AWS Connected Vehicle Solution

AWS Implementation Guide

Sean Senior

Chris Rec

Hitendra Nishar

Tom Horton

November 2017

Last updated: January 2018 (see revisions)

Contents

About This Guide

This implementation guide discusses architectural considerations and configuration steps for deploying the Amazon Web Services (AWS) connected vehicle solution on the AWS Cloud. It includes links to an AWS CloudFormation template that launches and configures the AWS services required to deploy this solution using AWS best practices for security and availability.

The guide is intended for IT infrastructure architects, administrators, and DevOps professionals who have practical experience with connected vehicle services and the AWS Cloud.

Overview

Amazon Web Services (AWS) enables automotive manufacturers and suppliers to build serverless IoT applications that gather, process, analyze, and act on connected vehicle data, without having to manage any infrastructure.

With AWS IoT, customers can connect vehicles and devices to the AWS Cloud securely, with low latency and with low overhead. Customers can combine AWS IoT with other AWS services to build event-driven applications that help track vehicle diagnostics and health, predict required maintenance, and provide recommendations to car owners.

The connected vehicle solution provides secure vehicle connectivity to the AWS Cloud, and includes capabilities for local computing within vehicles, sophisticated event rules, and data processing and storage. The solution features fast and robust data ingestion; highly reliable and durable storage of vehicle telemetry data; simple, scalable big data services for analyzing the data; and global messaging and application services to connect with consumers.

This solution is designed to provide a framework for connected vehicle services, allowing you to focus on extending the solution's functionality rather than managing the underlying infrastructure operations. You can build upon this framework to address a variety of use cases such as voice interaction, navigation and other location-based services, remote vehicle diagnostics and health monitoring, predictive analytics and maintenance alerts, media streaming services, vehicle safety and security services, head unit applications, and mobile applications.

Cost

You are responsible for the cost of the AWS services used while running this solution. The total cost for running the connected vehicle solution depends on the amount of data being requested, collected, stored, processed, and presented. For full details, see the pricing webpage for each AWS service you will be using in this solution.

Architecture Overview

Deploying this solution with the **default parameters** builds the following environment in the AWS Cloud.

