Automotive OTA

The potential and the challenge

Dr. Walter J. Buga, CEO

AUTOMOTIVE LINUX SUMMIT

Tokyo, Japan
May 28, 2013

arynga

OTA Potentials After Car Sale

- Cost effective update of vehicle software and firmware
- Ability to manage much shorter lifecycle of software and firmware
- Tracking ECU software down to the VIN, including software dependencies
- Reduce warranty costs for OEM's
- Improve dealer profitability
- Enhance driver satisfaction and promote brand loyalty
- Allow for new revenue opportunities for vehicle OEM's

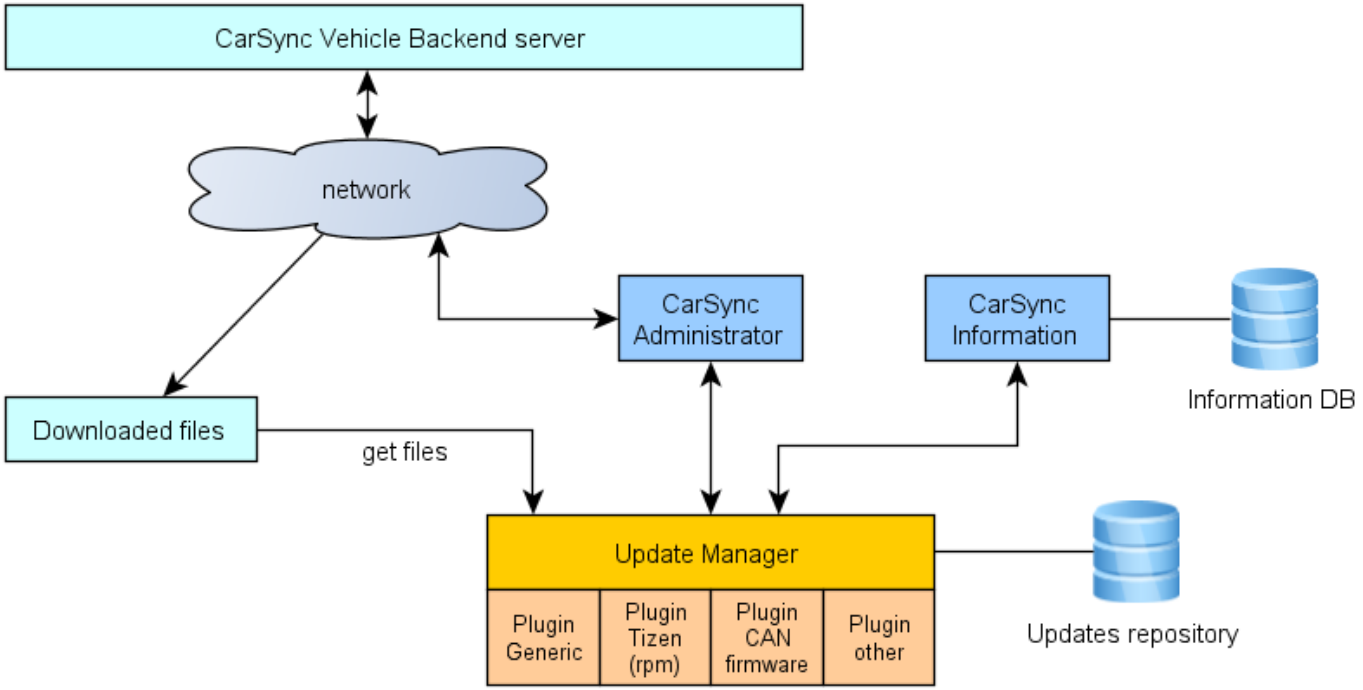
OTA Requirements

- Safety
  - no image can be downloaded until it has been verified as the correct image for the specific ECU subcomponent; utilizing CRCs, checksums, and other release version compatibilities used.
- Security
  - data should be sent utilizing security protocol and encryption mechanisms.
- Reliability
  - should utilize data aggregation over one or more connection sequences (with no loss of data) with final reassembly done only after aggregation has completed.
- Scalability
  - Should be design to support large number of vehicles by providing multicast and broadcast capabilities in addition to unicast
- Flexibility
  - Should support national and regional requirements
  - Should provide an efficient mechanism to support multiple car models and car groups

