



Eclipse OpenSOVDCore – Status Quo

Ramachandran Chakravadhanula

The Task

SDV architecture brings new challenges to diagnostic systems



Interoperability and Standardization Challenges

- Dynamic software configuration
- Consistent access to vehicle data

Operational and Scalability Limitations

- Demand for fast OTA:
software updates | remote diagnostics
- Connected fleets require a scalable architecture

Security and Cost Impact

- Seamless user authentication and access management.
- OEM or supplier specific implementations – Need for Opensource development.

OpenSOVD | Open. Standard. Next Gen Diagnostics

Project of Eclipse SDV Working Group

SOVD - A unified, future-proof diagnostic system for modern SDVs and existing fleets

Unified diagnostics

Standardized Diagnostics for all electronics components which supports classic and SDV E/E architectures

Modern design

Modular, Secure and Scalable: Seamless OTA, Remote Diagnostics and Machine Learning Use Cases

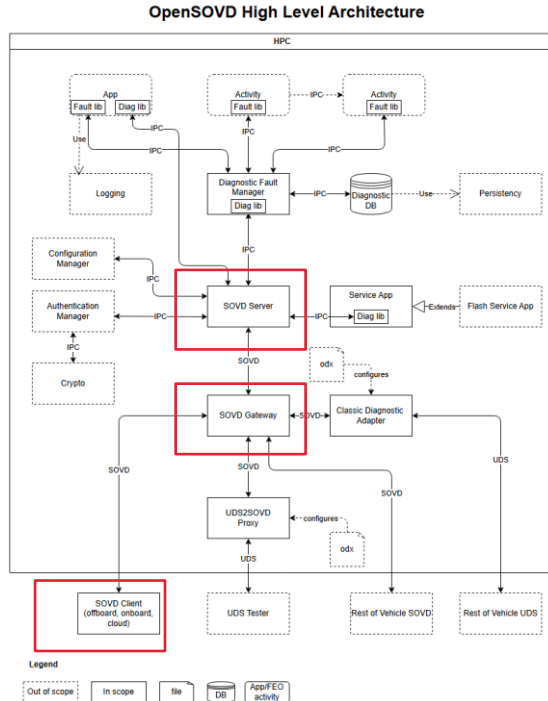
Standardized

ASAM/ISO Groups | Eclipse OpenSOVD | S-Core Projects



Eclipse OpenSOVD Components

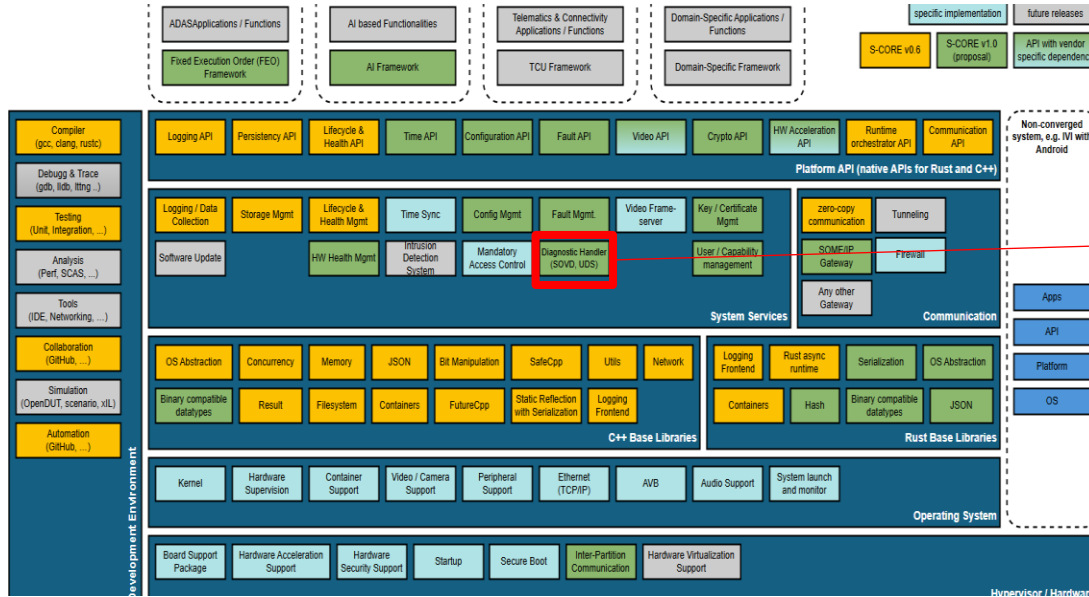
OpenSource Implementation for SDV Diagnostics



