# COVESA VSS – Gaps

Feb2025

Ford

## Lack of Signal Relationships

The VSS lacks a clear representation of relationships between signals, which significantly restricts its usefulness for advanced analytics. Example:

- In a Vehicle Sensor System (VSS), various signals are monitored, including engine temperature, engine temperature quality factor (QF), oil pressure, and coolant level. Here's how the relationships between these signals can be crucial:
  - An increase in engine temperature might indicate an underlying issue if accompanied by a decrease in oil pressure. However, without a defined relationship, detecting this correlation is challenging.
  - A high engine temperature might also be due to low coolant levels. Without linking these signals, the cause of temperature rise might be misdiagnosed.
  - The engine temperature QF indicates the reliability of the temperature reading. If the QF is low, it might suggest sensor malfunction or data noise, affecting the interpretation of temperature data.

What’s needed to effectively run advanced diagnostics or analytics -

- Establishing a relationship between engine temperature and coolant level can help in diagnosing cooling system problems more accurately.
- By explicitly defining the relationship between engine temperature and oil pressure, the system can alert users to potential overheating issues related to lubrication failures.
- By associating the engine temperature QF with temperature readings and other related signals, the system can adjust analytics or provide warnings about sensor reliability, ensuring more accurate diagnostics.



# Lack of Signal Relationships – Proposed Solution

Without defined relationships, it's challenging to understand how changes in one signal might influence others – This could be achieved with supporting signal relationships.

In this example, the analytics.engine_temperature signal has dependencies that reflect its relationship to other signals, helping to clarify how changes in coolant level, pressure, and quality factor can influence engine temperature.

```
analytics.engine_temperature:
  description: engine temperature is indirectly proportionate to engine coolant level and
    directly proportinate to engine coolant pressure and quality factor
  relationship: dependency
  signal: Vehicle.Powertrain.EngineCoolant.Temperature
  depends_on:
    - Vehicle.Powertrain.EngineCoolant.Level
    - Vehicle.Powertrain.EngineCoolant.Pressure
    - Vehicle.Powertrain.EngineCoolant.QualityFactor
```

```yaml
Vehicle.Powertrain.EngineCoolant.Temperature:
  description: "engine temperature is indirectly proportionate to engine coolant level and
    directly proportinate to engine coolant pressure and quality factor"
  type: sensor
  datatype: float32
  signal_context:
    operational_conditions: "Normal operation, not during engine startup"
    dependencies:
      - Vehicle.Powertrain.EngineCoolant.Level
      - Vehicle.Powertrain.EngineCoolant.Pressure
```

OR

Ford

# Lack of Signal Metadata

The current specification of the Vehicle Sensor System (VSS) lacks adequate metadata to comprehensively describe the signals for different cross-functional needs.

Sample example with metadata for different cross-functional and network serialization needs from AKM:

```yaml
EngineCoolant.Temperature.Value:
  datatype: float
  type: sensor
  unit: celsius
  description: Engine coolant temperature.
  admin_data:
    cyber_security: C3
    functional_safety: QM
    platform_system: motion_plus
    feature_apps: []
    feature_of_interest: EngineCoolant
  data_collection:
    privacy_domain: VehicleData
  representations:
    - signals:
        EngClntTeActlDiag:
          canonical_name: EngClntTeActlDiag
          type: can_signal
          long_name: Engine Coolant Actual Temperature
          deserialized_datatype: int16
          global_identifier: 928
          max: 215
          min: -40
          offset: -40
          resolution: 0.1
```

Example from VSS:

Cybersecurity

```yaml
Temperature:
  datatype: float
  type: sensor
  unit: celsius
  description: Engine coolant temperature.
```

Functional Safety

Data Collection Network Serialization

- conversions
- Unique identifier
- Init value
- Update period
- e2e properties

# Lack of Signal Metadata – Proposed solution

There is a need to support standard metadata incorporating aspects like cybersecurity, functional safety, data collection, and network serialization to support various cross-functional requirements.

This could also be accomplished using overlays; however, this approach may quickly become unmanageable, as different OEMs might develop their own schema, necessitating custom tools for every aspect if we don't provide support in the default schema.

```yaml
Vehicle.Powertrain.Engine.RPM:
  description: "The current revolutions per minute of the engine."
  type: sensor
  datatype: uint16
  unit: "rpm"

  metadata:
    cybersecurity:
      encryption: "AES-256"
      authentication: "HMAC-SHA256"
      access_control: "RBAC"
      cybersecurity_rating: "Level 3"

    functional_safety:
      safety_critical: true
      safety_integrity_level: "ASIL 2"
      failure_modes:
        - "Engine over revolution"

    data_collection:
      sampling_rate: "100 ms"
      retention_policy: "30 days"
      data_privacy: "anonymized"

    network_serialization:
      cycle_time: "10ms"
      message_priority: "High"
      transmission_mode: "Periodic"
```

Ford

# Incomplete complex datatype definitions

The current approach to struct type definitions within VSS presents several limitations that need to be addressed for better usability and flexibility.

Here are some key improvements needed:

- Defining nested structs using branching strategy cumbersome.
- Currently, there is limited scope for reusing nested struct definitions.
- The generated file formats are frequently flat, and the resulting IDL files do not accurately represent the intended struct definitions.

Sample nested struct:

```
PowerTrainInfoSignalGroup.PowerTrainInfoStruct:
  type: struct
  description: "A struct with datatype property defined."

PowerTrainInfoSignalGroup.PowerTrainInfoStruct.AccumulatedBrakingEnergy:
  datatype: float
  type: property
  unit: kWh
  description: The accumulated energy from regenerative braking over lifetime.

PowerTrainInfoSignalGroup.PowerTrainInfoStruct.Range:
  datatype: uint32
  type: property
  unit: m
  description: Remaining range in meters using all energy sources available in the vehicle.

PowerTrainInfoSignalGroup.PowerTrainInfoStruct.time:
  type: struct
  description: "nested struct"

PowerTrainInfoSignalGroup.PowerTrainInfoStruct.time.TimeRemaining:
  datatype: uint32
  type: property
  unit: s
  description: Time remaining in seconds before all energy sources available in the vehicle
```



# Incomplete complex datatype definitions – Proposed Solution(s)

This could be accomplished by simply enabling support for elements. Additionally, signal definitions could be allowed to be reused, as illustrated.

```
PowerTrainInfoStruct:
  description: "Represents the overall vehicle system."
  type: struct
  elements:
    - AccumulatedBrakingEnergyStruct:
      description: " "
      type: struct
      elements:
        - Value:
```

```
PowerTrainInfoStruct:
  description: "Represents the overall vehicle system."
  type: struct
  elements:
    - AccumulatedBrakingEnergyStruct:
      description: " "
      type: struct
      elements:
        - Value: PowerTrain.AccumulatedBrakingEnergy
        - QualityFactor: PlatformTypes.QualityFactor
    - TimeRemaining: PowerTrain.Time
```



# Our Next Steps

1. We'll concentrate on enhancing signal completeness by adding new signals to VSS spec.
2. We will also investigate and extend tooling support to generate C/C++ bindings from the spec.
3. Looking forward to participate in working sessions to help address the gaps highlighted in this presentation.