Open Geospatial Consortium

Approval Date: 2013-01-18

Posted Date: 2013-02-05

Reference number of this document: OGC 12-093

Reference URN for this document: http://www.opengis.net/def/doc-type/per/ows9-ssi-ugas

Category: Engineering Report

Editor: Clemens Portele

OGC® OWS-9 System Security Interoperability (SSI) UML-to-GML-Application-Schema (UGAS) Conversion Engineering Report

Document type:	OGC® Engineering Report
Document subtype:	NA
Document stage:	Approved for public release
Document language:	English

Abstract

The main scope of the schema automation activities in the OWS-9 initiative was twofold:

- Support for the SWE Common 2.0 XML Schema encoding rule
- Development of and support for an encoding rule for JSON instance data

In both cases the scope includes implementation of the encoding rules in ShapeChange.

In addition, an initial analysis of the possibilities for generating SWE Common 2.0 record descriptions from schemas in UML has been conducted and the results are described in this document.

The approach and results to both work items are described and discussed in this engineering report. This Engineering Report has been prepared as part of the OGC Web Services Phase 9 (OWS-9) initiative.

Keywords

ogcdoc, ows9, ssi, ugas, json, swe common, uml, 19109, geojson, gml, geoservices rest api

What is OGC Web Services 9 (OWS-9)?

OWS-9 builds on the outcomes of prior OGC interoperability initiatives and is organized around the following threads:

- **Aviation**: Develop and demonstrate the use of the Aeronautical Information Exchange Model (AIXM) and the Weather Exchange Model (WXXM) in an OGC Web Services environment, focusing on support for several Single European Sky ATM Research (SESAR) project requirements as well as FAA (US Federal Aviation Administration) Aeronautical Information Management (AIM) and Aircraft Access to SWIM (System Wide Information Management) (AAtS) requirements.

- **Cross-Community Interoperability (CCI)**: Build on the CCI work accomplished in OWS–8 by increasing interoperability within communities sharing geospatial data, focusing on semantic mediation, query results delivery, data provenance and quality and Single Point of Entry Global Gazetteer.

- **Security and Services Interoperability (SSI)**: Investigate 5 main activities: Security Management, OGC Geography Markup Language (GML) Encoding Standard Application Schema UGAS (UML to GML Application Schema) Updates, Web Services Façade, Reference Architecture Profiling, and Bulk Data Transfer.

- **OWS Innovations**: Explore topics that represent either new areas of work for the Consortium (such as GPS and Mobile Applications), a desire for new approaches to existing technologies to solve new challenges (such as the OGC Web Coverage Service (WCS) work), or some combination of the two.

- **Compliance & Interoperability Testing & Evaluation (CITE)**: Develop a suite of compliance test scripts for testing and validation of products with interfaces implementing the following OGC standards: Web Map Service (WMS) 1.3 Interface Standard, Web Feature Service (WFS) 2.0 Interface Standard, Geography Markup Language (GML) 3.2.1 Encoding Standard, OWS Context 1.0 (candidate encoding standard), Sensor Web Enablement (SWE) standards, Web Coverage Service for Earth Observation (WCS-EO) 1.0 Interface Standard, and TEAM (Test, Evaluation, And Measurement) Engine Capabilities.

The OWS-9 sponsors are: AGC (Army Geospatial Center, US Army Corps of Engineers), CREAF-GeoViQua-EC, EUROCONTROL, FAA (US Federal Aviation Administration), GeoConnections - Natural Resources Canada, Lockheed Martin Corporation, NASA (US National Aeronautics and Space Administration), NGA (US National Geospatial-Intelligence Agency), USGS (US Geological Survey), UK DSTL (UK MoD Defence Science and Technology Laboratory).

Contents

Page

Figures

Page

Tables

Page

OGC® OWS-9 System Security Interoperability (SSI) UML-to-GML-Application-Schema (UGAS) Conversion Engineering Report

1 Introduction

1.1 Scope

The main scope of the schema automation activities in the OWS-9 initiative is twofold:

- Support for the SWE Common 2.0 XML Schema encoding rule
- Development of and support for an encoding rule for JSON instance data

In both cases the scope includes implementation of the encoding rules in ShapeChange.

In addition, an initial analysis of the possibilities for generating SWE Common 2.0 record descriptions from schemas in UML has been conducted and the results are described in this document.

The approach and results to both work items are described and discussed in this engineering report.

1.2 Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Name	Organization
Clemens Portele (Editor)	interactive instruments GmbH
Paul Birkel	Mitre
Ellen Badgley	Mitre

1.3 Revision history

Date	Release	Editor	Primary clauses modified	Description
2012/07/23	0.1	C. Portele	All	Initial document template and contents
2012/11/04	0.2	C. Portele	7	Text for SWE Common added
2012/12/09	0.3	C. Portele	1, 5, 6	Text for JSON Encoding Rule added
2012/12/20	0.4	C. Portele	All	Updates based on comments/review by Ellen Badgley and Paul Birkel

1.4 Future work

Improvements in this document are desirable

- by testing and analyzing the use of JSON data encoded according to the encoding rule in more depth
- in particular by improving the GeoServices JSON encoding rule, e.g. by using domains and aliases as tested in the demo service
- by developing a JSON schema for GeoJSON and extending the JSON Schema encoding rules with GeoJSON options
- by testing potential extensions of the GeoServices JSON encoding, e.g. for including hyperlinks
- by testing and analyzing in more depth, if UML can be useful to model templates for SWE Common data components

In a future revision of the SWE Common Data Model, the incorrect statement that "all elements are substitutable for gml:AbstractValue (and thus transitively for gml:AbstractObject) so that they can be used directly by GML application schemas" should be removed. In such a revision it could also be considered, if the SWE Common encoding rule (see 7.2) should be included in the standard.

1.5 Forward

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

2 References

The following documents are referenced in this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

JSON Schema, IETF Draft 3, http://tools.ietf.org/html/draft-zyp-json-schema-03

OGC Geography Markup Language, Version 3.2, Open Geospatial Consortium (OGC)

OGC Geography Markup Language, Version 3.3, Open Geospatial Consortium (OGC)

OGC SWE Common Data Model Encoding Standard, Version 2.0, Open Geospatial Consortium (OGC)

GeoServices REST API – Part 1: Core, RFC Version, Open Geospatial Consortium (OGC)

GeoJSON, Version 1.0, http://geojson.org/geojson-spec.html

3 Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard [OGC 06-121r3] and in the normative references shall apply.

4 Conventions

GML	Geography Markup Language
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
OGC	Open Geospatial Consortium
OWS	OGC Web Services
UGAS	UML-to-GML-Application-Schema conversion
UML	Unified Modeling Language
XML	eXtended Markup Language

4.1 UML notation

Diagrams that appear in this standard are presented using the Unified Modeling Language (UML) static structure diagram, as described in ISO/TS 19103.

5 Schema automation in OWS-9 overview

UML is frequently used to specify and maintain conceptual schemas for geographic information. In order to derive implementation schemas from these conceptual schemas in order to use them in the communication between systems or as a basis for implementations in software, rules are needed to help automate these processes. Tools can then be built that implement these rules and support building solutions based on the conceptual schemas.

ShapeChange is such a tool that takes application schemas for geographic information and converts them into a variety of target representations including GML application schemas, XML schemas according to ISO/TS 19139, feature catalogues, code list dictionaries, etc. More information about ShapeChange is available at http://shapechange.net/.

The schema automation activities in the OWS-9 initiative

- have specified and implemented an encoding rule for JSON (see Clause 6).
- have clarified and implemented the SWE Common 2.0 XML Schema encoding rule (see 7.2),
- have analyzed possibilities for generating SWE Common 2.0 record descriptions from schemas in UML (see 7.3).

6 Support for JSON

6.1 Overview

The scope of the activity discussed in this Clause is to explore the generation of JSON Schema – or alternative means – for specifying JSON-based data exchange structures from UML similar to the UGAS approach.

Figure 1 illustrates the conversion process. An organisation, in this example NGA, manages an application schema according to ISO 19109, in this example NGA TDS. This application schema determines the structures in which data is captured and stored as datasets in spatial databases.

ShapeChange is used to derive representations from the application schema. In this case the XML Schema (a GML application schema) and JSON Schema (consistent with the GeoServices JSON feature model) representations are shown. The process has been used for GML application schemas for many years now. The GML application schemas will then typically be used to publish the data via an OGC Web Feature Service.

The capability for JSON Schema representations has been specified and implemented during OWS-9. In the JSON Schema variant that is consistent with GeoServices JSON from the OGC candidate standard GeoServices REST API, the JSON encoded features would typically be published via a GeoServices REST API Feature Service.

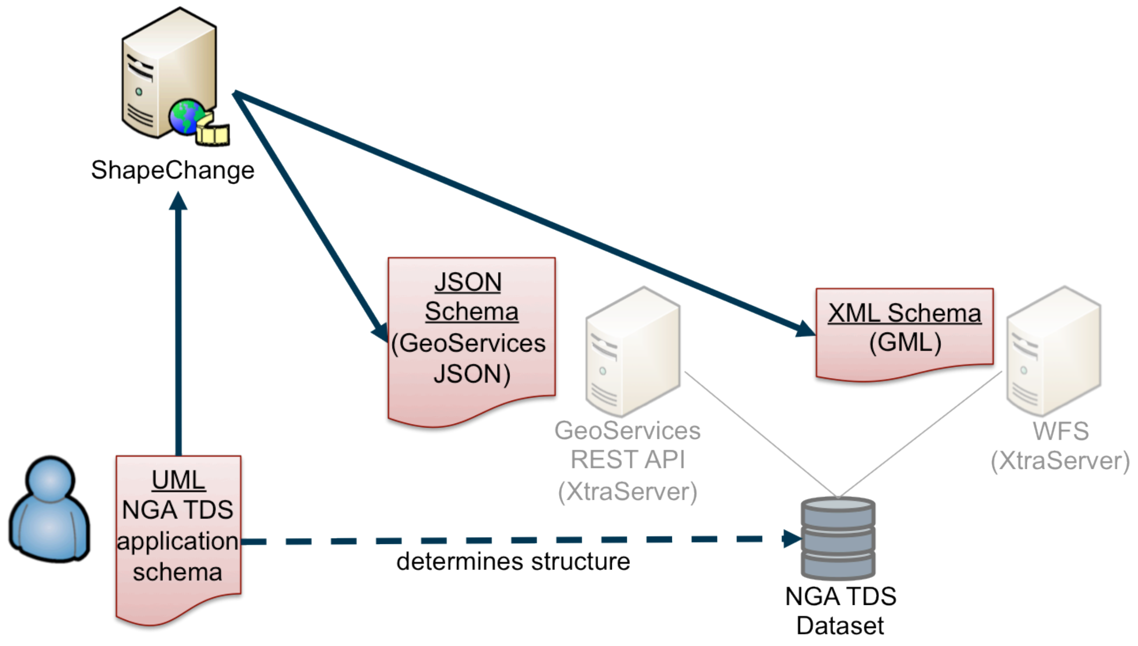

ShapeChange
JSON Schema (GeoServices JSON)
XML Schema (GML)
GeoServices REST API (XtraServer)
WFS (XtraServer)
UML NGA TDS application schema
determines structure
NGA TDS Dataset

Figure 1 – Converting an application schema to implementation schemas

Figure 2 shows the test scenario where NGA TDS data in a database is provided in both representations, JSON via a GeoServices REST API Feature Service and GML via a Web Feature Service. Both services use XtraServer from interactive instruments. The services are accessed with clients from other sources (ArcGIS.com, ArcMap, Gaia). This provides a range of clients with access to the same data via the service interface they support best.

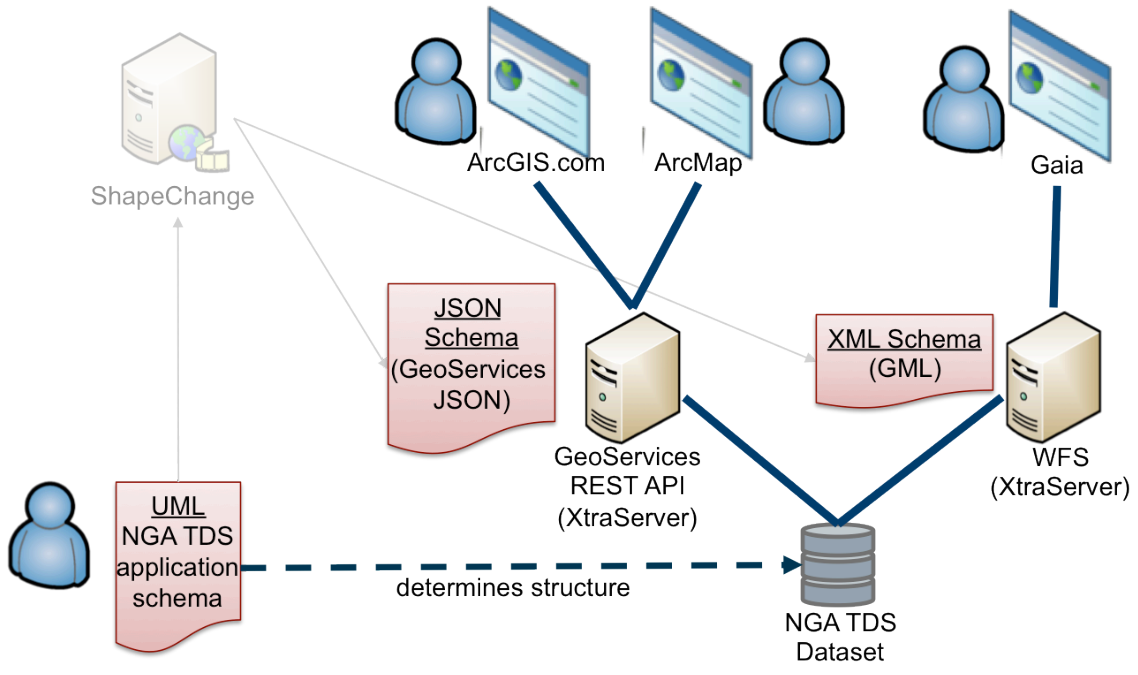


Figure 2 – Accessing the same dataset in JSON and GML representations

This already shows one option for representing the data in JSON. Other options are explored in the next sections.

6.2 Analysis

6.2.1 Overview

This sub-clause analyzes different options, documents the decisions and provides a rationale for the decisions.

6.2.2 The general approach

We have considered three basic options for encoding features in JSON, that will be discussed in more detail below.

1. Use the feature and geometry encoding of the GeoServices JSON encoding specified by the GeoServices REST API. The GeoServices REST API is currently is candidate OGC standard.
2. Use the feature and geometry encoding of the GeoJSON encoding.
3. Derive a JSON encoding from the GML encoding using one of the existing XML-to-JSON-mappings.