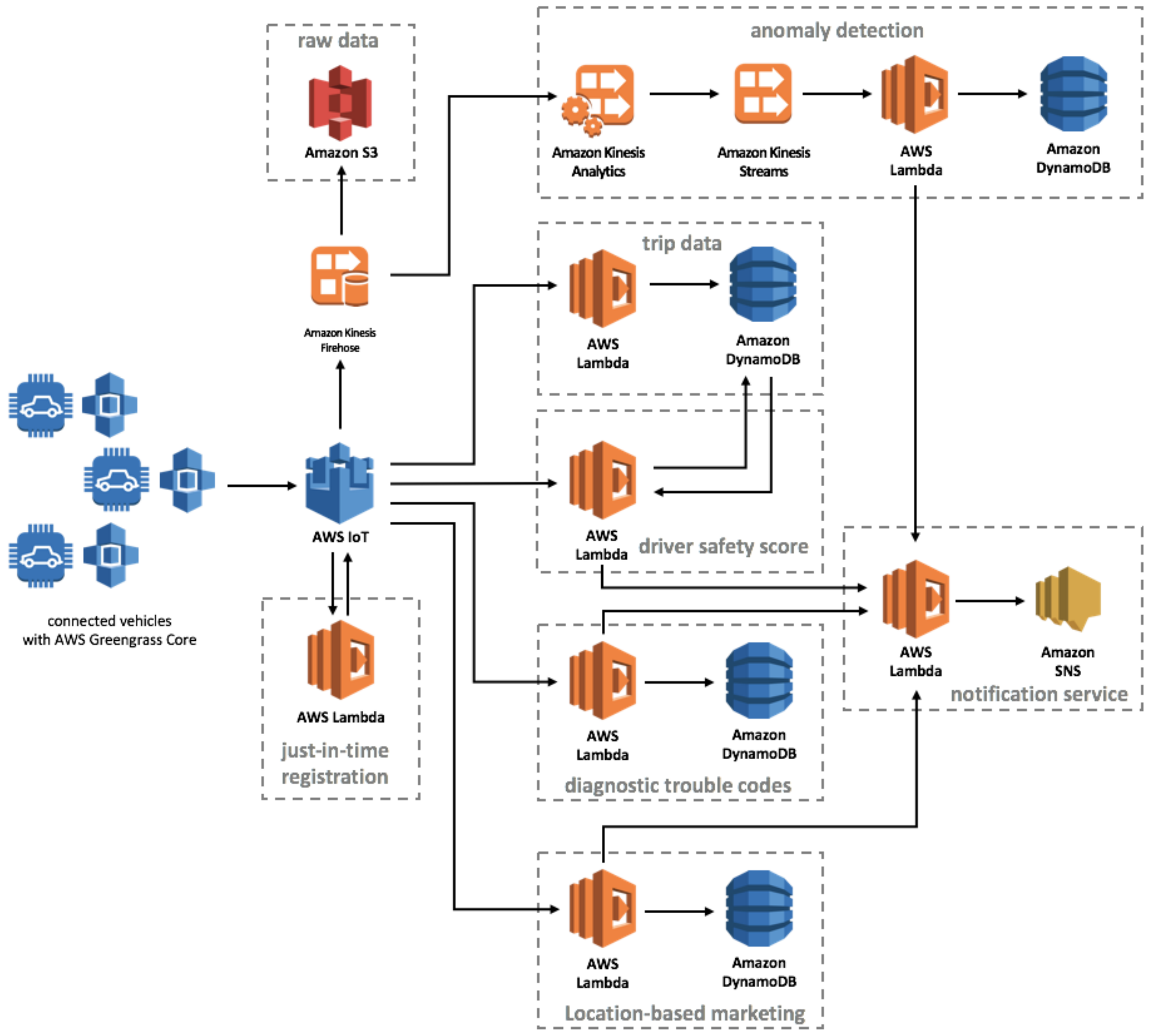

Figure 1: The connected vehicle solution architecture

This connected vehicle solution leverages the AWS IoT platform which authenticates messages from connected vehicles and processes data according to six business rules. The solution's AWS CloudFormation template deploys eight unique Amazon DynamoDB tables that store various details about vehicle health, trips, and vehicle owners; a set of microservices (AWS Lambda functions) that process messages and data; an Amazon Kinesis Firehose delivery stream that encrypts and loads data to an Amazon Simple Storage Service (Amazon S3) bucket; an Amazon Kinesis Analytics application that analyzes data for anomalies; an Amazon Kinesis stream which enables real-time processing of anomalous data; and an Amazon Simple Notification Service (Amazon SNS) topic which sends alerts to users.

When AWS IoT receives a message, it authenticates and authorizes the message and the Rules Engine executes the appropriate rule on the message, which routes the message to the appropriate backend application.

Anomaly Detection

An AWS IoT rule sends telematics data to an Amazon Kinesis Firehose delivery stream, which encrypts and streams raw vehicle telematics data to an Amazon S3 bucket for future analysis or replay. An Amazon Kinesis Analytics application analyzes data from the delivery stream using a windowed machine learning algorithm to detect anomalies in the data. If an anomaly is detected, the record is sent to an Amazon Kinesis stream which invokes a Lambda function that parses the record and stores it in a DynamoDB table. The Lambda function also triggers an Amazon SNS notification to users with the detected anomaly.

Trip Data

The trip data AWS IoT rule invokes a Lambda function that processes vehicle telematics data during a trip and stores it in a DynamoDB table. Data is continuously updated until the trip is completed.

Driver Safety Score

The driver safety score AWS IoT rule detects the end of a trip and invokes a Lambda function that processes aggregate trip data, and executes an algorithm to generate a driver's safety score. The function then triggers an Amazon SNS notification to the driver with their safety score. The score is added to the trip data DynamoDB table.

Diagnostic Trouble Codes

The diagnostic trouble code AWS IoT rule detects diagnostic trouble codes in the IoT topic and invokes a Lambda function that stores the trouble code in a DynamoDB table, translates the trouble code into layman’s terms, and triggers an Amazon SNS notification to the user that contains the translated trouble code.

Location-Based Marketing

The location-based marketing AWS IoT rule detects the location of the vehicle and invokes a Lambda function that determines whether the vehicle is near a point of interest. If the vehicle is near a point of interest, the function logs the location in a DynamoDB table and triggers an Amazon SNS notification to the user that contains an advertisement.

Data Interaction

The solution configures Amazon API Gateway to host the solution’s RESTful APIs, and deploys an Amazon Cognito user pool, which you can use to add user registration and sign-in to your existing connected vehicle applications.

