Remote Vehicle Interaction Architecture

An open source solution for the automotive industry

Magnus Feuer - Jaguar Land Rover

Background

- **80% of connected vehicle functionality shared across platforms**
  Core IVI and Server functionality are similar, regardless of vendor. The final 20% are the services that defines the user experience.

- **A shared, open source platform will benefit OEMs**
  A joint architecture and reference implementation allows OEM to minimize cost, vendor dependencies, and security risks, letting them focus on applications and services that make a difference.

- **A shared, open source platform will benefit service providers**
  A common architecture allows service providers to easily port their products to additional OEMs, thus giving them a wider revenue stream from multiple vendors.

Objectives

- **Open Source reference implementation of RVI system**
  End-to-end system for PoC. Prepared for production upgrade.

- **Hosted by Linux Foundation**
  AGL publishes open source designs, specifications, and reference implementation to guarantee equal technology access.

- **Pluggable architecture**
  APIs are core – Reference system is an example.

- **Device, service, and connection agnostic**
  Any device shall be able to access services on any other visible device, with or without Internet connection.

Benefits

- **Enable new breed of 3rd party service providers**
  Open source implementation enables start-ups to develop in-vehicle apps and their corresponding backend services, and showcase the finished product to OEMs.

- **Alleviate vendor dependency**
  OEM can replace components at will, using either external or internal resources. All IP of the replaced components belongs to the OEM.

- **Wider talent pool**
  Large competence base provided through open source community, OEMs, app developers, and professional service vendors.

Feature Set

- **Peer-to-peer based**
  Two nodes can exchange services without an internet connection.

- **Provisioning**
  Services and nodes can be added and deleted from the system.

- **Authentication & authorization**
  All executed services are authenticated and authorized against certified credentials.

- **Service discovery**
  Services and applications can discover and invoke other services.

- **Invocation**
  Services can be remotely invoked over a sparsely connected network.

Upcoming Demos

- **Remote HVAC control**
  Pre-set the climate control of your vehicle from your mobile phone.

- **Software Over The Air (SOTA)**
  Transfer, install, and validate a software package from a backend server to an IVI unit.

- **Remote CAN bus monitoring**
  Remotely subscribe to specific CAN frames, and have them delivered to a backend server.

- **Remote control of IVI nav system**
  Use remote mobile device to setup POI in vehicle's navigation system.

