ALL MEMBER MEETING '21

Common Vehicle Interface Initiative Session 2

Technology Stack Implementation

October 06 2021

CVII – Technology Stack

AGENDA

- **Overview** of ongoing and planned tech-stack components and projects
- **VSC**: The potential for a common services language and the vehicle-service-catalog
- **Why vehicles need an event-driven platform** (Bosch)

Technology Stack definition:

"Any and all (software) technologies involved in the transfer and use of the *standard data model* and *standard services description model*"

Defined by:
- Implementation projects (e.g. open-source)
- Specifications.

ALL MEMBER MEETING '21

Technology Stack

Overview and Introduction

October 06 2021

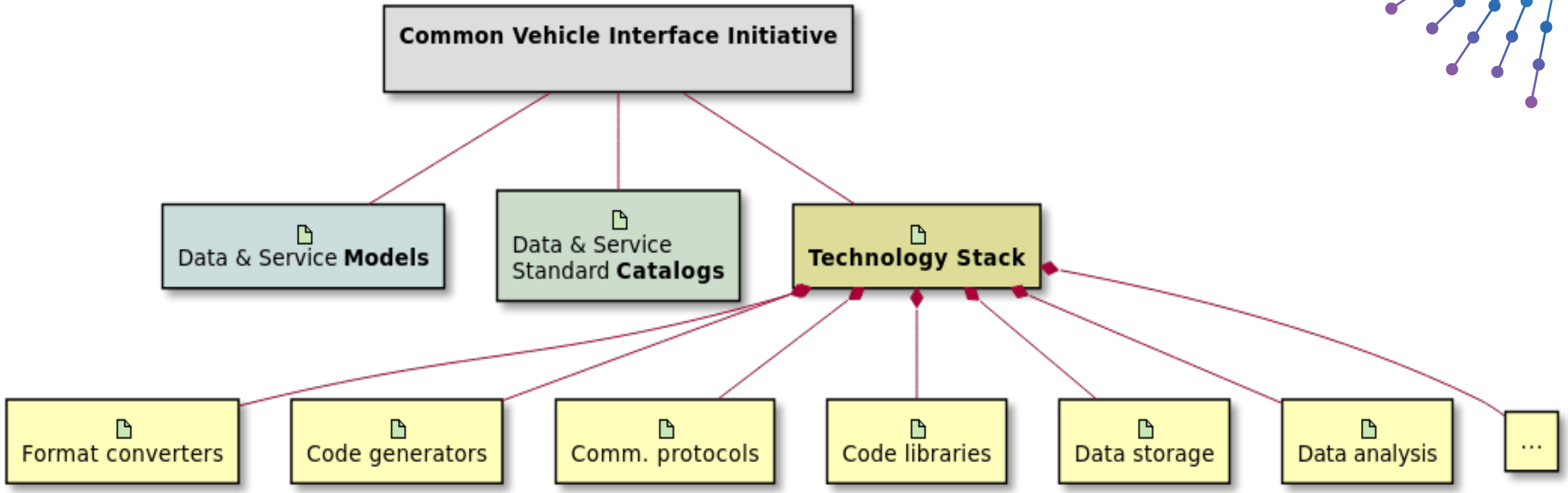
Technology Stack definition

“Any technologies (software) involved in the transfer and use of the *standard data model* and *standard services description model”*

<u>Defined by:</u>

- Implementation (Community development projects)
- Specifications

Technology Stack



Web Protocols, HTTP/REST
gRPC, GraphQL,
SOME/IP, DDS, ARA:COM, D-Bus, …

Protobuf, AVRO, NATS …
Spark, Kafka, NiFi, …
Time-series databases
Containers, cloud-deployment …

Tech Stack Projects

- Specifications, such as W3C VISS Protocol specification
  - Implementations, e.g. servers/clients following W3C spec
- VSS-Tools
  - Collection of tools of conversion from/to VSS
  - Code generation tools (more to do)
  - Franca, JSON, Protobuf, GraphQL,
    support code for C programming API & Go-programming API
  - Android Automotive Vehicle API (Vehicle Properties) from VSS data server (code generation)
- VSC-tools
  - Early implementation of service-to-code generation
  - Flexible, template-driven
- Framework / larger combination projects
  - including Bosch IoT-event-analytics, vehicle-edge, KUKSA and related projects
  - AOS project
  - PoCs, demos, many internally/proprietary or under development
- Company-internal tools, for VSS, Franca, not open-source
- CCS architecture implementation

ALL MEMBER MEETING '21

Vehicle Service Catalog

The case for a common services description language

October 06 2021

Why work on a "common service description model"?

