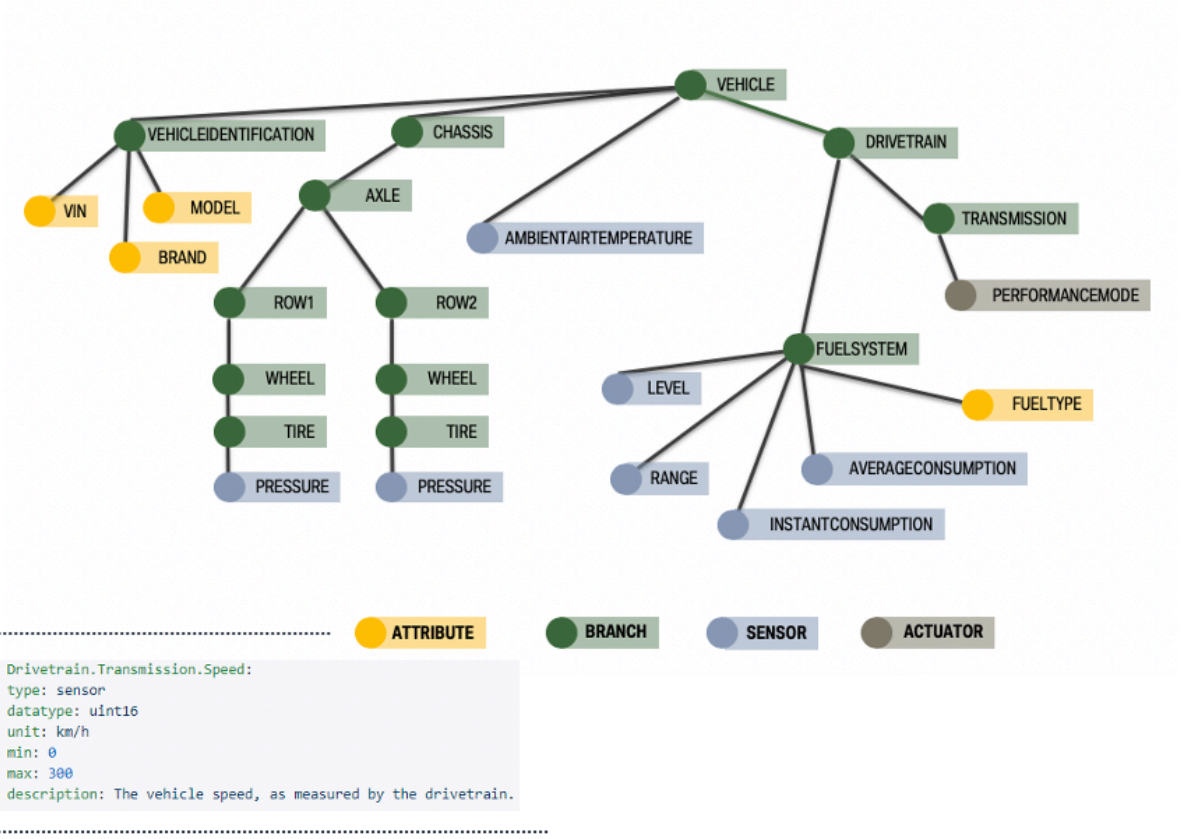
Covesa proposal to hierarchical data model for VSS

Author: swen.schisler@endava.vom

VSS

The Vehicle Signal Specification introduces a domain taxonomy for vehicle signals. In short this means that VSS introduces:

- A syntax for defining vehicle signals in a structured manner.
- A catalog of signals related to vehicles.




```
- Drivetrain.Transmission.Speed:
  type: sensor
  datatype: uint16
  unit: km/h
  min: 0
  max: 300
  description: The vehicle speed, as measured by the drivetrain.
```

For example:

id	EngineOil.Temperature
datatype	float
type	sensor
unit	celsius
description	EOT, Engine oil temperature.

