Android Automotive SIG – Vehicle Data APIs / VHAL
Project status report
Stefan Wysocki, TietoEVRY

GENIVI Tech Summit | October, 2020
Agenda

• External Data Server Framework
  • PoC implementation

• Android-side (Head Unit) Components Status
  • Framework Layer: Authentication service
  • HAL Layer : VHAL
  • Application Layer: App

• Non-Android In-Vehicle Components Status
  • VSS feeder
  • VSS database
  • GraphQL
External Data Server Framework
Proof-Of-Concept

Use-case:

• Application wants to read the current fuel level and the tank capacity

Requirements:

• Read access for signals needs to be protected by a permission: org.genivi.vss.permission.FUEL_SYSTEM_READ

• Permission needs to be granted by an authority which is secure

• Data needs to be defined and structured according to VSS

• Data needs to be accessible by the framework as well
Android-side (Head Unit) Components Status
Framework Layer: Authentication service

• Authority that is able to verify the granted permissions to calling application and generate the token with authentication data

To be agreed:

• Token signature and the encryption
HAL Layer : VHAL

- Support for Google Automotive Services (GAS)
- Support for framework services (like CarUxRestrictions)
- Based on TietoEVRY VHAL
- Extension has been implemented to provide the properties to the Android Framework queried from GraphQL server
- Currently, the support is limited to „get“ calls and „subscribe“

Focus

Proof-of-Concept

- Technology: Android HAL
- Language: Java/C++
- Source: GitHub
- Status: WIP
Application Layer

2 applications:

• Application is directly using GraphQL to connect to the server and uses Authenticator Service to generate the token with permissions bundled.

• KitchenSink that uses standard Android API
Non-Android In-Vehicle Components Status
In-vehicle VSS feeder

• vss-feeder writes the signals from OpenDS into SQLite file, and the other components read from the file. I.e. the database "API" is SQL

• Note: SQLite database schema was not (yet) aligned with CCS project schema

Proof-of-Concept

<table>
<thead>
<tr>
<th>Technology</th>
<th>NodeJS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language</td>
<td>JavaScript</td>
</tr>
<tr>
<td>Source</td>
<td>GitHub</td>
</tr>
<tr>
<td>Status</td>
<td>WIP</td>
</tr>
</tbody>
</table>
In-vehicle VSS database

- Access realized by Apollo GraphQL
- Resolve requested data for the APP from the VSS data structure in SQL
- Internal resolvers fetch the data from SQLite.

To be done:

- Implement the permissions enforcement in the data server

Focus

Proof-of-Concept

<table>
<thead>
<tr>
<th>Technology</th>
<th>NodeJS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language</td>
<td>TypeScript</td>
</tr>
<tr>
<td>Source</td>
<td>GitHub</td>
</tr>
<tr>
<td>Status</td>
<td>WIP</td>
</tr>
</tbody>
</table>
Conclusions and future plans
Conclusions

• No GraphQL client library found for native services
• „Real time” subscription needed
• Development moved to Android 10, what about 11?
Future plans

• Implement the permissions enforcement in the data server
• Implement the output of the discussion about the token security
• Updating the property values - usecase needed!
• Restructure the source repositories
Repository structure

• For Android components: AOSP + local_manifests

• Additional repository for „Genivi” flavoured devices containing the product configuration

• Unify branching strategies for maintaining multiple Android version

• For non-Android components: centralized repository with sub-repositories (VSS feeder + Apollo GraphQL)
Contributing

• Weekly telcos:
  Tuesdays – 17:00 CET (US friendly time) – Vehicle Data APIs / VHAL

• Android Automotive Project Wiki:
  https://at.projects.genivi.org/wiki/x/XgA4Ag
Thank you!

Visit GENIVI:
http://www.genivi.org
http://projects.genivi.org

Contact us:
help@genivi.org