- Possibly outlook for "common catalog(s)" of services, like we see for data
- Capture all automotive systems (vehicle to cloud) needs in one IDL
- Creation of a new, **flexible and lightweight tool suite**
- Make the required bindings between technologies used in automotive
  - Bind ARXML to gRPC, Protobuf IDL to SOME/IP, FrancaIDL to OpenAPI, etc.
- The Common *Data* Model, with VSS as its starting point, is clearly taking off
  - … but wherever there is data exchange, remote-procedure-calls are often requested
- Move towards an industry-standard *common language* for describing behavior of any subsystem
  - Bind ARXML to gRPC, Protobuf IDL to SOME/IP, FrancaIDL to OpenAPI, etc.

Consider Established Technologies

Interface/service descriptions

- OpenAPI
- AsyncAPI
- Franca IDL
- gRPC, …

Service invocation

- Web protocols, gRPC, …

Established Auto technologies

- SOME/IP, DDS, ARA:COM, …

Solution

- Discuss preferences
- Stay close to existing popular choices
- Work in concert with VSS development
- Use flexible, independent, intermediary format (YAML based)
- Develop better tools
  - Convert/migrate from legacy IDLs
  - Bind to established protocols/sw

“What about OpenAPI”

- OpenAPI is optimized for REST*ful* HTTP interfaces
- $NEW_IDL (and Franca IDL) is flexible for **all types** of interfaces
- We aim to use a **main** IDL that is generic
- OpenAPI *should* be part of the development ecosystem
  - (convert to/from existing interface descriptions, leverage OpenAPI tools)

“What about AsyncAPI”

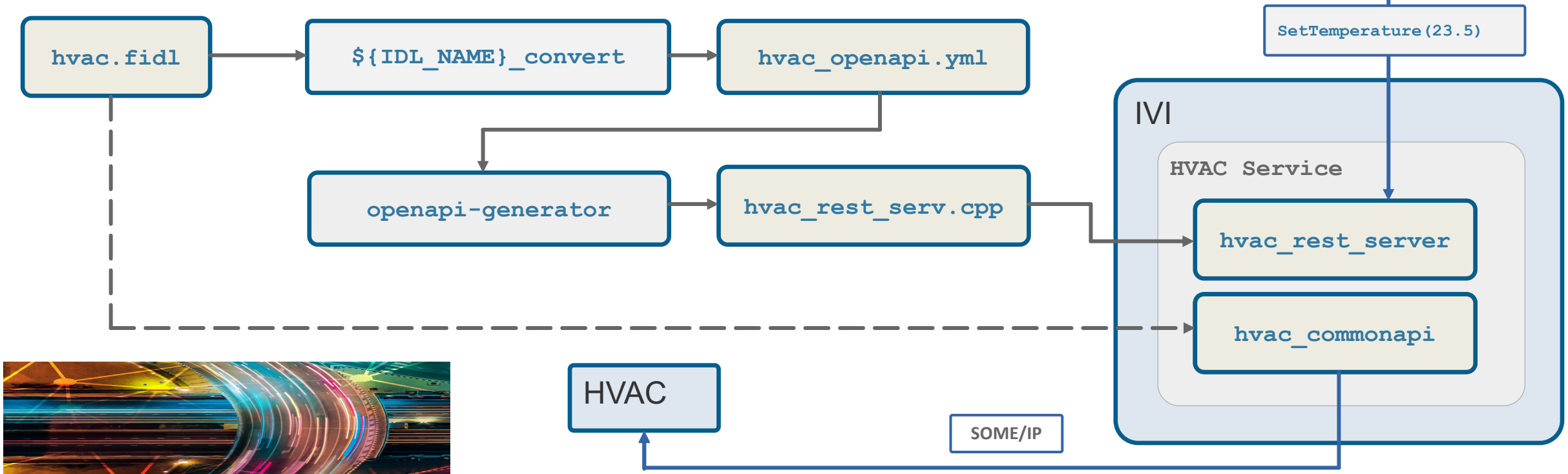
- AsyncAPI describes publish/subscribe data exchange interfaces, not generic method calls, etc.
- $NEW_IDL (and Franca IDL) is flexible for **all types** of interfaces
- We aim to use a **main** IDL that is generic
- AsyncAPI *should* be part of the development ecosystem
  - (convert to/from existing interface descriptions, leverage OpenAPI tools)
- Being pub/sub focused, it is rather a discussion for the VSS signal ecosystem