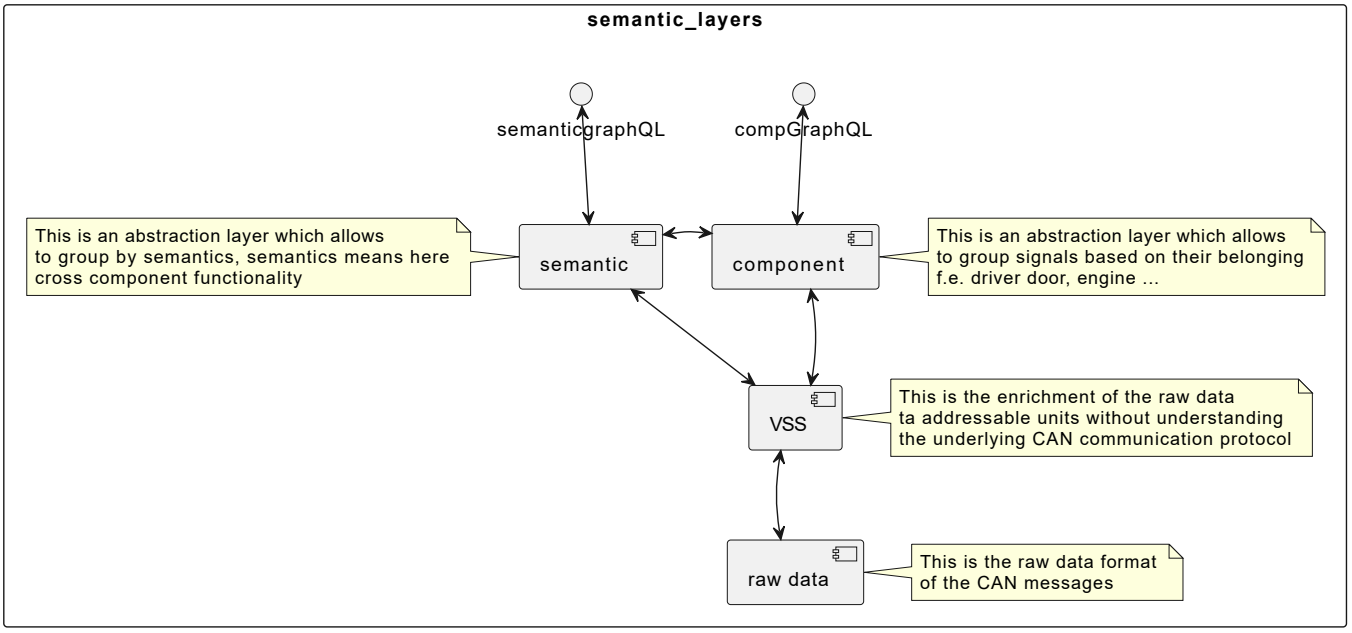
Problems with VSS (collection)

- Long identifiers
- missing semantic layers with corresponding API's

The use of Overlays will help to extend VSS with semantic models.

Layered semantic model

The proposal is to use a hierarchical Model (similar to the proposal by Ulf/Ford) to get more complex structures.



raw data format

The raw data layer represents all the data travelling on cars bus-systems back and forth. This can range from single data types (value, unit) to more complex one. All of them are usually not suitable for web protocols (usually CAN).

VSS is the layer which builds a flat list of all available signals and describe what they contain and also defines a representation. It abstracts the car bus system to a data oriented access.

If they're made accessible via API's they need to be well protected and rate limited to avoid damage by misusing the signals.

component layer

The component level should be used by car makers to associate VSS signals to components (f.e. door, engine, gearbox...). It is a view that a car maker uses to identify dependencies between specific parts and signals. The superset of all signals makes the car. Thus every modelyear, variant can have its on description. The root node is the vehicle with its characteristics.

id	Vehicle
type	SUV
MY	2024
variant	GT
description	2024 GT SpaceRanger SUV

By using the overlays, we extend the base vss datatype with tags. These tags can be used to group single signals into a bigger context. For example:

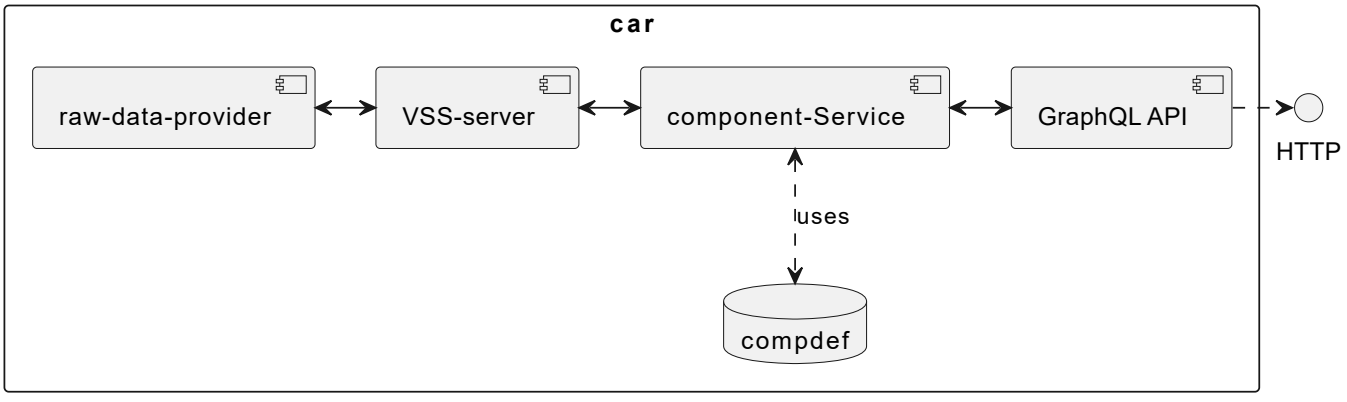
id	EngineOil.Temperature
datatype	float
type	sensor
unit	celsius
description	EOT, Engine oil temperature.
component	engine

On the component part the VSS signal is enriched with this capability. But to access this information someone needs to use the appropriate API (comGraphQL). A local database deployed in the car holds the car manufacturers definition for that vehicle.

This allows to query the API now to respond to questions like "All what belongs to the engine".

The interface is build on GraphQL and allows querying, information is provided as requested (requester specifies what he needs). And you can still access VSS signals indirectly via the component to VSS connector.

Implementation



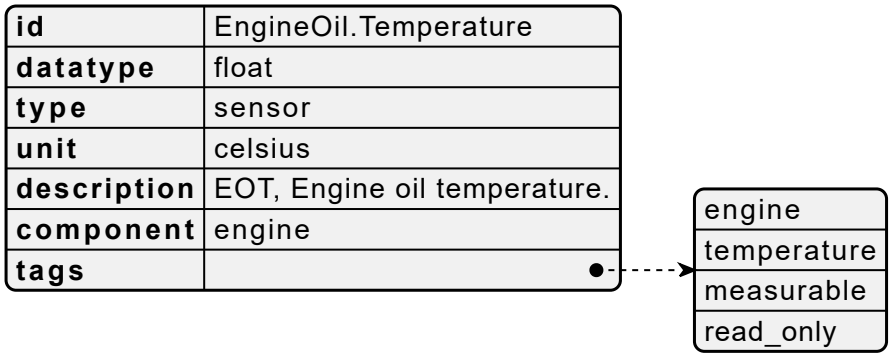
In GraphQL you can query and define the content of the returned values. GraphQL on component level would query the component-entity in the tree.

semantic layer

A semantic view can live inside but also outside of the vehicle. It is driven by use case and allow access to data, specific to that use case. On top of the VSS specification we introduce for every signal a tagging mechanism which will group the signals.

For the semantic layer we need a runtime changeable method to group signals. This is again held in a database and the mapping from signals to tags is used. The API searches for signals which have the same tag.

During runtime there is a configuration interface which allows to alter, append or delete tags in the database. The semantic API allows for use cases which are very complex.



This allows to query the API now to respond to questions like "All engine values", or "all temperature values" asoasf.

Physically the semantic API supports transaction based requests (GET,POST etc.), subscribe (push if changed), and also a streaming interface should be useful to get a continouos stream of data.