As a basis for a decision on the general approach, a simple BuildingGeopoint feature from the Monterey data set used in OWS-9 is used to analyze at the different options. The feature encoded in GML is shown below:

```
<tds:BuildingGeopoint gml:id="StructurePoints8"
xmlns:tds="http://metadata.dod.mil/mdr/ns/GSIP/3.0/tds/3.0"
xmlns:gml="http://www.opengis.net/gml/3.2">
  <tds:geometry>
    <gml:Point gml:id="StructurePoints8.Geom_0"
srsName="http://metadata.ces.mil/mdr/ns/GSIP/crs/WGS84E_2D">
      <gml:pos>36.732821045 -121.633961892</gml:pos>
    </gml:Point>
  </tds:geometry>
  <tds:address>No Information</tds:address>
  <tds:aeroObstacleLightPresent>false</tds:aeroObstacleLightPresent>
  <tds:angleOfOrientation>115.00000000</tds:angleOfOrientation>
  <tds:area>568.99756643</tds:area>
  <tds:conditionOfFacility>fullyFunctional</tds:conditionOfFacility>
  <tds:controllingAuthority>noInformation</tds:controllingAuthority>
  <tds:featureFunction-1>education</tds:featureFunction-1>
  <tds:featureFunction-2>noInformation</tds:featureFunction-2>
  <tds:featureFunction-3>noInformation</tds:featureFunction-3>
  <tds:floorCount>-999999</tds:floorCount>
  <tds:geointAssuranceMetadata.processStep.source.resourceContentOrigin>noInformation
</tds:geointAssuranceMetadata.processStep.source.resourceContentOrigin>
  <tds:geoNameCollection.memberGeoName.fullName>No
Information</tds:geoNameCollection.memberGeoName.fullName>
  <tds:geoNameCollection.memberGeoName.nameIdentifier>-
999999</tds:geoNameCollection.memberGeoName.nameIdentifier>
  <tds:heightAboveSurfaceLevel>33.00000000</tds:heightAboveSurfaceLevel>
  <tds:highestElevation>-999999.00000000</tds:highestElevation>
  <tds:length>29.60813040</tds:length>
  <tds:manufacturingInfo.byProduct-1>noInformation</tds:manufacturingInfo.byProduct-1>
  <tds:manufacturingInfo.byProduct-2>noInformation</tds:manufacturingInfo.byProduct-2>
  <tds:manufacturingInfo.byProduct-3>noInformation</tds:manufacturingInfo.byProduct-3>
  <tds:manufacturingInfo.product-1>noInformation</tds:manufacturingInfo.product-1>
  <tds:manufacturingInfo.product-2>noInformation</tds:manufacturingInfo.product-2>
  <tds:manufacturingInfo.product-3>noInformation</tds:manufacturingInfo.product-3>
  <tds:manufacturingInfo.rawMaterial-1>noInformation</tds:manufacturingInfo.rawMaterial-
1>
  <tds:manufacturingInfo.rawMaterial-2>noInformation</tds:manufacturingInfo.rawMaterial-
2>
  <tds:manufacturingInfo.rawMaterial-3>noInformation</tds:manufacturingInfo.rawMaterial-
3>
  <tds:navigationLandmark>false</tds:navigationLandmark>
  <tds:note.memorandum>No Information</tds:note.memorandum>
  <tds:religiousInfo.religiousDesignation>noInformation
</tds:religiousInfo.religiousDesignation>
  <tds:religiousInfo.religiousFacilityType>noInformation
</tds:religiousInfo.religiousFacilityType>
  <tds:restriction.securityAttributesGroup_resClassification>U
</tds:restriction.securityAttributesGroup_resClassification>
  <tds:restriction.securityAttributesGroup_resNonIntelComMarkings>No
Information</tds:restriction.securityAttributesGroup_resNonIntelComMarkings>
  <tds:restriction.securityAttributesGroup_resOwnerProducer>No
Information</tds:restriction.securityAttributesGroup_resOwnerProducer>
  <tds:roofShape-1>noInformation</tds:roofShape-1>
  <tds:roofShape-2>noInformation</tds:roofShape-2>
  <tds:roofShape-3>noInformation</tds:roofShape-3>
  <tds:specifiedEnumerants>No Information</tds:specifiedEnumerants>
  <tds:uniqueEntityIdentifier>No Information</tds:uniqueEntityIdentifier>
  <tds:verticalConstMaterial-1>noInformation</tds:verticalConstMaterial-1>
  <tds:verticalConstMaterial-2>noInformation</tds:verticalConstMaterial-2>
  <tds:verticalConstMaterial-3>noInformation</tds:verticalConstMaterial-3>
  <tds:verticalObstIdentifier>No Information</tds:verticalObstIdentifier>
  <tds:width>19.29426795</tds:width>
  <tds:wirelessTelecomInfo.wirelessTelecomType>noInformation
</tds:wirelessTelecomInfo.wirelessTelecomType>
```

```
</tds:BuildingGeopoint>
```

A straightforward mapping to GeoServices JSON of this feature would be:

```
{
   "geometry":{
      "x":-121.633961892,
      "y":36.732821045,
      "spatialReference":{
         "wkid":4326
      }
   },
   "attributes":{
      "objectId":8,
      "address":"No Information",
      "aeroObstacleLightPresent":"false",
      "angleOfOrientation":115.00000000,
      "area":568.99756643,
      "conditionOfFacility":"fullyFunctional",
      "controllingAuthority":"noInformation",
      "featureFunction-1":"education",
      "featureFunction-2":"noInformation",
      "featureFunction-3":"noInformation",
      "floorCount":-999999,
      "geointAssuranceMetadata.processStep.source.resourceContentOrigin":"noInformation",
      "geoNameCollection.memberGeoName.fullName":"No Information",
      "geoNameCollection.memberGeoName.nameIdentifier":-999999,
      "heightAboveSurfaceLevel":33.00000000,
      "highestElevation":-999999.00000000,
      "length":29.60813040,
      "manufacturingInfo.byProduct-1":"noInformation",
      "manufacturingInfo.byProduct-2":"noInformation",
      "manufacturingInfo.byProduct-3":"noInformation",
      "manufacturingInfo.product-1":"noInformation",
      "manufacturingInfo.product-2":"noInformation",
      "manufacturingInfo.product-3":"noInformation",
      "manufacturingInfo.rawMaterial-1":"noInformation",
      "manufacturingInfo.rawMaterial-2":"noInformation",
      "manufacturingInfo.rawMaterial-3":"noInformation",
      "navigationLandmark":"false",
      "note.memorandum":"No Information",
      "religiousInfo.religiousDesignation":"noInformation",
      "religiousInfo.religiousFacilityType":"noInformation",
      "restriction.securityAttributesGroup_resClassification":"U",
      "restriction.securityAttributesGroup_resNonIntelComMarkings":"No Information",
      "restriction.securityAttributesGroup_resOwnerProducer":"No Information",
      "roofShape-1":"noInformation",
      "roofShape-2":"noInformation",
      "roofShape-3":"noInformation",
      "specifiedEnumerants":"No Information",
      "uniqueEntityIdentifier":"No Information",
      "verticalConstMaterial-1":"noInformation",
      "verticalConstMaterial-2":"noInformation",
      "verticalConstMaterial-3":"noInformation",
      "verticalObstIdentifier":"No Information",
      "width":19.29426795,
      "wirelessTelecomInfo.wirelessTelecomType":"noInformation"
   }
}
```

In this mapping, the following conversion of primitive types from ISO/TS 19103 to JSON Schema types is used:

- CharacterString, Character: string
- Integer: integer

- Real, Decimal, Number: number
- Boolean: boolean
- URI: string with format "uri"
- DateTime: string with format "date-time" or integer with format "utc-millisec"
- Date: string with format "date" or integer with format "utc-millisec"
- Time: string with format "time"

The benefit of using this option is that a GeoServices REST API Feature Service would provide the feature using this representation and all the existing clients that are able to handle features encoded in GeoServices JSON would be immediately able to process and display such features.

One aspect to note here is that this representation does not include information that states that this is a BuildingGeopoint feature; as in the GeoServices REST API this information would be implicit. For example, the feature is a sub-ordinate resource to the feature type/layer "BuildingGeopoint". This is discussed in more detail below.

Note that the coordinate reference system information could also be specified using a WKT representation, if this is desired.

Example (WKT):

```
   "geometry":{
      "x":-121.633961892,
      "y":36.732821045,
      "spatialReference":{
         "wkt" :
"GEOGCS[\"GCS_WGS_1984\",DATUM[\"D_WGS_1984\",SPHEROID[\"WGS_1984\",6378137,298.257223563
]],PRIMEM[\"Greenwich\",0],UNIT[\"Degree\",0.017453292519943295]]"
      }
   }
```

Encoding the same feature GeoJSON would result in a very similar JSON:

```
{
   "type":"Feature",
   "geometry":{
      "type":"Point",
      "coordinates":[
         -121.633961892,
         36.732821045
      ]
   },
   "properties":{
      "id":8,
      "address":"No Information",
      "aeroObstacleLightPresent":"false",
      "angleOfOrientation":115.00000000,
      "area":568.99756643,
      "conditionOfFacility":"fullyFunctional",
      "controllingAuthority":"noInformation",
      "featureFunction-1":"education",
      "featureFunction-2":"noInformation",
      "featureFunction-3":"noInformation",
      "floorCount":-999999,
      "geointAssuranceMetadata.processStep.source.resourceContentOrigin":"noInformation",
      "geoNameCollection.memberGeoName.fullName":"No Information",
      "geoNameCollection.memberGeoName.nameIdentifier":-999999,
      "heightAboveSurfaceLevel":33.00000000,
      "highestElevation":-999999.00000000,
      "length":29.60813040,
      "manufacturingInfo.byProduct-1":"noInformation",
      "manufacturingInfo.byProduct-2":"noInformation",
      "manufacturingInfo.byProduct-3":"noInformation",
      "manufacturingInfo.product-1":"noInformation",
      "manufacturingInfo.product-2":"noInformation",
      "manufacturingInfo.product-3":"noInformation",
      "manufacturingInfo.rawMaterial-1":"noInformation",
      "manufacturingInfo.rawMaterial-2":"noInformation",
      "manufacturingInfo.rawMaterial-3":"noInformation",
      "navigationLandmark":"false",
      "note.memorandum":"No Information",
      "religiousInfo.religiousDesignation":"noInformation",
      "religiousInfo.religiousFacilityType":"noInformation",
      "restriction.securityAttributesGroup_resClassification":"U",
      "restriction.securityAttributesGroup_resNonIntelComMarkings":"No Information",
      "restriction.securityAttributesGroup_resOwnerProducer":"No Information",
      "roofShape-1":"noInformation",
      "roofShape-2":"noInformation",
      "roofShape-3":"noInformation",
      "specifiedEnumerants":"No Information",
      "uniqueEntityIdentifier":"No Information",
      "verticalConstMaterial-1":"noInformation",
      "verticalConstMaterial-2":"noInformation",
      "verticalConstMaterial-3":"noInformation",
      "verticalObstIdentifier":"No Information",
      "width":19.29426795,
      "wirelessTelecomInfo.wirelessTelecomType":"noInformation"
   }
}
```

Like in the GeoServices JSON, the GeoJSON representation does not include the name of the feature type "BuildingGeopoint".

A third option for a JSON representation could be to ignore the existing work on JSON feature and geometry representations and generically map the GML to JSON. There are several attempts at general XML-to-JSON mappings. Here is an example of what an instance could look like:

```
{
   "BuildingGeopoint":{
      "id":"StructurePoints8",
      "geometry":{
         "Point":{
            "id":"StructurePoints8.Geom_0",
            "srsName":"http://metadata.ces.mil/mdr/ns/GSIP/crs/WGS84E_2D",
            "pos":[
               36.732821045,
               -121.633961892
            ]
         }
      },
      "address":"No Information",
      "aeroObstacleLightPresent":"false",
      "angleOfOrientation":115.00000000,
      "area":568.99756643,
      "conditionOfFacility":"fullyFunctional",
      "controllingAuthority":"noInformation",
      "featureFunction-1":"education",
      "featureFunction-2":"noInformation",
      "featureFunction-3":"noInformation",
      "floorCount":-999999,
      "geointAssuranceMetadata.processStep.source.resourceContentOrigin":"noInformation",
      "geoNameCollection.memberGeoName.fullName":"No Information",
      "geoNameCollection.memberGeoName.nameIdentifier":-999999,
      "heightAboveSurfaceLevel":33.00000000,
      "highestElevation":-999999.00000000,
      "length":29.60813040,
      "manufacturingInfo.byProduct-1":"noInformation",
      "manufacturingInfo.byProduct-2":"noInformation",
      "manufacturingInfo.byProduct-3":"noInformation",
      "manufacturingInfo.product-1":"noInformation",
      "manufacturingInfo.product-2":"noInformation",
      "manufacturingInfo.product-3":"noInformation",
      "manufacturingInfo.rawMaterial-1":"noInformation",
      "manufacturingInfo.rawMaterial-2":"noInformation",
      "manufacturingInfo.rawMaterial-3":"noInformation",
      "navigationLandmark":"false",
      "note.memorandum":"No Information",
      "religiousInfo.religiousDesignation":"noInformation",
      "religiousInfo.religiousFacilityType":"noInformation",
      "restriction.securityAttributesGroup_resClassification":"U",
      "restriction.securityAttributesGroup_resNonIntelComMarkings":"No Information",
      "restriction.securityAttributesGroup_resOwnerProducer":"No Information",
      "roofShape-1":"noInformation",
      "roofShape-2":"noInformation",
      "roofShape-3":"noInformation",
      "specifiedEnumerants":"No Information",
      "uniqueEntityIdentifier":"No Information",
      "verticalConstMaterial-1":"noInformation",
      "verticalConstMaterial-2":"noInformation",
      "verticalConstMaterial-3":"noInformation",
      "verticalObstIdentifier":"No Information",
      "width":19.29426795,
      "wirelessTelecomInfo.wirelessTelecomType":"noInformation"
   }
}
```

While the last option is the most general approach, it ignores the existence of the existing JSON representations for features and geometries in the geospatial community. It seems preferable to build on the existing and widely supported representations, GeoServices JSON and GeoJSON. This is the direction that we have followed in the OWS-9 work.

Both GeoJSON and GeoServices JSON have been developed in parallel. They are both used heavily in practice and very likely will both continue to be used in the future and supported by multiple products. As a consequence, both JSON encodings for features should be supported. As the difference between the feature/geometry encoding of GeoJSON and GeoServices JSON is quite small and the transformation not difficult[1], the encoding rules are similar.

One amendment to the existing GeoServices JSON and GeoJSON feature encoding is that the information about the feature type should be part of the JSON representation. Since this is not a regular feature property, an additional top-level JSON property "entityType" with a string value, e.g. "BuildingGeopoint", will be added. The use of "entityType" instead of "featureType" is deliberate as it supports also types that are objects with identity, but not features.

[1] some JavaScript examples are on GitHub, e.g. https://github.com/odoe/esritogeojson or https://github.com/Esri/geojson-utils/tree/master/src

An id property will be added to each object. To keep with the naming conventions, "objectId" will be used in the GeoServices encoding rule and "id" will be used in the GeoJSON encoding rule. The type will be an integer.

6.2.3 More complex instances

The example used above is simple and restricted in several ways:

- properties have a maximum multiplicity of 1
- properties have simple values only
- features have no relationships with other features
- there is only one geometry property per feature
- there is only limited support for ISO 19100 types used in application schemas

We will look at these issues one-by-one.

In most cases we will support two different encoding rule options:

- Extended: A representation that maps concepts to JSON and extends the base JSON schemas, but makes full use of the JSON capabilities.
- Simple: A representation that maps concepts to a JSON representation consistent with the GeoServices REST API feature representation and the GeoServices REST API Feature Service; this simplified representation supports a larger range of clients due to the restrictions on the complexity of the JSON encoding.

Both approaches will be supported as an option in the encoding rule. This has been implemented in ShapeChange as different encoding rules of the JSON Schema target.

Unless noted, there is no difference in the GeoServices JSON and GeoJSON instances.

6.2.3.1 Properties with a maximum multiplicity greater than 1

Extended: Value of the JSON property is an array and each item in the array represents one value in the value collection.

Example:

```
"featureFunction":["education","medical"]
```

Simple: Like in the TDS example, multiple properties are created, i.e. "property-1", "property-2" and "property-3".

Example:

```
"featureFunction-1":"education",
"featureFunction-2":"medical"
```

The number of properties will be a global option of the conversion process, the default will be 3.

6.2.3.2 Properties with values that are data types

Examples are properties where the value type is a data type (including unions) defined in an application schema.

Extended: Value of the JSON property is an object and each property of the data type is represented as a property of that object.

Example:

```
"geoNameCollection":{
   "memberGeoName":{
      "fullName":"some name",
      "nameIdentifier":null
   }
}
```

Simple: Like in the TDS example, the type hierarchy is flattened. This requires that also the flattening approach is used for multiplicities > 1.

Example:

```
 "geoNameCollection.memberGeoName.fullName":"some name",
 "geoNameCollection.memberGeoName.nameIdentifier":null,
```

Note that this flattening approach can result in quite long lists of properties, if any of the properties contains properties with a multiplicity > 1. So this option has to be used with care with complex models.

6.2.3.3 Properties that allow nil values and metadata on those nil values

Extended: Basically a nil value would be a null in JSON. However, if the equivalent of a nilReason value should be encoded in JSON, too, then we need something different, i.e. a special nil-metadata-object.

Example (nil value without a reason):

```
"geoNameCollection":{
   "memberGeoName":{
      "fullName":null,
      "nameIdentifier":null
   }
}
```

Example (nil value with a reason):

```
"geoNameCollection":{
   "memberGeoName":{
      "fullName":{
         "nilReason":"No information"
      },
      "nameIdentifier":{
         "nilReason":"No information"
```

```
        }
     }
  }
```

This is not entirely satisfying as - unlike in the XML Schema case - the information is lost that it is a nil value whenever reason metadata about the nil value is provided.

