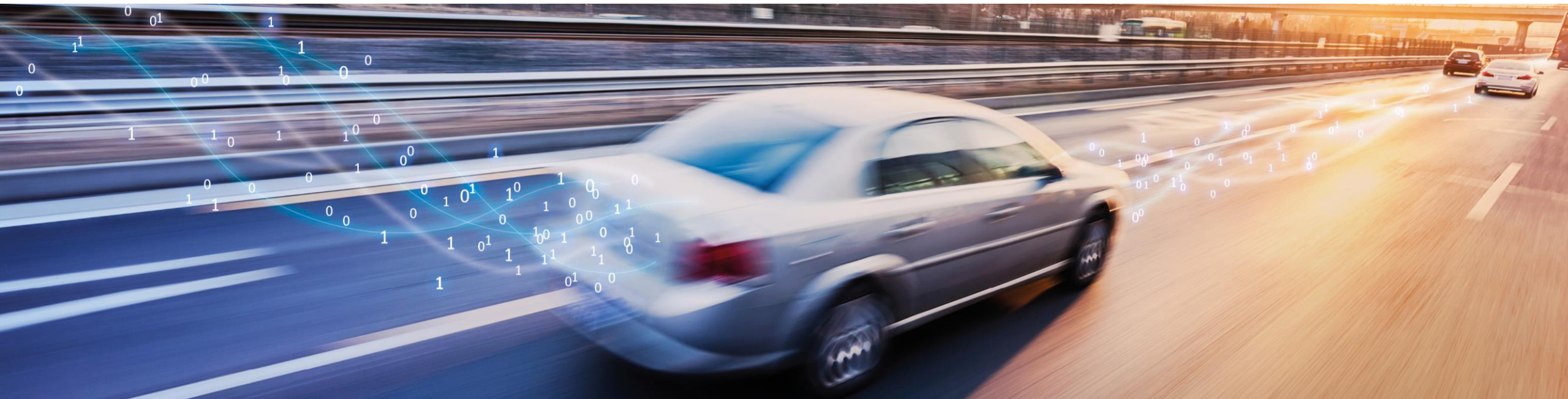


SOVD – Service Oriented Vehicle Diagnostics

Bernd Wenzel
ASAM e.V.

Tobias Weidmann
Vector Informatik GmbH

Stuttgart/Dresden
27.04.2022



Agenda

1	Motivation
2	Concepts
3	Methods Overview
4	CDA
5	Next Steps

Motivation

- Support of next generation software architectures
- ADAS using HPC's (software-based systems)
- Continuously Update of software in the vehicle, also providing new functionalities
- Vehicle as IOT device
- Analysis of software while running (not simple reading of Error codes)
- Not limited to data Use cases, also considered process related use cases
- Support of interactive generic diagnostics

Motivation

Why is there a need for something beyond UDS?

- UDS is still the choice for classic ECUs but will not cover all requirements of future systems
- Not designed to be flexible, Requires static description of content
 - Hard to keep this up-to-date if the vehicle is constantly updated
- Data required for diagnosing SW-based systems does really fit to todays UDS (byte-) based world
 - Read and filter accumulated and structured logs & traces
 - Read faults and crashes with environment data like e.g. stack traces
 - Install and remove apps, Update software
 - Access terminals
 - Continuously stream logs, traces, metrics and data like e.g. camera input
- SOVD will not replace UDS, we expect a co-existence

SOVD API (Next Generation Diagnostics)

Remote

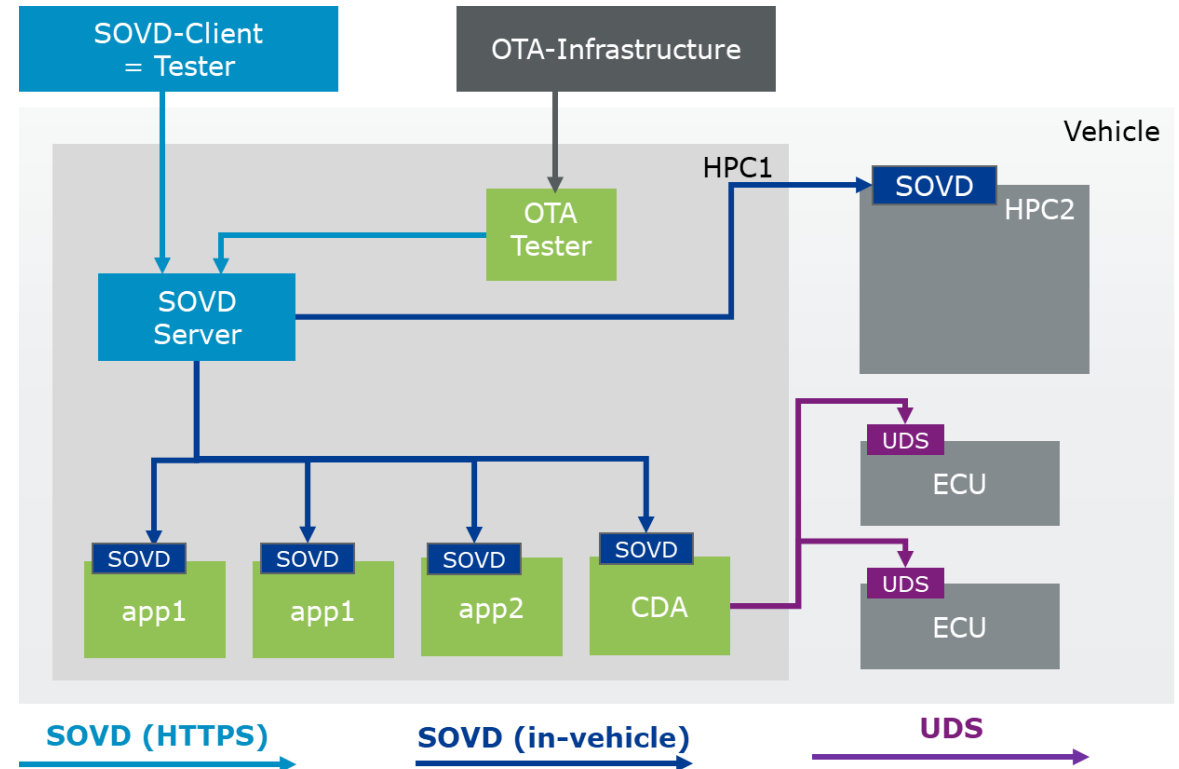
- SOTA
- Backend evaluation
- Fleet management
- Remote assistance (also on roadside)
- Activation on functionality as paid by customer