Memory and other challenges

- In order to achieve reliability and stability when remotely updating ECUs additional RAM and Flash must be added to one or more of the ECUs within the vehicle; particularly if the process is transparent to the vehicle operations.
- The Cloud Server distributing the releases must be design and implement as configuration management system that act as a distribution center with proper intelligence and knowledge.
- The Vehicles must have built-in intelligence to be capable of receiving and distributing incoming ECU releases
  - This is particularly true if the ECUs are expected to have rollback capabilities and/or can receive updates via one-way communication mechanisms such as Satellite Radio or USB Drives
- The following functionality must be exposed by the bootloader:
  - Means of reading the currently running software version
  - Means of changing the software version to be run after reboot
  - Means of reading whether an error has occurred when booting the firmware
- Partitioning and storage requirements
  - Update Manager must keep track of replaced versions of firmware and software packages for the use in offline rollback scenario. This requirement implies that considerable storage may be required.
- For the purpose of firmware update, it is required to have a least 2 partitions for keeping firmware versions.
  - The bootloader shall be able to boot either partition as instructed.

OTA Update Process and Architecture



1. The OEM/Tier1 makes a new Release Package (RP) available on the backend server.
2. Once Administrator process identifies that a new RP is available, the required files get downloaded and stored on the In-Vehicle server.
3. The Update Manager process distributes the updates using the most appropriate methods for each package as specified.

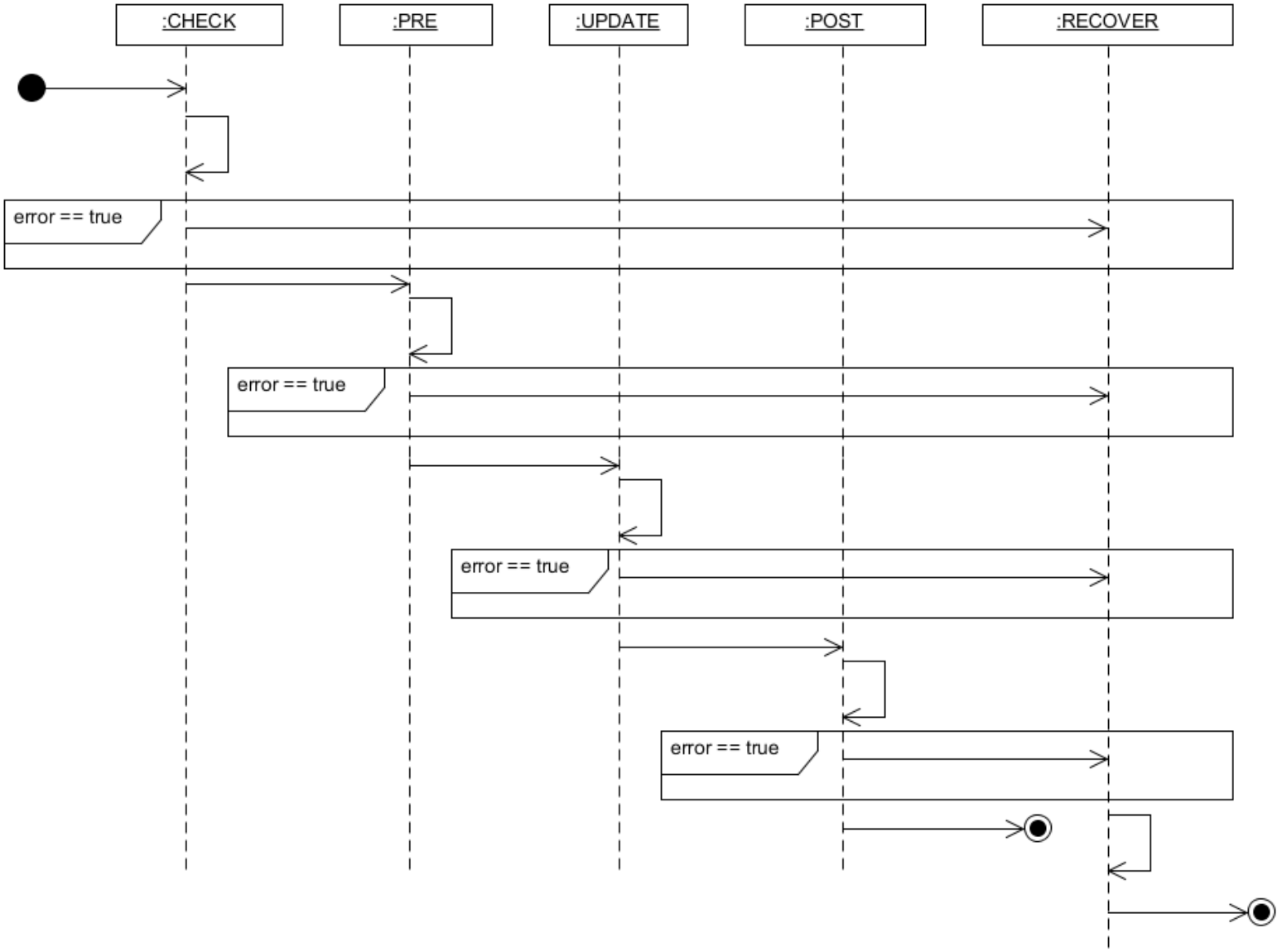
A number of plugins can be used to support various environments, formats and update methods, and to provide proper adaptation:

- DUMM Genivi
- RPM, DEB
- Tizen, Ubuntu, QNX

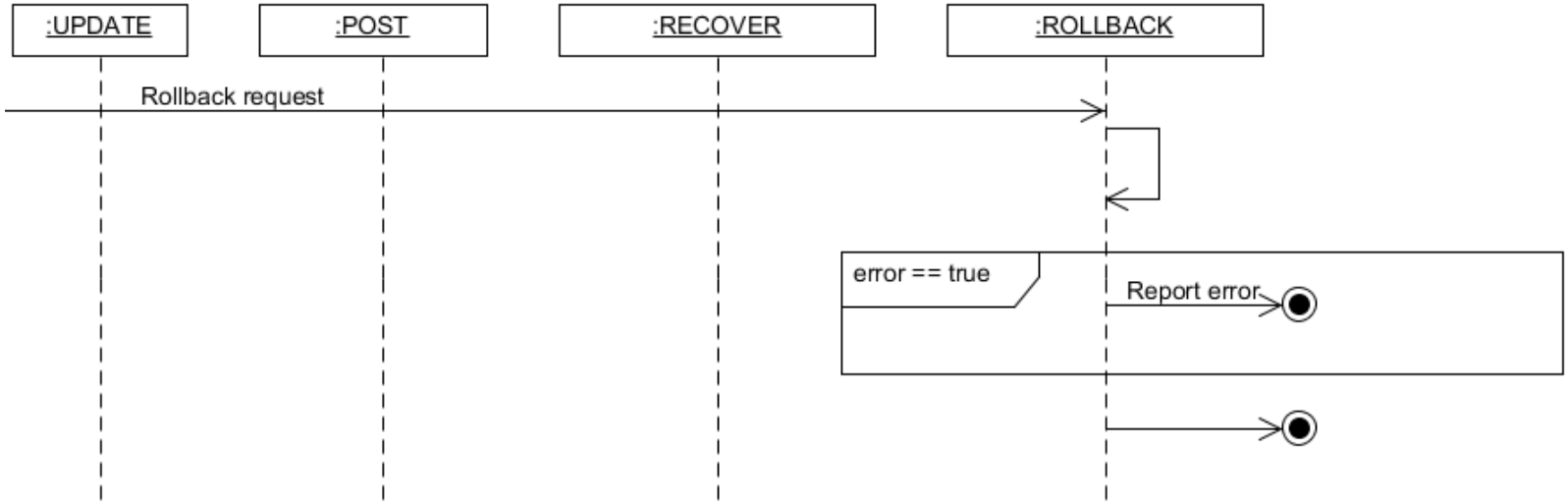
Update Manager Details

- End-to-End OTA solution provides means and tools for preparation and delivery of the update files, as well as it executes the update process by enforcing correct update states sequence.
- OEM/Tier1 is responsible for providing the update that can be successfully applied, with the relevant applications remaining stable and operational.
  - Steps such as data format changes, restating of relevant processes, as well as rollback support need to be properly provided in the scripts that are run within the scope of given package format.
- The update process is considered a transaction, hence all packages must be successfully updated or the update fails and is rolled back.
- Update Manager has no knowledge of the required modifications to the data or resources used by respective packages, neither of the actual operations that need to be executed to ensure the proper level of update integrity.
- Update Manager provides means of logging, so that each step is logged, allowing for the update process to be traced, or even span across device reboots.
  - Tools for error reporting are included.
- The rollback scenario may be executed if update process failed, or can be triggered on demand.
  - The rollback to last known configuration shall be possible even with limited connectivity, as the update might have introduced a change that caused connectivity loss.

