ROYAL ENFIELD	TECHNICAL DOCUMENT	
Title: SOR_EV_SW_development	**Technical Doc No:**	**Date:** 08/05/23
		Sheet 1 of 24

Technical Requirements Input

Sign		
Date	15/02/2023	
DR No.		
Issue		

Prepared By:	Checked By:	Approved By:
Signature -Sd-	Signature: -Sd-	Signature: -Sd-
Name: Veeraju and Tarun	Name: Gaurav Gupta	Name: Umesh Krishnappa

Review /Change History

S.no	Version no	Change description	Date	Author	Approved
1	1.0	First versions	08/05/23	Veeraju, Venkat, & Tarun	

Table of contents

Section 1 to 4 are related to Embedded C based S/w development.

Section 5 and 6 are Android based S/w development.

1 Head Unit Functions

The head unit of the vehicle is the dashboard or instrument cluster that displays important information to the rider, such as speed, battery level, range, and other indicators. The head unit is also responsible for the power supply sent to the display and touch feedback from the touch panel. Major functions are mentioned as below: -

Functions	Description
PWM for Backlight control	The PWM signal controls display brightness, adjusted based on data from an ALS sensor. Display power supply is calibrated based on PWM signal for automatic brightness adjustment.
Fail Safe Display	The fail-safe display for the head unit ensures the display functions in case of SOM failure by detecting miss in either SoM heartbeat signal on CAN or MIPI DSI communication. In such a case, a minimum information should still be displayed on screen using vehicle CAN bus and at a refresh rate of 5 frames / second.
Touch Response	Capacitive touch response detects voltage changes from the user's touch on the screen. It uses I2C protocol for communication with the touch controller, which provides

	the system with touch input coordinates.
Bootloader	The bootloader is responsible for loading the firmware onto the microcontroller memory (the built-in flash memory).

2 UDS(Unified Diagnostic Services)

UDS stands for Unified Diagnostic Services, which is a communication protocol used in the automotive industry for diagnosing and communicating with electronic control units (ECUs) in vehicles. It is a standardized protocol that allows a diagnostic tester to access and interact with the various ECUs present in a vehicle, such as the engine control module, transmission control module, and other control units.

Service ID (Hexadecimal)	Service Name	Yes/No
0x10	Diagnostic Session	Yes
0x11	ECU Reset	Yes
0x14	Clear Diagnostic Information	Yes
0x19	Read DTC Information	Yes
0x22	Read Data By Identifier	Yes
0x23	Read Memory By Address	Yes
0x24	Read Scaling Data By Identifier	Yes
0x27	Security Access	Yes
0x28	Communication Control	Yes
0x2A	Read Data By Periodic Identifier	Yes
0x2C	Dynamically Define Data Identifier	Yes
0x2E	Write Data By Identifier	Yes
0x2F	Input Output Control By Identifier	Yes
0x31	Routine Control	Yes
0x34	Request Download	Yes

0x35	Request Upload	Yes
0x36	Transfer Data	Yes
0x37	Request Transfer Exit	Yes
0x3D	Write Memory By Address	Yes
0x3E	Tester Present	Yes
0x83	Access Timing Parameter	Yes
0x84	Secured Data Transmission	Yes
0x85	Control DTC Setting	Yes
0x86	Response On Event	Yes
0x87	Link Control	Yes

3 OBD (On-Board Diagnostics)

OBD stands for On-Board Diagnostics, which is a system used in vehicles to monitor and report the performance of various components and systems. The OBD system is typically located in the vehicle's engine control module (ECM) and is designed to detect malfunctions or abnormal behavior in the engine, transmission, emissions, and other systems.

Mode	Mode Name	Yes/No
1	Show Current Data	Yes
2	Freeze Frame Data	Yes
3	Show Stored Diagnostic Trouble Codes	Yes
4	Clear/Reset Diagnostic Information	Yes
5	Oxygen Sensor Monitoring	Yes
6	On-Board Monitoring Test Results	Yes
7	Show Pending Diagnostic Trouble Codes	Yes
8	Control Operation of On-Board Systems	Yes
9	Vehicle Information	Yes
0A	Permanent Diagnostic Trouble Codes	Yes

4 CAN TP

CAN TP (CAN Transport Protocol) is a communication protocol that is used in automotive applications for transferring large amounts of data between Electronic Control Units (ECUs) in a vehicle.