We will therefore use the following approach, where additional properties are added as shown in the next example.

Example (nil value with a reason, alternative approach):

```
"geoNameCollection":{
   "memberGeoName":{
      "fullName":null,
      "fullName_nilReason":"No information",
      "nameIdentifier":null,
      "nameIdentifier_nilReason":"No information"
   }
}
```

Simple: Like in the TDS example, nil and nilReason values are mapped to special values (that are documented outside of the schema).

Example:

```
"geoNameCollection.memberGeoName.fullName":"No Information",
"geoNameCollection.memberGeoName.nameIdentifier":-999999
```

6.2.3.4 Properties with values that are enumerants

The value is a string with a constraint that the valid values are from a fixed list.

Example:

```
"controllingAuthority":"military",
```

6.2.3.5 Properties with values that are code list values

Extended: Value of the JSON property is a URI identifying the code list value. It is assumed that the URI can be dereferenced and provides a representation of the code list value (GML, SKOS, JSON).

Example:

```
"conditionOfFacility":
"http://metadata.ces.mil/mdr/ns/GSIP/conditionOfFacility/fullyFunctional",
```

Simple: Like in the TDS example, the value would be a code. The link with the underlying code list would be specified outside of the schema.

Example:

```
"conditionOfFacility":"fullyFunctional"
```

6.2.3.6 Properties with values that are of types Date, Time or DateTime

Extended: The value of the JSON property is a string with a format constraint. The format constraints specified by JSON Schema match the data types specified in ISO/TS 19103 (DateTime: date-time, Date: date, Time: time). The formats are defined as (see http://tools.ietf.org/html/draft-zyp-json-schema-03#section-5.23):

- date-time: YYYY-MM-DDThh:mm:ssZ in UTC time
- date: YYYY-MM-DD
- time: hh:mm:ss

I.e., the date and time representations are somewhat more restricted than the basic types from ISO/TS 19103 which support also values that do not include a month or day, minute or second.

Example:

```
"lastUpdate":"2012-06-05T10:26:34Z"
```

Simple: The GeoServices REST API JSON encodes timestamps as an integer representing the milliseconds between the specified time and midnight, 00:00 of January 1, 1970 UTC, i.e. they follow the "utc-millisec" pattern.

Example:

```
"lastUpdate":1338891994000
```

As a consequence, there is no option to appropriately represent Time or Date values and these would be represented as strings, too.

In the "simple" GeoJSON encoding rule the string representation is used, too.

6.2.3.7 Properties with values that are of type Measure or any of its subtypes

Extended: The value of the JSON property is an object with two properties, "value" and "unit". The unit values follow the pattern supported in the uom XML attributes in GML, i.e. "unit" may either be a conventional unit of measure symbol or the URI of the definition of a unit of measure.

Example (conventional unit):

```
"width":{"value":19.29426795,"unit":"m"}
```

Example (URI):

```
"width":{"value":19.29426795,"unit":"http://www.opengis.net/def/uom/epsg/0/9001"}
```

Simple: There are two approaches which could be used. One is like in the TDS example, where the unit is fixed to some unit.

Example (fixed unit):

```
"width":19.29426795
```

The other option would be to add an additional property for the unit:

Example (variable unit):

```
"width":19.29426795,
"width_unit":"m"
```

The second option is more general, but avoiding unit conversions in clients by fixing the unit (and requiring that the conversion, if necessary, is done on the server side) may also be a benefit. We will therefore follow the fixed unit approach.

6.2.3.8 String properties with regular expression patterns

In this case, the question is rather how this will be represented in the UML model. In OWS-7/8 regular expression patterns have been expressed in an OCL constraint using the syntax *property.matches(regex)*.

Originally, the assumption was that we would ignore such constraints in the JSON encoding rule as there is no JSON equivalent for Schematron. However, JSON Schema supports specifying patterns on string properties using the regular expression pattern in the ECMA 262/Perl 5 format which includes – according to current knowledge – the patterns supported by XPath 2.0.

6.2.3.9 Properties with values that are features/objects

Extended: Value of the JSON property is a URI identifying the object. It is assumed that the URI can be dereferenced and provides a representation of the object.

Example:

```
"owner":"http://example.com/Person/123"
```

Simple: In GeoJSON a URI will be used, too.

In the GeoServices REST API, relationships to other objects are documented separate from the schema using joins. The objects will contain properties that represent foreign and primary keys.

Example:

```
"owner":123
```

Here, 123 is the foreign key that references a Person object with the objectId 123.

For cases, when the URI scheme of the server where the data will be deployed is known at schema creation time, we can also encode the relationship in the schema. An example:

Assume we have feature types BuildingGeopoint and Person and they would be available at *http://example.com/TDS/FeatureServer/1/{objectId}?f=json* and *http://example.com/TDS/FeatureServer/2/{objectId}?f=json* respectively. Then the link to the owner could be expressed using the link concept in JSON schema shown below. Of course, that requires that the URI scheme is known at the time the schema is created. If this is acceptable, then we could perhaps think of a way in which to encode this information also in the simple encoding rule.

```
{
   "$schema":"http://json-schema.org/draft-03/schema#",
   "id":"http://example.com/schema/tds/BuildingGeopoint.json",
   "title":"Building feature with point geometry",
   "type":"object",
   "properties":{
      "entityType":{
         "type":"string",
         "title":"type of the feature type",
         "enum":[
            "BuildingGeopoint"
         ]
      },
      "geometry":{
         "title":"feature geometry",
         "$ref":"http://schemas.opengis.net/gsr/1.0/point.json"
      },
      "attributes":{
         "title":"feature/object attributes",
         "type":"object",
         "properties":{
            "objectId":{
               "type":"integer",
               "description":"Object identifier",
               "required":true
            },
            "owner":{
               "description":"Owner of the buidling",
               "type":"integer"
            }
         }
      },
      "links":[
         {
            "rel":"related",
            "href":"http://example.com/TDS/FeatureServer/2/{#/attributes/owner}?f=json"
         }
      ]
   }
}
```

6.2.3.10 Multiple geometry properties per feature

The restriction to one geometry is a limitation of both GeoServices JSON and GeoJSON and is very much part of the foundation of these feature representations as expressed by the specific "geometry" property, separate from the rest of the feature properties. Therefore, the two obvious options to allow for additional geometry properties (additional geometry properties as part of the "attribute"/"properties" object or additional geometry properties like "geometry-2" etc.) do seem to go against the basic concepts of these JSON encodings.

Therefore, the encoding rule includes a statement in the application schema requirements that each feature should not have more than one geometry property. ShapeChange reports a warning, if a feature type has multiple geometry properties.

This is not considered a limitation for the mobile web mapping use case that is a driver for the JSON work in OWS-9 ("mobile" in the sense that the application is mobile and can be run on a wide range of devices, not limited to smartphones and other mobile devices).

6.2.3.11 Types without geometry properties

Instances of these types will simply lack the "geometry" property.

6.2.3.12 Support for ISO 19100 types used in application schemas

As the encoding rule does not aim at defining JSON schemas for base types specified in the ISO 19100 series, the encoding rule will only support types from the ISO 19100 standards for which a JSON encoding exists as part of GeoServices JSON and GeoJSON, or where a conversion has been defined above.

The encoding rule includes a list of all supported ISO 19100 types in the application schema requirements. For properties with other types a warning is issued and the type is mapped to object (in the extended encoding rule) or string (in the simple encoding rule).

"Other types" include the types, for example, from ISO 19108 (e.g. TM_Instant), ISO 19115 (e.g. LI_Lineage, CI_ResponsibleParty, DQ_Result) and ISO 19123 (e.g. CV_RectifiedGrid) as well as those from ISO 19107 that go beyond the spatial geometries supported by ISO 19125 (e.g. GM_Solid, TP_Edge).

6.3 Implementation and test application schema

6.3.1 Implementation in ShapeChange

Support for the simple and extended JSON Schema encoding rules has been added to ShapeChange[2]. Details can be found at http://shapechange.net/targets/json/.

NOTE The webpages are not yet complete. Content from this chapter will be transferred once the Engineering Report is final.

[2] OCL constraints on classifiers are currently not supported by the implementation.

6.3.2 The test application schema

A test application schema has been set up in Enterprise Architect that includes the different aspects discussed in the previous sub-clause.

The application schema is part of the ShapeChange distribution and unit tests. In the unit test, the schema documents are compared against the JSON Schema documents included in the next sub-clauses.

NOTE This test application schema uses the stereotype <<voidable>> on properties to indicate that the property value may be nil including optional metadata for the nil value (see 6.2.3.3). ShapeChange supports this as a shorthand notation, the explicit NilUnion type also included in the application schema models this more explicitly.

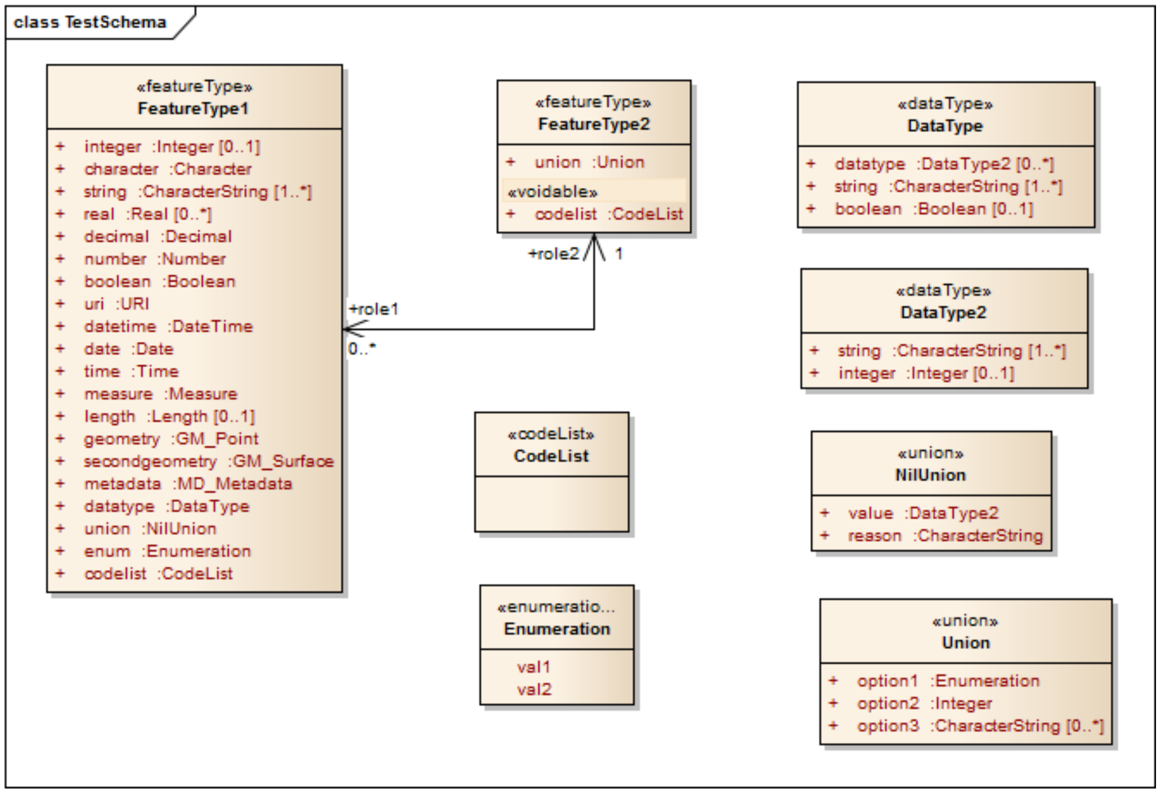


Figure 3 - Test application schema

6.3.3 Simple GeoServices JSON encoding rule

The conversion results in the following schemas.

<u>FeatureType1:</u>

```
{
    "$schema":"http://json-schema.org/draft-03/schema#",
    "id":"http://portele.de/ows9/test/FeatureType1.json",
    "title":"FeatureType1",
    "description":"This is a feature type.",
    "type":"object",
    "properties":{
        "entityType":{
            "title":"feature/object type",
            "type":"string",
            "default":"FeatureType1"
        },
        "geometry":{
            "$ref":"http://schemas.opengis.net/gsr/1.0/point.json"
        },
        "attributes":{
            "title":"feature/object attributes",
            "type":"object",
            "properties":{
                "integer":{
                    "title":"integer",
                    "description":"This is an integer.",
                    "type":"integer"
                },
                "character":{
                    "title":"character",
                    "type":"string",
                    "required":true
                },
                "string-1":{
                    "title":"string",
                    "description":"This is a string.",
                    "type":"string",
                    "required":true
                },
                "string-2":{
                    "title":"string",
                    "description":"This is a string.",
                    "type":"string"
                },
                "string-3":{
                    "title":"string",
                    "description":"This is a string.",
                    "type":"string"
                },
                "real-1":{
                    "title":"real",
                    "type":"number"
                },
                "real-2":{
                    "title":"real",
                    "type":"number"
                },
                "real-3":{
                    "title":"real",
                    "type":"number"
                },
                "decimal":{
                    "title":"decimal",
                    "type":"number",
                    "required":true
                },
                "number":{
                    "title":"number",
                    "type":"number",
                    "required":true
                },
                "boolean":{
                    "title":"boolean",
                    "type":"boolean",
                    "required":true
                },
                "uri":{
                    "title":"uri",
                    "type":"string",
                    "format":"uri",
                    "required":true
                },
                "datetime":{
                    "title":"datetime",
                    "type":"integer",
                    "format":"utc-millisec",
                    "required":true
                },
                "date":{
                    "title":"date",
                    "type":"string",
                    "format":"time",
                    "required":true
                },
                "time":{
                    "title":"time",
                    "type":"string",
                    "format":"time",
                    "required":true
                },
                "measure":{
                    "title":"measure",
                    "type":"number",
                    "required":true
                },
                "length":{
                    "title":"length",
                    "type":"number"
                },
                "metadata":{
                    "title":"metadata",
                    "type":"string",
                    "required":true
                },
                "datatype.datatype-1.string-1":{
                    "title":"string",
                    "type":"string"
                },
                "datatype.datatype-1.string-2":{
                    "title":"string",
                    "type":"string"
                },
                "datatype.datatype-1.string-3":{
                    "title":"string",
                    "type":"string"
                },
                "datatype.datatype-1.integer":{
                    "title":"integer",
                    "type":"integer"
                },
                "datatype.datatype-2.string-1":{
                    "title":"string",
                    "type":"string"
                },
                "datatype.datatype-2.string-2":{
                    "title":"string",
                    "type":"string"
                },
                "datatype.datatype-2.string-3":{
                    "title":"string",
                    "type":"string"
                },
                "datatype.datatype-2.integer":{
                    "title":"integer",
                    "type":"integer"
                },
                "datatype.datatype-3.string-1":{
                    "title":"string",
                    "type":"string"
                },
                "datatype.datatype-3.string-2":{
                    "title":"string",
                    "type":"string"
                },
                "datatype.datatype-3.string-3":{
                    "title":"string",
                    "type":"string"
                },
                "datatype.datatype-3.integer":{
                    "title":"integer",
                    "type":"integer"
                },
                "datatype.string-1":{
                    "title":"string",
                    "type":"string",
                    "required":true
                },
                "datatype.string-2":{
                    "title":"string",
                    "type":"string"
                },
                "datatype.string-3":{
                    "title":"string",
                    "type":"string"
                },
                "datatype.boolean":{
                    "title":"boolean",
                    "type":"boolean"
                },
                "union.value.string-1":{
                    "title":"string",
                    "type":"string"
                },
                "union.value.string-2":{
                    "title":"string",
                    "type":"string"
                },
                "union.value.string-3":{
                    "title":"string",
                    "type":"string"
                },
                "union.value.integer":{
                    "title":"integer",
                    "type":"integer"
                },
                "union.reason":{
                    "title":"reason",
                    "type":"string"
                },
                "enum":{
                    "title":"enum",
                    "type":"string",
                    "enum":["val1","val2"],
                    "required":true
                },
                "codelist":{
                    "title":"codelist",
                    "type":"string",
                    "required":true
                },
                "role2":{
                    "title":"role2",
                    "type":"integer",
                    "required":true
                }
```

```
            }
        }
    },
    "links":[
        {
            "rel":"related",
            "href":"http://example.com/TDS/FeatureServer/2/{#/attributes/owner}?f=json"
        }
    ]
}
```

