

# Collecting vehicle data more efficiently with AWS IoT FleetWise

by Aruna Ravi | on 16 DEC 2021 | in [Announcements](#), [Automotive](#), [AWS Re:Invent](#), [Internet Of Things](#) | [Permalink](#) | [Share](#)

Today, we think of connected vehicles as an advanced class of vehicle with internet connectivity. However, we will soon refer to these simply as vehicles, since by 2030, over 95 percent of new vehicles sold globally will be connected to the internet, up from about 50 percent today.<sup>1</sup> Greater vehicle connectivity gives automakers opportunities to improve vehicle quality, safety, and autonomy, but it also brings challenges—namely, how to efficiently collect and utilize the massive amounts of data generated by connected vehicles. In this post, we will walk through [AWS IoT FleetWise](#), a new service that makes it easy and cost effective for you to collect and transform data from millions of vehicles and transfer it to the cloud in near-real time. Once the data is in the cloud, you can use it for tasks like analyzing fleet-wide vehicle health or training machine learning (ML) models that improve autonomous driving and advanced driver assistance systems (ADAS).

## Challenges with collecting vehicle data

### Data variety

Every variation of a vehicle model generates data in a unique format, which causes a mind-boggling amount of potential unique vehicle data configurations, data structures, and schemas. For example, an automaker may have 10-15 models in its lineup, with each model having hybrid, all-wheel drive (AWD), and advanced safety options.

Additionally, most vehicle data is not readable by humans and is encoded in proprietary formats specific to automakers or suppliers, such as data sent over a vehicle's [Controller Area Network Bus \(CAN Bus\)](#). To make the data usable, automakers must first decode it then reconcile it across their fleets. For example, data coming from a fuel pressure sensor might be represented as Fuel\_Press on model A, and Injector\_Press on model B. Collecting and reconciling this data across multiple variations of vehicle models is a heavy lift and requires automakers to build, scale, and maintain custom data collections systems.

### Data volume

Not only are there increasing numbers of connected vehicles, but each vehicle also has increasing numbers of sensors generating data. Each sensor has capacity to generate richer data, especially advanced sensors like radars and cameras. For example, vehicles today now have multiple cameras, and cameras are evolving from 1 to 3 to 8 megapixels. In short, data volume is increasing at an exponential rate, which makes it more difficult to manage.

As vehicles continue transitioning to higher levels of autonomy, automakers need to transfer increasing volumes of data to cloud so they can use it for continuous AI/ML model training and improvement. However, cloud data transfer is cost prohibitive across a fleet of production vehicles. A single autonomous vehicle can generate up to 2 TiBs of data hourly per vehicle. As a result, automakers often resort to using autonomous test fleets with specially built on-board storage as a work-around for getting the data they need to train AI/ML models.

## Getting started with AWS IoT FleetWise

### Pre-requisites

AWS IoT FleetWise has both cloud and embedded software components. You can deploy AWS IoT FleetWise completely in the cloud before deploying on physical vehicle hardware to simulate collecting vehicle data; the only prerequisite is an AWS account and an [Amazon Timestream](#) table. To deploy on physical hardware and real-life vehicles, AWS IoT FleetWise Edge requires a POSIX-based operating system (OS). Knowledge of C/C++, POSIX APIs, and in-vehicle networking protocols such as CAN and external connectivity protocols such as MQTT are helpful when using AWS IoT FleetWise.

## Model a virtual vehicle

AWS IoT FleetWise helps solve the data variety problem with virtual vehicle modeling. When you model a vehicle in the cloud, you standardize vehicle attributes (e.g. a two-door coupe) and sensors (e.g. fuel pressure, engine temperature) across multiple vehicle types, so a signal like fuel pressure is always represented as fuel\_pressure. This modeling process allows for easy fleet-wide data analysis in the cloud.

