AWS Well-Architected Framework

Connected Mobility Lens

Connected Mobility Lens: AWS Well-Architected Framework

Table of Contents

Connected Mobility Lens

Publication date: **January 3, 2024** (*Document revisions*)

This paper describes the Connected Mobility Lens for the AWS Well-Architected Framework, which enables you to review and improve your cloud-based architectures and better understand the impact of design decisions. We present general design principles and specific best practices aligned to the six pillars of the Well-Architected Framework.

Introduction

Connected mobility refers to the integration of technology into transportation systems to improve the flow of traffic and enhance the overall mobility experience. This includes the use of connected vehicles, smart infrastructure, and advanced data analytics to help improve traffic flow, improve safety, and reduce emissions. Connected mobility also includes the integration of various modes of transportation, such as cars, public transit, bicycles, and pedestrian walkways, to create a seamless, efficient, and safe transportation system.

This initial version of the Connected Mobility Lens primarily addresses scenarios related to connected vehicles, with a focus on how each scenario impacts the architecture from the edge to the cloud. A connected vehicle is defined as a vehicle that can communicate with other systems outside of the vehicle. These scenarios and use cases are not exhaustive, but they encompass common patterns in connected vehicles. We present a background on each scenario, general considerations for the design of the system, and a reference architecture for customers to consider for how the scenarios can be implemented.

Lens availability

The Connected Mobility Lens is available as an AWS-official lens in the Lens Catalog of the AWS Well-Architected Tool.

To get started, follow the steps in Adding a lens to a workload and select the **Connected Mobility Lens**.

General design principles

The Well-Architected Framework identifies a set of general design principles to help customers design in the cloud for building and managing connected mobility workloads.

Improving the customer experience with data integrity and data privacy

Customer experience and confidence in technology and processes used by the automotive manufacturer are key considerations in buyer decisions. According to a recent report:

> "Consumers are not comfortable sharing personal information which results from uncertainty as to what information is being shared; how the information is being used; whether it is being stored; and, if so, stored securely."

Selecting synchronous versus asynchronous communication pattern

The selection of a synchronous versus asynchronous pattern depends upon your use cases. Request/Reply use cases, such as vehicle state management, require responses to be given synchronously. Other use cases, such as remote commands, require asynchronous communication. The asynchronous design pattern should always be considered first before switching to other patterns of communication. Asynchronicity leads to loosely coupled systems that are scalable, failure tolerant, and have evolvable architecture. Systems that must absorb vehicle traffic spikes and can accommodate asynchronous processing can improve reliability by allowing clients to quickly release resources by using message queues. A robust tracking mechanism should be implemented for the requests received, responses delivered and failures.

Event-driven architecture enables the global scale

Event-driven architecture (EDA) is better suited for building global scale distributed microservice-based applications. EDA can bring agility, cost optimization, and reliability to the connected mobility implementations. EDA requires an enterprise-wide strategy in designing data contracts for the event producers and consumers. As all the event-driven applications are distributed, it's important to use tracing to understand and observe service dependencies and diagnose any bottlenecks and issues in the application.

Optimizing the frequency of data transferred from vehicle to cloud

According to a study by McKinsey & Company, by 2030, 95% of the cars shipped globally will connect to the internet, generating 10 exabytes of data per month. Vehicle manufacturers have been addressing the data synchronization challenge between vehicle and cloud in different ways. Some companies are opting for low frequency rather than high frequency, while others are using off-peak hour scheduling of low priority data. Intelligent data filtering, sensor and event prioritization, and compression can also lead to lower data transfer charges and better resiliency in managing the data.

The biggest challenge with delaying the offloading of data is the buffer size in the vehicle. A recommended approach is to use rule- or condition-based data collection schemes. A promising approach to address these issues is the application of machine learning (ML) powered, context-aware communication that exploits the dynamics of the communication channel. This approach allows the scheduling of delay-tolerant transmissions in an opportunistic way, which increases the transmission efficiency with regard to data rate, packet loss probability, and energy consumption.

Alignment with local regulations in handling data transfer, storage, and usage

Connected vehicles operate in diverse geographies so it is essential to abide by the regulations to build customer trust and for privacy compliance. For more information, see Security design principles.

Don't guess about your performance requirements

The connected mobility scenarios generally have a time bound pattern of traffic generation. The observability tools can give a trend of historic traffic based on location, time and use cases. Certain features, such as remote commands to start the vehicle and set climate control, can have high traffic during peak hours of the day and peak seasons. It is recommended to have automated scaling of services to accommodate periodic and seasonal shifts in customer usage and demands.

Scenarios

In this section, we cover the six key scenarios that are common in many connected mobility implementations. We describe how they influence the design and architecture of your connected mobility application, also referred to as *connected mobility platform* in this whitepaper. We present the assumptions made for each of these scenarios, the common drivers for the design, and a reference architecture for how these scenarios could be implemented.

Scenarios

- CM-S01 Vehicle and user provisioning
- CM-S02 Vehicle connectivity management
- CM-S03 Vehicle data management and insights
- CM-S04 Connected mobility core services
- CM-S05 Connected mobility supported systems
- CM-S06 Customer experience management

CM-S01 Vehicle and user provisioning

Vehicle Provisioning links the Telematics Control Unit (TCU) with the Vehicle Identification Number (VIN). Vehicle Provisioning also enables secure and automatic provisioning of security certificates and installation of latest firmware. Such activities are performed before the vehicle leaves the factory or upon swapping of the TCU. End of Line (EOL) processes involve configuration and validation of the Telematics Control Unit which includes: setup the certificate for identity of the vehicle, provision the SIM with the Mobile Network Operator, and set the state of the vehicle in the Connected Mobility Platform. User Provisioning involves activating the Connected Mobility services for the customer in the Customer Relationship Management (CRM) and the Billing systems, and activate the SIM in the Mobile Network Operator (MNO) systems which allows the customer to actually use the service. The following are user stories in the Vehicle and User Provisioning scenario:

User stories

CM-S01-UC01 Vehicle provisioning at the factory: Configuration of the Telematics Control Unit to set up the certificate for identity of the vehicle. The Connected Mobility Systems also maps the

TCU with the VIN number in this step. All of this can be automatically initiated when the vehicle is started and the TCU registers itself to the provisioning service.

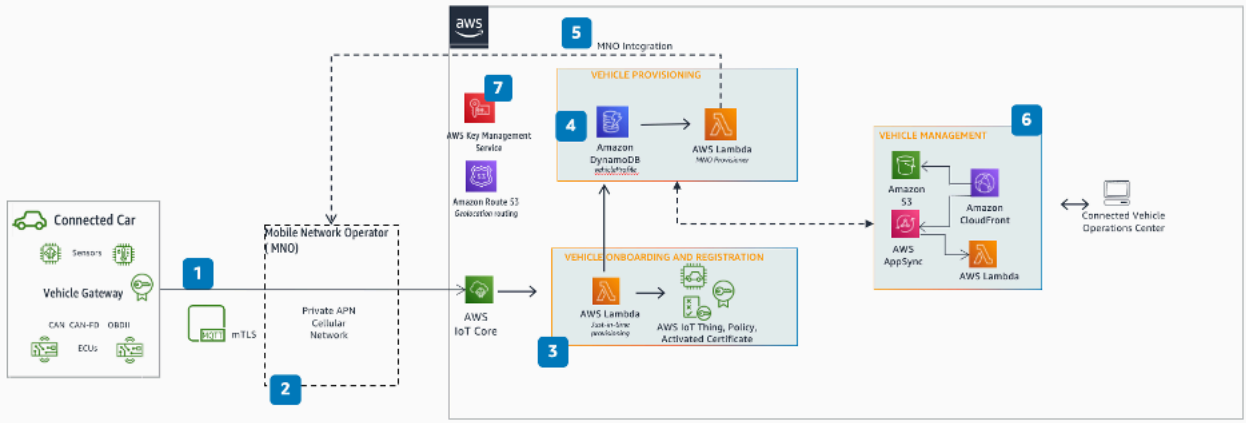
CM-S01-UC02 Mobile Network Operator (MNO) integration: The Telematics Control Unit (TCU) comes with a Subscriber Identity Module (SIM) which is used to transmit or receive data or SMS on the mobile network. As part of the vehicle provisioning process the API of the MNO is invoked to register the SIM. When the vehicle is delivered to the owner, the Connected Mobility service activation process will activate the SIM. Throughout the lifecycle of vehicle ownership, the vehicle owner may buy Mobility services like Entertainment, Wi-Fi Hotspot etc. which will require integration with the MNO to activate data packs. Troubleshooting any network connection issues will require integration with MNO's APIs to get the status and any health metrics

CM-S01-UC03 Driver profile management: When the user is provisioned, a Connected Mobility Platform may allow save and restore of one or more driver preferences per vehicle, such as seat adjustments, temperature preferences, media setting, data sharing preferences etc. These profiles may be portable to any vehicle owned or rented by the customer as long as they have an active Connected Mobility subscription.

CM-S01-UC04 Application and configuration updates: Update latest software and configurations in vehicles' electronic control units (ECUs) by sending continual and reliable updates. These updates provided by customers should:

- Have an audit trail of the update.
- Comply with security standards (such as Uptane).
- Comply with regulations (such as A-SPICE, UNR156, and ISO24089).
- Ensure safety of vehicle and occupants while installing updates by conducting checks (for example, vehicle operating status).
- Provide scalability to any number of vehicles.
- Have the ability to target a subset or fleet of vehicles.

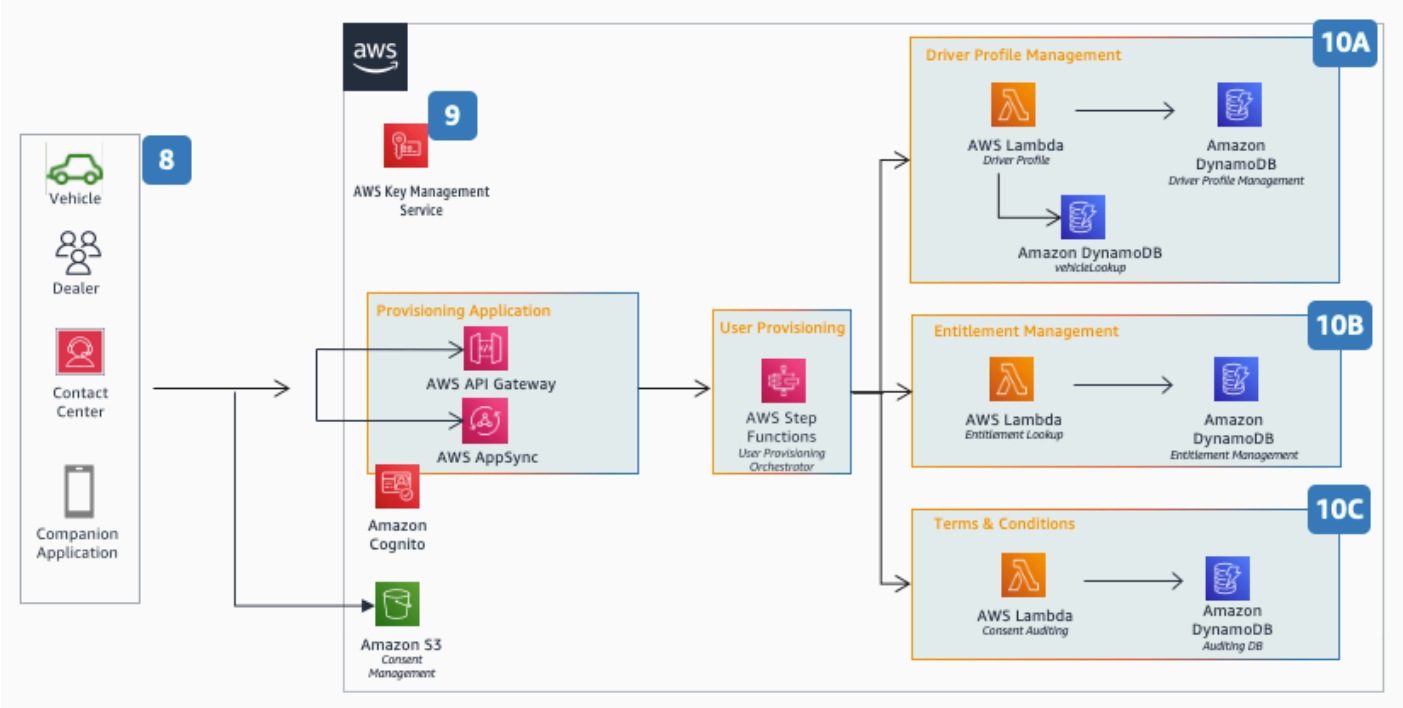
Reference architecture



Vehicle provisioning reference architecture

Figure 1: CM-S01-a: Vehicle provisioning reference architecture

1. Embedded in-vehicle devices with a unique identity principal (X.509 certificate) publish telemetry via MQTT to AWS IoT Core. To minimize in-vehicle software, only libraries necessary to connect to AWS IoT Core are implemented. The certificate is pre-installed in the vehicle during the End of the Line process.
2. The connection is made to AWS IoT Core through a private Access Point Name (APN) provided by the MNO and utilizing the customer's own AWS IoT Core endpoint. All traffic is sent over MQTT protocol secured using mTLS.
3. Upon connecting to AWS IoT Core with the private certificate, the Lambda validates the Gateway and creates the IoT Thing and IoT Policy. Each vehicle ECU should have a unique certificate and potentially a unique IoT policy associated with it allowing only what is needed for the ECU to communicate to AWS.
4. Associate the Telematics Control Unit (TCU) with the Vehicle Identification Number (VIN). The vehicle is registered. The VIN is obtained from the telemetry data (see #1).
5. Use the Mobile Network Operator (MNO) API to register the Subscriber Identity Module (SIM).
6. Vehicle Management application allows the connected vehicle operations center to manage any discrepancy or out of band process during the vehicle registration.
7. Encryption at rest on the server-side is available in all the services with encryption keys managed in AWS Key Management Service (AWS KMS).



User provisioning reference architecture

Figure 2: CM-S01-b: User Provisioning reference architecture

8. User Provisioning can be performed either through self service by vehicle owner using an application, assisted through intermediaries (for example, Dealers, and Contact Center) or options through the vehicle.
9. Encryption at rest on the server-side is available in all the services with encryption keys managed in AWS Key Management Service (AWS KMS).

10A. Assign Driver to Vehicle and manage one or manage preferences per vehicle.

10B. Entitlement management helps ensure that user permissions are assigned based on user subscription and entitlements.

10C. Consent management is used to document, audit, and manage users consent to terms and conditions.

CM-S02 Vehicle connectivity management

Vehicle connectivity management helps ensure resilient, secure, bidirectional connection between the vehicle and the cloud. It enables high throughput data transfer, low latency events and

supports communication across all devices, in both low latency local area and wide area network connection. The following user stories are supported in Vehicle Connectivity Management.

User stories

CM-S02-UC01 Connectivity: The ability of the vehicle to connect and exchange data with its surroundings, such as city infrastructure, the driver, passengers, pedestrians, and other vehicles, is commonly referred as V2X. The ability of the vehicle to connect and exchange data with the cloud is referred to as V2C (Vehicle to Cloud) or C2V (Cloud to Vehicle). Connectivity to these external systems should be secure and data should be relevant to satisfy real time vehicle functions.

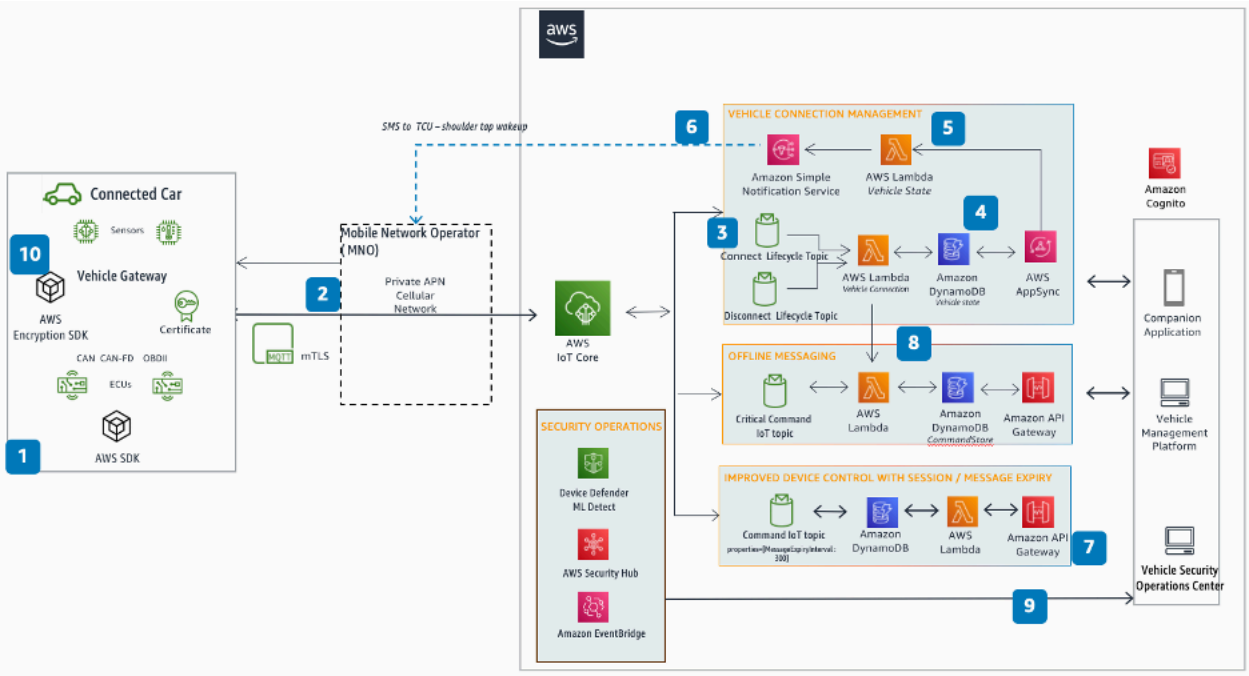
To achieve these objectives, vehicle connectivity management should be able to provide the following:

- Be able to withstand connection drop and re-establish connection.
- Provide connectivity state and vehicle state in the cloud.
- Keep the vehicle network connected for extended periods while the vehicle is switched off.
- Ability for the connected vehicle to connect to the cloud Region that is in its geo-proximity and if it's not available, then it can failover to the alternate Region selected by the customer.
- Perform secure authentication and authorization.
- Ability to failover to a different network provider in case the Mobile Network Operator (MNO) service degrades.
- Ability to receive and act upon predictive Quality of Service (QoS) notifications from the MNO.
- Ability to run commands with low latency to ultra-low latency. Low latency is typically latency within 2–3 secs round trip and ultra-low latency is typically less than 200ms round trip. Remote commands and tele-operations are example use cases for running with low latency and ultra-low latency respectively.

CM-S02-UC02 Messaging: Vehicle Connectivity Management should be able to deliver messages to an individual vehicle or to a fleet of vehicles. To ensure relevant messages are sent, there should be capability to control message expiration and set message priority. It becomes essential to provide message delivery functionality that ensures to deliver message at-least once and provide traceability of the message generation. In certain instances, such as running remote commands, there should be the ability to wake the vehicle. Such capability should balance between low battery usage and latency in responding to the commands. Some common mechanisms include extended discontinuous reception (eDRX) failing over to shoulder tap in the case where the vehicle

goes into deep sleep. The Vehicle Connectivity Management should also have the ability to serialize and deserialize messages over the air.

Reference architecture



Vehicle connectivity management reference architecture

Figure 3: CM-S02 Vehicle connectivity management reference architecture

1. The connected vehicle acts as an IoT device, with a unique identity principal (X.509 certificate). AWS IoT Core is used for communication from edge-to-cloud to collect, analyze, and act upon the sensor data gathered from the vehicle.

2. The connection is made to AWS IoT Core through a private cellular network using the customer's own AWS IoT Core endpoint. All traffic is sent over the MQTT protocol secured using mTLS.
3. Upon connection, AWS IoT Core publishes the vehicle's connected state to the Connect Topic. This reserved topic is where connection events are published automatically upon connection.
4. The vehicle connection state is stored in the database for implementing observability solutions.
5. Messages published from the cloud use the Vehicle State AWS Lambda function to check connected state.
6. If the device is in a disconnected state, the vehicle state Lambda function invokes Amazon SNS, which will send an SMS to the dialable MSISDN on the SIM on the TCU to indicate that a command is waiting and to wake up and subscribe to command topics.

7. Messages are published based on vehicle connectivity state. Messages can also be expired by the MQTT Broker if it is not delivered on time. If the vehicle is connected, the command payload is published using the Command MQTT topic in AWS IoT Core that the vehicle has subscribed.
8. Critical system messages sent from the cloud to the vehicle can be stored, processed, and delivered when the vehicle comes back online.
9. AWS IoT Device Defender sends device audit and violation findings (such as unusual device behavior or software and configuration vulnerabilities) to AWS Security Hub CSPM, where security findings from other AWS services and AWS Partner products are aggregated and normalized. Security Hub CSPM sends findings to EventBridge, which routes them to a remediation workflow implemented in a vehicle security operations center.
10. The vehicle can use the AWS Encryption SDK using keys in AWS KMS for client-side encryption. The ECU can get temporary API credentials from the IoT credential provider to call AWS KMS APIs. You can also implement your own key management system for encryption keys.

CM-S03 Vehicle data management and insights

Connected vehicles have hundreds of controllers and sensors producing thousands of individual data elements for operating, and conveying the state of a vehicle. Vehicle data management helps vehicle manufacturers to harness data as an asset, to drive sustained innovation and create actionable insights and improve their customer experience. Vehicle manufacturers are seeking cost-effective ways to simplify the process of collecting data from vehicles that are connected to the cloud help power insights and improve vehicle performance while maintaining the highest levels of confidentiality and security. There are regulations (such as CCPA and GDPR) related to data privacy and data sharing that require customers to provide data lineage and data governance capabilities.

User stories

CM-S03-UC01 Telemetry: Automakers collect a variety and large volume of data that is configurable based on rules and filters to support business needs, such as predictive maintenance, location-based services, and insurance claims. In the next generation of vehicles, with expansive amounts of data continuously ingested at high rates from the vehicle, there are multiple verticals that would want to access the vehicle's data, including insurance, municipalities, and content providers. With automakers monetizing the vehicle data, managing the balance between cost, customer experience, data privacy, and revenue realized important.

When determining the backend architecture for the vehicle's payload, there are a few key concepts to keep in mind, namely size and frequency. Automakers could operate a hybrid approach to

ingestion, with support for both high-frequency as well as low-frequency data ingestion. This hybrid approach allows automakers to optimize their cost while maximizing the data monetization potential for the most valuable data attributes from the vehicle.

The telemetry collection system should be able to withstand a sudden connection loss and maintain data integrity. The majority of telemetry data is time series data. To maximize the value of this data, processing and visualization of this data is a key capability needed to help create valuable insights and build new solutions to support those insights. Time series data differs from traditional vehicle sensor data in that it's used to perform queries in time windows across differing time frames.

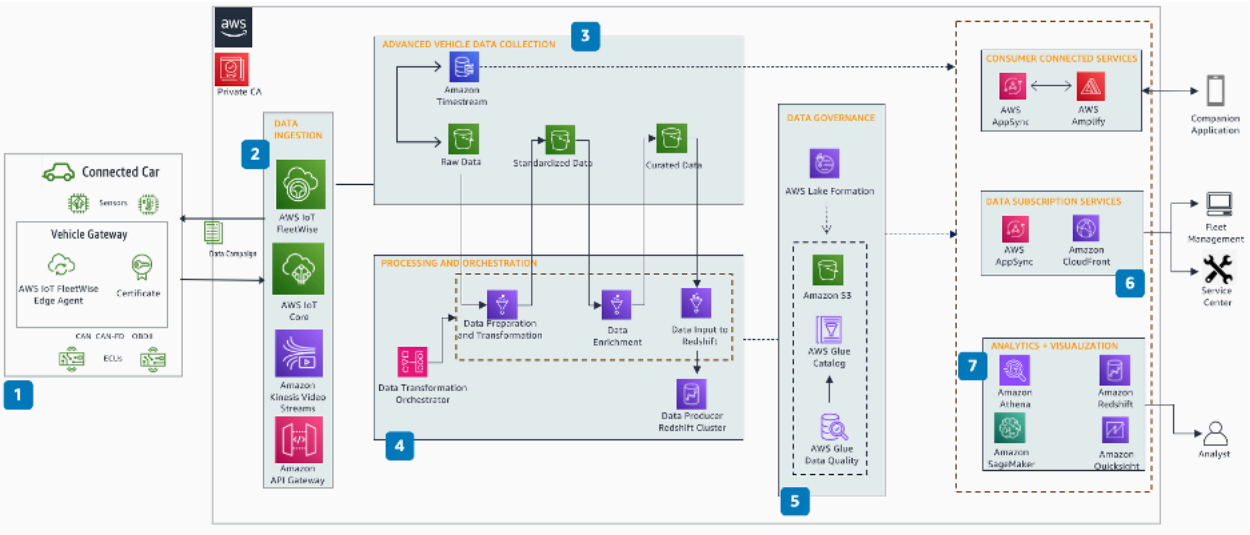
CM-S03-UC02 Data management and governance: Vehicle manufacturers and vehicle owners need to establish data governance standards for collection, storing and dissemination of data and adhere to local rules and regulations (such as CCPA and GDPR). Vehicle owners and manufacturers need a consent management system to control, share, and delete data that supports meeting local, national, and international data privacy regulations and protects privacy of consumers.

CM-S03-UC03 Data sharing and neutral servers: Neutral servers are servers operated and financed by operators not connected with vehicle manufacturers. Such servers can be used by vehicle manufacturers to make vehicle data readily discoverable and accessible to interested vendors with necessary security and privacy controls.

CM-S03-UC04 Insights and generating value from connected vehicle data: Vehicle manufacturers would like to generate value for their customers from the connected vehicle data acquired throughout the lifetime of the vehicle, for example:

- Provide Usage Based Insurance (UBI) dependent upon driver behavior, distance driven, and so on.
- Provide subscription-based value-added service, such as Feature on Demand through over-the-air (OTA) update.
- Increasing advertising reach and targeted advertising.
- Independent service stations would like to access vehicle data and fault codes to repair and maintain vehicles.
- Smart maintenance scheduling to reduce maintenance cost, increase fleet availability, and deliver a differentiating customer service.
- Drive vehicle optimization, efficiency, and early detection of deviations from normal operating condition using ML algorithms and advanced simulations on a digital twin of the vehicle in the cloud.

Reference architecture



Vehicle data management and insights reference architecture

Figure 4: CM-03: Vehicle data management and insights reference architecture

1. The connected vehicle, with a unique identity principal (X.509 certificate), has hundreds of sensors to collect data. AWS IoT FleetWise Edge Agent collects, stores, and organizes data from vehicle. Based on the campaign defined in AWS IoT FleetWise, the agent decodes signals from the vehicle and sends data payloads through AWS IoT Core.
2. The vehicle can communicate using OEM chosen protocols, such as MQTT and HTTPs, and use AWS services, such as AWS IoT Core and Amazon API Gateway. Vehicles can send video streams to the cloud using Amazon Kinesis Video Streams.
3. Improve data relevance by creating time- and event-based data collection campaigns that send the exact data you need to Amazon Timestream or Amazon S3.
4. By using purpose-built data processing components, vehicle manufacturers can generate data ready for consumption, organized by subject areas, segments, and profiles.
5. AWS Lake Formation makes it easier to centrally govern, secure, and globally share data. A data lake can be used to store and analyze multiple data types from wide variety of sources. Use AWS Glue Data Quality to measure and monitor the data quality and take corrective actions.
6. Enable different user personas from fleet aggregators to data scientists, analysts, and vehicle owners.
7. You can use Amazon SageMaker AI improve ADAS/AV models to optimize vehicle design for performance and efficiency. Insights from structured and semistructured data can be gathered

by using Amazon Redshift. Utilizing Quick and other analytics platforms to continually improve vehicle quality, safety, and autonomy using near real time data from AWS IoT FleetWise.

CM-S04 Connected mobility core services

Vehicle manufacturers can deliver value-added services to fleet operators and vehicle operators that helps them improve the vehicle operating experience, such as remote lock or unlock, remote vehicle monitoring, usage-based insurance, and improve experience throughout the vehicle lifecycle.

User stories

CM-S04-UC01 Companion mobile app for vehicle operators: Vehicle manufacturers can provide a companion mobile app to vehicle operators to remotely interact with the vehicles and view on demand vehicle information. Some examples are:

- Send remote commands to their vehicle to lock/unlock, start/stop their vehicle.
- Send destination and waypoints to the in-vehicle navigation system.
- Get EV charging status.
- Get diagnostic information, such as tire pressure, battery charge or fuel status, and oil life.
- Autonomous capabilities such as summoning the vehicle or parking.

CM-S04-UC02 Predictive service: Analyze component usage and provide predictive maintenance guidelines to help uptime and manage the service experience.