Existing applications and third-party service providers can interact with connected vehicle data securely through the RESTful APIs. The APIs act as a “front door” for access to vehicle data stored in Amazon DynamoDB. You can also use the APIs to access any extended functionality you build into the solution.

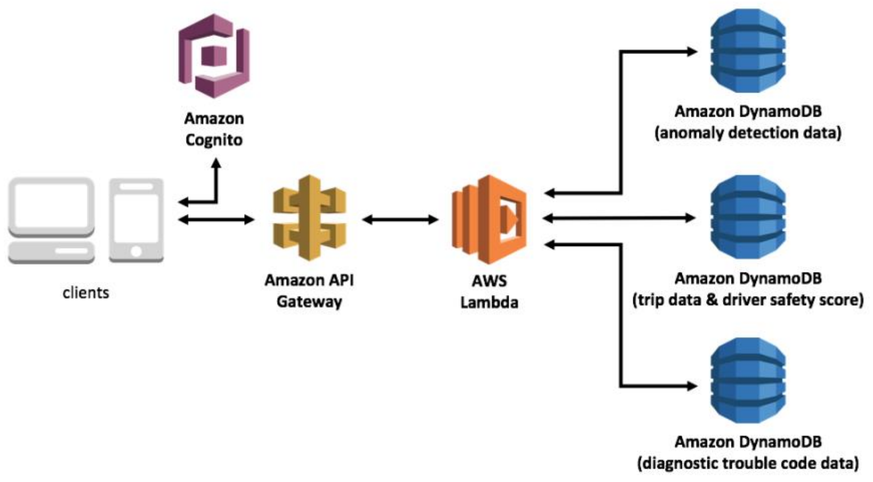
Amazon Cognito
clients
Amazon API Gateway
AWS Lambda
Amazon DynamoDB (anomaly detection data)
Amazon DynamoDB (trip data & driver safety score)
Amazon DynamoDB (diagnostic trouble code data)

Figure 2: How applications interact with connected vehicle solution data

The applications can use Amazon Cognito to authenticate users. Once authenticated, users can submit requests to the solution's APIs (Amazon API Gateway endpoints). Based on the request, Amazon API Gateway invokes the appropriate Lambda function to perform the necessary tasks on the data stored in the Amazon DynamoDB tables. You can use this data, as well as near-real-time MQTT data, to build detailed graphs, charts, and reports.

Components

AWS Greengrass Core

Customers can use AWS Greengrass Core to send telematics data to the connected vehicle solution. AWS Greengrass Core provides a message broker for preprocessing vehicle data that is sent to AWS IoT. AWS Greengrass Core also helps manage over-the-air updates by acting as a listener for updates to the instrument cluster and in-vehicle infotainment unit, and an orchestrator for executing those changes locally.

The minimum hardware requirements for AWS Greengrass Core are:

- Minimum 1GHz of compute
- Minimum 128MB of RAM
- Linux kernel version 4.4.11+ with OverlayFS and user namespace enabled

- CPU Architectures: x86_64, ARMv7, AArch64 (ARMv8)

AWS Greengrass Core devices can be configured to communicate with one another in an AWS Greengrass Group. If the AWS Greengrass Core device loses connection to the cloud, devices in the AWS Greengrass Group can continue to communicate with each other over the local network.

AWS IoT Device SDK

Customers can also use the AWS IoT Device SDK to send data to the solution. The AWS IoT Device SDK helps you to easily and quickly connect your hardware device to AWS IoT. For example, you can leverage the AWS IoT Device SDK to build a substrate layer that scans on-board diagnostic data (OBD-II) and publishes sensor data to AWS IoT via the AWS Greengrass Core.

Device Gateway

The AWS IoT Device Gateway enables devices to securely and efficiently communicate with AWS IoT. For this solution, connected vehicles communicate with the Device Gateway using a publication/subscription model. In the pub/sub model, vehicles publish messages to specific logical communication channels called *topics*. Vehicles subscribe to the topics to receive messages. The solution includes the following topics.