Proximity

- Workshop / Service
- Manufacturing (e.g. EOL)
- Emission check and ePTI

In-Vehicle

- Monitoring (sporadic errors)
- Predictive maintenance
- Health status access



Concepts

HTTP/REST in a nutshell

- REST is based on HTTP, basically a web browser is sufficient to execute
- Resources are the core element
- Dedicated HTTP Verbs are called on the resources offered by the server
- Knowledge of the initial URL is sufficient, further links are provided to discover the API
- REST is stateless, i.e. Every request contains all the relevant information that the server can process it



The screenshot shows a web browser window with the address bar displaying the URL: `localhost:34568/MyServer/Vehicle/ecus/body_ctrl_front/features/iddata/activediagnosticinformation`. The page content displays a JSON response from the API. The JSON structure is as follows:

```
1 // 20200626074717
2 // http://localhost:34568/MyServer/Vehicle/ecus/body_ctrl_front/features/iddata/activediagnosticinformation
3
4 {
5   "activediagnosticinformation": {
6     "Active_Diagnostic_Session": {
7       "encoding": "UTF8_FIELD",
8       "name": "Active Diagnostic Session",
9       "value": "Extended"
10    },
11   "Active_Diagnostic_Variant": {
12     "encoding": "UNS",
13     "name": "Active Diagnostic Variant",
14     "value": "0"
15   },
16   "Active_Diagnostic_Version": {
17     "encoding": "UNS",
18     "name": "Active Diagnostic Version",
19     "value": "0"
20   },
21 }
```

No automotive specific stack needed on Client side

Structure of the Service Specification

Method:

GET/<entity path>/data/<ResourceName>

Method description

Service semantic (described as OpenAPI spec)

