Central Data Service playground proposal

Status

This proposal is a Work in Progress (WIP).
The outline used to create this document can be found in a sub-page here.

Introduction

This page describes a proposal for community led joint work on a Central Data Service playground.

The origins of the proposal are discussions in the Data Architecture / Infrastructure pillar of the Data Expert Group and an idea the author discussed with Christian Muehlbauer and Felix Reichenbach at the Spring 2023 AMM. The author thanks the Data Architecture pillar members for their input.

Why?

Covesa needs:

- A neutral playground to investigate internals of data services. In the context of data-centric architectures for example.
- A neutral playground to investigate, illustrate and disseminate combining data services and the Covesa eco-system with other parts of the vehicle and off-board. For example, to demonstrate how VSS data can be used with VISS for newcomers.
- Repeated use of shared terminology, patterns and tools leads to quicker understanding in discussions, shared costs in development and can lead to quicker outcomes.
  - Jump from 'what is the box' to 'how are we using the box' for example.

Logical Concepts

From a communication and community perspective it is important to maintain descriptions of the logical concepts. Discussion at a logical level allows different parties to collaborate on common concepts, whilst making different implementation decisions, e.g. in product/technology selection or system architecture for example. That however does not mean we need spend months in philosophical discussions before moving to implementation. Instead logical concepts (why, what) can be developed alongside implementation.

It is therefore proposed that the project maintain concepts for both logical and implementation alongside each other.

There is a parallel proposal within the Data Architecture group for documenting Covesa design patterns, architectures and Howtos, that could be used to publish such documents that result from work on this proposal.

Central Data Service Playground (Why, What, How)

Why Central Data Service?

- A repeating pattern of discussion in the Data Architecture pillar is the combination of VSS Data Server and VSS Data Store and their connection southbound to feeders/native data and northbound to clients and off-board. Hence *Data Service*.
- VSS is a mechanism of abstraction. The logical architecture for the Data Expert Group and VSS places operation in the zonal ECUs and above. Discussion of next-gen and data-centric architectures suggests investigation into data services in zone, domain and HPC controller scenarios and the cooperation between them. Hence *Central*.
- The name Central Data Service is not an attempt to introduce a new category of component. It is used here simply as a useful synonym for what otherwise would be a longer descriptive phrase explaining combinations of VSS centric Data Server and Store and their location in the vehicle.

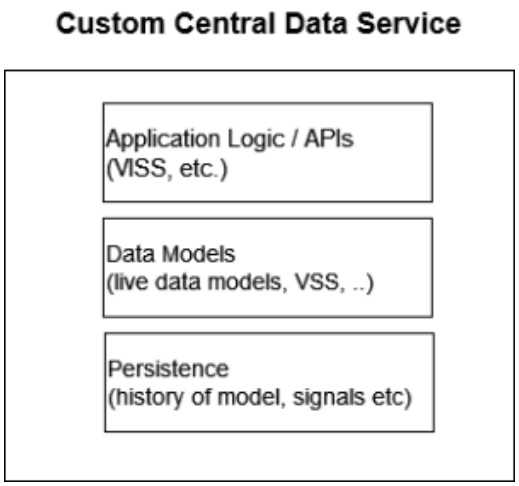
Why playground? Why not PoC?

- A PoC is often a snapshot in time and often specific in scope. *Playground* suggests greater flexibility. The central service, the *Lego building block*, is intended to be flexible and evolving. Similarly with what it is combined with to illustrate Covesa concepts and technology.
- The Playground could certainly be *used* to implement a PoC.
- Patterns such as view/controller, out of the box data servers, data stores linked to applications (e.g. SQLite) etc are well known. The Service could in part be defined by what's not known, by open questions in next-gen architectures such as data-centric architectures, that needs to be tackled down. The Playground is the means to doing that and illustrating the results.