CM-S04-UC03 Driver and passenger safety: Connected vehicle platforms are designed to provide seamless communication between vehicles and remote applications to receive alerts of hazardous situations, enable more time to react, help prevent accidents and automate emergency call in the event of an accident. Automakers should implement redundancies in handling safety use cases to avoid any single point of failures.

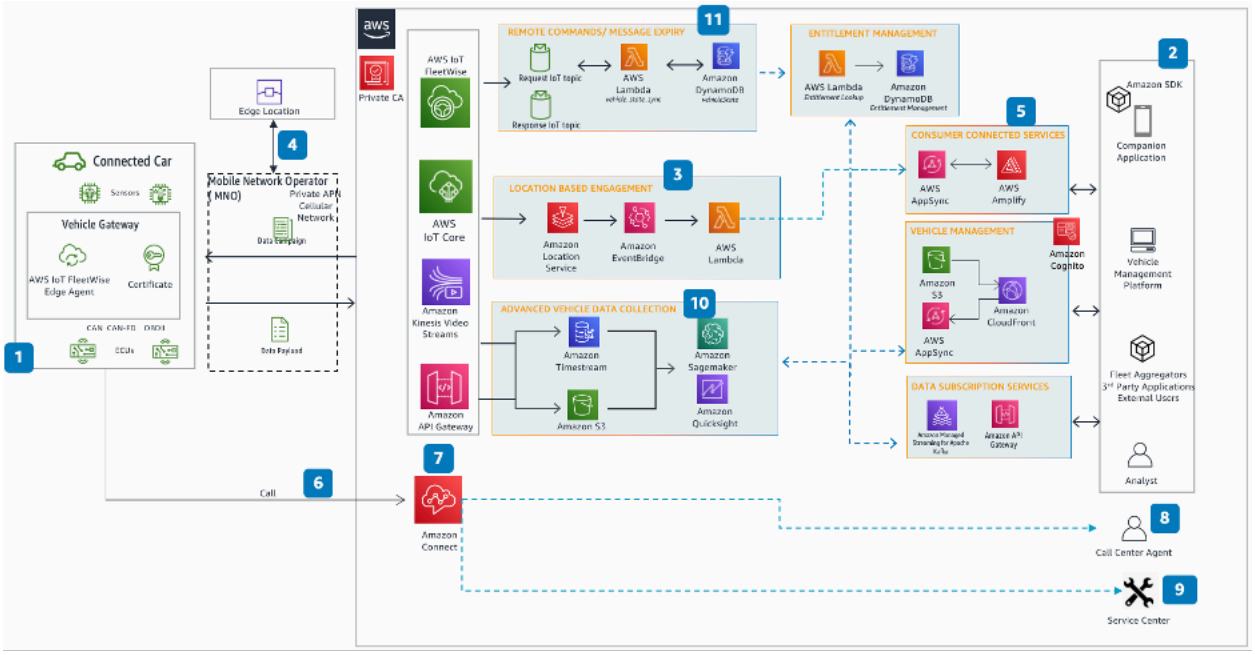
CM-S04-UC04 Vehicle security: With connected vehicles capability to remotely operate the vehicle and the amount of personal data collected and stored in the cloud, it becomes essential to provide guardrails to prevent bad actors from using these new attack surfaces. Authorities are rolling out cybersecurity regulations like WP.29 Cybersecurity Vehicle Regulation Compliance regulation and related automotive standards such as ISO/SAE 21434 standard to mitigate the cybersecurity risks posed to vehicles.

CM-S04-UC05 Diagnostics: Send real-time and scheduled diagnostic reports to the customer regarding the vehicle health like oil life, tire pressure, battery health, and fuel or charge status. Call centers and repair centers can also automatically find the reason for the vehicle breakdown.

CM-S04-UC06 Location-based services:

- Fleet operators and vehicle operators receive breakdown assistance by easily calling local roadside assistance in case of a vehicle breakdown. Mobile repair centers have the ability to get the accurate location of the vehicle to ensure efficient service to vehicle operators.
- Capability to set up geofencing to ensure vehicle safety by preventing unauthorized vehicle use.
- Location specific offers from merchants.

Reference architecture



Connected mobility core services reference architecture

Figure 5: CM-S04 Connected mobility core services reference architecture

1. The connected vehicle, with a unique identity principal (X.509 certificate), has hundreds of sensors to collect data. AWS IoT FleetWise Edge Agent collects, stores, and organizes data from vehicle. Based on the campaign defined in AWS IoT FleetWise, the agent decodes signals from the vehicle and sends data payloads through AWS IoT Core.

2. Adding location awareness to apps enables an enhanced experience, such as sending real-time messages, and information and service based on user location (for example, repair centers have the ability to get the accurate location of the vehicle to ensure efficient service to vehicle operators).
3. Amazon Location Service features, such as maps, trackers, and geofence collections, are used to send geolocation data to track and follow the vehicle location. Events are initiated on Amazon EventBridge when the vehicle enters or exits a geofence and notifies vehicle and fleet operators.
4. Applications that are latency sensitive, such as tele-operations or Cellular V2X applications, can be deployed at the edge. AWS Wavelength could be used to deploy such applications.
5. Applications developed with AWS Amplify and AWS AppSync are used by vehicle operators and owners to create messages and business rules for geofences and notify events to the users.
6. A vehicle can initiate a call to the emergency response center, such as when it detects a hard impact or airbag deployment. Emergency services can be dispatched even if the driver is unresponsive.
7. Amazon Connect starts the automatic contact flow for the call and based on the nature of the call can be attended by the call center agent, road side assistance, or virtual voice assistant.
8. Critical details, such as the speed at impact, the number of airbags deployed, vehicle operation status, and even video footage, can be transmitted to the operator.
9. The solution also integrates with roadside service assistance providers and shares any relevant data for assistance.
10. Collect real time telemetry, analyze performance and component usage metrics to provide prescriptive guidance to vehicle owners.
11. The Request/Response messaging pattern in AWS IoT Core is a method to track responses to client requests in an asynchronous way. For vehicle remote commands, this enables the publisher to specify a topic for the response to be sent for a particular message, ensuring proper messaging to the customer on success or failure of the command state. Using the Message Expiry feature of AWS IoT Core, the vehicle operator could specify how long to attempt the remote command before expiring the message.

CM-S05 Connected mobility supported systems

Connected mobility is a foundational component in enabling systems such as autonomous driving, battery health management, and fleet management. These downstream systems have stringent Quality of Service (QoS) requirements related to performance and resiliency, which the connected

mobility platform has to satisfy. Thus, vehicle manufacturers have to give careful consideration while designing and developing such platform. Following are some examples of supported systems enabled by connected mobility platform.

User stories

CM-S05-UC01 Autonomous driving: Vehicle manufacturers can use sensors to gather and generate data about condition of the vehicle and surrounding environment. Vehicle manufacturers can use the connectivity management platform developed by them to share such data among the fleet of connected vehicles and send data to other systems.

CM-S05-UC02 Fleet management: Fleet operators aim to reduce fleet operation cost, improve operational efficiency such as optimizing fleet routes, fleet utilization, driver support and reduce fleet disruption using centralized monitoring of vehicles and real time operational insights. To help achieve these objectives, fleet operators desire flexible and extensible data ingestion pipeline to capture near real time and historical data from the fleet. Further, they also desire tailored insights to visualize the health and status of the fleet and rely on connected mobility platform to achieve the same.

CM-S05-UC03 Battery health management: Fleet operators and vehicle owners aim to improve the efficiency and safety of their fleet operations by monitoring the battery health, improve battery performance, predict battery issues and extend lifespan of the battery. To achieve these objectives, they must collect and transfer Battery Management System (BMS) parameters to the cloud and rely on data gathered through connected mobility platform.

CM-S05-UC04 Charging station ecosystem: Vehicle manufacturers and charging station operators rely on Vehicle Connectivity and Data Management capabilities to provide value added solutions such as vehicle charging status, direction to nearest charging station, and remotely monitor charging station.

CM-S05-UC05 Navigation, traffic, and travel management: Vehicle drivers aim to improve mobility by reducing travel times, make informed decision regarding routes and modes of transportation and receiving real time events and hazards around them. Data gathered through connected mobility platform enables these functionalities.

CM-S05-UC06 Tele-operation: Vehicle connectivity management functionalities are utilized by fleet operators to remotely monitor vehicles using video feed from onboard cameras and in special circumstances control the vehicles.

CM-S06 Customer experience management

Insights from vehicle data help provide a personalized in-cabin experience, which can also be portable to other vehicles. Vehicle Manufacturers can also use such data to predict customer's needs at different touch points throughout the vehicle lifecycle and provide a more seamless experience to address those needs.

User stories

CM-S06-UC01 Driver experience: By monitoring inside and outside the vehicle, vehicle manufacturers can recommend changes to improve their customer’s driving experience and safety. Vehicle manufacturers can also provide customized infotainment options based on operator’s previous behavior. Allow the driver to take their profile to other vehicles they own or rent, subject to their approval and affirmative consent.

CM-S06-UC02 Contact center: Automakers need omni-channel customer service infrastructure that streamlines the customer support process and makes it simple for agents to resolve customer inquiries faster.

CM-S06-UC03 Emergency response: Vehicle drivers would like to automatically receive help from emergency services should the vehicle experience an accident.

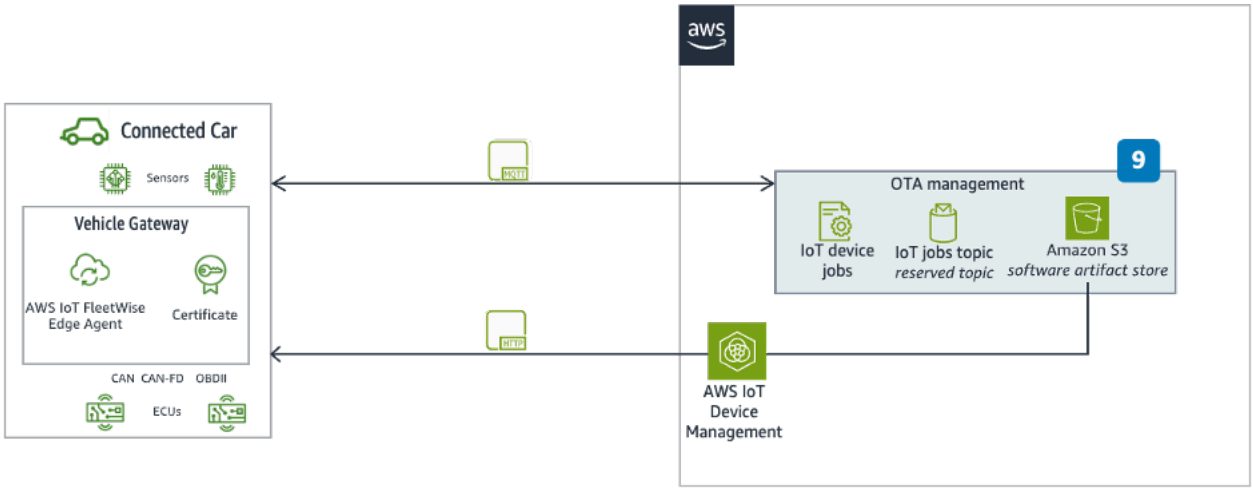
CM-S06-UC04 Retention, renewal, and churn: Predict the customer propensity to renew their subscription and create a personalized marketing campaign.

CM-S06-UC05 Service experience: Improve vehicle service experience based on data-driven analytics and machine learning to help OEMs build innovative applications. For example, if the connected vehicle data shows a trend of extreme tread wear, then the system can recommend a tire replacement on the infotainment screen, provide a capability to shop for new tires and schedule a service appointment with a few clicks.

Reference architecture

Customer experience management reference architecture

Figure 6: CM-S06-a Customer experience management reference architecture



Customer experience management reference architecture

Figure 7: CM-S06-b: Customer experience management reference architecture

1. Gather customer interaction and sentiment from source systems such as clickstreams, call center logs, vehicle sensor data, social sentiment etc.
2. Ingesting data across customer touchpoints into marketing data lake using variety of protocols.

3. Provide rich experience in the car, lifelike conversational services for vehicle breakdown, emergencies, vehicle-related questions, notifications, and concierge services. Vehicle manufacturers can use speech analytics powered by machine learning to access live call transcripts, understand customer sentiment, and identify call reasons in near-real time.
4. Transform raw data into consumption ready data using purpose-built data processing components and transformation libraries.
5. Analytics layer natively integrated with consumption ready data for quality and trend analysis. Customer sentiments can be analyzed using Contact Lens for Amazon Connect. Amazon Comprehend can also be used to perform post call analysis to gain further insights about the customer call.
6. For convenience and concierge services, use Amazon Alexa features. For example, an electric vehicle driver can use voice commands to learn the best route that will provide minimum charging time while enabling the driver to enjoy their favorite food while they wait.
7. Activate multiple customer channels such as mobile push, voice, and email for targeted marketing communications.
8. The companion application provides real time vehicle information, personalization and push notifications.
9. Use AWS IoT Device Management to implement OTA management through AWS IoT jobs data and use AWS IoT Fleet Indexing to manage state, connectivity, and device violations and to organize, investigate, and troubleshoot your fleet of devices.

The Pillars of the Well-Architected Framework

This section describes each of the Well-Architected pillars and includes definitions, best practices, questions, considerations, and essential AWS services that are relevant for customers designing, building and managing connected mobility workloads.

Pillars

Operational excellence pillar

The operational excellence pillar includes operational practices and procedures used to manage connected mobility workloads. Operational excellence comprises how planned changes are performed, as well as responses to unexpected operational events. Change execution and responses should be automated. All processes and procedures of operational excellence must be documented, tested, and regularly reviewed. By adopting the practices in this paper, you can build architectures that provide insight to the connected mobility workload status, perform effective and efficient operation and event response, and can continue to improve and support your business goals.

Design principles

The following are design principles for operational excellence in the cloud:

- Based on the workload and your business objectives, identify connected mobility key performance indicators (KPIs). The connected mobility KPIs should be linked to metrics from all layers: business, application, data, security, and infrastructure. Once identified, use the connected mobility KPIs to work backwards and identify operational KPIs.

- Implement end-to-end observability and a single pane view of the connected mobility workload status from vehicle to the cloud and full stack of the application: Proactively monitor end user experience in-vehicle.
- Develop a testing framework to simulate production-like conditions: Site Reliability Engineering (SRE) team are responsible for reliability of the application amidst frequent updates from development teams. The SRE team works closely with the development team to create new features and stabilize production systems. One of the challenges faced by the SRE team is to simulate the vehicle while trying to test or debug a connected mobility feature or a production issue. Develop a virtual vehicle in the cloud that has capability to simulate various real-world conditions including load and Mobile Network Operator (MNO) failures.
- Implement mechanisms to improve developer productivity enabling reduction in Time to Market (TTM), which is the time it takes to take connected mobility features from concept to launching it in the industry. Develop deployment templates and provide a developer portal to quickly spin up new development and lab environments.
- Implement a multi-stakeholder runbook to effectively respond to production events. Connected mobility is a complex system with multiple stakeholders from end users to Mobile Network Operators, cloud hyperscalers, downstream data consumers, and OEMs. To reduce the Mean Time to Restore (MTTR), a runbook should be developed for all critical events. Eventually, automate runbooks to streamline operational processes, reduce human error, and improve efficiency, leading to faster incident resolution and enhanced operational excellence.

Definitions

Observability: Observability allows users to understand a system's state from its external output and take (corrective) action. Observable systems yield meaningful, actionable data to their operators, allowing them to achieve favorable outcomes (faster incident response, increased developer productivity) and less toil and downtime.

Best practices implementing observability include:

- **Monitor what matters:** Start with the connected mobility business key performance indicators (KPIs) and work backwards to the application and infrastructure metrics.
- **Collect telemetry from all layers:** Connected mobility has dependencies and interactions with vehicles, Mobile Network Operators, cloud providers, internet service providers, AWS Partners, and other components—both within and outside your control—that can impact your business outcomes. It is important that you have a holistic view of your entire workload.

- **Propagate context:** Collect logs, metrics, and traces, and propagate transaction IDs, which helps to perform correlation, analysis, anomaly detection, dashboarding, and alarms.

Leading versus lagging indicators: Leading indicators are metrics that are used to measure future performance. For example, customer satisfaction and connected mobility feature usage metrics can be used to predict renewal rates, as a happy and engaged customer is more likely to renew the paid subscription. Similarly, quality metrics of the feature releases can be a leading indicator to predict the failure rate of the system.

Lagging indicators are metrics used to measure past performance for example connected mobility subscription renewal rates, Mean Time Between Failures (MTBF), and remote command latency. Lagging indicators provide valuable feedback on the effectiveness of past decisions and help identify areas for improvement. Both leading and lagging indicators are important for managing and measuring operational efficiency of connected mobility workloads.

Best practices

Following are the operational excellence best practice areas:

- Organization: Understand your organization's priorities, your organizational structure, and how your organization supports your team members, so that they can support your business outcomes.
- Prepare: To prepare for operational excellence, you have to understand your workloads and their expected behaviors. You will then be able to design them to provide insight to their status and build the procedures to support them.
- Operate: By understanding the health of your workload and operations, you can identify when organizational and business outcomes may become at risk, or are at risk, and respond appropriately.
- Evolve: Evolution is the continual cycle of improvement over time. Implement frequent small incremental changes based on the lessons learned from your operations activities and evaluate their success at bringing about improvement.

In addition to what is covered by the Well-Architected Framework related to process, runbooks, and game days, there are specific areas you should review to drive operational excellence within your connected mobility platform.

Organization

There are no best practices specific to this area for connected mobility.

Prepare

CMOPS_1: How do you define the meaningful monitoring KPIs and metrics of your connected mobility platform?

Connected mobility platforms handle millions of vehicles and end users, and billions of messages being generated. The scale and distributed nature of connected mobility platforms create unique challenges for monitoring. Defining the critical KPIs and metrics is a key step of your monitoring strategy, and should include end user mobile applications, vehicles, mobile cellular networks, and cloud infrastructure. Consider starting with the benefits that the connected mobility platform provides to its end users and work backwards to identify KPIs and metrics, which are leading and lagging indicators of success.

[CMOPS_BP1.1] Define end-to-end KPIs and metrics for a connected mobility platform.

Collecting key metrics, logs, and trace information from all parts of your connected mobility platform will give you end to end visibility across the solution. It can decrease the Mean Time to Detect (MTTD), Mean Time to Repair (MTTR), and Mean Time to Restore Service (MTRS) by allowing you to detect issues quickly, and troubleshoot and debug with precision. It can also help you recognizing trends before critical issues occur.

Define KPIs that include business, application, and infrastructure across the solution. Start this process by creating a working team that includes stakeholders from various teams including business teams. Start by defining business KPIs, and then supporting KPIs for the application and infrastructure. These KPIs should be mapped to the metrics collected from tools such as AWS CloudWatch, which helps to observe resources in AWS, on-premises, and in other clouds. Some sample KPIs are as follows:

Business KPIs:

- Metrics from end user mobile app including the number of remote commands issued, errors running remote commands, remote command latency, errors from user portals, and errors from infotainment systems.

- Mobile network Quality of Service (QoS) metrics such as packet loss, latency, jitter, and interference.
- Vehicle health indicators such as diagnostic codes, battery health, fuel efficiency, and repair and maintenance data.
- Vehicle security metrics such as the number of unauthorized requests to high-risk ECUs and vehicle disconnected duration metric.

Application KPIs:

- Number of transaction failures, application errors, and number of retries for key services.
- Response times to and from vehicle, and latency across the application and in critical functions.
- Health checks for critical features and functions of the application.
- Database metrics such as query response times, number of connections, and IOPS metrics.

Infrastructure KPIs:

- Usage metrics that include CPU, Memory, Storage, and Network for critical connected mobility infrastructure. These metrics play a crucial role in proactive infrastructure management, optimizing resource allocation, and enhancing the overall user experience of Connected Mobility.
- Service quota metrics to enable proactive service quota management and avoid downtime due to reaching a service quota. Effective quota management prevents connected mobility service disruption.
- Security metrics including unpatched instances, security vulnerabilities, security events, and non-compliant resources. These metrics help assessing security status, make informed decisions, and prioritize actions to mitigate risks.
- Cloud specific metrics for key services. As an example, AWS Lambda is commonly used as part of connected mobility implementations on AWS. Key Lambda metrics include number of invocations, number of errors, number of retries, number of invocations that were throttled, and duration of function processing. These metrics play a vital role in preventing and addressing various issues that can affect the reliability and performance of your connected mobility platform.

CMOPS_2: How do you observe the health of your connected mobility platform and proactively identify anomalies?

The connected mobility platform includes the vehicle edge, connectivity, cloud-based infrastructure and applications, enables various services like fleet management, remote diagnostics, and predictive maintenance. Implementing observability enables insights about end-to-end system health, availability, performance, and scalability, which in turn helps reduces the time to restore the service after a disruption.

By monitoring metrics such as response times, resource utilization, and error rates, the customer can identify potential bottlenecks or issues that may impact the overall performance of the system. This allows for proactive measures to be taken, such as scaling up resources or optimizing the platform's architecture, to provide uninterrupted service delivery.

Monitoring the health of edge devices is crucial because these devices form the backbone of connected mobility. Edge devices, such as vehicle telematics units or sensors, collect and transmit data to the cloud for analysis and are typically used for decision-making. By continuously monitoring their health, you can detect any anomalies, performance degradation, or malfunctions in real-time and provide timely interventions.

Observability of connected mobility platforms requires that the data from all the subsystems and microservices are aggregated in a data lake, there is a capability to correlate the records, and dive deep to identify the root cause of the issue.

[CMOPS_BP2.1] Implement an observability data lake that aggregates telemetry from all connected mobility components.

Validate that all components of the connected mobility platform are able to send the telemetry data (logs, traces, and metrics) to the observability data lake. Implement transaction correlation IDs in those records to trace the transaction across multiple services.

To investigate intermittent errors in a set of vehicles, it requires a capability to activate debug level logging inside the vehicle. These logs can help the operations team to replay transactions in a test environment and also debug the root cause of the errors in production. As the data loggers will generate a lot of data, this capability should be enabled only during an event. Implement a log and trace framework, based on the standard defined by the AUTOSAR 4 DLT. The end user also

should be able to enable this through a diagnostic app in the infotainment screen to assist the troubleshooting with the service engineer.

The logs can then be transferred to the cloud using AWS IoT Core where the AWS IoT rules engine can be used to upload in-vehicle log records to Amazon CloudWatch. You can also upload the log records using Amazon S3 pre-signed URLs if the MQTT publish payload is more than 256 MB. The connected mobility platform should have a capability to import these logs into vehicle simulators in the cloud to replay these transactions in non-production environments

Implement an observability data pipeline to orchestrate and automate data processing workflows, ensuring seamless ingestion and transformation of observability data.

[CMOPS_BP2.2] Set up real-time monitoring and alerting capabilities to detect anomalies promptly.

Use Amazon CloudWatch to store logs and to monitor key metrics such as vehicle status, telemetry, or network connectivity. Use AWS Glue to transform and analyze the log data using services like Amazon Athena or Amazon OpenSearch Service. Implement custom log analysis solutions or use AWS Partner Network offerings for advanced log analytics. When predefined thresholds or anomalies are detected, CloudWatch Alarms can cause notifications via Amazon Simple Notification Service (Amazon SNS) or perform automated actions using AWS Lambda functions.

Implement AWS Lambda functions to process and analyze streaming data from the edge devices in near real-time, triggering alerts or notifications based on predefined thresholds. By leveraging Lambda, you can apply custom logic or machine learning models to the incoming observability data, enabling real-time anomaly detection and triggering alerts or notifications based on specific conditions.

Perform synthetic transaction monitoring that simulates the typical end user journey. The synthetic transaction agents should be deployed in various geographical locations, which periodically sends test transactions to the connected mobility services deployed in the cloud. Implement a vehicle health monitoring agent solution as per AUTOSAR to monitor the vehicle to cloud communication. If the monitoring detects any issues, such as a breach in a performance threshold or any error, it will alert the operations team.

[CMOPS_BP2.3] Implement predictive analytics and proactive operations.

To improve operations and detection of patterns in the cloud, predictive analytics and proactive operations play a significant role in avoiding costly downtimes. By using predictive analytics, historical data is analyzed to identify patterns and trends that can predict potential component

failures. This enables proactive maintenance scheduling, reducing the risk of unexpected downtime. Machine learning (ML) models can be trained on historical data, allowing for accurate predictions. Applying predictive analytics algorithms helps generate insights and optimize maintenance planning.

For the application stack in the cloud, use an ML service like Amazon DevOps Guru to detect abnormal operating patterns to identify operational issues. You can extend predictive capability by developing ML algorithms using a service like Amazon SageMaker AI and leveraging an observability data lake. This insight can be used by the solution to automatically recommend remedial actions and runbooks to restore the service

In a connected mobility platform, the most challenging aspect is debugging vehicle to cloud communications. Following best practices can help you debug the in-vehicle connected vehicle components remotely.

[CMOPS_BP2.4] Implement robust remote diagnostic capabilities.

To troubleshoot, robust remote diagnostic capabilities are essential. This entails developing comprehensive diagnostic algorithms that can accurately identify potential issues. Real-time data collected from in-vehicle system is continuously analyzed to detect anomalies and raise timely alerts. By leveraging advanced ML techniques and cloud-based analytics platforms, the diagnostic accuracy can be improved, enabling efficient identification of problems.

Use services such as AWS IoT Core for near real-time diagnostic data ingestion and processing. Use serverless functions such as AWS Lambda to develop diagnostic algorithms that can analyze the events in real-time. This enables the diagnosis of complex patterns and the generation of actionable insights.

[CMOPS_BP2.5] Provide remote access for troubleshooting purposes.

Authorized support personnel can remotely access in-vehicle systems, which expedites the troubleshooting process. However, it's crucial to implement secure communication protocols to protect data privacy and prevent unauthorized access. Utilizing end user consent and virtual private networks (VPNs) or encrypted remote access tools helps establish a secure connection. Regularly auditing and updating access permissions further enhances security while preventing unauthorized access. Follow the security best practices (CMSEC_6) for the remote troubleshooting.

CMOPS_3: Have you set up a validation environment that has feature equality with production environment of the connected mobility platform?

[CMOPS_BP3.1] Validate the system in a non-production environment that has feature equality with production.

The connected mobility platform requires additional validation to incorporate variability of network connectivity, external systems, in-vehicle components, model years of vehicles and diverse environments that the vehicles are operated.

Use digital twin concepts to create a vehicle simulator to introduce common failure scenarios and test subsystem behavior. Modularizing the digital twin model enables additional subsystems to be added in the future. Leverage vehicle observability data collected over time to generate machine learning models and perform predictive testing. Create curated test dataset to test application logic, regression, and performance. Use simulation tools (such as virtual ECUs) to test for scenarios throughout the application/sub-system development lifecycle and to validate subsystem behavior.

Perform real world tests for connectivity (with different hardware and network equipment/environment), security, navigation, entertainment, data collection and analytics. Such tests should be performed with hardware-in-loop using modular test benches.

CMOPS_4: How do you do agile development of the connected mobility platform, with a focus on minimizing disruptions?

[CMOPS_BP4.1] Leverage microservices architecture for connected mobility platform.

The microservices architecture streamlines the development, deployment, scaling, and maintenance of the entire connected mobility platform. For example, the remote commands microservice, which is responsible for sending lock, unlock, and remote start commands to the vehicle, can be independently scaled and deployed. This operation can be performed separately from the vehicle diagnostic microservices, which send periodic vehicle diagnostic reports to the end user. This architecture also enables each microservice to have different disaster recovery (DR) strategies based on their individual recovery time objective (RTO) and recovery point objective (RPO).

[CMOPS_BP4.2] Implement API-first architecture to facilitate the exchange of data and services while developing connected mobility platform.

API operations help in agile development and faster feature-to-market in the context of connected mobility software, it provides clear guidelines, enable modular development, and support collaboration with external stakeholders. For example, automotive Original Equipment

Manufacturers (OEMs) deploy developer portals to crowdsource app development on the connected mobility platform. These portals give secure access to the connected vehicle platform through public API operations. Connected vehicle API operations could also be used to securely share connected vehicle data to authorized repairers for compliance with Right to Repair regulations. The API-first approach also helps to easily extend the platform to support multiple model years of vehicles with backward compatibility as longs the API contracts are maintained. It also reduces the impact of changes and reduces time to market for new features without causing disruption to the connected mobility platform.

[CMOPS_BP4.3] Implement DevOps automation.

Incorporate DevOps automation to improve productivity, enhance developer experience, and improve quality of the connected mobility feature releases. Implement a self-service developer portal which provisions pattern-based software templates with preapproved resources and configurations requiring limited manual intervention. Create connected mobility platform templates in the developer portal that include opinionated pre-built modules for observability and other connected mobility best practices baked in. Adopt Continuous Integration/Continuous Deployment (CI/CD) pipelines to automate the testing and deployment of the software updates, ensuring rapid and error-free releases. Automate testing, to simulate scenarios and real-world conditions, helps validate the connected mobility platform for safety and performance.

