CVII Tech Stack technologies - Protobuf

Information related to translation or use of Protocol Buffers (a.k.a. protobuf), together with the common data or service model.

- Recently a proposal for vspec2proto in vss-tools

Purpose:

- As a means to define an efficient binary serialization format
- As a means to reuse code-generators for skeleton-code to work with VSS data in various languages.

Alternatives:

- AVRO?

Useful links:

- Main site for Protocol Buffers: https://developers.google.com/protocol-buffers/
- Protobuf 3 reference specification
- Tutorials (Google) https://developers.google.com/protocol-buffers/docs/tutorials
- Comparison of data serialization formats: https://en.wikipedia.org/wiki/Comparison_of_data-serialization_formats

Concerns

- How to decide on groups of parts
- You have to provide all sub-branches even if only interested in one leaf - especially where a branch has both sub-branches and leaves, this is unexpected.
- Some groups make sense always, e.g. GPS.{Lat,Long}
- optionality is supported in Protobuf. Should all branches/leaves even be optional in the message definition?

One translation proposal encodes each branch as its own type (protobuf: *message)* . Leaves are, but branches (message types) containing branches (other message types) are also shown:

Example translation VSS -> Protobuf

```
message VehicleBodyWindshieldFrontHeating {
  bool Status = 1;
}

message VehicleBodyWindshieldFrontWasherFluid {
  bool LevelLow = 1;
  uint32 Level = 2;
}

message VehicleBodyWindshieldFront {
  VehicleBodyWindshieldFrontWiping Wiping = 1;
  VehicleBodyWindshieldFrontHeating Heating = 2;
  VehicleBodyWindshieldFrontWasherFluid WasherFluid = 3;
}

message VehicleBodyWindshieldRearWiping {
  string Status = 1;
}

message VehicleBodyWindshieldRearHeating {
  bool Status = 1;
}

message VehicleBodyWindshieldRearWasherFluid {
  bool LevelLow = 1;
  uint32 Level = 2;
}

message VehicleBodyWindshieldRear {
  VehicleBodyWindshieldRearWiping Wiping = 1;
  VehicleBodyWindshieldRearHeating Heating = 2;
  VehicleBodyWindshieldRearWasherFluid WasherFluid = 3;
}

message VehicleBodyWindshield {
  VehicleBodyWindshieldFront Front = 1;
  VehicleBodyWindshieldRear Rear = 2;
}

message VehicleBodyLights {
  bool IsHighBeamOn = 1;
  bool IsLowBeamOn = 2;
  bool IsRunningOn = 3;
  bool IsBackupOn = 4;
  bool IsParkingOn = 5;
  bool IsBrakeOn = 6;
  bool IsRearFogOn = 7;
  bool IsFrontFogOn = 8;
  bool IsHazardOn = 9;
  bool IsLeftIndicatorOn = 10;
  bool IsRightIndicatorOn = 11;
}
```

Protobuf as definition of serialization

The above is essentially describing the VSS data tree / catalog and is very similar to the definition of the full tree in GraphQL as can be seen here.

Alternatively we can focus on the serialization aspect. This is to define the exact *payload* in a generic communication protocol - e.g. MQTT which describes the transport and topic structure, but does not in itself define the payload format. A definition of what is actually transferred on topics is required:

The page Data serialization / value formats gives a high-level proposal for message types that are needed in a typical generic data transfer protocol.

Another option is to mirror the message structure of the VISS web protocol specification. Message types are described in the JSON Schema in the Transport Specification and this "schema" could be described using Protobufs.

```
TimestampedRecord:

message TimestampedRecord_uint32 {
   string ts = 1;   # Zulu time, ISO std with microseconds
   uint32 value = 2;  # (with specific uint32 type -- only valid for signals of that type)
}

# Or alternative that can encode multiple value types:
message TimestampedRecord {
   string ts = 1;   # Zulu time, ISO std with microseconds
   google.protobuf.any value; # There is a default "any" type defined by protobuf named google.protobuf.any but
it might not be the only choice
}

message GeospatialRecord {
  GeoPosition pos;
  TimeStampType ts = 1;      # Some efficient time stamp type?   Zulu time, ISO std with microseconds
  google.protobuf.any value; # ValueType needs to be a union/any/variant type.
}

# Just a proposal for a GNSS position type:
message GeoPosition {
  float latitude = 1;
  float longitude = 2;
  float accuracy = 3;
}
```

VISS message format mirrored in protobuf

VISS schema includes a lot of parts, some that seem optional (descriptions etc.)? There is also some hierarchy that could be included, or flattened.
In this example the hierarchy is roughly the same, the subtype "dp" is included and it encompasses the value and time stamp.

```
message VISS_data_message {
   string path = 1;
   dp_type dp = 2;
   # Other optional parts?  Description, properties ...
}

message dp_type {
   google.protobuf.any value = 1;
   string ts = 2;  # Also here, a more efficient time stamp type??
}
```