Functions	Support
Half-Duplex	The TP layer must support Half-Duplex
Full-Duplex	Optional
Bootloader	CAN TP must support Bootloader
UDS	CAN TP must support UDS
Flow Control	CAN TP must support flow control
Error Control	CAN TP must support error control

5 ECOS (Electrical Check Out System):

The Electrical Check Out System (ECOS) app is a software application designed to perform electrical tests on vehicles during the end-of-line (EOL) process. It allows users to scan the vehicle's barcode, retrieve information about the vehicle, and perform a range of tests to ensure that the vehicle's electrical system is functioning properly. The app includes features like Wi-Fi connectivity, user acceptance testing, and motor tuning.

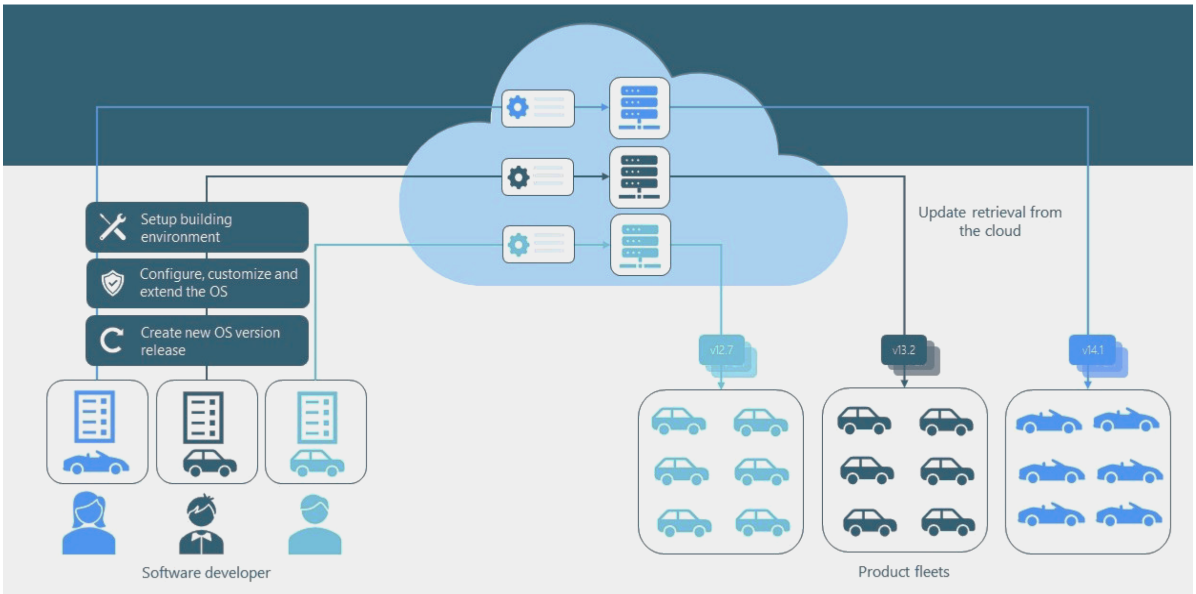
Functions	Description
Scanner	The ECOS app should include a scanner function for scanning barcodes or QR codes on the vehicles to retrieve information about the vehicle.
Wi-Fi connectivity	The app should be able to connect to a Wi-Fi network to download vehicle information and upload test results.
General testing	The app should include a general testing function to ensure that all the required features and functions like Indicator lighting, Horn, Head unit display are working properly.

Motor tuning	The app should include a function for automatically tuning the motor of the vehicle to ensure optimal performance.
Configuration	Different UDS parameter setting as per vehicle variant, VIN series, LTE/GPS setting as per export market, etc

6 Android Applications

6.1 OTA Manager

- Over-The-Air known as OTA refers to the capability of remotely updating software or firmware on devices without the need for physical connections. OTA technology allows for the seamless and efficient delivery of updates, patches, and new features to devices over wireless networks.
- The main purpose of OTA is to keep devices updated with the most recent software releases, assuring improved performance, security, and functionality.
- Users no longer have to manually upgrade their devices by plugging them into a computer or going through difficult installation processes. Instead, updates can be sent wirelessly, usually through cellular networks, Wi-Fi, or other wireless communication channels, straight to devices.



Type of OTA updates:

Based on User end perspective, it is of two types:

- **Automatic/ Forced/ Silent OTA:**
  - -No user interaction is involved
  - -The update is configure via a Mobile Device Management Service (MDM)
  - -MDM is used to configure, provision and manage embedded device remotely

- **Manual OTA:**
  - -Requires end user to take some action to confirm the update
  - -Either initially configured to check for updates regularly or the user must explicitly click on a link to check for updates.
  - -Does not require MDM to execute the updates

Full Image update OTA:

-Entire update must be installed at once or not installed at all

Incremental update OTA or Delta OTA:

-Smaller software updates or patches

-Usually has tiny binary patches that are already in use in the device

A/B System updates:

-A/B system contains two partitions called slots. The current slot is the active & bootable slot and the update is applied to the inactive slot. So if the update fails, the device can still run the old OS

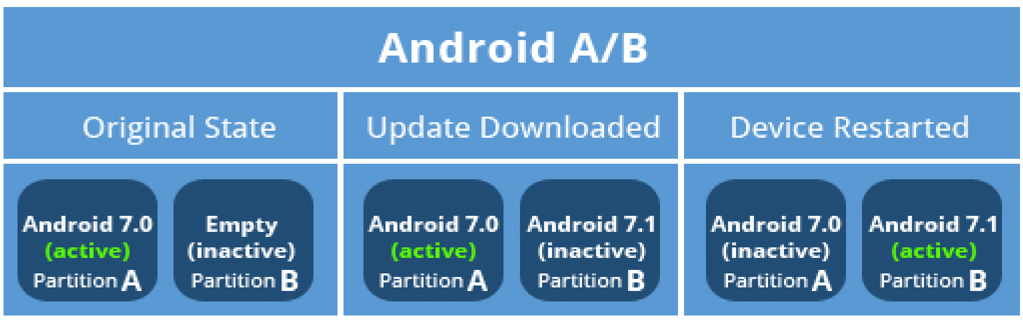
-A/B system keeps a duplicate copy of the system firmware, i.e /boot, /system, /vendor and other vendor specific partitions

-/boot partition now has the recovery Ramdisk and the system uses updated_engine to apply updates

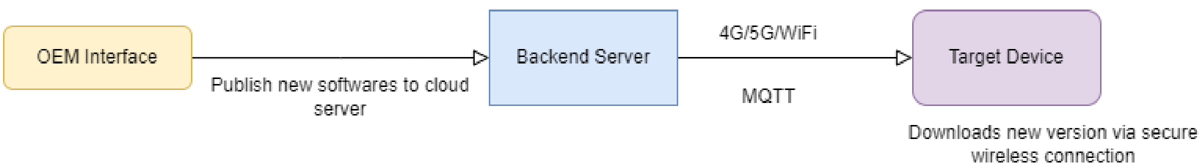
-update_engine is a background daemon used by the A/B system for updates, it reads the update package from the current slot and writes it to the other slot. Once done, it marks the other slot as active and bootable.

-This update can be done while the device is still running and the system downtime is minimal

-Also supports streaming updates, which can allow the android system to apply updates as they are downloaded, since the full package does not need to be downloaded, /cache partition not needed for streaming updates.



OTA workflow:



OTA flow is classified into three major stages:

1. **Building OS image**
2. **Hosting OS image**
3. **Installing OS image**

BUILDING OS IMAGE:

1. Development and CI
    a. -Developers write and test code for automotive software
    b. -The code is committed to a version control system
    c. -CI servers monitor the repository for changes and build processes

2. Build and Unit Testing:
    a. -CI server pulls the latest code and starts the build process
    b. -Compiles code, resolves dependencies and generates executable files and packages.
    c. -Unit tests ensure the individual components are working correctly

3. Integration Testing:
    a. -Once the build and unit testing are successful, integration testing is performed.
    b. -Integration tests verify the interactions between various software components and their compatibility.

4. Packaging and Release:
    a. -After passing integration testing, the software is packaged for deployment.
    b. -The software is bundled into appropriate packages, such as firmware or software update packages, compatible with the automotive system.
    c. -Versioning and metadata are added

HOSTING OS IMAGE:

1. Deployment of OTA server:

   a. The packaged software is deployed to an OTA server or a dedicated software management platform.
   b. The OTA server acts as a central repository for managing software updates and distribution to vehicles.
2. Vehicle registration and target selection:
   a. Vehicles are registered with the OTA server to establish communication channels.
   b. Vehicle identification, software version, and other relevant information are stored in the OTA server's database.
   c. Target vehicles for the software update are selected based on criteria such as software version, hardware compatibility, and user preferences.
3. Scheduling and delivery:
   a. The OTA server schedules the delivery of the software update to target vehicles.
   b. The delivery may be immediate or scheduled for a specific time window based on vehicle availability, user preferences, or network conditions.
   c. The OTA server sends notifications to the target vehicles, informing them about the availability of the update.
   d. It is preferred to send software updates in batches to the vehicles in order to avoid flooding the server with requests from users

INSTALLING OS IMAGE:

1. Over the Air update:

   a. Vehicles receive the update notification and establish a secure connection with the OTA server.
   b. The software update package is securely downloaded over the air to the vehicle's onboard system.
   c. The update is verified, and necessary validations, such as cryptographic checks, are performed to ensure integrity and authenticity.
2. Installation and Validation:
   a. The update package is installed on the vehicle's system, replacing the existing software or adding new features.
   b. Post-installation validations and tests are conducted to ensure the update was successful and the system operates as expected.

