RE-SOM FOTA
Software Design Document
(Version 1.0.0)

S.No	Prepared By	Review By	Version No
1	Venu	Hareesha/Aparna	1.0.0
2	Jayanth		1.0.0
3	Varshitha	Hareesha/Aparna	1.0.0

Table of Contents

1. Introduction

FOTA enables seamless, efficient, and secure distribution of firmware updates, eliminating the need for manual intervention or physical access to devices. This transformative technology not only enhances user experience by ensuring devices remain up-to-date with the latest features and security patches but also significantly reduces operational costs and downtime associated with traditional update methods.

The FOTA process allows ECU software updates to be safely paused and resumed across driving cycles, without hindering vehicle functionality. It rigorously verifies new software for authenticity and integrity, ensuring updates are secure. Upon successful verification, updates require a specific ECU mode for activation, preventing execution while driving and maintaining safety. Additionally, an internal rollback feature keeps the previous version available for reactivation, enhancing system resilience. This streamlined approach ensures seamless and secure software integration, safeguarding vehicle operation and safety.

2. OTA Bundle

➢ FOTA Components

1. Firmware/Software and Manifest file

Component	Description
1) Firmware/Software	The updated software for installation on Electronic Control Units (ECUs).
2) Manifest File	Provides metadata such as version numbers, compatibility notes, release notes, dependencies, and a file list.

2. Digital Signatures and Configuration Files

Component	Description
3) Digital Signatures and Cryptographics	Ensures the authenticity and integrity of software updates.
4) Configuration Files	Contains necessary settings for the updated software if required.

3. Validation Mechanisms

Component	Description
5) Validation Mechanisms	Implements methods to validate the correctness and integrity of the updated software.

➢ OEM Backend

The OEM Backend, responsible for managing updates, devices, and campaigns, includes functions like Update Management, which involves uploading binaries to the cloud server and configuring communication types; Device Management, storing vehicle data such as variant model, VIN, and ECU details and versions; and Campaign Management, handling the creation, scheduling, and status tracking of update campaigns.

➢ OTA Campaign

The OTA Campaign process encompasses defining and executing a strategy for delivering updates to groups of devices in a coordinated and controlled manner, outlining the development and distribution strategy. Additionally, OTA Packages are prepared for each ECU in the vehicle, such as Battery Management System (BMS), Motor Control Unit (MCU), On-Board Charger (OBC), Head Unit, Electronic Steering Column Lock (ESCL), Anti-lock Braking System (ABS), and Switch Cubes.

3. Firmware Update Management with FOTAManager

The capability to update firmware over-the-air (FOTA) is crucial for enhancing functionalities, fixing bugs, and ensuring the highest level of security. The FOTAManager Java class provides an integrated solution for managing these updates seamlessly, without requiring physical access to the vehicle's systems. This document outlines the comprehensive approach taken by the FOTAManager to handle firmware updates, showcasing its interaction with other components and its role in the update process.

Initialization and Update Notification
The FOTAManager initializes by setting up a connection to Firebase, a popular cloud-based platform, to listen for notifications related to firmware updates. This initialization step is critical for real-time update management, ensuring that the vehicle can react promptly to available updates. The manager actively monitors for these notifications, which are triggers indicating that a new firmware update is available for download.

Download Management
Upon receiving an update notification, the FOTAManager interacts with Android's DownloadManager to facilitate the download of the firmware update package. This integration ensures that the download process is handled efficiently, with support for resuming interrupted downloads and providing feedback on the download progress. The use of Android's native download manager leverages robust system-level support for managing large files, which is often the case with firmware updates.

Firmware Extraction and Validation
After the download completes, the FOTAManager employs the OTABundleExtractor to unpack the OTA bundle. This step is crucial for extracting the firmware image and any associated metadata, including version information and checksums for validation. The extraction process ensures that the firmware update is intact and has not been tampered with during transmission. This validation phase is a critical security measure, safeguarding against the installation of corrupted or malicious firmware.

