을 생성하려는 경우

- 생성 중인 정책의 *AWSIoTFleetwiseAccessTimestreamPermissionsPolicy*이름으로 바꾸십시오.
- *permissions-policy*을 권한 정책이 포함된 JSON 파일 이름으로 교체합니다.

```
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": "*"
    }
  ]
}
```

IAM 역할에 권한 정책을 연결하려는 경우

1. 출력에서 권한 정책의 Amazon 리소스 이름(ARN)을 복사합니다.
2. IAM 역할에 IAM 권한 정책을 연결하려면 다음 명령을 실행합니다.

- 이전 단계에서 복사한 *permissions-policy-arn*ARN으로 교체합니다.
- 생성한 IAM 역할의 *TimestreamExecutionRole*이름으로 바꾸십시오.

```
aws iam attach-role-policy --policy-arn permissions-policy-arn --role-
name TimestreamExecutionRole
```

더 많은 예시 정책은 IAM 사용 설명서IAM 사용자 안내서의 AWS 리소스에 대한 액세스 관리를 참조하세요.

AWS IoT를 위한 Identity 및 Access Management FleetWise

AWS Identity and Access Management (IAM) 은 관리자가 리소스에 대한 액세스를 안전하게 제어할 수 AWS 서비스 있도록 AWS 도와줍니다. IAM 관리자는 AWS IoT FleetWise 리소스를 사용할 수 있는 인증 (로그인) 및 권한 부여 (권한 보유) 할 수 있는 사용자를 제어합니다. IAM은 추가 AWS 서비스 비용 없이 사용할 수 있습니다.

주제

- 고객
- 자격 증명을 통한 인증
- 정책을 사용한 액세스 관리
- AWS IoT와 FleetWise IAM의 작동 방식
- IoT를 위한 ID 기반 정책 예제 AWS FleetWise
- AWS IoT FleetWise ID 및 액세스 문제 해결

고객

사용 방법 AWS Identity and Access Management (IAM) 은 AWS IoT에서 수행하는 작업에 따라 다릅니다. FleetWise

서비스 사용자 - AWS IoT FleetWise 서비스를 사용하여 작업을 수행하는 경우 관리자가 필요한 자격 증명과 권한을 제공합니다. 더 많은 AWS IoT FleetWise 기능을 사용하여 작업을 수행함에 따라 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방식을 이해하면 적절한 권한을 관리자에게 요청할 수

있습니다. AWS FleetWiseIoT의 기능에 액세스할 수 없는 경우 을 참조하십시오AWS IoT FleetWise ID 및 액세스 문제 해결.

서비스 관리자 — 회사에서 AWS IoT FleetWise 리소스를 담당하고 있다면 AWS IoT에 완전히 액세스할 수 있을 것입니다 FleetWise. 서비스 사용자가 액세스해야 하는 AWS IoT FleetWise 기능 및 리소스를 결정하는 것은 여러분의 몫입니다. 그런 다음, IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해하십시오. 회사에서 AWS FleetWise IoT와 함께 IAM을 사용하는 방법에 대해 자세히 알아보려면 을 참조하십시오AWS IoT와 FleetWise IAM의 작동 방식.

IAM 관리자 — IAM 관리자라면 IoT AWS 액세스를 관리하는 정책을 작성하는 방법에 대해 자세히 알아보는 것이 좋습니다. FleetWise IAM에서 사용할 수 있는 AWS IoT FleetWise ID 기반 정책의 예를 보려면 을 참조하십시오. IoT를 위한 ID 기반 정책 예제 AWS FleetWise

자격 증명을 통한 인증

인증은 ID 자격 증명을 AWS 사용하여 로그인하는 방법입니다. IAM 사용자로 인증 (로그인 AWS) 하거나 IAM 역할을 맡아 인증 (로그인) 해야 합니다. AWS 계정 루트 사용자

ID 소스를 통해 제공된 자격 증명을 사용하여 페더레이션 ID로 로그인할 수 있습니다. AWS AWS IAM Identity Center (IAM ID 센터) 사용자, 회사의 싱글 사인온 인증, Google 또는 Facebook 자격 증명이 페더레이션 ID의 예입니다. 연합형 ID로 로그인할 때 관리자가 이전에 IAM 역할을 사용하여 ID 연합을 설정했습니다. 페더레이션을 사용하여 액세스하는 경우 AWS 간접적으로 역할을 맡게 됩니다.

사용자 유형에 따라 AWS Management Console 또는 AWS 액세스 포털에 로그인할 수 있습니다. 로그인에 대한 자세한 내용은 AWS 로그인 사용 설명서의 내 로그인 방법을 참조하십시오. AWS AWS 계정

AWS 프로그래밍 방식으로 액세스하는 경우 자격 증명을 사용하여 요청에 암호화 방식으로 서명할 수 있는 소프트웨어 개발 키트 (SDK) 와 명령줄 인터페이스 (CLI) 를 AWS 제공합니다. AWS 도구를 사용하지 않는 경우 요청에 직접 서명해야 합니다. 권장 방법을 사용하여 직접 요청에 서명하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 AWS API 요청 서명을 참조하십시오.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어, AWS 계정의 보안을 강화하기 위해 다단계 인증 (MFA) 을 사용할 것을 권장합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서의 다중 인증 및 IAM 사용 설명서의 AWS에서 다중 인증(MFA) 사용을 참조하십시오.

AWS 계정 루트 사용자

계정을 AWS 계정만들 때는 먼저 계정의 모든 AWS 서비스 리소스에 대한 완전한 액세스 권한을 가진 하나의 로그인 ID로 시작합니다. 이 ID를 AWS 계정 루트 사용자라고 하며, 계정을 만들 때 사용한 이

메일 주소와 비밀번호로 로그인하여 액세스할 수 있습니다. 일상적인 작업에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 보안 인증 정보를 보호하고 루트 사용자만 수행할 수 있는 작업을 수행하는 데 사용합니다. 루트 사용자로 로그인해야 하는 태스크의 전체 목록은 IAM 사용 설명서의 루트 사용자 보안 인증이 필요한 태스크를 참조하세요.

연동 보안 인증

가장 좋은 방법은 관리자 액세스가 필요한 사용자를 비롯한 수동 AWS 서비스 사용자가 ID 공급자와의 페더레이션을 사용하여 임시 자격 증명을 사용하여 액세스하도록 하는 것입니다.

페더레이션 ID는 기업 사용자 디렉토리, 웹 ID 공급자, Identity Center 디렉터리의 사용자 또는 ID 소스를 통해 제공된 자격 증명을 사용하여 액세스하는 AWS 서비스 모든 사용자를 말합니다. AWS Directory Service페더레이션 ID에 AWS 계정액세스하면 이들이 역할을 맡고 역할은 임시 자격 증명을 제공합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center을 사용하는 것이 좋습니다. IAM Identity Center에서 사용자 및 그룹을 생성하거나 자체 ID 소스의 사용자 및 그룹 집합에 연결하고 동기화하여 모든 사용자 및 애플리케이션에서 사용할 수 있습니다. AWS 계정 IAM Identity Center에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 IAM Identity Center란 무엇입니까?를 참조하십시오.

IAM 사용자 및 그룹

IAM 사용자는 단일 사용자 또는 애플리케이션에 대한 특정 권한을 AWS 계정 가진 사용자 내 자격 증명입니다. 가능하면 암호 및 액세스 키와 같은 장기 자격 증명이 있는 IAM 사용자를 생성하는 대신 임시 자격 증명을 사용하는 것이 좋습니다. 하지만 IAM 사용자의 장기 자격 증명이 필요한 특정 사용 사례가 있는 경우 액세스 키를 교체하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 장기 보안 인증이 필요한 사용 사례의 경우 정기적으로 액세스 키 교체를 참조하십시오.

IAM 그룹은 IAM 사용자 컬렉션을 지정하는 자격 증명입니다. 사용자는 그룹으로 로그인할 수 없습니다. 그룹을 사용하여 여러 사용자의 권한을 한 번에 지정할 수 있습니다. 그룹을 사용하면 대규모 사용자 집합의 권한을 더 쉽게 관리할 수 있습니다. 예를 들어, IAMAdmins라는 그룹이 있고 이 그룹에 IAM 리소스를 관리할 권한을 부여할 수 있습니다.

사용자는 역할과 다릅니다. 사용자는 한 사람 또는 애플리케이션과 고유하게 연결되지만, 역할은 해당 역할이 필요한 사람이라면 누구나 수임할 수 있습니다. 사용자는 영구적인 장기 보안 인증 정보를 가지고 있지만, 역할은 임시 보안 인증 정보만 제공합니다. 자세한 정보는 IAM 사용 설명서의 IAM 사용자를 만들어야 하는 경우(역할이 아님)를 참조하십시오.

IAM 역할

IAM 역할은 특정 권한을 가진 사용자 AWS 계정 내의 자격 증명입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. 역할을 AWS Management Console 전환하여 에서 일시적으로 IAM 역할을 맡을 수 있습니다. AWS CLI 또는 AWS API 작업을 호출하거나 사용자 지정 URL을 사용하여 역할을 수임할 수 있습니다. 역할 사용 방법에 대한 자세한 정보는 IAM 사용 설명서의 IAM 역할 사용을 참조하십시오.