FeatureType2:

```
{
    "$schema":"http://json-schema.org/draft-03/schema#",
    "id":"http://portele.de/ows9/test/FeatureType2.json",
    "title":"FeatureType2",
    "type":"object",
    "properties":{
        "entityType":{
            "title":"feature/object type",
            "type":"string",
            "default":"FeatureType2"
        },
        "attributes":{
            "title":"feature/object attributes",
            "type":"object",
            "properties":{
                "codelist":{
                    "title":"codelist",
                    "type":"string",
                    "required":true
                },
                "union.option1":{
                    "title":"option1",
                    "type":"string",
                    "enum":["val1","val2"]
                },
                "union.option2":{
                    "title":"option2",
                    "type":"integer"
                },
                "union.option3-1":{
                    "title":"option3",
                    "type":"string"
                },
                "union.option3-2":{
                    "title":"option3",
                    "type":"string"
                },
                "union.option3-3":{
                    "title":"option3",
                    "type":"string"
                },
                "role1-1":{
                    "title":"role1",
                    "type":"integer"
                },
                "role1-2":{
                    "title":"role1",
                    "type":"integer"
                },
                "role1-3":{
                    "title":"role1",
                    "type":"integer"
                }
            }
        }
    },
    "links":[
        {
            "rel":"related",
            "href":"http://example.com/TDS/FeatureServer/1/{#/attributes/owner}?f=json"
        }
    ]
}
```

The ShapeChange configuration file used in the conversion is:

```
<ShapeChangeConfiguration xmlns:xi="http://www.w3.org/2001/XInclude"
xmlns="http://www.interactive-instruments.de/ShapeChange/Configuration/1.1"
xmlns:sc="http://www.interactive-instruments.de/ShapeChange/Configuration/1.1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.interactive-instruments.de/ShapeChange/Configuration/1.1
http://shapechange.net/resources/schema/ShapeChangeConfiguration.xsd">
    <input>
        <parameter name="inputModelType" value="EA7"/>
        <parameter name="inputFile" value="src/test/resources/test.eap"/>
        <parameter name="appSchemaName" value="Test Schema"/>
        <parameter name="publicOnly" value="true"/>
        <parameter name="checkingConstraints" value="disabled"/>
        <parameter name="sortedSchemaOutput" value="true"/>
        <xi:include href="http://shapechange.net/resources/config/StandardAliases.xml"/>
    </input>
    <log>
        <parameter name="reportLevel" value="INFO"/>
        <parameter name="logFile" value="testResults/ea/log_JsonGsr.xml"/>
    </log>
    <targets>
        <Target class="de.interactive_instruments.ShapeChange.Target.JSON.JsonSchema"
mode="enabled">
            <targetParameter name="outputDirectory"
value="testResults/ea/json/geoservices"/>
            <targetParameter name="defaultEncodingRule" value="geoservices"/>
            <xi:include href="
http://shapechange.net/resources/config/StandardJsonMapEntries.xml"/>
        </Target>
    </targets>
</ShapeChangeConfiguration>
```

6.3.4 Extended GeoServices JSON encoding rule

The conversion results in the following schemas.

FeatureType1:

```
{
    "$schema":"http://json-schema.org/draft-03/schema#",
    "id":"http://portele.de/ows9/test/FeatureType1.json",
    "title":"FeatureType1",
    "description":"This is a feature type.",
    "type":"object",
    "properties":{
        "entityType":{
            "title":"feature/object type",
            "type":"string",
            "default":"FeatureType1"
        },
        "geometry":{
            "$ref":"http://schemas.opengis.net/gsr/1.0/point.json"
        },
        "attributes":{
            "title":"feature/object attributes",
"type":"object",
"properties":{
    "integer":{
        "title":"integer",
        "description":"This is an integer.",
        "type":"integer"
    },
    "character":{
        "title":"character",
        "type":"string",
        "required":true
    },
    "string":{
        "title":"string",
        "description":"This is a string.",
        "type":"array",
        "items":{
            "type":"string",
            "minItems":"1"
        }
    },
    "real":{
        "title":"real",
        "type":"array",
        "items":{
            "type":"number"
        }
    },
    "decimal":{
        "title":"decimal",
        "type":"number",
        "required":true
    },
    "number":{
        "title":"number",
        "type":"number",
        "required":true
    },
    "boolean":{
        "title":"boolean",
        "type":"boolean",
        "required":true
    },
    "uri":{
        "title":"uri",
        "type":"string",
        "format":"uri",
        "required":true
    },
    "datetime":{
        "title":"datetime",
        "type":"string",
        "format":"date-time",
        "required":true
    },
    "date":{
        "title":"date",
        "type":"string",
        "format":"time",
        "required":true
    },
    "time":{
        "title":"time",
        "type":"string",
        "format":"time",
        "required":true
    },
    "measure":{
        "title":"measure",
        "$ref":"http://shapechange.net/tmp/ows9/json/measure.json"
```

```
                },
                "length":{
                    "title":"length",
                    "$ref":"http://shapechange.net/tmp/ows9/json/measure.json"
                },
                "metadata":{
                    "title":"metadata",
                    "type":"any",
                    "required":true
                },
                "datatype":{
                    "title":"datatype",
                    "description":"This is a data type.",
                    "$ref":"http://portele.de/ows9/test/DataType.json"
                },
                "union":{
                    "title":"union",
                    "$ref":"http://portele.de/ows9/test/NilUnion.json"
                },
                "enum":{
                    "title":"enum",
                    "type":"string",
                    "enum":["val1","val2"],
                    "required":true
                },
                "codelist":{
                    "title":"codelist",
                    "type":"string",
                    "format":"uri",
                    "required":true
                },
                "role2":{
                    "title":"role2",
                    "type":"string",
                    "format":"uri",
                    "required":true
                }
            }
        }
    }
}
```

where http://shapechange.net/tmp/ows9/json/measure.json is

```
{
    "$schema":"http://json-schema.org/draft-03/schema#",
    "id":"http://shapechange.net/tmp/ows9/json/measure.json",
    "title":"measure",
    "type":"object",
    "properties":{
        "value":{
            "title":"measure value",
            "type":"number",
            "required":true
        },
        "uom":{
            "title":"unit of measure",
            "description":"the value may be a conventional unit of measure symbol or the 
URI of the definition of a unit of measure"
            "type":"string",
            "required":true
        }
    }
}
```

FeatureType2:

```
{
    "$schema":"http://json-schema.org/draft-03/schema#",
    "id":"http://portele.de/ows9/test/FeatureType2.json",
    "title":"FeatureType2",
    "type":"object",
    "properties":{
        "entityType":{
            "title":"feature/object type",
            "type":"string",
            "default":"FeatureType2"
        },
        "attributes":{
            "title":"feature/object attributes",
            "type":"object",
            "properties":{
                "codelist":{
                    "title":"codelist",
                    "type":["string","null"],
                    "format":"uri",
                    "required":true
                },
                "codelist_nullReason":{
                    "title":"Reason for null value in property codelist",
                    "type":"string"
                },
                "union":{
                    "title":"union",
                    "$ref":"http://portele.de/ows9/test/Union.json"
                },
                "role1":{
                    "title":"role1",
                    "type":"array",
                    "items":{
                        "type":"string",
                        "format":"uri"
                    }
                }
            }
        }
    }
}
```

DataType:

```
{
    "$schema":"http://json-schema.org/draft-03/schema#",
    "id":"http://portele.de/ows9/test/DataType.json",
    "title":"DataType",
    "type":"object",
    "properties":{
        "attributes":{
            "title":"feature/object attributes",
            "type":"object",
            "properties":{
                "datatype":{
                    "title":"datatype",
                    "description":"This is another data type.",
                    "type":"array",
                    "items":{
                        "$ref":"http://portele.de/ows9/test/DataType2.json"
                    }
                },
                "string":{
                    "title":"string",
                    "type":"array",
                    "items":{
                        "type":"string",
                        "minItems":"1"
                    }
                },
                "boolean":{
                    "title":"boolean",
                    "type":"boolean"
                }
            }
        }
    }
}
```

DataType2:

```
{
    "$schema":"http://json-schema.org/draft-03/schema#",
    "id":"http://portele.de/ows9/test/DataType2.json",
    "title":"DataType2",
    "type":"object",
    "properties":{
        "attributes":{
            "title":"feature/object attributes",
            "type":"object",
            "properties":{
                "string":{
                    "title":"string",
                    "type":"array",
                    "items":{
                        "type":"string",
                        "minItems":"1"
                    }
                },
                "integer":{
                    "title":"integer",
                    "type":"integer"
                }
            }
        }
    }
}
```

UnionType:

```
{
    "$schema":"http://json-schema.org/draft-03/schema#",
    "id":"http://portele.de/ows9/test/Union.json",
    "title":"Union",
    "type":"object",
    "properties":{
        "attributes":{
            "title":"feature/object attributes",
            "type":"object",
            "properties":{
                "option1":{
                    "title":"option1",
                    "type":"string",
                    "enum":["val1","val2"]
                },
                "option2":{
                    "title":"option2",
                    "type":"integer"
                },
                "option3":{
                    "title":"option3",
                    "type":"array",
                    "items":{
                        "type":"string"
                    }
                }
            }
        }
    }
}
```

<u>NilUnionType:</u>

```
{
    "$schema":"http://json-schema.org/draft-03/schema#",
    "id":"http://portele.de/ows9/test/NilUnion.json",
    "title":"NilUnion",
    "type":"object",
    "properties":{
        "attributes":{
            "title":"feature/object attributes",
            "type":"object",
            "properties":{
                "value":{
                    "title":"value",
                    "$ref":"http://portele.de/ows9/test/DataType2.json"
                },
                "reason":{
                    "title":"reason",
                    "type":"string"
                }
            }
        }
    }
}
```

The ShapeChange configuration file used in the conversion is:

```
<ShapeChangeConfiguration xmlns:xi="http://www.w3.org/2001/XInclude"
xmlns="http://www.interactive-instruments.de/ShapeChange/Configuration/1.1"
xmlns:sc="http://www.interactive-instruments.de/ShapeChange/Configuration/1.1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.interactive-instruments.de/ShapeChange/Configuration/1.1
http://shapechange.net/resources/schema/ShapeChangeConfiguration.xsd">
    <input>
        <parameter name="inputModelType" value="EA7"/>
        <parameter name="inputFile" value="src/test/resources/test.eap"/>
        <parameter name="appSchemaName" value="Test Schema"/>
        <parameter name="publicOnly" value="true"/>
        <parameter name="checkingConstraints" value="disabled"/>
        <parameter name="sortedSchemaOutput" value="true"/>
        <xi:include href="http://shapechange.net/resources/config/StandardAliases.xml"/>
    </input>
    <log>
        <parameter name="reportLevel" value="INFO"/>
        <parameter name="logFile" value="testResults/ea/log_JsonGsrExtended.xml"/>
    </log>
    <targets>
        <Target class="de.interactive_instruments.ShapeChange.Target.JSON.JsonSchema"
mode="enabled">
            <targetParameter name="outputDirectory"
value="testResults/ea/json/geoservices_extended"/>
            <targetParameter name="defaultEncodingRule" value="geoservices_extended"/>
            <xi:include href="
http://shapechange.net/resources/config/StandardJsonMapEntries.xml"/>
        </Target>
    </targets>
```

```
</ShapeChangeConfiguration>
```

6.4 Examples

6.4.1 TDS

TDS is the National System for Geospatial Intelligence (NSG) Topographic Data Store (TDS). Its content specification is applicable to the collection, storage, manipulation, interchange, and exploitation of geospatial intelligence data. Systems participating within the NSG may utilize the NSG TDS Content Specification in order to ensure consistent NSG-wide geospatial data semantics, adopt common conditions for geospatial intelligence collection/exchange, support net-centric geospatial services, and achieve geospatial data interoperability. More information about version 3.0, the version used in OWS-9, can be found at https://nsgreg.nga.mil/doc/view?i=82045.

The JSON schemas for both the extended and simple GeoServices JSON, the GML 3.2 application schema, and a feature catalogue derived from the TDS 3.0 model, as well as the associated ShapeChange configuration file are available at http://shapechange.net/tmp/ows9/tds.

Figure 4 and Figure 5 show selected TDS features in the ArcGIS.com MapClient on top of the topographic basemap. The features are accessed in JSON from the GeoServices REST API Feature Service and rendered on the map. If a feature is "clicked", the popup displays selected properties of the feature.

Figure 6 and Figure 7 show the same situation and data in Gaia and the features are accessed in GML from the WFS.

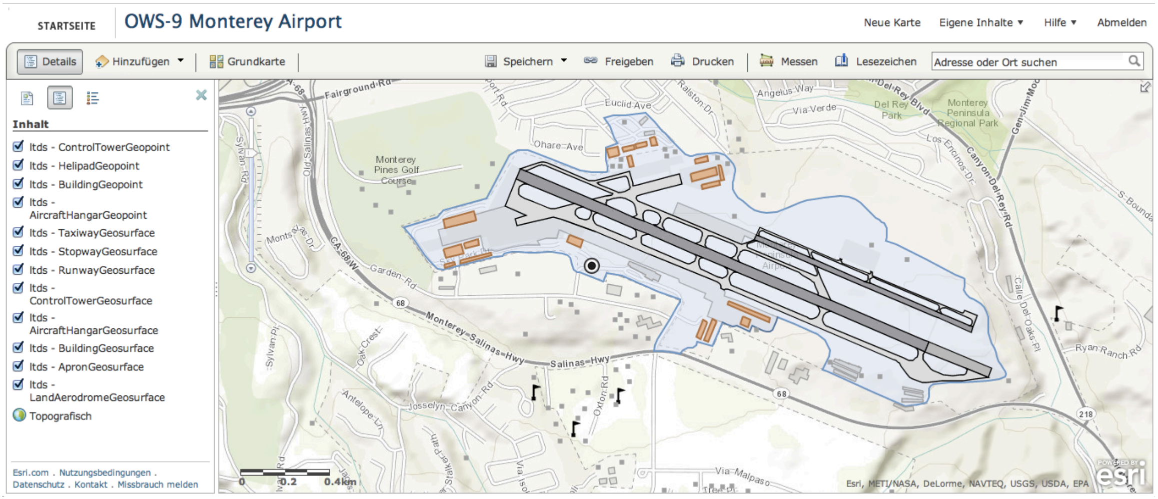
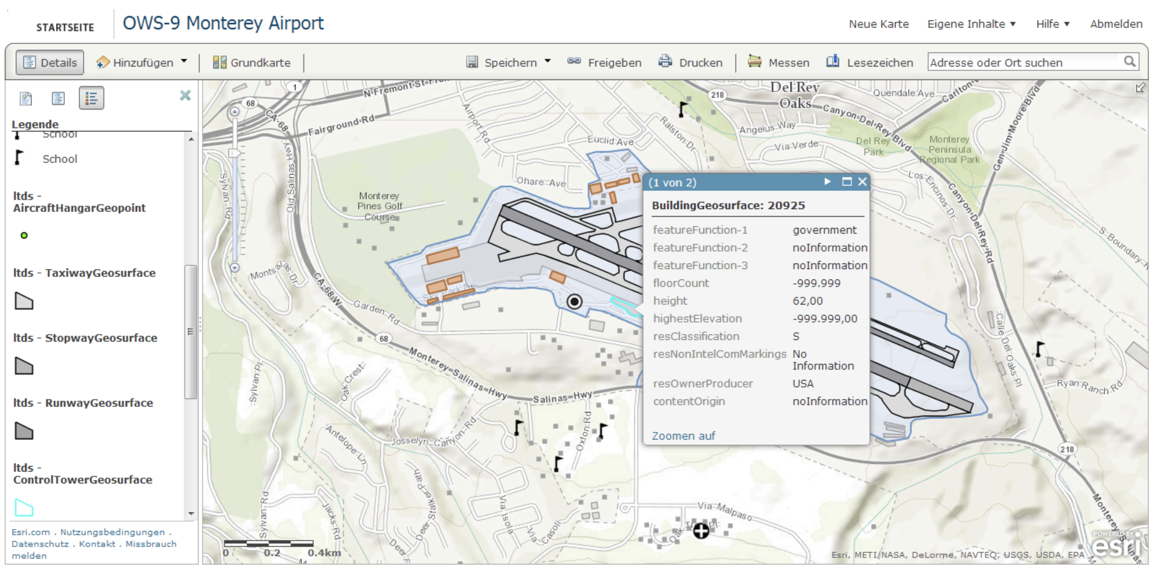