Operate

CMOPS_5: How do you respond to an incident within your connected mobility platform?

Responding to disruptions is an important aspect of operating your connected mobility platform as it impacts end user's ability to access critical features of their vehicle, may impact company revenue, and could erode customer trust. Understanding the scope of an outage, including which functions are affected, allows for timely and accurate communication with your customers and key business stakeholders, and minimizing your Mean Time to Repair (MTTR), and Mean Time to Restore Service (MTRS).

[CMOPS_BP5.1] Determine the scope and impact of the incident to your connected mobility platform.

The connected mobility platform includes many functions, and understanding which functions are affected, and the scale of impact is an important step to guide your response. An example of a

critical function is the ability for customers to use remote commands. Start by analyzing your end-to-end monitoring dashboards for your connected mobility platform, review reported incidents to your Service Desk, and determine the cause of the disruption. If it is related to an AWS service event, find out if it is impacting a Region, or localized impact to a specific Availability Zone. You can use the Health Dashboard to get more detailed information on AWS service specific events.

Reference your incident management playbook to investigate the appropriate process to initiate. The runbook should guide your teams on communicating the disruption to your customers, include step by step instructions on how to investigate the incident, remediation options, and how to validate recovery.

[CMOPS_BP5.2] Communicate with customers about the incident in a timely fashion.

Connected mobility disruptions require effective communication for the safety of the users and brand protection. The notification should be event driven and multi-channel, example notify affected customers immediately through the customer's mobile app via push notifications, SMS messages, and email about the issue. Provide clear and accurate information about the problem, its cause, and the expected resolution time. Establish a dedicated line for inbound customer queries. Communicate on the regular basis on progress.

[CMOPS_BP5.3] Recover the application using runbooks and automation.

After identifying the issue, refer to the designated runbook for application recovery. Following the runbook instructions, you might need to shift to an alternate Availability Zone (AZ) if an AZ failed, or move to another Region in the case of a Region-wide failure. For example, if a crucial business component, with low RTO and RPO requirements, experiences a disruption, it is essential to refer to the disaster recovery runbook to facilitate the transition of your application.

Evolve

There are no best practices specific to this area for connected mobility.

Key AWS services

- AWS Lambda
- AWS IoT Core
- AWS Glue
- Amazon DevOps Guru
- Amazon SageMaker AI

- Amazon S3
- Amazon Athena
- Amazon OpenSearch Service
- Amazon Simple Notification Service (Amazon SNS)
- Amazon CloudWatch
- AWS Systems Manager

Resources

Documentation and blogs:

- How to get started with the new disconnected duration metric in AWS IoT Device Defender
- Building an Automotive Embedded Linux Image for Edge and Cloud Using Arm-based Graviton Instances, Yocto Project, and SOAFEE
- Using Digital Twins to Drive Electric Vehicle Battery Insights with MHP and AWS
- Build an observability solution using managed AWS services and the OpenTelemetry standard
- Monitoring your IoT fleet using CloudWatch
- Monitoring AWS IoT
- What is Site Reliability Engineering (SRE)?
- What is observability? (AWS Observability Best Practices)
- AUTOSAR_SWS_DiagnosticLogAndTrace (PDF)
- AUTOSAR_EXP_SystemHealthMonitoring (PDF)
- Diagnostic Log and Trace daemon (GitHub)

Security pillar

This section describes security best practices when building connected mobility applications on AWS. The document will cover the well-known security pillars from the Security Pillar of the AWS Well-Architected Framework, with a focus on Connected Mobility. The security pillar includes the ability to protect your connected vehicle services and systems while delivering business value through risk assessments and mitigation. Connected vehicle environments have unique security considerations around how identities are provisioned to vehicles and operators, protecting vehicle and consumer applications, protecting connected mobility data, providing detection and incident

response of vehicle and systems related to vehicles (such as companion applications, and charging stations), and application security best practices. This document covers each of those areas in detail, and provides security best practices and guidance.

Design principles

There are a number of principles that can help strengthen the security of your connected mobility solution in addition to the overall Well-Architected Framework security design principles:

Identify automotive and general regulatory and compliance standards: Design and architect solutions to help adhere to industry-standard security frameworks and regulations. The automotive industry has specific regulations and standards to consider. Automotive specific standards and frameworks that should be considered by you and your applicable stakeholders include, and are not limited to: ISO/SAE 21434 for cybersecurity, ISO 26262 for functional safety, ASPICE for software development lifecycle process, ISO 24089 for OTA, and regulations like UNR 155/156 for developing a cyber security management system and software update management system in automotive. It is important to consider and align to general standards when building connected mobility applications like ISO 27001, NIST Cybersecurity Framework, and privacy laws like GDPR. This helps to verify that the systems in scope are validated against the highest levels of safety, security, and privacy.

You have a choice of protocols that can be used for vehicle connectivity to the cloud: There are several protocols that can be used to for vehicle connectivity to the cloud. The protocols and services that you use can influence the security properties of the connection between vehicle to cloud. Common protocol patterns include HTTPS, MQTT/TLS, Kafka over TLS, Secure RTP, and gRPC over TLS. Consider cost, performance, bandwidth, business applicability, and more when deciding which protocol fits your connectivity use-case.

Implement secure identity lifecycle management for vehicles and consumers: An end-to-end identity management system should manage the entire lifecycle for vehicles, owners, drivers, operators and partner identities. There are unique considerations for electronic control units (ECUs) device certificates such as provisioning, expiry, rotation/revocation, validation, and certificate management. Consumer identities need to be considered as they interact with different vehicles and vehicle features. It is important to verify the validity and mapping between the different identity types (owner, driver, and vehicle) and pro-actively revoke or extend identities and the associated permissions over time.

Establish least privilege permissions for vehicles and systems: Vehicles should have fine-grained access permissions to backend systems and access to vehicles from other systems should

follow the same guiding principle of least privilege. Vehicles receive communication from many endpoints such as charging stations, backend systems, and consumer applications. Scoping down the permissions will help reduce the risk of lateral movements from a compromised entity (vehicle or system). Vehicle ECUs should only be able to communicate with topics or endpoints that they are required to operate. Permissions of vehicle ECUs should be audited and reviewed continuously.

Classify and protect data collected from systems and applications related to connected mobility: Connected mobility applications interact with sensitive data including but not limited to personal identifiable information (PII), location data, VINs, and user data. Customers need to implement data protection mechanisms such as data encryption, data anonymization, and data access control to safeguard the sensitive data. Use secure communication channels such TLS to confirm that data transmitted between vehicles and the cloud is protected from interception, tampering, or other types of attacks. Vehicles, sensors, and systems should only collect the data necessary to facilitate their function.

Design for continuous monitoring and verify trusted software and firmware: implement continuous monitoring mechanisms of your connected mobility service such as in-vehicle and the vehicle environment (such as charging stations, and mobile applications) to detect and respond to security incidents and alert your purpose-built vehicle security operations center (VSOC). Use secure firmware and software update mechanisms to create timely security patches and updates over the air. Implement secure boot and chain of trust mechanisms to verify that only trusted software components are loaded and executed on the vehicle's hardware.

Design for resiliency in-vehicle and in the cloud: implement redundancy and resilient mechanisms to check that critical systems such as brakes, steering, and power train continue to operate even in the event of a security issue.

Confirm that backend systems in the cloud are also resilient and able to recover from such events. Consider implementing managed services and highly available backend systems by leveraging multiple Availability Zones and plan the disaster recovery for your solution in another AWS Region. Backup your critical data frequently to help meet the desired Recovery Point Objective (RPO) of the solution. Implement vehicle shadow logic to maintain state during loss of connectivity to the cloud.

Define a vehicle and system threat model: implement threat modeling to identify potential security risks and vulnerabilities in the system (vehicle, vehicle related infrastructure, applications, backend) and implement appropriate security controls to mitigate these risks. Confirm that you follow automotive threat modeling standards such as Threat Analysis and Risk Assessment (TARA) or another methodology to cover in-vehicle threats and measure appropriate risk ratings.

Train your staff so that they are able to address vehicle security risks and the impact across the organizations: Deliver a training program that can provide your security team and developers with vehicle cybersecurity and compliance training, cloud security training, and fundamental information technology training. It is important to evaluate program effectiveness and update periodically.

Best practices

Topics

Security foundations

CMSEC_1: How are you maintaining consumer privacy during actions such as in-vehicle purchase transactions or data collection?

Vehicles today contain a multitude of sensors which enable many of the features we rely on for both convenience and safety. These sensors collect large amounts of data which must be both transmitted, stored, and potentially correlated between distributed systems which provide this functionality. When building these architectures, it is important to understand all applicable laws and compliance requirements which govern what data is necessary to collect, store or share with third parties, and how it should be anonymized to protect the privacy of consumers and their passengers. The system should be designed with these requirements in early stages and not after the system is in production.

[CMSEC_BP1.1] Personally identifiable information (PII) and unique identifiers (IDs) assigned to a vehicle or consumer should be anonymized.

Information which can be used to directly or indirectly identify an individual must be protected from disclosure. This information includes but is not limited to name, address, location data, or combination of other data elements such as Vehicle Identification Number (VIN) and International Mobile Equipment Identity (IMEI). This data should be encrypted in transit using secure protocols such as TLS and at rest using your own robust encryption keys or those provided by AWS Key Management Service. Restrict access to data for authorized users by classifying data and creating separate views of this data according to the principles of least privilege using AWS Identity and Access Management or masking sensitive data using third party solutions such as *DataMasque*.

[CMSEC_BP1.2] Location data collected from navigation systems should be appropriately secured to protect anonymity.

Data which can be used to locate a consumer or track travel routes should also be encrypted in transit and at rest. Location data which is sent to third parties should have any PII or IDs scrubbed from the request. In addition, the precision of coordinates for both latitude and longitude can be adjusted to provide additional privacy. As an example, *Amazon Location Service* supports up to 6 decimal points for these coordinates which is accurate up to approximately 4 inches from the object of interest. As the number of decimal points is decreased so does the precision and depending on the requirements for the service being provided this may be an acceptable threshold.

[CMSEC_BP1.3] Data collected from cameras, microphones, biometric, and other types of sensors should be appropriately secured.

Sensors should only collect the minimum amount of data required to facilitate their function. When possible, sensors such as microphones should leverage a wake word or activation of a tactile button to begin the data collection process. These sensors should also contain idle timeout logic to limit this collection. Still or video image captures should have identifying features such as the faces of passengers and pedestrians for example blurred using a proven process. The AWS blog post Creating a serverless face blurring service for photos in Amazon S3 outlines how this can be accomplished using Amazon S3. Data should use strong encryption in transit and at rest to verify privacy when transmitting to OEM or third-party systems. Other types of sensors such as those that collect biometric data may be subject to other compliance standards such as the Health Insurance Portability and Accountability Act (HIPAA) and require different configurations for data handling and retention.

CMSEC_2: Your risk management program should align to security standards, laws, regulations, and frameworks in the connected vehicle space.

Your risk management program should be used to capture functional safety, cybersecurity, privacy, and secure software development requirements throughout the lifecycle of the vehicle. This can be accomplished by incorporating guidance from automotive industry specific frameworks such as ISO-21434, information security standards like NIST 800-53 or ISO 27001, and the new UNR-155 and UNR-156 regulations concerning type approval with regards to cyber security management systems and software update management systems. These standards, frameworks, and regulations can inform your organization on how to design a cyber security program that covers both the vehicle and the systems and resources that interact with vehicles. This requires input and collaboration from a number of cross-functional areas including but not limited to management, security, and legal to address the needs that are specific to your organization.

Identity and access management

CMSEC_3: How do you manage the identities of your vehicles and individual ECUs?

When working with connected vehicles, you must determine how you will manage the identities of your vehicles and individual Electronic Control Units (ECUs). These identities can come in different forms and are used to authenticate and authorize access to a variety of services in the connected vehicle environment. This includes other vehicles, charging stations, backend services, OTA servers, and more. For example, an ECU can be uniquely identified with a private key (usually stored in a secure hardware module such as a TPM or an HSM) and an X.509 certificate corresponding to that private key. These X.509 certificates may have subject names or extensions with vehicle and ECU information such as VIN or ECU ID.

There are important considerations when provisioning, rotating, revoking, and managing the lifecycle of certificates on ECUs. Each ECU can have several X.509 certificates, each used for different purposes such as an attestation certificate (or birth certificate) used to authenticate an ECU, and one or more operational certificates. These operational certificates are used to authenticate to connected vehicle services. For example, when the ECU connects to a connected vehicle service using Mutual TLS, it can present its client certificate and cryptographically prove that it possesses the private key. The TLS server can validate the certificate to authenticate that the connection originates from the ECU that owns the private key.

In addition to X.509 certificates, vehicles and ECUs may use other forms of identities such as tokens, API credentials, and HTTP cookies. For example, a user signs in to a streaming music application on the in-vehicle infotainment (IVI) system and after the user's authorization, the music

application receives an OIDC Access Token that is used to authenticate requests to the backend APIs of the streaming music service. In another example, an ECU may exchange its operational certificate for a set of time-limited cloud API credentials used to invoke cloud service APIs such as uploading video or telemetry data.

The following best practices can help you manage identities for your vehicles and ECUs.

[CMSEC_BP3.1] Securely manage the lifecycle of identities and credentials for your vehicles and ECUs

It is important to check that each request or action on a connected vehicle service is authenticated and authorized. The security depends on appropriately managing the lifecycle stages of the identities and credentials for your vehicles and ECUs:

- **Provision**

  This is the process of installing identities into your vehicles and ECUs. See CM-S01 Vehicle and User Provisioning from scenarios section. Typically, this begins at the manufacturing process where a unique Attestation X.509 certificate (also called a Birth Certificate) is issued for the ECU to prove its authenticity and provenance. The private key for the certificate can be generated within an on-board secure hardware module such as a TPM or HSM, a Certificate Sign Request (CSR) is generated and presented to a Certificate Authority (CA) that issues a certificate. The CA can be part of a Private Key Infrastructure (PKI) owned and operated by the OEM or the supplier of the ECU. For more information see this blog post that discusses provisioning certificates using AWS IoT. Additionally, the ECU can have one or more operational certificates that are used to authenticate to connected vehicle services. It is important to verify that the process of requesting and issuing certificates is authenticated, authorized, and audited to check that only genuine ECUs are issued certificates. If you are unable to generate private keys in an on-board secure module and must generate private keys on a server, see that access to these private keys is restricted and tracked as the security of client certificate-based authentication relies on protecting the private keys. You have the option to design, build and operate your own PKI and CAs, or you can take advantage of AWS Private Certificate Authority (AWS PCA) that allows you to create private certificate authority (CA) hierarchies, including root and subordinate CAs, without the investment and maintenance costs of operating an on-premises CA.
- **Enroll or register**

  After certificates are issued and installed in ECUs, you may also need to enroll or register the certificates with your backend services. For example, you can register the ECU's certificate in an asset database associated with the ECU ID (serial number) and corresponding authorization

policies that are applied to requests from that ECU. You can build your own asset database using a self-hosted database or an AWS managed cloud database service. If you are using AWS IoT Core to connect your ECUs to the cloud, IoT Core provides a built-in registry for Things and their registered X.509 certificates. Each certificate registered in IoT Core can be associated with an IoT Core Policy (either directly or via Thing Groups) that authorizes actions that the ECU can perform on the IoT Core service such as allowing connections, publishing or subscribing to certain MQTT topics. IoT Core also allows you to register certificates the first time an ECU connects using Just-in-time-Registration by registering the Certificate Authority (CA) that issued the certificate.

- **Validate**

  Client certificates presented by ECUs are validated by the connected vehicle service as part of the mutual TLS handshake. This process includes checking that the certificate has not expired, validating the certificate chain to verify the certificate is issued by a trusted Certificate Authority, checking for certificate revocation (see below), and checking the certificate against an asset database to check that it is an authentic and active ECU. AWS IoT verifies that a client certificate is active when it authenticates a connection. You can create and register client certificates without activating them so they can't be used until you want to use them. You can also deactivate active client certificates to disable them temporarily.

- **Renew**

  Each X.509 certificate has a validity period (determined by the Not Before and Not After fields) that is checked by servers before accepting the certificate as valid. One strategy to avoid renewing is to issue certificates with a long validity period - such as 100 years - that is beyond the expected service lifetime of the ECU using the certificate. This strategy then relies entirely on the revocation process (see below) to manage the validity of certificates. A second strategy is to issue certificates with a medium validity period - typically 1 to 3 years. A third strategy is to issue short-lived certificates - typically a few hours or days - especially for cases where it is important to prevent tracking and attribution that can happen with static long-lived certificates. In the latter two strategies, it is important to have a secure process for renewing certificates **before** they expire. This process must be secured to the same degree as the process used to originally issue and provision certificates. On AWS, AWS IoT Device Defender periodically audits client certificates registered in IoT Core and publishes findings for certificates that are expiring within the next 30 days. You can send the findings to a remediation workflow in AWS Step Functions, using Lambda functions and IoT Jobs to trigger the ECU to generate a new CSR, issue a new certificate, and send the new certificate to the ECU. Once the ECU has successfully installed and tested the new operational certificate, the workflow can revoke the old expiring certificate in IoT Core.

- **Revoke**

  When you detect a security issue with a certificate (such as the private key is compromised) or are notified that the ECU has been replaced, you may choose to permanently revoke the certificates associated with the ECU. Certificates are revoked by invoking the CA that issued the certificate. The CA publishes the revocation status of certificates via Certificate Revocation Lists (CRLs) or via Online Certificate Status Protocol (OCSP) or both. Connected vehicle services that validate client certificates must check whether the certificate is revoked using the CA's supported method before accepting the certificate as valid. AWS Private CA supports both CRL and OCSP as revocation methods, and this blog post helps you choose the best method for your use-cases. You can also revoke a certificate that was previously registered in IoT Core and this denies connections that use that certificate.

- **Retire**

  Finally, when the ECU has reached its end of life or has been replaced and you no longer want the certificates to be trusted, you can choose revoke the certificate (see above) and delete the certificate from your asset database. This will help validate that no future connections or requests from the ECU are accepted by connected vehicle services. IoT Core allows you to delete a Thing and certificates.

[CMSEC_BP3.2] Design and securely operate a PKI for vehicle identities

The previous section described how vehicle identities are often X.509 certificates issued by a vehicle Private Key Infrastructure (PKI). The security of a scheme that relies on X.509 certificates for authentication and authorization depends on the security of the PKI that issues the certificates. It is important to start by documenting your policies and practices for operating your PKI including details of your Certificate Authority (CA) structure, policies for the validity period of your CAs, the process for succession of CAs, description of the administrative permissions and controls, roles and responsibilities. You can capture this information in two documents, known as Certification Policy (CP) and Certification Practices Statement (CPS). Refer to RFC 3647 for a framework for capturing important information about your PKI operations.

Minimize the use of the root CA to only issue certificates for intermediate CAs. Store the private key of the root CA in a secure Hardware Security Module (HSM) and tightly control physical and logical access.

In some circumstances, as an OEM you may need to share parts of your PKI with suppliers and vendors. example, you may create an intermediate CA for a supplier and allow that supplier to

issue attestation and operational certificates that are embedded in ECUs during manufacturing. In this case, you are relying on the policies and processes of your supplier for the security of the ECU's identities. Review the supplier's Certification Policy (CP) and Certification Practices Statement (CPS) to check that these meet your overall PKI requirements.

You can use AWS Private CA to create, manage, and operate a vehicle PKI on AWS. PCA allows you to create a CA structure with multiple levels and gives you fine-grained control over each CA and whether and how it is shared with other entities such as suppliers.

[CMSEC_BP3.3] Design a mechanism to tie various vehicle identities together as necessary

As described in CMSEC_BP3.2, a vehicle will have several identities, multiple identities per ECU and multiple identities for the vehicle. Based on your application needs, you may require the ability to tie these identities together so that you can map an authenticated ECU identity to its ECU ID or serial number, and to the vehicle's VIN where the ECU is installed. You can maintain an asset database that stores a record of the different identities and their relationships. Because this database stores sensitive information that can be used to identify, relate, and track identities, the database must be secured and all accesses to the database must be authenticated, authorized, and audited. To protect the privacy of your vehicles, owners and operators, check that only authorized staff have access to this database and unauthorized or unusual query patterns are detected and alerts are sent to the data owners. The asset database must be updated with changes such as when certificates are renewed (see above) and ECUs are replaced in repair shops. Verify that only authorized staff are allowed to update the database and changes are tracked and associated with authorized work orders or repair bills of material.

[CMSEC_BP3.4] Define a mechanism for how your vehicles will be identified towards external entities

You may have use-cases where you need to send information about your vehicles and ECUs to external entities such as business partners (dealers, suppliers, repair shops, and vehicle insurance providers or example). Consider designing a mechanism that maps stable unique identifiers, such as VIN and ECU Serial Number (ESN), to anonymized or temporary identities so that you can preserve the privacy of your vehicles, owners, and operators and prevent unauthorized or unintended sharing of information with your partners. You may choose to create a mapping database that stores the mapping from stable identifiers to the anonymized or temporary identifiers sent to your partners. Validate that such a mapping database has strong authentication, access control, audit and logging.

On AWS, you can choose from a number of fully managed purpose-built database services to store mappings.

CMSEC_4: How do you manage the identities of the owners, drivers, and operators of your vehicles and map them to the vehicle's identities?

Connected vehicle services often need information on the identities of users: owners, drivers, and operators of the vehicles to provide service features corresponding to the user. For example, a vehicle is being driven by a teenaged family member that is not the vehicle owner. The vehicle IVI invokes a personalization service to get the preferences of both the owner and the family member to apply personalized preferences and restrictions. The teenaged driver is restricted to a top speed limit configured by the vehicle owner but is able to see their favorite music applications on the IVI home screen.

To implement such features, the vehicle ECUs must recognize and identify the current driver, and the connected vehicle service must authenticate and authorize requests based on the identities of the current driver and owner and other contextual information such as the time of the day.

Consider the following best practices when managing the identities of owners, drivers and operators of your vehicles.

[CMSEC_BP4.1] Design an authentication mechanism for users of your vehicles

Design an authentication mechanism for users of your vehicles based on the sensitivity and risk of the actions being performed. For example, unlocking a vehicle is a high-risk action and should require the use of a physical key fob or a digital key from an application on a phone. Setting IVI home screen preferences may be a low-risk action and may be acceptable to use factors such as facial recognition of the driver. This allows physical key fobs to be shared between family members but still recognizes the current driver of the vehicle. The driver of a delivery vehicle that is part of a fleet is authenticated using an application on a hand-held scanner device. The fleet operator uses the identity to track the driver of each vehicle and the routes taken for safety, efficiency, and audit purposes.

In contrast to vehicle ECUs that are typically authenticated and authorized using X.509 certificates (see CMSEC3_BP5 regarding designing a PKI for vehicle identities), users are usually authenticated using tokens. Consider using an identity provider that supports widely adopted identity standards such as OpenID Connect (built on OAuth 2.0) to authenticate your users and generate tokens.

OpenID Connect identity providers issue JSON Web Tokens (JWT) that are cryptographically signed by the identity provider and delivered to the in-vehicle ECUs that make requests to connected vehicle services on behalf of the user. The ECUs provide the Java Web Tokens (JWTs) as part of the request for example in the HTTPS request header and the tokens can be validated on the connected vehicle service to authenticate the request, and the request is authorized based on claims and scopes in the token. Because tokens are used directly to authenticate requests to services, check that tokens are stored securely on ECUs and are not exposed to unauthorized users. Consider the token lifecycle, using short-lived tokens to limit the impact of an exposed token, use refresh tokens to obtain new tokens without requiring user interaction, and revoke the token when you detect that a token is compromised.

Because the in-vehicle user interfaces such as the IVI may be limited in terms of user input capabilities - lacking a physical keyboard for example asking the user to type a long, complex password on a touch-screen may be impractical and error prone. In such cases, consider implementing the OAuth 2.0 Device Authorization grant (also called Device Flow) that allows devices that have no built-in browser or limited input capability to get an access token by using a browser on a different device such as a mobile phone or desktop. The user interacts with the browser on the phone or desktop to authenticate to the identity provider and grant access to the IVI.

You can build and operate your own identity provider or use a cloud SaaS identity provider. *Amazon Cognito* is a fully managed identity provider that allows you to add user sign-up and sign-in features and control access to your web and mobile applications. Amazon Cognito user pools provides an identity store that scales to millions of users, supports social and enterprise identity federation, and offers advanced security features to protect your consumers and business. You can implement Device Authorization grant using Cognito and AWS Lambda. AWS services such as Application Load Balancer (ALB), API Gateway, AppSync, and IoT Core can authenticate and authorize requests using tokens issued by Cognito User Pools.

[CMSEC_BP4.2] Consider patterns for mapping a user's identity to the vehicle's identities

The connected vehicle service will often need to map between the user's identity and vehicle's identities to provide service features. For example, a personalization service needs to know the identity of the current driver, the owner, and of the vehicle's ECUs to respond with the corresponding policies and preferences.

One approach is to combine two authentication schemes:

- **Vehicle ECUs:** Use X.509 client certificates to authenticate vehicle ECUs when they connect to a connected vehicle service using mTLS, and
- **Users:** Send JWT access or ID tokens in the request (over mTLS) as HTTPS headers or the MQTT payload to authenticate users to a connected vehicle service.

This approach inherently maps the user's identity to the vehicle ECU's identity and your backend service can validate each identity cryptographically. The backend service can use the ECU's identity to query an asset database for information such as the owner or operator of the vehicle.

Consider how mapping information is updated over the lifecycle of the vehicle. For example, information about the owner of a vehicle must be updated in the asset database when a vehicle is sold.

Validate that access to the mapping between users and vehicles is stored securely and access to this mapping is restricted to applications and administrators on a need-to-know basis.

CMSEC_5: How do you manage the identities of your business partners, such as dealers, suppliers, service providers, or other third parties?

Connected vehicle services operate in a rich environment with multiple business partners such as dealers, suppliers, repair shops, and service providers, to name only a few. In addition to managing the identities of your vehicles and users it is equally important to manage the identities of your business partners that interact with connected vehicle services. For example, a dealer for an OEM uses a connected vehicle service to set the owner for a vehicle when a vehicle is sold. As this operation has security and safety implications, the dealer must be authenticated and authorized to perform this operation, and all actions must be logged and audited.

Consider the following best practices to manage the identities of your business partners.

[CMSEC_BP5.1] Separate identities of your external partners from internal employees and contractors

It is very common for a single connected vehicle service to have multiple types of users that interact with it: vehicle drivers, owners, business partners, employees, and contractors. Consider creating strict segregation boundaries between the identities of your employees and contractors on one side, and business partners on the other.

You can implement this separate by creating separate identity provider pools (also called domains, directories, or tenants - the terminology depends on the identity provider software or service you are using) for the two types of users. This approach has the following benefits:

- Each identity provider pool can have different security policies such as whether user sign-ups are allowed, password strengths enforced, and the use of multi-factor authentication (MFA).
- You can configure federation for the employee identity pool only to your corporate identity provider, allowing your employees to single-sign on to your custom applications.
- User tokens contain a unique issuer element and are signed by a key that is unique to the identity provider pool. This provides a cryptographic way to validate that a particular identity originates from a particular pool.

On AWS, you can separate different types of users into separate Cognito User Pools and configure each pool with different properties and policies.

[CMSEC_BP5.2] Verify that application authorization is based on the type of identity

The simplest way to implement this is to segregate the identities into separate identity provider pools and have each connected vehicle service API explicitly trust each identity provider pool to authenticate users from that pool. For example, your dealer-facing APIs trust the dealer identity pool, but your employee facing APIs trust only the employee identity pool. A token issued to a dealer from the dealer identity pool cannot be used to authenticate requests to the employee APIs, because the token validation fails.

A single API resource that trusts both the employee and business partner identity pools should be considered an exception. The configuration and code of such an API must be closely reviewed to check that authorization is appropriately granted to each type of user.

As you design your application authorization scheme, verify that only information contained in a cryptographically verified token (such as a JWT) or information queried from a trusted backend database is used for the authorization decision. Do not rely on information for authorization that can be spoofed such as query parameters, the request body, or unsigned browser cookies that cannot be cryptographically verified by the server for authenticity and integrity.

On AWS, you services such as Application Load Balancer (ALB), API Gateway, AppSync, and IoT Core can be configured to authenticate tokens issued by specific Cognito User Pools. With API Gateway, you can also implement custom authorization logic using *Lambda authorizers*.

CMSEC_6: How do you manage access permissions to vehicle functions?

In the context of connected vehicle services, privileged actions are those that are initiated from the backend and can impact the safety and security of the vehicle. For example, an owner speaks to a customer support agent in a call center and requests a remote unlock of their vehicle because they lost their physical key fob. In another example, the driver of a vehicle authorizes the OEM's support engineers to remotely diagnose and fix issues with their vehicle. As is evident in both these examples, the actions initiated from connected vehicle services backends can directly impact the safety, security, and operation of the vehicle. Permissions to perform these actions must be carefully managed and audited.