To create a virtual vehicle, use the [AWS IoT FleetWise Console](#) or [APIs](#) to upload automotive standard files (such as a CANDBC), which AWS IoT FleetWise parses into a draft virtual vehicle model. You also have the choice to pick one of the pre-configured templates in AWS IoT FleetWise, such as OBD-II signals, which automatically creates a vehicle model for you based on the OBD-II standard.

To create an OBD standard model:

1. Open the [AWS IoT FleetWise Console](#).
2. Navigate to the **Vehicle models** menu item.
3. Click the **Add provided template** button.
4. Select OBD\_II, and input CAN Channel (Default is can0) and click **Add**.

AWS IoT FleetWise

Vehicle models

Settings

Successfully created 1 model.  
The model has been created.

Vehicle models

OBD II

OBD\_II

Create decoder manifest

Summary

Vehicle model ARN  
arn:aws:iotfleetwise:us-east-1:490825878307:model-manifest/OBD\_II

Signal catalog ARN  
arn:aws:iotfleetwise:us-east-1:490825878307:signal-catalog/DefaultSignalCatalog

Date created  
November 22, 2021, 22:21 (UTC-08)

Status  
ACTIVE

Description  
-

Signals

Decoder manifests

Vehicles

Signals (294)

Name	Fully qualified name	Data type	Signal type	Unit	Description
AbsoluteLoad	OBD.AbsoluteLoad	DOUBLE	Sensor	percent	PID 43 - Absolute load value
AbsoluteThrottlePositionG	OBD.AbsoluteThrottlePositionG	DOUBLE	Sensor	-	Throttle Position G
AcceleratorPositionD	OBD.AcceleratorPositionD	DOUBLE	Sensor	percent	PID 49 - Accelerator pedal position D
AcceleratorPositionE	OBD.AcceleratorPositionE	DOUBLE	Sensor	percent	PID 4A - Accelerator pedal position E
AcceleratorPositionF	OBD.AcceleratorPositionF	DOUBLE	Sensor	percent	PID 4B - Accelerator pedal position F
ActualEgrA	OBD.ActualEgrA	DOUBLE	Sensor	-	Commanded EGR and EGR Error
ActualEgrB	OBD.ActualEgrB	DOUBLE	Sensor	-	Commanded EGR and EGR Error
ActualEngineTorque	OBD.ActualEngineTorque	DOUBLE	Sensor	-	Actual engine - percent torque
AmbientAirTemperature	OBD.AmbientAirTemperature	DOUBLE	Sensor	celsius	PID 46 - Ambient air temperature
AvgDemandedReagentConsumption	OBD.AvgDemandedReagentConsumption	DOUBLE	Sensor	-	NOx reagent system
AvgReagentConsumption	OBD.AvgReagentConsumption	DOUBLE	Sensor	-	NOx reagent system
Bank1Sensor1Catalyst	OBD.Bank1Sensor1Catalyst	DOUBLE	Sensor	celsius	PID 3C - Catalyst temperature from bank 1, sensor 1
Bank1Sensor2Catalyst	OBD.Bank1Sensor2Catalyst	DOUBLE	Sensor	celsius	PID 3E - Catalyst temperature from bank 1, sensor 2
Bank2Sensor1Catalyst	OBD.Bank2Sensor1Catalyst	DOUBLE	Sensor	celsius	PID 3D - Catalyst temperature from bank 2, sensor 1
Bank2Sensor2Catalyst	OBD.Bank2Sensor2Catalyst	DOUBLE	Sensor	celsius	PID 3F - Catalyst temperature from bank 2, sensor 2

When you create an OBD model, AWS IoT FleetWise creates a decoder manifest automatically for you based on the OBD standard. The decoder manifest allows AWS IoT FleetWise to decode the proprietary signals on your vehicle. You can view decoder manifests within the vehicle model detail page:

Once you have a model and associated decoder manifest, you can create vehicles using the [Create Vehicle API](#).

## Set up rules-based data collection

## OBD\_II

Duplicate

Create vehicle 

Create decoder manifest

After a decoder manifest is associated with this vehicle model, choose Create vehicle. Follow the instructions in the AWS IoT FleetWise Developer Guide to use the API to vehicles from the vehicle model. You also need to create campaigns for your vehicles.