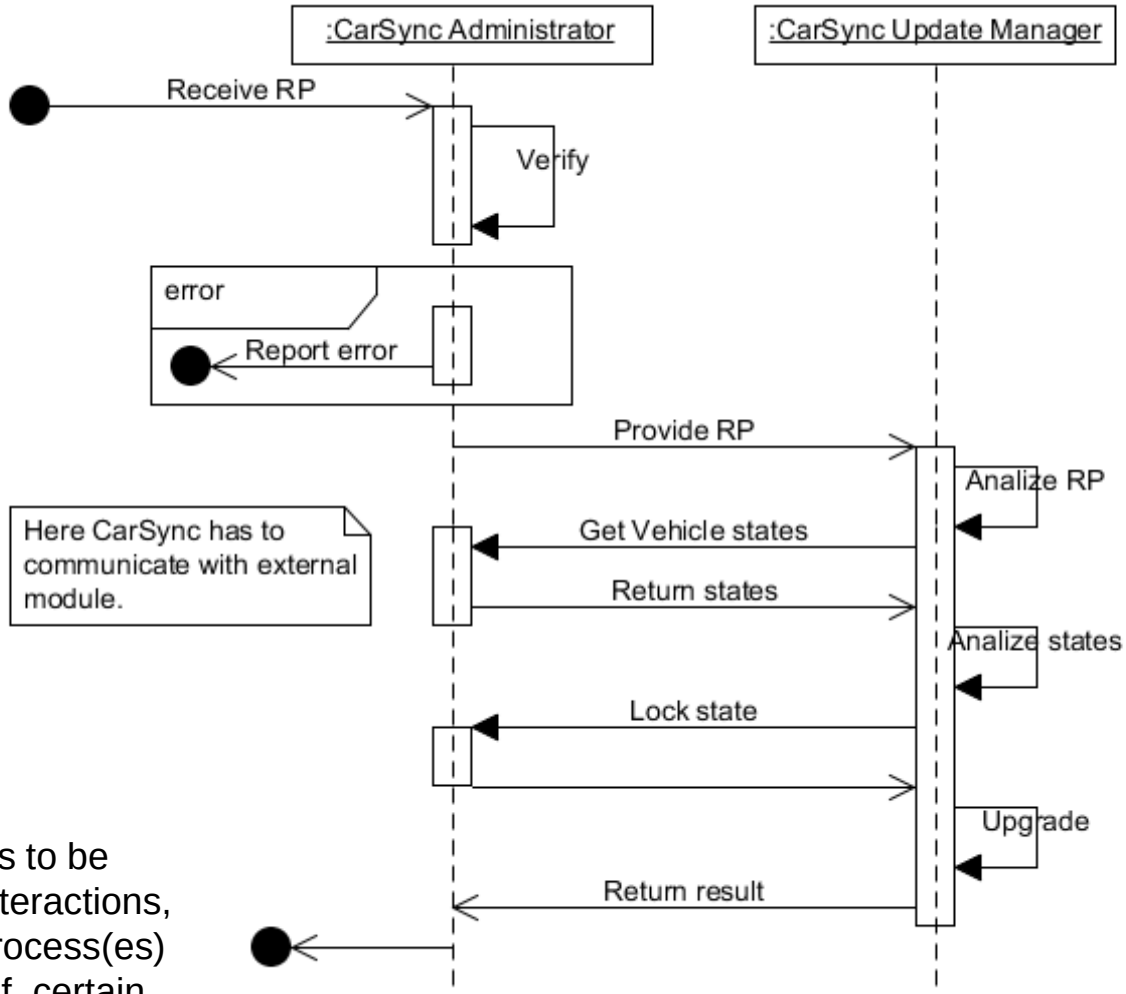
Update Steps and Processes



Rollback Steps and Processes

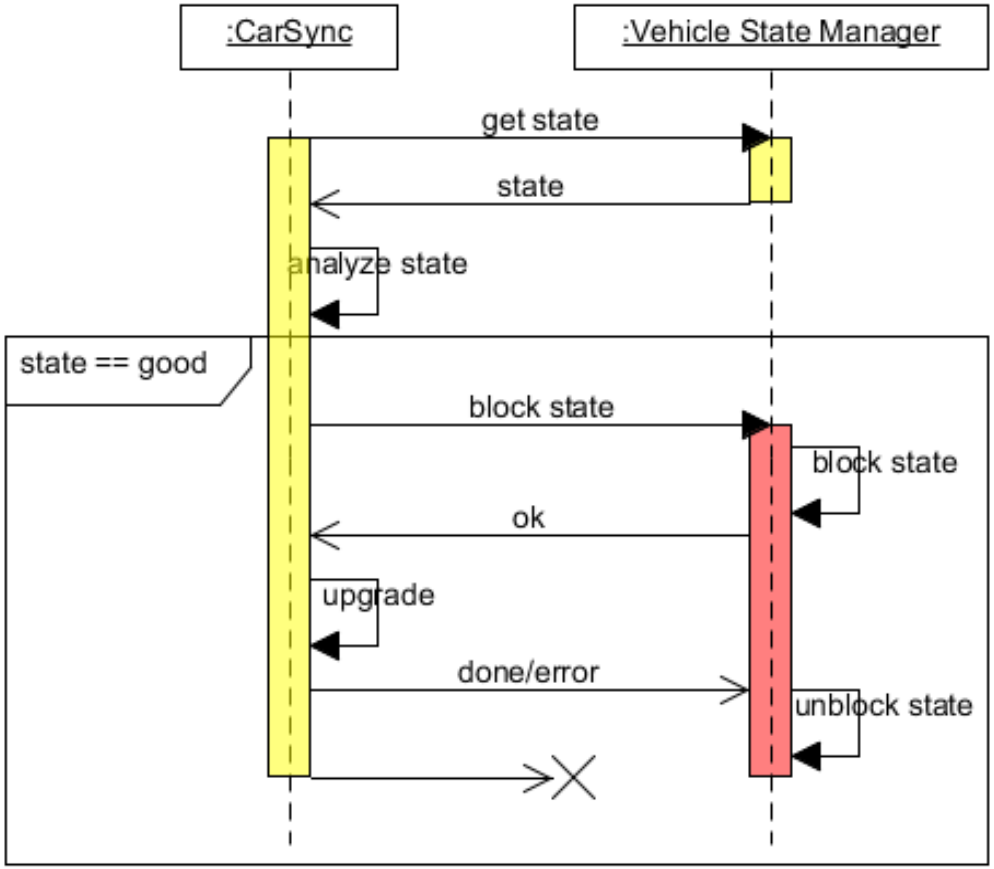


Admin & Update Interactions



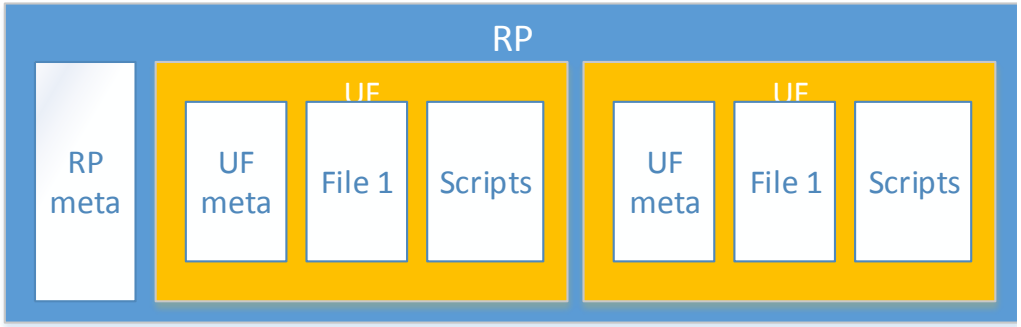
A vehicle state needs to be considered during interactions, as certain update process(es) will only be allowed if certain conditions are met.