The first step is to identify the use-cases that involve restricted or administrative actions and focus on implementing the following best practices to manage accesses to those actions:

[CMSEC_BP6.1] Define a process for support staff to gain permissions to perform administrative actions on vehicle functions

The process that your support staff (such as engineers, and customer support agents) follow to gain permissions to perform administrative actions must require appropriate approvals and traceability to the context of the request, for example, the customer support case ID or trouble ticket that corresponds to the access request. Access to human actors must be granted for a limited time and for specific actions on specific ECUs: such as grant the ability to remotely collect logs and diagnostic data from a specific vehicle's battery management ECU for the next 30 minutes. Access must be granted only after obtaining required approvals from managers and after passing validation rules: such as verify that the ECU ID in the support case is the same as the ECU ID for which access is being requested, and the support engineer is indeed on call for the current shift, and they belong to the appropriate security groups in your directory. For highly sensitive and impactful actions, consider implementing quorum approval processes (also called M of N approvals) where multiple managers or peers must review and approve an access request before access permissions are granted. It is recommended that this process is automated and based on requestor entitlements to avoid the risk of collusion.

[CMSEC_BP6.2] Implement fine-grained audit logging of every privileged action on vehicles

Validate that every privileged action on vehicles, whether initiated by a human actor or by an automated system, is logged with details such as the identity of the human or system actor, the timestamp, the action performed, the target of the action (for example, ECU ID, and Vehicle ID)

and the result of the action. Verify also that each step in the process to gain access is also logged (request, review, approval, grant), so it can be audited and tracked. Design a mechanism that protects the integrity of access audit logs by denying permissions to modify, delete or overwrite log events, and by appending cryptographic signatures to log events that can be used to verify their integrity. You must also check that the log events are indexed and searchable for later audit and analysis.

A centralized logging solution is needed that aggregates and indexes such audit log events from your fleet of compute servers. You can deploy and manage a log collection and indexing solution or you can use a managed SaaS service. Amazon CloudWatch Logs enables you to centralize the logs from all of your systems, applications, and AWS services that you use, in a single, highly scalable service. You can then easily view them, search them for specific error codes or patterns, filter them based on specific fields, or archive them securely for future analysis. CloudWatch Logs also supports querying your logs with a powerful query language, auditing and masking sensitive data in logs, and generating metrics from logs using filters or an embedded log format. (See CMSEC_BP11.1)

[CMSEC_BP6.3] Alert on unauthorized attempts to access privileged actions on high-risk ECUs

In addition to generating fine-grained audit log events for every privileged action (see above), consider creating automated alert notifications when unauthorized attempts to access privileged actions on high-risk ECUs such as the IVI or Head Unit. For example, you can create a rule that triggers an alert notification when a single support staff performs privileged actions on multiple ECUs in a short period of time. This can trigger your Vehicle Security Operations Center (VSOC) to investigate the alert to determine if this is a case of compromised credentials or identity being used in an unauthorized manner. (See CMSEC_BP22.1)

CMSEC_7: How do you verify that vehicles are granted least privilege access to perform actions on your backend systems?

As your vehicle ECUs make requests to connected vehicle services, you must check that each ECU is authorized to perform the smallest set of actions that it needs to fulfill its function. For example, you may want to authorize an ECU to publish telemetry data to a MQTT topic named with its ECU ID, and to subscribe to real-time road condition updates from another topic if the vehicle owner has subscribed to a premium service.

Consider the following best practices to grant least privilege authorization to ECUs:

[CMSEC_BP7.1] Authorize every action taken by a vehicle ECU individually.

Validate that every action taken by a vehicle's ECU on your connected vehicle services is authorized individually. You can use the identity of the ECU you defined in CMSEC_3 to define the baseline policies that grant specific actions to the ECU. You can enhance these policies to grant or deny other actions based on dynamic context conditions such as the time of the day, location of the vehicle, and state of subscription services.

You may consider externalizing the authorization policy evaluation logic into a shared, centralized policy evaluation service where policies can be written in a domain specific language (DSL) such as Cedar, that is easier to author, understand and audit than programming language code. Amazon Verified Permissions is a scalable permissions management and fine-grained authorization service for the applications that you build. Amazon Verified Permissions uses the Cedar open-source policy language This pattern splits the responsibilities for authorization: the connected vehicle service is responsible for gathering the identity and context information and providing it to the policy evaluation service. The policy evaluation service implements the logic to parse the policy DLS and respond with a allow or deny decision to the connected vehicle service. This pattern also splits the responsibilities of the team that authors policies in the DLS from the developers that write the code for your connected vehicle service, allowing each team to be more efficient and deploy changes at their own pace.

You can authorize requests on the connected vehicle services backends based on scopes in access tokens, or fields from X.509 certificates. Several AWS services offer built-in authorization features allowing you to define policies that control access to the service's data plane. AWS IoT Core allows you to define IoT Core Policies that control the actions a device connected to IoT Core can perform. You can attach policies to each client X.509 certificate and to Thing Groups to control access at multiple levels. Amazon API Gateway provides multiple ways to control access to a REST API. Specifically, the API Gateway Lambda authorizer allows you to implement your custom business logic in Lambda function code and return a policy that API Gateway will cache and use to authorize future requests. Amazon Verified Permissions is a scalable, fine-grained permissions management and authorization service for custom applications. The service centralizes fine-grained permissions for custom applications and helps developers authorize user actions within applications. Amazon Verified Permissions uses the *Cedar* policy language to define fine-grained permissions for application users.

CMSEC_8: How do you validate that your backend systems have least privilege access to perform actions on vehicle functions?

As you design connected vehicle service backend systems, validate that each system is only granted the smallest set of permissions to perform actions on vehicles. This is especially important for CMSEC_7 regarding privileged access, as this is critical for backend systems that initiate privileged actions on vehicles because it can impact the safety, security, and operations of the vehicle.

Consider the following best practices:

[CMSEC_BP8.1] Verify your backend systems are authenticated and authorized for every action on vehicle functions.

Implement secure authentication and authorization of every request from a backend system that results in actions on vehicles. For example, a customer support uses a customer support application to initiate remote door unlock on behalf of the vehicle owner who has lost their keys. In this scenario, check that the customer support agent is authenticated and authorized to perform the action, and verify that the support application is authenticated and authorized to perform the unlock action on a connected vehicle service.

In another example, your software administrator has scheduled the roll-out of an Over the Air software update to a fleet of vehicles. An OTA service executes this update by sending commands to each vehicle in the fleet to download the update files securely from an update service. The OTA service must be authenticated and authorized before it is allowed to such a command to a vehicle ECU.

In both of the previous examples, one service interacts with another service to perform an action on a vehicle. This is a type of machine-to-machine communication where the calling service must be authenticated and authorized by the called service. You can choose several approaches to authenticate machine-to-machine application scenarios.

On AWS, you can choose AWS Signature v4 (sigv4) that is supported by all AWS API endpoints, and can be used to authenticate and authorize requests to your custom APIs created in API Gateway. You can use sigv4 with AWS IoT Core to authenticate and authorize actions such as publish and subscribe to MQTT topics using MQTT over WebSocket. If the backend service is deployed to AWS, you can associate an IAM Role with the AWS compute service (for example, Amazon EC2, Lambda functions, Amazon ECS, Amazon EKS) that provides temporary security credentials that can be used by the backend service to sign sigv4 requests. If the backend service is deployed outside AWS, consider using IAM Roles Anywhere as the mechanism to obtain temporary security credentials in exchange for X.509 certificates.

[CMSEC_BP8.2] Implement fine-grained audit logging of actions performed by your backends on vehicle functions

It is important to implement fine-grained audit logging of actions performed by your backend services on vehicle functions. Having such an audit log will help you with operational troubleshooting and as security incident response investigations.

You can implement audit logging in your connected vehicle service by emitting log events when actions are invoked. You can centrally aggregate and store these log events in a Security Incident and Event Management (SIEM) system that indexes the events and allows you to search and create dashboards and visualizations of these events.

On AWS, consider enabling the built-in logging features of AWS services that you use to implement your backend services. Enable AWS IoT logging to deliver log entries to *CloudWatch Logs* for events on the IoT service such as a connect, publish, subscribe, and many others. You can enable CloudWatch logs for your custom APIs in API Gateway to get both execution logging and access logging events delivered to CloudWatch Logs. Consider ingesting these log events from CloudWatch Logs into your SIEM tool to be indexed and analyzed. Or consider implementing the centralized logging with OpenSearch solution.

Detection

CMSEC_9: How do you monitor your in-vehicle software, components, and network for threats?

Monitoring in-vehicle activity from threats, vulnerabilities, and fraud is becoming increasingly important as vehicles become connected. Standards like ISO 21434 and regulations like UNR 155 have provisions for monitoring vehicles and systems where detecting, reporting, refining, and analyzing vehicle related data is a requirement. Monitoring in-vehicle events that may have security implications requires providing solutions both on-board and off-board the vehicle. Before data collection, you must understand the sources of data that you need to collect from. You then develop the people, processes, and technology to address the data collection, ingestion, enrichment, and analysis of that data. You can then understand the threat landscape, use threat intelligence to proactively address risks, and monitor and analyze threats from collected data. The final step, triaging and incident response will be covered in the "Incident Response" section.

Data collection requires the ability to gather and record data from the vehicle and vehicle environments. Some vehicle data may be stored locally at smaller volumes and for specific use cases (e.g., crash event data) but is usually sent to an off-board system for refinement and

analysis. The collection of this data may require onboard components depending upon how much processing capability you want to implement onboard before sending the data off-board.

Ingestion requires sending data to an off-board service. Oftentimes this is referred to as a VSIEM (Vehicle Security Information Event Management System) or VSOC (Vehicle Security Operation Center). This data stream will be sent to other backend systems for enrichment and refinement.

[CMSEC_BP9.1] Collect and detect events using onboard components in the vehicle.

You must decide how you will collect data from the vehicle, and how much processing of security events will be done at the vehicle edge. It is recommended that you implement software that can detect vehicle attacks and raise an event to an IDS manager on the ECU. There may be some filtering by the IDS manager at this stage depending upon vehicle software and hardware capability. The events will then be passed to local event memory for storage, where it can be analyzed by an authorized user with a diagnostic tool. The majority of the event data should be sent to the VSOC system from the vehicle gateway for further refinement of raw logs sent from the vehicle.

[CMSEC_BP9.2] Ingest and enrich data by preparing, moving, aggregating, normalizing, transforming, and analyzing the vehicle data.

There are several mechanisms to send your vehicle data to the cloud (see Design principles). As mentioned in the Identity and access management section, we recommend using unique vehicle identities and sending data using secure protocols (see data protection).

You must take a wide a range of actions on your data in order to provide valuable capability to VSOC analysts. The typical pattern is to ingest, store, process, analyze/train, and then visualize the data. At the same time, this flow can include response and remediation in parallel, which is covered in the Incident response section of the pillar. In order to build out detection capabilities, you need to:

Ingest: You can use AWS IoT Fleetwise to send vehicle telemetry in VSS format, a COVESA specification standard for vehicle data that currently supports CAN and OBD-II, or with any data logger in the vehicle. You may also send raw vehicle data using protocols like HTTP or MQTT and build proprietary ingestion schemes on top of these protocols. You also want to use Amazon CloudWatch metrics, which allows you to capture message broker metrics such as number of connections and other metrics related to AWS IoT Core. AWS IoT Device Defender can capture cloud-side metrics, which are agentless metrics generated from IoT devices. You can also capture device-side metrics which are metrics generated from agents installed on the ECU.

Store: AWS provides the ability to easily securely send vehicle data to AWS. You can use AWS IoT Core Rules that can send vehicle logs to over 20 different services. Commonly you will use IoT Core Rules to route vehicle data to Amazon S3, which is an object storage service that offers industry-leading scalability, data availability, security, and performance. It is important to define storage requirements based on your organization and risk determinations. AWS provides *storage lifecycle* mechanisms in Amazon S3 to define data lifecycle policies based on your retention and cost requirements. Customers can also use Amazon Security Lake to centralize security logs from AWS and custom sources like the vehicle, charging stations, and more in an open-source standard format called OCSF. Security Lake enables both the centralization of log data and helps customers extract valuable insights quicker from normalizing to the open standard.

Process: You can then process and normalize that data in any way that fits your business case. AWS provides a service called AWS Glue to normalize, prepare, and integrate this data into a catalog for analysis. This allows customers to build schemas and normalize data to fit their needs.

Analyze: You can then run machine learning and analytics on this data to understand vehicle related baselines and build models for detection using services like Amazon SageMaker AI. Amazon SageMaker AI allows you to build, train, and deploy machine learning (ML) models for any use case with fully managed infrastructure, tools, and workflows.

Visualize: VSIEMs and VSOC can be used to identify security patterns based on the data and analytics that are run on the large data sets. Selecting the right visualization tool and strategy is an important aspect of identifying security trends and KPIs. AWS provides a number of services to help with visualizing vehicle security patterns in a detailed graph and report. Quick is a cloud-native serverless business intelligence service that provides data visuals, interactive dashboards, and data analytics powered by ML. Amazon Managed Grafana is a fully managed service for open-source Grafana, a popular open-source analytics service for querying, visualizing, and understanding your metrics no matter where they are stored. AWS OpenSearch with Kibana gives you the ability to aggregate logs from all your systems and applications, analyze these logs, and create visualizations for application and infrastructure monitoring, faster troubleshooting, security analytics, and more.

You can build a pipeline for vehicle data using the AWS services highlighted above. You can also consider adopting partner services from Argus Cyber Security or Upstream Security.

CMSEC_10: How do you correlate events from vehicles, backend systems, suppliers, vendors, and AWS Partners to generate actionable findings?

Organizations must integrate data from many data sources as inputs to VSOCs. It is necessary to gather information feeds from many relevant data sources that may be relevant when analyzing vehicle logs and determining vulnerabilities. Correlation requires trained expertise, proper tooling, and the correct data sources and learning models to provide accurate true positive results from different vehicle scenarios. Based on the analysis results, you can use runbooks to guide your response and recovery on the fleet or vehicle, which is covered in the incident response section of the security pillar.

Analyzing baselines and threat intelligence feeds can provide information that a VSOC analyst can use to build common detection patterns that are likely to occur. As the threat intelligence data is constantly changing and new threats and vulnerabilities emerge, it is important to provide a mechanism to continuously monitor for false positives and improve rules based on new data. Organizations should focus on building monitoring capability around the highest risks to safety and security, the scope of a possible security issue, and the plausibility of the security issue.

[CMSEC_BP10.1] Gather relevant supplier, vendor, production, API, and partner data to enhance detection mechanisms by adding further context.

A vehicle environment contains several different systems that interact with the vehicle and consumer. Log sources should be collected from any relevant system that can provide important security and operational insights related to the connected vehicle. Data collected from backend systems, companion APIs, charging stations, diagnostics databases, AUTOSAR XML databases, production data, partners, and other custom databases that are relevant to a connected mobility system. This data can also be stored in Amazon S3 or Amazon Security Lake in OCSF format.

The goal is to correlate data and build rules based on the models that are trained on this data. Lastly, correlating findings requires the ability to utilize threat intelligence and threat hunting resources to inform your environment and your baselines, and this can be done by integrating threat intelligence and vulnerability feeds to Amazon SageMaker AI mentioned above. The data can then be integrated into Amazon OpenSearch Service for analytics and uncovering insights, or to a central IT SOC on AWS or an AWS Partner such as Datadog, Sumo Logic, New Relic, Honeycomb, or Splunk**.** Amazon OpenSearch Service makes it possible for you to perform interactive log analytics, real-time application monitoring, website search, and usage of Security Analytics for OpenSearch that provides out of the box security visibility. These tools can provide the ability to detect suspicious vehicle interactions. Commands from a vehicle might be anomalous, or a vehicle may be communicating to a backend system that it shouldn't or usually does not communicate to.

It is also important to collect logs from applications that interact with vehicles such as companion or fleet applications. You can use API logs to determine issues such as an unauthorized user gaining access to vehicles they are not authorized to interact with. You can collect logs from sources like Amazon API Gateway and AWS IoT Core.

As previously stated, you can build a pipeline to gather relevant data to enhance detection using the AWS services highlighted above. You can also consider adopting partner services from Argus Cyber Security or Upstream Security.

CMSEC_11: How do you monitor and detect unauthorized use of vehicle credentials and identities?

You should be monitoring your ECUs for security best practices continuously. Due to the volume of ECUs and client certificates, it is important to monitor certificate and CA expiration and revocation. Organizations must monitor ECU permissions to detect overly permissive actions, and follow best practices when developing authentication and authorization mechanisms. It is also important to track changes in vehicle patterns that are significantly anomalous and deviating from known expected baselines.

[CMSEC_BP11.1] Detect security posture deviations and anomalous behavior

Monitoring certificate expiry and revocation can be a challenging process. Organizations must create processes and policies around the distribution and management of CAs and client certificates. AWS IoT Device Defender Audit provides the ability to monitor CA and client certificates that are nearing expiration (30 days) or that have been revoked. Organizations must detect overly permissive ECUs and scope down permissions as needed. AWS IoT Device Defender Audit can help to identify overly permissive policies that may be granting access to a broad set of resources instead of just a few.

Finally, organizations must build out vehicle baselines and automated alerting on anomalous behavior. Baselines might include expected number of connections, expected subscribed topics, expected protocols, expected payloads, and more. If a vehicle is deviating from those, such as sending suspicious commands, these should be detected. AWS IoT Device Defender Detect can help with the creation of rules or ML baselines to detect anomalous traffic based on pre-defined or custom metrics specific to your fleets, or even a group of devices in a profile.

For example, AWS IoT Device Defender Detect can provide you with visibility into ECU behavior such as the ECU failing authorization to multiple topics which may indicate the ECU trying to gain access to a topic that it is not authorized to publish or subscribe to. You can then trigger an Amazon CloudWatch Alarms which allows you to watch CloudWatch metrics and to receive notifications when the metrics fall outside of the levels (high or low thresholds) that you configure.

Alarms can send Amazon SNS notifications, which is a web service that coordinates and manages the delivery or sending of messages to subscribing endpoints or clients. Detect and Rule based findings integrate with AWS Security Hub CSPM, which is a cloud security posture management service that performs security best practice checks, aggregates alerts, and enables automated remediation.

CMSEC_12: How do you stay current and monitor on new threats and vulnerabilities after concept and design phase?

Organizations must continually ingest data from the vehicle and internal systems, as well as public and private data sources. Vehicles have several potential vulnerabilities that are unlike traditional IT systems. Defining a threat intelligence process and collecting data from a wide range of relevant sources will provide information on expected and emerging threats and vulnerabilities.

While protecting software and firmware from vulnerabilities is important, vehicle security programs must also consider emerging threats such as sensor fusion risks to the vehicle. An organization should develop mechanisms to consume a large variety of threat intelligence as well as share and collaborate across a sharing body such as Auto-ISAC to share learnings in a timely manner.

[CMSEC_BP12.1] Subscribe to threat intelligence feeds from many different sources to detect threats and vulnerabilities

You should analyze threat intelligence feeds from several data sources both public and private, beyond what is coming off of the vehicle or what is in your environment. This can include things like governments, vendors, information sharing bodies, peers, suppliers, open-source intelligence, and more. You may subscribe to specific vulnerability feeds regarding specific software and firmware running on the vehicle to address emerging threats and further enhance detection baselines to respond when emerging threats and vulnerabilities are known. It is important to maintain an inventory of embedded software and firmware, as well as operating systems,

packages, and libraries that are used per vehicle model and type so that you are aware of what is relevant to your environment based on threat intelligence feeds.

Infrastructure protection

CMSEC_13: How do you protect your systems and APIs from unauthorized access or exposure?

[CMSEC_BP13.1] Define a baseline of normal behavior for your vehicle as a reference and reject and alert on any request patterns that deviate.

Implement a solution to create a baseline of metrics for vehicles within your fleet. For example, AWS IoT Device Defender can be used for devices which connect to AWS IoT Core. This baseline can help in detecting anomalies such as impossible motion speeds, or a vehicle existing in multiple locations simultaneously. Through the use of Security Profiles within IoT Core and the enablement of metrics by Device Defender a threshold can be established to define this anomalous behavior. These thresholds can be either rule-based or use Machine Learning to evaluate how these devices should behave. An alert is generated when the defined threshold is crossed which can be used to perform automated mitigation actions you define such as updating a policy or certificate and can also send notifications via Amazon Simple Notification Service (SNS).

CMSEC_14: How do you test Electronic Control Units (ECUs) and implement configuration changes to minimize vulnerabilities?

[CMSEC_BP14.1] Conduct threat modeling using a formal methodology such as the MITRE Threat Assessment and Threat Analysis Risk Assessment (TARA) on in-vehicle systems.

[CMSEC_BP14.2] Conduct tests on ECUs to determine if there are any unnecessary configurations such as open ports, default permissions, unnecessary services running, or other vulnerabilities.

[CMSEC_BP14.3] Develop requirements and processes to address and remediate vulnerabilities discovered through testing.

CMSEC_15: How do you manage processes for updating the software of components within your connected vehicles?

CMSEC_BP15.1] Validate all software updates are cryptographically signed upon release and verified before installation.

For devices which use AWS IoT, software updates can be signed using AWS Signer a fully managed code-signing service. AWS Signer uses certificates imported to AWS Certificate Manager to sign software, AWS Identity Access Management (IAM) to assign roles to restrict who can sign and in what Region. All actions within AWS Signer are also captured in AWS CloudTrail for auditing purposes.

[CMSEC_BP15.2] Test software packages for vulnerabilities and errors prior to deployment.

[CMSEC_BP15.3] Perform controlled and secure deployment of software to vehicle fleets with rollback capabilities in case of failures.

CMSEC_16: How do you maintain an accurate inventory of your connected vehicles and associated software or hardware components?

[CMSEC_BP16.1] Maintain an accurate and continuously updated inventory of vehicles and any versions of software packages or underlying hardware.

This should include any applicable Software Bill of Material (SBOM) documents as outlined by something such as The Minimum Elements For a Software Bill of Materials (SBOM) pursuant to Executive Order 14028 including any Vulnerability Exploitability eXchange (VEX) documents as outlined by the United States Department of Energy in coordination with the National Telecommunications and Information Administration. A VEX document is an attestation and contains information about a product and whether it is affected by a known vulnerability.

CMSEC_17: How do you verify the integrity of the software running within your vehicles?

[CMSEC_BP17.1] Use methods for ensuring integrity such as through secure boot and software signing methodologies.

CMSEC_18: How do you secure the Software Development Lifecycle (SDLC)?

[CMSEC_BP18.1] Develop SDLC policies which govern these requirements and provide prescriptive guidance for implementing required controls.

CMSEC_19: How do you manage vulnerabilities for your connected vehicles?

[CMSEC_BP19.1] Continuously monitor informational feeds on threat intelligence from vendors, suppliers, third-party intelligence providers, repositories, and communities.

Threat intelligence feeds provide additional insights into signatures of potentially malicious activity such as malware, botnets, and phishing. This information helps to supplement the tools already in use by security teams to understand the current threat landscape at a broader scope. AWS Partners such as Recorded Future and Tenable provide threat intelligence feeds and are readily available through the AWS Marketplace. IP based threat lists from these providers can also be used to supplement detections within AWS GuardDuty our purpose-built threat detection service. Customers can also leverage an AWS partner like Argus Cyber Security to help implement in-vehicle security systems like an Intrusion Detection System for Controller Area Networks (CAN), vehicle vulnerability management, and firewall capabilities to protect components. Industry specific feeds such as Auto-ISAC can provide additional insights based on contributions from organizations within automotive.

[CMSEC_BP19.2] Conduct vulnerability impact analysis and testing in isolated and secure environments.

When conducting tests of vulnerabilities on systems or investigating potentially compromised systems an isolated and secure environment should be used. This verifies that other resources are not inadvertently impacted or compromised by these activities. When conducting this testing in AWS environments a separate Account should be created to provide segmentation from other Accounts and resources. Strong defense-in-depth configurations should also be used such as for forensic investigations. Some of these strategies are outlined in the Forensic investigation environment strategies in the AWS Cloud blog post.

[CMSEC_BP19.3] Define a process for classifying and prioritizing vulnerabilities for remediation.

Processes for classification, prioritization, and remediation of vulnerabilities should be formally documented and readily accessible to stakeholders. Playbooks and runbooks specific to your organizations technology stack and operational processes should be used as a means of automating segments of the investigation and help minimize manual errors during a potentially stressful situation. Automation can be accomplished through solutions such as Jupyter Notebooks or running automation documents within AWS Systems Manager for example.

Data protection

CMSEC_20: What are your encryption requirements for vehicle data?

Vehicles can collect and generate data with different levels of sensitivity and risk to your organization. You must check that the data is protected using appropriate means to comply with applicable standards, regulations, and laws. Encrypting the data in transit and at rest and controlling access to the encryption keys can be an effective technical means to help meet your data protection requirements. Begin by classifying the data generated by the vehicle to determine the type of encryption required in transit and at rest.

Consider the following best practices for encryption:

[CMSEC_BP20.1] Implement TLS 1.2+ to encrypt data in transit

By implementing TLS 1.2+ from the ECU to the connected vehicle service backend you get encryption in transit. With the addition of X.509 client certificates on the ECUs for mutual TLS you also get authorization of requests. Verify that you follow the best practices on managing the lifecycle of certificates as described above.

On AWS, you can use mutual TLS 1.2+ with AWS IoT Core and Amazon API Gateway REST APIs to authenticate ECUs using X.509 client certificates. You can also terminate mutual TLS on software deployed to AWS compute services (Amazon EC2, Amazon EKS, or Amazon ECS) behind a Network Load Balancer (NLB). Individual AWS service API requests are authenticated using the *sigv4* algorithm.

[CMSEC_BP20.2] Consider client-side encryption of highly sensitive data at the ECU

When highly sensitive data is transmitted from the vehicle to a connected vehicle backend, you may consider implementing client-side encryption of the data. In this approach, the vehicle ECU encrypts the sensitive data payload ***before*** transmitting it to the connected vehicle backend service. The encrypted payload is transmitted via any intermediate systems to the backend system that has access to the keys to decrypt the payload and recover the sensitive data in the clear. Note that message parts such as HTTP headers and MQTT topic names should still be transmitted without client-side encryption so that intermediate nodes can route and queue messages. The benefit is that any intermediate systems such as a load-balancer or message broker are unable to see the sensitive data payload in the clear as these systems do not have permissions on the keys. This scheme relies on managing the encryption keys securely and granting least-privilege permissions on keys to the ECU and backend servers.

An effective pattern for client-side encryption is envelope encryption in conjunction with a key management system. In this pattern, the plaintext sensitive data is encrypted by the ECU with a *data key* and the data key is encrypted under another key that is managed in a key management system. The ECU transmits the encrypted data and the encrypted data key to the backend service. The backend service invokes the key management system to decrypt the encrypted data key, recovering the plaintext data key. It can then use the plaintext data key to decrypt the encrypted data. The ECU can cache the data key for a period of time or for a certain volume of data, improving performance by minimizing the need to invoke the key management service for every sensitive data element. An intermediate system that receives the encrypted data key does not have permissions to invoke the key management system to decrypt the data key and as a consequence cannot decrypt the encrypted data. The sensitive data payload remains encrypted end-to-end from the ECU to the backend service in transit and at rest.

On AWS, you can use the AWS Key Management Service (AWS KMS) to securely manage your encryption keys for envelope encryption. You can use the AWS Encryption SDK to implement envelope encryption with data key caching on the ECU and backend servers to improve performance, help reduce cost, and stay within the AWS KMS service limits as your application scales. Your ECUs can obtain temporary AWS API credentials to invoke AWS KMS API calls by using the AWS IoT Credential Provider.

CMSEC_21: How do you identify and protect data in your connected vehicle service?

Classify the data processed by your connected vehicle service as described in Well-Architected Security Data Protection. Specifically, consider how you will classify data such as Vehicle Identifier

(VIN), ECU serial number, vehicle location (coarse vs precise), and operator identity. Consider the specific requirements that apply to your organization, depending on factors such as where you conduct business, any industry-specific regulations, the type of data, where or from whom the data originates, and where the content is stored and processed.

Consider the following best practices to protect data.

[CMSEC_BP21.1] Do not use sensitive data to name and reference resources

You may have several resources in your connected vehicle architecture that must be uniquely named. Avoid using sensitive data such as the VIN, customer identity, location in these resources so that you can more effectively control access to the data. For example, when using MQTT, avoid using the VIN in the topic names used to send and receive messages from a vehicle's ECUs. MQTT topic names are often logged to audit log files, and can be used when subscribing to messages. Using a hashed VIN or another unique identifier that cannot be mapped to a VIN will help you limit the number of locations, systems and people that can see the VIN.

On AWS, you can use Amazon Macie to discover and help protect your sensitive data. Macie uses a combination of criteria and techniques, including machine learning and pattern matching, to detect sensitive data in Amazon Simple Storage Service (Amazon S3) objects. Macie can detect a large and growing list of sensitive data types for many countries and regions, including multiple types of credentials data, financial data, personal health information (PHI), and personally identifiable information (PII). VIN can be detected using a managed data identifier. Additionally, you can build custom data identifiers using regular expressions (regex) to match vehicle specific identifiers, such as ECU IDs, and ECU serial numbers.