Path Parameters

Identification of requested resource

Query Parameter ^{*1}

Selection of optional response members, typically used for include-schema

Request Header

Selection of requested Content-Type and handover of authorization credentials

Request Body ^{*1}

Data transmitted from SOVD Client to SOVD Server

Parameter Name / Attribute	Type	Convention	Description
		M / O / C	

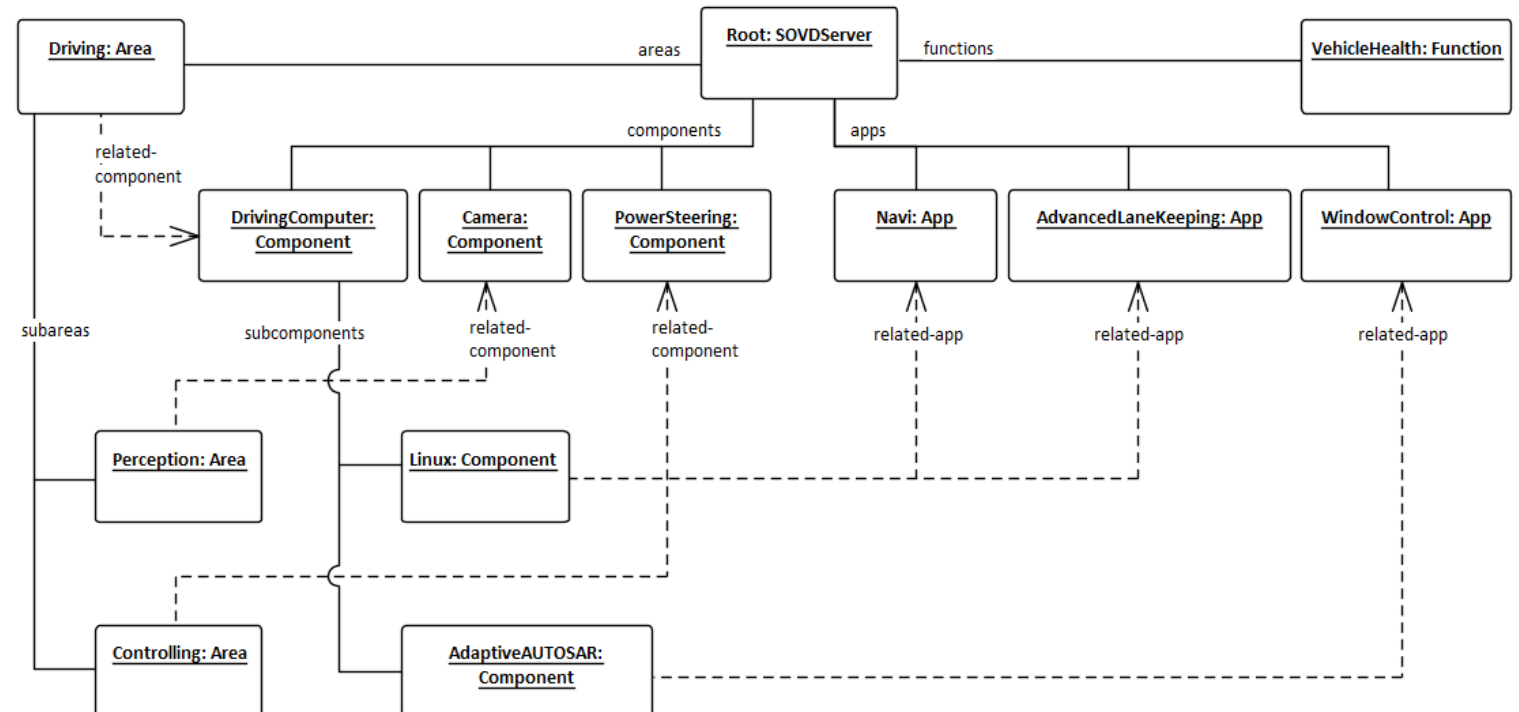
Method for Capability discovery

Access to Capability Description Content

- Query an Online Capability Description

Discovering of Entities and Resources

- Discover Contained Entities
- Query Sub-Entities of an Entity
- Query related Entities of an Entity
- Query Entity Capabilities



Identical format for Offline and Online Capability description used, based on OpenAPI format

Method for Fault Handling

Provided Methods

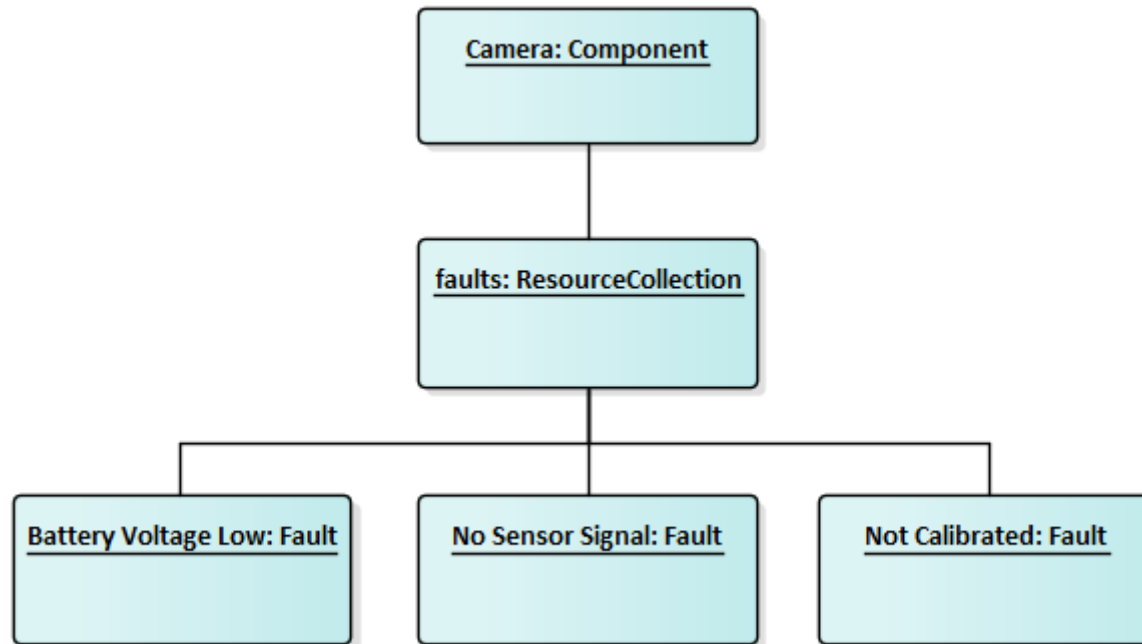
- Read Faults from an Entity
- Read Details for a Fault
- Delete all Faults of an Entity
- Delete Single Fault of an Entity