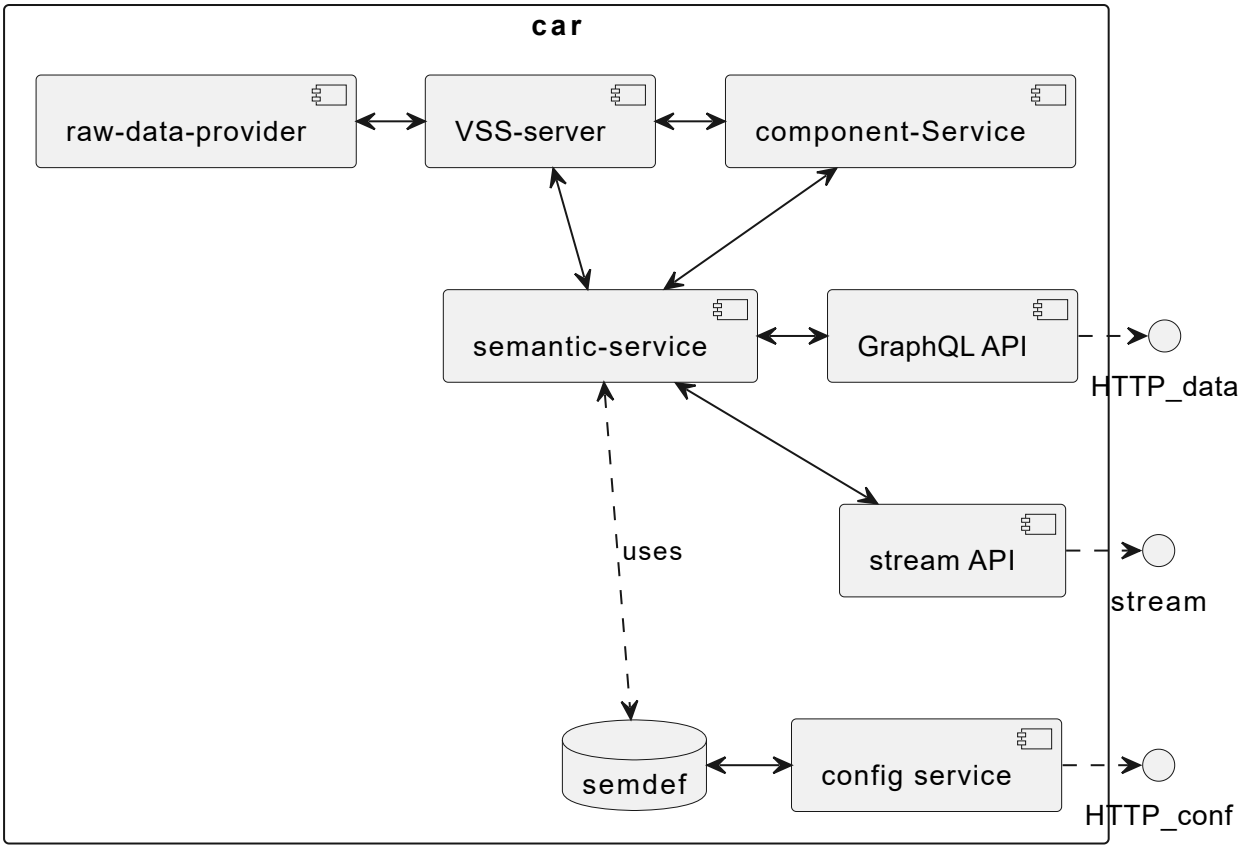
Every signal can be tagged by anyone with granted access to the API (it's a business model) on his wish.

The component and the semantic layers allow you to query based on tags. F.e. get door data will return all data that belongs to Signals tagged with door. To access a low level component you can resolve it with the tagging scheme. f.e. door lock driver side you will get the door data, get the lock id from the returned value and can access it directly via the VSS interface.

It requires a configuration interface as well, so that a service provider can add tags on signals to drive his use cases. F.e. a detection of all broken sensors after a crash could be queried to make a statement about the damage the car has suffered.

Implementation

Similar to the component layer



Access to API's

Low level (raw data) should not be exposed

VSS can be exposed, but the access and access frequency must be restricted (for write access)

component shall be exposed behind a working IAM solution and also access frequency limitations should be implemented (for write access)

semantic should be exposed with the same limitations as component

All access to API can be hidden behind a paywall. This is interesting as a business model. If a service provider want's to create a business use case, the access to this API's can be licensed by the car manufacturer.