Incident response

CMSEC_22: Does your team deploy a VSOC? If so, what are the required capabilities of a VSOC?

A VSOC has become a compliance requirement for automotive cybersecurity management systems very recently. An incident may involve safety implications, making incident response a critical component of a security program. A VSOC must contain automotive specific playbooks that are relevant to your business case, collaboration across IT SOC, cloud SOC, and VSOC, continuous improvement and lessons learned. A VSOC must have many of the detection mechanisms we described above [refer to CM_SECBP20 for detection] as well as providing triage, containment,

recovery, and lessons learned to your monitoring and remediation program. A VSOC also requires trained personnel that are capable of addressing different vehicle alerts with varying complexity.

[CMSEC_BP22.1] Define VSOC capabilities and requirements, then develop and test your VSOC incident response plan.

To successfully respond to an incident, you must first define VSOC capabilities and requirements. This aligns with compliance or risk-based security requirements. You then prepare your incident response plan and runbooks that are referenced during a potential security incident. The incident response plan contains several components that span across an organization, and cover both business and technical steps. AWS provides Automation runbooks which define the actions that Systems Manager performs on your managed instances and other AWS resources when an automation runs. runbooks contain one or more steps in sequential order. You can use runbooks for manual approvals or trigger a workflow.

The VSOC should include security orchestration automation and response (SOAR) and ticketing capability to provide timely responses and workflows that can prioritize and address incidents. Customers can send findings to AWS Security Hub CSPM, which integrates with issue tracking systems like ServiceNow and Jira. The incident response procedure must be consistent, accurate, and updated when necessary. Running game days and tabletop exercises can provide insight about the ability to handle an incident in a practice environment. Game days, also known as simulations or exercises, are internal events that provide a structured opportunity to practice your incident management plans and procedures during a realistic scenario.

[CMSEC_BP22.2] Train or outsource personnel to manage security incidents at multiple layers of complexity.

One of the major observed gaps is a lack of expertise in vehicle incident response. You must determine if your organization has the resources, processes, and skills in place to address vehicle incidents and events at multiple tiers based on the severity and complexity of the incident. Organizations might shift responsibility to an AWS Partner or vendor to manage their VSOC and personnel to identify, prepare, analyze, contain, and recover from a vehicle security incident. Argus Cyber Security provides consultive and managed VSOC services that can help address those security needs.

CMSEC_23: How do you contain, recover, and learn from incidents that can span vehicles, backend systems, APIs, IT, cloud, AWS Partners, and supplier resources?

Organizations must be able effectively respond to an incident and notify the organization of severity and scope. This involves processes to triage and prioritize, response and recovery activities, and properly monitoring and closing an incident after confirming the incident has been fully resolved. Organizations will then continuously improve by going through lessons learned and using the process to inform the next set of service improvements after the incident has been mitigated and resolved.

[CMSEC_BP23.1] Mitigate and respond to potential incidents by creating and testing policies, procedures, and playbooks

During a potential vehicle security issue, it is necessary to attempt to automate and respond to an incident promptly. By using runbooks, you can automate several workflow tasks like notifying different stakeholders and creating a ticket. You must be able to contain the scope of the issue. Depending on the finding, you can issue APIs to AWS IoT Core where you can automate changing a certificate status based on the incident. If a vehicle is compromised, you can build a workflow that will inactivate or revoke a certificate and block communications to AWS IoT Core while you investigate the incident. You should follow your incident response procedure to then recover the vehicle back to a non-compromised state which can vary in complexity depending on the incident.

Application security

CMSEC_24: How do you make sure that only trusted software components are running on vehicle hardware?

[CMSEC_BP24.1] Consider digitally signing software and firmware with a certificate that can be verified by the vehicle hardware during runtime ensuring that only trusted code can run on the vehicle.

Code signing and secure boot are essential for ensuring the security and safety of vehicle software. They help you validate authenticity and integrity of the software running in vehicle making sure that it comes from a trusted source and its behavior has not been altered. A secure boot mechanism helps prevent unauthorized code from running on the vehicle hardware by verifying the integrity of the boot loader and operating system. You can leverage AWS Private Certificate Authority (AWS Private CA) that allows you to create private certificate authority (CA) hierarchies, including root, code signing certificates issuing CAs and code signing certificates. If you are leveraging AWS IoT you can also use AWS Signer to sign code that you create for IoT devices

supported by Amazon FreeRTOS and AWS IoT device management. Code signing for AWS IoT is integrated with AWS Certificate Manager.

CMSEC_25: How do you check that you are writing, testing, validating, and deploying vehicle software securely?

[CMSEC_BP25.1] Implement a Secure Development Lifecycle (SDLC) for your vehicle software following open standards.

Start by identifying and creating a list of cybersecurity metrics and requirements that needs to be met to verify your software security regarding ISO 21434 standard from conception, product development to cybersecurity validation.

Create and document early in the design phase a threat model for your in-vehicle software or system such Head Units (HU) or Telematic Control Units (TCU). You can leverage established frameworks such STRIDE (Spoofing, Tampering, Repudiability, Information Disclosure, Denial of Service, and Elevation of Privilege) or the TARA framework in ISO 21434 for in-vehicle products and services, from identifying assets in the target system, vulnerabilities and attack scenarios and their impact, those most likely to be exploited and risk treatment decision. AWS provides an example threat modeling workshop and corresponding blog post to apply some principles when threat modeling a connected vehicle cloud application.

Implement secure coding, development process and architectural design following open standards guidelines such AUTOSAR and ASPICE (examples: input validation, buffer overflow protection, error handling to prevent attacks such as SQL injection and cross-site scripting and hardcoded credentials). Conduct code reviews and security audit of the code to identify security vulnerabilities that may be missed during development. Test and validate the software in simulated and real-world scenarios to check it is safe and reliable.

You can leverage the KPIT Cloud Native Engineering Workbench solution on AWS to help accelerate your software-defined vehicle (SDV) development and testing.

Key AWS services

- Amazon Inspector
- AWS Signer

- AWS Private CA
- AWS Secrets Manager
- AWS WAF
- AWS Shield
- AWS Developer Tools
- AWS Security Hub CSPM
- AWS Identity and Access Management
- AWS Glue
- AWS IoT Family
- Amazon Security Lake
- AWS Key Management Service
- AWS Systems Manager
- Amazon Macie
- AWS Identity and Access Management
- Amazon CloudWatch
- Quick
- Amazon Managed Grafana
- Amazon OpenSearch Service

Resources

Documentation and blogs

- Securing Modern Connected Vehicle Platforms with AWS IoT
- How to approach threat modeling
- Best practices securing Amazon Location Services
- How to detect anomalies in device metrics and improve security posture with AWS IoT Device Defender
- Forensic investigation environment strategies in the AWS cloud

AWS Partner solutions

- DataMasque
- Argus Cyber Security
- Recorded Future
- Tenable
- Upstream Security

Workshops

- Automating operations with Playbooks and Runbooks
- Incident Response labs
- Threat modeling for builders
- Security for Developers

Reliability pillar

The reliability pillar for connected mobility encompasses the ability of connected vehicles to deliver the services as intended and consistently. Resiliency is a shared responsibility between AWS and the customer. It is important that you understand how high availability (HA) and disaster recovery (DR), as part of resiliency, operate under this shared model.

As an Auto OEM, your responsibility would change based on the configuration that is needed for a particular service. If your connected mobility uses Amazon EC2 for the connectivity gateway and vehicle data platform, then you are responsible for deploying EC2 instances across multiple locations such as Availability Zones and Regions, implementing self-healing systems like AWS Auto Scaling. If you have managed services, such as API Gateway, AWS IoT, Amazon S3 and Amazon DynamoDB, AWS operates the infrastructure layer, the operating system, and platforms, and you access the endpoints to store and retrieve data. In both cases, you are responsible for managing resiliency of your data including backup, versioning, and replication strategies.

Design principles

In addition to the overall Well-Architected Framework design principles, the following are the design principles for reliability for connected mobility:

- **Design for failure and resiliency:** It's essential to plan for resiliency on the vehicle-based telematics unit. Depending on your use case, resiliency might entail robust retry logic for

intermittent connectivity, ability to roll back firmware updates, ability to fail over to a different networking protocol for critical message delivery, ability to perform a factory reset.

- **Implement synthetic monitoring:** Simulate the vehicles behavior by implementing digital clone of the in-vehicle software communicating with the connected mobility application. The monitoring must share no cloud resources with the connected mobility backed that is monitoring to being able to assess impairments of the cloud part.
- **Plan for poor network connectivity:** Vehicles must tolerate connectivity or backend errors. Connectivity is subject to continuous interruption given the position of the vehicle, like tunnels, underground parks or remote locations with low to none signal. Cloud backend applications must tolerate vehicles that often transition between being connected and disconnected. Availability of connected services and user experience on disconnection is the core aspect and basic reason of this platform's existence.
- **Invalidate commands that have passed their usability threshold:** Connected mobility platforms enable the communication of commands to the vehicle. The commands are intended to be run on the vehicle at the requested or configured time, generally in near real time. At times, the state of the vehicle or the backend system may be such that this communication fails to execute in expected timeframe. Due to this delay the command may either have been overridden by next command, are no longer relevant or may be not safe enough to execute. For example, a door unlock command might not be safe enough or relevant to run after few minutes of delay. Based upon the use case and the nature of the commands, the backend system should invalidate the commands if they are stale or passed their usability threshold.
- **Design for idempotent systems:** There can be situations where the connected mobility system might receive multiple messages that are identical thus causing an overload. Although the complexity of implementation would be higher, idempotency reduces the risks of system overload, failure, and reduced availability.
- **Aggregate for simplicity but be prepared to dive deep into logs for causal analysis:** Connected mobility platforms collect data at high velocity and volume. Vehicle manufacturers use various mechanisms to aggregate this data for specific vehicle or fleet over a time. Based upon the use case, a deep dive into the system and application logs may be needed to troubleshoot issues, determine performance bottlenecks, execution steps, or to meet compliance regulations.

Best practices

Topics

- Change management
- Failure management

Foundations

CMREL_1: Have you defined your message prioritization policy?

Connected vehicles generate a large amount of data that is sent to the cloud via messages. It's important to prioritize the message flow and handle different message severities with different policies.

Connected vehicles generate a large amount of data that must be sent to the cloud via messages. It's important to prioritize the message flow and handle different severities with different policies.

[CMREL_BP1.1] Defining message tier matrix

The connected mobility solution must classify messages from the vehicle fleet to the cloud backend in at least two tiers, such as:

- High priority messages
- Low priority messages

These two tiers must be processed using different approaches when it comes to resiliency as described in the following sections. The tier structure is multidimensional to account for messages that should be sent as soon as possible or not sent at all (urgency dimension), and for relevancy. In the urgency dimension are messages that pertain to GPS and ADAS messages that must be sent, even if they cannot be sent in real time (relevance dimension). Messages for emergency calls are also in this dimension. Lowest on the priority list (low urgency and low relevance) include telemetry messages that can be sent in batches on a given schedule.

[CMREL_BP1.2] Have a strategy for critical functions

Connected mobility should implement critical functions. such as emergency calls in which messages from the vehicle must be delivered to the cloud no matter what, to overcome transient failures in the connectivity as well as in the control plane. Those messages should sit on top of the

tier matrix (high relevance, high urgency). In such cases, you should plan for redundancy for every physical or logical component of in-vehicle as well as the communication layer with the cloud (such as SIM failure, modem failure, and telco failure).

Consider building the control plane in multiple Regions and have the vehicles connect to the first available endpoint for these critical functions.

CMREL_2: How do you ensure that you do not overflow your communication channel to the cloud?

Messaging from vehicles to the cloud is likely to be unpredictable, generating spikes that might approach AWS service quotas. If proper measures are not taken, these spikes could result in message throttling, application impairment, or both.

[CMREL_BP2.1] Guardrail chatty vehicles

The connected mobility solution must tolerate anomalies in the message workflows received from the vehicle fleet. The connected mobility control plane that receives messages from the vehicle fleet must tolerate spikes in the vehicle's messaging flow throttling low priority messages down to avoid exceeding service quotas and unwanted cost spikes. If throttling of low priority messages is not enough to avoid exceeding a service quota, messages must be classified in more than two tiers and a policy must be implemented to throttle messages down when appropriate.

[CMREL_BP2.2] Avoid connection surge to the cloud

A vehicle’s side application and firmware must implement a randomized connection to the backend cloud applications to avoid unnecessary peak traffic. Situations where the entire fleet attempts the same operation at the same time must be avoided. Traffic generated from low priority tier messages should be randomized. Navigating the tier structure bottom up, trade off decision must be taken and implemented whether to randomize the traffic avoiding connection peaks or increase a service quota (where applicable). The same randomized behavior must be implemented on the cloud backend for messages that are being sent to the vehicles.

CMREL_3: Do you have a strategy in case connection certificates are unavailable or accidenta lly deleted?

As a best practice, vehicles establish a connection to the backend by exchanging certificates. The connected mobility control plane must store these certificates with the highest level of durability, considering that installing new certificates to the client likely requires the vehicles to be called back at the OEM or auto dealership. Backup strategy must be built and emergency registration procedure must be available. Be sure to test both the restore procedure and the emergency procedure with tabletop exercises.

[CMREL_BP3.1] Implement just-in-time provisioning and registration

When using AWS IoT Core to connect vehicles to the AWS Cloud, it's a best practice to implement just-in-time provisioning (JITP), just-in-time registration (JITR), or both. In both cases, certificates are provisioned the first time devices connect to AWS IoT Core by using a certificate template (JITP) or an AWS Lambda function (JITR). Using JITP and JITR can help restore a compromised device registry.

[CMREL_BP3.2] Manual backup of the device registry

AWS IoT Core stores information about your devices in the device registry. It also stores CA certificates, device certificates, and device shadow data. In the event of hardware or network failures, this data is automatically replicated across Availability Zones but not across Regions.

AWS IoT Core publishes MQTT events when the device registry is updated. You can use these messages to back up your registry data and save it somewhere, like a DynamoDB table. You are responsible for saving certificates that AWS IoT Core creates for you or those you create yourself.

CMREL_4: How is your connected mobility solution resilient to unintended access?

Every infrastructure is susceptible to unintended access and threat actors. Connected mobility is no exception. Your connected mobility solution should implement the security by design approach to reduce the scope of impact and mitigate and help prevent inadvertent access. The solution should be resilient to events even if some functions are impaired by these issues.

[CMREL_BP4.1] Implementing a layered approach

An OEM can mitigate the negative impact of unintended access events on connected vehicles by adopting a layered approach. In vehicle design, physical networks can be separated by artificial layers. For example, you can create an engine control unit layer, an in-vehicle communication network layer, and an external interfaces layer. The layer creation will involve technologies such as

gateways, firewalls, message authentication, encryption, and intrusion detection and prevention systems. During vehicle manufacturing, several practices can also be considered to identify and mitigate connected vehicle issues and risks, including:

- Developing over-the-air (OTA) update capabilities for connected vehicle software and firmware.
- Conducting risk assessment and attack testing.
- Creating domain separation for in-vehicle networks.

  Mission- and safety-critical components in a connected vehicle can be separated from non-critical components, and given limited connectivity to external networks through a few specific communication channels.

CMREL_5: How do you mitigate the impact of impairments in the connection layer between vehicles and the AWS Cloud?

Connected mobility is a peculiar scenario in Internet of Things (IoT) given that the things don't have a reliable Local Area Network (LAN) to which they are connected. Vehicles use mobile connectivity which is likely provided by a third party, and vehicles might move to locations where the mobile connectivity is limited or not available at all.

[CMREL_BP5.1] Account for telco providers outages

Vehicles connect to the cloud control plane leveraging on telematic providers services (telco), the connected mobility solution must account for failures in this transport layer. The solution must be able to distinguish if a connection drop has been caused by the vehicle, the telco, or the control plane in the cloud.

[CMREL_BP5.2] Implement in-vehicle functionalities

Connected mobility application reliability must also encompass the vehicle itself. Vehicles might be operating in remote locations and deal with intermittent connectivity, or loss in connectivity, due to a variety of external factors that are out of your connected mobility application's control. For example, if an ISP is interrupted for several hours, how will the vehicle behave and respond to these long periods of potential network outage? Implement a minimum set of embedded operations on the vehicle to make it more resilient to the nuances of managing connectivity and communication to AWS control plane.

The vehicle must be able to operate without internet connectivity. You must implement robust operations in your vehicle firmware and software to provide the basic capabilities. Store important messages durably offline and, once reconnected, send those messages to the AWS control plane. Implement exponential retry and back-off logic when connection attempts fail.

Change management

At any OEM, connected mobility landscape is continuously evolving with new use cases and implementation patterns. On the vehicle side, the number of sensors collecting data have been rapidly multiplying. This results in refactoring and rewrite of some of the processing logic that feeds the vehicle data platform and enables deriving of insights. Thus, there is greater need to have streamlined processes that introduce and manage change in your environment. But based on the use cases, application, and infrastructure, the run-books and deployment strategies vary. These changes to the workloads and environment should be anticipated, monitored, accommodated and carefully executed for reliability of the connected mobility platform.

Monitor workload resources

CMREL_6: Are you monitoring all components of the workloads, including vehicle-based units?

[CMREL_BP6.1] Monitor what matters.

Observability is understanding the working of the system based on the telemetry that the system emits. A recommended approach is to implement a Vehicle Network Operations Center (V-NOC) which is an intelligent observability system that is aware of the health of the systems running in the vehicle, the backend systems supporting the platform and the applications that are user facing. It is essential to monitor all components of the workloads including vehicle-based units.

Generally, a Connected Mobility platform has several integrations with internal and external systems. Any system change that can impact the interfacing message contracts needs deep dive impact analysis. Enable the observability on these integrations for quicker troubleshooting, failure analysis and performance profiling.

Vehicle based telematic units should have health monitoring and data transmission mechanism that can relay the state of systems in near real time when connection is available or ship buffered logs when connectivity is restored.

Prescriptive guidance:

AWS provides native monitoring, logging, alarming, and dashboards with Amazon CloudWatch and tracing through AWS X-Ray. When deployed together, they provide the three pillars (metrics, logs, and traces) of an observability solution. AWS services such as Amazon CloudWatch apply statistical and machine learning algorithms to continually analyze metrics of systems and applications, determine normal baselines, and surface anomalies with minimal user intervention. Anomaly detection algorithms account for the seasonality and trend changes of metrics.

If the preference is for open-source based managed services Amazon Managed Service for Prometheus and Amazon Managed Grafana are two such services providing additional options for customers to choose from. AWS also launched AWS Distro for OpenTelemetry (ADOT) - a secure, production-ready, AWS-supported distribution of the OpenTelemetry project. Part of the Cloud Native Computing Foundation, OpenTelemetry provides open-source APIs, libraries, and agents to collect distributed traces and metrics for application monitoring. With AWS Distro for OpenTelemetry, you can instrument your applications just once to send correlated metrics and traces to multiple AWS and Partner monitoring solutions.

It is important to be aware of end-to-end functioning of all endpoints that implement the connected mobility use cases. This active monitoring can be done with synthetic transactions which periodically run a number of common tasks matching actions performed by clients of the workload. You can also combine the synthetic canary client nodes with AWS X-Ray console to pinpoint which synthetic canaries are experiencing issues with errors, faults, or throttling rates for the selected time frame.

In all cases, tool interoperability and extensibility are an important consideration in observability.

CMREL_7: Are your connected mobility logs from various systems lacking the correct level of information?

[CMREL_BP7.1] Log interpretation and context propagation

Considering the number of systems and services supporting a connected mobility setup, it is important to collect logs into a centralized store. But all log entries are not equal, in addition the logs may be too verbose or lacking right level of info that can be used in correlation. A system that enables the filtering of logs based on prefixes would reduce the amount of data that is retained and processed for insights. A processing / transformation engine is needed to make the logs more

usable or to enrich the log entries with additional information that can be used later for better correlation.

As connected mobility platforms result in high volume log generation, a system that simplify the searching, querying and visualizing of the logs in different ways based on need is recommended.

An ideal NOC should have capability to create dashboards, reports, and allow dynamic querying capability. Considering the mix of systems that are on-board and off-board the log entries may have sensitive information like location of the vehicle. Securing the logs with encryption at rest, auditing and masking sensitive data in logs is required for compliance validation.

Prescriptive guidance:

CloudWatch Logs enables you to centralize the logs from all of connected mobility systems, applications, and AWS services that you use, in a single, highly scalable service. You can then easily view them, live-tail, search them for specific error codes or patterns, filter them based on specific fields, or archive them securely for future analysis. CloudWatch Logs enables you to see all of your logs, regardless of their source, as a single and consistent flow of events ordered by time.

CloudWatch Logs Insights enables querying your logs with a powerful query language, visualizing log data and adding them to dashboard.

The filtering of the logs can be done using CloudWatch logs subscription filters that can have four different target services: Kinesis Data Streams, AWS Lambda, Amazon Data Firehose and Amazon OpenSearch Service

For a list of AWS services that publish logs to CloudWatch Logs, see the CloudWatch documentation.

Some AWS services can directly write logs to other destinations

Log type	CloudWatch Logs	Amazon S3	Firehose
CloudFront: access logs		✓	
CloudWatch Evidently evaluation event logs	✓	✓	

Log type	CloudWatch Logs	Amazon S3	Firehose
Amazon ElastiCache (Redis OSS) logs	✓		✓
AWS Global Accelerator flow logs		✓	
Amazon MSK broker logs	✓	✓	✓
Amazon MSK Connect logs	✓	✓	✓
AWS Network Firewall logs	✓	✓	✓
Network Load Balancer access logs		✓	
Amazon Route 53 resolver query logs	✓	✓	✓
EC2 Spot Instance data feed files		✓	
Amazon Virtual Private Cloud flow logs		✓	✓
Amazon VPC Lattice access logs	✓	✓	✓
AWS WAF logs	✓	✓	✓

CMREL_8: Is your metric collection aligned with a business outcome?

[CMREL_BP8.1] Define and calculate metrics (Aggregation)

Metric collection should always begin with an objective or business outcome. Defining metrics that are business outcome linked will result in quicker response from the system and teams supporting it than the generic metrics. As an OEM you are aware of regular pattern or trend for certain metrics in your system. For example, a critical KPI could be the number of remote start commands executed on the vehicles across various hours of the day. Generally, the count is higher in the morning and evenings which changes across time zones and seasons. With the release of a new version of the services that process such requests if the traffic has anomaly, it should get reflected on dashboard as an issue. Aggregations should be available on that metric to determine the severity of the impact.

Prescriptive guidance:

You can create metric filters to match terms in your log events and convert log data into metrics. When a metric filter matches a term, it increments the metric's count. For example, on release of a new firmware you can create a metric filter that counts the number of times the word `ECU_Connected` occurs in your log events. You can assign units and dimensions to metrics. For example, if you create a metric filter that counts the number of times the word `ECU_Connected` occurs in your log events, you can specify a dimension that's called `ECUConnectionCount` to show the total number of log events that contain the word `ECU_Connected` and filter data by reported firmware version.

CMREL_9: Does your connected mobility network operations center (NOC) have the correct level of searchability and interactivity?

[CMREL_BP9.1] Real-time processing and alarming with notifications and automated response

Some OEMs may have existing processes and tools for alarming, trouble tracking and automated response. For better visibility across the organization, the Vehicle-NOC should be integrated with these systems. With thousands to millions of vehicles connecting to the platform the number of data points can be too many so having an intelligent system that can reduce noise by finding pattern across various issues can boost productivity and reduce mean time to resolve (MTTR). Typically, such intelligent systems also reduce the number of repeat notifications for same issue.

Prescriptive guidance:

Alerts can be sent to Amazon Simple Notification Service (Amazon SNS) topics, and then pushed to any number of subscribers. For example, Amazon SNS can forward alerts to an email alias or messaging channel so that technical staff can respond.

When you enable anomaly detection for a metric, CloudWatch applies statistical and machine learning algorithms. These algorithms continuously analyze metrics of systems and applications, determine normal baselines, and surface anomalies with minimal user intervention. In addition, anomaly detection on metric math is a feature that you can use to create anomaly detection alarms on the output metric math expressions.

Design your workload to adapt to changes in demand

CMREL_10: How does your connected mobility workload adapt to vehicle traffic demand on resources?

[CMREL_BP10.1] Use automation when obtaining or scaling resources

[CMREL_BP10.2] Scale resources reactively on impairment to restore workload availability

[CMREL_BP10.3] Scale resources proactively to meet demand and avoid availability impact

Telemetry traffic from vehicle has patterns that vary based upon various factors like time of the day, weather condition during a day, season, geographic location. If the connected mobility systems are scaling to meet the demand, automating the process by using managed services will aid in better control.

When you are automating for scaling it is important to know the target utilization which generally based upon observations of historic trends and extrapolating. In a typical set up for connected mobility this may turn out to be a complex task due to the typical mix of several different services.

Another challenge is to monitor your connected mobility applications / services as the capacity is added or removed in real time as the demand changes. This will build confidence to have right level of end user experience as the workloads change periodically or unpredictably.

Prescriptive guidance:

The exact nature of the automation depends upon the type of services that are supporting the landscape. Managed services like AWS Lambda, Amazon S3, Amazon CloudFront and others scale

automatically based upon the load condition. There may be limitations imposed by the Service Quotas which need to be taken into account.

Self-managed services like Amazon EC2 require careful planning to ensure that the load distribution meets the requirements for the business function. The automation should ensure that the instance returns to the state that it is expected to be in to handle the traffic. Automation is vital to efficient DevOps, and getting your fleets of Amazon EC2 instances to launch, provision software, and self-heal automatically is a key challenge. Amazon EC2 Auto Scaling provides essential features for each of these instance lifecycle automation steps. Use load balancers to distribute the traffic across the instances and Route 53 to provide flexibility in cloud service configuration without impacting the ECU based client applications.

To make the scaling experience better and simple AWS launched predictive scaling policies, which uses machine learning to analyze each resource's historical workload and regularly forecasts the future load for the next two days. To bring about wider level of control, in 2023, the forecast and scaling were decoupled so that you could get forecasting which is prescriptive in nature as compared to forecast and scaling. Predictive scaling can be used for applications where demand changes rapidly but with a recurring pattern.

Automotive companies generally use the AWS managed services to reduce the overheads of administrating the infrastructure and let AWS handle the undifferentiated heavy lifting. Configuring these managed services is Customer's responsibility. But considering the varied nature of these services, configuring them for auto-scaling and with cohesiveness becomes challenging very quickly. Thus, even in this case, you need a service that orchestrates the auto-scaling across the workload. With Application Auto Scaling, you can configure automatic scaling for the various resources beyond Amazon EC2.

[CMREL_BP10.4] Load and stress test your workload

Having followed the best practices above it is important to test if the scaling activities meet the requirements of connected mobility functions. Stress testing the platform would reveal the robustness of various connected mobility functions under extreme load conditions. Some use cases may deserve higher level of robustness than others due to various business and compliance reasons. The load testing frameworks should have the ability to execute various testing strategies with varied traffic patterns that cover both the regular business as usual cases and the corner cases of failure and subsequent recovery. Considering the agility and cost optimization that AWS cloud computing provides, setting up of stage environments for load testing should be less time, money and effort intensive. While load testing in the stage environment should be done, training the teams to handle such events in production environment is also a must-do activity.

Prescriptive guidance:

Distributed load testing can help load test the applications and determine the bottlenecks before releasing to production. The framework can simulate thousands of connected vehicle applications and generate traffic patterns to uncover issues. The systems can be tested using simple get requests or for higher level of customization you can create JMeter scripts.

Game Days can simulate a failure or event to test systems, processes, and team's responses.

Implement change

CMREL_11: How are you controlling the changes that are deployed?

The number of features and scenarios that are supported by connected mobility platform have been growing rapidly. It is recommended to control the impact of these changes while deploying new functions.

[CMREL_BP11.1] Use runbooks for standard activities such as deployment

Vehicles which are connected to the backend are generally operating in various geographies and time zones. It is important to plan the change activities at a time that can cause least disruption to the services. The documentation should be less verbose and specific to conducting the tasks. The execution of the steps should be controlled by right level of authorizations that requires approvals and has appropriate reason captured. Unauthorized changes can result in downtime in the connectivity of vehicles to backend. It is important to track the changes that were implemented so that the deployment rollbacks can be analyzed for integrity.

Prescriptive guidance:

Runbooks provide an excellent mechanism to communicate the actions that can be performed with minimum information. The change activities are generally audited and can be rolled back to the previous version. AWS Config performs compliance checks based on these rules, and Audit Manager reports the results as compliance check evidence.

Resources:

Using AWS Config managed rules with Audit Manager

Using AWS Config custom rules with Audit Manager

Troubleshooting AWS Config integration with Audit Manager

[CMREL_BP11.2] Integrate functional and resilience testing as part of your deployment

Functional and resilience tests should be part of automated deployment. If success criteria are not met, the pipeline is halted or rolled back. These tests are run in a stage environment and done as part of a deployment pipeline.