Figure 4 - ArcGIS.com Map Viewer accessing selected feature types of the TDS dataset (JSON)

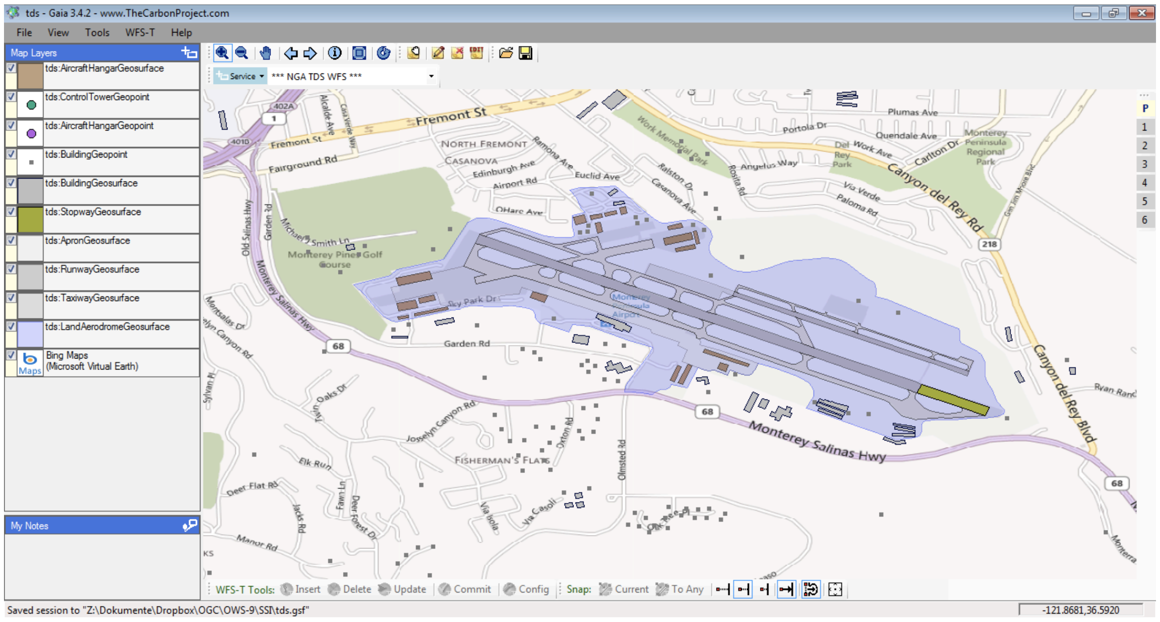


(1 von 2)

BuildingGeosurface: 20925

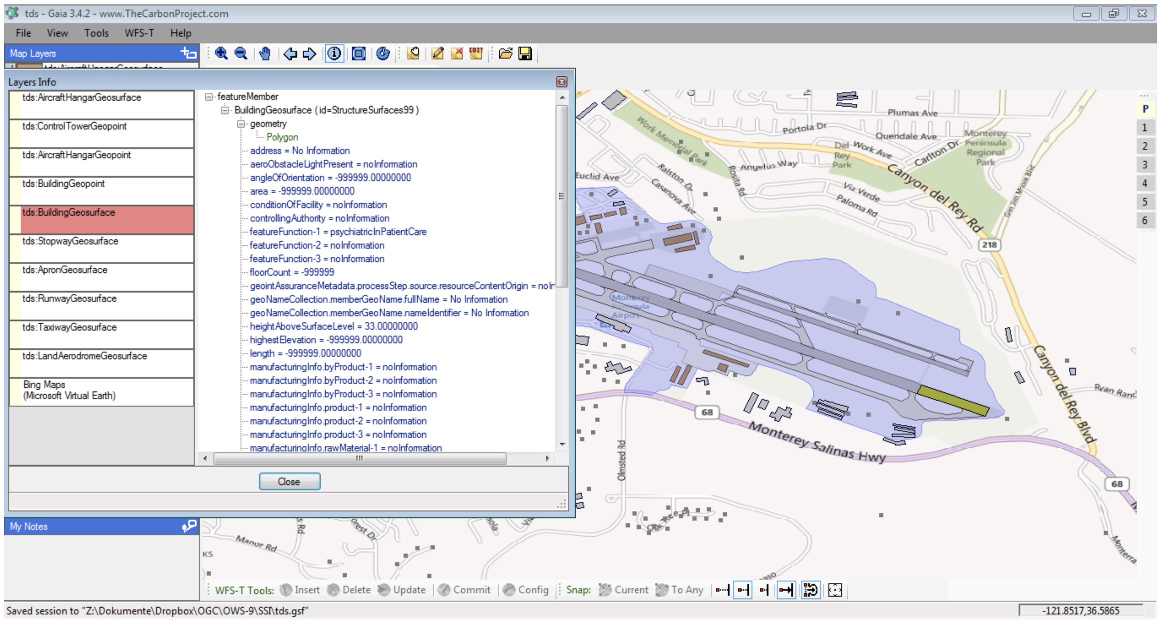
featureFunction-1	government
featureFunction-2	noInformation
featureFunction-3	noInformation
floorCount	-999.999
height	62,00
highestElevation	-999.999,00
resClassification	S
resNonIntelComMarkings	No Information
resOwnerProducer	USA
contentOrigin	noInformation

Figure 5 – Popup showing some of the attributes of a BuidingGeosurface feature (JSON)



Saved session to "Z:\Dokumente\Dropbox\OGC\OWS-9\SSI\tds.gsf"

Figure 6 – Gaia accessing selected feature types of the TDS dataset (GML)



```
featureMember
  BuildingGeosurface ( id=StructureSurfaces99 )
    geometry
      Polygon
    address = No Information
    aeroObstacleLightPresent = noInformation
    angleOfOrientation = -999999.00000000
    area = -999999.00000000
    conditionOfFacility = noInformation
    controllingAuthority = noInformation
    featureFunction-1 = psychiatricInPatientCare
    featureFunction-2 = noInformation
    featureFunction-3 = noInformation
    floorCount = -999999
    geointAssuranceMetadata.processStep.source.resourceContentOrigin = noIn
    geoNameCollection.memberGeoName.fullName = No Information
    geoNameCollection.memberGeoName.nameIdentifier = No Information
    heightAboveSurfaceLevel = 33.00000000
    highestElevation = -999999.00000000
    length = -999999.00000000
    manufacturingInfo.byProduct-1 = noInformation
    manufacturingInfo.byProduct-2 = noInformation
    manufacturingInfo.byProduct-3 = noInformation
    manufacturingInfo.product-1 = noInformation
    manufacturingInfo.product-2 = noInformation
    manufacturingInfo.product-3 = noInformation
    manufacturingInfo.rawMaterial-1 = noInformation
```

Saved session to "Z:\Dokumente\Dropbox\OGC\OWS-9\SSI\tds.gsf"   -121.8517,36.5865

Figure 7 – Additional window showing some of the attributes of a BuildingGeosurface feature (GML)

Table 1 shows a BuildingGeopoint feature encoded in JSON and accessed via the GeoServices REST API Feature Service interface.

Table 3 shows the same feature in encoded in GML and accessed via the WFS interface.

Note that the JSON uses an improved variant of the JSON encoding rule to benefit from capabilities of the GeoServices REST API. The improvements reduce the amount of data that needs to be submitted.

The improvements are that the JSON uses

- shorter property names and
- numeric codes for coded value fields.

In both cases, the shorter names / codes can be considered as aliases for the full names / coded values. The full names and complete coded values are accessed by clients once as part of accessing information about a feature type, the clients show the full information, see Figure 5, for example.

Table 2 shows the information about the BuildingGeosurface feature type. The relevant parts have been highlighted as bold text.

Note that the information for the feature type includes default styling information, too, so that clients can display features automatically using proper symbolization.

Table 1 - JSON instance of a BuildingGeosurface feature

```
{
   "attributes":{
      "objectid":19411,
      "adr":"No Information",
      "awp":-999999,
      "aoo":-999999,
      "ara":-999999,
      "fun":6,
      "caa":-999999,
      "ffn1":813,
      "ffn2":810,
      "ffn3":-999999,
      "bnf":-999999,
      "zi005_fna":"No Information",
      "zi005_nfn":"-999999",
      "hgt":23,
      "zvh":-999999,
      "len_":-999999,
      "zi014_pby1":-999999,
      "zi014_pby2":-999999,
      "zi014_pby3":-999999,
      "zi014_ppo1":-999999,
      "zi014_ppo2":-999999,
      "zi014_ppo3":-999999,
      "zi014_prw1":-999999,
      "zi014_prw2":-999999,
      "zi014_prw3":-999999,
      "lmc":-999999,
      "zi006_mem":"No Information",
      "zi037_rel":-999999,
      "zi037_rfa":-999999,
      "zsax_rs0":"U",
      "zsax_rx3":"No Information",
      "zsax_rx4":"USA",
      "ssr1":-999999,
      "ssr2":-999999,
      "ssr3":-999999,
      "zi001_rcg":-999999,
      "oth":"No Information",
      "ufi":"No Information",
      "vcm1":-999999,
      "vcm2":-999999,
      "vcm3":-999999,
      "voi":"No Information",
      "wid":-999999,
      "zi018_wit":-999999
   },
   "geometry":{
      "rings": [[[-121.896016257,36.5846073490001],[-121.896059618,36.5838938690001],[-
121.89766002,36.5839924160001],[-121.897649917,36.5841248790001],[-
121.898144871,36.5841619170001],[-121.898109394,36.584611291],[-
121.897529938,36.58456793],[-121.897539105,36.58445182],[-
121.897135157,36.5844246890001],[-121.897119983,36.5846743610001],[-
121.896016257,36.5846073490001]]],
      "spatialReference":{"wkid":4326}}
   }
}
```

Table 2 – Feature type information in the GeoServices REST API Feature Service for BuildingGeosurface (shortened)

```
{
   "id":12005,
   "name":"BuildingGeosurface",
   "type":"Feature Layer",
   "displayField":"objectid",
   "description":"",
   "copyrightText":"",
   "relationships":[

   ],
   "geometryType":"GeometryPolygon",
   "minScale":25000,
   "maxScale":10,
   "extent":{
      "xmin":-122.0,
      "ymin":36.40,
      "xmax":-121.0,
      "ymax":37.10,
      "spatialReference":{
         "wkid":4326
      }
   },
   "drawingInfo":{
      "renderer":{
         "type":"simple",
         "symbol":{
            "type":"SFS",
            "style":"SFSSolid",
            "color":[
               160,
               160,
               160,
               64
            ],
            "outline":{
               "type":"SLS",
               "style":"SLSSolid",
               "color":[
                  160,
                  160,
                  160,
                  255
               ],
               "width":1
            }
         },
         "label":"",
         "description":""
      },
      "transparency":0,
      "labelingInfo":null
   },
   "hasAttachments":false,
   "htmlPopupType":"ServerHTMLPopupTypeNone",
   "objectIdField":"objectid",
   "globalIdField":"",
   "typeIdField":"",
   "fields":[
      {
         "name":"objectid",
         "type":"FieldTypeOID",
         "alias":"id",
         "editable":false,
         "domain":null
      },
      {
         "name":"adr",
         "type":"FieldTypeString",
         "length":250,
         "alias":"address",
         "editable":true,
         "domain":null
      },
      {
         "name":"awp",
         "type":"FieldTypeInteger",
         "alias":"aeroObstacleLightPresent",
         "editable":true,
         "domain":{
            "type":"codedValue",
            "name":"BuildingGeosurface_awp",
            "codedValues":[
               {
                  "name":"noInformation",
                  "code":-999999
               },
               {
                  "name":"false",
                  "code":1000
               },
               {
                  "name":"true",
                  "code":1001
               }
            ]
         }
      },
      {
         "name":"aoo",
         "type":"FieldTypeDouble",
         "alias":"angleOfOrientation",
         "editable":true,
         "domain":null
      },
      {
         "name":"ara",
         "type":"FieldTypeDouble",
         "alias":"area",
         "editable":true,
         "domain":null
      },
      {
         "name":"fun",
         "type":"FieldTypeInteger",
         "alias":"conditionOfFacility",
         "editable":true,
         "domain":{
            "type":"codedValue",
            "name":"BuildingGeosurface_fun",
            "codedValues":[
               {
                  "name":"noInformation",
                  "code":-999999
               },
               {
                  "name":"underConstruction",
                  "code":1
               }, ...
               {
                  "name":"planned",
                  "code":17
               },
               {
                  "name":"other",
                  "code":999
               }
            ]
         }
      },
      {
         "name":"caa",
         "type":"FieldTypeInteger",
         "alias":"controllingAuthority",
         "editable":true,
         "domain":{
            "type":"codedValue",
            "name":"BuildingGeosurface_caa",
            "codedValues":[
               {
                  "name":"noInformation",
                  "code":-999999
               },
               {
                  "name":"private",
                  "code":3
               }, ...
               {
                  "name":"public",
                  "code":17
               },
               {
                  "name":"other",
                  "code":999
               }
            ]
         }
      },
      {
         "name":"ffn1",
         "type":"FieldTypeInteger",
         "alias":"featureFunction-1",
         "editable":true,
         "domain":{
            "type":"codedValue",
            "name":"BuildingGeosurface_ffn1",
            "codedValues":[
               {
                  "name":"noInformation",
                  "code":-999999
               },
               {
                  "name":"agriculture",
                  "code":2
               }, ...
               {
                  "name":"meetingPlace",
                  "code":970
               },
               {
                  "name":"other",
                  "code":999
               }
            ]
         }
      }, ...
      {
         "name":"wid",
         "type":"FieldTypeDouble",
         "alias":"width",
         "editable":true,
         "domain":null
      },
      {
         "name":"zi018_wit",
         "type":"FieldTypeInteger",
         "alias":"wirelessTelecomInfo.wirelessTelecomType",
         "editable":true,
         "domain":{
            "type":"codedValue",
            "name":"BuildingGeosurface_zi018_wit",
            "codedValues":[
               {
                  "name":"noInformation",
                  "code":-999999
               },
               {
                  "name":"cellularPhone",
                  "code":1
               }, ...
               {
                  "name":"television",
                  "code":7
               },
               {
                  "name":"other",
                  "code":999
               }
            ]
         }
      }
   ],
   "types":[

   ],
   "templates":[

   ],
   "capabilities":"Query"
}
```