Architecture

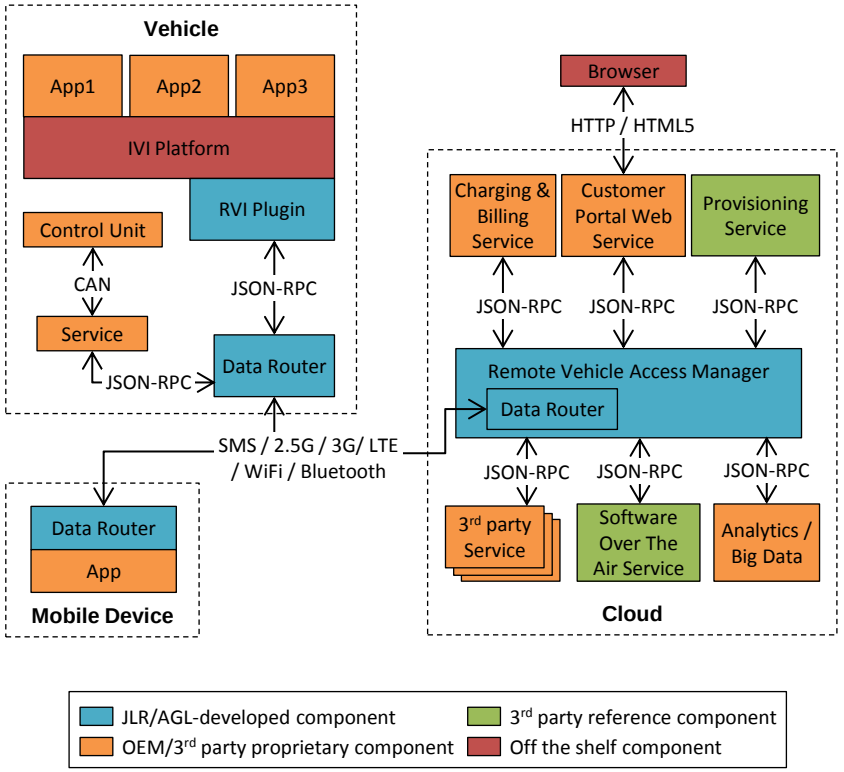
Architecture - Overview

- **API based**
  The API is the driving technology. Implementation is secondary.

- **Data Router commonality**
  Data Router connects all services on all devices.

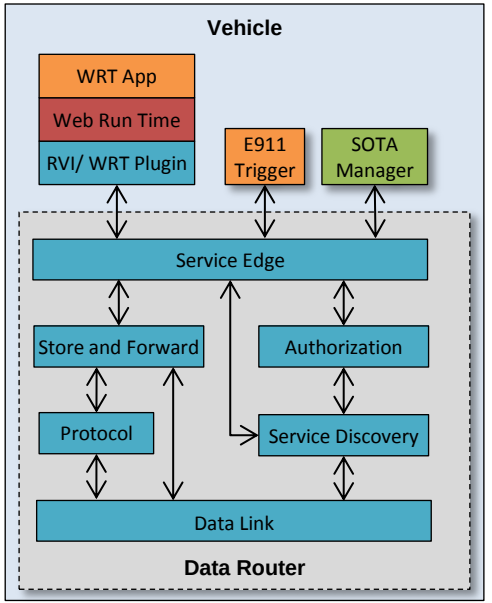
- **Mix of open and closed source**
  Components can be off the shelf, OSS, proprietary, or a combination of the above.

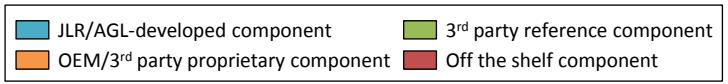
- **Network complexity shielding**
  A clean transaction API alleviates services and applications from connectivity concerns.



Architecture – Data Router

- **Service Edge**
  Handles all traffic toward locally connected services.
- **Authorization**
  Handles certificates and authorization for all traffic.
- **Store and Forward**
  Handles traffic holding when destinations are unavailable.
- **Data Link**
  Controls communication channels to other node.
- **Service Discovery**
  Identifies and locates local and remote services.
- **Protocol**
  Encodes and transmits traffic to other nodes.



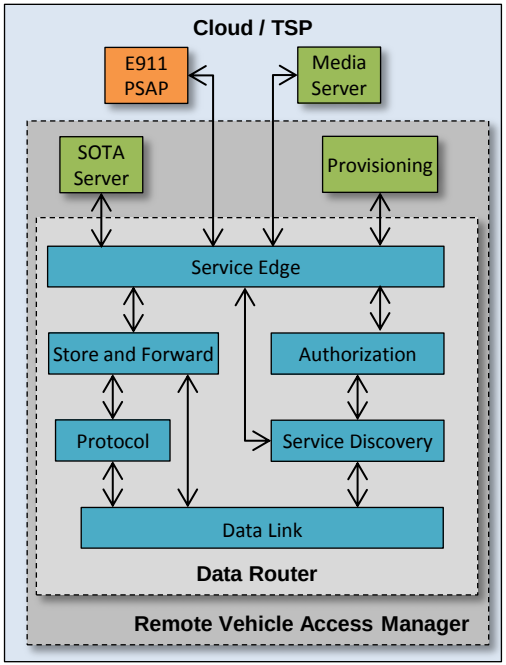


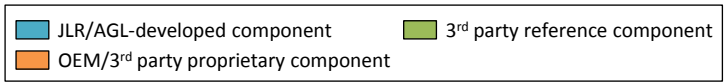
Architecture – Backend Server

- **Data Router**
  Standard deployment.