임시 보안 인증이 있는 IAM 역할은 다음과 같은 상황에서 유용합니다:

- 연합 사용자 액세스 - 연합 아이덴티티에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 연합 아이덴티티가 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션 역할에 대한 자세한 내용은 IAM 사용 설명서의 타사 자격 증명 공급자의 역할 생성을 참조하세요. IAM Identity Center를 사용하는 경우 권한 집합을 구성합니다. 인증 후 아이덴티티가 액세스할 수 있는 항목을 통제하기 위해 IAM Identity Center는 권한 세트를 IAM의 역할과 상관짓습니다. 권한 세트에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 권한 세트를 참조하십시오.
- 임시 IAM 사용자 권한 - IAM 사용자 또는 역할은 IAM 역할을 수임하여 특정 작업에 대한 다양한 권한을 임시로 받을 수 있습니다.
- 크로스 계정 액세스 - IAM 역할을 사용하여 다른 계정의 사용자(신뢰할 수 있는 보안 주체)가 내 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입니다. 그러나 일부 AWS 서비스경우에는 역할을 프록시로 사용하는 대신 정책을 리소스에 직접 연결할 수 있습니다. 교차 계정 액세스를 위한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조하십시오.
- 서비스 간 액세스 — 일부는 다른 AWS 서비스서비스의 기능을 AWS 서비스 사용합니다. 예컨대, 어떤 서비스에서 호출을 수행하면 일반적으로 해당 서비스는 Amazon EC2에서 애플리케이션을 실행하거나 Amazon S3에 객체를 저장합니다. 서비스는 호출하는 보안 주체의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 작업을 수행할 수 있습니다.
  - 순방향 액세스 세션 (FAS) — IAM 사용자 또는 역할을 사용하여 작업을 수행하는 경우 보안 AWS 주체로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는 전화를 거는 주체의 권한을 다운스트림 AWS 서비스서비스에 AWS 서비스 요청하기 위한 요청과 결합하여 사용합니다. FAS 요청은 다른 서비스 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 서비스가 수신한 경우에만 이루어집니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.
  - 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 태스크를 수행하기 위해 맡는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 역할을 생성하여 AWS 서비스에게 권한 위임을 참조하십시오.

- 서비스 연결 역할 — 서비스 연결 역할은 에 연결된 서비스 역할의 한 유형입니다. AWS 서비스서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은 사용자에게 AWS 계정 표시되며 해당 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.
- Amazon EC2에서 실행되는 애플리케이션 — IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 API 요청을 AWS CLI 하는 애플리케이션의 임시 자격 증명을 관리할 수 있습니다. AWS 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS 역할을 할당하고 모든 애플리케이션에서 사용할 수 있게 하려면 인스턴스에 연결된 인스턴스 프로필을 생성합니다. 인스턴스 프로파일에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 보안 인증 정보를 얻을 수 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 참조하십시오.

IAM 역할을 사용할지 또는 IAM 사용자를 사용할지를 알아보려면 IAM 사용 설명서의 IAM 역할(사용자 대신)을 생성하는 경우를 참조하십시오.

정책을 사용한 액세스 관리

정책을 생성하고 이를 AWS ID 또는 리소스에 AWS 연결하여 액세스를 제어할 수 있습니다. 정책은 ID 또는 리소스와 연결될 때 AWS 해당 권한을 정의하는 객체입니다. AWS 주도자 (사용자, 루트 사용자 또는 역할 세션) 가 요청할 때 이러한 정책을 평가합니다. 정책의 권한이 요청 허용 또는 거부 여부를 결정합니다. 대부분의 정책은 JSON 문서로 AWS 저장됩니다. JSON 정책 문서의 구조와 콘텐츠에 대한 자세한 정보는 IAM 사용 설명서의 JSON 정책 개요를 참조하십시오.

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

기본적으로, 사용자와 역할에는 어떠한 권한도 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 수임할 수 있습니다.

IAM 정책은 작업을 수행하기 위해 사용하는 방법과 상관없이 작업에 대한 권한을 정의합니다. 예를 들어, `iam:GetRole`태스크를 허용하는 정책이 있다고 가정합니다. 해당 정책을 사용하는 사용자는 AWS Management Console, AWS CLI, 또는 AWS API에서 역할 정보를 가져올 수 있습니다.

보안 인증 기반 정책

보안 인증 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 보안 인증에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수

행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하십시오.

자격 증명 기반 정책은 인라인 정책 또는 관리형 정책으로 한층 더 분류할 수 있습니다. 인라인 정책은 단일 사용자, 그룹 또는 역할에 직접 포함됩니다. 관리형 정책은 내 여러 사용자, 그룹 및 역할에 연결할 수 있는 독립형 정책입니다. AWS 계정관리형 정책에는 AWS 관리형 정책과 고객 관리형 정책이 포함됩니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책과 인라인 정책의 선택을 참조하십시오.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 연동 사용자 등이 포함될 수 있습니다. AWS 서비스

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. IAM의 AWS 관리형 정책은 리소스 기반 정책에 사용할 수 없습니다.

액세스 제어 목록(ACL)

액세스 제어 목록(ACL)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACL은 JSON 정책 설명서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

ACL을 지원하는 서비스의 예로는 아마존 S3와 아마존 VPC가 있습니다. AWS WAF ACL에 대해 자세히 알아보려면 Amazon Simple Storage Service 개발자 안내서의 ACL(액세스 제어 목록) 개요를 참조하십시오.

기타 정책 타입

AWS 일반적이지 않은 추가 정책 유형을 지원합니다. 이러한 정책 타입은 더 일반적인 정책 타입에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 – 권한 경계는 자격 증명 기반 정책에 따라 IAM 엔터티(IAM 사용자 또는 역할)에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 개체에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻는 권한은 개체의 보안 인증 기반 정책과 그 권한 경계의 교집합입니다. `Principal` 필드에

서 사용자나 역할을 보안 주체로 지정하는 리소스 기반 정책은 권한 경계를 통해 제한되지 않습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 정보는 IAM 사용 설명서의 IAM 엔터티에 대한 권한 경계를 참조하십시오.

- 서비스 제어 정책 (SCP) - SCP는 조직 또는 조직 단위 (OU) 에 대한 최대 권한을 지정하는 JSON 정책입니다. AWS Organizations AWS Organizations 사업체가 소유한 여러 AWS 계정 개를 그룹화하고 중앙에서 관리하는 서비스입니다. 조직에서 모든 특성을 활성화할 경우 서비스 제어 정책 (SCP)을 임의의 또는 모든 계정에 적용할 수 있습니다. SCP는 구성원 계정의 엔티티 (각 엔티티 포함) 에 대한 권한을 제한합니다. AWS 계정 루트 사용자조직 및 SCP에 대한 자세한 정보는 AWS Organizations 사용 설명서의 SCP 작동 방식을 참조하십시오.
- 세션 정책 – 세션 정책은 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할의 자격 증명 기반 정책의 교집합과 세션 정책입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 자세한 정보는 IAM 사용 설명서의 세션 정책을 참조하십시오.

여러 정책 타입

여러 정책 타입이 요청에 적용되는 경우 결과 권한은 이해하기가 더 복잡합니다. 여러 정책 유형이 관련되어 있을 때 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하십시오.

AWS IoT와 FleetWise IAM의 작동 방식

IAM을 사용하여 IoT에 대한 액세스를 관리하기 전에 AWS FleetWise IoT에서 사용할 수 있는 IAM 기능에 대해 알아보십시오. AWS FleetWise

IoT와 함께 AWS 사용할 수 있는 IAM 기능 FleetWise

IAM 특성	AWS IoT FleetWise 지원
ID 기반 정책	예
리소스 기반 정책	아니요
정책 작업	예
정책 리소스	예

IAM 특성	AWS IoT FleetWise 지원
정책 조건 키	예
ACL	아니요
ABAC(정책 내 태그)	부분
임시 보안 인증 정보	예
보안 주체 권한	예
서비스 역할	아니요
서비스 연결 역할	아니요

AWS IoT FleetWise 및 기타 AWS 서비스가 대부분의 IAM 기능과 어떻게 작동하는지 자세히 알아보려면 IAM 사용 설명서의 IAM과 연동되는AWS 서비스를 참조하십시오.

IoT를 위한 ID 기반 정책 AWS FleetWise

ID 기반 정책 지원	예

자격 증명기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 자격 증명에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하십시오.

IAM ID 기반 정책을 사용하면 허용되거나 거부되는 태스크와 리소스 뿐만 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. 자격 증명 기반 정책에서는 보안 주체가 연결된 사용자 또는 역할에 적용되므로 보안 주체를 지정할 수 없습니다. JSON 정책에서 사용하는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하십시오.

IoT를 위한 ID 기반 정책 예제 AWS FleetWise

AWS IoT FleetWise ID 기반 정책의 예를 보려면 을 참조하십시오. IoT를 위한 ID 기반 정책 예제 AWS FleetWise

IoT 내 리소스 기반 정책 AWS FleetWise

리소스 기반 정책 지원	아니요

리소스 기반 정책은 리소스에 연결하는 JSON 정책 문서입니다. 리소스 기반 정책의 예는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 등이 포함될 수 있습니다. AWS 서비스

크로스 계정 액세스를 활성화하려는 경우 전체 계정이나 다른 계정의 IAM 개체를 리소스 기반 정책의 보안 주체로 지정할 수 있습니다. 리소스 기반 정책에 크로스 계정 보안 주체를 추가하는 것은 트러스트 관계 설정의 절반밖에 되지 않는다는 것을 유념하십시오. 보안 주체와 리소스가 다른 AWS 계정경우 신뢰할 수 있는 계정의 IAM 관리자는 보안 주체 개체 (사용자 또는 역할) 에게 리소스에 액세스할 수 있는 권한도 부여해야 합니다. 개체에 자격 증명 기반 정책을 연결하여 권한을 부여합니다. 하지만 리소스 기반 정책이 동일 계정의 보안 주체에 액세스를 부여하는 경우 추가 자격 증명 기반 정책이 필요하지 않습니다. 자세한 정보는 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조하십시오.