Table 3 - GML instance of the BuildingGeosurface feature from Table 1

```
<tds:BuildingGeosurface gml:id="StructureSurfaces19">
  <tds:geometry>
    <gml:Polygon gml:id="StructureSurfaces.ObjectID.19.SHAPE.Geom_0">
      <gml:exterior>
        <gml:LinearRing>
          <gml:posList>36.5846073490001 -121.896016257 36.5838938690001 -121.896059618
36.5839924160001 -121.89766002 36.5841248790001 -121.897649917 36.5841619170001 -
121.898144871 36.584611291 -121.898109394 36.58456793 -121.897529938 36.58445182 -
121.897539105 36.5844246890001 -121.897135157 36.5846743610001 -121.897119983
36.5846073490001 -121.896016257</gml:posList>
        </gml:LinearRing>
      </gml:exterior>
    </gml:Polygon>
  </tds:geometry>
  <tds:address>No Information</tds:address>
  <tds:aeroObstacleLightPresent>noInformation</tds:aeroObstacleLightPresent>
  <tds:angleOfOrientation>-999999.00000000</tds:angleOfOrientation>
  <tds:area>-999999.00000000</tds:area>
  <tds:conditionOfFacility>fullyFunctional</tds:conditionOfFacility>
  <tds:controllingAuthority>noInformation</tds:controllingAuthority>
  <tds:featureFunction-1>subnationalGovernment</tds:featureFunction-1>
  <tds:featureFunction-2>administration</tds:featureFunction-2>
  <tds:featureFunction-3>noInformation</tds:featureFunction-3>
  <tds:floorCount>-999999</tds:floorCount>

<tds:geointAssuranceMetadata.processStep.source.resourceContentOrigin>noInformation</tds:
geointAssuranceMetadata.processStep.source.resourceContentOrigin>
  <tds:geoNameCollection.memberGeoName.fullName>No
Information</tds:geoNameCollection.memberGeoName.fullName>
  <tds:geoNameCollection.memberGeoName.nameIdentifier>-
999999</tds:geoNameCollection.memberGeoName.nameIdentifier>
  <tds:heightAboveSurfaceLevel>23.00000000</tds:heightAboveSurfaceLevel>
  <tds:highestElevation>-999999.00000000</tds:highestElevation>
  <tds:length>-999999.00000000</tds:length>
  <tds:manufacturingInfo.byProduct-1>noInformation</tds:manufacturingInfo.byProduct-1>
  <tds:manufacturingInfo.byProduct-2>noInformation</tds:manufacturingInfo.byProduct-2>
  <tds:manufacturingInfo.byProduct-3>noInformation</tds:manufacturingInfo.byProduct-3>
  <tds:manufacturingInfo.product-1>noInformation</tds:manufacturingInfo.product-1>
  <tds:manufacturingInfo.product-2>noInformation</tds:manufacturingInfo.product-2>
  <tds:manufacturingInfo.product-3>noInformation</tds:manufacturingInfo.product-3>
  <tds:manufacturingInfo.rawMaterial-1>noInformation</tds:manufacturingInfo.rawMaterial-
1>
  <tds:manufacturingInfo.rawMaterial-2>noInformation</tds:manufacturingInfo.rawMaterial-
2>
  <tds:manufacturingInfo.rawMaterial-3>noInformation</tds:manufacturingInfo.rawMaterial-
3>
  <tds:navigationLandmark>noInformation</tds:navigationLandmark>
  <tds:note.memorandum>No Information</tds:note.memorandum>

<tds:religiousInfo.religiousDesignation>noInformation</tds:religiousInfo.religiousDesigna
tion>

<tds:religiousInfo.religiousFacilityType>noInformation</tds:religiousInfo.religiousFacili
tyType>

<tds:restriction.securityAttributesGroup_resClassification>U</tds:restriction.securityAtt
ributesGroup_resClassification>
  <tds:restriction.securityAttributesGroup_resNonIntelComMarkings>No
Information</tds:restriction.securityAttributesGroup_resNonIntelComMarkings>
  <tds:restriction.securityAttributesGroup_resOwnerProducer>No
Information</tds:restriction.securityAttributesGroup_resOwnerProducer>
  <tds:roofShape-1>noInformation</tds:roofShape-1>
  <tds:roofShape-2>noInformation</tds:roofShape-2>
  <tds:roofShape-3>noInformation</tds:roofShape-3>
  <tds:specifiedEnumerants>No Information</tds:specifiedEnumerants>
  <tds:uniqueEntityIdentifier>No Information</tds:uniqueEntityIdentifier>
  <tds:verticalConstMaterial-1>noInformation</tds:verticalConstMaterial-1>
  <tds:verticalConstMaterial-2>noInformation</tds:verticalConstMaterial-2>
  <tds:verticalConstMaterial-3>noInformation</tds:verticalConstMaterial-3>
  <tds:verticalObstIdentifier>No Information</tds:verticalObstIdentifier>
  <tds:width>-999999.00000000</tds:width>

<tds:wirelessTelecomInfo.wirelessTelecomType>noInformation</tds:wirelessTelecomInfo.wirel
essTelecomType>
</tds:BuildingGeosurface>
```

The JSON data can be validated against their schemas using schema validators. The GeoServices REST API contains a test framework using the JSV validator. The tests included in the draft GeoServices REST API JSON Schema and Examples package (12-068r1, https://portal.opengeospatial.org/files/?artifact_id=49462) contains tests that validate all schemas and all examples used in the draft standard against the schemas.

6.4.2 WXXM

The conversion of the WXXM application schema to JSON Schema is discussed in the OWS-9 Aviation Engineering Report "AIRM Derivation" (12-094).

In general, the encoding rules seem to meet the requirements, but further testing with data is needed.

The main issue identified in the work with WXXM is that for JSON in general no pre-defined encodings for standard types specified in the ISO 19100 series of standards exists. In XML such encodings exist for many of these types either as part of the GML schema, a GML application schema (e.g. GMLCOV, OM-XML) or an ISO/TS-19139-based schema.

6.5 Results and recommendations

The goals of the activity have been achieved. JSON Schema encoding rules have been defined (see 6.6), implemented (see 6.3) and successfully tested (see 6.4).

During the specification, implementation and testing, a number of aspects have been identified that would benefit from additional work on the encoding rules. These are:

- Additional tests using JSON data encoded according to the encoding rule in clients should be executed. This should result in further improvements to the encoding rules in order to construct JSON data in a way that is best suited for consumption by clients. A particular example is improving the GeoServices JSON encoding rule, e.g. by using domains and aliases as tested in the demo service.
- Also, a JSON schema for GeoJSON could be developed, either by the GeoJSON authors or alternatively as an OGC Best Practice. This would allow the JSON Schema encoding rules to be extended with a GeoJSON option.
- Potential extensions of and change requests to the GeoServices JSON encoding could be tested, e.g. for including hyperlinks.

6.6 Encoding rules

6.6.1 Encoding requirements

The requirements specified in GML 3.2 E.2.1.1 apply for application schemas to be converted to JSON Schema, too.

GML 3.2/3.3 and standardized GML application schemas like GMLCOV and OM-XML as well as ISO/TS 19139 provide XML encodings of a significant number of types from the ISO 19100 series of International Standards and the OGC Abstract Specifications that are typically used in application schemas. These do not exist for JSON (yet). As a result, the number of types that should be used in application schemas that are intended to serve as a basis for a JSON encoding are limited. Other types may be used, too, but for those types the JSON schema can only contain generic implementations.

Table 4 specifies the JSON Schema implementation of types from the ISO 19100 series that are commonly used in application schemas. In some cases the implementation depends on the encoding rule. Currently, two encoding rules are specified:

- "geoservices": Encoding rule for GeoServices JSON that is consistent with the OGC candidate standard GeoServices REST API
- "geoservices_extended": Encoding rule for an extended version of GeoServices JSON that goes beyond the underlying feature model supported by the GeoServices REST API

At this point, no JSON Schema for GeoJSON exists and no encoding rule can be specified. However, as the feature and geometry model of GeoJSON is very similar to the one from GeoServices JSON, adding a GeoJSON encoding rule would be straightforward once a JSON Schema for GeoJSON exists.

Table 4 – Existing implementations of types from the ISO 19100 series in JSON Schema

Type	Encoding Rule	Implementation in JSON Schema
Character	all	string
CharacterString	all	string
Boolean	all	boolean
Integer	all	integer
Decimal	all	number
Number	all	number
Real	all	number
Date	all	string with format=time
DateTime	geoservices geoservices_extended	integer with format=utc-millisec string with format=date-time
Time	all	string with format=time
Year	all	integer
URI	all	string with format=uri
GenericName	all	string
LocalName	all	string
ScopedName	all	string
Measure Distance Length Angle Speed Velocity Area Volume Weight Height Pressure Percentage Temperature Bearing	geoservices geoservices_extended	number ref:http://shapechange.net/tmp/ows9/json/measure.json
GM_Envelope	geoservices_extended	ref:http://schemas.opengis.net/gsr/1.0/envelope.json
DirectPosition	all	ref:http://schemas.opengis.net/gsr/1.0/point.json
GM_Point	all	ref:http://schemas.opengis.net/gsr/1.0/point.json
GM_MultiPoint	all	ref:http://schemas.opengis.net/gsr/1.0/multipoint.json
GM_PointArray	all	ref:http://schemas.opengis.net/gsr/1.0/multipoint.json
GM_Curve	all	ref:http://schemas.opengis.net/gsr/1.0/polyline.json
GM_LineString	all	ref:http://schemas.opengis.net/gsr/1.0/polyline.json
GM_CompositeCurve	all	ref:http://schemas.opengis.net/gsr/1.0/polyline.json
GM_MultiCurve	all	ref:http://schemas.opengis.net/gsr/1.0/polyline.json
GM_Surface	all	ref:http://schemas.opengis.net/gsr/1.0/polygon.json
GM_Polygon	all	ref:http://schemas.opengis.net/gsr/1.0/polygon.json
GM_CompositeSurface	all	ref:http://schemas.opengis.net/gsr/1.0/polygon.json
GM_MultiSurface	all	ref:http://schemas.opengis.net/gsr/1.0/polygon.json
GM_Primitive	geoservices_extended	ref:http://schemas.opengis.net/gsr/1.0/geometry.json
GM_Object	geoservices_extended	ref:http://schemas.opengis.net/gsr/1.0/geometry.json

Tagged values may be used to provide additional information to the conversion process.

- *jsonEncodingRule* on application schema packages or model elements in the package: This value controls the applicable conversion rule on a model element. Typically this will be set only on the application schema level or provided as external input to the conversion process.
- *jsonDirectory* on application schema packages: The local path of the schema document of a feature type is "{jsonDirectory}/{typeName}.json".
- *jsonBaseURI* on application schema packages: The URI of a feature type is, for example, "{jsonBaseURI}/{jsonDirectory}/{typeName}.json".
- *jsonLayerTableURI* on a feature type (optional): The URI of the associated Layer/Table resource in a GeoServices REST API Feature Service. This can usually not be set as there will be more then one service that provides information on a feature type. However, if it is provided, explicit "links" properties as specified by JSON Schema can be provided in the schema of the feature type.

If the values are not provided for the first three, the conversion shall use reasonable default values or obtain the information from other another source.

The use of GeoServices JSON / GeoJSON as a basis implies additional requirements for application schemas:

- Every feature type shall have no more than one geometry property (if a feature type has more then one geometry property, the conversion may ignore the additional properties)
- Every geometry property shall have a maximum multiplicity of 1 (the minimum multiplicity may be 0, i.e. in these cases feature instances may also have no geometry)
- Data types and unions shall not have geometry properties

6.6.2 Conversion rules

6.6.2.1 UML package with stereotype <<applicationSchema>>

A local directory with the name of the tagged value *jsonDirectory* is created, if missing.

NOTE If jsonDirectory is not set, ShapeChange uses the value of the tagged value *xmlns*, if it is set, otherwise "default".

6.6.2.2 UML classifier with stereotype <<featureType>> or no stereotype

A JSON Schema file with file name "{classifierName}.json" is created in the directory of the application schema. classifierName is the name of the classifier.

The schema includes the following properties:

- "$schema": "http://json-schema.org/draft-03/schema#"
- "id": "{jsonBaseURI}/{jsonDirectory}/{classifierName}.json"
- "title": a human readable name of the class, if available (optional)
- "description": the documentation of the feature type, if available
- "type": "object"
- "properties":
  - "entityType": a string with the feature type name as the default
  - "geometry": if the type includes a geometry property, a reference to the JSON schema of the associated JSON object implementing the type according to Table 4.
  - "attributes": an object with the properties of the type including all the properties of all supertypes (see 6.6.2.6)

NOTE It has been considered to include an explicit "_links" property as specified in HAL (http://stateless.co/hal_specification.html) to provides a capability to encode href-links to other resources (only for encoding rule "geoservices_extended"). However, as currently no client would know how to use such an extension this has been dropped for now pending further experiments.

EXAMPLE The relevant part of the TDS 3.0 feature type BuildingGeopoint is:

```
{
    "$schema":"http://json-schema.org/draft-03/schema#",
    "id":"http://shapechange.net/tmp/ows9/tds/json/tds/BuildingGeopoint.json",
    "title":"BuildingGeopoint",
    "description":"Building Geospatial Point: A free-standing self-supporting
construction that is roofed, usually walled, and is intended for human occupancy (for
example: a place of work or recreation) and/or habitation. [desc] For example, a
dormitory, a bank, and a restaurant.",
    "type":"object",
    "properties":{
        "entityType":{
            "title":"feature/object type",
            "type":"string",
            "default":"BuildingGeopoint"
        },
        "geometry":{
            "$ref":"http://schemas.opengis.net/gsr/1.0/point.json"
        },
        "attributes":{
            "title":"feature/object attributes",
            "type":"object",
            "properties":{
                . . .
            }
        }
    }
}
```

6.6.2.3 UML classifier with stereotype <<dataType>> or <<union>>

In the encoding rule "geoservices_extended", a JSON Schema file with file name "{classifierName}.json" is created in the directory of the application schema. classifierName is the name of the classifier.

The contents are constructed according to the same rules as in 6.6.2.2 with the following changes:

- No property "entityType" is created
- No property "geometry" is created

In the encoding rule "geoservices", these classifiers are not converted to a JSON Schema file. See 6.6.2.6 for the rules how properties that have a data type / union as a value are converted in this encoding rule.

6.6.2.4 UML classifier with stereotype <<enumeration>> or <<codeList>>

These classifiers are not converted.

6.6.2.5 UML classifier with other stereotypes

These classifiers are not converted.

6.6.2.6 UML properties

Every property is converted according to the following rules. See 6.3 for examples.

If Table 4 specifies an implementation in JSON schema for the value type of the property in the encoding rule, this information is used. If the implementation is a JSON Schema type, e.g. "string", this type is used. If a format is specified, a property "format" is added. If the implementation is a reference to a JSON schema (value in Table 4 has prefix "ref:"), the "$ref" property is set).

If the value type is a type in another application schema and that type is converted to a JSON Schema document in the encoding rule, a "$ref" property is set.

Table 5 – Existing implementations of types from the ISO 19100 series in JSON Schema

Stereotype of value type	**Encoding Rule**	**Implementation in JSON Schema**
codeList	geoservices	string
codeList	geoservices_extended	string with format=uri
enumeration	all	string with enum=an array with all enumerants
featureType, no stereotype	geoservices	integer if tagged value jsonLayerTybleURI is set on the type also add a link with rel=related and href= *jsonLayerTableURI*/{#/attributes/*pname*}?f=json where *pname* is the name of the property
featureType, no stereotype	geoservices_extended	string with format=uri
dataType, union	geoservices	do not represent this property in the JSON Schema, but each of the properties of the data type, i.e. the nested types are flattened to direct properties of the entity type. To ensure uniqueness of the property names, concatenate the names of the nested properties with

		"." as a spearate, e.g. "address.postalCode".
dataType, union	geoservices_extended	$ref to JSON schema of the type

If the value type is specified outside of the application schema and has no known implementation, the JSON Schema type "string" is used in the encoding rule "geoservices" and "any" is used in the encoding rule "geoservices_extended".

If the property has a tagged value <<voidable>>, in the encoding rule "geoservices_extended" add "null" to the value type. In addition, add another property with the name "*pname*_nullReason" of type "string", where pname is the name of the property.

If the property has a minimum and maximum multiplicity of 1, the property is set to required=true.

If the property has a maximum multiplicity greater than 1, the conversion differs between the encoding rules:

- geoservices: create *n* properties with names "*pname*-1", "*pname*-2", …, "*pname-n*" where *pname* is the name of the property and *n* a parameter of the conversion. The default value for n is 3. If the minimum multiplicity of the property is 1, the "*pname*-1" is set to required=true.
- geoservices_extended: create a single property with an array value. minItems is set to the minimum multiplicity of the property, if the value is greater than 0.

6.6.2.7 OCL constraints

OCL constraints may be specified as part of the application schema, but are ignored in the conversion process.

Extensions to this encoding rule may provide conversion rules for constraints.

6.6.2.8 Documentation

If documentation is provided for model elements, these may be converted to "description" properties of the JSON object that represents the model element.

7 SWE Common support

7.1 Background

The OGC SWE Common Data Model 2.0 standard defines amendments and extensions to the encoding rules specified in GML 3.2 (ISO 19136).

The main scope of this activity in OWS-9 is to add support for the SWE Common 2.0 encoding rule in ShapeChange.

In the discussions about the SWE Common 2.0 support, additional requirements have surfaced and these are documented in this clause, too.

7.2 SWE Common 2.0 Encoding Rule

7.2.1 The Encoding Rule

Since the SWE Common Data Model 2.0 encoding rule has been developed using the GML 3.2 encoding rule as a starting point, it shares a large number of conversion rules with the GML 3.2 encoding rule.

The standard itself lists the following differences from the GML 3.2 encoding rule in Annex C:

- Relaxed rule on the mandatory ‘id’ attribute that is kept optional in the SWE Common Data Model schemas.
- Introduced the additional tagged value ‘soft-typed’, so that soft-typed-properties can be encoded in XML with an additional ‘name’ attribute.
- Added support for encoding certain simple-type properties as XML attributes by introducing the additional tagged value ‘asXMLAttribute’.
- Used different base type for <<Type>> stereotype (Elements are derived from swe:AbstractSWE instead of gml:AbstractGML).