Message Type	Topic	Action	Description
Telematics	connectedcar/telemetry/<VIN>	publish	Vehicle sensor and telematics data `{ timestamp:x, trip_id:x, vin:x, name:x, value:x }`
Vehicle Trip Info	connectedcar/trip/<VIN>	publish	Aggregated trip data
Diagnostic Trouble Code	connectedcar/dtc/<VIN>	publish	Diagnostic trouble codes (DTC) `{ timestamp:x, trip_id:x, vin:x, name:'dtc', value:x }`
Anomaly Alert	connectedcar/alert/<VIN>/anomaly	subscribe	Anomaly detection alert `{ type:’anomaly’,message:x }`
DTC Alert	connectedcar/alert/<VIN>/dtc	subscribe	DTC alert `{ type:’dtc’,message:x }`
Driver Score Alert	connectedcar/alert/<VIN>/driverscore	subscribe	Driver safety score alert `{ type:’driverscore’,message:x }`
Advertisement Alert	connectedcar/telemetry/<VIN>/info	subscribe	Advertisement alert `{ type:’info’,message:x }`

Rules Engine

When a connected vehicle publishes a message to the connected vehicle solution, the AWS IoT Rules Engine evaluates, transforms, and delivers the message to the appropriate backend services based on defined rules. The solution includes the following rules.

Message Type	Topic	Description
ConnectedVehicleJITR	`SELECT * FROM '$aws/events/certificates/registered/<CA CERTIFICATE ID>'`	Activates unknown certificates signed by registered CAs and attaches vehicle identification number (VIN) policies to them
ConnectedVehicleTelematicsStorage	`SELECT * FROM 'connectedcar/telemetry/#'`	Processes inbound vehicle telemetry data and sends messages to persistent storage
ConnectedVehicleTelematicsDtc	`SELECT * FROM 'connectedcar/dtc/#'`	Selects inbound vehicle diagnostic trouble code (DTC) data and triggers an AWS Lambda function to process DTC information
ConnectedVehicleTrip	`SELECT * FROM 'connectedcar/trip/#'`	Selects inbound vehicle aggregated trip data and stores it in a DynamoDB table
ConnectedVehicleDriverScore	`SELECT * FROM 'connectedcar/trip/#' WHERE ignition_status = 'off'`	Detects the end of a trip, then triggers a Lambda function to calculate a driver safety score from aggregated trip data
ConnectedVehicleLocationBasedMarketing	`SELECT * FROM 'connectedcar/telemetry/#' WHERE name = 'location'`	Selects inbound vehicle location and triggers a Lambda function to determine whether the vehicle is located near a point of interest

The Rules Engine can be configured to route inbound telemetry data from AWS IoT to several other AWS services such as Amazon Kinesis Streams or Amazon DynamoDB, or from one AWS IoT topic to another. Data can also be sent to custom applications running on AWS Lambda, giving manufacturers maximum flexibility and power to process connected vehicle data.

Authentication

This solution takes advantage of mutual authentication and encryption at all points of connection to AWS IoT to ensure that data is never exchanged between the vehicle and AWS IoT without proven identity. We recommend leveraging MQTT connections with X.509 certificate based authentication for vehicles that connect to this solution. You can register your preferred Certificate Authority (CA), which is used to sign and issue the vehicle certificate(s), with AWS IoT. Each registered vehicle certificate has a policy that allows that vehicle to publish and subscribe only to topics associated with its vehicle identification number (VIN).

Just-in-Time Registration

When a vehicle connects to AWS IoT for the first time, AWS IoT detects the unknown certificate. If the certificate is signed by a registered CA, the connected vehicle solution attempts to register the vehicle certificate automatically during the Transport Layer Security (TLS) handshake.

An MQTT registration event is published on a registration topic associated with the registered CA certificate. The registration event invokes an AWS Lambda function that activates the certificate and attaches the VIN policy to it. When the certificate is activated and the policy is attached, the certificate can be used for authentication and authorization with AWS IoT.

User Management

After the connected vehicle solution is deployed, administrators can invite privileged users and customize their permissions to implement granular access-control policies. Privileged users can browse, search, and access data. They can also build and maintain cloud applications, and invite customers to use those applications.

This solution also integrates with Microsoft Active Directory.

Logging and Metrics

The connected vehicle solution logs API calls, latency, and error rates to Amazon CloudWatch which you can use to set alarms based on defined thresholds. The connected vehicle solution also monitors traffic at the REST API level. Optionally, you can enable detailed metrics for each method of the Connected vehicle solution REST API from the Amazon API Gateway deployment configuration console. Detailed metrics will incur an extra cost.