AWS IoT를 위한 정책 조치 FleetWise

정책 작업 지원	예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

JSON 정책의 `Action`요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 태스크를 설명합니다. 정책 작업은 일반적으로 관련 AWS API 작업과 이름이 같습니다. 일치하는 API 작업이 없는 권한 전용 작업 같은 몇 가지 예외도 있습니다. 정책에서 여러 작업이 필요한 몇 가지 작업도 있습니다. 이러한 추가 작업을 종속 작업이라고 합니다.

연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하십시오.

AWS IoT FleetWise 작업 목록을 보려면 서비스 권한 부여 참조의 AWS FleetWise IoT에서 정의한 작업을 참조하십시오.

AWS IoT의 정책 조치는 조치 앞에 다음 접두사를 FleetWise 사용합니다.

```
iotfleetwise
```

단일 문에서 여러 작업을 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
      "iotfleetwise:action1",
      "iotfleetwise:action2"
]
```

와일드카드(*)를 사용하여 여러 작업을 지정할 수 있습니다. 예를 들어, `List`(이)라는 단어로 시작하는 모든 작업을 지정하려면 다음 작업을 포함합니다.

```
"Action": "iotfleetwise:List*"
```

AWS IoT FleetWise ID 기반 정책의 예를 보려면 을 참조하십시오. IoT를 위한 ID 기반 정책 예제 AWS FleetWise

AWS IoT를 위한 정책 리소스 FleetWise

정책 리소스 지원	예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Resource` JSON 정책 요소는 작업이 적용되는 하나 이상의 객체를 지정합니다. 보고서에는 `Resource` 또는 `NotResource` 요소가 반드시 추가되어야 합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 리소스 수준 권한이라고 하는 특정 리소스 유형을 지원하는 작업에 대해 이 작업을 수행할 수 있습니다.

작업 나열과 같이 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

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

AWS IoT FleetWise 리소스 유형 및 해당 ARN 목록을 보려면 서비스 권한 부여 FleetWise 참조의 AWS IoT로 정의된 리소스를 참조하십시오. 각 리소스의 ARN을 지정할 수 있는 작업에 대해 알아보려면 AWS IoT에서 정의한 작업을 참조하십시오. FleetWise

AWS IoT FleetWise ID 기반 정책의 예를 보려면 을 참조하십시오. IoT를 위한 ID 기반 정책 예제 AWS FleetWise

AWS IoT를 위한 정책 조건 키 FleetWise

서비스별 정책 조건 키 지원	예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Condition` 요소(또는 `Condition`블록)를 사용하면 정책이 발효되는 조건을 지정할 수 있습니다. `Condition` 요소는 옵션입니다. 같거나 적음 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다.

한 문에서 여러 `Condition`요소를 지정하거나 단일 `Condition`요소에서 여러 키를 지정하는 경우 AWS 는 논리적 `AND`태스크를 사용하여 평가합니다. 단일 조건 키에 여러 값을 지정하는 경우 는 논리적 `OR` 연산을 사용하여 조건을 AWS 평가합니다. 명령문의 권한을 부여하기 전에 모든 조건을 충족해야 합니다.

조건을 지정할 때 자리 표시자 변수를 사용할 수도 있습니다. 예를 들어, IAM 사용자에게 IAM 사용자 이름으로 태그가 지정된 경우에만 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM 정책 요소: 변수 및 태그를 참조하십시오.

AWS 글로벌 조건 키 및 서비스별 조건 키를 지원합니다. 모든 AWS 글로벌 조건 키를 보려면 IAM 사용 AWS 설명서의 글로벌 조건 컨텍스트 키를 참조하십시오.

AWS IoT FleetWise 조건 키 목록을 보려면 서비스 인증 FleetWise 참조의 AWS IoT용 조건 키를 참조하십시오. 조건 키를 사용할 수 있는 작업 및 리소스에 대해 알아보려면 AWS IoT에서 정의한 작업을 참조하십시오 FleetWise .

AWS IoT FleetWise ID 기반 정책의 예를 보려면 을 참조하십시오. IoT를 위한 ID 기반 정책 예제 AWS FleetWise

IoT의 AWS 액세스 제어 목록 (ACL) FleetWise

ACL 지원	아니요

액세스 제어 목록(ACL)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACL은 JSON 정책 설명서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

IoT를 통한 속성 기반 액세스 제어 (ABAC) AWS FleetWise

ABAC(정책 내 태그) 지원	부분

속성 기반 액세스 제어(ABAC)는 속성을 기반으로 권한을 정의하는 권한 부여 전략입니다. AWS에서는 이러한 속성을 태그라고 합니다. IAM 개체 (사용자 또는 역할) 및 여러 AWS 리소스에 태그를 첨부할 수 있습니다. ABAC의 첫 번째 단계로 개체 및 리소스에 태그를 지정합니다. 그런 다음 보안 주체의 태그가 액세스하려는 리소스의 태그와 일치할 때 작업을 허용하도록 ABAC 정책을 설계합니다.

ABAC는 빠르게 성장하는 환경에서 유용하며 정책 관리가 번거로운 상황에 도움이 됩니다.

태그를 기반으로 액세스를 제어하려면 `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* 또는 `aws:TagKeys` 조건 키를 사용하여 정책의 조건 요소에 태그 정보를 제공합니다.

서비스가 모든 리소스 유형에 대해 세 가지 조건 키를 모두 지원하는 경우 값은 서비스에 대해 예입니다. 서비스가 일부 리소스 유형에 대해서만 세 가지 조건 키를 모두 지원하는 경우 값은 부분입니다.

ABAC에 대한 자세한 정보는 IAM 사용 설명서의 ABAC란 무엇인가요?를 참조하십시오. ABAC 설정 단계가 포함된 자습서를 보려면 IAM 사용 설명서의 속성 기반 액세스 제어(ABAC) 사용을 참조하십시오.

> **Note**
>
> AWS IoT는 `CreateCampaign` API 작업에 필요한 FleetWise 지원만 지원합니다`iam:PassRole`.

AWS IoT에서 임시 자격 증명 사용 FleetWise

임시 보안 인증 정보 지원	예

임시 자격 증명을 사용하여 로그인하면 일부 자격 증명이 AWS 서비스 작동하지 않습니다. 임시 자격 증명을 사용하는 방법을 AWS 서비스 비롯한 추가 정보는 IAM 사용 설명서의 IAM과AWS 서비스 연동 되는 내용을 참조하십시오.

사용자 이름과 암호를 제외한 다른 방법을 AWS Management Console 사용하여 로그인하면 임시 자격 증명을 사용하는 것입니다. 예를 들어 회사의 SSO (Single Sign-On) 링크를 AWS 사용하여 액세스하는 경우 이 프로세스에서 자동으로 임시 자격 증명을 생성합니다. 또한 콘솔에 사용자로 로그인한 다음 역할을 전환할 때 임시 보안 인증 정보를 자동으로 생성합니다. 역할 전환에 대한 자세한 정보는 IAM 사용 설명서의 역할로 전환(콘솔)을 참조하십시오.

또는 API를 사용하여 임시 자격 증명을 수동으로 생성할 수 있습니다 AWS CLI . AWS 그런 다음 해당 임시 자격 증명을 사용하여 액세스할 수 AWS있습니다. AWS 장기 액세스 키를 사용하는 대신 임시 자격 증명을 동적으로 생성할 것을 권장합니다. 자세한 정보는 IAM의 임시 보안 자격 증명 섹션을 참조하십시오.

IoT에 대한 AWS 크로스 서비스 주체 권한 FleetWise

전달 액세스 세션(FAS) 지원	예

IAM 사용자 또는 역할을 사용하여 작업을 수행하는 AWS경우 보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는 전화를 거는 주체의 권한을 다운스트림 서비스에 AWS 서비스 요청하라는 요청과 결합하여 사용합니다. AWS 서비스 FAS 요청은 다른 서비스 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 서비스가 수신한 경우에만 이루어집니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.

AWS IoT의 서비스 역할 FleetWise

서비스 역할 지원	아니요

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하기 위해 수임하는 IAM role(IAM 역할)입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 역할을 생성하여 AWS 서비스에게 권한 위임을 참조하십시오.

> **Warning**
>
> 서비스 역할에 대한 권한을 변경하면 AWS IoT FleetWise 기능이 손상될 수 있습니다. AWS IoT가 이에 대한 지침을 FleetWise 제공하는 경우에만 서비스 역할을 편집하십시오.

IoT의 서비스 연계 역할 AWS FleetWise

서비스 연결 역할 지원	아니요

서비스 연결 역할은 에 연결된 서비스 역할 유형입니다. AWS 서비스서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은 사용자에게 AWS 계정 표시되며 해당 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

서비스 연결 역할 생성 또는 관리에 대한 자세한 내용은 IAM으로 작업하는AWS 서비스 섹션을 참조하세요. Service-linked role(서비스 연결 역할) 열에서 Yes이(가) 포함된 서비스를 테이블에서 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 Yes(네) 링크를 선택합니다.

AWS IoT용 서비스 연결 역할 사용 FleetWise

AWS IoT는 AWS Identity and Access Management (IAM) 서비스 연결 역할을 FleetWise 사용합니다. 서비스 연결 역할은 AWS IoT에 직접 연결된 고유한 유형의 IAM 역할입니다. FleetWise 서비스 연결 역할은 AWS IoT에 의해 사전 정의되며, AWS FleetWise IoT가 Amazon에 메트릭을 전송하는 데 FleetWise 필요한 권한을 포함합니다. CloudWatch 자세한 정보는 Amazon CloudWatch로 AWS IoT FleetWise 모니터링을 참조하세요.

서비스 연결 역할을 사용하면 필요한 권한을 수동으로 추가할 필요가 없으므로 AWS IoT를 FleetWise 더 빠르게 설정할 수 있습니다. AWS IoT는 서비스 연결 역할의 권한을 FleetWise 정의하며, 달리 정의되지 않는 한 AWS IoT만 역할을 맡을 FleetWise 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함됩니다. 이 권한 정책은 다른 어떤 IAM 엔터티에도 연결할 수 없습니다.

먼저 관련 리소스를 삭제한 후에만 서비스 연결 역할을 삭제할 수 있습니다. 이렇게 하면 FleetWise 리소스 액세스 권한을 실수로 제거할 수 없으므로 AWS IoT 리소스가 보호됩니다.

서비스 연결 역할을 지원하는 기타 서비스에 대한 자세한 내용은 IAM으로 작업하는AWS 서비스를 참조하고 서비스 연결 역할(Service-linked roles) 열에 예(Yes)가 있는 서비스를 찾으세요. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 Yes 링크를 선택합니다.

AWS IoT의 서비스 연결 역할 권한 FleetWise

AWS IoT는 AWS IoT의 모든 out-of-the-box 권한에 사용되는 AWS 관리형 정책이라는 AWSServiceRoleForIoTFleetWise서비스 연결 역할을 FleetWise 사용합니다. FleetWise

AWSServiceRoleForIoTFleetWise 서비스 연결 역할은 다음 서비스를 신뢰하여 역할을 맡습니다.

- `IoTFleetWise`

이름이 지정된 역할 권한 정책은 AWS IoT가 지정된 리소스에서 다음 작업을 FleetWise 완료할 수 AWSIoTFleetwiseServiceRolePolicy 있도록 허용합니다.

- 작업: `cloudwatch:PutMetricData` 리소스의 경우: `*`

IAM 엔터티(사용자, 그룹, 역할 등)가 서비스 링크 역할을 생성하고 편집하거나 삭제할 수 있도록 권한을 구성할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 권한을 참조하십시오.

AWS IoT를 위한 서비스 연결 역할 생성 FleetWise

서비스 링크 역할은 수동으로 생성할 필요가 없습니다. AWS IoT FleetWise 콘솔 AWS CLI, 또는 AWS API에서 계정을 등록하면 AWS IoT가 서비스 연결 역할을 대신 FleetWise 생성합니다. 자세한 정보는 설정 구성을 참조하세요.

AWS IoT에서 서비스 연결 역할 생성 FleetWise (콘솔)

서비스 링크 역할은 수동으로 생성할 필요가 없습니다. AWS IoT FleetWise 콘솔, AWS CLI 또는 AWS API에서 계정을 등록하면 AWS IoT가 서비스 연결 역할을 대신 FleetWise 생성합니다.

AWS IoT의 서비스 연결 역할 편집 FleetWise

AWS IoT에서는 AWSServiceRoleForIoTFleetWise 서비스 연결 역할을 편집할 수 없습니다. FleetWise 생성한 서비스 연결 역할을 여러 엔터티가 참조할 수 있기 때문에 역할의 이름을 변경할 수 없습니다. 그러나 IAM을 사용하여 역할의 설명을 편집할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 서비스 연결 역할 편집을 참조하십시오.

서비스 연결 역할을 정리

IAM을 사용하여 서비스 연결 역할을 삭제하기 전에 먼저 역할에서 사용되는 리소스를 삭제해야 합니다.

> **Note**
>
> 리소스를 삭제하려고 할 때 AWS FleetWise IoT가 역할을 사용하는 경우 삭제가 실패할 수 있습니다. 이 문제가 발생하면 몇 분 기다렸다가 작업을 다시 시도하세요. 콘솔, AWS CLI 또는 AWS API를 service-linked-role 통해 삭제하는 방법을 알아보려면 IAM 사용 설명서의 서비스 연결 역할 사용을 참조하십시오.

이 서비스 연결 역할을 삭제한 후 다시 생성해야 하는 경우, AWS IoT에 계정을 등록할 수 있습니다. FleetWise FleetWise 그러면 AWS IoT가 서비스 연결 역할을 다시 생성합니다.

IoT를 위한 ID 기반 정책 예제 AWS FleetWise

기본적으로 사용자와 역할에는 AWS IoT FleetWise 리소스를 만들거나 수정할 권한이 없습니다. 또한 AWS Management Console, AWS Command Line Interface (AWS CLI) 또는 AWS API를 사용하여 작업을 수행할 수 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 맡을 수 있습니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하십시오.

각 리소스 유형의 ARN 형식을 비롯하여 AWS FleetWise FleetWise IoT에서 정의한 작업 및 리소스 유형에 대한 자세한 내용은 서비스 권한 부여 참조의 AWS IoT용 작업, 리소스 및 조건 키를 참조하십시오.

주제

- 정책 모범 사례
- AWS IoT FleetWise 콘솔 사용
- 사용자가 자신의 고유한 권한을 볼 수 있도록 허용
- Amazon Timestream의 리소스에 액세스

정책 모범 사례

ID 기반 정책은 누군가가 계정에서 AWS IoT FleetWise 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부를 결정합니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. 자격 증명 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따르십시오.

- AWS 관리형 정책으로 시작하여 최소 권한 권한으로 이동 — 사용자와 워크로드에 권한을 부여하려면 여러 일반적인 사용 사례에 권한을 부여하는 AWS 관리형 정책을 사용하세요. 해당 내용은 에서 사용할 수 있습니다. AWS 계정사용 사례에 맞는 AWS 고객 관리형 정책을 정의하여 권한을 더 줄이는 것이 좋습니다. 자세한 정보는 IAM 사용 설명서의 AWS 관리형 정책 또는 직무에 대한AWS 관리형 정책을 참조하세요.
- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우 작업을 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명서의 IAM의 정책 및 권한을 참조하세요.
- Use conditions in IAM policies to further restrict access(IAM 정책의 조건을 사용하여 액세스 추가 제한) – 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어 SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 생성할 수 있습니다. 예를 AWS 서비스들어 특정 작업을 통해 서비스 작업을 사용하는 경우 조건을 사용하여 서비스 작업에 대한 액세스 권한을 부여할 수도 AWS CloudFormation있습니다. 자세한 정보는 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하십시오.
- IAM Access Analyzer를 통해 IAM 정책을 검증하여 안전하고 기능적인 권한 보장 – IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 신규 및 기존 정책을 검증합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 권장 사항을 제공하여 안전하고 기능적인 정책을 생성하도록 돕습니다. 자세한 정보는 IAM 사용 설명서의 IAM Access Analyzer 정책 검증을 참조하세요.
- 멀티 팩터 인증 (MFA) 필요 - IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 AWS 계정 MFA를 활성화하십시오. API 작업을 직접적으로 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 정보는 IAM 사용 설명서의 MFA 보호 API 액세스 구성을 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하십시오.

AWS IoT FleetWise 콘솔 사용

AWS IoT FleetWise 콘솔에 액세스하려면 최소 권한 집합이 있어야 합니다. 이러한 권한을 통해 내 AWS IoT FleetWise 리소스를 나열하고 세부 정보를 볼 수 있어야 AWS 계정합니다. 최소 필수 권한보

다 더 제한적인 ID 기반 정책을 만들면 콘솔이 해당 정책에 연결된 개체(사용자 또는 역할)에 대해 의도대로 작동하지 않습니다.

AWS CLI 또는 AWS API만 호출하는 사용자에게 최소 콘솔 권한을 허용할 필요는 없습니다. 그 대신, 수행하려는 API 작업과 일치하는 작업에만 액세스할 수 있도록 합니다.

사용자와 역할이 AWS IoT FleetWise 콘솔을 계속 사용할 수 있도록 하려면 AWS IoT FleetWise `ConsoleAccess` 또는 `ReadOnly` AWS 관리형 정책도 엔티티에 연결하십시오. 자세한 내용은 IAM 사용 설명서의 사용자에게 권한 추가를 참조하십시오.

사용자가 자신의 고유한 권한을 볼 수 있도록 허용

이 예제는 IAM 사용자가 자신의 사용자 자격 증명에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 AWS CLI 또는 AWS API를 사용하여 프로그래밍 방식으로 이 작업을 완료할 수 있는 권한이 포함됩니다.

