Blueprint for Vehicle Data Oriented Strategy - Return of Experience

NEUTRAL SERVER

Kevin Valdek, CTO HIGH MOBILITY
Presented at GENIVI AMM on 16.05.2019
CONTENTS

1. The role of a Neutral Server
2. Market status
3. 3rd party expectations
4. Experience so far
5. Outlook
CONTENTS

1. The role of a Neutral Server
2. Market status
3. 3rd party expectations
4. Experience so far
5. Outlook
ACEA Position Paper

Access to vehicle data for third-party services
Extended Vehicle & Neutral Server recap

**EXTENDED VEHICLE**
- Sharing of vehicle telematics data with 3rd parties
- Both anonymous and personalised vehicle data
- Customer consent and customer choice in focus
- Read-only data

**NEUTRAL SERVER**
- Independent intermediary engaged by OEMs
- Allowed to broker data to 3rd parties within allowed scope
- Protects direct visibility of 3rd party business models from OEMs
Neutral Server - a cross-OEM trusted entity
Customer consent

• Customer consent a prerequisite
• 3rd party has to provide sufficient value to the customer
• Neutral Server can provide compatibility between OEM tech differences
• Good practices in use
Benefits for all sides

**OEM BENEFITS**
- Neutral Servers multiply the business potential
- Minimised effort from OEMs
- Support team and verification responsibilities shifted
- Possible to cater to new data consumer segments

**3RD PARTY BENEFITS**
- Lowers the barrier for new mobility services
- Benefiting from added data partnerships
- *If done right*: one data contract, one data integration
CONTENTS

1. The role of a Neutral Server
2. Market status
3. 3rd party expectations
4. Experience so far
5. Outlook
Current vehicle data availability

**BMW CarData**
since May 2017

**Mercedes-Benz**
since Dec 2018

Availability ~
The journey to a developer ecosystem through APIs

Leadership & Functionality

1. Entertainment
   - Infotainment system

2. Connected
   - Telematics box
   - Telematics service provider
   - Non-standardised API

3. Internal API
   - Developer tools
   - Standardised functions = Car API

4. External API
   - Active community
   - Onboarding workflow
   - Automated billing connected to APIs

5. Developer Ecosystem

Maturity
Reception from 3rd parties

**3RD PARTIES LOVE**

- Finally a scalable model
- No need for to administer hardware or dongles
- Choice in selecting which Neutral Server to integrate with
- Possible to integrate once and continuously support new car brands
- Can avoid individual negotiations with each OEM

**3RD PARTY CHALLENGES**

- Trade-off when it comes to data update rates
- Pricing and request limits vary significantly
- SLAs of high importance to invest in the effort
What's possible today: data bundle examples

- PAYD Insurance: Starting from €0.39 /mo
- Fueling: Starting from €0.29 /mo
- Charging: Starting from €0.29 /mo
- Damage prevention: Starting from €0.29 /mo
- Theft prevention: Starting from €0.29 /mo
- Parking: Starting from €1.56 /mo
- Fleet management: Starting from €2.34 /mo
- Maintenance: Starting from €0.78 /mo
CONTENTS

1. The role of a Neutral Server
2. Market status
3. 3rd party expectations
4. Experience so far
5. Outlook
What attracts 3rd parties: comprehensive developer tools

1. **Standardised API**

2. **Tooling**
   - SDKs
   - Datasets
   - Testing env

3. **Resources**
   - SDKs
   - Documentation
   - Support

4. **Community**
   - Analytics
   - Chat support
   - Forums

5. **Marketplace**
   (app store)

Verification
Publishing
Real car data
CONTENTS

1. The role of a Neutral Server
2. Market status
3. 3rd party expectations
4. Experience so far
5. Outlook
The Auto API is an open protocol that specifies car data and functions in an abstract format, removing the specifics of each car model or brand.

It’s flexible and can be easily expanded to accommodate new car architectures or functions.

SDKs for iOS, Android, Node.js and a REST API to developers to work with the Auto API.

Removes the fragmentation and minimises effort on the developer side.