Prescriptive guidance:

Vehicle owners or the related systems can take different functional paths while using the connected services. It is important to track such paths and simulate them in steps using services like Amazon CloudWatch Synthetic monitoring. As a bonus this also reveals the uptime of the services.

Chaos engineering with AWS Fault Injection Service can be used to build confidence in the system's ability to survive certain corner cases. To execute such tests many of the principles discussed above are applicable- determining the target state, run experiments in production and automation.

Resources:

Videos:

AWS re:Invent 2022 - Building confidence through chaos engineering on AWS

[CMREL_BP11.3] Use a canary or blue/green deployment when deploying applications in immutable infrastructures

The operational complexity of the connected mobility platform builds up with the numerous distributed services that support various scenarios. Release cycles, patching, monitoring the changes all result in overhead that is difficult to manage and error prone. Offboard teams begin to delay the release of changes to avoid disruptions in availability of services. A solution to this problem is to have immutable infrastructure, where infrastructure is not updated or fixed rather it is replaced.

Prescriptive guidance:

Canary release and Blue/green deployment approaches are recommended for deployments into the immutable infrastructure.

Failure management

> "Everything fails, all the time"
> Werner Vogels, CTO - Amazon.com

Connected mobility platforms have been evolving over the years and the relative pace has been more than any time in past few decades. There has been an equally growing concern about vehicle safety with the technology in vehicle, the data generated and the backend supporting the platform. With the growing complexity, system failures are highly probable. While failures are difficult to predict, reliability requires that the systems are aware of the failures and react accordingly to avoid impact on availability. Building and deploying robust systems should be the target state of every automotive company.

Back up data

CMREL_12: How do your strategy and mechanisms manage failures to prevent impact on your workload?

[CMREL_BP12.1] Identify and back up all data that needs to be backed up, or reproduce the data from sources

[CMREL_BP12.2] Perform data backup automatically

While there might be *n* data sources that the connected mobility platforms have to integrate with, not all data is equally important. It's essential that data entities are classified and criticality determined. This helps with making right strategic decisions for data recovery based on RPO, that is either backing up the data or reproduce it. In some of the connected mobility use cases like emergency-call or breakdown-calls, where compliance is critical the balance has to be adequate. Classifying data also helps in determining the retention periods for various data entities, one of the primary drivers for cost optimization.

Prescriptive guidance:

Backup capabilities are available with all AWS data sources. Some services provide additional features such as point-in-time-recovery (PITR), continuous replication and cross Region copy. With

the growing number of services, managing backup configurations can add to the overhead. Tools such as AWS Backup and AWS Elastic Data Recovery can boost the productivity, availability and have a RPO in seconds.

Automotive companies generally have variations hybrid infrastructure setup with systems running on cloud and on-premises data centers. AWS services such as AWS Storage Gateway or AWS DataSync can help in these deployment modes.

[CMREL_BP12.3] Secure and encrypt backups

> "Encrypt Everything."
> Werner Vogels, CTO - Amazon.com

Safeguard the backups just as you secure the vehicle data. Securing access to the data with right level of permissions prevents tampering. Encrypt the backups so that the data cannot be accessed in case of an accidental leak.

Prescriptive guidance:

Use AWS Identity and Access Management (IAM) to create roles and policies that have least privileged access. Use AWS managed policies where applicable. As more connected vehicles are rolled out, the volume of vehicle data that is hosted in vehicle data lakes keeps growing exponentially. Considering that S3 is the most popular data store for vehicle data lakes, following the best practices in handling this data is essential.

Ensure the backups are not accidentally deleted as this could result in vehicle regulatory compliance violations. Legal hold is a feature in AWS Backup service that prevents accidental deletions. Multiple departments/ units within an organization can have their holds on the backup.

[CMREL_BP12.4] Perform periodic recovery of the data to verify backup integrity and processes

> "Hope for the best and be prepared for the worst" - Maya Angelou

With backups configured, an equal effort is needed in validating with recovery test whether the backups meet Recovery Time Objectives (RTO) and Recovery Point Objectives (RPO). This

exercise should be repeated periodically using well-defined mechanisms to ensure that the data is recovered within RTO and with expected data loss as established in RPO.

Prescriptive guidance:

Testing the restores build confidence in the system and trains teams to handle the disaster situations better. There are various features in AWS services that aid in testing the restores and assess RTO and RPO. Amazon RDS and DynamoDB have allow point-in-time recovery (PITR). AWS Elastic Disaster Recovery offers continual point-in-time recovery snapshots of Amazon EBS volumes.

Use fault isolation

The architecture should ensure that there isn't any single point of failure. Systems should be resilient enough to handle the partial or complete failure of the infrastructure stack. Fault isolated boundaries limit the effect of a failure within a workload to a limited number of components. Components outside of the boundary are unaffected by the failure. Using multiple fault isolated boundaries, you can limit the impact on your workload.

CMREL_13: How do you ensure that you don't have a single point of failure?

[CMREL_BP13.1] Deploy the workload to multiple locations

To ensure high availability and prevent a single point of failure, the services should be deployed as diverse as required. The system should rely on redundant components that are decoupled from each other.

Prescriptive guidance:

A typical connected mobility set up consists of several distributed systems which creates an ideal condition to spread out and isolate. Multi-AZ solutions are a must for most of the connected mobility use cases. Multi-region though ideal but may be cost prohibitive for wider scope so should be done selectively where required. A multi-Region approach is common for disaster recovery strategies to meet recovery objectives when one-off large-scale events occur. AWS Local Zones can be used to deploy workloads closer to vehicles for low-latency requirements.

In certain cases, vehicles can communicate with edge locations rather than the cloud regions. This deployment pattern is more relevant in case of low latency requirements. Static content

can be delivered to end user from Amazon CloudFront with millisecond level latency. Other services that also enable edge computing include AWS Global Accelerator, Amazon API Gateway, and Lambda@Edge. Amazon CloudFront can scale automatically to deliver software with over-the-air (OTA) updates at scale with high transfer rates.

Evaluate AWS Outposts for your workload. If your workload requires low latency to your on-premises data center or has local data processing requirements such as in case of plant systems. Then run AWS infrastructure and services on premises using AWS Outposts.

Key AWS services

Refer to Appendix A: Designed-For Availability for Select AWS Services in the Reliability Pillar whitepaper.

Resources

Change management

Hands-on labs:

- One Observability Workshop
- Viewing Amazon CloudWatch metrics with Amazon Managed Service for Prometheus and Amazon Managed Grafana

Reference architecture:

- Guidance for Deep Application Observability on AWS

Videos:

- AWS Summit SF 2022 - Full-stack observability and application monitoring with AWS
- AWS Cloud Operations - How to
- Publishing custom metric
- Comprehensive observability for Amazon EKS

Monitoring tools:

- AWS Observability Services

- AWS observability repositories

Workload load and stress testing tools:

- Taurus
- Apache JMeter

Blogs:

- Ensure Optimal Application Performance with Distributed Load Testing on AWS

Guides:

- Amazon EC2 Auto Scaling now gives recommendations about activating predictive scaling policy
- Step and simple scaling policies for Amazon EC2 Auto Scaling
- Target tracking scaling policies for Amazon EC2 Auto Scaling
- Scheduled scaling for Amazon EC2 Auto Scaling
- AWS services that you can use with Application Auto Scaling
- Using AWS Config managed rules with Audit Manager
- Using AWS Config custom rules with Audit Manager
- Troubleshooting AWS Config integration with Audit Manager
- Distributed Load Testing on AWS

Amazon Builders' Library:

- Ensuring rollback safety during deployments

Failure management

Reference architecture:

- Data Protection Reference Architecture with AWS Backup

Blogs:

- Best practices for data lake protection with AWS Backup

Guides:

- Automating backups with Backup plan
- Point-in-time recovery for DynamoDB
- Feature availability by resource
- Feature availability by AWS Region
- Security best practices in IAM
- Legal hold
- AWS Backup Vault Lock
- Audit backups and create reports with AWS Backup Audit Manager
- What Is AWS Backup?
- Data protection in AWS Backup
- Encryption for backups in AWS Backup
- Reliability Pillar: AWS Well-Architected

Infrastructure map:

- AWS Global Infrastructure

Videos:

- AWS Summit ANZ 2021 - Everything fails, all the time: Designing for resilience

Whitepapers:

- AWS Fault Isolation Boundaries
- Choose the right Amazon RDS deployment option: Single-AZ instance, Multi-AZ instance, or Multi-AZ database cluster
- AWS Auto Scaling: How Scaling Plans Work
- Implementing Microservices on AWS

- Disaster Recovery of Workloads on AWS: Recovery in the Cloud

Workshops:

- AWS Well-Architected Reliability Labs
- Advanced Multi-AZ Resilience Patterns
- Level 200: Testing Backup and Restore of Data

AWS Builders' Library:

- The Amazon Builders' Library: How Amazon builds and operates software

Books:

- Robert S. Hammer, *Patterns for Fault Tolerant Software*
- Andrew Tanenbaum and Marten van Steen, *Distributed Systems: Principles and Paradigms*

Performance efficiency pillar

The performance efficiency pillar for connected mobility provides guidance and best practices for customers to meet the needs of delivering low latency workloads that are scalable across multiple geographies in which they operate. Considering the needs of different workloads based on specific use-cases, this pillar focuses on the key aspects that should be considered during the design of the overall architecture: customer-defined SLAs and KPIs measured by the operations phase to meet business needs, as well as providing a unified, connected mobility experience to vehicle users that can handle failovers and still allow them to commute to their destinations safely.

Design principles

Core capabilities

- Vehicle provisioning
- Command and control functions
- Telemetry
- Over-the-air updates (OTA)
- Mobile applications – Remote operations (for example, remote start)

Edge processing and vehicle shadow service architecture: The fundamental idea behind this principal is that the automotive industry is transforming itself to provide a more software centric mobility experience to their customer through applications and services for both safety and non-safety critical features. This ranges from safety features in AV/ADAS space as well non-safety critical features like personalization, content recommendation, trips, navigation, and more. This pushes the need to have more on-board vehicle compute and storage with high performance CPUs and GPUs. While this is good, there will never be enough storage and compute on-board vehicles to meet growing industry needs to provide these experiences and match them with the necessary safety standards. Therefore, the need exists to have more hybrid architecture and services in place to offload portions of non-safety critical workloads to edge locations closer to the vehicles to meet latency requirements.

Design for failover: Failures in connected mobility systems are inevitable especially when vehicles are traversing in suburbs and terrains where connectivity is spotty and is critically impacted. Working through all possible "what-if" scenarios during design phase and also having test cases to validate possible scenarios is a critical aspect of building these systems. Apart from connectivity, it's also important to build resilient and highly-available cloud systems with a fallback system in place for worst case scenarios.

Auto scaling everywhere: Running and maintaining a connected mobility application can be expensive, especially when adhering to specific SLAs and KPIs that cannot be compromised. By using the appropriate services on AWS and applying best practices and recommendations, customers can handle variable vehicle-cloud data volume spikes and high vehicle density in specific geographies. This ensures that systems are automatically scalable on a needed basis and are not running at full capacity at all times, especially during low-usage periods.

Multiple connectivity channels: Vehicle-cloud connectivity is an integral part of providing a good connected mobility experience. Relying on a single connectivity channel is error prone due issues like coverage, quality of service (QoS) and bandwidth capacity and availability. Therefore, it is recommended to explore more than one connectivity channel not only as a fallback mechanism in case of fail-over, but also as a means to use different channels for different workload needs. Options can vary, but could include using secure Wi-Fi for large OTA upgrades to save data costs, using 5G for high bandwidth, low latency connectivity needs for edge computer-vision or AI/ML workloads, or using satellite for vehicle-cloud connectivity for high bandwidth needs where latency is not critical and 5G connectivity is unreliable.

Every workload is different: Connected mobility workloads have their own set of challenges ranging from collecting data when the vehicle is in motion, processing them and offloading them.

It includes latency and bandwidth requirements for data transfer, and its own SLAs and KPIs that must be met to function optimally. This is especially important for safety critical features which cannot be offloaded to edge locations but still rely on cloud connectivity for some operations. So rather using a modular architecture approach for all use cases, using these as guiding principles to design what would be the optimal location for workload execution (edge or region) and tradeoffs can be made keeping the user safety in mind.

Measuring and improving with growth: Having mechanisms in place where critical workloads are monitored for specific KPI's for performance and ensuring that it meets the desired SLAs is critical. Even with features like auto-scalable systems, as more and more vehicles are on-boarded to connected mobility systems it's inevitable that feature level and overall system performance will degrade over time. Therefore, understanding tradeoffs in latency and cost in order to support continuously evolving systems, architectures and mechanisms are needed in order to support critical workloads wherever the vehicle is operating.

Regulatory compliance tradeoffs: Adhering to local laws and regulatory compliance is mandatory. Care must be taken to assess and consider any impact on performance while adhering to them. Based on data processing compliance requirements design your workload architecture to process the data closer to the edge whenever possible can offset some of the performance concerns.

Best practices

Topics

- Selection
- Review
- Monitoring
- Tradeoffs

Selection

CMPERF_1: Have you identified the critical components of the architecture, performance data collection to determine the best performing architecture?

The connected mobility ecosystem falls under multiple areas based on the type of use cases. Architecture needs to account for vehicle, cloud, and communication between the vehicle and

cloud. Break down your application into smaller, independent microservices that can be managed and scaled independently as needed. By treating data as a product, package it at the device (vehicle) by categorizing the data, before sending it to the cloud. Classify the data based on frequency (for example, every 60 seconds), events, on-demand (for example, during a failure event, send diagnostic logs). Architect your data processing platform by decoupling the data collection from processing and offloading to cloud. In cloud similar architecture approach needs to be in place to process data, before sending them to downstream application consumption.

Additional aspects to take into account on data:

- Timeliness of the data by minimizing latency (for example, 200 milliseconds or less).
- Data as current as possible (close to real time).
  - Primary goal – To support internal applications (for example, last location of vehicle, and fuel information.)
  - Vehicle state service – Last known values per vehicle
- Business intelligence data
  - Primary goal – To support the business for understanding the features use (for example, infotainment, driver usage patterns, and environmental data)

When developing a connected mobility platform on AWS focusing on performance efficiency, there are essential architectural components and methods for performance data collection that should be identified and optimized. Let's break this down:

[CMPERF_BP1.1] Confirm critical components of the connected vehicle platform

The critical components of a connected vehicle platform into the simplified structure of data plane, control plane, and consuming applications:

- **Data plane:** This is concerned with the movement of data throughout the system.
  - **In-vehicle data plane:**
    - **Data ingestion:** The vehicle collects real-time data from its sensors and systems.
    - **Edge processing:** The vehicle's onboard systems process this data locally, especially vital in areas with spotty connectivity. This can involve analyzing sensor data, determining the current vehicle status, or even making immediate decisions like automatic braking in emergencies.
  - **Cloud data plane:**

    - **Data ingestion:** This is where real-time data from vehicles is received, ingested, and stored in the cloud for further processing or analysis.
- **Control plane:** This component manages and coordinates the operations of the system, including ensuring the system's proper function and orchestrating event-driven processes.
    - **In-vehicle control plane:**
        - Directly interacts with the vehicle's hardware and systems. It takes action based on the processed data, such as adjusting vehicle settings or activating certain features.
    - **Cloud control plane:**
        - **Compute:** This involves running code in response to specific triggers, like changes in data or custom events. With a serverless architecture, this can be event-driven, scaling automatically based on demand.
- **Consuming applications:** These are interfaces and applications that end users (drivers, passengers, or vehicle owners) interact with.
    - **Mobile applications:** Drivers and users can access mobile apps to control and monitor their vehicles remotely. This includes functions like starting the vehicle remotely, locking/unlocking doors, checking oil levels, and scheduling starts to pre-warm or cool the vehicle.

[CMPERF_BP1.2] Enforce performance data collection

To determine the best performing architecture, it's vital to continuously collect, monitor, and analyze performance data.

- Collects and tracks metrics, collects and monitors log files, and sets alarms. It can be used to collect and track custom metrics.
- Get insights into the behavior of your applications, helping to understand how they are performing and where bottlenecks are occurring.
- Use cloud-native tools to review the resources health, performance, and optimizations recommendations.

> **Note**
>
> To derive cost per vehicle = Total cloud cost / Number of vehicles connecting to the platform.

CMPERF_2: Have you conducted performance tests for vehicle-to-connected mobility platform communication across various connectivity methods, including LTE, Wi-Fi, satellite, and scenarios with no connection?

The performance of vehicle-to-connected mobility platform communication across different connectivity methods is crucial for the seamless functioning of connected vehicles.

Steps to test performance across various connectivity methods:

[CMPERF_BP2.1] Define key metrics

Decide what you'll measure, including latency, bandwidth, data consistency, and uptime, across different connectivity methods.

Simulate environments:

- **LTE:** Mimic a mobile LTE connection with varying signal strengths.
- **Wi-Fi:** Test under various conditions, for example, stable home networks, and public Wi-Fi with many connected devices.
- **Satellite:** If possible, conduct tests in environments where only satellite connectivity is available, such as remote or maritime areas.
- **No connection:** Simulate scenarios where a vehicle loses connection. Monitor how much data is stored locally and then synced when connectivity is restored.

> **Caution**
>
> There will be storage limitations on the vehicle side and it is critical to implement logic to persist critical data versus non-critical data.

[CMPERF_BP2.2] Check data consistency

Ensure that data is consistent across various connection states, especially when transitioning between them.

[CMPERF_BP2.3] Evaluate fail-over and redundancy. If one connection method fails, does the system seamlessly switch to another available method?

A successful connected vehicle platform not only requires solid performance under various connectivity conditions but also robust security, redundancy, and fail-over mechanisms.

CMPERF_3: Have you evaluated the performance of the connected mobility platform to ensure it can accommodate future vehicle demand projections?

When designing a connected mobility platform, especially for performance efficiency, it's crucial to evaluate its performance to ensure it can accommodate future vehicle demand projections. When you are testing the performance and scalability of the platform, keep the current capacity as 70% utilized, and 30% of capacity available for future needs (for new vehicles load including data). Based on your vehicle demand projections, forecast 30% of the infrastructure capacity is available with your vendors (cloud, marketplace, licensing) for your workloads

[CMPERF_BP3.1] Evaluate performance and scalability

- **Assess current performance:** Determine the platform's current performance metrics, including throughput, latency, error rates, and capacity.
- **Load, stress, and scalability testing:** Simulate the expected number of vehicles and users to test how the system behaves under anticipated loads. This will help identify any bottlenecks or performance issues. Push the system beyond its expected limits. This helps to understand the system's breaking point and ensures that it fails gracefully. Determine if the system can scale out (add more instances) or scale up (add more resources to an instance) to meet increased demand. This is crucial for accommodating future growth.
- **Performance monitoring:** Continuously monitor system performance in real-time to identify and address issues before they impact users.
- **Future demand projections:** Use data analytics to project future vehicle demands based on current trends, market research, and other relevant factors.
- **Infrastructure evaluation:** Ensure that the underlying infrastructure (servers, databases, networks) can support the projected increase in demand.
- **Database optimization:** As the number of vehicles and data points increase, database performance becomes critical. Regularly optimize queries, indexes, and storage solutions.
- **Geographical distribution:** If the platform serves a global audience, consider distributing data centers geographically to reduce latency and improve performance for users worldwide.

- **Redundancy and failover:** Ensure that there are backup systems in place to take over if the primary system fails. This is crucial for maintaining uptime during unexpected events.
- **Feedback loop:** Regularly gather feedback from users and stakeholders to understand any performance-related issues they face and address them proactively.
- **Threshold limits:** In addition to load test, scalability, performance it is critical to understand the limits on the cloud side especially on compute resources, load balancer limits, data transfer across regions / 3rd party providers. It's good practice to keep an eye on service limits to make sure monitoring is in place to alert and take proactive action.

CMPERF_4: Have you assessed the performance of the connected mobility platform when integrated with third-party vendor solutions?

Connected mobility platforms when integrated with third-party vendor solutions evaluate and select a high-performing architecture that aligns with your organization's goals for efficiency.

[CMPERF_BP4.1] Implement steps for performance assessment

- Map integration: Identify data flows, dependencies, and potential bottlenecks.
- Baseline metrics: Establish key performance metrics like latency, throughput, response time and error rates.
- Testing: Perform load and stress tests to simulate real-world traffic.
- Dependency checks: Assess software and hardware dependencies.
- Data integrity: Ensure data consistency and integrity between integrated systems.
- Security and compliance: Evaluate security protocols and ensure legal compliance.

[CMPERF_BP4.2] Collect key performance metrics

- Scalability: Can the system handle growth efficiently?
- Latency and throughput: Assess speed and capacity under different loads.
- Resilience: Evaluate fault tolerance and fail-over capabilities.
- Resource utilization: Monitor CPU, memory, and bandwidth usage.
- Interoperability: Assess ease of integration with other systems.
- Monitoring: Use analytics to identify bottlenecks or inefficiencies.

- Cost: Evaluate total cost of ownership (TCO).

When assessing the performance of third-party integrations on a platform like the connected mobility platform, it's essential to employ a mix of monitoring, testing, and optimization tools. AWS offers a robust suite of services that can be leveraged for these purposes.

CMPERF_5: Have you considered irregular data traffic patterns during the day or within Regions and can the system handle sporadic traffic without impacting the overall Service Level Agreement (SLA) and user or vehicle experience?

[CMPERF_BP5.1] Address irregular data traffic patterns and ensure that the system handles sporadic traffic efficiently.

These steps are crucial for a connected mobility platform or any large-scale system. AWS offers various solutions and services that can help in managing these unpredictable workloads without impacting the SLA, user, or vehicle experience.

[CMPERF_BP5.2] Take into account that mobility patterns can vary significantly.

When selecting an architecture, some factors to consider are peak commuting hours, weekends, holidays, climatic conditions (summer versus winter), or even unexpected events such as accidents and natural disasters.

- Dynamic scaling: Your system architecture should be able to scale up or down based on demand to handle irregular data traffic patterns. Cloud-based solutions often provide auto-scaling features to meet this need.
- Quality of Service (QoS): Implement QoS mechanisms to prioritize essential or time-sensitive data over less critical data, ensuring that crucial functions are always available.
- Caching and data aggregation: Use caching mechanisms to store frequently used data temporarily. Data aggregation techniques can help in reducing the amount of data that needs to be transmitted, thus relieving pressure on the system during peak loads.
- Rate limiting: Consider implementing rate limiting for non-essential or batch operations. This ensures that the system remains available for real-time and high-priority tasks.
- Redundancy and fail-over: Design the architecture with redundancy in mind to handle failures without affecting the SLA. Implement failover systems that can take over in case one part of the system becomes overloaded or fails.

- Traffic analysis: Use real-time monitoring and analytics to identify traffic patterns. This analysis can help in pre-scaling resources before a predicted high-traffic event.
- SLA monitoring: Implement tools that continuously monitor the system to ensure that it meets the agreed-upon SLAs, even during times of sporadic or high traffic.
- Geographical distribution: Consider deploying resources in multiple geographic locations if you expect irregular traffic patterns across different regions.
- User experience monitoring: Keep track of metrics that directly reflect user or vehicle experience such as app load times, navigation responsiveness, and real-time updates.
- Testing: Perform extensive stress and load testing simulating irregular traffic patterns to validate that the system can meet SLAs and provide a good user/vehicle experience under these conditions.

By accounting for these factors in your architecture selection, you can better prepare the connected mobility platform to handle irregular and sporadic traffic patterns without adversely affecting performance or user experience.

CMPERF_6: Have you taken into account any data regulatory requirements for vehicles and tested that might affect performance?

[CMPERF_BP6.1] Consider data regulatory requirements

- Data privacy and protection: Regulations like the General Data Protection Regulation (GDPR) in the European Union or the California Consumer Privacy Act (CCPA) in the United States mandate strict controls over personal data. Ensuring compliance can introduce additional processing steps, like encryption or anonymization, which might impact performance.
- Data localization: Some regulations require data to be stored in the same country or region where it's generated. This can affect the architecture of your platform, potentially introducing latency if data needs to be accessed from distant locations.
- Data retention and access: Regulations might dictate how long you can store data and who can access it. Implementing these controls can introduce additional database queries or checks, impacting performance.
- Data transmission: Secure transmission of data, SSL termination and off-loading especially in real-time applications, is crucial. Implementing robust encryption protocols (for example, mTLS) can introduce latency, affecting SLAs.

- Audit and logging: To ensure compliance, you might need to maintain detailed logs of all data accesses and changes. Writing and storing these logs can have performance implications.
- Data recovery and backups: Regulations might require you to have robust data recovery solutions in place. While this might not directly impact performance, it can affect SLAs, especially in terms of data recovery times.
- Third-party integrations: If your platform integrates with third-party services, you need to ensure they are also compliant with relevant regulations. Their performance and SLAs can directly impact your platform.
- Continuous monitoring and reporting: To ensure ongoing compliance, continuous monitoring and reporting mechanisms might be needed. These systems can introduce additional loads on the platform.

[CMPERF_BP6.2] Take regulatory requirements into account as you define SLAs for a connected mobility platform

These requirements can have a direct impact on performance and the Service Level Agreements (SLAs) you set. For example, if data encryption introduces a delay, factor that into your SLA. Similarly, if data localization introduces latency, that should be considered when setting performance benchmarks.

CMPERF_7: Have you considered a vehicle simulation tool to create artificial load scenarios for better understanding and analyzing the performance metrics of the system?

[CMPERF_7.1] Simulate real-world scenarios to understand how the system will behave under various conditions

Using a vehicle simulation tool can be invaluable in this context. Using vehicle simulation tools is an excellent approach to stress-test a connected mobility platform. By simulating various load scenarios, you can evaluate how the system performs under different conditions, anticipate potential bottlenecks, and optimize accordingly.

- Predicting system behavior: By simulating different vehicle behaviors and traffic scenarios, you can predict how the system will respond under different conditions. This can help in identifying potential bottlenecks or performance issues before they occur in a real-world scenario.

- Scalability testing: Connected mobility platforms need to handle a large number of vehicles, especially in urban environments. Using a simulation tool, you can artificially increase the number of vehicles to test the system's scalability.
- Real-time data processing: Many connected vehicle applications require real-time data processing, such as collision avoidance or traffic rerouting. By simulating these scenarios, you can ensure that the system can handle real-time data processing efficiently.
- Network load testing: Connected vehicles will continuously send and receive data. Simulating this can help in understanding the network load and ensuring that the infrastructure can handle it.
- Understanding edge cases: While typical scenarios are essential, it's also crucial to understand how the system behaves under edge cases. For instance, what happens if a large number of vehicles suddenly go offline? Or if there's a sudden surge in data transmission?
- Improving SLAs: By understanding the system's behavior under various scenarios, you can set more accurate Service Level Agreements (SLAs) and ensure that they are met.
- Cost optimization: Running simulations can also help in understanding the cost implications. For instance, if you're using cloud services, understanding the data transmission rates can help in predicting costs and optimizing them.

[CMPERF_BP7.2] Build your vehicle simulation tool to be compatible with AWS

When selecting a vehicle simulation tool, ensure its compatible with AWS, or consider building a custom simulation environment that uses the full power of AWS services.

Whatever your choice, AWS provides a robust suite of tools and services that can be harnessed to understand and optimize the performance metrics of the connected mobility platform under various simulated load scenarios.

CMPERF_8: Have you considered an edge inference simulation solution to continuously test infer or predict based on the past data?

Edge inference is a crucial concept, especially for applications where real-time data processing is essential, such as autonomous vehicles, industrial IoT, and smart cities. By running inference at the edge (closer to where data is generated), you can reduce latency, save bandwidth, and respond quickly to local changes. Continuously testing the ability of edge devices to infer or predict based on past data is essential to maintain system reliability and performance.

[CMPERF_BP8.1] Edge inference and simulation play a pivotal role in the realm of connected mobility platforms

This is especially important when focusing on performance efficiency. Edge computing allows data processing to occur closer to the data source, such as a vehicle, rather than sending it to a centralized cloud server. This approach can significantly enhance the performance and responsiveness of the system.

- Latency reduction: One of the primary benefits of edge inference is the drastic reduction in latency. By processing data on the edge (for example, onboard a vehicle or a nearby edge server), the need to transmit data to a central server and wait for a response is eliminated. This is crucial for real-time applications like autonomous driving or collision avoidance.
- Bandwidth efficiency: Transmitting large volumes of raw data from vehicles to central servers can be bandwidth-intensive. Edge inference allows for initial processing and filtering of data, sending only essential information to the central server, thus saving bandwidth.
- Continuous testing and learning: An edge inference simulation solution can continuously test and adapt based on past data. This iterative process can help in refining algorithms and improving prediction accuracy over time.
- Real-world scenarios: Edge simulation tools can recreate real-world driving scenarios, allowing the system to test and infer based on historical data. This can be invaluable in understanding how the system would react in specific situations.
- Data privacy and regulatory compliance: Processing data on the edge can also address data privacy concerns. By analyzing and making decisions locally, sensitive data might not need to be transmitted or stored centrally, aiding in compliance with data protection regulations.
- Scalability: As the number of connected vehicles grows, central servers might become overwhelmed with data. Edge computing distributes the processing load, ensuring that the system remains scalable.
- Resilience and reliability: Edge inference provides a decentralized approach, ensuring that even if there's a failure in one part of the system, other parts can continue to operate independently.
- Integration with other systems: Edge devices can integrate with other local systems, such as traffic light controllers or local weather stations, to make more informed decisions.