```
{
    "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 Timestream의 리소스에 액세스

AWS IoT를 사용하기 전에 AWS 계정 FleetWise, IAM 및 Amazon Timestream 리소스를 등록하여 AWS FleetWise IoT에 사용자 대신 차량 데이터를 AWS 클라우드 전송할 수 있는 권한을 부여해야 합니다. 등록하려면 다음이 필요합니다.

- Amazon Timestream 데이터베이스.
- 지정된 Amazon Timestream 데이터베이스에서 생성된 테이블.
- AWS IoT가 Amazon FleetWise Timestream으로 데이터를 전송할 수 있도록 하는 IAM 역할입니다.

절차 및 예제 정책을 포함한 자세한 내용은 구성 설정을 참조하세요.

AWS IoT FleetWise ID 및 액세스 문제 해결

다음 정보를 사용하면 AWS IoT 및 IAM으로 작업할 때 발생할 수 있는 일반적인 문제를 FleetWise 진단하고 해결하는 데 도움이 됩니다.

주제

- AWS IoT에서 작업을 수행할 권한이 없습니다. FleetWise
- 저는 IAM을 수행할 권한이 없습니다. PassRole
- 외부 사용자가 내 AWS IoT FleetWise 리소스에 액세스할 수 있도록 AWS 계정 허용하고 싶습니다.

AWS IoT에서 작업을 수행할 권한이 없습니다. FleetWise

작업을 수행할 권한이 없다는 AWS Management Console 메시지가 표시되면 관리자에게 도움을 요청해야 합니다. 관리자는 로그인 보안 인증 정보를 제공한 사람입니다.