Summary [Info](#)

Vehicle model ARN

 arn:aws:iotfleetwise:us-east-1:490825878307:model-manifest/OBD\_II

Signal catalog ARN

 arn:aws:iotfleetwise:us-east-1:490825878307:signal-catalog/DefaultSignalCatalog

Date created

November 25, 2021, 02:38 (UTC-08)

Status

 ACTIVE

Description

-

Signals


Decoder manifests

Vehicles

Decoder manifests (1) [Info](#)

Delete

 Filter decoder manifests by name< 1 > 

	Name ▲	ARN	Description ▼
<input type="radio"/>	DefaultDecoderManifest	 arn:aws:iotfleetwise:us-east-1:490825878307:decoder-manifest/DefaultDecoderManifest	-

AWS IoT FleetWise helps solve the data volume problem with rules-based data collection, which reduces the amount of unnecessary data transferred to the cloud. You select what data to collect, such as data from safety equipment, EV battery charge, or any other data generated by the vehicle's sensors. Then, you define rules and events for when to transfer that data based on parameters such as weather, location, or vehicle type. Setting up these data collection rules helps to keep costs low and gives access to more useful data.

The rules you define are contained within JSON documents known as schemes. There are two primary types of schemes: time-based collection and event-based collection. Time-based collection selects signals of your choosing at a given time interval as shown below:

The below scheme collects the Throttle Position signal every 10000MS or 10 seconds.

JSON

```
{
  "compression": "SNAPPY",
  "diagnosticsMode": "SEND_ACTIVE_DTCS",
  "spoolingMode": "TO_DISK",
  "collectionScheme": {
    "timeBasedCollectionScheme": {
      "periodMs": 10000
    }
  },
  "postTriggerCollectionDuration": 0,
  "signalsToCollect": [
    {
      "maxSampleCount": 1,
      "signalName": "Throttle__Position"
    }
  ]
}
```

An event-based collection scheme is similar to time-based, but instead of collecting data at regular time intervals, you create a rule to trigger AWS IoT FleetWise to collect data. Below is an example event-based collection scheme, which collects two signals [Vehicle\_Speed and Instant\_Torque] when a specific condition is met; specifically, when the throttle position is greater than 0. AWS IoT FleetWise will collect these signals for 1000ms after the event is detected as instructed by the “postTriggerCollectionDuration” field in this scheme.

JSON

```
{
  "compression": "SNAPPY",
  "diagnosticsMode": "SEND_ACTIVE_DTCS",
  "spoolingMode": "TO_DISK",
  "collectionScheme": {
    "conditionBasedCollectionScheme": {
      "conditionLanguageVersion": 1,
      "expression": "$variable.`Throttle__Position` > 0",
      "minimumTriggerIntervalMs": 1000,
      "triggerMode": "RISING_EDGE"
    }
  },
  "postTriggerCollectionDuration": 1000,
  "signalsToCollect": [
    {
      "maxSampleCount": 10,
      "signalName": "Vehicle_Speed"
    },
    {
```

Once you create schemes, you deploy them to vehicles using the [create and approve campaign](#) operations within the AWS IoT FleetWise Console. Once schemes deploy to vehicles, you will see data start flowing through AWS IoT FleetWise into your Amazon Timestream database.

## Conclusion

In this post, we showed how AWS IoT FleetWise helps you standardize vehicle data through vehicle modeling and intelligently filter data with rules-based data collection. Overall, these capabilities help you avoid the heavy lift of building custom data collection systems as well as the expense and complexity of transferring unnecessary vehicle data to the cloud.

To learn more, head over to our [AWS IoT FleetWise](#) site or [login to the console](#) to get started. We look forward to your feedback and questions.

*<sup>1</sup>McKinsey Center for Future Mobility, 2021*

## About the author

Aruna Ravi is the Product Manager for AWS IoT FleetWise.

TAGS: [automotive](#), [AWS IoT](#), [iot](#)