- **Provisioning**
  Creates and distributes certificates granting access rights to nodes.

- **SOTA server**
  Manages and distributes software packages to nodes.





Services

Services – Requirements

- **Global namespace for all services on all nodes, worldwide**
  All services on all provisioned devices must be addressable through a single schema.

- **Localized service discovery**
  Locally connected nodes must be able to discover each other's services without Internet access.

- **Zero configuration**
  No configuration outside authorization shall be needed for a newly deployed node to join the system.

- **Network agnostic**
  A service shall be accessed the same way, regardless of the communication method used.

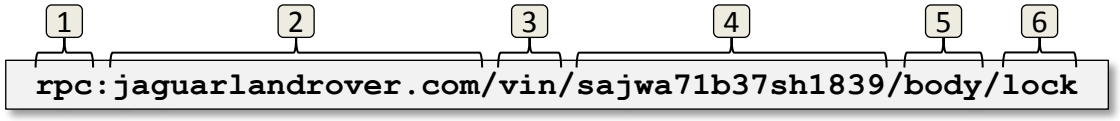
Services – Addressing

- **Single name space for all services**
  New services can be addressed by creating a unique name for them.

- **Service name identifies hosting node**
  Each service name, being unique across the system, carries enough information for Service Discovery to identify where the node can be found.

- **Hide network complexity**
  All service interaction with other services are done through the service name space, allowing the actual communication to be carried out behind the scenes.

Services – Service Name Example



#	Name	Description
1	Type	Traffic type accepted by service
2	Organization	Specifies a sub-section hosted by a specific entity
3	VIN sub-tree	Specifies sub section for all vehicles
4	VIN	Vehicle Identification Number
5	Service name	Name of service
6	Command	Command supported by service

Services - Routing

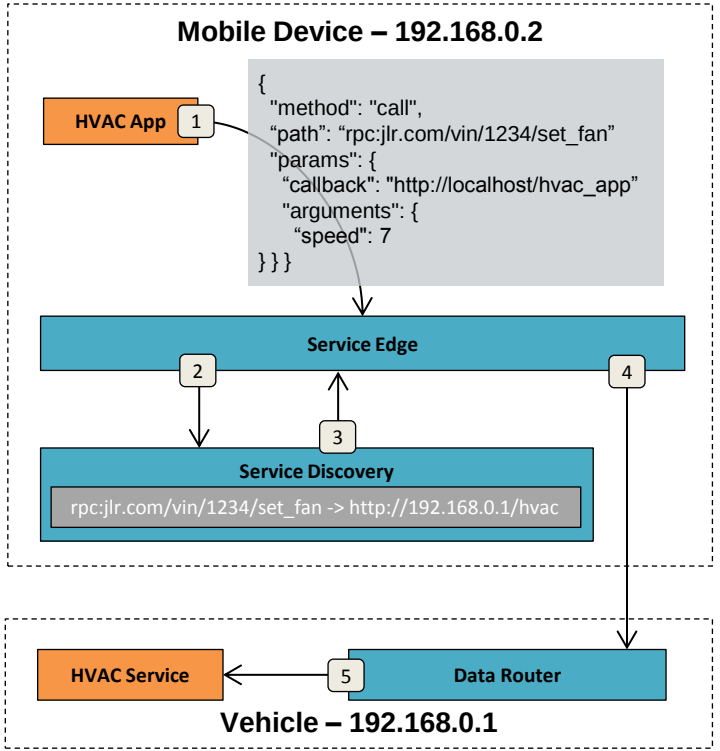
1. **Application sends RPC command**
   HVAC App sends an RPC command, targeting a given service URI, to Service Edge.

2. **Locate target node**
   Service Edge asks local service Service Discovery to resolve service name to a network address.

3. **Return network address**
   Specifies where the target service can be reached.

4. **Send request to Vehicle**
   The vehicle data router processes the command.

5. **Forward request to HVAC Service**
   The HVAC Service in the vehicle executes the command.