Regional Deployment

This solution uses Amazon Cognito, AWS IoT, Amazon Kinesis Firehose, Amazon Kinesis Analytics, and Amazon Kinesis Streams which are available in specific AWS Regions only. Therefore, you must deploy this solution in a region that supports these services. For the most current service availability by region, see the AWS service offerings by region.

AWS CloudFormation Template

This solution uses AWS CloudFormation to automate the deployment of the connected vehicle solution. It includes the following AWS CloudFormation template, which you can download before deployment:

View template

aws-connected-vehicle-solution.template: Use this template to launch the solution and all associated components. The default configuration deploys Amazon Kinesis Firehose delivery streams, an Amazon Kinesis Analytics application, an Amazon Kinesis stream, Amazon DynamoDB tables, AWS Lambda functions, an Amazon Simple Storage Service (Amazon S3) bucket and an Amazon Simple Notification Service (Amazon SNS) topic, Amazon API Gateway, and an Amazon Cognito user pool, but you can also customize the template based on your specific needs.

Automated Deployment

Before you launch the automated deployment, please review the architecture, configuration, and other considerations discussed in this guide. Follow the step-by-step instructions in this section to configure and deploy the connected vehicle solution into your account.

Time to deploy: Approximately five minutes

Launch the Stack

This automated AWS CloudFormation template deploys the connected vehicle solution.

Note: You are responsible for the cost of the AWS services used while running this solution. See the Cost section for more details. For full details, see the pricing webpage for each AWS service you will be using in this solution.

1. Sign in to the AWS Management Console and click the button to the right to launch the `aws-connected-vehicle-solution` AWS CloudFormation template.

   Launch Solution

   You can also download the template as a starting point for your own implementation.

2. The template is launched in the US East (N. Virginia) Region by default. To launch the solution in a different AWS Region, use the region selector in the console navigation bar.

   Note: This solution uses Amazon Cognito, AWS IoT, Amazon Kinesis Firehose, Amazon Kinesis Analytics, and Amazon Kinesis Streams which are available in specific AWS Regions only. Therefore, you must launch this solution an AWS Region where these services are available. [1]

3. On the **Select Template** page, verify that you selected the correct template and choose **Next**.
4. On the **Specify Details** page, assign a name to your solution stack and choose **Next**.
5. On the **Options** page, choose **Next**.
6. On the **Review** page, review and confirm the settings. Be sure to check the box acknowledging that the template will create AWS Identity and Access Management (IAM) resources.
7. Choose **Create** to deploy the stack.

   You can view the status of the stack in the AWS CloudFormation Console in the **Status** column. You should see a status of CREATE_COMPLETE in roughly five minutes.

Note: In addition to the AWS Lambda functions that process and store data, this solution includes the `connected-vehicle-helper` Lambda function, which runs only during initial configuration or when resources are updated or deleted.

When running this solution, the `connected-vehicle-helper` function is inactive. However, do not delete the function as it is necessary to manage associated resources.

Security

When you build systems on AWS infrastructure, security responsibilities are shared between you and AWS. This shared model can reduce your operational burden as AWS operates, manages, and controls the components from the host operating system and virtualization layer down to the physical security of the facilities in which the services operate. For more information about security on AWS, visit the AWS Security Center.

IAM Roles

When you configure an existing application to communicate with the connected vehicle solution, you must specify which data the application can access and process. The connected vehicle solution automatically creates a custom AWS Identity and Access Management role for the application that permits it to get and decrypt the data.

[1] For the most current service availability by region, see https://aws.amazon.com/about-aws/global-infrastructure/regional-product-services/

Authentication

AWS IoT provides mutual authentication and encryption at all points between the connected device and the AWS IoT Device Gateway so that data is never exchanged without proven identity. AWS IoT supports Signature Version 4 and X.509 certificate based authentication. With AWS IoT, you can use AWS IoT generated certificates as well as those signed by your preferred Certificate Authority (CA).

Additional Resources

AWS services

- AWS IoT
- Amazon Kinesis Firehose
- Amazon Kinesis Analytics
- Amazon Kinesis Streams
- Amazon DynamoDB
- Amazon Simple Storage Service
- Amazon Simple Notification Service
- AWS CloudFormation
- AWS Lambda

Appendix: Connected Vehicle Solution API

The connected vehicle solution API enables you to expose collected vehicle data in a secure manner. The API acts as a "front door" for applications to access vehicle data, business logic, and extended functionality from the connected vehicle backend microservices.

