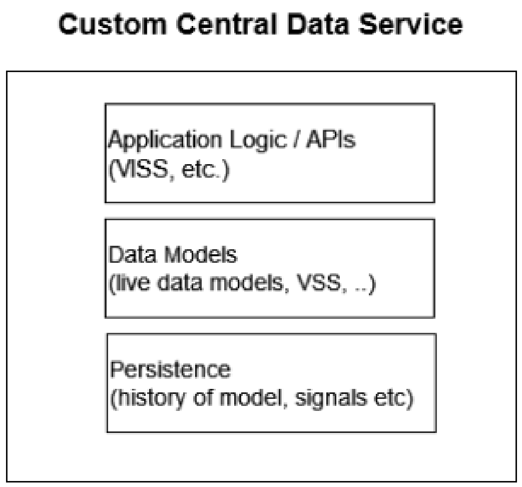
Central Data Service Playground outline

Outline starts here

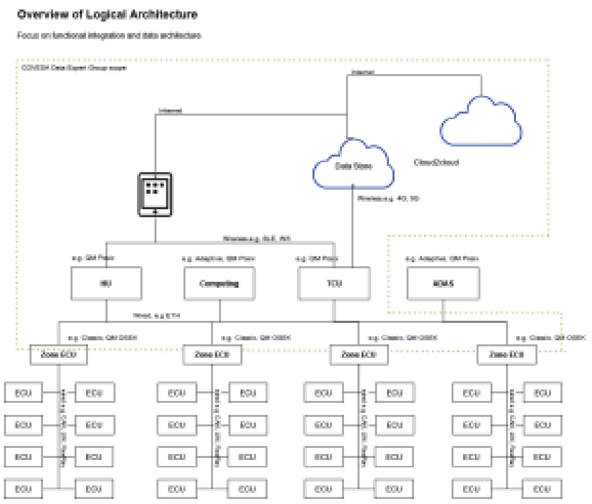
Below is the draft outline from which the proposal document was built. This is here for historical reference only.

Outline

- Intro
    - Why?
        - Covesa needs:
            - 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'
            - A neutral playground is needed to investigate internals of such services. In the context of data-centric architectures for example.
            - A neutral playground is needed to investigate, illustrate and disseminate combining data services and the Covesa eco-system with other parts of the vehicle. To demonstrate how VSS data can be used with VISS for newcomers for example.
        - 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*.
        - 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*, is intended to be flexible and evolving. Similarly with what it is combined with to illustrate Covesa concepts and technology.
            - Patterns such as view/controller, out of the box data servers, application stores 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.
    - What?
        - As a *starting point* a basic building block combining VISS Data Server with highly functional VSS Data Store.
        - This piece of lego can then be extended and reused in various ways. The Data Architecture group for instance has various topics it wishes to investigate related to data-centric architectures.
        - Details in following sections.
    - 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 on automotive hardware (or its simulation), e.g. Yocto, container orchestration, SOA etc.
        - 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.
        - Maintain logical conceptual descriptions of the work. This supports flexible technology choices and allows discussion of shared concepts between companies that may make different choices in system architecture or technology.
    - Success factors
        - 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.
        - 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.
        - Supporting materials such as patterns, diagrams, cookbooks etc are adopted as useful assets within and outside Covesa, which in turn helps socialisation.
- Logical concepts
    - From a communication and community perspective it is important to maintain descriptions of the logical concepts. That does not mean we need spend months in philosophical discussions before moving to implementation. Instead logical concepts (why, what) can be developed alongside implementation.
    - Central Data Service
        - 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.
        - At its core the service has features in three key areas:


Custom Central Data Service
Application Logic / APIs (VISS, etc.)
Data Models (live data models, VSS, ..)
Persistence (history of model, signals etc)

        - It is recognised that additional features, such as synchronisation are 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.
    - Service in context
        - Zone/domain/HPC controller (system view)
            - The service is likely deployed on a Zone, Domain or Central controller.
            - Logical architecture for VSS - zonal ECU and above:



            - In the Data Architecture group it is recognised that zone/domain specific data services will need to synchronise and cooperate between themselves and/or with a central vehicle computer, e.g. Inter-controller sync/cooperation:
            - blocked URL
        - Southbound: connections to feeders, Autosar etc.
        - Northbound: connections to clients, mobile, cloud, in-vehicle Android/Apple, etc.
        - Logical: prove out connections between data layer and other logical data domains. For example, to knowledge generating concepts as suggested in the knowledge layer proposal made in the Data Architecture Group
- Implementation concepts
    - 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 on automotive hardware (or its simulation), e.g. Yocto, container orchestration, SOA etc.
    - 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.
        - Deployment 1 (assumed first target): x86 host Docker containers
        - Deployment 2 (assumed second target): ARM64 using common automotive deployment, e.g. Yocto Linux.
    - Investigation sketches:
        - 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.