c. The validation is done by calculating a checksum by CRC (Cyclic redundancy check)
    i. **Checksum:** Used to verify whether the data transmitted and received are the same and integrity is maintained.
    ii. The CRC algorithm performs calculations on data and generates a checksum that is stored on the last byte at the source.
    iii. The calculation is again done at recipient and Checksum is matched and verified
d. Other validation mechanisms include Certificate validation, Metadata validation, Digital signatures.
e. If any issues are detected, rollback mechanisms or error handling procedures are implemented.
f. Rollback mechanism is a feature that allows reverting back to the previous state or version of the software. I.e acts as a safety net. Possible errors forcing rollback may be
    1. Interrupted download process
    2. Interrupted installation
    3. Flaws in updated firmware
    4. Firmware incompatibility
g. Rollback installation too is validated to ensure proper working of the device
h. The software update is again initiated after a cool-off period

3. <u>Reporting and analytics:</u>

    a. The vehicle reports the installation status, logs, and relevant data back to the OTA server.
    b. The OTA server collects analytics, performance metrics, and feedback for further analysis and improvement of the software.

4. <u>Monitoring and maintenance:</u>
    a. The OTA server monitors the status of deployed updates and tracks the software versions on each vehicle.
    b. Continuous monitoring helps identify potential issues, performance bottlenecks, or user feedback for further updates and improvements.

c. Regular maintenance activities, such as security patch releases or feature enhancements, are performed based on the feedback and monitoring data.

OTA BUNDLE:

The components are

1. **Firmware/Software** - The updated software that needs to installed
2. **Manifest file** - This provides metadata and information about the version number, compatibility notes, release notes, dependencies and file list
3. **Digital signatures** and cryptographics
4. **Configuration files** if needed
5. **Validation** mechanisms
6. **Rollback** data
7. Contains OTA packages for each **ECUs** in the vehicle namely,
   a. BMS
   b. MCU
   c. OBC
   d. Head Unit
   e. ESCL
   f. ABS
   g. Switch cubes

OEM Backend:

The Backend is responsible for

1. **Update management:**
   a. Uploading binaries to the cloud server
   b. Configuration of the communication type
2. **Device Management:**
   a. Storing Vehicle data such as variant model, VIN, and ECU details and the version
3. **Campaign Management:**
   a. Create a campaign

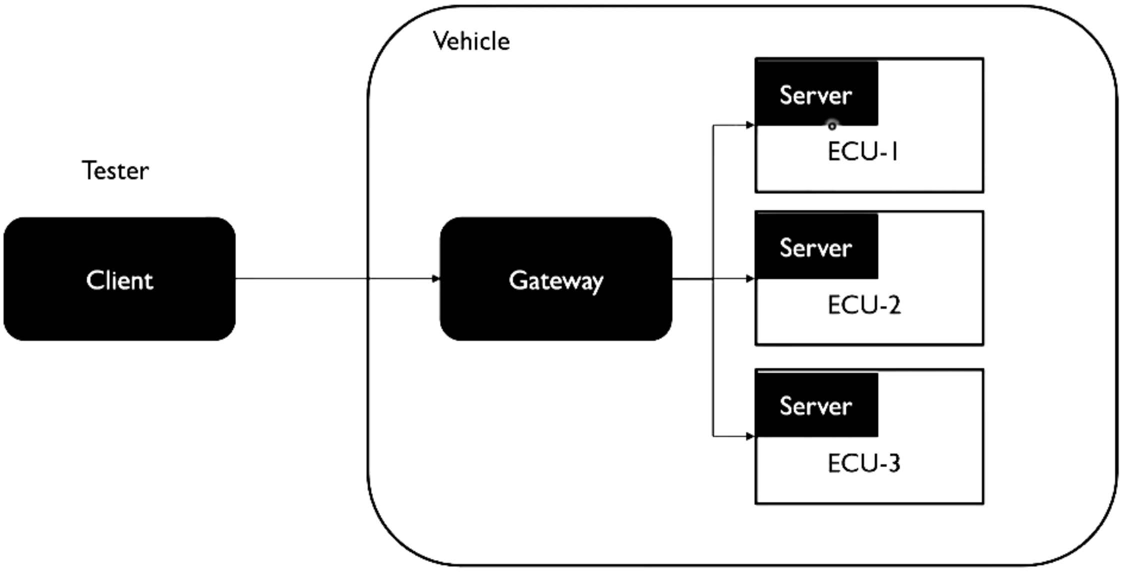
b. Save the campaign data, schedule the campaign and status of campaign
c. Handle notifications via MQTT to TCU

OTA CAMPAIGN:

This refers to the coordinated and planned process of delivering updates to a group of devices i.e developing a strategy to outline the development and distribution of the updates. It involves managing and executing the deployment of updates in a controlled and organized manner.

