SUMS Framework and Architecture

SUMS stands for **Software Update Management System**, a critical framework used in automotive cybersecurity (as per ISO 24089). It ensures **secure and reliable software updates** for ECUs (Electronic Control Units) in vehicles. SUMS is often implemented alongside **CSMS (Cybersecurity Management System)** defined in ISO/SAE 21434.

1. SUMS Framework

The SUMS framework defines:

- **Processes** for the management of software updates throughout a vehicle's lifecycle.
- **Roles and responsibilities** for OEMs (vehicle manufacturers), suppliers, and other stakeholders.
- **Cybersecurity measures** to prevent malicious updates and unauthorized access.

Key Objectives:

1. Ensure only **authorized and verified updates** are installed.
2. Provide **traceability** of updates (who, what, and when).
3. Ensure **integrity, authenticity, and availability** of update packages.
4. Maintain **compliance** with regulatory requirements like UNECE R155 and R156.

2. SUMS Architecture

A typical SUMS architecture includes the following components:

A. Backend Server

- Hosted by OEMs to manage update packages.
- Performs signing of update files.
- Maintains a database of version history and patch details.
- Manages secure communication with vehicles.

B. Vehicle Update Manager (in-vehicle)

- A dedicated ECU or service that receives, validates, and distributes updates.
- Verifies **digital signatures** and checks integrity (e.g., using hashes).
- Coordinates with other ECUs for OTA (Over-the-Air) updates.

C. Communication Layer

- **OTA (Over-the-Air)** or via physical media (USB, service centers).
- Secure protocols like **TLS, HTTPS, or secure CAN**.

D. Security Features

- **Secure Boot**: Ensures ECUs only run trusted software.
- **Cryptographic Validation**: Digital signatures and certificates.
- **Rollback Protection**: Prevents downgrading to vulnerable software.

3. SUMS Process Flow

1. **Update Creation** – OEM creates a new software update.
2. **Signing and Packaging** – Update is digitally signed to ensure authenticity.
3. **Distribution** – Update is pushed OTA or via dealership/service centers.
4. **Verification in Vehicle** – SUMS verifies the signature, integrity, and compatibility.
5. **Deployment** – Update is applied to the targeted ECU(s).
6. **Reporting** – Update status is sent back to OEM.

4. Relation to Standards

- **UNECE R156** – Regulation for software updates and SUMS.
- **ISO 24089** – Standard detailing software update engineering in road vehicles.
- Works closely with **CSMS (ISO/SAE 21434)** to ensure security during updates.

5. SUMS(R156) - Process Requirements

configuration control for recording the hardware and software versions relevant to a vehicle type, including integrity validation data for the software

Identifying the software and hardware on a vehicle relevant to a specific UN regulation and tracking if that software changes (the RXSWIN concept)

Verifying the software on a vehicle component is what should be there

Identifying interdependencies of systems, particularly with respect to software updates Identifying target vehicles and verifying their compatibility with an update

Assess if a software update will affect type approvals or other legally defined parameters for a given target vehicle (including adding or removing functionality) u Assess if an update will affect the safety of safe driving of a vehicle

Inform vehicle owners of updates

6.SUMS(R156) - Type Approval Requirements

Software Update Management System is in place and its application to vehicles on the road is available

Protect SW Update delivery mechanism and ensure **integrity and authenticity**

Handling **RxSWIN(Software identification numbers)** in the vehicle

For **Over-The-Air** software updates:

1. Restore function if update fails
2. Execute update only if sufficient power
3. Ensure safe execution
4. Inform users about each update and about their completion
5. Ensure vehicle is capable of conducting update
6. Inform user when a mechanic is needed

7.SUMS(R156) - RxSWIN

RX Software Identification Number (RXSWIN):

1. Dedicated identifier, defined by the vehicle manufacturer
2. Representing information about the type approval relevant software of the Electronic Control System contributing to the Regulation N ° X type approval relevant characteristics of the vehicle

3. In case the type approval relevant software is modified by the vehicle manufacturer, the RXSWIN will be updated leading to a type approval extension. Modification of software are type approval relevant if they lead to a modification of the vehicle type according to this regulation or if functionalities are extended

Requirements :

1. Uniquely identifiable
2. Easily readable in a standardized way(e.g. OBD port) u Protect the RXSWINs and/or software version(s) on a vehicle against unauthorized modification

Core Modules of SUMS

1. Update Management Module

This is the **central module** of SUMS responsible for controlling the entire update process.

Key Functions:

- **Update Orchestration:** Determines which ECUs require updates and schedules the process.
- **OTA (Over-the-Air) Management:** Handles secure download and delivery of update packages to the vehicle.
- **Rollback Management:** Provides mechanisms to revert to a previous version if the update fails.
- **Version Control:** Tracks the current and historical versions of each ECU's software.
- **Compatibility Check:** Ensures that the new software is compatible with the vehicle's existing configuration.

2. Dependency Management Module

This module ensures that updates are installed without **breaking ECU dependencies**.