Query Parameters

- Status, based on Key Value Pair
- Severity

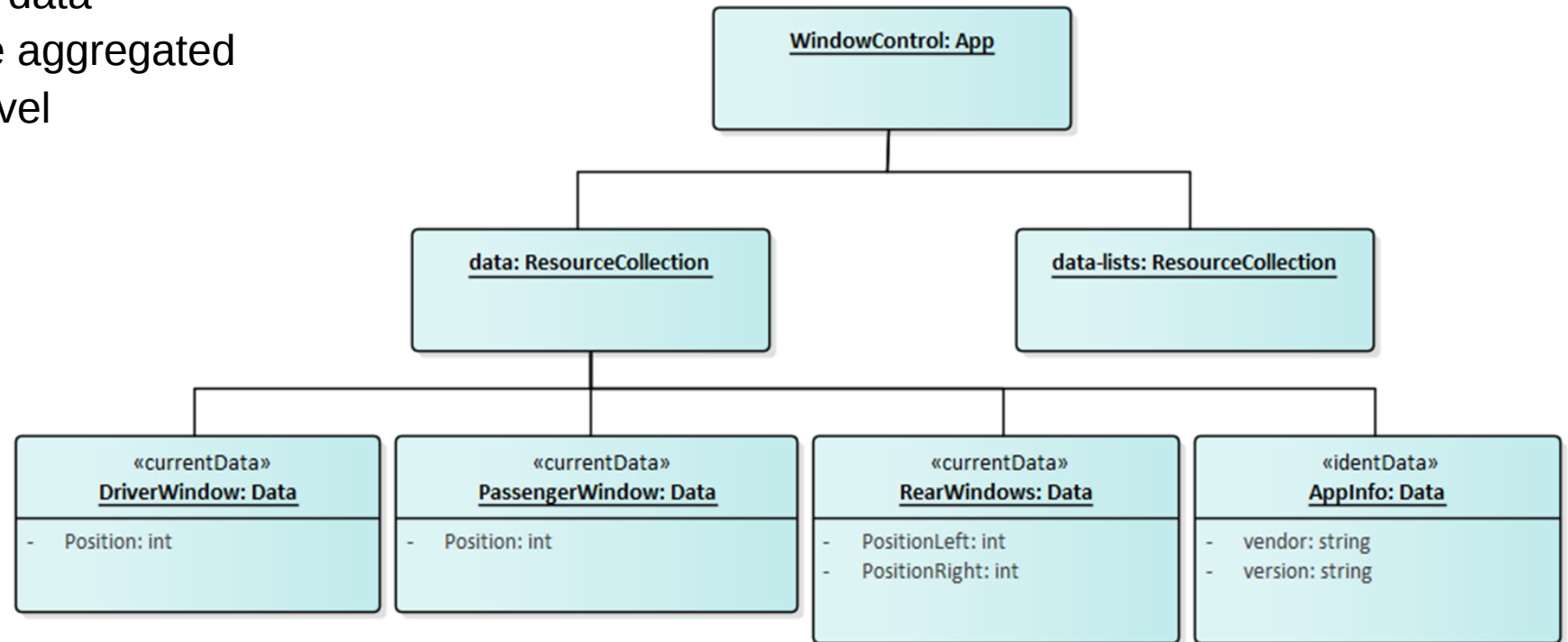
Access to environment data for a single fault code

- OEM specific key value pairs



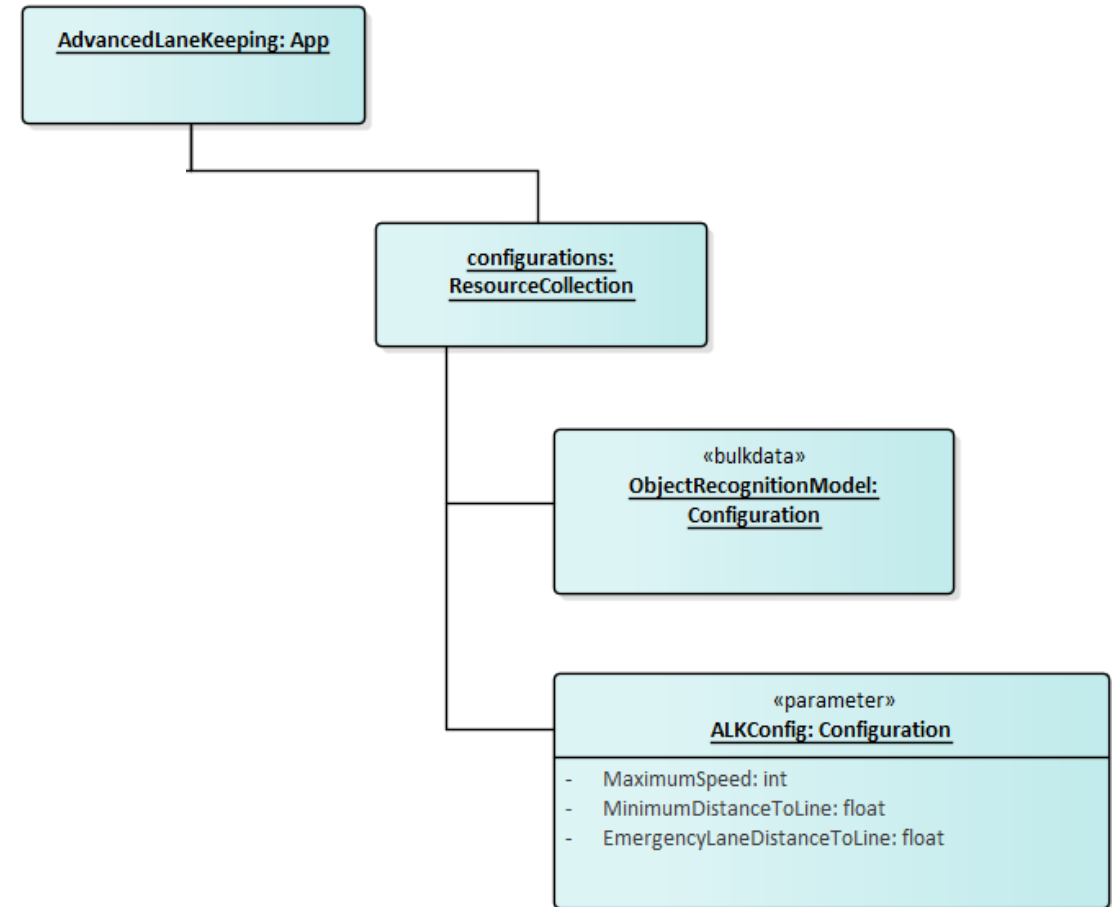
Method for Data Resource read / write access