Planned approach

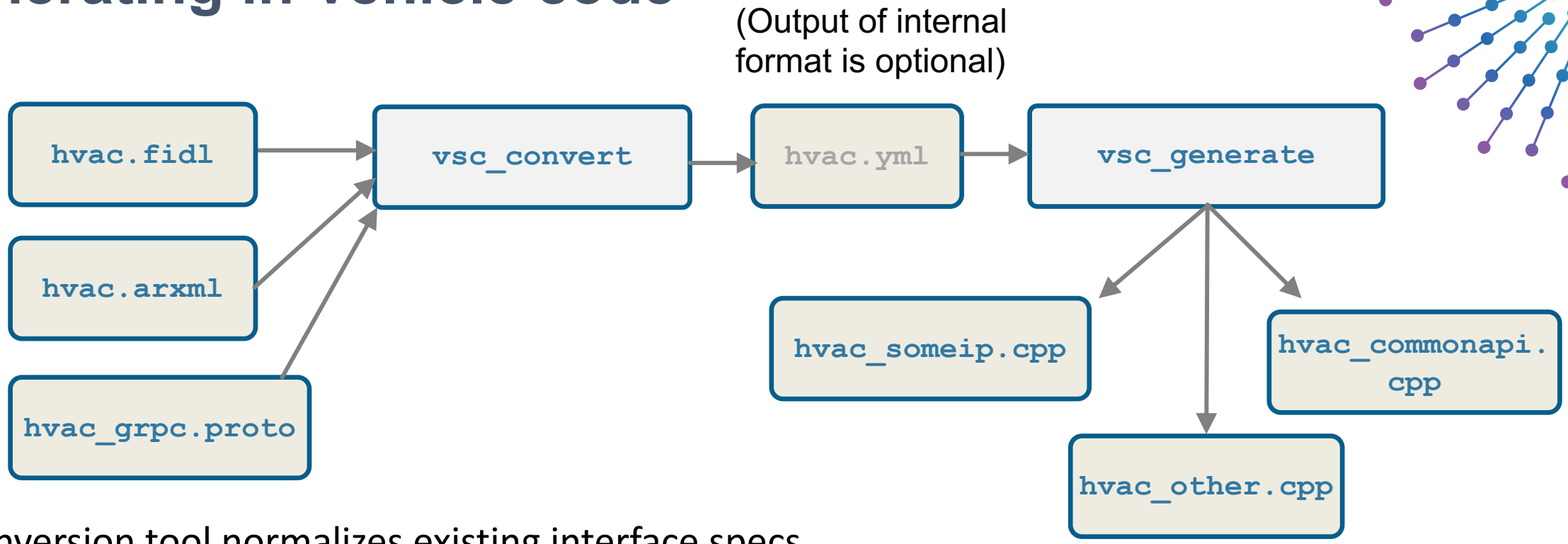
- Define an automotive-tailored, **generic** YAML-based IDL description format
  - Stay close to established YAML-based formats (OpenAPI, AsyncAPI, …)
  - Stay semantically compatible with Franca IDL, and popular technologies
  - Learn from good Franca IDL design (IDL separate from deployment model)
  - Goal: Practical interoperability with **VSS,** covers data/services together
- Seek alignment on which is the *preferred* description format for common service catalogs
  - Is it standard Franca IDL?
  - Is it YAML-based format?
  - Is it other, e.g. gRPC?

Tools process *preferred* format and enable interoperability

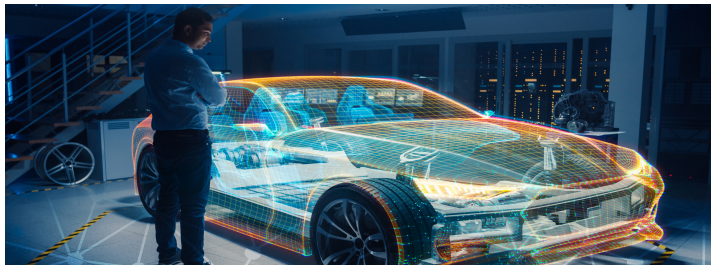
- Example: Generate **OpenAPI**-based REST server from existing **FrancaIDL**
- Automates error-prone protocol generation, integration, unit-test