This solution uses an Amazon Cognito user pool integrated with Amazon API Gateway (`cc-authorizer`) for identification and authorization. When a user pool is used with the API, clients are only allowed to call user pool enabled methods after they provide a valid identity token.

The following operations are available in the connected vehicle solution API.

Vehicle

- GET /vehicles
- POST /vehicles
- GET /vehicles/{vin}

Anomalies

- GET /vehicles/{vin}/anomalies
- GET /vehicles/{vin}/anomalies/{anomaly_id}
- PUT /vehicles/{vin}/anomalies/{anomaly_id}/acknowledge

Diagnostic Trouble Codes

- GET /vehicles/{vin}/dtc
- GET /vehicles/{vin}/dtc/{dtc_id}
- PUT / vehicles/{vin}/dtc/{dtc_id}/acknowledge

Health Reports

- GET /vehicles/{vin}/healthreports
- GET /vehicles/{vin}/healthreports /{report _id}

Trips

- GET /vehicles/{vin}/trips

- GET /vehicles/{vin}/ trips /{trip _id}

GET /vehicles

Description

The `GET /vehicles` operation enables you to retrieve the vehicle ID and other details about the vehicle from the vehicle's endpoint.

Response

Name	Description
`vin`	The vehicle identification number (VIN) for the vehicle
`owner_id`	The unique user ID of the vehicle owner
`nickname`	The vehicle nickname
`odometer`	The vehicle's odometer reading

For information about the errors that are common to all actions, see Common Errors.

POST /vehicles

Description

The `POST /vehicles` operation enables you to create a new registered vehicle for an owner. The response includes details about the newly registered vehicle.

Request Body

Name	Description
`vin`	The VIN for the vehicle
`nickname`	The vehicle nickname

Response

Name	Description
`vin`	The VIN for the vehicle
`owner_id`	The unique user ID of the vehicle owner
`nickname`	The vehicle nickname
`odometer`	The vehicle's odometer reading

For information about the errors that are common to all actions, see Common Errors.

GET /vehicles/{vin}

Description

The `GET /vehicle/{vin}` operation enables you to retrieve information about a specific vehicle registered to a user.

Request Parameter

`vin`

The VIN used to filter vehicles

Type: String

Required: Yes

Response

Name	Description
`owner_id`	The unique user ID of the vehicle owner
`nickname`	The vehicle nickname
`odometer`	The vehicle's odometer reading

For information about the errors that are common to all actions, see Common Errors.

GET /vehicles/{vin}/anomalies

Description

The `GET /vehicle/{vin}/anomalies` operation enables you to retrieve information about a specific vehicle's historical anomalies.

Request Parameter

`vin`

The VIN used to filter anomalies

Type: String

Required: Yes

Response

Name	Description
`anomaly_id`	The unique identifier of the anomaly

Name	Description
`vin`	The VIN for the vehicle
`telemetric`	The telemetric where the anomaly occurred
`value`	The telemetric value where the anomaly occurred
`anomaly_score`	The assessed score of the identified anomaly
`acknowledged`	Flag indicating that the anomaly is acknowledged
`description`	The description of the anomaly
`identified_at`	The date and time the anomaly was identified
`measured_at`	The date and time the anomaly was measured
`created_at`	The date and time the anomaly was created

For information about the errors that are common to all actions, see Common Errors.

GET /vehicles/{vin}/anomalies/{anomaly_id}

Description

The GET `/vehicle/{vin}/anomalies/{anomaly_id}` operation enables you to retrieve information about a specific anomaly for a specific vehicle.

Request Parameter

`vin`

The VIN used to filter anomalies

Type: String

Required: Yes

`anomaly_id`

The anomaly to return

Type: String

Required: Yes

Response

Name	Description
`anomaly_id`	The unique identifier of the anomaly

Name	Description
`vin`	The VIN for the vehicle
`telemetric`	The telemetric where the anomaly occurred
`value`	The telemetric value where the anomaly occurred
`anomaly_score`	The assessed score of the identified anomaly
`acknowledged`	Flag indicating that the anomaly is acknowledged
`description`	The description of the anomaly
`identified_at`	The date and time the anomaly was identified
`measured_at`	The date and time the anomaly was measured
`created_at`	The date and time the anomaly was created

For information about the errors that are common to all actions, see Common Errors.