A closer analysis revealed that the differences are greater than implied by the standard itself.

As a result, the SWE Common Data Model 2.0 encoding rule implemented in ShapeChange (encoding rule identifier: "ogcSweCommon2") shares all conversion rules with the GML 3.2 encoding rule, with the following differences.

The following conversion rules replace the respective GML 3.2 conversion rules ("rule-xsd-all-naming-gml" and "rule-xsd-cls-standard-swe-property-types"):

rule-xsd-all-naming-swe	Use the naming strategy for schema components as specified by SWE Common Data Model 2.0. The naming of XML Schema elements and types is the same as in the GML 3.2 encoding rule except that a new abstract base XML Schema type (AbstractSWEType) and XML Schema element (AbstractSWE) is introduced instead of the GML base types and elements.
rule-xsd-cls-standard-swe-property-types	The conversion rules for reusing existing or creating new property types are similar to those in GML 3.2, with the following differences: - SWE Common specifies its own swe:AssociationAttributeGroup attribute XML

	Schema group and swe:ReferenceType with a similar content model as the GML 3.2 equivalents. Property values from a code list are referenced using Xlinks as in the GML 3.3 encoding rule. Soft-typed properties are treated differently, see rule-xsd-prop-soft-typed.

In addition, the following additional conversion rules have been implemented in ShapeChange and are included in the SWE Common Data Model 2.0 encoding rule:

rule-xsd-cls-union-as-group-property-type	<<union>> types are encoded as XML Schema groups and property types reference these groups
rule-xsd-prop-xsdAsAttribute	If the tagged value 'asXMLAttribute' is set to 'true' on a property, the property has a maximum multiplicity of 1 and the value of the property is simple, then the property is converted to an XML attribute instead of an XML element. This is actually not a new conversion rule, but the existing conversion rule has been amended to support the 'asXMLAttribute' tagged value in addition to 'xsdAsAttribute'. Both are treated as aliases.
rule-xsd-prop-soft-typed	Properties with a tagged value "soft-typed" with a value "true" are encoded as property elements in XML Schema, but with an additional NCName-valued "name" XML Schema attribute for further disambiguation.
rule-xsd-prop-initialValue	If an attribute has an initial value, the initial value is converted to a default value in XML Schema. If the attribute carries the constraint "{frozen}", too, the initial value is converted to a fixed element value in XML Schema.

The following should be noted, too:

- basicTypes.xsd from SWE Common 2.0 can only be constructed manually and not by automated conversion;
- SWE Common claims in its Foreword that "all elements are substitutable for gml:AbstractValue (and thus transitively for gml:AbstractObject) so that they can

be used directly by GML application schemas;" this seems to be an outdated statement as it is not correct.

Additional information and examples can be found on the ShapeChange website at http://shapechange.net/targets/xsd/swe/.

7.2.2 Test Model

The UML model of SWE Common Data Model 2.0 has been used to test the implementation. It has been integrated also in the ShapeChange unit tests that are run in every new build of ShapeChange.

The UML model has been modified as follows:

1. Special encodings: Data types that have been implemented in the XML Schema manually, i.e. not following the implementing rule, have been deleted. These are all in the package "Basic Types": TimePair, RealPair, IntegerPair, TokenPair, NilValue, EncodedValues, UnitReference, TimePosition, TimeIso8601. They have been implemented in a mapping file[3], so that these map entries are loaded in all standard configurations of ShapeChange. The mapping file is also included as Annex A.
2. Model issues: The aggregation from NilValues to data type NilValue has been changed to an attribute. As a data type, the property must be either an attribute or a composition; this is an issue in the SWE Common model.
3. ShapeChange issues: Due to the original approach in ISO 19103 to keep all classifier names unique, ShapeChange currently assumes that all local classifier names are indeed unique. As a result, the SWE Common types Boolean and Time cannot be reliably distinguished from the ISO 19103 basic types when used as value types of a property. As a consequence, the variants byTime and byBoolean have been removed from the unions AnyNumerical and AnyScalar. Once ShapeChange supports multiple classifiers with the same local name, this should be corrected.

The generated XML Schema documents are compared in the unit tests against reference versions of the SWE Common Data Model 2.0 XML Schema documents. These are modified in two ways:

- The changes in the UML model described above have been reflected.
- Changes to the XML Schema documents that have no impact on the content model, but that follow the schema style used in the GML 3.2 encoding rule implementation in ShapeChange, are also reflected. These changes relate to how anonymous property types are encoded and the inclusion of the all-components schema document.

The test model, the SWE Common test configuration and the reference schemas are included in the ShapeChange distribution.

[3] http://shapechange.net/resources/config/StandardMapEntries_sweCommon.xml

7.3 Related topics

7.3.1 Overview

In the discussions about the SWE Common 2.0 support, additional requirements have surfaced:

- A need for specific attention to supporting the use of SWE Common 2.0 types for representing/encoding the Record type from ISO 19115, as used in DQ_Result.
- The OWS-9 GPS study is developing templates for SWE Common 2.0 data records. As these "record templates" are in a way also the result of a modeling activity, it might be of value to explore if and how UML could be used to describe/model such templates. At the moment, the records are directly modeled as XML instances.

7.3.2 Use of SWE Common data components in ISO 19115-based metadata

As SWE Common Data Model 2.0 provides a generic mechanism for encoding data components that is used in geographic information in the context of sensors, there is a need to use SWE Common encoded data directly in ISO 19115 based metadata.

For example, where ISO 19115 uses the data type Record, the XML encoding specified by ISO/TS 19139 only allows gco:Record, e.g. in

gmd:DQ_QuantitativeResult/gmd:value/gco:Record

However, since swe:DataRecord is an XML encoding of Record, too, it should also be valid to encode data quality results using SWE Common data components, e.g. as

gmd:DQ_QuantitativeResult/gmd:value/swe:DataRecord

Since a revision of ISO/TC 19139 is being prepared by a new project 19115-3 of ISO/TC 211 (and for data quality by project 19157), a comment has been submitted to the project team.

In practice, the only option at the moment that is schema-valid (but conceptually rather a hack) is to embed the swe:DataRecord in the gco:Record, i.e.

gmd:DQ_QuantitativeResult/gmd:value/gco:Record/swe:DataRecord

It should be clarified in the new standard ISO 19115-3 if gco:Record should be a container for XML implementations of record (like swe:DataRecord) or that other implementations of record should be valid, too.

7.3.3 UML Templates for SWE Data Streams and Data Records

7.3.3.1 Scope

The OWS-9 GPS study has been developing templates for SWE Common 2.0 data streams and records. As these templates are in a way also the result of a modeling activity, it might be of value to explore if and how UML could be used to describe/model such templates, too. At the moment, the records are usually modeled directly as XML instances.

This subject was originally not planned to be addressed in OWS-9 and as a result could only be explored on a general level. A sample data stream from the GPS study, the "MSNCC-EPOCHA Data Stream", has been modeled in UML. The template as developed in XML is included as Annex B.

7.3.3.2 UML Profile and Mapping

A UML profile has been specified to allow modeling the different aspects of the sample data stream template in UML. The profile and mapping is described in

Table 6. The Enterprise Architect XML file of the profile is included in Annex C.

Before defining a new profile, an attempt was made to use consider such data stream templates as application schemas according to ISO 19109 and model them according to the existing UML profile for application schemas. However, a model with sufficient information to convert it to SWE Common data components requires a significant amount of additional information and thus requires extensions to the UML profile in any case. Another aspect is that instead of a metamodel for features like the General Feature Model from ISO 19109, we are essentially using a different metamodel for (SWE Common) data components. We could have added all the necessary information as tagged values only instead of defining new stereotypes, but have opted to use stereotypes to link the concepts to the underlying SWE Common metamodel in a clearer way.

The MSNCC-EPOCHA stream modeled in UML using this profile is shown in Figure 8.

The SWE Common XML encoding of the MSNCC-EPOCHA Data Stream template could be derived automatically from the UML model of the data stream. Of course, this is only an example and more a complete mapping a more in-depth analysis and testing would be required.

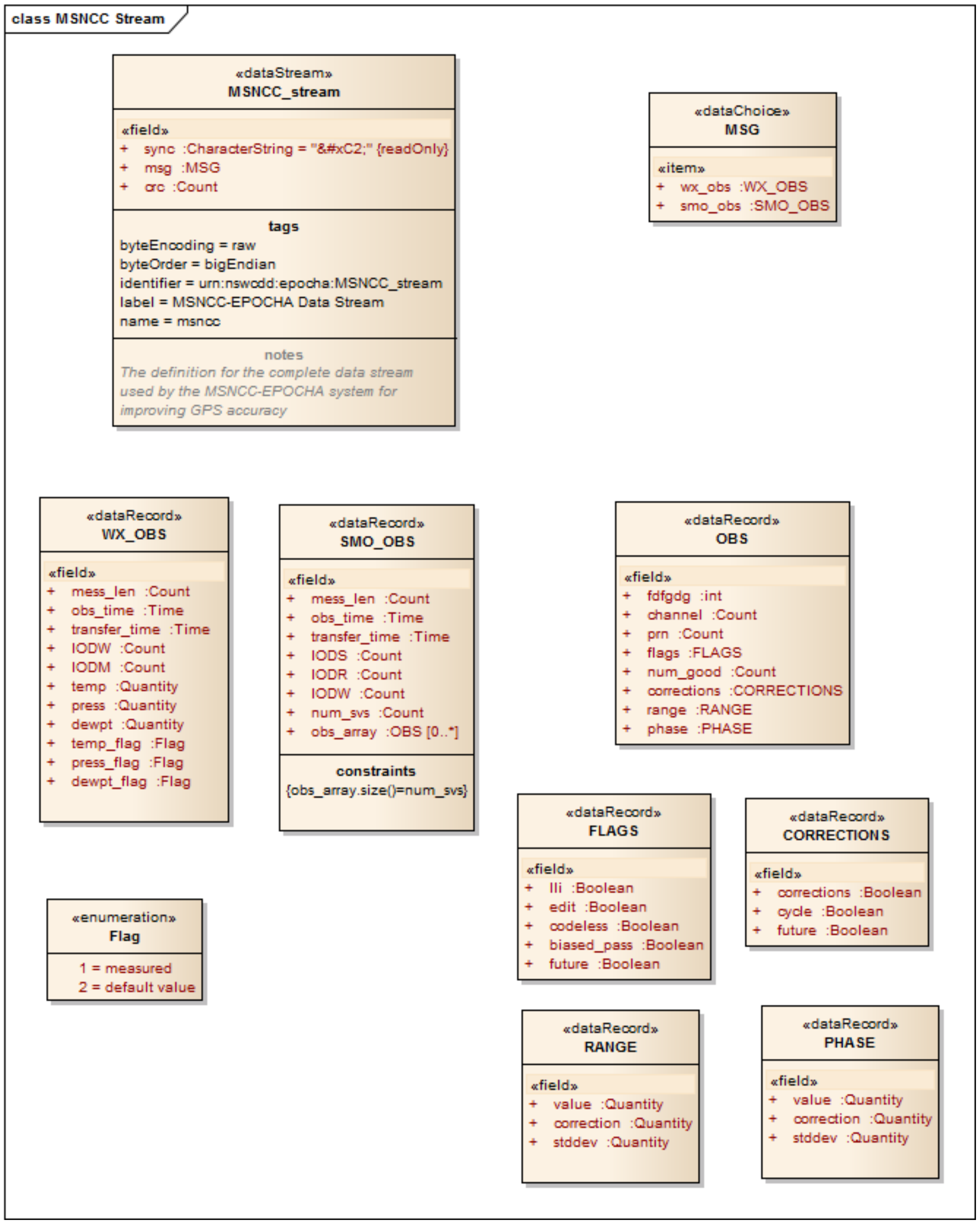


Figure 8 - MSNCC-EPOCHA Data Stream

Table 6 – UML Profile and Mapping of the MSNCC-EPOCHA Data Stream

Stereotype	Sub-element	SWE Common element	Notes
<<dataStream>> (classifier)	-	swe:DataStream object element	
	classifier name	swe:DataStream/@id	
	documentation	swe:DataStream/swe:description	
	tag: identifier	swe:DataStream/swe:identifier	
	tag: label	swe:DataStream/swe:label	
	tag: byteEncoding	swe:DataStream/swe:encoding/ swe:BinaryEncoding/@byteEncoding	Additional information will be needed in the profile to cover other encoding types, too.
	tag: byteOrder	swe:DataStream/swe:encoding/ swe:BinaryEncoding/@byteOrder	Additional information will be needed in the profile to cover other encoding types, too.
	tag: name	swe:DataStream/swe:elementType/ @name	
	attribute with stereotype <<field>>	swe:DataStream/swe:elementType/ swe:DataRecord/swe:field	
<<dataRecord>> (classifier)	-	swe:DataRecord	
	attribute with stereotype <<field>>	swe:DataRecord/swe:field	
<<field>> (attribute)		swe:field	
	attribute name	swe:field/@name	
	value type, multiplicity [1]	Character → swe:field/swe:Text CharacterString → swe:field/swe:Text Count → swe: field/swe:Count	

		Time → swe: field /swe:Time Quantity → swe: field /swe:Quantity etc. custom type → swe: field/swe:DataRecord or swe: field/swe:DataChoice	
	value type, multiplicity [0..*]	as above, but nested in swe:DataArray/swe:elementType For example: swe:field/swe:DataArray/ swe:elementType/swe:Count	swe:elementCount constraint is in the example expressed as an OCL constraint.
	documentation	swe:field/*/swe:description	
	initial value with constraint {readOnly}	swe:field/*/swe:constraint/ swe:AllowedTokens/swe:value	
	tag: definition	swe:field/*/@definition	
	tag: label	swe:field/*/swe:label	
	tag: binaryDataType	in swe:BinaryEncoding: swe:Component/@dataType where swe:Component/@ref is the path to the field in the data stream	
	tag: uom (only for Time, Quantity)	swe:field/*/swe:uom/@code	
	tag: referenceTime (only for Time)	swe:field/*/@referenceTime	
<<dataChoice>> (classifier)	-	swe:DataChoice	
	tag: binaryDataType	in swe:BinaryEncoding: swe:Component/@dataType where swe:Component/@ref is the path to the choiceValue in the data stream	
	attribute with stereotype <<item>>	swe:DataChoice/swe:item	
<<item>> (attribute)	-	swe:item	
	tag: token	../swe:DataChoice/swe:choiceValue/ swe:Category/swe:constraint/	

		swe:AllowedTokens/swe:value
	attribute name	swe:item/@name
	value type	Character → swe:item/swe:Text CharacterString → swe:item/swe:Text Count → swe:item/swe:Count Time → swe:item/swe:Time Quantity → swe:item/swe:Quantity etc. custom type → swe:item/swe:DataRecord or swe:item/swe:DataChoice
<<enumeration>> (classifier)	-	swe:Category
	initial value or enum	swe:Category/swe:constraint/ swe:AllowedTokens/swe:value

7.4 Results and recommendations

In a future revision of the SWE Common Data Model, the following changes should be considered:

- Remove the incorrect statement that “all elements are substitutable for gml:AbstractValue (and thus transitively for gml:AbstractObject) so that they can be used directly by GML application schemas”.
- The current Annex C "UML to XML Schema Encoding Rules" is incomplete. The SWE Common encoding rule in 7.2 might be used to provide a complete encoding rule.

The work on modeling data component templates in UML is experimental and in early stages. Thorough discussion and testing is required to determine, if UML can be useful to model templates for SWE Common data components.