State Managers Interactions



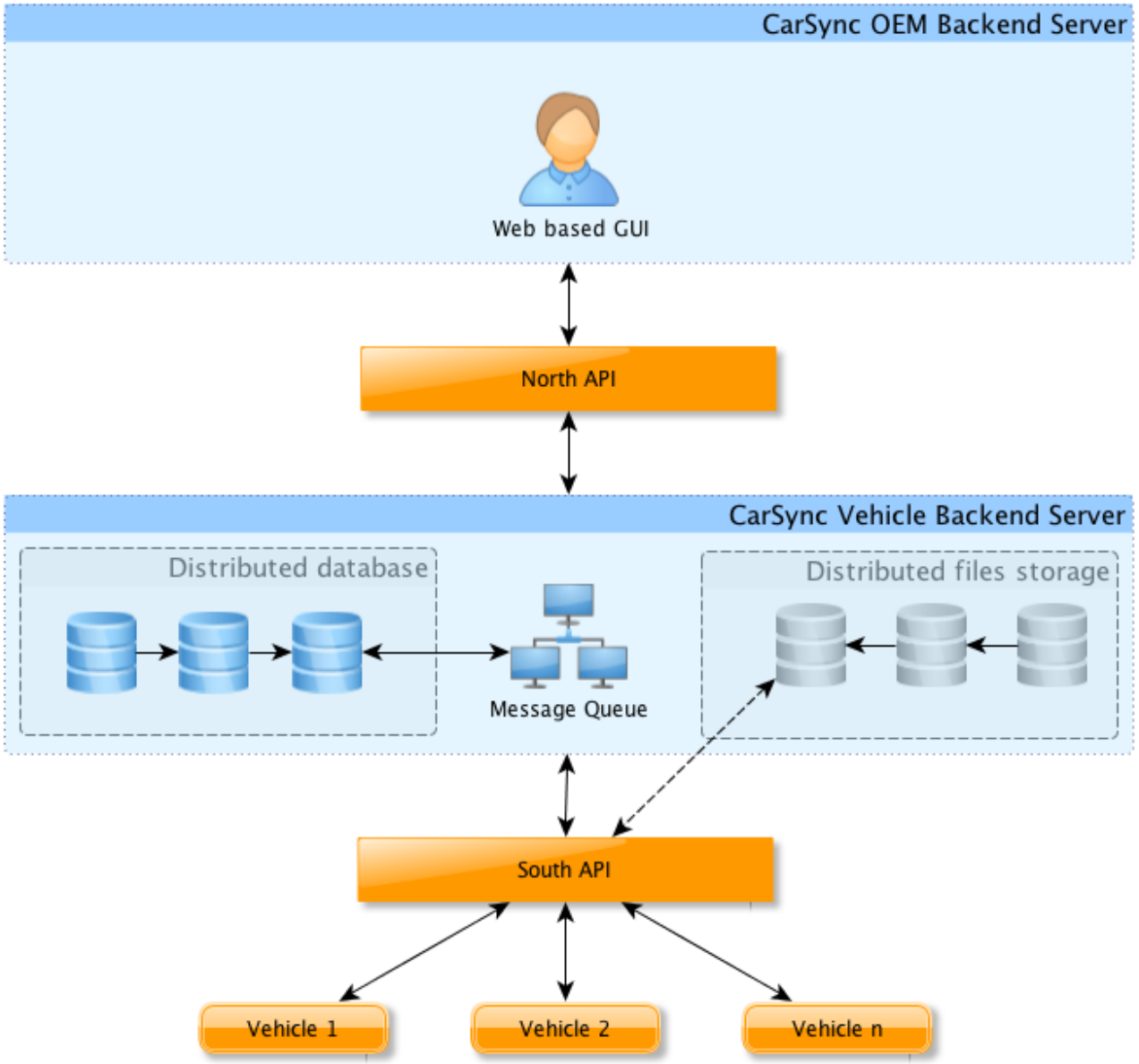
Vehicle State Manager (VSM) is responsible for management and communication of vehicle states.