6.2 UDS Client

- Requirements for Diagnostic systems.
- UDS is a collection of diagnostic services that can be requested by a tester as a client and performed or provided by the ECU as server
- And it finds existence on the Application layer of the OSI model, where:
  - ISO 14229-1 specifies, Specifications and Requirements.
  - ISO 14229-3 specifies UDS on CAN.
  - ISO 14229-4 specifies UDS on FlexRay.
  - ISO 14229-5 specifies UDS on IP.
  - ISO 14229-6 specifies UDS on K-Line.
  - ISO 14229-7 specifies, UDS on LIN.

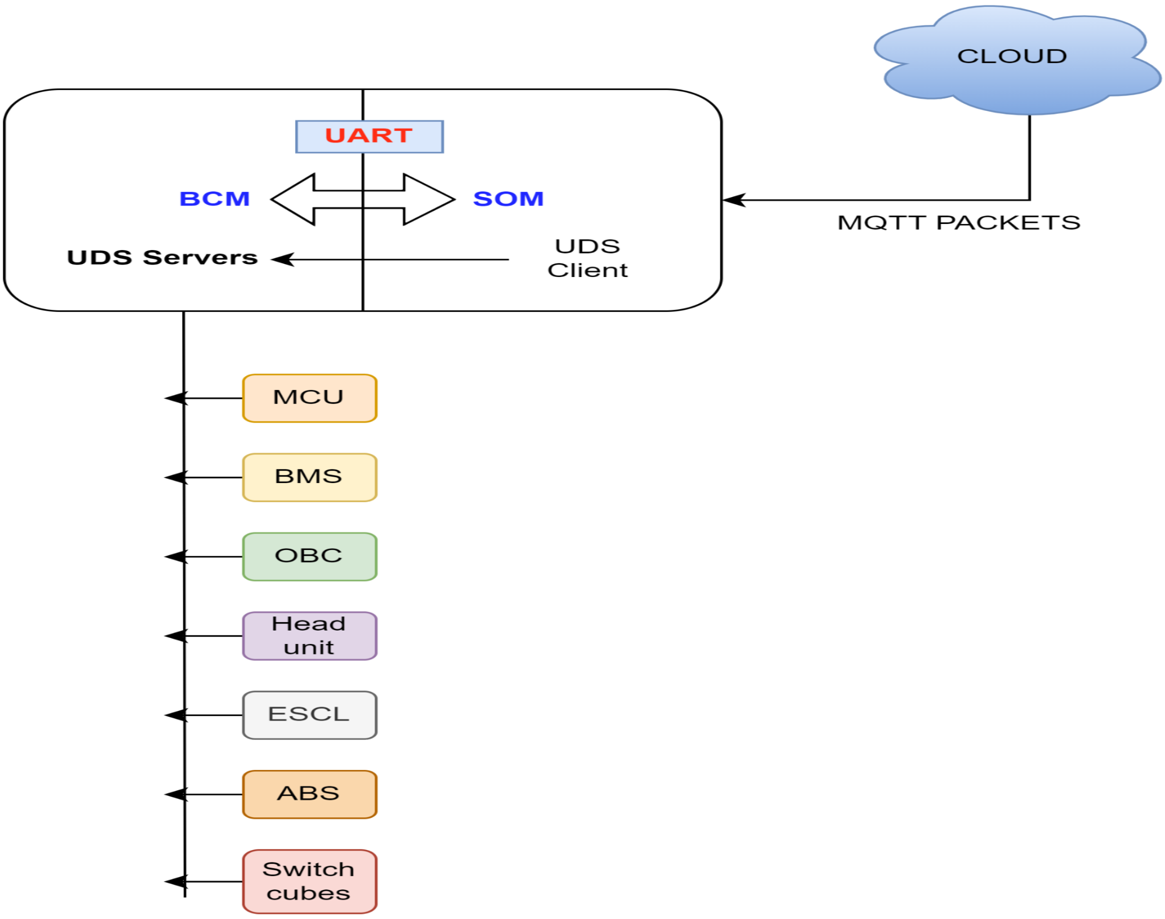


UDS SERVICES for illustration:

Service ID (hex)	Service	Description
0x11	ECU Reset	Reset the ECU ● Hard reset ● Soft reset ● Key off on reset
0x22	Read Data By Identifier	Retrieve one or more values of a control unit. This can be

		information of all kinds and of different lengths.
0x27	Security Access	Seed-Key mechanism is used to unlock security critical data
0x28	Communication Control	With this service, both the sending and receiving of messages can be turned off in the control unit.
0x2E	Write Data By Identifier	Values in a DID can be changed
0x31	Routine Control	Start, stop and query the result of a service
0x34	Request Download	Downloading new data to ECUs
0x35	Request Upload	Data from ECUs are transferred to tester
0x36	Transfer Data	Used for actual transmission of data
0x37	Transfer Exit	Exits data transmission process
0x3E	Tester Present	Signals that client is still present to avoid ECUs to exit session

Concept image:



- Cloud initiates a request for a specific UDS service, this request is transmitted to the SOM via MQTT packets.
- The SOM is responsible for handling UDS protocol stack, which includes message formatting, session handling and error handling
- Based on the request, SOM generates UDS commands specific to the request such as read, write etc.
- The SOM transmits the UDS commands to the BCM via UART, the BCM acts as a gateway and relays the commands over to the relevant ECUs via CAN bus
- The ECUs receive the commands and process them accordingly
- ECUs generate their response based on the outcome and send it back to BCM, which relays it to SOM
- The SOM collects the ECU responses and prepares them and sends them to the cloud for analysis.

Diagnostics Communication Manager (DCM):

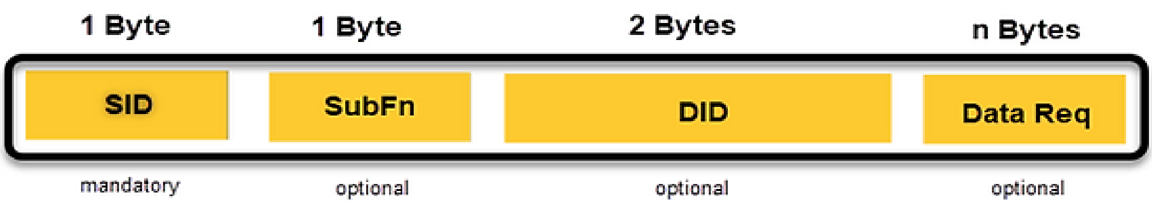
- DCM refers to a software module that manages the communication between a diagnostic tester and ECUs in a vehicle
- It acts as an intermediary between diagnostic tester and the ECUs, facilitating the exchange of diagnostic messages and data.
- It provides a standardized interface for communication and compatibility between different ECUs and diagnostic tools
- It takes a request from the tester, processes the request, accumulates the response and sends back the proper response.

  Functions:
  1. Initializes diagnostic communication with ECUs, configures communication parameters and channels.
  2. Manages diagnostic session between tester and ECUs. It handles session establishment, switching between sessions and session termination
  3. Handles transmission and reception of diagnostic messages between client and server
  4. Incorporates security access to control access to protected data and information.
  5. Monitors and handles error conditions from ECUs, reports error to client and provides Diagnostic Trouble Codes (DTC) and other information for troubleshooting.
  6. Manages storage and retrieval of diagnostic data exchanged between client and server

SERVICE REQUEST MESSAGE:

- A service request message refers to the message sent by a diagnostic tester to an ECU to request a specific diagnostic service or operation. The service request message initiates the interaction between the client and the ECU, indicating the desired action or information needed from the ECU.
- The message is sent by Client to Server
- Each diagnostic service has Service Identifier (SID) of 1 byte length, with which the server understands what service is requested by the client
- Ex:0x22 - Read data by identifier

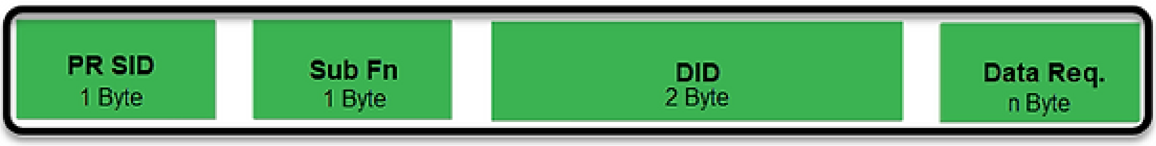
- SID is the first byte of the message
- The next byte is Subfunction. This is optional, tells server, for service, which specific sub functionality is requested
    - Ex: 0x31 is routine control service, for it 0x01 is subfunction start the routine
- The next 2 bytes are the Data Identifier (DID).
    - Client and server communicate only by numbers, therefore for all the data elements that the tester might read, it is assigned a number already.
    - While in OBD, DIDs are standardized globally, for UDS, OEM's define their own UDS. Thus only their tester tools can read these DIDs
    - Single or multiple DIDs can be present in a service request message.
- Data Record field specifies what new value has to be stored in that data element. It can be considered as the metadata of DID.