- Retrieve the list of data available for an entity
- Data is categorized according to its semantic
 - E.g. currentData, identData, storedData, sysInfo
- Read/Write Access to Data
- Possibilities to group data
- Possibilities to create aggregated data sets on entity level



Method for Configuration

- Retrieve List of all Configurations Provided by the Entity
- Read and Write Configuration as Parameters
- Read and Write Configuration as bulk data



Method for Control of Operations

Operations (SW-internal functions, Actuators)

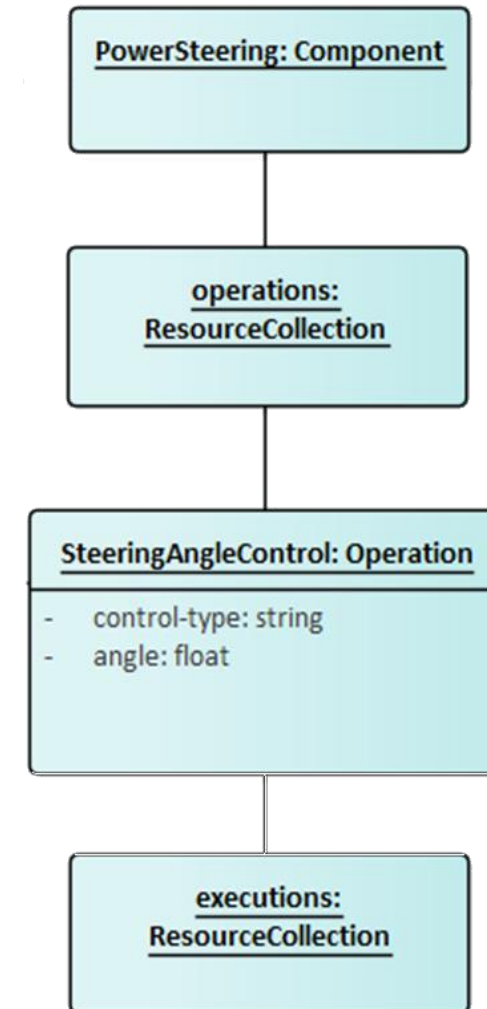
- Retrieve List of all Available Operations from an Entity
- Get Details of a Single Operation
- Start Execution of an Operation
- Get Executions of an Operation
- Get the Status of an Operation Execution
- Stop the Execution of an Operation
- Support for execute / freeze / reset and OEM-specific Capabilities