Update Deployment
With the firmware validated, the FOTAManager proceeds to the deployment phase. It utilizes the DiagMain class to send the firmware to the vehicle's systems via diagnostic CAN messages. This step demonstrates the integration of the FOTAManager with lower-level vehicle communication protocols, enabling the update process to be conducted directly with the vehicle's electronic control units (ECUs).

The deployment process is designed to be as unobtrusive as possible, often scheduled to run at times when the vehicle is not in use, thereby minimizing any impact on the vehicle's availability to the user. Additionally, the FOTAManager implements robust error handling and logging mechanisms throughout the update process. These mechanisms are essential for diagnosing issues that may arise during the update, ensuring that any problems can be quickly identified and resolved.

The FOTAManager class represents a sophisticated approach to managing firmware updates in connected vehicles. Through its integration with cloud services for update notifications, efficient handling of download processes, secure validation of firmware packages, and direct communication with vehicle systems for update deployment, the FOTAManager provides a comprehensive solution for FOTA updates. This system not only enhances vehicle security and functionality but also ensures that the process is seamless and minimally intrusive to the end-user, marking a significant advancement in connected vehicle technology.

4. UDS Flow Configuration

Pre-Programming Procedure
Purpose: To prepare the ECU for programming by ensuring software compatibility, sufficient power supply, and secure communication.
Steps:

- Check software version and compatibility.
- Ensure vehicle and ECU are in a safe state for programming.
- Backup existing ECU data/configuration, if necessary.

Data-Block Programming Procedure
Purpose: To update or program the ECU by transferring and writing new firmware/data blocks.
Steps:

- Sequentially transfer data blocks to the ECU, ensuring data integrity.
- Write data blocks to specified memory locations within the ECU.
- Verify integrity of each written data block.

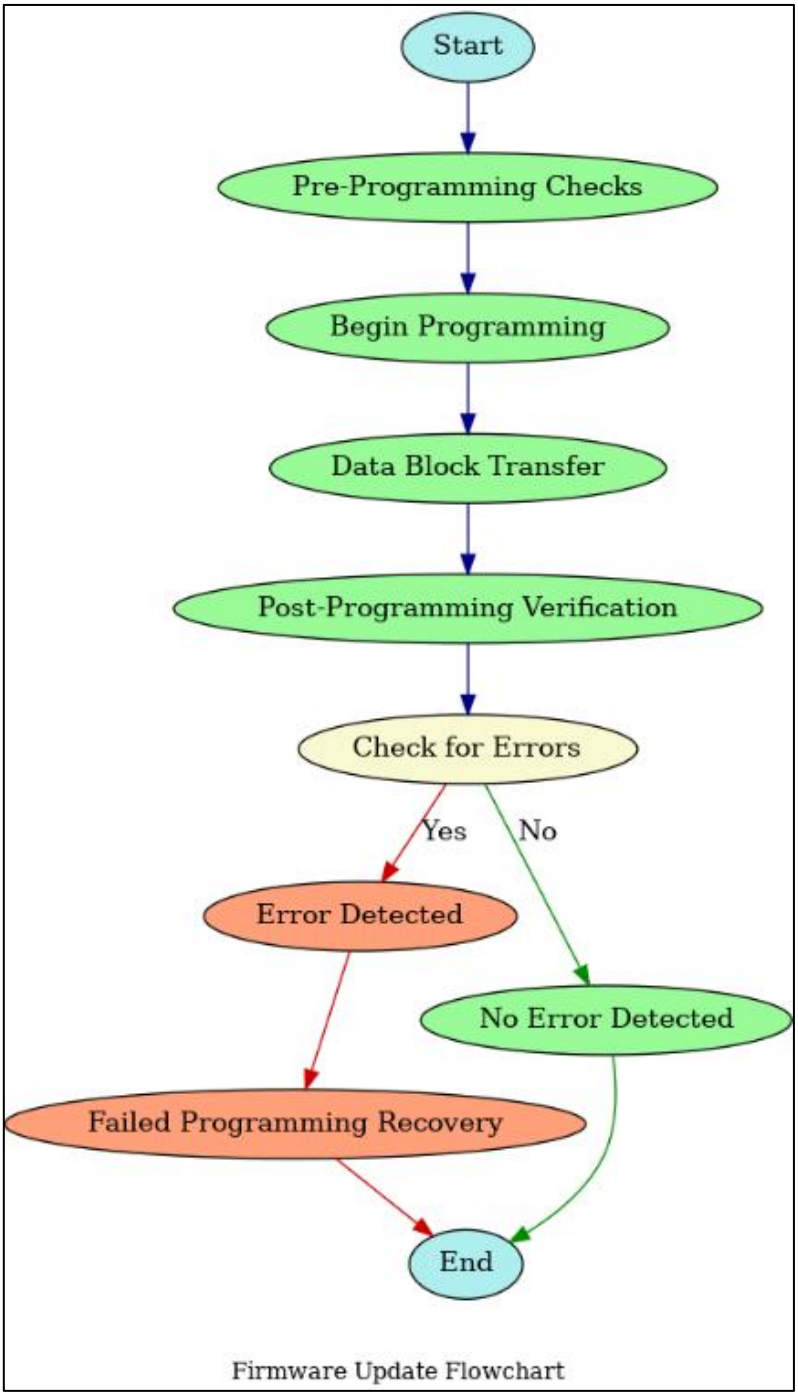
Post-Programming Procedure
Purpose: To validate the new programming, restore configurations if needed, and safely conclude the programming session.
Steps:

- Reset/restart the ECU to apply the new firmware.
- Verify successful update and functionality of the ECU.
- Restore any previous configurations and perform recalibrations.

Failed Programming Procedure
Purpose: To recover the ECU to a safe or previous working state in case of programming failure.
Steps:

- Retry the programming process if possible.
- Restore the ECU to its original state using backups, if available.
- If restoration fails, proceed with ECU replacement or specialized recovery.

5. FIRMWARE UPDATE FLOWCHART



Firmware Update Flowchart

6. WORKFLOW

➢ ECU SPECIFIC UPDATES

The update package targets three specific ECUs: Body Control Module (BCM), Anti-lock Braking System (ABS), and Head Unit. Each ECU has associated firmware and configuration files detailed for update:

Firmware Attributes: fileName, RequestCANId, ResponseCANId, fileSize, UpdateVersion, checksum
UDSFlowConfiguration Attributes: fileName, fileSize, checksum
These attributes are consistent across the following ECUs:

- BCM (Body Control Module)
- ABS (Anti-lock Braking System)
- HeadUnit

➢ COMPATIBILITY REQUIREMENTS

- Minimum Battery Level (30%): The device needs at least 30% battery power before the update process can begin. This ensures that the device won't shut down during the update, which could lead to corrupted software or incomplete installation.
- Minimum Signal Strength (60%): A minimum signal strength of 60% is required, likely to ensure a stable connection for downloading the update files. This is crucial for avoiding download errors or data corruption.
- Roaming Allowed (false): The update process is not permitted while the device is roaming, possibly to prevent excessive data charges or due to the unreliability of roaming connections.

➢ INSTALLATION INSTRUCTIONS

- Reboot Required (true): The update process necessitates restarting the device. This is common for firmware updates, as it allows the new software to fully replace the old version and resolve in-memory inconsistencies.
- Estimated Time (600 seconds): The update is expected to take approximately 10 minutes. This gives users an idea of how long the device will be unavailable.
- Instructions: Detailed guidance for the user on how to proceed with the update, emphasizing not to turn off the device during the process to avoid any potential issues.

➢ SECURITY SETTINGS

- Encryption Enabled (true): The update will be encrypted, ensuring that the data cannot be intercepted or tampered with during transmission.
- Authentication Required (true): Authentication is necessary before the update can be applied, which helps to prevent unauthorized software modifications.
- Checksum Validation (true): This involves verifying the integrity of the downloaded update files against a known checksum value. It's a crucial step to ensure the update is not corrupted.

➢ LOGGING SETTINGS

- Log Update Events (true): The device will log events related to the update process, which can be useful for troubleshooting issues or verifying the update's success.
- Send Update Reports (true): The device will send reports about the update to a designated server or service. This can help with monitoring the update's rollout and identifying any widespread issues.