PUT /vehicles/{vin}/anomalies/{anomaly_id}/acknowledge

Description

The `GET /vehicle/{vin}/anomalies/{anomaly_id}/acknowledge` operation enables you to acknowledge a specific anomaly for a specific vehicle.

Request Parameter

`vin`

The VIN used to filter anomalies

Type: String

Required: Yes

`anomaly_id`

The anomaly to return

Type: String

Required: Yes

Response

Name	Description
`anomaly_id`	The unique identifier of the anomaly

Name	Description
`vin`	The VIN for the vehicle
`telemetric`	The telemetric where the anomaly occurred
`value`	The telemetric value where the anomaly occurred
`anomaly_score`	The assessed score of the identified anomaly
`acknowledged`	Flag indicating that the anomaly is acknowledged
`description`	The description of the anomaly
`identified_at`	The date and time the anomaly was identified
`measured_at`	The date and time the anomaly was measured
`created_at`	The date and time the anomaly was created

For information about the errors that are common to all actions, see Common Errors.

GET /vehicles/{vin}/dtc

Description

The `GET /vehicle/{vin}/dtc` operation enables you to retrieve information about a specific vehicle's historical diagnostic trouble codes.

Request Parameter

`vin`

The VIN used to filter diagnostic trouble codes

Type: String

Required: Yes

Response

Name	Description
`dtc_id`	The unique identifier of the diagnostic trouble code
`vin`	The VIN for the vehicle
`dtc`	The diagnostic trouble code identifier
`description`	The description of the diagnostic trouble code
`steps`	The resolution step of the diagnostic trouble code
`acknowledged`	Flag indicating that the diagnostic trouble code is acknowledged
`description`	The description of the diagnostic trouble code

Name	Description
`identified_at`	The date and time the diagnostic trouble code was identified
`measured_at`	The date and time the diagnostic trouble code was measured

Resolution Step

Name	Description
`id`	The resolution step identifier
`detail`	The resolution step details

For information about the errors that are common to all actions, see Common Errors.

GET /vehicles/{vin}/dtc/{dtc_id}

Description

The `GET /vehicle/{vin}/dtc/{dtc_id}` operation enables you to retrieve information about a specific diagnostic trouble code for a specific vehicle.

Request Parameter

`vin`

The VIN used to filter diagnostic trouble codes

Type: String

Required: Yes

`dtc_id`

The diagnostic trouble code to return

Type: String

Required: Yes

Response

Name	Description
`dtc_id`	The unique identifier of the diagnostic trouble code
`vin`	The VIN for the vehicle

Name	Description
`dtc`	The diagnostic trouble code identifier
`description`	The description of the diagnostic trouble code
`steps`	The resolution step of the diagnostic trouble code
`acknowledged`	Flag indicating that the diagnostic trouble code is acknowledged
`description`	The description of the diagnostic trouble code
`identified_at`	The date and time the diagnostic trouble code was identified
`measured_at`	The date and time the diagnostic trouble code was measured

Resolution Step

Name	Description
`id`	The resolution step identifier
`detail`	The resolution step details

For information about the errors that are common to all actions, see Common Errors.

GET /vehicles/{vin}/dtc/{dtc_id}/acknowledge

Description

The `GET /vehicle/{vin}/dtc/{dtc_id}/acknowledge` operation enables you to acknowledge a specific diagnostic trouble code for a specific vehicle.

Request Parameter

`vin`

The VIN used to filter diagnostic trouble codes

Type: String

Required: Yes

`dtc_id`

The diagnostic trouble code to return

Type: String

Required: Yes

Response

Name	Description
`dtc_id`	The unique identifier of the diagnostic trouble code
`vin`	The VIN for the vehicle
`dtc`	The diagnostic trouble code identifier
`description`	The description of the diagnostic trouble code
`steps`	The resolution step of the diagnostic trouble code
`acknowledged`	Flag indicating that the diagnostic trouble code is acknowledged
`identified_at`	The date and time the diagnostic trouble code was identified
`measured_at`	The date and time the diagnostic trouble code was measured

For information about the errors that are common to all actions, see Common Errors.

GET /vehicles/{vin}/healthreports

Description

The `GET /vehicle/{vin}/healthreports` operation enables you to retrieve information about a vehicle's historical health report.