다음 예제 오류는 `mateojackson` IAM 사용자가 콘솔을 사용하여 가상 *`myVehicle`* 리소스에 대한 세부 정보를 보려고 하지만 `iotfleetwise:`*`GetVehicleStatus`* 권한이 없을 때 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 iotfleetwise:GetVehicleStatus on resource: myVehicle
```

이 경우 Mateo는 *myVehicle* 작업을 사용하여 `iotfleetwise:GetVehicleStatus` 리소스에 액세스하도록 허용하는 정책을 업데이트하라고 관리자에게 요청합니다.

저는 IAM을 수행할 권한이 없습니다. PassRole

작업을 수행할 권한이 없다는 오류가 발생하는 경우 AWS IoT에 역할을 전달할 수 있도록 정책을 업데이트해야 FleetWise 합니다. `iam:PassRole`

일부 AWS 서비스 서비스에서는 새 서비스 역할 또는 서비스 연결 역할을 만드는 대신 기존 역할을 해당 서비스에 전달할 수 있습니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 있어야 합니다.

다음 예제 오류는 이라는 IAM 사용자가 콘솔을 사용하여 AWS FleetWise IoT에서 작업을 `marymajor` 수행하려고 할 때 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 수 있는 권한을 가지고 있지 않습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

이 경우 Mary가 `iam:PassRole`작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

외부 사용자가 내 AWS IoT FleetWise 리소스에 액세스할 수 있도록 AWS 계정 허용하고 싶습니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 지원하는 서비스의 경우 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세히 알아보려면 다음을 참조하십시오.

- AWS IoT가 이러한 기능을 FleetWise 지원하는지 알아보려면 을 참조하십시오AWS IoT와 FleetWise IAM의 작동 방식.
- 소유한 리소스에 대한 액세스 권한을 AWS 계정 부여하는 방법을 알아보려면 IAM 사용 설명서에서 소유하고 AWS 계정 있는 다른 IAM 사용자에게 액세스 권한 제공을 참조하십시오.

- 제3자에게 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사 AWS 계정AWS 계정 소유에 대한 액세스 제공을 참조하십시오.
- 자격 증명 연동을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(자격 증명 연동)을 참조하십시오.
- 교차 계정 액세스를 위한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조하십시오.

AWS IoT를 위한 규정 준수 검증 FleetWise

> **Note**
>
> AWS IoT는 AWS 규정 준수 프로그램의 범위에 포함되지 FleetWise 않습니다.

특정 규정 준수 프로그램의 범위 내에 AWS 서비스 있는지 알아보려면AWS 서비스 규정 준수 프로그램의 범위별, 규정 준수AWS 서비스 프로그램별 참조하여 관심 있는 규정 준수 프로그램을 선택하십시오. 일반 정보는 AWS 규정 준수 프로그램AWS 보증 프로그램 규정AWS 참조하십시오.

를 사용하여 AWS Artifact타사 감사 보고서를 다운로드할 수 있습니다. 자세한 내용은 의 보고서 https://docs.aws.amazon.com/artifact/latest/ug/downloading-documents.html 참조하십시오 AWS Artifact.

사용 시 규정 준수 AWS 서비스 책임은 데이터의 민감도, 회사의 규정 준수 목표, 관련 법률 및 규정에 따라 결정됩니다. AWS 규정 준수에 도움이 되는 다음 리소스를 제공합니다.

- 보안 및 규정 준수 퀵 스타트 가이드 - 이 배포 가이드에서는 아키텍처 고려 사항을 설명하고 보안 및 규정 준수에 AWS 중점을 둔 기본 환경을 배포하기 위한 단계를 제공합니다.
- Amazon Web Services의 HIPAA 보안 및 규정 준수를 위한 설계 — 이 백서에서는 기업이 HIPAA 적격 애플리케이션을 만드는 AWS 데 사용할 수 있는 방법을 설명합니다.

  > **Note**
  >
  > 모든 AWS 서비스 사람이 HIPAA 자격을 갖춘 것은 아닙니다. 자세한 내용은 HIPAA 적격 서비스 참조를 참조하십시오.

- AWS 규정 준수 리소스AWS — 이 워크북 및 가이드 모음은 해당 산업 및 지역에 적용될 수 있습니다.

- AWS 고객 규정 준수 가이드 — 규정 준수의 관점에서 공동 책임 모델을 이해하십시오. 이 가이드에서는 보안을 유지하기 위한 모범 사례를 AWS 서비스 요약하고 여러 프레임워크 (미국 표준 기술 연구소 (NIST), 결제 카드 산업 보안 표준 위원회 (PCI), 국제 표준화기구 (ISO) 등) 에서 보안 제어에 대한 지침을 매핑합니다.
- AWS Config 개발자 안내서의 규칙을 사용하여 리소스 평가 — 이 AWS Config 서비스는 리소스 구성이 내부 관행, 업계 지침 및 규정을 얼마나 잘 준수하는지 평가합니다.
- AWS Security Hub— 이를 AWS 서비스 통해 내부 AWS보안 상태를 포괄적으로 파악할 수 있습니다. Security Hub는 보안 제어를 사용하여 AWS 리소스를 평가하고 보안 업계 표준 및 모범 사례에 대한 규정 준수를 확인합니다. 지원되는 서비스 및 제어 목록은 Security Hub 제어 참조를 참조하십시오.
- AWS Audit Manager— 이를 AWS 서비스 통해 AWS 사용량을 지속적으로 감사하여 위험을 관리하고 규정 및 업계 표준을 준수하는 방법을 단순화할 수 있습니다.

IoT의 AWS 레질리언스 FleetWise

AWS 글로벌 인프라는 AWS 지역 및 가용 영역을 중심으로 구축됩니다. 리전은 물리적으로 분리되고 격리된 다수의 가용 영역을 제공하며, 이러한 영역은 짧은 지연 시간, 높은 처리량 및 높은 중복성을 갖춘 네트워크를 통해 연결되어 있습니다. 가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 극복 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 지역 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하십시오.

> **Note**
>
> AWS FleetWise IoT로 처리된 데이터는 Amazon Timestream 데이터베이스에 저장됩니다. 타임스트림은 다른 가용 AWS 영역 또는 지역으로의 백업을 지원합니다. 하지만 Timestream SDK를 사용하여 자체 애플리케이션을 작성하여 데이터를 쿼리하고 선택한 대상에 저장할 수 있습니다.
> Amazon Timestream 에 대한 자세한 내용은 Amazon Timestream 개발자 가이드를 참조하세요.

AWS IoT의 인프라 보안 FleetWise

관리형 서비스인 AWS FleetWise IoT는 AWS 글로벌 네트워크 보안으로 보호됩니다. AWS 보안 서비스 및 인프라 AWS 보호 방법에 대한 자세한 내용은 AWS 클라우드 보안을 참조하십시오. 인프라 보안

모범 사례를 사용하여 AWS 환경을 설계하려면 Security Pillar AWS Well-Architected Framework의 인프라 보호를 참조하십시오.

AWS 게시된 API 호출을 사용하여 네트워크를 FleetWise 통해 AWS IoT에 액세스합니다. 고객은 다음을 지원해야 합니다.

- 전송 계층 보안(TLS). TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Ephemeral Diffie-Hellman)와 같은 완전 전송 보안(PFS)이 포함된 암호 제품군. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 주체와 관련된 비밀 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 보안 인증 정보를 생성하여 요청에 서명할 수 있습니다.

모든 네트워크 위치에서 이러한 API 작업을 호출할 수 있지만 AWS FleetWise IoT는 소스 IP 주소에 따른 제한을 포함할 수 있는 리소스 기반 액세스 정책을 지원합니다. 또한 AWS IoT FleetWise 정책을 사용하여 특정 Amazon VPC (가상 사설 클라우드) 엔드포인트 또는 특정 VPC에서의 액세스를 제어할 수 있습니다. 이를 통해 네트워크 내의 AWS 특정 VPC로부터 특정 AWS IoT FleetWise 리소스에 대한 네트워크 액세스를 효과적으로 분리할 수 있습니다.

주제

- 인터페이스 VPC FleetWise 엔드포인트를 통해 AWS IoT에 연결

인터페이스 VPC FleetWise 엔드포인트를 통해 AWS IoT에 연결

AWS 인터넷을 FleetWise 통해 연결하는 대신 VPC (가상 사설 클라우드 AWS PrivateLink) 의 인터페이스 VPC 엔드포인트 () 를 사용하여 IoT에 직접 연결할 수 있습니다. 인터페이스 VPC 엔드포인트를 사용하는 경우 VPC와 AWS FleetWise IoT 간의 통신은 전적으로 네트워크 내에서 수행됩니다. AWS 각 VPC 엔드포인트는 하나 이상의 탄력적 네트워크 인터페이스(ENI) 및 VPC 서브넷의 프라이빗 IP 주소로 표현됩니다.

인터페이스 VPC 엔드포인트는 인터넷 게이트웨이, NAT 디바이스, VPN 연결 또는 연결 FleetWise 없이 VPC를 AWS IoT에 직접 연결합니다. AWS Direct Connect VPC의 인스턴스는 AWS IoT FleetWise API와 통신하는 데 퍼블릭 IP 주소가 필요하지 않습니다.

VPC를 FleetWise 통해 AWS IoT를 사용하려면 VPC 내부의 인스턴스에서 연결하거나 (VPN) 또는 를 사용하여 VPC에 프라이빗 네트워크를 연결해야 합니다. AWS Virtual Private Network AWS Direct

Connect Amazon VPN에 대한 내용은 Amazon Virtual Private Cloud 사용 설명서의 VPN 연결을 참조하세요. 에 대한 AWS Direct Connect자세한 내용은 사용 설명서의 연결 생성을 참조하십시오.AWS Direct Connect

AWS 콘솔 또는 AWS Command Line Interface (AWS CLI) 명령을 사용하여 AWS FleetWise IoT에 연결할 인터페이스 VPC 엔드포인트를 생성할 수 있습니다. 자세한 내용은 인터페이스 엔드포인트 생성을 참조하세요.

인터페이스 VPC 엔드포인트를 생성한 후 엔드포인트에 프라이빗 DNS 호스트 이름을 활성화하면 기본 AWS IoT 엔드포인트가 VPC FleetWise 엔드포인트로 확인됩니다. AWS IoT의 기본 서비스 이름 엔드포인트는 다음 FleetWise 형식입니다.