Authorization

Authorization – Overview

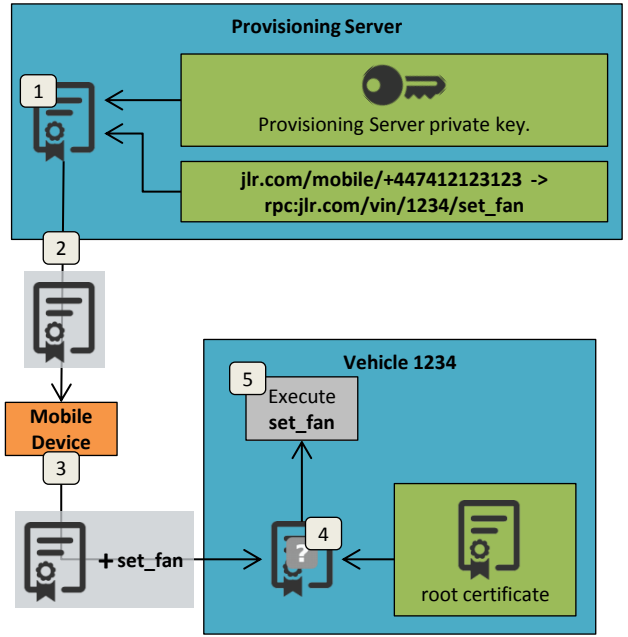
- **Certificate based**
  Certificates, signed by a trusted provisioning server, grants node access to services.

- **Self-carried authorization**
  A node presents its certificates to another node to access its services, without provisioning server connection.

- **Service – service specific certificates**
  A certificate authorizes a specific set of services to access another specific set of services, and cannot be used outside that context.

Authorization – Use Case

1. **Create and sign certificate**
   A certificate granting access to the mobile device is created and signed with provisioning server's private key.
2. **Distribute certificate to mobile device**
   The targeted device receives its certificate
3. **Send request and certificate to Vehicle**
   The certificate states that mobile device has the right to execute the given request
4. **Validate credentials**
   The certificate and request is validated by the vehicle through a root certificate
5. **Execute request**
   The validated command is forwarded to the target service for execution



Authorization – Examples

Access List Format

`[type]:[organization]/[path]` + wildcards

`rpc:jaguarlandrover.com/vin/sajwa71b37sh1839/body/lock`

Specifies a specific vehicle's lock rpc in the body service.

`rpc:jaguarlandrover.com/vin/*/media/volume`

Specifies the volume control command of the media service on all vehicles.

`msg:jaguarlandrover.com/cloud/vehicle_tracking/*`

Specifies all commands under the vehicle_tracking service.

Authorization – Topics not covered

- **Protection of certificate inside a node**
  A credential received by the mobile device needs to be secured in accordance with the mobile device/IVI/server platform

- **Certificate – device binding**
  A stolen certificate can be presented by another device to gain service access. Device binding is done on an implementation level using hardware-specific mechanisms

- **Secure communication**
  Protocol implementations are responsible for securing data transmission between nodes using SSL/TLS or similar technologies

Conclusion

- **Connected Vehicle architecture for next generation services**
- **Open source design, specification, and reference implementation**
- **Benefits the whole industry**
- **Hosted and driven by AGL**

Next Steps

1. **Form AGL expert group**
   The group will drive the specification and implementation work.

2. **Complete design, specification, and planning**
   Review and rework of design is followed by detailed specifications and project roadmap.

3. **Resource commitment**
   Participants commit funds or FTE for reference system implementation.

4. **Build reference implementation**
   AGL will coordinate participating vendors' effort to build the OSS version of the architecture.

Thank You

Magnus Feuer

System Architect – Open Software Initiative

mfeuer@jaguarlandrover.com

High Level Design Draft available at:

https://download.automotivelinux.org/POC/PoC_Spec/15-456-POC-Tizen3-RVI-HLD_Draft4_clean.pdf

-or-

http://bit.ly/1pcOu0Y