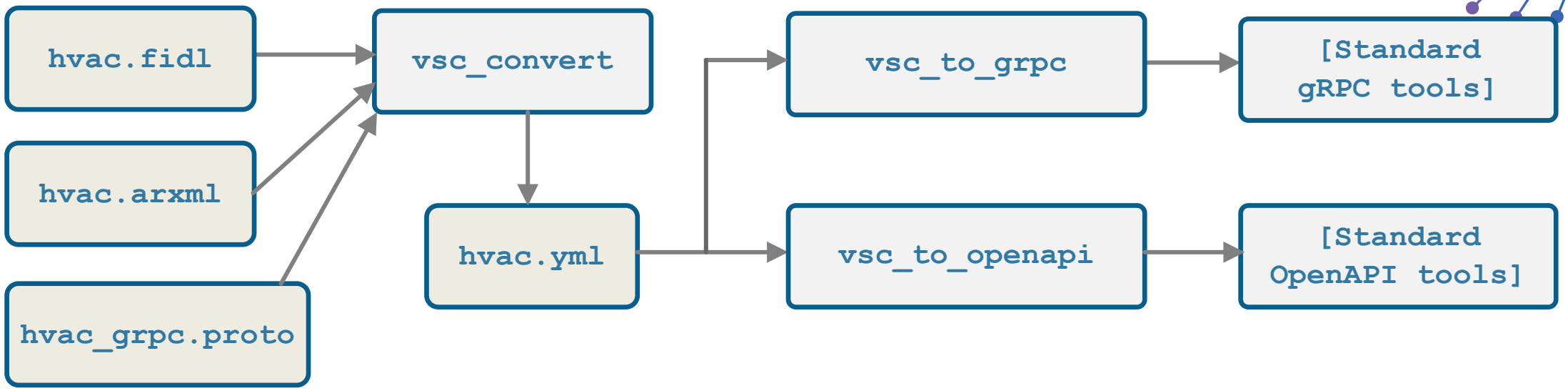
Generating in-vehicle code



- Conversion tool normalizes existing interface specs to internal services model format
- Code generation tool creates automotive-targeted stub code
- Links to CommonAPI and other existing stacks

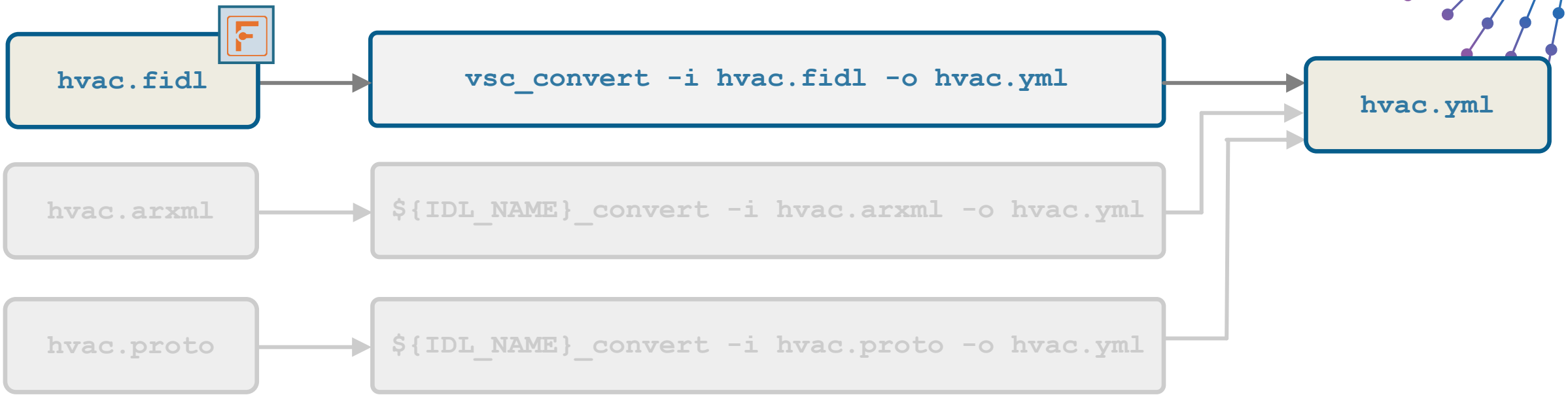


Generating non-vehicle code



- Single IDL specification used as single source of truth that is fed into automated tool chains
- Normalized VSC specification can be converted to multiple other target IDL formats
- Standard target tooling used to create stub code for non-vehicle deployment

Where is Franca IDL in all this?