[CMPERF_BP8.2] Your edge inference simulation solution should provide a versatile approach to address many variables.

When considering an edge inference simulation solution for a connected mobility platform, it's essential to select a solution that can:

- Simulate various real-world scenarios.
- Continuously learn and adapt based on past data.
- Integrate seamlessly with the vehicle's onboard systems and other edge devices.
- Ensure data security and privacy.
- Simulate the expected volume and variety of data at the edge.
- Test the machine learning models' accuracy and speed under various conditions.
- Measure the latency and reliability of communication between edge devices and the central cloud or data centers.

In conclusion, edge inference simulation is a forward-thinking approach that can significantly enhance the performance efficiency of a connected mobility platform. It allows for real-time processing, continuous learning, and adaptation, ensuring that the platform remains responsive, accurate, and efficient.

CMPERF_9: Have you identified an appropriate communication protocol based on use case?

[CMPERF_BP9.1] For vehicle-cloud communications, different protocols must address your specific use case.

- Pub-Sub protocols such as MQTT and AMQP can be used to exchange telemetry messages, command-control functions.
- HTTP or gRPC based protocol can be used to deliver over the air (OTA) updates.
- WebRTC protocol (RTMP) can be used to deliver video stream with real-time latency.
- SMS can be used for device wake-up, and run simple commands.

  The appropriate communication protocol often depends on the specific requirements of the use case. Here's a brief overview of common communication protocols and the typical use cases they serve:

- HTTP/HTTPS:

  - Use cases: Web applications, mobile applications, and many modern internet-based applications.
  - Advantages: Well-known, widely used, and supported. Secure version (HTTPS) available.
- MQTT (Message Queuing Telemetry Transport):
  - Use cases: Internet of Things (IoT) devices, real-time analytics, mobile applications, and communication in unreliable networks.
  - Advantages: Lightweight protocol with low bandwidth requirements. Supports QoS (Quality of Service) levels.
- WebSocket:
  - Use cases: Real-time web applications, gaming, chat applications, live sports updates.
  - Advantages: Provides full-duplex communication channels over a single TCP connection. More efficient than repeatedly opening new HTTP connections.
- AMQP (Advanced Message Queuing Protocol):
  - Use cases: Message-oriented middleware, cloud services, and brokered messaging.
  - Advantages: Supports message orientation, queuing, and routing.
- Payload efficiency
  - For optimal payload efficiency in a Connected Vehicle Platform, adopt compact data formats like Google Protocol Buffers (GPB) or MessagePack, which provide efficient serialization.
  - Send only changed or relevant data, employing compression algorithms like gzip for textual data, and utilizing delta encoding can minimize the data's size.
- Consider the trade-offs between data freshness and bandwidth utilization, tailoring strategies to the specific application's needs.

> **Note**
>
> Refer to Resources for links to the mentioned protocols

[CMPERF_BP9.2] Choose a protocol based on the factors that apply to your use case

When choosing a communication protocol, it's important to assess the specific needs of the use case and test the chosen protocol in realistic scenarios to ensure it meets the requirements.

The choice of protocol will typically depend on:

- Data volume and frequency.
- Power and bandwidth constraints.
- Required communication range.
- Latency and real-time requirements.
- Security and privacy needs.

Review

CMPERF_10: Have you considered the implementation of scalable, cost-effective, and low-maintenance managed services for high-performance computing workloads to process diverse data types (such as high and low fidelity data, logs, and commands) collected from vehicles?

[CMPERF_BP10.1] Build a cost-effective solution that is scalable and yet easy to manage as more vehicles connect to the cloud.

We need to think about use cases that are critical in order to bring a good user experience to end consumers. For example, vehicle state services play a key role for consumers to know the state of vehicle, for example, door locked, or window down. In this case, data caching solutions (Redis Cache or MemoryDB) are important to quickly access last available data with low latency (200 milliseconds or less) interval. Any new data will move classify earlier as historical data, this can be stored in either No SQL database such as DynamoDB or data lake for further processing. Training can be done to improve machine learning models and later it can be deployed for prediction based on the type of use case (for example, recommend cabin temperature based on historical data in vehicle)

Recommendation - Telemetry data strategy

- Top 50 Properties – In memory cache (open standard numbers)
- Next 500 Properties – Microsecond interval
- 5000+ Properties – Seconds or higher interval

CMPERF_11: Have you tested the ability of your platform to seamlessly adopt, replace, or upgrade various compute solutions, including standalone systems, container-based architectures, and serverless technologies?

Ensuring that your platform can seamlessly adopt, replace, or upgrade various compute solutions is crucial for scalability, adaptability, and resilience.

When considering AWS services, here's how you can ensure flexibility across different compute solutions:

[CMPERF_BP11.1] Self-managed systems (Amazon EC2)

- Compute: Virtual servers in the cloud where you can run applications.
- Load balancing: Distributes incoming application traffic across multiple targets, such as EC2 instances.
- Auto scaling: Ensures that you have the right number of EC2 instances available to handle the load for your application.
- Seamless adoption and upgrades: Use Amazon Machine Images (AMIs) to create and save configurations, making it easier to scale, replace, or upgrade.

[CMREF_BP11.2] Container-based architectures

- Containerization: A highly scalable, high-performance container orchestration service that supports Docker containers.
  - Serverless compute for containers. You don't need to provision, configure, or scale clusters of virtual machines to run containers.
- Seamless adoption and upgrades: Use container orchestration to manage the lifecycle of containers, ensuring that services can be updated or rolled back without downtime.

[CMPERF_BP11.3] Serverless technologies

- Run code without provisioning or managing servers. You pay only for the compute time that you consume.
- API Gateway: For creating, deploying, and managing APIs along with serverless function to create serverless applications.

- Seamless adoption and upgrades: With serverless, deployments can be versioned, allowing for easy rollbacks.

Monitoring

CMPERF_12: Have you implemented end-to-end monitoring and logging (between edge, vehicle, and cloud) of your system along with notifications?

[CMPERF_BP12.1] Monitoring, logging, and setting up notifications are critical for maintaining the health, performance, and security of a system.

AWS offers a comprehensive suite of tools to help with these tasks:

- Ensure that all services, applications, and resources report their metrics and logs.
- Regularly review and adjust your monitoring strategy to adapt to changes in your environment and application.
- Ensure that you're not just collecting data but also deriving actionable insights from it.

[CMPERF_BP12.2] Implement device monitoring at the edge device (vehicle), data transmission monitoring, monitor cloud services, and log monitoring.

As a general practice, establish alerts to monitor different workloads, applications, database services, load balancers, and network monitoring. Notify site reliability engineering (SRE) team once a certain threshold level is breached. These actions will help to define KPIs such as round-trip time, and network latency between vehicle-cloud and internal applications.

CMPERF_13: Have you built the right dashboards and widgets for your prioritized actionable insights?

Building an effective dashboard involves focusing on the key performance indicators (KPIs) that matter most to your organization and displaying them in an easily understandable and visually appealing way.

[CMPERF_BP13.1] Follow best practices when creating dashboards

User-centric design:

- Tailor the dashboard to the needs of its primary users. Consider who will be using the dashboard and why.
- Use a clear, organized layout and meaningful naming conventions.

Prioritize key metrics:

- Show the most important data points prominently, ensuring they are easily accessible and visible.
- Avoid cluttering the dashboard with non-essential metrics.

Use appropriate visualizations:

- Choose the right type of graph or visualization based on the nature of the data. For instance, time-series data is best viewed with line charts.

Interactive elements:

- Add interactive filters, drill-down capabilities, or time-span selectors to allow users to explore the data more deeply.

Consistent refresh rates:

- Determine how frequently the dashboard needs updating. Some metrics might require real-time updates, while others might be daily or weekly.

Alerting:

- Integrate alert mechanisms, using services like Amazon SNS, to notify stakeholders of important events or thresholds.

Feedback loop:

- Regularly gather feedback from users and iterate on the dashboard design and widgets.

Tools such as dashboards, container insights, serverless insights, along with external tools such as Datadog can be used to build custom dashboards that can provide better insights. When you are running applications in containers or using serverless architectures, visibility into your workloads becomes paramount for optimizing performance, troubleshooting issues, and ensuring security.

[CMPERF_BP13.2] Use tools and best practices to gain insights

- End-to-end visibility: Ensure that you're tracking the complete lifecycle of a request or a transaction across all components of your application.
- Alerts and alarms: Set up meaningful alerts and alarms based on the metrics and traces collected.
- Anomaly detection: Anomaly detection to create alarms that watch for anomalies in your metrics.
- Regularly review metrics and logs: Periodically review the metrics and logs, even if there's no issue, to understand the standard behavior of your applications.
- Security: Always monitor for security threats, especially in serverless applications where the application's perimeter might not be as defined.
- Cost optimization: Especially for serverless architectures, where you're billed for what you use, monitoring can also help in understanding the cost patterns and optimize.

Remember, while AWS provides the tools to monitor and gain insights, it's the combination of these tools with best practices that will give you the most valuable insights into your container and serverless environments.

Tradeoffs

CMPERF_14: What criteria have you identified where tradeoffs can be made?

[CMPERF_BP14.1] Develop criteria where trade-offs can be made based on the type of functionality.

For example, infotainment functionality brings higher customer satisfaction where certain tradeoffs are acceptable (for example, retry actions are configurable to avoid draining battery). But when it comes to vehicle safety critical OTA updates are not inhibited by tradeoffs. Additional criteria can be applied:

- Ensure data collection concerning predictive maintenance are not being hindered
- Develop a revision process on actionable insights received from monitoring

[CMPERF_BP14.2] Architecture design and trade-offs

The following options have benefits which bring scalability, higher performance, faster response but the tradeoff is cost. With volume of vehicles that connect to a connected vehicle platform increases the cost to maintain them is critical in the long run. In addition, the amount of data that gets offloaded from vehicle continues to grow.

- Self-managed option
- AWS Managed Service option
- Serverless option

It is critical to identify and categorize the functions, vehicle signals and states, software updates that are needed to support the connected vehicle platform by processing and storing them.

Key AWS services

- AWS Lambda - serverless provides scalable processing
- AWS IoT Core - Connect, manage, and scale your edge device fleets easily and reliably without provisioning or managing servers.
- Amazon Data Firehose
- Amazon CloudWatch
- AWS Systems Manager
- Amazon Elastic Kubernetes Service
- Amazon DynamoDB
- Amazon Elastic Load Balancing
- AWS IoT FleetWise
- Amazon Managed Streaming for Apache Kafka (Amazon MSK)

In summary, AWS provides a comprehensive suite of services that can be tailored to the specific needs of a connected mobility platform. It's essential to continuously evaluate and adapt the architecture based on real-world performance data and changing requirements.

Resources

Documentation and blogs:

- AWS Connected Vehicle Reference Architecture
- Device Communication Protocols
- gRPC Framework
- WebSocket Protocol
- AWS IoT Core
- AWS IoT FleetWise

Whitepapers:

- Designing Next Generation Vehicle Communication with AWS IoT Core and MQTT

AWS Partner solutions:

- Commoditize connected mobility with WirelessCar on AWS

Training Materials:

- Building Connected Vehicle platforms with AWS IoT

Videos:

- Building Connected Vehicle and Mobility Platforms with AWS

Cost optimization pillar

The cost optimization pillar for connected mobility aims to help organizations optimize their AWS usage to reduce costs, improve efficiency, maximize ROI and provide the ability to run systems to deliver business value at the lowest price point. This pillar focuses on the design principles that should be considered to build efficiently connected vehicles use cases on AWS at lowest costs. The tenets behind architecting workloads with the most effective use of services and resources is to achieve business goals with maximum ROI. The cost optimization pillar provides an overview of those design principles, and best practices.

Design principles

In addition to the general Well-Architected cost optimization design principles, there are some design principles specific for cost optimization for connected mobility:

Manage cost and business value tradeoffs: Managing tradeoffs between cost and other factors in connected mobility requires a careful balance between short-term costs, such as installation, setup and subscription fees, and long-term costs, such as data storage, management and infrastructure upgrades cost, cost-benefit analysis, and risk management. For example, if you want to optimize for speed to market with low cost?

Choosing a solution solely based on cost without considering its business value can result in increased operational cost overtime. Factors such as functionality, scalability, security, speed of innovation, and going to market should be carefully evaluated along with cost to select the right services/solutions. For example, a logistics company wants to implement a connected mobility solution to track packages and optimize delivery routes. Implementing a flexible architecture that can scale to accommodate growth and demand may come with an increased short-term cost, however, it also comes with an increased business value creation which will result in cost savings in long-term.

Design the system to filter relevant data: Connected mobility applications could generate 10 exabytes of data per month. That's why it is important to identify the data to be collected throughout your vehicle's fleets based on the corresponding business use-case or security/compliance requirements. Identify opportunities to stop collecting unnecessary data and consider collecting data targeted on business case and moving to event-based collection instead of interval whenever possible.

Evaluate where to process data: Data transfer between vehicle and cloud backend incur cost. Evaluation based on the use case through several dimensions:

- The network cost private or public may have different cost implications. In the context of connected vehicles, a private network refers to a dedicated and secure communication network established specifically for the vehicles and related systems within a closed environment. This network is isolated from public networks and is designed to provide a controlled and reliable communication infrastructure for connected vehicle operations.

  For example, imagine a fleet of autonomous delivery vehicles operated by a logistics company. To ensure seamless and secure communication between these vehicles, as well as with the central control system, the company sets up a private network. This network could consist

of dedicated cellular connections, Wi-Fi hotspots, or even a custom-built communication infrastructure.

- The latency requirements to process the data. Latency refers to the delay between when data is generated or transmitted and when it is received and processed by the relevant systems. Here are two examples of latency requirements for processing data in connected vehicles:

Emergency collision avoidance:

Data source: Sensors on a vehicle detect an imminent collision with another vehicle.

Processing needs: The data from these sensors must be processed immediately to assess the risk and decide on an appropriate action (for example, applying brakes, or changing lanes).

Latency tolerance: In this scenario, extremely low latency is critical. A delay of even a few milliseconds could be the difference between a successful collision avoidance and an accident.

Traffic signal optimization:

Data source: The connected vehicle is equipped with technology to communicate with traffic signals in real-time.

Processing needs: The vehicle's system sends data to the traffic signal control system to request a green light or to adjust the signal timing for optimal traffic flow.

Latency tolerance: While not as critical as collision avoidance, low latency is still important. Delays should be minimized to ensure smooth traffic flow and reduce unnecessary stops.

In both cases, low latency is essential for the effective operation of connected vehicle systems. It allows for timely decision-making and actions, enhancing safety and efficiency on the road.

- The compute capacity requirements where processing the data makes more sense, at edge in the vehicle or in the AWS Cloud. In this scenario, edge processing (in-vehicle) is critical for immediate decision-making and ensuring passenger safety. It allows each vehicle to operate autonomously, reacting swiftly to its environment.

On the other hand, cloud processing is beneficial for higher-level, aggregate analysis. It can be used to optimize routes across the entire fleet, predict traffic patterns, and perform long-term planning. Overall, a combination of edge and cloud processing can offer the best of both worlds, allowing for real-time decision-making at the vehicle level while also leveraging the cloud's computational power for broader fleet optimization and analytics.

Best practices

There are five best practice areas for cost optimization:

Topics

Design decisions are sometimes guided by haste as opposed to empirical data, as the temptation always exists to overcompensate *just in case* rather than spend time benchmarking for the most cost-optimal deployment.

This often leads to over-provisioned and poorly optimized deployments. The following sections provide techniques and strategic guidance for the initial and ongoing cost optimization of your deployment.

Cost-effective resources

CMCOST_1: How do you optimize your raw vehicle data storage?

Managing raw vehicle data is important because it allows organizations to leverage the data to drive innovation, improve their business processes and is the original source of analytics. Vehicle data can provide valuable insights into driving patterns, vehicle performance, and other areas that can be used to optimize operations, improve safety, and enhance the customer experience, however the amount of data generated can also be a big driver for cost. Vehicular data is a key source of information that can be used to drive automotive transformation, but it can be a challenge to manage due to the volume, velocity, and variety of the data. By effectively managing the raw vehicular data, organizations can unlock the value of the data and use it to drive innovation and improve business processes.

[CMCOST_BP1.1] Store raw data in a scalable and cost-effective way

Efficient data storage can help you avoid high costs associated with storing large amounts of data. Object storage is recommended for large amounts of unstructured data, especially when durability, unlimited storage, scalability, and complex metadata management are relevant factors for overall performance.

Use object storage and evaluate the appropriate storage classes.

Raw vehicle data typically includes large volumes of data that require different levels of access speed and frequency. To optimize storage costs, you can use appropriate storage classes based on the access speed and frequency of the data. For example, you can use Amazon S3 Standard for frequently accessed data and Amazon Glacier for archived data. Amazon S3 Intelligent-Tiering can also automatically move data between storage classes based on changing access patterns.

Avoid costs by using a schema-on-read service, such as Amazon Athena, to query the data in its native form. Using Athena can help reduce the need for large-scale storage arrays or always-on databases to read raw archival data.

To optimize storage costs, you can also use data lifecycle policies to automatically move data between different storage classes based on the stage of the data. For example, you can use Amazon S3 lifecycle policies to move data from S3 Standard to Amazon Glacier for archiving after a certain period of time.

Amazon DynamoDB Accelerator (DAX) or Amazon Timestream for time-series: If you are using Amazon DynamoDB for real-time data storage, DAX can help improve read performance by caching frequently accessed data. It reduces the need to retrieve data from the database directly, thus optimizing data access. Alternatively, you can use Amazon Timestream as a fast, scalable, and serverless time-series purpose-built database for short-lived real-time time series data.

[CMCOST_BP1.2] Use data partitioning for optimize performance and scalability

Raw vehicle data can be partitioned based on time, geography, or other factors to optimize storage and retrieval costs. For example, you can partition data by time to store data generated in different time intervals in separate Amazon S3 prefixes. This allows you to scan only the relevant data during query operations, reducing the amount of data scanned and the cost of querying.

Implementing data partitioning offers several significant advantages, making it an essential approach for handling substantial volumes of data efficiently and effectively:

Splitting the raw data into smaller partitions based on relevant attributes (e.g., time, vehicle ID) can improve data retrieval and reduce costs. Additionally, compressing the data before storage can

significantly reduce storage costs and improve data transfer speed. AWS Glue or Amazon Redshift can be used for data partitioning, while Amazon S3 provides built-in data compression options.

[CMCOST_BP1.3] Choose the right services by evaluating storage characteristics and requirements for your use case.

Vehicles generate a massive amount of data, which needs to be stored for various reasons, including compliance, regulatory requirements, and future analysis. As the volume of data grows, so does the cost of storing and managing it. Cold storage, such as Amazon Glacier, is a popular option for storing data that is infrequently accessed but needs to be retained for the long term. However, due to factors like cost of retrieving the data from cold storage, the overall data retention cost can be high. Effectively managing this data is crucial for enhancing vehicle performance, improving user experiences, and enabling data-driven decision-making.

Evaluate velocity, the volume of data coming and data retention/transfer cost from vehicles when selecting storage services.

In this scenario, using a general-purpose database might be inefficient and costly. It would require constant maintenance to handle the high volume of incoming data and manage the expiration of outdated information.

- For vehicle's short living time series data consider purpose-built database service, Amazon Timestream. Imagine a fleet management system for a ride-sharing company. This system tracks the GPS coordinates, speed, and passenger information of each vehicle in real-time. However, this data is highly time-sensitive and loses its relevance after a short period, typically a few days.
- Here's where Amazon Timestream comes into play. It is optimized for precisely this type of data. It's designed to handle large volumes of time series data efficiently, automatically managing the retention of old data and ensuring high availability for querying recent information. By employing Amazon Timestream, the ride-sharing company can effectively manage their vehicle data without the overhead of handling it in a traditional database. This not only leads to cost savings but also allows for smoother and more efficient operations.
- Your vehicle data use case may experience varying data loads over time. Consider services like Amazon Aurora or Amazon Redshift for scalable and flexible database solutions that can adapt to changing demands without compromising performance.
- For data at lower scale of time, devices, or other characteristics—Consider Amazon DynamoDB or Amazon Aurora for short-term historical data. Use your data lifecycle policies to optimize what is kept in the short-term storage.

- Amazon DynamoDB is a fully managed NoSQL database service that is designed to handle low-scale to high-scale workloads. It provides fast and predictable performance with seamless scalability.

[CMCOST_BP1.4] Sanitize your data and ensure that only essential and accurate data is transferred to the cloud.

Collecting only the necessary data is important for several reasons because it helps to reduce the amount of irrelevant data that can accumulate over time, which can clutter the system and make it more difficult to extract useful insights. Data sanitizing is a critical step in ensuring that only essential and accurate information is transferred to the cloud. This process involves identifying, cleansing, and validating data before it's sent for storage or processing in the cloud.

For example, outliers in GPS coordinates that are far from regular routes might be flagged for further validation or omitted. Additionally, sensitive information like vehicle identification numbers or proprietary algorithms may be anonymized or encrypted to protect intellectual property and user privacy.

By sanitizing the data before integration with the cloud, the connected vehicle system optimizes bandwidth usage, reduces storage costs, and ensures that the information stored in the cloud is of the highest quality and security. This practice is fundamental in building a robust and efficient connected vehicle ecosystem.

Process data at edge as much as possible:

- Edge analytics and decision making: Use edge computing platforms to perform analytics and real-time decision-making directly in the vehicle. Implement AWS Lambda functions on the edge to process data locally, enabling immediate actions without incurring cloud round-trip latency.
- Prioritize critical or high-priority data for immediate transmission to the cloud, while less time-sensitive data can be sent during periods of lower network activity. This strategy optimizes data transfer efficiency and reduces cloud processing costs.
- Priority data (high sensitivity):
  - Scenario: A vehicle detects an imminent collision with an obstacle or another vehicle.
  - Response: This data is considered critical and requires immediate transmission to the cloud. The system prioritizes it for real-time analysis and action, ensuring the safety of the vehicle and its surroundings.
- Routine telemetry (lower sensitivity):

- Scenario: The vehicle's telemetry data, including GPS location, fuel levels, and tire pressure, is collected for regular maintenance and analysis.
- Response: This data, while important, isn't as time-sensitive. It can be queued for transmission during periods of lower network activity, such as during off-peak hours or when the vehicle is in a low-traffic area.

Data filtering and pre-processing:

- Implement data filtering and pre-processing directly in the vehicle's onboard systems. Use edge computing solutions to analyze and filter out irrelevant or redundant data at the source. This reduces the data volume that needs to be sent to the cloud, leading to cost savings on data transfer fees.
- Aggregate data locally in the vehicle to reduce the number of individual data packets sent to the cloud. Apply data compression techniques to further optimize data transfer, minimizing data transmission costs.

Use cost calculators to model different approaches for message size and count.

- The AWS Pricing Calculator can estimate costs for specific message sizes, traffic, and operations.

CMCOST_2: How do you optimize your network consumption and interactions between vehicles and cloud?

With an expansive amount of data collected from vehicles, optimizing network consumption and payload size is essential in to ensure efficient and effective data communication between vehicles and the cloud. With limited bandwidth and the need for real-time data processing, optimizing network consumption reduces data usage and reduces costs. Lightweight and efficient protocols such MQTT, data filtering, compression, caching, quality of service, and optimized routing is some of the best practices that can be employed.

- MQTT (Message Queuing Telemetry Transport): MQTT is a lightweight and efficient messaging protocol designed for devices with limited processing power or bandwidth. It's based on a publish-subscribe model, allowing efficient communication between devices and the server.

For example, in a connected vehicle system, if various sensors need to send data to the cloud, MQTT can be employed. It minimizes overhead and ensures reliable communication.

- Data filtering: Data filtering involves the process of selectively extracting relevant information from a larger dataset based on predefined criteria. This reduces the volume of data that needs to be transmitted.

  For example, in a fleet of vehicles, not all sensor data may be equally important. Filtering out non-critical data ensures that only essential information is transmitted, saving bandwidth.
- Compression: Compression reduces the size of data before transmission, decreasing the amount of bandwidth required. This is particularly useful for transmitting large datasets efficiently.

  For example, in connected vehicles, images or video feeds from cameras can be compressed before sending them to the cloud, reducing the amount of data that needs to be transmitted.
- Caching: Caching involves storing frequently accessed data locally on the device or in a nearby server. This minimizes the need for repeated requests to the cloud, reducing latency and bandwidth usage.

  For example, in a connected vehicle, frequently requested map data or software updates can be cached locally, reducing the need for continual downloads.
- Quality of Service (QoS): QoS defines the level of service reliability and delivery assurance during data transmission. It ensures that data is delivered accurately and reliably.

  For example, in a connected vehicle scenario, a high QoS level might be assigned to safety-critical data like collision warnings, while less critical telemetry data might have a lower QoS requirement.
- Optimized routing: Optimized routing involves selecting the most efficient path for data transmission to minimize delays and reduce congestion on the network.

  For example, in a fleet of vehicles, data can be routed through the most stable and low-latency network connections available, ensuring timely delivery.

By implementing these practices, connected vehicle solutions can operate seamlessly and effectively, improving overall efficiency.

[CMCOST_BP2.1] Compress and aggregate data whenever possible to reduce the amount of data that needs to be transmitted over the network.

Data filtering and aggregation:

- Implement data filtering and aggregation logic at the edge to send only relevant and summarized data to the cloud. Use AWS IoT Core rules engine to perform data transformations and filtering before transmitting data to the target systems for storage or consumption by microservice.
- A common way to achieve better data transmission efficiency is by combining a series of measurements into a single message enabling also more efficient compression as the volume of the message increases. You can leverage AWS IoT GreenGrass v2 components to implement your aggregation function at edge.

Compress data whenever possible to reduce size of data transmitted.

- Techniques such as `gzip` compression can reduce significantly the size of data being sent. You can use AWS IoT GreenGrass v2 components to implement your compression function at edge.
- You can also use Protocol Buffers (protobuf – binary format) that provides an efficient structured compressing mechanism. You can use AWS IoT Core and AWS Lambda to ingest and process Protobuf for consumption.
- If using AWS IoT FleetWise for data collection you can configure your campaign to compress signals before transmitting data using SNAPPY.

[CMCOST_BP2.2] Adjust collection frequency depending on the context.

Evaluate and adjust the frequency of data collection depending on functional and business need. By adjusting the collection frequency of data from a connected vehicle based on the context, you can minimize unnecessary data transmission and optimize cloud resources. This approach helps to ensure that the most relevant and critical data is delivered to the AWS Cloud while reducing data transfer costs and improving overall data efficiency.

- Adjust frequency as needed based on events or context in the vehicle such as increased frequency telemetry when operating in autonomous mode. You can configure Rules-based collection campaign in AWS IoT FleetWise or develop an AWS IoT GreenGrass v2 component that dynamically adapt collection frequency depending on an event in the vehicle. Similarly, you can leverage same event-based collection schemes to send to the cloud telemetry only if an event happens or a rule is matched such collecting weather telemetry only if engine overheat.

- Define threshold values for each event or context parameter that determine when the data collection frequency should be adjusted. Set triggers to respond to changes in the context, such as exceeding a certain speed limit or encountering specific driving conditions.

[CMCOST_BP2.3] Choose the right communication service and configuration depending on the use case.

Use MQTT 5 protocol properties to optimize bandwidth.

- MQTT Protocol for Lightweight Communication: Use the lightweight MQTT (Message Queuing Telemetry Transport) protocol for communication between vehicles and AWS IoT Core. MQTT is efficient in terms of bandwidth and is well-suited for IoT applications.
- You can use MQTT 5 properties to further optimize the bandwidth between vehicle and backend. You can set "Message Expiry Interval" to not hold messages indefinitely on the client when client disconnects, this should be done in particular for messages that have a performance expectations like Remote Operations that should be very short lived.
- AWS IoT Core device shadow: Use the AWS IoT Core device shadow to store and synchronize the current state of vehicles with the cloud (fleet management systems or user mobile devices). This enables vehicles to retrieve the latest desired state from the cloud without continual communication, reducing network consumption.

Optimize routing and use caching mechanisms:

- **Caching:** Caching involves storing frequently accessed data locally on the device or in a nearby server. This minimizes the need for repeated requests to the cloud, reducing latency and bandwidth usage. Example: In a connected vehicle, frequently requested map data or software updates can be cached locally, reducing the need for continual downloads.
- **Quality of Service (QoS):** QoS defines the level of service reliability and delivery assurance during data transmission. It ensures that data is delivered accurately and reliably. Example: In a connected vehicle scenario, a high QoS level might be assigned to safety- critical data like collision warnings, while less critical telemetry data might have a lower QoS requirement.
- **Optimized routing:** Optimized routing involves selecting the most efficient path for data transmission to minimize delays and reduce congestion on the network. Example: In a fleet of vehicles, data can be routed through the most stable and low-latency network connections available, ensuring timely delivery.