SOVDCore Components

- **SOVD Server** – Diagnoses High Performance Computer & Software Applications.
- **SOVD Gateway** - Provides proximity access to the vehicle (exposes private SOVD Servers).
- **Diag Lib** – Interface between HPC Apps and SOVD Server
- **SOVD Client** – Offboard client to access the SOVD Server

SCORE Ecosystem Fit



OpenSOVD is seen as diagnostics stack for the SCORE Middleware.

diag_lib as the interface for the SCORE based APPs to OpenSOVD.

Sources: eclipse-score/inc_diagnostics:
[Incubation repository for diagnostics](https://eclipse-score/inc_diagnostics)
[feature](https://eclipse-score/inc_diagnostics)

Eclipse OpenSOVDCore Workstream

Open-source SOVD Server, Client & Gateway

- Eclipse Foundation project (eclipse-opensovd/opensovd-core)
- ISO 17978-3:2026 implementation, SOVD Specification version 1.1
- Rust
- Apache 2.0
- Contributors: Liebherr, ZF & community

Design Tenets

- Spec Compliant URI structure and response envelopes.
- Pluggable data backends behind a single trait (no framework lock-in)
- Topology is dynamic; entity add/remove broadcast at runtime
- Security first-class (JWT + Rego + TLS/mTLS).
- Single-binary deployment, release-small profile (LTO, opt-level=z, strip)

SOVD Terminology

A handful of nouns — everything else in the standard is built from them.

Entities

the things you talk to

Component

hardware - an HPC or an ECU

App

software located on a component

Area

logical zone / domain view

Function

cross-entity view, e.g. VehicleHealth

Resources

what an entity offers

data/

values you read and write

faults/

DTCs and fault status

operations/

routines, I/O control, sw functions

...

configurations · logs · modes · locks · updates

Data categories

how data is classified

identData

fixed parameters - part number, VIN

currentData

measurements - battery voltage

storedData

parameters, read + write

sysInfo

dynamic system resources - CPU load, RAM etc.,

Relations tie entities together: hosts, is-located-on, belongs-to, contains, depends-on — and every entity describes its own capabilities