- Franca IDL *could* be the *preferred* input format
- Bidirectional translation between Franca IDL and Vehicle Service Catalog formats fully supported
  - VSC could be seen as a YAML variant of Franca IDL with no loss of information
  - VSC drives improvements to Franca IDL → next step in Franca evolution?

Common Vehicle Interface Initiative (CVII)
Alignment recap

- **CVII** drives the automotive industry conversation around alignment of core standards and technologies
- **CVII Tech Stack** assumes *data-model* and *services-model* commonality have been or will be achieved in other tracks
- **CVII Tech Stack** selects/aligns on a *reasonable number* of protocols and technology bindings

Approach:

1) ? “Develop” a full-featured IDL with heavy influence from existing choices, Franca IDL in particular
2) Provide tools/bindings to core technologies with a *simple and extensible* approach
3) Create conversions to/from other choices *where appropriate*:
   - To ensure smooth migration
   - To ensure efficient leverage of existing ecosystems and implementations
4) Promote movement over time towards *industry-standard* IDL
5) Avoid *everything-to-everything* conversion approach, which simply continues fragmentation

   Strategic and methodical avoidance of the **XKCD standards effect**

(*Google it if by any chance you need to)

Corporate adoption strategies

- Keep existing IDL specification library: minimize disruption
- Normalize library of definitions to VSC format
  -> leverage conversion / generation tools
- Normalize a vendor-provided IDL spec to VSC format to leverage same tool chain
- Use IDL spec to write automated tests that can validate multiple service protocols (gRPC, SOME/IP, etc)
- Optionally, develop new specifications directly in VSC format, gradually retiring original IDL format, participating in the definition of industry-standard service catalog(s)

Common Vehicle Interface Initiative (CVII)
Alignment recap

- **CVII** drives the automotive industry conversation around alignment of core standards and technologies
- **CVII Tech Stack** assumes *data-model* and *services-model* commonality have been or will be achieved in other tracks
- **CVII Tech Stack** selects/aligns on a *reasonable number* of protocols and technology bindings

Approach:

1) ? “Develop” a full-featured IDL with heavy influence from existing choices, Franca IDL in particular
2) Provide tools/bindings to core technologies with a *simple and extensible* approach
3) Create conversions to/from other choices *where appropriate*:
   - To ensure smooth migration
   - To ensure efficient leverage of existing ecosystems and implementations
4) Promote movement over time towards *industry-standard* IDL
5) Avoid *everything-to-everything* conversion approach, which simply continues fragmentation

   Strategic and methodical avoidance of the **XKCD standards effect**

(*Google it if by any chance you need to)

Why Yet Another Standard?

- **Language and protocol agnostic**
  We need to try out different languages, protocols, and philosophies before we commit to something we want to standardize

- **Scale across 100s of interoperating services**
  Name spacing, interface imports, deployment models, and API vs. Implementation version management are all needed in large-scale deployment

- **Lightweight**
  CLI oriented. Five minutes to running tutorial. Small, componentized codebase

- **Cross-IDL portability**
  We need to be able to import (and export) existing IDL formats into a generic, easy-to-parse syntax while maintaining semantic equivalence

Market drivers to standardize services

- **OEM drivers**
  - Use standardized APIs to decouple solutions from vendor-specific technologies
  - Push for standard-compliance in RFIs & RFQs to ease side-by-side bid comparison
  - Use open source, standardized tools, and joint industry effort to create a higher starting point, allowing programs to focus resources on brand-differentiating experiences

- **Tier 1 & 2 drivers**
  - Implement standardized API to minimize program customization and maintenance, migrating toward off-the-shelf offers to OEMs
  - Portal/Host value-added services from third parties

- **Non-automotive drivers**
  - Manage mixed-asset fleets with minimum of system integration and maintenance
  - Widen and accelerate market for new 3rd party automotive services

Call to action!
Join to resolve open questions:

- Are you *on-board* with the creation of the automotive-industry common services description model?
  - Yes, No / Why not?
- Preferred IDL for service catalog(s) in - VSC YAML or FrancaIDL?
- Which input formats to support for carefully selected* interoperability
- Which output formats to support
- Which target protocols do we need to generate code for?
- * (avoid everything-to-everything conversion strategy. Fight fragmentation)

ALL MEMBER MEETING '21

NEXT: Why vehicles need an event based system? (Bosch)

ALL MEMBER MEETING '21

AFTER THE BREAK:

CVII Session 3

Alignment and Adoption

SOTA and Insurance