Release Package (RP) Concept

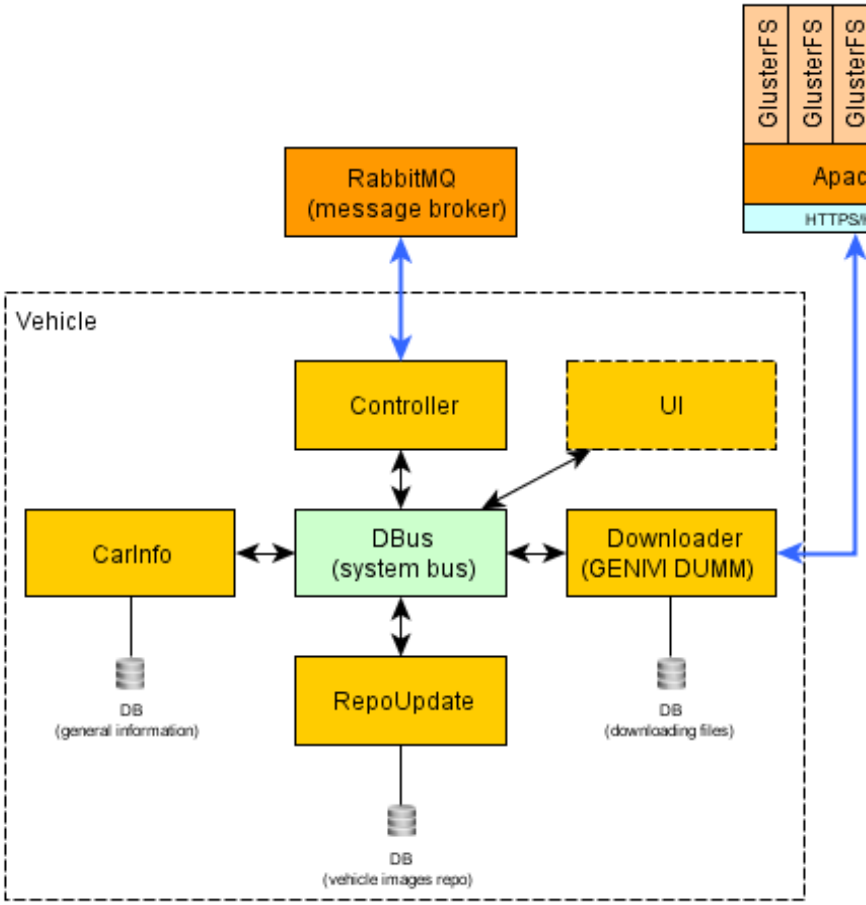


- Each update will be delivered in form of a RP, consisting of Update Files (UFs).
  - An Update File is an abstract name of a file that has to be delivered to a device.
  - RP is a virtual container that includes one or more UFs in one update.
  - Every RP has a set of properties that include information such as version, range of VINs etc.
  - Every Update File is downloaded individually.
- RP is versioned, where an RP of subsequent version number may update, downgrade, add or remove packages from the system.