Positive Response Message:

- This is a specific type of response message sent by an electronic control unit (ECU) to acknowledge and fulfill a Service Request Message from a client.
- It indicates that the requested operation has been successfully executed or that the requested information is provided.
- The positive response message serves as a confirmation from the ECU to the diagnostic tester that the requested operation has been successfully performed or that the requested information is available.
- It allows the diagnostic tester to proceed with further diagnostic activities based on the received response.

- Once the SRM is received by the server, it checks the message and performs the service requested and sends positive response to the client.
- The message format has a PR SID (positive response SID) which is the addition of SID requested with 0x40.
- It is followed by subfunction, DID same as SRM
- The data record field displays the value of data element represented by the DID



Negative Response Message:

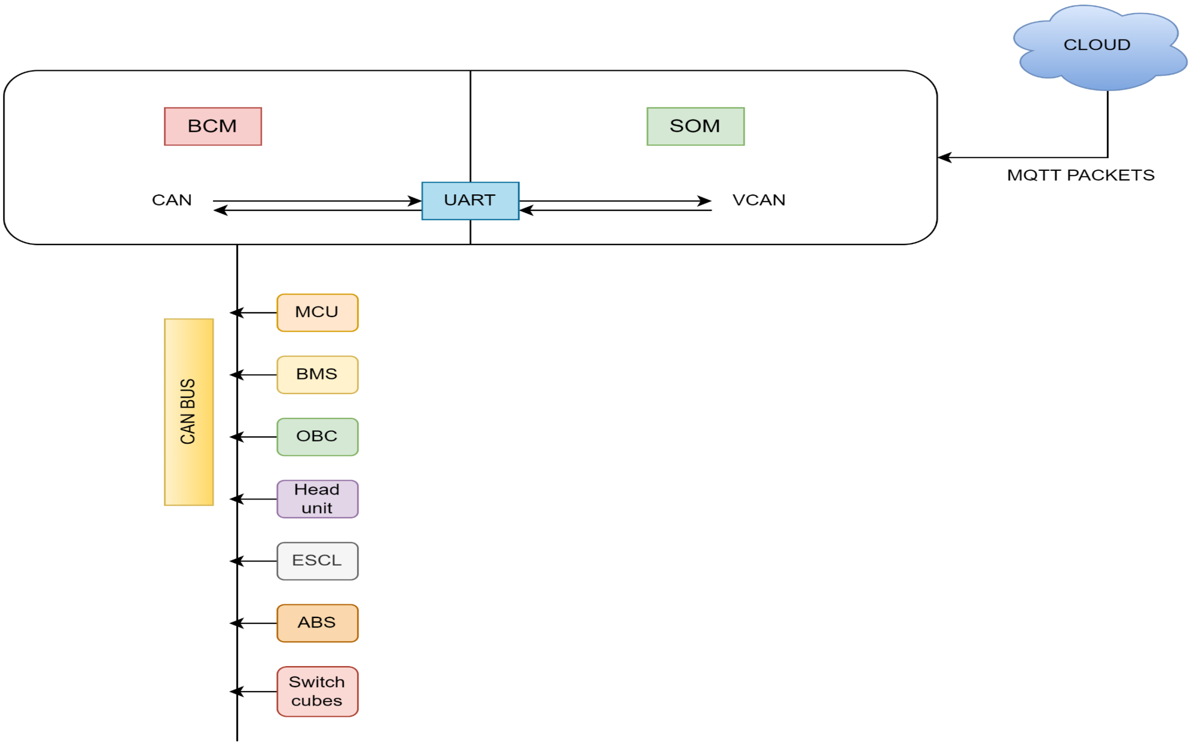
- A negative response message is sent by ECU to indicate an error or failure in response to a requested diagnostic service from a client.
- It signifies that the requested operation cannot be fulfilled or that an error has occurred during the processing of the request.
- The message format has a NR_SID, negative response SID at first byte which is predefined as 0x7F irrespective of service requested
- Second byte is Service ID requested, SIDRQ by the client
- Final byte is Negative Response Code (NRC) indicates the reason for not performing the requested service

6.3 CAN DATA AGGREGATOR

- CAN data aggregator collects and consolidates data from multiple ECUs connected to a CAN bus.
- Its primary purpose is to gather data from different ECUs on the CAN bus and provide a centralized view or representation of that data.
- It is the primary gateway for receiving data messages from the ECUs and processing them further for analysis and transmission.