A STANDARDISED CROSS-OEM VEHICLE API
3rd party developers

Back-end apps

IOS apps

Android apps

Back-end

REST API

HMKit Crypto

OEM1 Adapter

SDK Endpoint

WebSocket API

OEM2 Adapter

GraphQL API

OEM... Adapter

Front-end emulator

GraphQL API

WebSocket API

Auto API parser

OEM Back-end

REST API

Consent Flow

WebSocket API

GraphQL API

Emulator configuration and state

Auto API messages

OEM data protocol
Approach taken to harmonise vehicle data for 3rd parties: Auto API

**REQUIREMENTS**

- Has to work in an IOT environment, embedded solutions
- Supports both polling and pushing of data
- Can efficiently be encrypted/decrypted
- Is specific for automotive use cases

**SOLUTION**

- Went ahead with two different components: HMKit, Auto API
- Binary format at its core — with abstraction layers
- SDKs to handle the security layer and platform specifics
- First prototypes with Bluetooth Low Energy, Telematics added later
REST API
Back-end
3rd party developers
Back-end apps
Android apps
iOS apps
Auto API data protocol
WebSocket API
GraphQL API
Auto API parser
Auto API messages
Emulator configuration and state
OEM data protocol
OEM Back-end
REST API
Consent Flow
OEM1 Adapter
OEM2 Adapter
OEM... Adapter
Auto API data protocol
HMKit & Auto API

- **HMKit** — Transport protocol with security layer and SDKs
- **Auto API** — Vehicle data specification

Diagram:
- Telematics container
- Client serial
- Vehicle serial
- Auto API
- Signature

- Bluetooth container
HMKit & Auto API

- Telematics container
- Client serial
- Vehicle serial
- Auto API
- Signature

Example: Diagnostics, Diagnostics State
- Capability ID
- Message type
- Size
- Properties

Examples: Mileage, Engine oil temperature
- Property ID
- Size
- Components

Components: value, timestamp, failure
- Component ID
- Size
- Data
**Diagnostics State**

This message is sent when a Get Diagnostics State message is received by the car. The new state is included in the message payload and may be the result of user, device or car triggered action.

**MESSAGE SPECIFICATION**

<table>
<thead>
<tr>
<th>Message Direction</th>
<th>Sent from Car to Smart Device</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data[0..1]</strong></td>
<td></td>
</tr>
<tr>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x33</td>
<td></td>
</tr>
<tr>
<td><strong>Data[2]</strong></td>
<td></td>
</tr>
<tr>
<td>0x01</td>
<td>Diagnostics State</td>
</tr>
<tr>
<td><strong>Property Data[0]</strong></td>
<td></td>
</tr>
<tr>
<td>0x01</td>
<td>Identifier for Mileage</td>
</tr>
</tbody>
</table>

**MESSAGE EXAMPLE**

```
[
  0x00, 0x33,  # RSP, LSB Message Identifier for Diagnostics
  0x01,  # Message Type for Diagnostics State

  0x01,  # Property Identifier for Mileage
        (uint 16) 7,  # Property size is 7 bytes
  0x01,  # Data component identifier
        (uint 16) 4,  # Data component size is 4 bytes
  0x00, 0x02, 0x49, 0xF0  # Diameter is 100 cm in km

  0x02,  # Property Identifier for Engine oil temperature
        (uint 16) 5,  # Property size is 5 bytes
  0x01,  # Data component identifier
        (uint 16) 2,  # Data component size is 2 bytes
  0x00, 0x63  # Engine oil temperature is 99 degrees Celsius
```
Auto API journey from Level 1 to Level 10

Building the SDKs through UX concepts

Auto API Level 1
- Door Locks
- HMKit iOS

Auto API Level 2-4
- Diagnostics
- Trunk Access
- Climate
- Charging
- HMKit Android

Auto API Level 5
- Capabilities
- Vehicle Status
- Vehicle Location
- Rooftop
- Windows
- Maintenance
- Engine
- Theft Alarm
- + more
- HMKit Node.js

Auto API Level 6-7
- Race
- Offroad
- Dashboard Lights
- Chassis Settings
- Seats
- Light Conditions
- + more
- REST API

Auto API Level 8-10
- Historical
- MultiCommand
- Usage
- + more

Neutral Server
Live

Platform open to developers

Car data & App Directory added
Parse the received command's bytes

```java
byte[] bytes = ...;
Command command = CommandResolver.resolve(bytes);

VehicleStatus vehicleStatus;
Capabilities capabilities;

if (command instanceof VehicleStatus) {
    vehicleStatus = (VehicleStatus) command;
} else if (command instanceof Capabilities) {
    capabilities = (Capabilities) command;
}
```

Get a specific state from the vehicle status

```java
LockState state = vehicleStatus.getState(LockState.TYPE);
if (state != null) {
    ...
}
```

Inspect whether the capability is supported for the vehicle

```java
if (capabilities.isSupported(LockState.TYPE)) {
    ...
}
```

... and more
CONTENTS

1. The role of a Neutral Server
2. Market status
3. 3rd party expectations
4. Experience so far
5. Outlook
HMKit & vehicle data standardisations

- Positive trend for vehicle data format specifications
- Most helpful if the impact includes external APIs
- Allows for better adoption among 3rd parties

Diagram:

- Telematics container
- Client serial
- Vehicle serial
- Auto API
- Signature

- W3C
- CCC
- Sensoris
Neutral Server
Thank you