```
iotfleetwise.Region.amazonaws.com
```

프라이빗 DNS 호스트 이름을 사용하지 않는 경우 Amazon VPC는 다음 형식으로 사용할 수 있는 DNS 엔드포인트 이름을 제공합니다.

```
VPCE_ID.iotfleetwise.Region.vpce.amazonaws.com
```

자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 VPC 엔드포인트 (AWS PrivateLink) 를 참조하십시오.

AWS IoT는 VPC 내의 모든 API 작업에 대한 호출을 FleetWise 지원합니다.

VPC 엔드포인트 정책을 VPC 엔드포인트에 연결하여 IAM 보안 주체에 대한 액세스를 제어할 수 있습니다. 보안 그룹을 VPC 엔드포인트와 연결하여 IP 주소 범위와 같은 네트워크 트래픽의 소스와 대상을 기반으로 인바운드 및 아웃바운드 액세스를 제어할 수도 있습니다. 자세한 정보는 VPC 엔드포인트를 통해 서비스에 대한 액세스 제어를 참조하세요.

IoT를 위한 VPC 엔드포인트 정책 생성 AWS FleetWise

AWS IoT용 Amazon VPC 엔드포인트에 대한 정책을 FleetWise 생성하여 다음을 지정할 수 있습니다.

- 작업을 수행할 수 있거나 수행할 수 없는 보안 주체
- 수행 가능 작업 또는 수행 불가 작업

자세한 정보는 Amazon VPC 사용 설명서의 VPC 엔드포인트를 통해 서비스에 대한 액세스 제어를 참조하십시오.

Example — 지정된 계정으로부터의 모든 액세스를 거부하는 VPC 엔드포인트 정책 AWS

다음 VPC 엔드포인트 정책은 엔드포인트를 사용하는 모든 API 호출을 AWS 계정 *123456789012* 거부합니다.

```
{
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        },
        {
            "Action": "*",
            "Effect": "Deny",
            "Resource": "*",
            "Principal": {
                "AWS": [
                    "123456789012"
                ]
            }
        }
    ]
}
```

Example – 지정된 보안 주체(사용자)에 대해서만 VPC 액세스를 허용하는 VPC 엔드포인트 정책

VPC ##### ### ## 123456789012# ## ###### ## #### #####. AWS 다른 모든 IAM 주체가 엔드포인트에 액세스하는 것을 거부합니다.

```
{
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::123456789012:user/lijuan"
                ]
            }
        }]
```

```
}
```

Example — IoT AWS 작업을 위한 VPC 엔드포인트 정책 FleetWise

다음은 AWS IoT를 위한 엔드포인트 정책의 FleetWise 예시입니다. *###### ### ## # ### 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",
            ]
           }
    ]
}
```

IoT의 구성 및 취약성 분석 AWS FleetWise

IoT 환경은 다양한 기능을 수행하고 장기적으로 사용되며 지리적으로 분산된 다수의 장치로 구성될 수 있습니다. 이러한 특성으로 인해 장치 설정이 복잡해지고 오류가 발생하기 쉬워집니다. 디바이스의 컴퓨팅 파워, 메모리 및 스토리지 기능이 한정된 경우가 많으므로 이에 따라 디바이스 자체에서 암호화 및 다른 형태의 보안을 사용하는 데 제약이 있습니다. 디바이스는 종종 취약점이 알려진 소프트웨어를 사용합니다. 이러한 요인으로 인해 IoT용 데이터를 수집하는 차량을 FleetWise 비롯한 AWS IoT 장치는 해커의 매력적인 표적이 되며 지속적으로 보안을 유지하기가 어렵습니다.

구성 및 IT 제어는 귀하와 당사 고객 간의 AWS 공동 책임입니다. 자세한 내용은 AWS 공동 책임 모델을 참조하십시오.

AWS IoT를 위한 보안 베스트 프랙티스 FleetWise

AWS IoT는 자체 보안 정책을 개발하고 구현할 때 고려해야 할 여러 보안 기능을 FleetWise 제공합니다. 다음 모범 사례는 일반적인 지침이며 완벽한 보안 솔루션을 나타내지는 않습니다. 이러한 모범 사례는 환경에 적절하지 않거나 충분하지 않을 수 있으므로 참고용으로만 사용하십시오.

AWS IoT 보안에 대해 알아보려면 AWS IoT 개발자 안내서의 보안 모범 사례를 참조하십시오. AWS IoT Core

가능한 최소 권한 부여

IAM 역할에 최소 권한 집합을 사용하여 최소 권한 원칙을 따릅니다. IAM 정책의 `Action` 및 `Resource` 속성에 대한 `*` 와일드카드 사용을 제한합니다. 대신, 가능한 경우 한정된 작업과 리소스를 선언합니다. 최소 권한 및 기타 정책 모범 사례에 대한 자세한 내용은 the section called “정책 모범 사례”을(를) 참조하세요.

민감한 정보를 기록하지 않음

자격 증명 및 기타 개인 식별 정보(PII)의 로깅을 방지해야 합니다. 다음과 같은 보호 장치를 구현하는 것이 좋습니다.

- 디바이스 이름에 민감한 정보를 사용하지 않습니다.
- 캠페인 이름, 디코더 매니페스트, 차량 모델, 신호 카탈로그, 차량 및 차량 ID 등 AWS IoT FleetWise 리소스의 이름 및 ID에 민감한 정보를 사용하지 마십시오.

API 호출 기록을 보는 데 사용합니다 AWS CloudTrail .

보안 분석 및 운영 문제 해결 목적으로 계정에서 이루어진 AWS IoT FleetWise API 호출 기록을 볼 수 있습니다. 계정에서 이루어진 AWS IoT FleetWise API 호출 기록을 받으려면 를 CloudTrail 켜면 됩니다 AWS Management Console. 자세한 정보는 the section called “CloudTrail 로그”을 참조하세요.

장치의 시계를 동기화 상태로 유지

장치에서는 정확한 시간을 유지하는 것이 중요합니다. X.509 인증서에는 만료 날짜와 시간이 있습니다. 장치의 시계는 서버 인증서가 여전히 유효한지 확인하는 데 사용됩니다. 장치 시계는 시간이 지나 드리프트 상태가 되거나 배터리가 방전될 수 있습니다.

자세한 내용은 AWS IoT Core 개발자 가이드의 디바이스 시계를 동기화하기 모범 사례를 참조하세요.

AWS IoT FleetWise 모니터링

AWS IoT FleetWise 및 다른 AWS 솔루션의 신뢰성, 가용성 및 성능을 유지하려면 모니터링이 중요합니다. AWS는 AWS IoT FleetWise를 모니터링하고, 이상이 있을 때 이를 보고하고, 필요한 경우 자동 조치를 취할 수 있도록 다음과 같은 모니터링 도구를 제공합니다.

- Amazon CloudWatch는 AWS에서 실행하는 AWS 리소스와 애플리케이션을 실시간으로 모니터링합니다. 지표를 수집 및 추적하고, 사용자 지정 대시보드를 생성할 수 있으며, 지정된 지표가 지정한 임계값에 도달하면 사용자에게 알리거나 조치를 취하도록 경보를 설정할 수 있습니다. 예를 들어 CloudWatch에서 Amazon EC2 인스턴스의 CPU 사용량 또는 기타 지표를 추적하고 필요할 때 자동으로 새 인스턴스를 시작할 수 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하세요.
- Amazon CloudWatch Logs를 사용하여 Amazon EC2 인스턴스, CloudTrail, 기타 소스의 로그 파일을 모니터링, 저장 및 액세스할 수 있습니다. CloudWatch Logs는 로그 파일의 정보를 모니터링하고 특정 임곗값에 도달하면 사용자에게 알릴 수 있습니다. 또한 매우 내구력 있는 스토리지에 로그 데이터를 저장할 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs 사용 설명서를 참조하세요.
- AWS CloudTrail는 AWS 계정에서 또는 이 계정을 대신하여 수행된 API 호출 및 관련 이벤트를 캡처합니다. 그러고 나서 사용자가 지정한 Amazon S3 버킷에 로그 파일을 전송합니다. 어떤 사용자 및 계정이 AWS를 호출했는지 어떤 소스 IP 주소에 호출이 이루어졌는지 언제 호출이 발생했는지 확인할 수 있습니다. 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하세요.

Amazon CloudWatch로 AWS IoT FleetWise 모니터링

Amazon CloudWatch 지표는 AWS 리소스와 그 성능을 모니터링하는 방법입니다. AWS IoT FleetWise는 지표를 CloudWatch에 전송합니다. AWS Management Console, AWS CLI 또는 API를 사용하여 AWS IoT FleetWise가 CloudWatch로 전송하는 지표를 나열할 수 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하세요.

> **Important**
>
> AWS IoT FleetWise가 지표를 CloudWatch로 전송할 수 있도록 설정을 구성해야 합니다. 자세한 내용은 설정 구성 섹션을 참조하세요.

`AWS/IoTFleetWise` 네임스페이스에 포함된 지표는 다음과 같습니다.