Request Parameter

`vin`

The VIN used to filter health reports

Type: String

Required: Yes

Response

Name	Description
`report_id`	The unique identifier of the health report
`vin`	The VIN for the vehicle
`owner_id`	The unique user ID of the vehicle owner

For information about the errors that are common to all actions, see Common Errors.

GET /vehicles/{vin}/healthreports/{report_id}

Description

The `GET /vehicle/{vin}/healthreports/{report_id}` operation enables you to retrieve information about a specific health report for a specific vehicle.

Request Parameter

`vin`

The VIN used to filter health reports

Type: String

Required: Yes

`report_id`

The health report to return

Type: String

Required: Yes

Response

Name	Description
`report_id`	The unique identifier of the health report
`vin`	The VIN for the vehicle
`owner_id`	The unique user ID of the vehicle owner

For information about the errors that are common to all actions, see Common Errors.

GET /vehicles/{vin}/trips

Description

The `GET /vehicle/{vin}/trips` operation enables you to retrieve information about a specific vehicle's historical trips.

Request Parameter

`vin`

The VIN used to filter trips

Type: String

Required: Yes

Response

Name	Description
`trip_id`	The unique identifier of the trip
`vin`	The VIN for the vehicle
`owner_id`	The unique user ID of the vehicle owner
`vehicle_speed_mean`	The mean vehicle speed, in kilometers per hour, during the trip
`engine_speed_mean`	The mean engine speed during the trip
`torque_at_transmission_mean`	The mean transmission torque speed during the trip
`oil_temp_mean`	The mean oil temperature during the trip
`accelerator_pedal_position_mean`	The mean accelerator pedal position during the trip
`brake_mean`	The mean brake pedal position during the trip
`odometer`	The difference between the odometer reading at the start of the trip and the end of the trip
`fuel_consumed_since_restart`	The fuel consumed during the trip
`fuel_level`	The fuel level after the trip
`start_latitude`	The latitude at the beginning of the trip
`start_longitude`	The longitude at the beginning of the trip
`stop_latitude`	The latitude at the end of the trip
`stop_longitude`	The longitude at the end of the trip
`start_time`	The time the trip started
`end_time`	The time the trip ended
`driver_safety_score`	The driver safety score for the trip

For information about the errors that are common to all actions, see Common Errors.

GET /vehicles/{vin}/trips/{trip_id}

Description

The `GET /vehicle/{vin}/trips` operation enables you to retrieve information about a specific trip for a specific vehicle.

Request Parameter

`vin`

The VIN used to filter trips

Type: String

Required: Yes

`trip_id`

The trip to return

Type: String

Required: Yes

Response

Name	Description
`trip_id`	The unique identifier of the trip
`vin`	The VIN for the vehicle
`owner_id`	The unique user ID of the vehicle owner
`vehicle_speed_mean`	The mean vehicle speed, in kilometers per hour, during the trip
`engine_speed_mean`	The mean engine speed during the trip
`torque_at_transmission_mean`	The mean transmission torque speed during the trip
`oil_temp_mean`	The mean oil temperature during the trip
`accelerator_pedal_position_mean`	The mean accelerator pedal position during the trip
`brake_mean`	The mean brake pedal position during the trip
`odometer`	The difference between the odometer reading at the start of the trip and the end of the trip
`fuel_consumed_since_restart`	The fuel consumed during the trip
`fuel_level`	The fuel level after the trip
`start_latitude`	The latitude at the beginning of the trip
`start_longitude`	The longitude at the beginning of the trip
`stop_latitude`	The latitude at the end of the trip
`start_longitude`	The longitude at the end of the trip
`start_time`	The time the trip started
`end_time`	The time the trip ended
`driver_safety_score`	The driver safety score for the trip

For information about the errors that are common to all actions, see Common Errors.

Common Errors

This section lists the common errors that all operations can return.

Error Code	Description
`400 Bad Request`	Invalid unique identifier
`404 Not Found`	Vehicle not found
`default`	Unexpected error

Send Us Feedback

We welcome your questions and comments. Please post your feedback on the AWS Solutions Discussion Forum.

You can visit our GitHub repository to download the templates and scripts for this solution, and to share your customizations with others.

Document Revisions

Date	Change	In sections
November 2017	Initial release	--
January 2018	Added location-based marketing functionality	Architecture Overview; Components