VSS to AOSP translation - WBS

Binding of VSS to Android Properties

There is a need to store the mapping between the VSS and Android properties. The following needs to be considered:

- Map between VSS leaf to Android Property ID
- Map between VSS leaf to Android Area
- Datatype conversion (e.g. int8 to float)
- Translation

As an example, it might look like

```
- Vehicle.Powertrain.FuelSystem.Level
        aospId: VehicleProperty::FUEL_LEVEL
        aospArea: VehicleArea::Global
        translation:
                -complex: „$INFO_FUEL_CAPACITY * _VAL_ / 100”
```

where complex translation is described with another language considered in different ticket

NOTE: The first analysis of translation and description of linear-equation translation is in

AASIG-122 - Jira project doesn't exist or you don't have permission to view it. - Spreadsheet

How to describe the complex translations

There is a need to store how to translate the signal from VSS to Android Property. The idea is to have some meta language that can use the references to the other signals and mathematical equations

As an example

```
- Vehicle.Powertrain.FuelSystem.Level
        translation:
                -complex: „$INFO_FUEL_CAPACITY * _VAL_ / 100”
```

By the above it means that: To translate the

`Vehicle.Powertrain.FuelSystem.Level` into `VehicleProperty::FUEL_LEVEL` the result value is = VehicleProperty::INFO_FUEL_CAPACITY * / 100.

It means that this should be generated to something like:

```
conversionMap["Vehicle.Powertrain.FuelSystem.Level"] = std::bind(convertFuelLevel,
           std::placeholders::_1, VehicleProperty::FUEL_LEVEL, toInt(VehicleArea::GLOBAL));

// ...

static VehiclePropValue convertFuelLevel(std::string value, VehicleProperty id, int32_t area) {
    VehiclePropValue prop = initializeProp(id, area);
    uint8_t valInt = std::stof(value);
    float mililiters = GET_PROP(VehicleProperty::INFO_FUEL_CAPACITY) * valInt / 100;
    prop.value.floatValues = std::vector<float> { mililiters };
    return prop;
}
```

Enumeration binding

There is a need to provide a way to bind enumerations, whether by meta language (see ticket [How to describe the complex translations]).
In Android: Subscribe for TIRE_PRESSURE for left-front wheel. In VSS is more flexible (the entities of TIRES can be more than 4)

Instances to VehicleArea binding

For the AOSP Props that have multiple instances, another level of binding needs to be introduced.

Generation of the map and translation functions

By idea the VHAL implementation is using the map between the VSS and Android Properties to translation:

```
VehiclePropValue aospValue = conversionMap[vssId](vssValue);
```

Conversion map fragment:

```
 conversionMap["Vehicle.Powertrain.FuelSystem.Level"] = std::bind(convertFuelLevel,
            std::placeholders::_1, VehicleProperty::FUEL_LEVEL, toInt(VehicleArea::GLOBAL));
```

The map can be included as a header file during compilation from the deployment files described in other ticket.

Inverse translation (form Android Properties to VSS)

There is a need for inverse to use "setProperty" from Android partition

Multiple instances of signal

Introduced new example: The Tire Pressure signal which has multiple instances in the car

```
- Vehicle.Chassis.Axle.Row1.Wheel.Left.Tire.Pressure
        aospId: VehicleProperty::TIRE_PRESSURE
        aospArea: VehicleAreaWheel::LEFT_FRONT

- Vehicle.Chassis.Axle.Row1.Wheel.Right.Tire.Pressure
        aospId: VehicleProperty::TIRE_PRESSURE
        aospArea: VehicleAreaWheel::RIGHT_FRONT

- Vehicle.Chassis.Axle.Row2.Wheel.Left.Tire.Pressure
        aospId: VehicleProperty::TIRE_PRESSURE
        aospArea: VehicleAreaWheel::LEFT_REAR

- Vehicle.Chassis.Axle.Row2.Wheel.Right.Tire.Pressure
        aospId: VehicleProperty::TIRE_PRESSURE
        aospArea: VehicleAreaWheel::RIGHT_REAR
```

The idea is to have each VSS leave mapped to a specific pair aospId + aospArea. In Android types. hal defines multipe areas (generic and some signal specific). See https://developer.android.com/reference/android/car/VehicleAreaWheel https://developer.android.com/reference/android/car/VehicleAreaSeat

It is not sufficient for all Car configurations. Open questions:

* When the Car has multiple "rows", which AOSP Area should be chosen to query for "Right rear" * What if the CAr has multiple tires on each side and axle?

Namespace "Android specific fields in VSS leave"

Current proposal includes the prefix "aosp" for each aosp-specific "decoration" of VSS Leaf object. Proposal of using the namespace to group the AOSP specific fields + adds some meta fields with versioning and other needed informations.

AD: Core fields for each namespace can be standardized inside Genivi.

Example:

```
- Vehicle.Chassis.Axle.Row1.Wheel.Left.Tire.Pressure
    AOSP:
      version: 2.0
      package: android.hardware.automotive.vehicle
      id: VehicleProperty::TIRE_PRESSURE
      area: VehicleAreaWheel::LEFT_FRONT

    ANY_OTHER_OPERATING_SYSTEM_OR_IPC:
      version: 1.0
      package: com.oem.foo
      other_data_for_translation: true
```

Pros:

- the fields introduced in namespaces can be kept in other vss layer files - less maintenance cost of conflicting field names
- Enables clean code generation.
- Relevant only during development process.

Cons:

- VSS Leaf starts to be a complex structure

Translation description

There is a need to translate the value from VSS to AOSP and vice versa when the signal originates in Android. For complex equations (like FuelLevel) there is no straight forward way for the compiler to inverse the translation.

Define the fields for bi-directional translations

To solve above we might have 2 fields. translation_set and translation_get (names are chosed arbitrary and are subject to be agreed)

For example when a need to convert from kpa(vss) to pa(aosp)

```
- Vehicle.Chassis.Axle.Row1.Wheel.Left.Tire.Pressure
    aospId: VehicleProperty::TIRE_PRESSURE
    aospArea: VehicleAreaWheel::LEFT_FRONT
    aospTranslationFromVSS: "_VAL_ / 1000"
    aospTranslationToVSS: "_VAL_ * 1000"
```

Focus on non-complex translation like linear

One additional field that represents the linear equation: aospValue = k * vssValue + m. The translation in other direction can be determined.

```
- Vehicle.Chassis.Axle.Row1.Wheel.Left.Tire.Pressure
    aospId: VehicleProperty::TIRE_PRESSURE
    aospArea: VehicleAreaWheel::LEFT_FRONT
    aospTranslationFromVSS:
      k: 1000
      m: 0
```

Approaches to define and evaluate calculation formula

1. Use RPN because it is easy to parse and process, no precedence rules to enforce etc.
2. Use standard math expression (with parantheses) and create a parser/interpreter
3. Copy standard math expression directly into target language (works in most cases)

=> Approach 3 seems to be our preference for now.

Next Actions

- Feedback from VSS working group about general strategies for metadata (hierarchical?) Gunnar Andersson
- Start work on code generator in vss-tools/contrib, output according to chapter "**How to describe the complex translations**" above.
    - Question: Introduce templating language (JINJA2) in vss-tools? It is used in vsc-tools and could be useful also for VSS for more complex code generation than we have today.
- Stefan awaiting approval to publish "hard-coded" example code.