신호 지표

지표	Description
IllegalMessageFromEdge	차량에서 보내고 AWS IoT FleetWise에서 수신한 메시지가 필요한 형식과 일치하지 않았습니다. 단위: 개 차원: VehicleName 유효 통계: Sum
MessageThrottled	차량에서 AWS IoT FleetWise로 전송된 메시지가 제한되었습니다. 이는 현재 지역에서 이 계정의 서비스 한도를 초과했기 때문입니다. 단위: 개 차원: VehicleName 유효 통계: Sum
ModelingError	차량에서 전송되고 AWS IoT FleetWise에서 수신한 메시지에는 차량 모델에 대한 검증에 실패한 신호가 포함되어 있습니다. 단위: 개 차원: ModelManifestName 유효 통계: Sum
DecodingError	차량에서 전송되고 AWS IoT FleetWise에서 수신한 메시지에는 차량의 디코더 매니페스트에 대한 검증에 실패한 신호가 포함되어 있습니다. 단위: 개 차원: DecoderName

지표	Description
	유효 통계: Sum

캠페인 지표

지표	Description
VehicleNotFound	차량에서 전송되고 AWS IoT FleetWise에서 수신한 메시지로, 차량을 알 수 없습니다. 단위: 개 차원: VehicleName 유효 통계: Sum
CampaignInvalid	차량에서 전송되고 AWS IoT FleetWise에서 수신한 메시지로, 캠페인이 유효하지 않은 경우입니다. 단위: 개 차원: CampaignName 유효 통계: Sum
CampaignNotFound	차량에서 전송되고 AWS IoT FleetWise에서 수신한 메시지로, 캠페인이 알려지지 않은 경우입니다. 단위: 개 차원: CampaignName 유효 통계: Sum

캠페인 데이터 대상 지표

지표	Description
TimestreamWriteError	AWS IoT FleetWise가 차량으로부터 Amazon Timestream 테이블에 메시지를 기록하지 못했습니다. 단위: 개 차원: DatabaseName, TableName 유효 통계: Sum
S3WriteError	AWS IoT FleetWise는 차량에서 Amazon Simple Storage Service(S3) 버킷에 메시지를 쓰지 못했습니다. 단위: 개 차원: BucketName 유효 통계: Sum
S3ReadError	AWS IoT FleetWise가 Amazon Simple Storage Service(S3) 버킷에서 차량의 객체 키를 읽지 못했습니다. 단위: 개 차원: BucketName 유효 통계: Sum

고객 관리 AWS KMS 주요 지표

지표	Description
KMSKeyAccessDenied	AWS KMS 키 액세스 거부 오류 때문에 AWS IoT FleetWise가 차량의 메시지를 Timestream

지표	Description
	테이블 또는 Amazon S3 버킷에 쓸 수 없었습니다. 단위: 개 차원: KMSKeyId 유효 통계: Sum

Amazon CloudWatch Logs를 통한 AWS IoT FleetWise 모니터링

Amazon CloudWatch Logs는 리소스에서 발생하는 이벤트를 모니터링하고 문제가 발생할 경우 사용자에게 알립니다. 알림을 받으면 로그 파일에 액세스하여 특정 이벤트에 대한 정보를 얻을 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs 사용 설명서를 참조하세요.

CloudWatch 콘솔에서 AWS IoT FleetWise 로그 보기

> **Important**
>
> CloudWatch 콘솔에서 AWS IoT FleetWise 로그 그룹을 확인하기 전에 다음 사항이 올바른지 확인합니다.
>
> - AWS IoT FleetWise에서 로깅을 활성화했습니다. 로깅에 대한 자세한 내용은 AWS IoT FleetWise 로깅 구성 섹션을 참조하세요.
> - 작업에서 작성한 AWS IoT 로그 항목이 이미 있습니다.

CloudWatch 콘솔에서 AWS IoT FleetWise 로그를 보려는 경우

1. CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그, 로그 그룹을 선택합니다.
3. 로그 그룹을 선택합니다.
4. 로그 그룹 검색을 선택합니다. 계정에 대해 생성된 로그 이벤트의 전체 목록이 표시됩니다.
5. 확장 아이콘을 선택하면 개별 스트림을 살펴보고 로그 수준이 ERROR와(과) 같은 모든 로그를 찾을 수 있습니다.

이벤트 필터링 검색 상자에 쿼리를 입력할 수도 있습니다. 예를 들면, 다음 쿼리를 수행할 수 있습니다.

```
{ $.logLevel = "ERROR" }
```

필터 표현식 생성에 대한 자세한 내용은 Amazon CloudWatch Logs 사용 설명서의 필터 패턴 구문을 참조하세요.

Example 로그 항목

```
{
  "accountId": "123456789012",
  "vehicleName": "test-vehicle",
  "message": "Unrecognized signal ID",
  "eventType": "MODELING_ERROR",
  "logLevel": "ERROR",
  "timestamp": 1685743214239,
  "campaignName": "test-campaign",
  "signalCatalogName": "test-catalog",
  "signalId": 10242
}
```

신호 이벤트 유형

이벤트 유형	Description
MODELING_ERROR	차량에서 전송되고 AWS IoT FleetWise에서 수신한 메시지에는 차량 모델에 대한 검증에 실패한 신호가 포함되어 있습니다. 속성: 차량 이름, 캠 페인 이름, 신호 카탈로그 이름, 신호 ID, 신호 값, 신호 값 범위 최소, 신호 값 범위 최대, 모델 매니페스트 이름
ILLEGAL_MESSAGE_FROM_EDGE	차량에서 보내고 AWS IoT FleetWise에서 수신한 메시지가 필요한 형식과 일치하지 않았습니다. 속성: 차량 이름, 캠페인 이름, 신호 카탈로그 이름

이벤트 유형	Description
DECODING_ERROR	차량에서 전송되고 AWS IoT FleetWise에서 수신한 메시지에는 차량의 디코더 매니페스트에 대한 검증에 실패한 신호가 포함되어 있습니다. 속성: campaignName, signalCatalogName, decoderManifestName, (선택 사항) signalName, (선택 사항) s3URI

캠페인 이벤트 유형

이벤트 유형	Description
VEHICLE_NOT_FOUND	차량에서 전송되어 AWS IoT FleetWise가 수신한 메시지로, 차량이 알려지지 않았습니다. 속성: 차량 이름, 캠페인 이름
CAMPAIGN_NOT_FOUND	차량에서 전송되어 AWS IoT FleetWise가 수신한 메시지로, 캠페인이 알려지지 않았습니다. 속성: vehicleName(선택 사항), campaignName
CAMPAIGN_INVALID	차량에서 전송되어 AWS IoT FleetWise가 수신한 메시지로, 캠페인이 유효하지 않습니다. 속성: vehicleName(선택 사항), campaignName

캠페인 데이터 대상 이벤트 유형

이벤트 유형	Description
TIMESTREAM_WRITE_ERROR	AWS IoT FleetWise가 차량으로부터 Amazon Timestream 테이블에 메시지를 기록하지 못했습니다.

이벤트 유형	Description
	속성: vehicleName, campaignName, timestreamDatabaseName, timestreamTableName
S3_WRITE_ERROR	AWS IoT FleetWise가 차량에서 Amazon Simple Storage Service(S3) 버킷에 메시지를 쓰지 못했습니다. 속성: campaignName, destinationName
S3_READ_ERROR	AWS IoT FleetWise가 Amazon Simple Storage Service(S3) 버킷에서 차량의 객체 키를 읽지 못했습니다. 속성: campaignName, destinationName

고객이 관리하는 AWS KMS 키 이벤트 유형

이벤트 유형	Description
KMS_KEY_ACCESS_DENIED	AWS KMS키 액세스 거부 오류 때문에 AWS IoT FleetWise가 차량의 메시지를 Timestream 테이블 또는 Amazon S3 버킷에 쓸 수 없었습니다.

속성

모든 CloudWatch Logs 로그 항목에는 다음 속성이 포함됩니다.

accountId

귀하의 AWS 계정 ID.

eventType

로그가 작성된 이벤트 유형입니다. 이벤트 유형의 값은 로그 항목을 생성한 이벤트에 따라 다릅니다. 각 로그 항목 설명에는 해당 로그 항목의 `eventType` 값이 포함됩니다.

logLevel

사용 중인 로그 수준. 자세한 내용은 https://docs.aws.amazon.com/iot/latest/developerguide/configure-logging.html#log-level 개발자 가이드의 AWS IoT Core로그 레벨 섹션을 참조하세요.

message

로그에 대한 특정 세부 정보가 들어 있습니다.

타임스탬프

AWS IoT FleetWise가 로그를 처리한 시점의 에포크 밀리초 타임스탬프입니다.

선택적 속성

CloudWatch Logs 항목에는 `eventType`에 따라 다음 속성이 선택적으로 포함됩니다.

decoderManifestName

신호가 포함된 디코더 매니페스트의 이름입니다.

destinationName

차량 데이터의 대상의 이름입니다. Amazon S3 버킷 이름입니다.

campaignName

캠페인의 이름.

signalCatalogName

신호가 포함된 신호 카탈로그의 이름입니다.

signalId

오류 신호의 ID입니다.

signalIds

오류 신호 ID 목록.

signalName

신호의 이름입니다.

signalTimestampEpochMs

오류 신호의 타임스탬프.

signalValue

오류 신호의 값입니다.

signalValueRangeMax

오류 신호의 최대 범위입니다.

signalValueRangeMin

오류 신호의 최소 범위.

s3URI

차량 메시지에 있는 Amazon Ion 파일의 Amazon S3 고유 식별자입니다.

timestreamDatabaseName

Timestream 데이터베이스의 이름입니다.

timestreamTableName

Timestream 테이블의 이름입니다.

vehicleName

차량 모델의 이름입니다.