Key Functions:

- **Dependency Analysis:** Checks for interdependencies among different ECUs (e.g., powertrain ECU requiring a specific version of a transmission ECU firmware).
- **Update Sequencing:** Defines the correct order of updates to avoid conflicts.
- **Impact Assessment:** Evaluates how one ECU's update will affect other ECUs and vehicle functions.
- **Conflict Resolution:** Automatically resolves or flags issues where incompatible versions are detected.

3. Validation Module

Ensures the **integrity, authenticity, and safety** of software updates before and after deployment.

Key Functions:

- **Digital Signature Verification:** Confirms the update is signed by the OEM (authenticity).
- **Hash & Integrity Check:** Ensures the update file is not tampered with during transfer.
- **Pre-Update Simulation:** Performs testing or sandbox validation before applying the update.
- **Rollback Testing:** Ensures the system can return to a stable version if the update fails.
- **Functional Validation:** Verifies that the updated ECU performs correctly post-installation.

4. Logging & Reporting Module

This module provides **traceability and auditability** of the entire update lifecycle.

Key Functions:

- **Update Logs:** Records every step of the update process (download, verification, installation).

- **Event Monitoring:** Tracks errors, failures, or anomalies during update.
- **Reporting to OEM Backend:** Sends detailed update reports (success/failure status, ECU version info).
- **Audit Trail:** Provides logs for compliance with regulations like **UNECE R156** and **ISO 24089**.
- **Security Event Logging:** Records security-related events like failed signature validation attempts.

Functional Flow of These Modules

1. **Update Management** receives a new package from the backend.
2. **Dependency Management** checks compatibility across ECUs.
3. **Validation Module** verifies signatures, hashes, and performs pre-update checks.
4. **Logging & Reporting** records each step and sends status back to the backend.

SUMS Interface with Vehicle and Backend

The **Software Update Management System (SUMS)** works as a **bridge between the backend (OEM servers)** and the **in-vehicle ECUs**, ensuring secure and reliable communication for software updates.

1. Interface Between SUMS and Backend (OEM Server)

Role of Backend:

- The OEM backend manages **update preparation**, **digital signing**, and **distribution** of update packages.

- It communicates with SUMS in the vehicle to deliver updates securely.

Key Interface Elements:

- **Secure Communication:**
  - Uses TLS/HTTPS or VPN tunnels to ensure **encrypted and authenticated** data transfer.
  - Backend sends digitally signed update packages.
- **Metadata Exchange:**
  - Includes version info, update size, dependencies, and checksums.
- **Update Scheduling:**
  - Backend can trigger OTA (Over-the-Air) updates or schedule updates when the vehicle is idle.
- **Telemetry & Feedback:**
  - SUMS sends update status, error reports, and vehicle software inventory back to the backend.

2. Interface Between SUMS and Vehicle (ECUs)

Role of SUMS in Vehicle:

- SUMS acts as the **Update Manager ECU** (sometimes part of the Telematics Control Unit).
- It coordinates the download, verification, and distribution of updates to other ECUs in the vehicle.

Key Interface Elements:

- **Secure Download and Distribution:**
  - SUMS receives updates from the backend and pushes them to individual ECUs.
  - Uses **secure CAN**, **DoIP (Diagnostics over IP)**, or **UDS (Unified Diagnostic Services)** protocols.
- **Validation Before Deployment:**
  - Verifies the **integrity and authenticity** of each update package using cryptographic checks.
- **Dependency and Compatibility Checks:**
  - SUMS checks ECU interdependencies before installing updates.
- **Rollback and Recovery:**

- SUMS ensures recovery to a stable software version if the update fails.

3. Typical Data Flow (Backend ↔ SUMS ↔ Vehicle ECUs)

1. **Backend Server:**
    a. Prepares and digitally signs update packages.
    b. Sends update package and metadata to SUMS in vehicle via secure OTA.
2. **SUMS in Vehicle:**
    a. Downloads and stores the update.
    b. Validates the package (hash/signature).
    c. Communicates with ECUs over **CAN or Ethernet** to deploy updates.
3. **ECUs:**
    a. Receive the software update via SUMS.
    b. Perform internal checks and report status back to SUMS.
4. **Feedback:**
    a. SUMS sends logs and update success/failure reports to the backend.

4. Security Considerations for Interfaces

- **Digital Certificates & PKI (Public Key Infrastructure)** for authentication.
- **Freshness values/nonces** to prevent replay attacks.
- **Message Authentication Codes (MAC)** for data integrity.
- **Fail-safe design** in case of communication interruption.

Software Update Management System (SUMS) is a cloud-based software update management system (SUMS) that helps vehicle manufacturers (OEMs) comply with UN Regulation No. 156. It provides a secure and efficient way to manage software updates throughout the vehicle's lifetime.