Functions:

- Message reception - receives CAN messages from ECUs such as sensor data, control commands, status updates.
- Collects and combines data into a unified data set
- Organizes the data based on desired structure, type or size
- Forwards the data to a cloud platform



Workflow:

1. There are 7 ECU's within the vehicle namely, MCU, BMS, OBC, Head Unit, ESCL, ABS and switch cubes, these ECU's communicate with each other and with BCM via CAN bus.
2. The BCM acts as the central hub for all the ECUs connected to CAN. It also serves as interface between ECUs and SoM
3. SoM is responsible for collecting CAN data and aggregate it and transmit to cloud
4. However, SoM lacks native CAN capability, It is connected to the BCM via UART.
5. The BCM converts all the CAN data into UART for communication with SoM
6. Since, SoM lacks CAN, it uses Virtual CAN (VCAN) to convert the UART data it received into CAN data

7. VCAN is a software through which we can emulate the behavior of CAN interface within the SoM. It provides the necessary protocols, drivers and functionality to interpret UART to CAN messages.
8. By utilizing VCAN, the SoM gains the ability to interact with the CAN data from ECUs enabling seamless integration of aggregated data.
9. The SoM connects to the cloud infrastructure via MQTT and establishes a secure connection and transmits the aggregated CAN data for storage, processing and analysis.
10. The SoM configures all the required data that needs to be sent to the cloud.
11. Android has the capability to allow clients to **subscribe** to obtain these data.

Data transfer to cloud:

Configuring the data transfer from SoM to cloud involves adjusting parameters such as packet size, size of data and frequency of transmission with respect to the type of data required to be sent.

Packet size:

It can be changed based on factors such as network bandwidth, latency and size limitations imposed by the infrastructure.

Some types:

1. Fixed Packet size: constant packet size is defined for all data transmissions. Each packet has a predetermined size, a specific number of bytes.
2. Variable Packet size: The packet size can dynamically change based on the amount of data to be sent. This allows for flexibility in the data transmission.

Maximum Transmission unit (MTU) refers to the maximum size of a packet that can be transmitted over a network protocol. Adherence to the MTU can ensure that the data packets can be transmitted without fragmentation or loss.

Size of Data:

- It is possible to distinguishingly send data from different ECUs by using different data sizes such as Kb,Mb etc. This approach can be beneficial in scenarios where the data generated by different ECUs varies significantly in

volume or importance. By allocating different sizes, we can optimize the data transmission process.
- However, it is crucial to have proper synchronization and coordination between the SoM and cloud to correctly interpret and handle different data sizes.

Types of Event:
The messages sent from SoM to Cloud, are batched or grouped based on the event type. These types are

- Low priority event
- High priority event
- High frequency event

LOW PRIORITY EVENT:

1. These messages contain non-critical or non-urgent information
2. These include periodic vehicle status updates, non-critical system events, sensor readings.
3. These information are all merged into a GENERAL VEHICLE PARAMETER PACKET and is sent to cloud in short regular intervals from 5-10s.
4. These messages are sent at lower frequencies than HPE or HFE and may not require immediate attention and processing.

HIGH PRIORITY EVENT MESSAGES:
1. These are critical and time-sensitive information that requires immediate attention or action.
2. These provide alerts for Critical faults of Grade 2, 3 and 4.
    a. Grade 2 faults are moderate level faults that do not pose immediate safety risk but cause certain subsystems and functionalities to be affected. Ex: Failure of Infotainment, minor sensor malfunctions
    b. Grade 3 faults indicate higher severity and can have more significant impact on vehicle operation. These are partial or complete loss of major drivetrain components, significant battery related issues, malfunctions in safety systems like braking, steering.

c. Grade 4 faults are the most severe faults. They pose immediate safety risks and result in complete loss of essential functions and lead to dangerous situations. Examples include critical failure in high voltage electrical systems, complete system shutdown and severe battery failure.

HIGH FREQUENCY EVENT:

1. These messages are transmitted at a high rate or frequency. They carry real time or near real-time information that needs to be updated and processed rapidly.
2. Examples are accidents and collisions.
3. Example scenario:
    - The vehicle has an Inertial Measurement Unit (IMU) that constantly monitors the vehicle's motion and orientation. When a sudden fall or impact is detected, the IMU registers a significant spike in the g-level measurement. This spike serves as an indicator of potential accident or collision
    - Once the IMU detects the fall, it triggers SoM to send a HFE message. The SoM collects relevant data for a specified duration before and after the event to capture critical information.
    - It retrieves data of all the relevant parameters. Let's say the SoM collects data for a time window of 1s before and after the event. It samples the data at a frequency of 100 milliseconds.
    - Once data collection is done, the SoM packages the data into a HFE message. This message contains a series of data snapshots each for a 100ms interval within the time window. The message is then sent to cloud