Target Modes

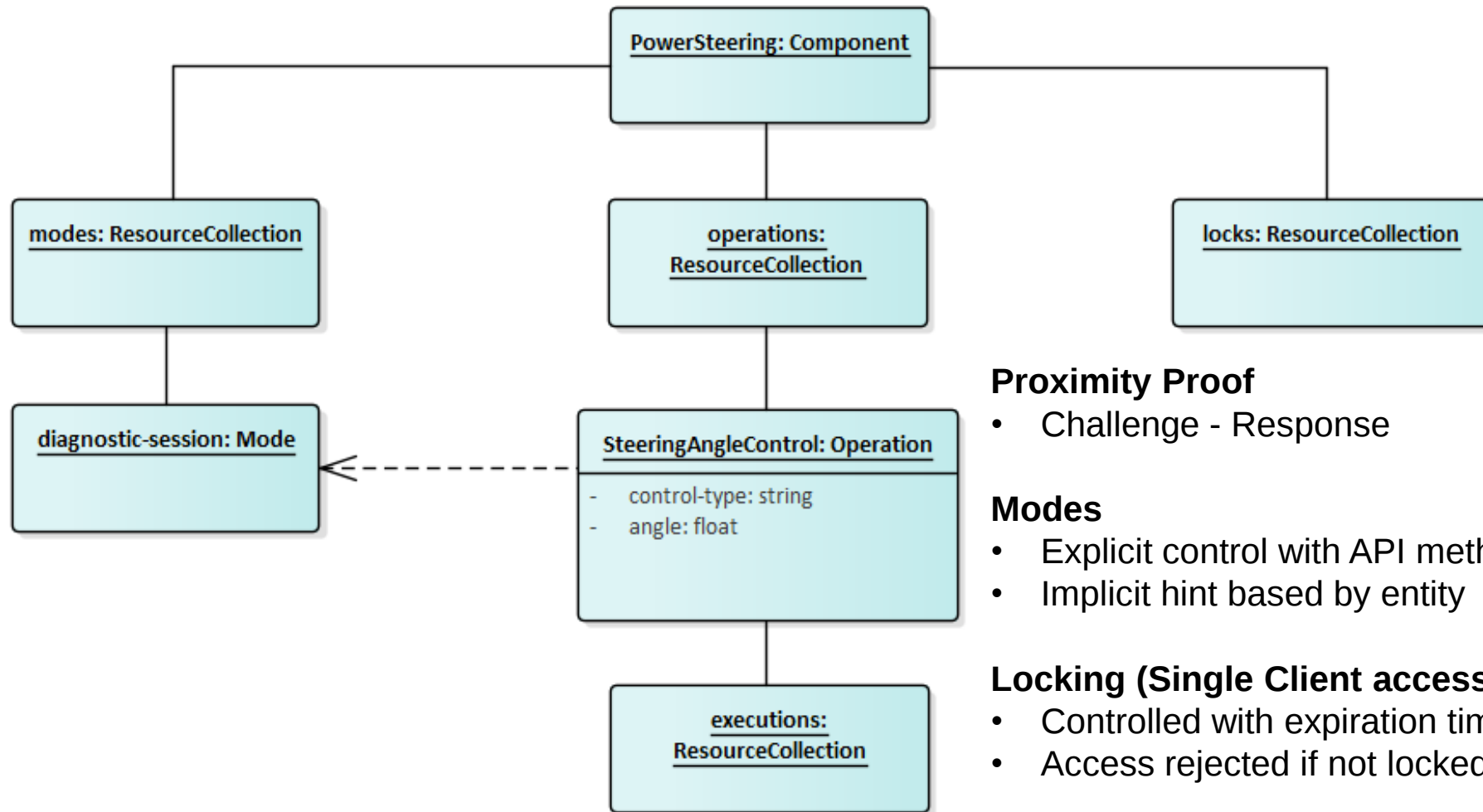
- Retrieve List of all Supported Modes of an Entity
- Get Details of a Single Mode of an Entity
- Explicit Control of Entity States via their Defined Modes

Locking

- Acquire a lock on an entity
- Get all acquired locks of an entity
- Get a single active lock of an entity
- Extend an acquired lock on an entity
- Release an acquired lock on an entity



Method for Control of Operations



Proximity Proof

- Challenge - Response

Modes

- Explicit control with API methods
- Implicit hint based by entity

Locking (Single Client access)