SUMS automates and assists in many of the tasks involved in software update management, such as:

- Creation of vehicle type and maintaining associated software package in configuration management.
- Identifying and tracking software changes
- Assessing the impact of software updates on regulation parameters & type approval.
- Collection of software packages and R156 regulation-relevant information from suppliers
- Packaging and distributing software updates to target vehicles via OTA or Garage diagnostics tools.
- Ensures the security of the software updates.
- Verifying the successful installation of software updates
- Maintain OEM Vehicles database with the last known R156 regulation relevant information.

1. Asset Management in SUMS

Asset management within a Software Update Management System (SUMS) is about creating a complete inventory and understanding of all the software and hardware components within a vehicle and its connected systems. This management enables secure, traceable, and efficient software updates.

Key Aspects:

- **Inventory Management:** Maintains a centralized list of all ECUs, firmware versions, and connected components.
- **Configuration and Version Control:** Tracks all version changes and ensures compatibility among components.
- **Unique Asset Identification:** Assigns unique identifiers to software and hardware elements for targeted updates.
- **Lifecycle Oversight:** Manages the deployment, updates, replacements, and end-of-life stages of components.
- **Security Assurance:** Ensures that only authorized and validated updates are implemented.
- **Dependency Analysis:** Detects interdependencies between modules to prevent incompatibility issues.

Asset management supports auditability, helps reduce risks associated with software failures, and guarantees regulatory compliance.

2. Soft Item Identification in SUMS

Soft item identification ensures that every software component is uniquely recognized and tracked. This concept is central to maintaining regulatory compliance with standards such as **UN R156** and **ISO 24089**.

Key Elements:

- **RXSWIN (Regulation X Software Identification Number):** Provides unique identification of type-approved software.
- **Digital Signatures:** Enhance authenticity and tamper-proofing of software.
- **Version Mapping:** Associates software identifiers with specific ECUs and system configurations.
- **Traceability:** Allows for a complete history of updates and changes, ensuring accurate rollback if needed.

This process ensures reliable update delivery and prevents mismatched or unauthorized software installations.

3. Asset Categorization and Criticality Mapping

Categorizing assets and mapping their criticality is vital for prioritizing updates and ensuring that the most safety-critical components receive attention first.

Techniques for Categorization:

- **Functional Grouping:** Categorizing assets by domain (e.g., powertrain, infotainment, ADAS).
- **Criticality Assessment:** Evaluating the potential impact of asset failure using risk factors (severity, likelihood, detectability).
- **Criticality Scoring:** Assigning numerical values to rank asset importance.
- **Dependency Mapping:** Understanding cross-module dependencies to avoid failures during updates.

Benefits:

- Enables risk-based update scheduling.
- Ensures robust testing for high-criticality modules.
- Supports regulatory audits with structured asset hierarchies.

4. Organizational and Process Structure of SUMS

SUMS requires a well-defined organizational framework complemented by robust process structures to handle updates safely and effectively.

Organizational Roles:

- **Software Owner:** Oversees the lifecycle and integrity of software components.
- **Security Reviewer:** Verifies cryptographic and cybersecurity measures.
- **Approver:** Grants formal approval for deployment of updates.

Process Framework:

- **Planning:** Defines update campaigns and their goals.
- **Packaging & Testing:** Creates and validates update packages.
- **Secure Deployment:** Ensures secure OTA (Over-The-Air) or in-workshop updates.
- **Monitoring:** Tracks updates post-deployment for feedback and issue resolution.

This dual approach ensures alignment with automotive standards like UN R156 and ISO 24089.

5. Lifecycle Integration with ASPICE and Agile

Integrating SUMS lifecycle activities with **ASPICE** (Automotive SPICE) and Agile workflows is essential to achieve both compliance and flexibility.

Key Integration Points:

- **Agile-SPICE Mapping:** User stories and sprints are mapped to ASPICE V-model work products.
- **Incremental Compliance:** Each sprint includes ASPICE-relevant deliverables such as requirement verification or traceability.
- **Tool Support:** Platforms like Jira, Codebeamer, and Polarion streamline traceability and documentation.
- **Feedback Loops:** Agile methods enable quick feedback and iterative improvements.

Advantages:

- Combines strict compliance with faster time-to-market.
- Improves software quality and testing efficiency.
- Ensures audit-readiness with continuous documentation.

6. SUMS Roles and Responsibilities

Clear role definition ensures accountability and effective execution of updates.

- **Software Owner:** Manages version control, testing, and update validation.
- **Security Reviewer:** Reviews cryptographic methods, potential vulnerabilities, and secure delivery methods.
- **Approver:** Authorizes updates after validating safety, compliance, and performance.
- **Project Manager (optional):** Coordinates across teams for smooth update rollouts.

Conclusion

SUMS serves as the backbone of modern automotive software management. By integrating asset management, criticality mapping, defined organizational roles, and lifecycle alignment with ASPICE and Agile, SUMS ensures safe, secure, and efficient updates. Its structured approach not only enhances operational efficiency but also guarantees compliance with evolving automotive standards like UN R156 and ISO 24089.