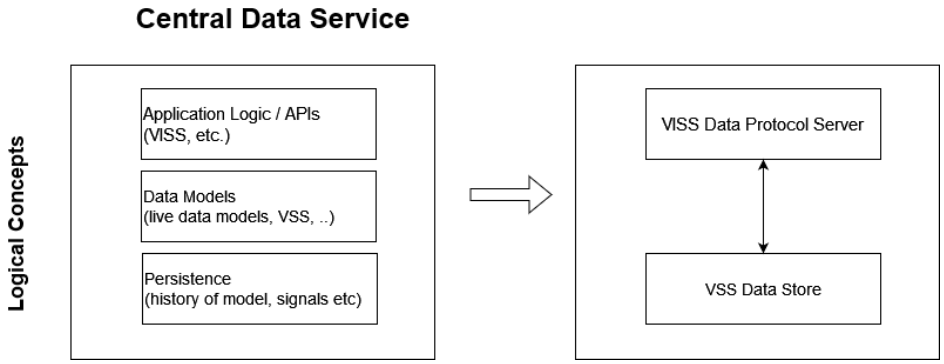
How?

- Address two high level implementation needs, keeping in mind a path towards production where possible:
    1. Easy to develop: make it easy to build, modify and trial by providing an instance running on a host, e.g. using Docker container(s).
    2. Closer to production: the same base code should be deployable to systems closer to production, including on automotive hardware (or its simulation), e.g. Yocto, container orchestration, SOA etc.
- A path towards production can be supported by using production components, rather than overly simple substitutes, where it makes sense. For instance a particular scenario may use Kafka in a cloud connection. The point is not to pick a winning product in a particular category, but to recognise that using a production tool can represent a category that is known to scale. Detailed requirements for a specific production project and product selection for it, is rightly left to that project.
- A generic code base for the basic building block should allow flexible compilation to meet those needs on x86 and ARM. The target for how it is used being a matter of deployment at a high level.
- Follow the OSS mantra of adopt where you can, extend if needed, create where necessary.

What?



- At its core the service has requirements in three key areas:
    - Data Models: the live data models - VSS as the abstracted view of the vehicle, along with other adjacent data models such as personal data.
    - Persistence: history of the model and signals etc - historical and cached timeseries data.
    - Application logic / APIs: for accessing the data such as VISS or GraphQL.
- It is recognised that additional features, such as synchronisation are absolutely desirable and have been a part of discussions in the Data Architecture group. For this *proposal* a base feature set is described to help readers quickly grasp the concept. If the proposal proceeds then additional details will be created collectively based on interest and participation.
- As a *starting point* it is suggested that the Service could be realised as a basic building block combining VISS Data Protocol Server with highly functional VSS Data Store.

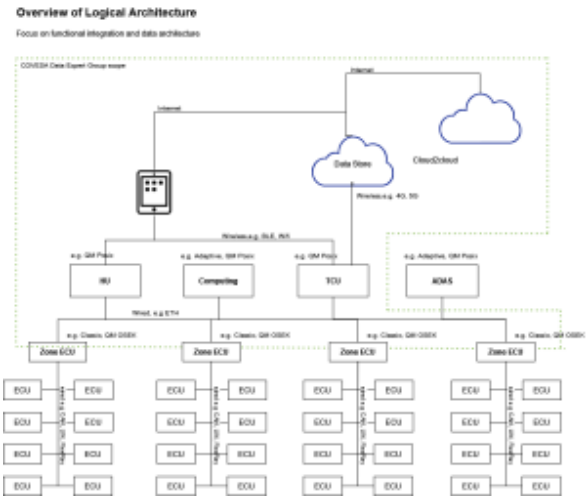


- As well as the mentioned flexibility in implementation, flexibility in use is also intended:
    - With some supporting documentation it could help meet the ongoing request from newcomers to the VSS eco-system as to how VSS can be used.
    - The Data Architecture group for instance has various topics it wishes to investigate related to data-centric architectures.
    - The playground can be used to investigate *internals* of such a service. For example, connecting VISS to medium and high speed data, or adding a protocol.
    - *External* connections with other systems may also be a focus. For example, combining the Service with other components to implement a particular touchpoint such as mobile.
- Further details can be found in the following sections.

The Service in context

To help understand its use lets quickly place the service in context.