GET /sovd/v1/{entity-path}/{resource-collection}/{resource-id}

GET /sovd/v1/components/Hpc1/data/battery-voltage

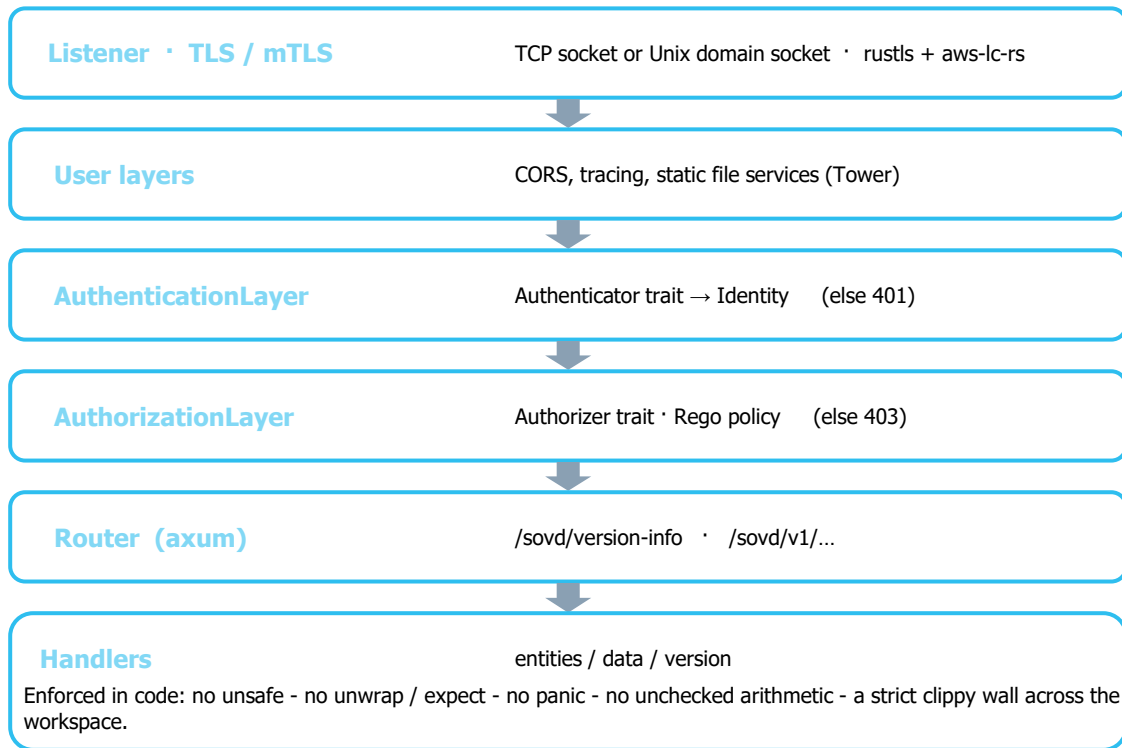
GET /sovd/v1/apps/App1/faults

Implemented OpenSOVDCore Resources

Resource	List	Read	Write
/version-info	—	✓	—
/components	✓	—	—
/components/{}/data	✓	✓	✓
/apps	✓	—	—
/apps/{}/data	✓	✓	—
/areas	✓	—	—

Inside SOVD Server

opensovd-server every request travels the same path, top to bottom.



AppState → Topology

Thread-safe registry (RwLock) + relationship indexes + change-event broadcast

- **Component** — HPC / ECU hardware
- **App** — software on a component
- **Area** — logical zone / domain

DataProvider trait

list read write + categories, groups, tags
one pluggable backend per entity

Workspace Layout



Binaries

opensovd-cli/gateway - HTTP entry point
opensovd-cli/mcp - LLM bridge over stdio



Server

opensovd-server - Axum router, AuthN/AuthZ middleware, TLS



Core libraries

opensovd-core - Topology, Component / App / Area, EntityRef, DataProvider trait
opensovd-models - Wire types, response envelopes, JsonSchema
opensovd-extra - JWT, Rego, unit conversion



Support

opensovd-providers - Builder helpers for in-memory data providers
opensovd-client - Typed HTTP + Unix transport
opensovd-mocks - Demo topology for tests and demos



workspace crates has one responsibility each. Clear separation: opensovd-core knows nothing about HTTP;
opensovd-server knows nothing about your data source.

Secure by Design

Every request flows through a middleware stack before it reaches a handler.

Request —▶ User layers (CORS, tracing)

- ▶ Authentication → identity (else 401)
- ▶ Authorization → policy (else 403)
- ▶ Route handler → (entity / data / version)

Enforced in code: a strict clippy wall denies unsafe, unwrap, expect, panic, indexing & arithmetic side-effects workspace-wide.

MCP Server — LLM Bridge

A second binary (opensovd-mcp) speaks Model Context Protocol over stdio against a running gateway. SOVD topology becomes tool calls and resources for Claude / Copilot / any MCP-aware client.

Currently exposed

- **Tools:** list_components, list_areas, list_apps
- **Resource:** sov://topology
- **Prompt:** explore-topology

Use case

A workshop tool asks the LLM "*which apps run on this ECU?*" and gets a structured answer drawn from live SOVD data - without writing client code per prompt.

Try it out/Demo

Run the gateway with mock data

```
ghcr.io/eclipse-opensovd/opensovd-gateway --mock
```

```
curl -s http://127.0.0.1:7690/sovd/version-info | jq
```

Roadmap

In flight

- SOVD Capabilities (Online)
- Service Discovery
- Platform hardening with Bazel

Q4 2026

- Support SCORE integration activities.
- SOVD App Diagnostics (app-level fault / data operations)
- Proximity Software Updates on HPC

Get Involved

Takeaways

- Diagnostics as an open, web-native, self-describing API
- Secure by design; AI-ready via MCP
- Rust implementation of ISO 17978-3

Join us

- github.com/eclipse-opensvd/opensvd-core
- Weekly Forum - Mondays 11:30 CET
- Slack #eclipse-opensvd