System Architecture



In-Vehicle Architecture

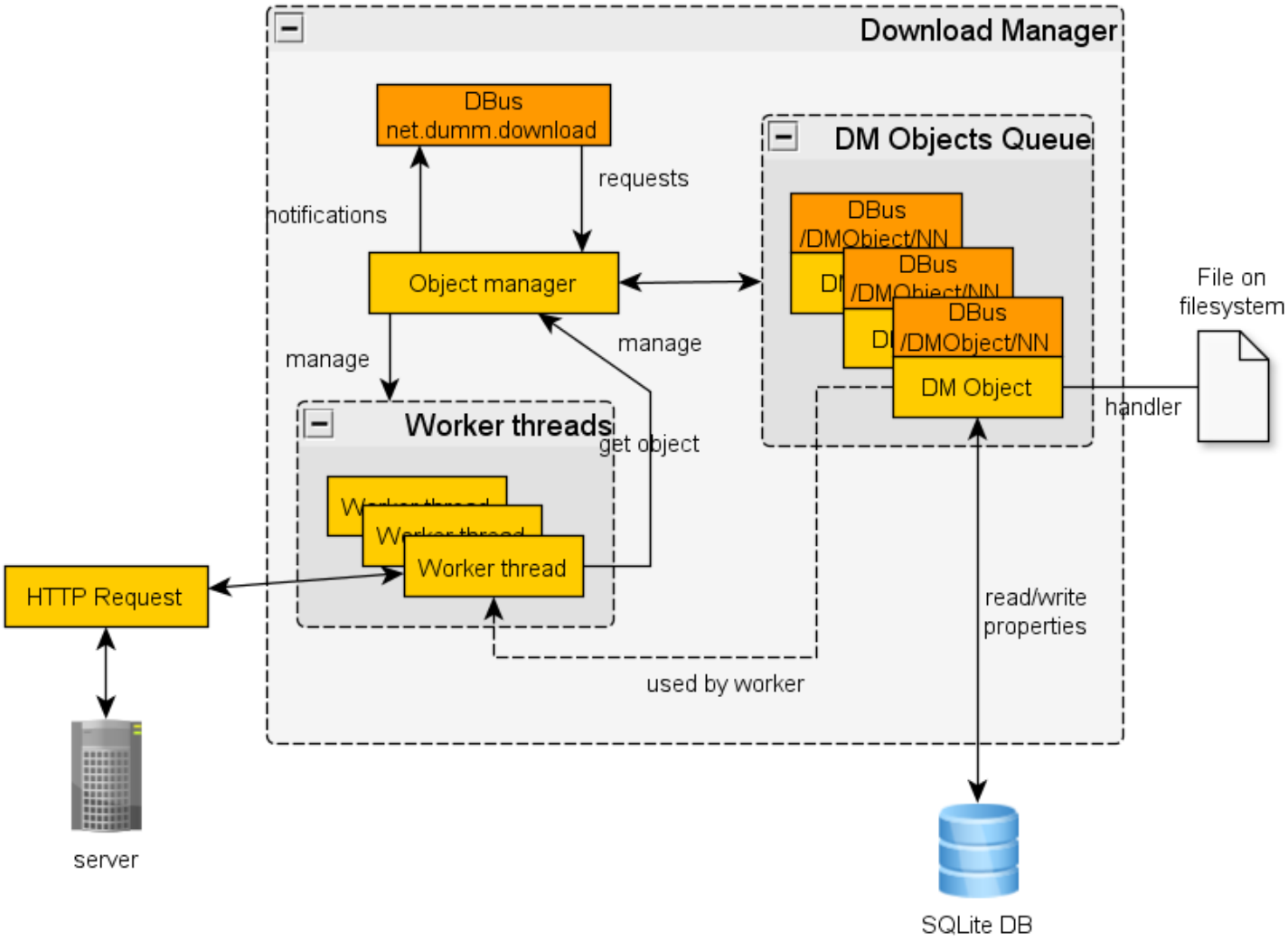


- **Controller** is responsible for controlling all other components in the vehicle and:
  - Exchanging messages
  - Requesting a file to be downloaded;
  - Starting the update process;
  - Displaying information via the UI to the user.
  - Handling error cases.
- **CarInfo** is responsible for providing vehicle information including the data can be retrieved by using CAN Bus, and:
  - H/W and S/W configurations
  - update/version history information
- **Download** is responsible for downloading files (updates) from Cloud Server.
  - it is compliant with GENIVI DUMM
- **RepoUpdate** is responsible for the update process and:
  - managing local image storage/data base
  - temporary updates storage

GENIVI DUMM (Open Source)

- Download Upload Messaging Manager (DUMM) is a component for exchanging data (download/upload files, exchanging messages) with remote servers over HTTP
- DUMM has three main subcomponents:
  - Download Manager (DM)
  - Upload Manager (UM)
  - Messaging Manager (MM)
- DUMM software was contributed by Arynga

GENIVI DUMM Architecture

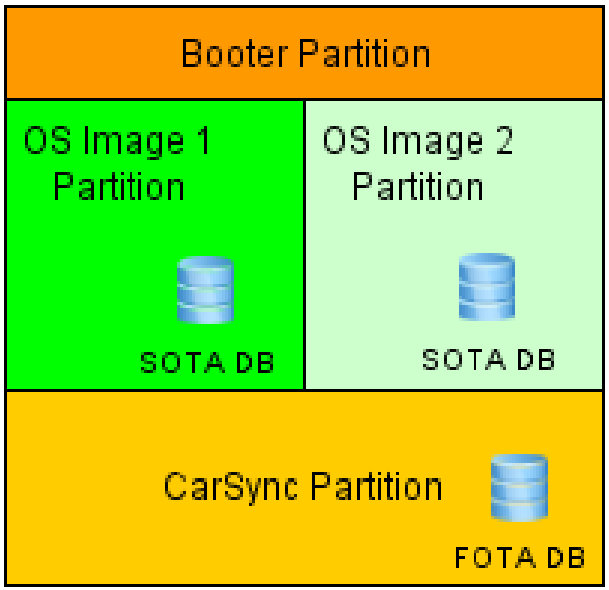


DUMM Development Status

- Finish current development tasks
  - Run DUMM as daemon
  - Add all features related to HTTP requests
  - Add support for all properties in DM objects
  - Verify and clarify DBus interface with others
- Add Uploading Manager
- Add Messaging Manager
- Integration with connman – GENIVI connection manager

FOTA vs SOTA

- Firmware and Software Release Packages are versioned separately, but they do interact with each other, hence dependencies between the two need to be considered.
- A Software Release Package describes the state of the software components that make up certain release.
- Firmware Release Package of a component that is versioned with Software RPs has to be considered as a frozen snapshot of the Software RP.



•Update Manager must keep track of replaced versions of firmware and software packages for the use in offline rollback scenario. This requirement implies that considerable storage may be required.
•For the purpose of firmware update, it is required to have a least 2 partitions for keeping firmware versions.
    •The bootloader shall be able to boot either partition as instructed.

Thank You

Dr. Walter J. Buga, CEO
Arynga Inc.
4225 Executive Square, Suite 400
La Jolla, CA 92037
walter@arynga.com
www.arynga.com