Vehicle Service Catalog (VSC) - Common Interface Description Model

Weekly Meeting

VSC weekly development meeting: Meeting Link

Recorded Meetings

You will find recorded meetings here.

Overview

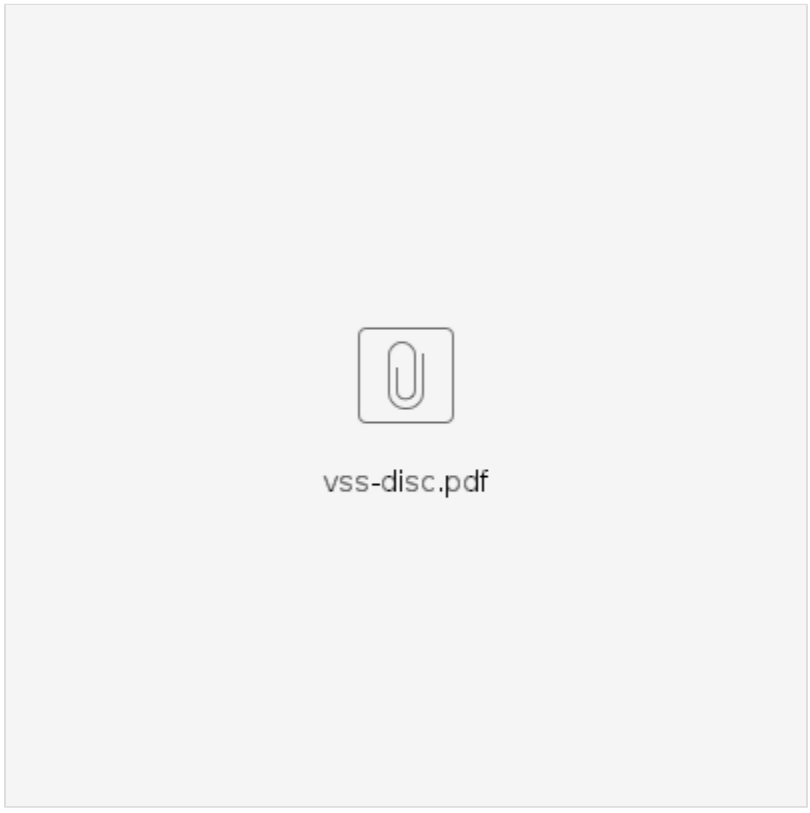
Mission

Create a standardized, extensible vehicle service catalog, and associated tools, to enable protocol-, language-and specification-agnostic interoperability between ECUs, infotainment, and cloud.

- **Standardized** – Version managed service specification with regular releases
- **Extensible** – Proprietary extensions can be added to an open-standard catalog
- **Tools** – Auto-generate network code and APIs from service specifications
- **Specification agnostic** – Translation to and from multiple specification formats
- **Interoperability** – Enable seamless communication in the vehicle and over the air

Presentations from April 2022 All Member Meeting

VSC And VSS Alignment



October 2021 AMM Presentation - Excellent Background on Current Thinking

Jump to 8:04 for VSC. Confluence won't let you embed a YouTube link with a start time.

Ongoing design/discussion points

For details it could be best to track the GitHub issues in VSC and VSC-tools

- Stakeholder input: #1 preferred source-language for interface descriptions. e.g. Franca IDL or other, or do companies use gRPC directly, etc.?
- Stakeholder input: Interesting mapped technologies, input-and-output formats and bindings. (e.g. gRPC, SOME/IP, DDS, HTTP/REST based remote procedure call protocols, ...)
- How to reference VSS signals in VSC model
- Error handling feature built in
  - Build in some support for success/error return
  - Note: Method invocation errors that are related to the particular target/protocol are separate from this mechanism. Those are defined in target/protocol separate specification.
    Example: "method name not found" on a web request.
  - Support a global error type enum. Configurable. More than one.
  - Specify which errors are expected/applicable per method
  - Error feature is optional but best-practice. Out-parameters still exist of course, and can be (mis)used to report invocation success
  - Next: Make syntax proposal. Consider is this feature reasonable to translate/map to all IDLs, target environments that we envision. How is it mapped?
- Franca <-> VSC YAML mapping
  - document, compare keywords
- Franca further development to a "next version"?
  - Interesting if desired new features are not covered. E.g. error-feature might be one such thing.
- Set up documentation on GitHub Wiki instead, or as a complement
- Terminology, names. Coin a useful name for the "VSC IDL language"
- Review OpenAPI, AsyncAPI, gRPC/protobuf, Franca, etc. for syntax similarity (names of things) and feature coverage. Find alignment oppportunities.
  - One stakeholder suggested: Could OpenAPI be extended to cover non-REST. I.e. "OpenAPI-next" == VSC language? Rationale: OpenAPI known outside automotive.

Frequently Asked Questions:

- Why another IDL?
- Why is adopting <insert existing choice> not enough? Or is it?*Several answers can be found in previous presentations/slide decks*

References, previous presentations...

- COVESA All Member Meeting

Deployment:

- VSC Deployment - Containers