Big picture: The logical architecture for the Data Expert Group places the scope of the work as Zonal ECUs and above as shown in the following diagram:



It is assumed that the service would likely be deployed on a Zone, Domain or Central controller, with corresponding h/w capabilities.

Interaction between Large ECU: In the Data Architecture group it is recognised that zone/domain specific data services will need to *synchronise* and *cooper ate* between themselves and/or with a central vehicle computer, e.g. Inter-controller sync/cooperation:

- blocked URL

In-vehicle southbound: The service will integrate *southbound* to lower parts of the vehicle and its native data, through data feeders and connectors. This will include making connections to other systems such as Autosar etc.

Northbound: connections will be made to clients, mobile, cloud and major in-vehicle domains such as IVI running Android/Apple etc.

Logical domains: Connections may also be made to other *logical* data domains. For example, there is knowledge layer proposal made in the Data Architecture Group that discusses the separation of concerns and interaction between knowledge, information and raw data layers. The Service described here could be used to provide the data layer services in its investigation.

Success Factors

1. Newcomers to Covesa technology use the playground to accelerate their understanding of how the technology can be used. That could be a looking at a simple instance of how a VSS data server is combined with a VSS data store and queried using VISS. It could also be a more complex instance that combines components to illustrate a longer specific end to end use case, e.g. mobile to vehicle connection.
2. Internal groups within Covesa naturally use the logical concepts and the playground implementation in combination with other components to develop and disseminate ideas. This especially applies to the Data Architecture and Infrastructure pillar.
3. Supporting materials such as patterns, diagrams, cookbooks etc are adopted as useful assets within and outside Covesa, which in turn helps socialisation.

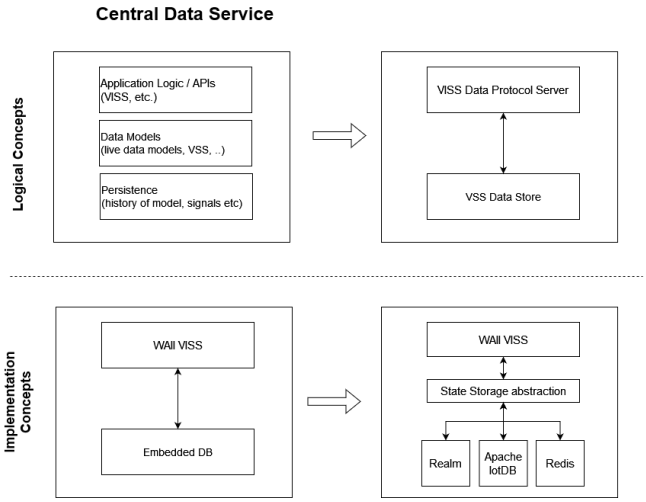
Implementation Concepts

The logical section above should be read to understand the concepts to be implemented. This section suggests starting points for discussion in the community.

Initial idea for the Central Data Service

As outlined in the logical description, as a *starting point* the Service could be realised as a basic building block combining VISS Data Server with highly functional VSS Data Store. The availability of generic code allows flexible deployment to meet the two high level implementation needs to support easy development trials, whilst also supporting investigation closer to production, including a path to production.

- Initial idea/sketch for base building block
  - Generic code: VISS Data Server with VSS Data Store backend
    - Data Architecture requirements: Add Apache IotDB (Apache eco-system, embedded and UDF) and Realm (embedded, sync) as backends to enable research.
    - Example using WAII VISS Data Protocol Server, which supports historical data and has a data store backend and various embedded databases:



- Deployment 1 - Easy to develop (assumed first target): x86 host Docker containers
- Deployment 2 - Closer to production (assumed second target): ARM64 using common automotive deployment, e.g. Yocto Linux.

- Investigation illustrations (examples):
  - Data Architecture and Infrastructure pillar
    - Sync
      - Multi-instance of deployment 1/2
    - Logical
      - knowledge layer proposal
        - e.g. sync up/down between knowledge generator and data layer
    - Data Service Internals (e.g. server, store connections)
      - Latency
      - Medium and high speed data
      - etc.
  - Touchpoints, common deployment scenarios, scenarios created by other Covesa groups etc.