Expenditure and usage awareness

CMCOST_3: How do you select the compute and storage solution for your vehicle data?

The right storage and compute solution for vehicular data depends on a variety of factors including the volume of data being generated, the speed at which it needs to be processed, its computation requirements, the security requirements, and the budget available for infrastructure. These factors are essential to ensure that the solution chosen can handle the amount of data generated by vehicular systems, process the data in real-time, store the data securely, and deliver insights that can be used to optimize vehicular operations.

[CMCOST_BP3.1] Analyze the data volume and evaluate the processing needs to save on computation costs

Improve the ROI by using serverless architecture.

- Implement serverless computing to minimize infrastructure costs and focus on developing innovative services. With AWS Lambda, you can run code without the need to manage servers, which makes it ideal for processing data from connected vehicles. Serverless architecture can improve return on investment (ROI) when it comes to vehicular data by reducing infrastructure costs, optimizing resource utilization, and enabling faster application development.
- Serverless function for real-time aggregation: Use AWS Lambda or a similar serverless function to aggregate trip data in real-time. This function should collect and process data from multiple sensors and sources during a trip.
- Stream processing for efficiency: Use Amazon Kinesis or similar stream processing services to handle data streams efficiently. This ensures that data is processed as it arrives, reducing latency and improving responsiveness.
- Data validation and error handling: Implement data validation checks within the serverless function to ensure the integrity of aggregated trip data. Handle any errors or exceptions gracefully.
- Real-time score calculation: Implement a serverless function that calculates driver scores based on aggregated trip data. This function should factor in various parameters like speed, acceleration, braking, and adherence to traffic rules.
- By implementing these strategies, you can harness the power of serverless architecture to efficiently aggregate trip data and calculate driver scores. This not only improves the overall

efficiency of the system but also contributes to increased ROI by promoting safer and more cost-effective driving behaviors. Velocity

- Cost efficiency: Evaluate the cost of the compute and storage services in relation to your budget and expected data workload. Consider services with pay-as-you-go pricing and cost optimization features like AWS Cost Explorer.
- Implement storage lifecycle policies to optimize cost and utilize Amazon S3 Intelligent-Tiering.
- Define data archiving and lifecycle policies to automatically move less frequently accessed data to cost-effective storage tiers like Amazon Glacier or Amazon S3 Intelligent-Tiering.

[CMCOST_BP3.2] Use data analytics to analyze vehicular data and develop new services with minimum investment.

Assess the volume of data generated by your vehicles and the velocity at which the data is produced. High-velocity data might require solutions with low latency and high throughput, while large volumes of data might demand scalable storage options.

Data volume and velocity:

- Analyze the data from connected vehicles to gain insights and develop new services. Analyzing vehicular data can help in identifying patterns and trends that can be used to improve existing services and prevent the need for reinventing the wheel.
- Data filtering and prioritization:
  - Guidance: Implement filters at the edge to capture only essential data, prioritizing critical information for immediate transmission.
  - Example: In a connected vehicle, prioritize safety-critical events like collision alerts over less critical data like routine diagnostics.
- Optimize data transmission protocols (for example, MQTT):
  - Guidance: Choose lightweight, efficient protocols like MQTT for communication. It minimizes overhead and is ideal for low-bandwidth environments.
  - Example: Use MQTT to transmit aggregated sensor data from a vehicle to the cloud with minimal packet overhead.
- Use of data compression techniques:
  - Guidance: Implement data compression algorithms to reduce the size of transmitted data.
  - Example: Compress image or video data from vehicle cameras before transmission, reducing the bandwidth required.

- Optimize frequency of telemetry updates:
  - Guidance: Adjust the frequency of telemetry updates based on need. Reduce update rates for less time-sensitive data.
  - Example: Decrease the update frequency for components with stable readings, like tire pressure, to conserve bandwidth.
- Implement data retention policies:
  - Guidance: Define policies for data retention. Store only relevant data and set expiration rules to manage storage costs.
  - Example: Store high-resolution telemetry data for a limited period and transition to lower-resolution data for historical analysis.
- By implementing these strategies, you can efficiently manage the volume and velocity of connected vehicle data, ensuring that only relevant, timely information is transmitted to the cloud. This not only optimizes costs but also enhances the overall performance of the IoT application.

[CMCOST_BP3.3] Integrating with existing infrastructure cost efficiently.

- Cost efficiency: Evaluate the cost of the compute and storage solutions in relation to your budget and expected data workload. Consider services that provide pay-as-you-go pricing and allow you to optimize costs based on actual usage.
- Use existing APIs and protocols:
  - Guidance: Use standard APIs and protocols for integration with existing systems. This minimizes the need for custom development.
  - Example: Integrate connected vehicle data using RESTful APIs, MQTT, or OPC UA, depending on compatibility with existing infrastructure.
- Implement edge computing for local processing:
  - Guidance: Use edge computing to process data locally before integration with existing systems. This reduces the load on centralized servers.
  - Example: Employ edge devices like IoT gateways to pre-process data from connected vehicles before sending it to the central system.
- Implement data transformation layers:
  - Guidance: Introduce data transformation layers to convert data formats between connected vehicle systems and existing infrastructure.

    - Example: Use AWS Lambda functions to transform and map incoming data from vehicles to match the format expected by the existing systems.
- Leverage message brokers for integration:
    - Guidance: Implement message brokers such as Apache Kafka or Amazon Managed Streaming for Apache Kafka for seamless integration with existing systems.
    - Example: Use Apache Kafka to buffer and process data streams from connected vehicles before ingestion into on-premises databases.
- Use standard data formats (for example, JSON or XML):
    - Guidance: Ensure that data exchanged between connected vehicles and existing infrastructure uses standard formats to ease integration. For example, convert vehicle telemetry data to JSON format before passing it to legacy systems that understand this format.

By following these strategies, you can efficiently integrate connected vehicles with existing infrastructure, ensuring seamless data flow while optimizing costs associated with integration efforts.

This involves fine-tuning resource allocation, enhancing scalability, and capitalizing on managed services to steer clear of over-provisioning.

- Managed services: Use AWS managed services like AWS IoT Core for device management, AWS DynamoDB for NoSQL database needs, and Amazon S3 for scalable and cost-effective object storage. These services reduce the overhead of managing infrastructure and are often more cost-efficient than self-managed solutions.
- Implement automatic scaling for compute resources like EC2 instances to dynamically adjust capacity based on demand: Automatic scaling helps ensure that you have the right number of resources at any given time, optimizing costs by only paying for what you use.
- Reserved Instances or Savings Plans: If you have predictable workloads, consider purchasing AWS Reserved Instances or Savings Plans. These offer upfront cost savings and discounted pricing compared to On-Demand Instances.
- AWS cost management tools: Set up AWS Budgets and cost alarms to receive notifications when your spending exceeds predefined thresholds, helping you maintain better control over costs.

Matching supply and demand

CMCOST_4: How do you optimize cost by matching backend resources with demand?

Connected vehicle solutions generate a massive amount of data and events, which can quickly become overwhelming for traditional, request-response-based architectures. Event-driven architectures allow for a more scalable solution, as events are processed asynchronously, allowing for faster and more efficient handling of large volumes of events and consume resources only when and if events occur.

[CMCOST_BP4.1] Implement event driven architectures for your backend systems.

- Use AWS managed services such Amazon EventBridge or Amazon Managed Streaming for Apache Kafka (MSK) for event bus, AWS Lambda or AWS Fargate for event processing, and AWS Step Functions for orchestrating the business. The combination of event driven architectures and AWS managed services helps you consume resources only when an event occurs and for the time processing the event while decoupling the different components of the system.
- You can leverage event-driven architecture to process events from vehicle as illustrated in this blog Building event-driven architectures with IoT sensor data. You can also expand the event-driven pattern to other business applications as illustrated for Building an Interactive Sales Portal for Automotive Using AWS Microservice Architecture

Evaluate and select your compute type depending on events processed by your backend systems.

- Consider using AWS Lambda for events that can be processed asynchronously with one or a cascade of processing steps short lived time (up to 15 minutes) that can be orchestrated with AWS Step Functions. As an example, Lambda are suitable for an API based service such subscription management for a service. Use AWS Step Functions to coordinate serverless workflows. It helps manage and control the execution order of your Lambda functions and other services, reducing the time and resources needed for complex tasks.
- Consider using containers through Amazon Elastic Kubernetes Service (Amazon EKS) or Amazon Elastic Container Service (Amazon ECS) when you need near real time event processing of large-scale number of events or require more processing time. Both Amazon EKS and Amazon ECS offer AWS Fargate that run your containers serverless without any underlying infrastructure to maintain.

- As an example, containers are suitable for large number of MQTT messaging based events that needs near real time processing such vehicle diagnose data collection and processing after a defect being detected.
- Amazon EC2 Spot Instances: For non-critical workloads, consider using EC2 Spot Instances. Spot Instances are available at significantly lower prices compared to On-Demand instances but can be interrupted with short notice when the Spot price exceeds your bid. Use Spot Instances to handle non-time-sensitive tasks and save costs.
- AWS Cost Explorer and AWS Budgets: Utilize AWS Cost Explorer and AWS Budgets to monitor and analyze your AWS costs. These tools provide insights into spending patterns and can help you identify opportunities for optimization.
- Finally, for large batch processing *AWS Batch* that helps you to run batch computing workloads on the AWS Cloud.

Cloud Financial Management

CMCOST_5: Have you defined a tagging strategy for your connected mobility workloads?

Resource classification allows you to categorize resources, providing clarity on usage and enabling you to allocate costs accurately to different departments or projects. Targeted Cost Allocation with tags, you can pinpoint specific resources and their costs, allowing for precise billing and enabling you to identify areas where cost-saving measures can be implemented effectively.

[CMCOST_BP5.1] Define and implement an organizational tagging strategy, and require key tags.

- Engage with relevant stakeholders across line which could include line of business teams, financial and governance teams, cloud operations teams, and other stakeholders. Define the use cases needed for tagging, define required versus optional tags, discuss ways to enforce tagging, and who will own the tagging of resources. As an example, you might want to define a tag to track resources used for a specific OEM to allow you to allocate costs appropriately.
- Publish a common and consistent naming schema. This should include naming and value conventions, publishing required and optional tags, and define and publish the process for adding new tags, or modifying existing tags.
- Implement your tagging strategy:

- For manually managed resources you can use AWS Config which can be used to look for required tags, and if missing, apply them to resources using Lambda.
- When using infrastructure as code (IaC), use the CloudFormation resources tag property to define tags to add required tags on resource creation. This action helps ensure that required tags are configured before resource creation. Use AWS CloudFormation Hooks to check before resource creation, and warn or prevent resource creation when missing key tags.

- Enforcing your tagging can be done using tagging policies and service control policies (SCP) in combination.
  - Tagging policies allow you to define and standardize your tag keys, including capitalization, and what values are allowed to be used within the specific tag.
  - Service control policies allow you to block resource creation when required tags are missing.

Optimize over time

CMCOST_6: How do you optimize the payload size to reduce cost in evolving generations of connected vehicles?

[CMCOST_BP6.1] Dynamically adjust the payload capacity to accommodate changing conditions or demands

Implement dynamic payload adjustment based on network conditions and available bandwidth. For example, adaptively vary payload size based on real-time network conditions to optimize for cost and performance.

Prioritize critical data: Ensure that critical or safety-related data is prioritized in the payload, while less time-sensitive data can be sent during off-peak times or in batches. Example: In a connected vehicle, prioritize alerts for collision warnings over routine diagnostics.

CMCOST_7: How do you optimize the cost of storing the state of connected mobility application over time?

[CMCOST_BP7.1] Implement a monitoring strategy.

Continually monitor and analyze usage patterns, network traffic, and associated costs.

- Define key cost metrics: Identify critical cost metrics relevant to connected mobility, such as data transmission costs, edge processing expenses, and cloud storage charges. For example, monitor data usage per vehicle, edge processing costs, and cloud storage fees to understand cost drivers.
- Set budgets and alarms: Establish budget thresholds for each cost metric and configure alarms to notify when thresholds are approaching or exceeded. Example: Set a budget for monthly data transmission costs and configure an alert to notify when 80% of the budget is reached.
- Regularly review cost reports: Conduct regular reviews of cost reports to identify anomalies, trends, or cost spikes that may require investigation or optimization. Example: Review weekly cost reports to spot any unexpected increases in data transmission costs.
- Optimize resource usage with AWS Trusted Advisor: AWS Trusted Advisor to receive recommendations for optimizing resource usage and reducing costs. Example: Act on recommendations to modify resource configurations for improved efficiency and cost savings.
- Implement cost-effective data management practices: Apply data retention policies, archival, and deletion strategies to minimize storage costs without sacrificing critical data availability. Example: Archive historical telemetry data to lower-cost storage solutions after a specified time period.
- Track and optimize data transmission costs: Monitor data transmission costs from vehicles to the cloud and consider techniques like data aggregation, compression, and prioritization to reduce expenses. Example: Implement payload optimization techniques to minimize the amount of data transmitted, thereby reducing costs.

Key AWS services

The following services and features are important in the five areas of cost optimization previous covered in this document:

- **Cost-effective resources:** Amazon S3, Amazon Athena, Amazon DynamoDB Accelerator, AWS IoT Greengrass, AWS IoT Core, AWS IoT Greengrass, AWS IoT FleetWise, API Gateway, CloudFront are AWS services that enable you to optimize storage of raw data, optimize network consumption to improve the cost effectiveness of compute resources.
- **Expenditure and usage awareness:** AWS Cost Explorer, AWS Budgets, Anomaly Detector, AWS Cost and Usage Reports are AWS services that enable you to view and analyze your costs and usage.

- **Matching supply and demand:** Analyzing the demands of workloads over time can help in matching the supply. Quick, and AWS Cost and Usage Reports, are two services that can help you perform a visual analysis of workload demand.
- **Cloud Financial Management:** AWS Config, AWS CloudFormation resource tags, service control policies, AWS tags are AWS services that can help enable a cost-conscious culture that drives accountability across all teams and functions.
- **Optimize over time:** In AWS, you optimize over time by reviewing new services and implementing them in your workload. The AWS What's New with Cost Optimization page is a resource for learning about what is newly launched as part of AWS's Cost Optimization Pillar.

Resources

Documentation and blogs:

- Connected Mobility Solutions
- Monitoring Tools in AWS
- Trends Dashboard with AWS Cost and Usage Reports, Amazon Athena and Quick
- AWS Resource Tagging
- AWS Cloud Financial Management Blogs

Sustainability Pillar

The sustainability pillar provides guidance on how to understand and optimize the environmental impact of connected mobility workloads running at the edge or in the cloud. It advises how to quantify impacts through the workload lifecycle, and how to apply design principles that help minimize these impacts. Most of the best practices in the AWS Sustainability Pillar whitepaper are applicable for connected mobility workloads and they are referred to frequently in this document. Some prescriptive guidance is especially called out due to the nature of connected mobility workloads, where edge devices might be in motion with respect to the cloud that they are using for data processing and storage.

Design principles

- Optimize the use of hardware and software efficiency that enables connected mobility.

    - **Reduce the amount of energy or resources required to run edge services:** Reduce or remove the need for customers to upgrade edge devices. Test using device farms to understand expected impact and test with customers to understand the actual impact from using your services.
    - **Adopt more efficient cloud hardware and managed services offerings:** Sharing services across a broad customer base helps maximize resource utilization, which reduces the amount of infrastructure needed to support cloud workloads. Use managed services to help minimize your workload impact.
- Optimize data generated at edge and storage requirements at cloud.
    - Make data generated by edge devices configurable to optimize storage, retention, and transfer.
    - Do maximum processing of data at the edge, if possible, to reduce data transfer volume and unnecessary storage in the cloud.
- Optimize network supporting connected mobility for sustainability.
    - Configure when and what data moves to the cloud for processing to conserve power and network bandwidth.
    - Reduce the number of connections from the edge to the cloud.
- Align connected mobility workload with wider sustainability goal of the organization.
    - Understand how you are contributing to the sustainability goals of the organization.
    - Understand sustainability is not a one time thing, it must be reviewed against changing guidelines from within and outside the organization.

Best practices

Topics

Region selection

CMSUS_1: Are you selecting the Region to meet both your business requirements and sustainability goals?

CMSUS_BP1.1: Choose a Region based on both your business requirements and sustainability goals

Choose a Region to optimize your KPIs, including those related to performance, cost, and carbon footprint. For more details, see SUS01-BP01 in the *Sustainability Pillar whitepaper*.

Prescriptive guidance:

- An edge device might be on the move and it should connect to the closest Region to meet cost, network latency, and sustainability requirements.
- As per business and regulatory requirements, configure the critical part of the workload as active/active.
- Avoid short lived connections from the edge to avoid connection overhead of establishing a new connection.

Alignment to demand

CMSUS_2: Are you scaling the infrastructure in use at edge and cloud as per workload requirements? Do you have this process under full automation?

CMSUS_BP2.1: Scale workload infrastructure dynamically

Use elasticity of the cloud and scale your cloud and connected devices dynamically to match supply of cloud resources to demand and avoid over-provisioned capacity in your workload. For more details, see SUS02-BP01 in the *Sustainability Pillar whitepaper*.

Prescriptive guidance:

- Monitor edge capacity and use onboard capability to the fullest for data processing at the edge, if possible. Make processing at the edge automated and configurable to run only required services based on the state of the vehicle (stopped or moving), and battery conditions (for EV).

CMSUS_3: Are you defining SLAs as per sustainability goals in terms of what data gets transferred and processed in real time versus batch upload at a later period?

CMSUS_BP3.1: Align SLAs with sustainability goals

Review and optimize workload service level agreements (SLAs) based on your sustainability goals to minimize the resources required to support your workload while continuing to meet business needs. For more details, see SUS02-BP02 in the *Sustainability Pillar whitepaper*.

Prescriptive guidance:

- Configure workloads to send only the minimum and required dataset to the cloud using low bandwidth network. The rest of the data can move to the cloud when the edge device is connected with Wi-Fi.
- If it meets business requirements, get data processed at the edge itself and only send processed data to the cloud for further usage. Transform data in a human readable format at the edge itself, if possible.

Software and architecture

CMSUS_4: Are you using efficient software designs and architectures to minimize the average resources required per unit of work?

CMSUS_BP4.1: Optimize areas of code that consume the most resources

Optimize the code that runs within different components of your architecture to minimize resource usage while maximizing performance. For more details, see SUS03-BP03, SUS03-BP04 in the *Sustainability Pillar whitepaper*.

Prescriptive guidance:

- Make software running at edge devices as efficient as possible. Run performance tests with managed device farms for testing to identify bottlenecks in advance. It might not be easy to upgrade hardware resources at the edge.

CMSUS_5: Are you selecting technologies to minimize data transfer, processing, and storage requirements?

CMSUS_BP5.1: Use software patterns and architectures that best support data access and storage patterns

Understand how data is used within your workload, consumed by your users, transferred, and stored. Use software patterns and architectures that best support data access and storage to minimize the compute, networking, and storage resources required to support the workload. For more details, see SUS03-BP05 in the *Sustainability Pillar whitepaper*.

Prescriptive guidance:

- Connected mobility applications might be generating different types of data. Use software and architecture patterns that align best to your data characteristics and access patterns. For example, use modern data architecture on AWS that allows you to use purpose-built services optimized for your unique use cases. These architecture patterns allow for efficient data processing and reduce the resource usage.

CMSUS_6: What is the downstream impact on the devices with the design and architecture?

CMSUS_BP6.1: Optimize impact on devices and equipment

Understand the devices and equipment used in your architecture and use strategies to reduce their usage. This can minimize the overall environmental impact of your cloud workload. For more details, see SUS03-BP04 in the *Sustainability Pillar whitepaper*.

Prescriptive guidance:

- Most of the automotive grade devices have a specific boot and runtime. Having the devices on for a long time (unnecessarily) or booting them from sleep state to run state has adverse effect on the device lifetime. Need for such reboots and wake-up signals are increasing due to use cases like Shared mobility and companion app driven controls. Best practice is to minimize these wake-up calls to get the status of the ECU instead define events that can only send Wake-up within the vehicle bus and send the logs (push instead of pull).
- Perform power studies to enable keeping long lived connections, while using the least amount of power.
- To wake up the device from deep-sleep (like the TCUs), use TCP/IP-based wake up instead of SMS message wake up through the mobile network.
- Encourage teams to implement Software Defined Vehicle Architectures with Virtualized Hardware development. With this, customers can benefit from unwanted hardware sample manufacturing and decrease carbon footprint.

Data

CMSUS_7: Are you optimizing data ingestion?

CMSUS_BP7.1: Use technologies that support data access and storage patterns and implement a data classification policy

For more details, see SUS04-BP1 and SUS04-BP2 in the *Sustainability Pillar whitepaper*.

CMSUS_BP7.2: Collect and store only what is needed

Optimize data collection frequency. For example, if battery data is being collected, you do not need to collect the data every millisecond although the data is available on the bus. Instead, collect it based on an event or at a larger time interval, such as every 5 seconds.

CMSUS_BP7.3: Avoid multiple connections from the edge

CMSUS_BP7.4: Maximize data transformation at the edge (make data human readable at the edge)

Prescriptive guidance:

- Collect only what is needed. Ensure that there is a business and engineering justification for all data being collected and revisit based on business and engineering needs (rather keeping it constant).
- Use onboard capability to the fullest (monitor edge capability).
- Before the data gets uploaded to the cloud, compress and aggregate the data at the edge wherever possible.
- Use purpose-built intelligent data collection to make the data collection deliberate and flexible.
- Transform and preprocess or polish the data at the edge wherever possible to minimize transactions with the cloud.

CMSUS_8: Are you optimizing data consumption?

CMSUS_BP8.1: Use policies to manage the lifecycle of your datasets

For more details, see SUS04-BP3 and SUS04-BP4 in the *Sustainability Pillar whitepaper*.

CMSUS_BP8.2: Batch the ingested data before storing, if possible, to conserve storage .

CMSUS_BP8.3: Select compute instances and storage mechanisms from a sustainable point of view

For example, use Graviton instances.

Prescriptive guidance:

- Use microservice-based architecture (run what is needed).
- Use data lifecycle policy to move data to into effective database (cheap/more sustainable storage)
- Collect carbon footprint data for the vehicle when it's in motion to help customers with efficient routing and fueling mechanisms.

CMSUS_9: Are you optimizing data analytics?

CMSUS_BP9.1: Use elasticity and automation to expand block storage or file system as data grows to minimize the total provisioned storage

For more details, see SUS04-BP3 and SUS04-BP4 in the *Sustainability Pillar whitepaper*.

CMSUS_BP9.2: Use reduced log level at production

CMSUS_BP9.3: Adopt campaign-based data collection for monitoring

CMSUS_BP9.4: Alert only if needed

Use the appropriate notification mechanism, such as push notifications, SMS, Amazon SNS, and Amazon SES. Adjust the notification threshold and interval as needed.

Prescriptive guidance:

- Use connected vehicle data to encourage the end user to be more sustainable (such as when to charge, and where to charge.)
- Create alerts and metrics based on the data collected and notify personas involved for actions.
- Move the data (old data, used data, and data not required often) to sustainable storage services in specific Regions.
- Delete data or create rules to delete data based on business logic.

Hardware and services

CMSUS_10: Are you using the correct compute instance type and the minimum size needed to process your workload?

CMSUS_BP10.1: Use the minimum amount of hardware to meet your needs and use instance types with the least impact

For more details, see SUS05-BP01 and SUS05-BP02 in the *Sustainability Pillar whitepaper*.

Prescriptive guidance:

- Right size your edge device to do required processing and maintain performance.

- Right size your cloud resources helps to reduce a workload's environmental impact, save money, and maintain performance benchmarks. Use Graviton-based EC2 instances to reduce cost and power consumption.

CMSUS_11: Are you using managed services and serverless?

CMSUS_BP11.1: Use managed services to operate more efficiently in the cloud

For more details, see SUS05-BP03 in the *Sustainability Pillar whitepaper*.

Prescriptive guidance:

- Use managed services in the cloud to shift responsibility to AWS for maintaining high utilization and sustainability optimization of the deployed hardware. Managed services also remove the operational and administrative burden of maintaining a service, which allows your team to have more time to focus on innovation.
- Use the managed service's agent or extension in the device to optimize data transfer and network connectivity.

Process and culture

CMSUS_12: Are you using connected vehicle data to encourage the end user to be more sustainable?

CMSUS_BP12.1: Adopt methods that can rapidly introduce sustainability improvements

For more details, see SUS06-BP01 in the *Sustainability Pillar whitepaper*.

Prescriptive guidance:

- Help customers with efficient routing, efficient fueling mechanism, and so on.

CMSUS_13: Are you making use of managed device farms for testing? Are you developing low-cost testing methods to enable delivery of small improvements over the lifetime of the edge devices and cloud?

CMSUS_BP13.1: Use managed device farms for testing

For more details, see SUS06-BP04 in the *Sustainability Pillar whitepaper*.

Prescriptive guidance:

- Establish cloud native testing practices to understand expected impact, and test with customers to understand the actual impact from using your services. Device farms are the second-best option.
- Use managed device farms to help you streamline the testing process for new features on a representative set of hardware. Managed device farms offer diverse device types including earlier, less popular hardware, and avoid customer sustainability impact from unnecessary device upgrades.

Key AWS services

- AWS Graviton Processor – Use GPU-enabled EC2 processing for your workloads.
- AWS Lambda – Serverless provides scalable processing.
- AWS IoT Core – Connect, manage, and scale your edge device fleets easily and reliably without provisioning or managing servers.
- AWS IoT GreenGrass – Bring cloud processing and logic locally to edge devices and program your devices to transmit only high-value data, making it easy to deliver rich insights at a lower cost.
- AWS IoT FleetWise – Intelligent data collection from edge devices that sends only needed data to the cloud for analysis.
- AWS Device Farm – Improve the quality of your applications testing across simulated IoT/Mobile devices hosted in the AWS Cloud
- AWS Compute Optimizer – Right-size workloads with artificial intelligence and machine learning-based analytics to reduce cost and improve sustainability.
- AWS Trusted Advisor – Optimize your AWS infrastructure with Trusted Advisor recommendations and reduce or remove unused resources.

- AWS Customer Carbon Footprint Tool – Measure the estimated carbon emissions from your use of AWS services.

Resources

- **Documentation and blogs:**
  - AWS Connected Vehicle Reference Architecture
  - Building and Modernizing Connected Vehicle platforms with AWS IoT
- **References:**
  - AWS Well-Architected
  - AWS Architecture Center
  - Sustainability in the Cloud
  - AWS enables sustainability solutions
  - The Climate Pledge
  - United Nations Sustainable Development Goals
  - Greenhouse Gas Protocol

Conclusion

The AWS Well-Architected Framework provides architectural best practices across the six pillars for designing and operating reliable, secure, efficient, cost-effective, and sustainable systems in the cloud. The Framework provides a set of questions that helps you to review an existing or proposed architecture. It also provides a set of AWS best practices for each pillar. Using the Framework in your architecture can help you produce stable and efficient systems, which allow you to focus on your functional requirements.

Contributors

The following individual authors, collaborators, and editors contributed to this document:

- Allan Holmes - Senior Technical Account Manager, AWS Enterprise Support
- Andrea Caldarone - Senior Technical Account Manager, AWS Enterprise Support
- Asif Haque - Technical Account Manager, AWS Enterprise Support
- Asif Khan - Principal Solutions Architect, AWS
- Bruce Ross - Well Architected Lens Lead Solution Architect, AWS
- Djamel Bourokba - Senior Global Solution Architect, AWS
- Elliston Kieng - Senior Technical Account Manager, AWS Enterprise Support
- Girish Shenoy - Senior Technical Account Manager, AWS Enterprise Support
- Harshika Thusu - Technical Account Manager, AWS Enterprise Support
- James Simon - Connected Mobility Tech Lead, AWS
- John Marciniak - Senior Solution Architect, AWS
- Juan Sostre - Senior Technical Account Manager, AWS Enterprise Support
- Karthik Manjunatha - Senior Consultant, AWs Professional Services
- Maitreya Ranganath - Principal Security Architect, AWS
- Mohan Udyavar - Principal Technical Account Manager, AWS Enterprise Support
- Omar Zoma - Senior Security Solution Architect, AWS
- Praveen Gupta - Technical Account Manager, AWS Enterprise Support
- Praveen Muppala - Senior Technical Account Manager, AWS Enterprise Support
- Sanjeev Kumar - Senior Technical Account Manager, AWS Enterprise Support
- Vijai Thoppae - Senior Solution Architect, AWS

Document revisions

To be notified about updates to this whitepaper, subscribe to the RSS feed.

Change	Description	Date
Initial publication	Whitepaper first published.	January 3, 2024

Note

To subscribe to RSS updates, you must have an RSS plug-in enabled for the browser that you are using.

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided "as is" without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.