- Controlled with expiration time
- Access rejected if not locked

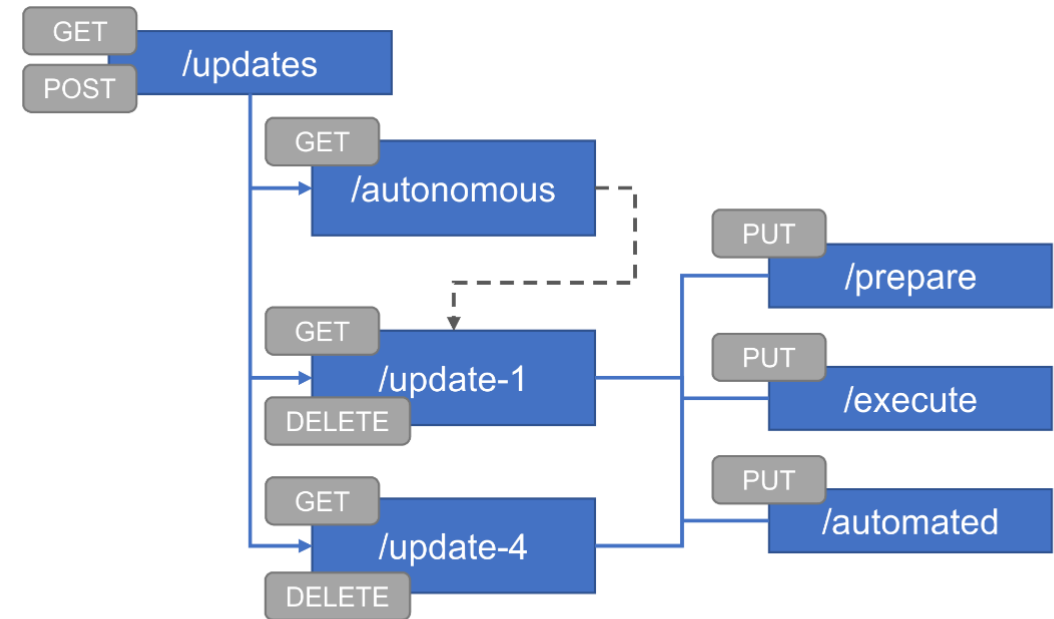
Method for Software Update

Basics

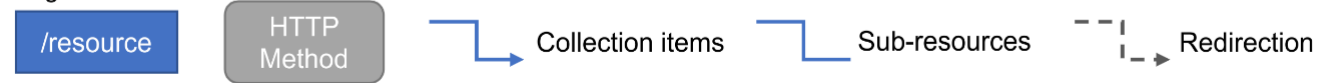
- It is assumed that there is a central component in the vehicle which performs the software update
- ASAM SOVD provides an API to trigger this central software update component
- Update procedure itself is not subject to ASAM SOVD

Methods

- Retrieve List of all Updates Provided by the Entity
- Get Details of Update
- Automated Installation of an Update
- Prepare Installation of an Update
- Execute Installation of an Update
- Get Status of an Update
- Delete Update Package from an SOVD server
- Register an Update at the SOVD server

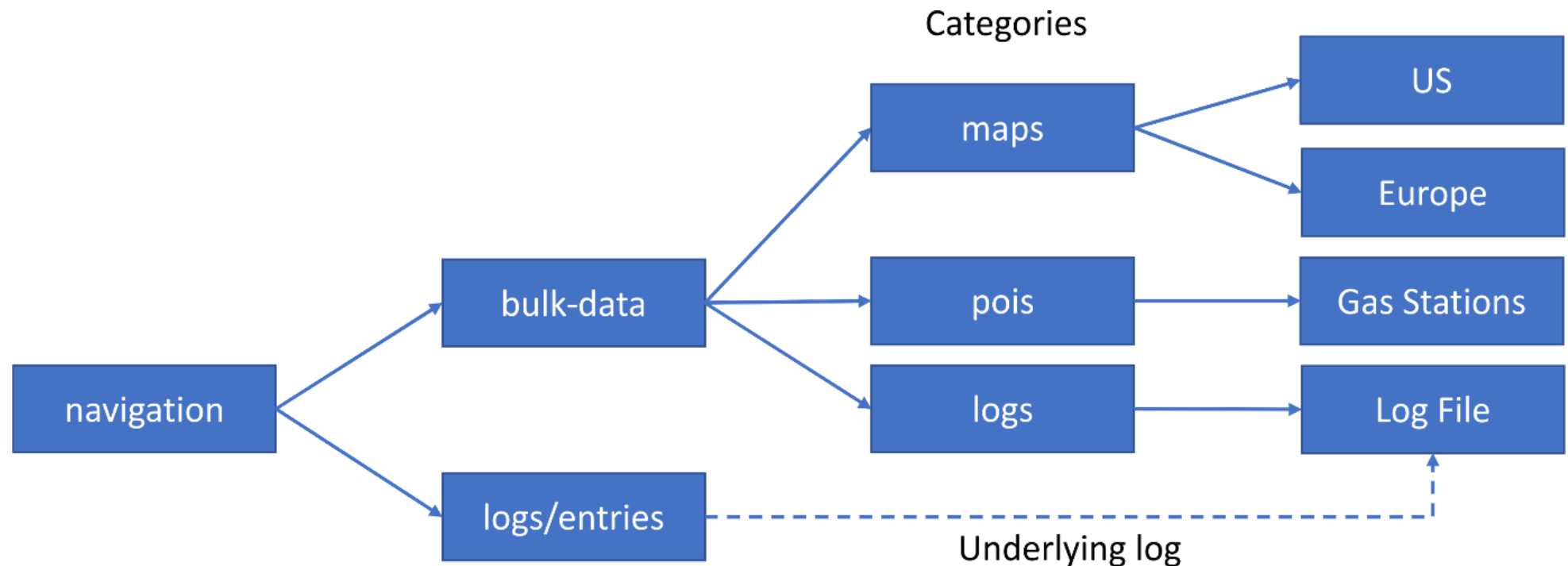


Legend:



Method for Handling of bulk-data

- Retrieve List of all bulk data Categories
- Read bulk data Meta Data
- Download bulk data
- Upload bulk data
- Delete all bulk data Defined by Category
- Delete specific bulk data Resource



Method for Logging

- Retrieve List of all log Information
- Configure SOVD Logging
- Retrieve the current SOVD Logging Configuration
- Reset SOVD Logging Configuration to Default