AWS IoT FleetWise 로깅 구성

AWS IoT FleetWatch 로그 데이터를 CloudWatch 로그 그룹으로 전송할 수 있습니다. CloudWatch Logs는 AWS IoT FleetWise가 차량의 메시지를 처리하지 못하는 경우에 대한 가시성을 제공합니다. 예를 들어 잘못된 구성이나 기타 클라이언트 오류로 인해 이러한 문제가 발생할 수 있습니다. 오류가 발생하면 알림을 받게 되므로 문제를 식별하고 완화할 수 있습니다.

CloudWatch로 로그를 전송하려면 CloudWatch 로그 그룹을 생성해야 합니다. AWS IoT FleetWise에서 사용한 것과 동일한 계정 및 동일한 리전으로 로그 그룹을 구성합니다. AWS IoT FleetWise에서 로깅을 활성화할 때 로그 그룹 이름을 제공하세요. 로깅이 활성화되면 AWS IoT FleetWise는 로그 스트림의 CloudWatch 로그 그룹에 로그를 전송합니다.

AWS IoT FleetWise에서 전송된 로그 데이터는 클라우드워치 콘솔에서 볼 수 있습니다. CloudWatch 로그 그룹 구성 및 로그 데이터 보기에 대한 자세한 내용은 로그 그룹 작업을 참조하세요.

로그를 CloudWatch에 게시하기 위한 권한

CloudWatch 로그 그룹에 대한 로깅을 구성하려면 이 섹션에 설명된 권한 설정이 필요합니다. 권한 관리에 관한 자세한 내용은 IAM 사용 설명서의 AWS 리소스에 대한 액세스 관리를 참조하세요.

이러한 권한이 있으면 로깅 구성을 변경하고, CloudWatch에 대한 로그 전송을 구성하고, 로그 그룹에 대한 정보를 검색할 수 있습니다.

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

모든 AWS 리소스에 대해 작업이 허용되면 정책에서 "Resource" 설정이 "*"로 표시됩니다. 즉, 각 작업이 지원하는 모든 AWS 리소스에 대해 해당 작업이 허용됩니다.

AWS IoT 플릿와이즈에서 로깅 구성(콘솔)

이 섹션에서는 AWS IoT FleetWise 콘솔을 사용하여 로깅을 구성하는 방법에 대해 설명합니다.

AWS IoT FleetWise 콘솔을 사용하여 로깅을 구성하려는 경우

1. AWS IoT FleetWise 콘솔을 엽니다.
2. 왼쪽 창에서 설정을 선택합니다.
3. 설정페이지의 로깅 섹션에서 편집을 선택합니다.
4. CloudWatch 로깅 섹션에서 로그 그룹을 입력합니다.
5. 변경 사항을 저장하려면 제출을 선택합니다.

로깅을 활성화한 후 CloudWatch 콘솔에서 로그 데이터를 볼 수 있습니다.

AWS IoT FleetWise(CLI)에 기본 로그인 구성

이 섹션에서는 CLI를 사용하여 AWS IoT FleetWise에 대한 로깅을 구성하는 방법에 대해 설명합니다.

여기에 표시된 CLI 명령에 해당하는 AWS API의 메서드를 사용하여 API로 이 절차를 수행할 수도 있습니다. GetLoggingOptions API 작업을 사용하여 현재 구성을 가져오고 PutLoggingOptions API 작업을 사용하여 구성을 수정할 수 있습니다.

CLI를 사용하여 IoT AWS FleetWise 로깅을 구성하려는 경우

1. 계정에 대한 로깅 옵션을 설정하려면, get-logging-options 명령을 사용합니다.

   ```
   aws iotfleetwise get-logging-options
   ```

2. 로깅을 활성화하려면 put-logging-options 명령을 사용합니다.

   ```
   aws iotfleetwise put-logging-options --cloud-watch-log-delivery
    logType=ERROR,logGroupName=MyLogGroup
   ```

   여기서 각 항목은 다음과 같습니다.

logType

CloudWatch Logs에 데이터를 전송할 로그 유형입니다. 로깅을 비활성화하려면 값을 OFF(으)로 변경합니다.

logGroupName

작업이 데이터를 보내는 대상 CloudWatch Logs 그룹입니다. AWS IoT FleetWise에 대한 로깅을 활성화하기 전에 로그 그룹 이름을 생성해야 합니다.

로깅을 활성화한 후에는 AWS CLI를 사용하여 로그 항목 검색을 참조하세요.

AWS CloudTrail을(를) 사용하여 AWS IoT 플릿와이즈 API 호출 로깅

AWS IoT FleetWise는 사용자, 역할 또는 AWS IoT FleetWise의 AWS 서비스에서 수행한 작업의 기록을 제공하는 서비스인 AWS CloudTrail와(과) 통합됩니다. CloudTrail은 AWS IoT FleetWise에 대한 모든 API 호출을 이벤트로 캡처합니다. 캡처된 호출에는 AWS IoT FleetWise 콘솔의 호출과 AWS IoT FleetWise API 작업에 대한 코드 호출이 포함됩니다. 추적을 생성하면 AWS IoT FleetWise용 이벤트를 포함한 CloudTrail 이벤트를 지속적으로 Amazon S3 버킷에 배포할 수 있습니다. 추적을 구성하지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록에서 최신 이벤트를 볼 수 있습니다. CloudTrail에서 수집한 정보를 사용하여 AWS IoT FleetWise에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하세요.

CloudTrail의 AWS IoT FleetWise 정보

CloudTrail은 계정 생성 시 AWS 계정에서 사용되도록 설정됩니다. AWS IoT FleetWise에서 활동이 발생하면 해당 활동은 이벤트 기록에 있는 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. AWS 계정에서 최신 이벤트를 확인, 검색 및 다운로드할 수 있습니다. 자세한 내용은 CloudTrail 이벤트 기록을 사용하여 이벤트 보기를 참조하세요.

AWS IoT Fleet Hub에 대한 이벤트를 포함하여 AWS 계정에 이벤트를 지속적으로 기록하려면 추적을 생성합니다. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 콘솔에서 추적을 생성하면 기본적으로 모든 AWS 리전에 추적이 적용됩니다. 추적은 AWS 파티션에 있는 모든 리전의 이벤트를 로깅하고 지정된 Amazon S3 버킷으로 로그 파일을 전송합니다. 또는

CloudTrail 로그에서 수집된 이벤트 데이터를 추가 분석 및 처리하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음 자료를 참조하세요.

- 추적 생성 개요
- CloudTrail 지원 서비스 및 통합
- CloudTrail에 대한 Amazon SNS 알림 구성
- 여러 리전에서 CloudTrail 로그 파일 수신
- 여러 계정에서 CloudTrail 로그 파일 수신

모든 AWS IoT FleetWise 작업은 CloudTrail에서 로깅되고 AWS API 참조에 기록됩니다. 예를 들어 `CreateCampaign`, `AssociateVehicleFleet` 및 `GetModelManifest` 작업을 호출하면 CloudTrail 로그 파일에 항목이 생성됩니다.

모든 이벤트 및 로그 항목에는 요청을 생성한 사용자에 대한 정보가 들어 있습니다. 자격 증명 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트로 했는지 아니면 IAM 사용자 보안 인증 정보로 했는지 여부.
- 역할 또는 연합된 사용자에 대한 임시 보안 자격 증명을 사용하여 요청이 생성되었는지 여부.
- 다른 AWS 서비스에서 요청했는지 여부.

자세한 내용은 CloudTrail userIdentity 요소를 참조하세요.

AWS IoT FleetWise 로그 파일 항목의 이해

추적이란 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 입력할 수 있게 하는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. 이벤트는 모든 소스의 단일 요청을 나타내며 요청된 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보를 포함합니다. CloudTrail 로그 파일은 퍼블릭 API 호출의 주문 스택 트레이스가 아니므로 특정 순서로 표시되지 않습니다.

다음 예제는 *`AssociateVehicleFleet`* 작업을 보여주는 CloudTrail 로그 항목입니다.

```
{
      "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 개발자 가이드용 문서 이력

다음 표에서는 AWS IoT FleetWise에 대한 문서 릴리스를 소개합니다.

변경 사항	설명	날짜
비전 시스템 데이터 미리 보기	AWS IoT FleetWise의 비전 시스템 데이터 미리 보기를 사용하여 카메라, 레이더, 라이다를 비롯한 차량 비전 시스템에서 데이터를 수집하고 구성할 수 있습니다. IoT FleetWise는 정형 및 비정형 비전 시스템 데이터, 메타데이터(이벤트 ID, 캠페인, 차량), 표준 센서 데이터(텔레메트리 데이터)를 클라우드에서 자동으로 동기화된 상태로 유지합니다.	2023년 11월 26일
AWS KMS 고객 관리형 키	AWS IoT FleetWise는 이제 AWS KMS 고객 관리형 키를 지원합니다. KMS 키를 사용하여 AWS 클라우드에 저장된 AWS IoT FleetWise 리소스(신호 카탈로그, 차량 모델, 디코더 매니페스트, 차량 및 데이터 수집 캠페인 구성)와 관련된 서버 측 데이터를 암호화할 수 있습니다.	2023년 10월 16일
Amazon S3의 객체 스토리지	AWS IoT FleetWise는 이제 Amazon Simple Storage Service(S3)를 사용한 데이터 저장을 지원합니다. Amazon Timestream 외에도 Amazon	2023년 6월 1일

	S3에 캠페인 중에 수집된 데이터를 저장할 수 있습니다.	
정식 출시	이것은 AWS IoT FleetWise의 공개 릴리스입니다.	2022년 9월 27일
최초 릴리스	AWS IoT FleetWise 개발자 가이드의 프리뷰 릴리스입니다.	2021년 11월 30일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.