Principle

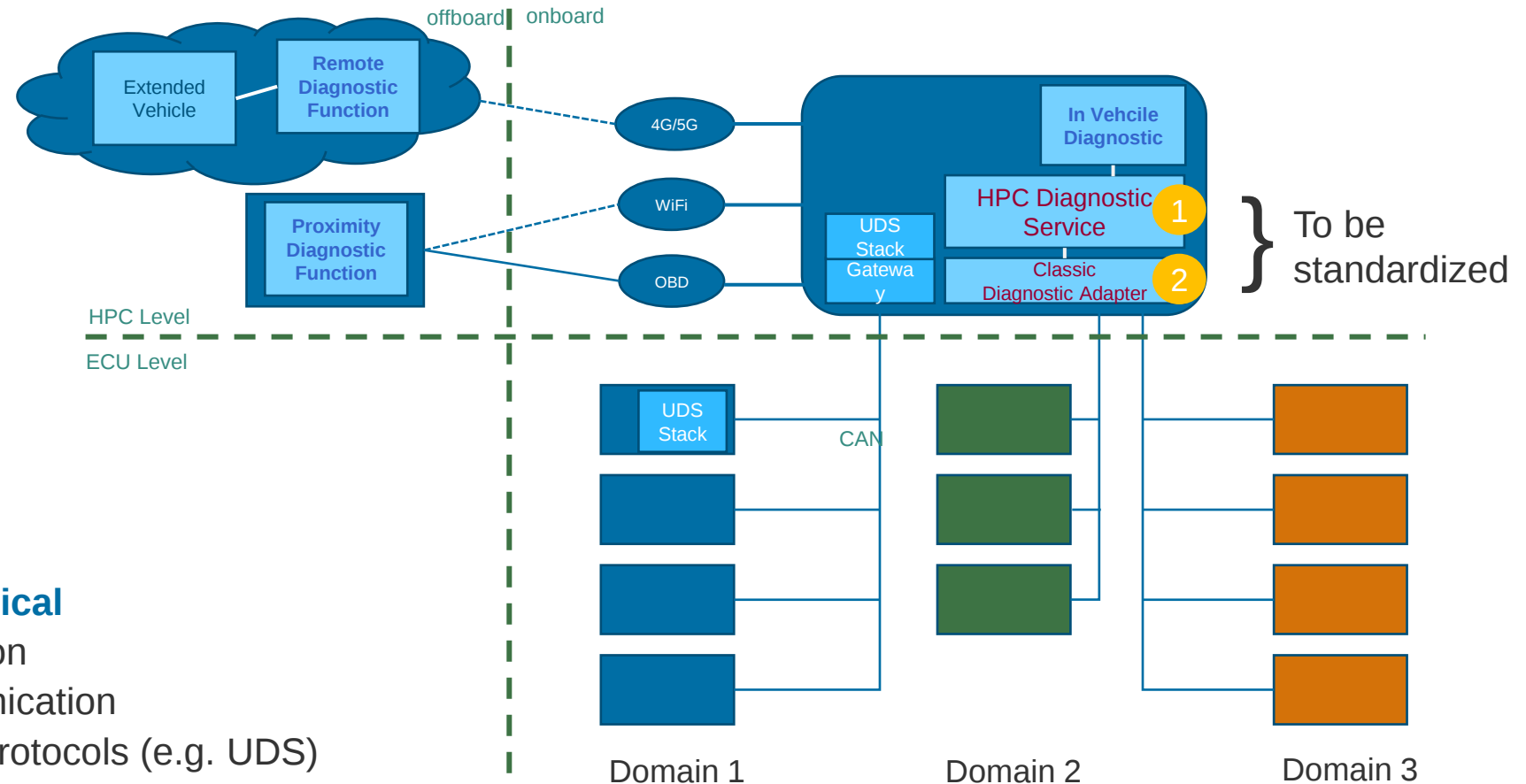
- Access to aggregated log information
- Evaluation by software experts
- Transport as bulk-data possible

Supported Context Types

- RFC 5424 (Syslog Protocol)
- AUTOSAR Diagnostic Log and Trace

Classic Diagnostic Adapter

Easy migration between classical and web-based access



Encapsulation of classical

- Stateful communication
- Signal based communication
- Usage of diagnostic protocols (e.g. UDS)

e.g. an Internal MVCI System

Mapping defined for UDS services

SOVD inside the standardization landscape

ASAM SOVD 1.0

- Public Review for SOVD 1.0 is in progress
- Release on TSC meeting in July

AUTOSAR Alignment

- Involved in Internal review
- Handled in concept group 704
- ara::diag extension

ISO Standardization

- ISO SC 31 / WG 2
- NWIP will be presented in June 2022

ASAM Follow Up project

- Planned for 09/2022 – 12/2024
- Compatible minor version
- Integration of ISO feedback
- Event based communication

Thank you for your attention!

Bernd Wenzel
Senior Technical Consultant
ASAM e.V.

Phone: +49 371 5607 742
Email: bernd.wenzel@asam.net

Tobias Weidmann
Manager Customer Services
Vector Informatik GmbH

Phone: +49 711 80670 2549
Email: tobias.weidmann@vector.com