Annex A
Map Entries of the SWE Common Data Model 2.0 Encoding Rule

A.1 General

This file is part of the standard ShapeChange configuration. The reference version is available online at

http://shapechange.net/resources/config/StandardMapEntries_sweCommon.xml

```
<xsdMapEntries
 xmlns="http://www.interactive-instruments.de/ShapeChange/Configuration/1.1">
    <!-- ISO/TS 19103 -->
    <XsdMapEntry type="Character" xsdEncodingRules="ogcSweCommon2"
xmlPropertyType="string" xmlType="string" xmlTypeType="simple" xmlTypeContent="simple"/>
    <XsdMapEntry type="CharacterString" xsdEncodingRules="ogcSweCommon2"
xmlPropertyType="string" xmlType="string" xmlTypeType="simple" xmlTypeContent="simple"/>
    <XsdMapEntry type="Integer" xsdEncodingRules="ogcSweCommon2"
xmlPropertyType="integer" xmlType="integer" xmlTypeType="simple"
xmlTypeContent="simple"/>
    <XsdMapEntry type="Boolean" xsdEncodingRules="ogcSweCommon2"
xmlPropertyType="boolean" xmlType="boolean" xmlTypeType="simple"
xmlTypeContent="simple"/>
    <XsdMapEntry type="Number" xsdEncodingRules="ogcSweCommon2" xmlPropertyType="double"
xmlType="double" xmlTypeType="simple" xmlTypeContent="simple"/>
    <XsdMapEntry type="Real" xsdEncodingRules="ogcSweCommon2" xmlPropertyType="double"
xmlType="double" xmlTypeType="simple" xmlTypeContent="simple"/>
    <XsdMapEntry type="Date" xsdEncodingRules="ogcSweCommon2" xmlPropertyType="date"
xmlType="date" xmlTypeType="simple" xmlTypeContent="simple"/>
    <XsdMapEntry type="DateTime" xsdEncodingRules="ogcSweCommon2"
xmlPropertyType="dateTime" xmlType="dateTime" xmlTypeType="simple"
xmlTypeContent="simple"/>
    <XsdMapEntry type="ClockTime" xsdEncodingRules="ogcSweCommon2"
xmlPropertyType="string" xmlType="string" xmlTypeType="simple" xmlTypeContent="simple"/>
    <XsdMapEntry type="Bit" xsdEncodingRules="ogcSweCommon2" xmlPropertyType="boolean"
xmlType="boolean" xmlTypeType="simple" xmlTypeContent="simple"/>
    <XsdMapEntry type="Decimal" xsdEncodingRules="ogcSweCommon2" xmlPropertyType="double"
xmlType="double" xmlTypeType="simple" xmlTypeContent="simple"/>
    <XsdMapEntry type="Time" xsdEncodingRules="ogcSweCommon2" xmlPropertyType="time"
xmlType="time"  xmlTypeType="simple" xmlTypeContent="simple"/>
    <XsdMapEntry type="Year" xsdEncodingRules="ogcSweCommon2" xmlPropertyType="gYear"
xmlType="gYear"  xmlTypeType="simple" xmlTypeContent="simple"/>
    <XsdMapEntry type="GenericName" xsdEncodingRules="ogcSweCommon2"
xmlPropertyType="anyURI" xmlType="anyURI" xmlTypeType="simple" xmlTypeContent="simple"
xmlTypeNilReason="false"/>
    <XsdMapEntry type="LocalName" xsdEncodingRules="ogcSweCommon2"
xmlPropertyType="anyURI" xmlType="anyURI" xmlTypeType="simple" xmlTypeContent="simple"
xmlTypeNilReason="false"/>
    <XsdMapEntry type="ScopedName" xsdEncodingRules="ogcSweCommon2"
xmlPropertyType="anyURI" xmlType="anyURI" xmlTypeType="simple" xmlTypeContent="simple"
xmlTypeNilReason="false"/>
    <XsdMapEntry type="Any" xsdEncodingRules="ogcSweCommon2" xmlElement="any"
xmlPropertyType="anyType" xmlTypeNilReason="false"/>
    <XsdMapEntry type="UnitOfMeasure" xsdEncodingRules="ogcSweCommon2"
xmlPropertyType="swe:UnitReference" xmlType="swe:UnitReference" xmlTypeContent="simple"
xmlTypeNilReason="false"/>
    <XsdMapEntry type="UomTime" xsdEncodingRules="ogcSweCommon2"
xmlPropertyType="swe:UnitReference" xmlType="swe:UnitReference" xmlTypeContent="simple"
xmlTypeNilReason="false"/>
    <XsdMapEntry type="Dictionary" xsdEncodingRules="ogcSweCommon2"
xmlPropertyType="swe:Reference" xmlType="swe:Reference" xmlTypeContent="simple"
xmlTypeNilReason="false"/>
```

```
    <!-- ISO 19108 -->
    <XsdMapEntry type="TM_Position" xsdEncodingRules="ogcSweCommon2"
xmlPropertyType="swe:TimePosition" xmlType="swe:TimePosition" xmlTypeType="simple"
xmlTypeContent="simple" xmlTypeNilReason="false"/>
    <XsdMapEntry type="TM_TemporalCRS" xsdEncodingRules="ogcSweCommon2"
xmlPropertyType="anyURI" xmlType="anyURI" xmlTypeType="simple" xmlTypeContent="simple"
xmlTypeNilReason="false"/>
    <XsdMapEntry type="TM_IndeterminateValue" xsdEncodingRules="ogcSweCommon2"
xmlPropertyType="swe:TimeIndeterminateValue" xmlType="swe:TimeIndeterminateValue"
xmlTypeType="simple" xmlTypeContent="simple" xmlTypeNilReason="false"/>
    <!-- ISO 19111 -->
    <XsdMapEntry type="SC_CRS" xsdEncodingRules="ogcSweCommon2" xmlPropertyType="anyURI"
xmlType="anyURI" xmlTypeType="simple" xmlTypeContent="simple" xmlTypeNilReason="false"/>
    <!-- ISO 19118 -->
    <XsdMapEntry type="Binary" xsdEncodingRules="ogcSweCommon2"
xmlPropertyType="hexBinary" xmlType="hexBinary"  xmlTypeType="simple"
xmlTypeContent="simple"/>
    <!-- SWE Common Basic Types -->
    <XsdMapEntry type="TokenPair" xsdEncodingRules="ogcSweCommon2"
xmlPropertyType="swe:TokenPair" xmlType="swe:TokenPair" xmlTypeType="simple"
xmlTypeContent="simple" xmlTypeNilReason="false"/>
    <XsdMapEntry type="IntegerPair" xsdEncodingRules="ogcSweCommon2"
xmlPropertyType="swe:IntegerPair" xmlType="swe:IntegerPair" xmlTypeType="simple"
xmlTypeContent="simple" xmlTypeNilReason="false"/>
    <XsdMapEntry type="RealPair" xsdEncodingRules="ogcSweCommon2"
xmlPropertyType="swe:RealPair" xmlType="swe:RealPair" xmlTypeType="simple"
xmlTypeContent="simple" xmlTypeNilReason="false"/>
    <XsdMapEntry type="TimePair" xsdEncodingRules="ogcSweCommon2"
xmlPropertyType="swe:TimePair" xmlType="swe:TimePair" xmlTypeType="simple"
xmlTypeContent="simple" xmlTypeNilReason="false"/>
    <XsdMapEntry type="EncodedValues" xsdEncodingRules="ogcSweCommon2"
xmlPropertyType="swe:EncodedValuesPropertyType" xmlTypeNilReason="false"/>
    <XsdMapEntry type="NilValue" xsdEncodingRules="ogcSweCommon2"
xmlPropertyType="swe:NilValue" xmlType="swe:NilValue" xmlTypeContent="simple"
xmlTypeNilReason="false"/>
    <XsdMapEntry type="TimePosition" xsdEncodingRules="ogcSweCommon2"
xmlPropertyType="swe:TimePosition" xmlType="swe:TimePosition" xmlTypeType="simple"
xmlTypeContent="simple" xmlTypeNilReason="false"/>
    <XsdMapEntry type="TimeIso8601" xsdEncodingRules="ogcSweCommon2"
xmlPropertyType="swe:TimeIso8601" xmlType="swe:TimeIso8601" xmlTypeType="simple"
xmlTypeContent="simple" xmlTypeNilReason="false"/>
    <XsdMapEntry type="UnitReference" xsdEncodingRules="ogcSweCommon2"
xmlPropertyType="swe:UnitReference" xmlType="swe:UnitReference" xmlTypeContent="simple"
xmlTypeNilReason="false"/>
</xsdMapEntries>
```

Annex B
MSNCC-EPOCHA Data Stream

```
<swe:DataStream id="MSNCC_stream"
   xmlns:swe="http://www.opengis.net/swe/2.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   xsi:schemaLocation="http://www.opengis.net/swe/2.0
http://schemas.opengis.net/sweCommon/2.0/swe.xsd">
  <!-- -->
  <swe:identifier>urn:nswcdd:epocha:MSNCC_stream</swe:identifier>
  <swe:label>MSNCC-EPOCHA Data Stream</swe:label>
  <swe:description>
      The definition for the complete data stream used by the MSNCC-EPOCHA system for
improving GPS accuracy
    </swe:description>
  <swe:elementType name="msncc">
    <swe:DataRecord>
      <swe:field name="sync">
        <swe:Text definition="urn:nswcdd:epocha:sync">
          <swe:label>Sync Byte</swe:label>
          <swe:constraint>
            <swe:AllowedTokens>
              <swe:value>&#xC2;</swe:value>
            </swe:AllowedTokens>
          </swe:constraint>
        </swe:Text>
      </swe:field>
      <swe:field name="msg">
        <swe:DataChoice>
          <swe:choiceValue>
            <swe:Category>
              <swe:constraint>
                <swe:AllowedTokens>
                  <swe:value>&#x0122;</swe:value>
                  <swe:value>&#x0120;</swe:value>
                </swe:AllowedTokens>
              </swe:constraint>
            </swe:Category>
          </swe:choiceValue>
          <!-- weather observation message -->
          <swe:item name="wx_obs">
            <swe:DataRecord>
              <swe:field name="mess_len">
                <swe:Count definition="urn:nswcdd:epocha:mess_len">
                  <swe:label>Message Length</swe:label>
                </swe:Count>
              </swe:field>
              <swe:field name="obs_time">
                <swe:Time
definition="http://www.opengis.net/def/property/OGC/0/SamplingTime"
referenceTime="1970-01-01T00:00:00Z">
                  <swe:label>Observation Time</swe:label>
                  <swe:description>Time of observation in modified Julian
Time</swe:description>
                  <swe:uom code="s"/>
                </swe:Time>
              </swe:field>
              <swe:field name="transfer_time">
                <swe:Time definition="urn:nswcdd:epocha:transfer_time">
                  <swe:label>Transfer Time</swe:label>
                  <swe:uom code="s"/>
                </swe:Time>
              </swe:field>
              <swe:field name="IODW">
```

```
                <swe:Count definition="urn:nswcdd:epocha:IODW">
                  <swe:label>IODW Message Pointer</swe:label>
                </swe:Count>
              </swe:field>
              <swe:field name="IODM">
                <swe:Count definition="urn:nswcdd:epocha:IODM">
                  <swe:label>IODM Message Pointer</swe:label>
                </swe:Count>
              </swe:field>
              <swe:field name="temp">
                <swe:Quantity definition="urn:nswcdd:epocha:temperature">
                  <swe:label>Atmospheric Temperature</swe:label>
                  <swe:uom code="Cel"/>
                </swe:Quantity>
              </swe:field>
              <swe:field name="press">
                <swe:Quantity definition="urn:nswcdd:epocha:atmosphericPressure">
                  <swe:label>Atmospheric Pressure</swe:label>
                  <swe:uom code="mbars"/>
                </swe:Quantity>
              </swe:field>
              <swe:field name="dewpt">
                <swe:Quantity definition="urn:nswcdd:epocha:dewPoint">
                  <swe:label>Dew Point</swe:label>
                  <swe:uom code="Cel"/>
                </swe:Quantity>
              </swe:field>
              <!-- NOTE: Consider whether to treat these flags as Category or Count with
allowed values of -->
              <!--    1=measured and 2=default; add codespace for definitions? -->
              <swe:field name="temp_flag">
                <swe:Category definition="urn:nswcdd:epocha:flags.temp_flag">
                  <swe:label>Temperature Source Flag</swe:label>
                  <swe:description>
Designates source of temperature measurement (1=measured, 2=default value)
</swe:description>
                  <swe:constraint>
                    <swe:AllowedTokens>
                      <swe:value>1</swe:value>
                      <swe:value>2</swe:value>
                    </swe:AllowedTokens>
                  </swe:constraint>
                </swe:Category>
              </swe:field>
              <swe:field name="press_flag">
                <swe:Category definition="urn:nswcdd:epocha:flags.press_flag">
                  <swe:label>Pressure Source Flag</swe:label>
                  <swe:description>Designates source of pressure measurement (1=measured,
2=default value)</swe:description>
                  <swe:constraint>
                    <swe:AllowedTokens>
                      <swe:value>1</swe:value>
                      <swe:value>2</swe:value>
                    </swe:AllowedTokens>
                  </swe:constraint>
                </swe:Category>
              </swe:field>
              <swe:field name="dewpt_flag">
                <swe:Category definition="urn:nswcdd:epocha:flags.dewpt_flag">
                  <swe:label>Dew Point Source Flag</swe:label>
                  <swe:description>Designates source of dew point measurement
(1=measured, 2=default value)</swe:description>
                  <swe:constraint>
                    <swe:AllowedTokens>
                      <swe:value>1</swe:value>
                      <swe:value>2</swe:value>
                    </swe:AllowedTokens>
                  </swe:constraint>
                </swe:Category>
              </swe:field>
            </swe:DataRecord>
          </swe:item>
          <!-- smooth observation message -->
          <swe:item name="smo_obs">
            <swe:DataRecord>
              <swe:field name="mess_len">
                <swe:Count definition="urn:nswcdd:epocha:mess_len">
                  <swe:label>Message Length</swe:label>
                </swe:Count>
              </swe:field>
              <swe:field name="obs_time">
                <swe:Time
definition="http://www.opengis.net/def/property/OGC/0/SamplingTime" referenceTime="1970-
01-01T00:00:00Z">
                  <swe:label>Observation Time</swe:label>
                  <swe:uom code="s"/>
                </swe:Time>
              </swe:field>
              <swe:field name="transfer_time">
                <swe:Time definition="urn:nswcdd:epocha:transfer_time">
                  <swe:label>Transfer Time</swe:label>
                  <swe:uom code="s"/>
                </swe:Time>
              </swe:field>
              <swe:field name="IODS">
                <swe:Count definition="urn:nswcdd:epocha:IODS">
                  <swe:label>IODS Message Pointer</swe:label>
                </swe:Count>
              </swe:field>
              <swe:field name="IODR">
                <swe:Count definition="urn:nswcdd:epocha:IODR">
                  <swe:label>IODR Message Pointer</swe:label>
                </swe:Count>
              </swe:field>
              <swe:field name="IODW">
                <swe:Count definition="urn:nswcdd:epocha:IODW">
                  <swe:label>IODW Message Pointer</swe:label>
                </swe:Count>
              </swe:field>
              <swe:field name="num_svs">
                <swe:Count id="NUM_SVS" definition="urn:nswcdd:epocha:num_svs">
                  <swe:label>Number of observations</swe:label>
                </swe:Count>
              </swe:field>
              <swe:field name="obs_array">
                <swe:DataArray>
                  <swe:elementCount xlink:href="#NUM_SVS"/>
                  <swe:elementType name="obs">
                    <swe:DataRecord>
                      <swe:field name="channel">
                        <swe:Count definition="urn:nswcdd:epocha:channel">
                          <swe:label>Receiver Channel</swe:label>
                        </swe:Count>
                      </swe:field>
                      <swe:field name="prn">
                        <swe:Count definition="urn:nswcdd:epocha:prn">
                          <swe:label>PRN</swe:label>
                        </swe:Count>
                      </swe:field>
                      <swe:field name="flags">
                        <swe:DataRecord>
                          <swe:field name="lli">
                            <swe:Boolean definition="urn:nswcdd:epocha:flags.lli"/>
                          </swe:field>
                          <swe:field name="edit">
                            <swe:Boolean definition="urn:nswcdd:epocha:flags.edit"/>
                          </swe:field>
                          <swe:field name="codeless">
                            <swe:Boolean definition="urn:nswcdd:epocha:flags.codeless"/>
                          </swe:field>
                          <swe:field name="biased_pass">
                            <swe:Boolean
definition="urn:nswcdd:epocha:flags.biased_pass"/>
                          </swe:field>
                          <swe:field name="future">
                            <swe:Boolean definition="urn:nswcdd:epocha:flags.future"/>
                          </swe:field>
                        </swe:DataRecord>
                      </swe:field>
                      <swe:field name="num_good">
                        <swe:Count definition="urn:nswcdd:epocha:num_good">
                          <swe:label>Number of observations</swe:label>
                        </swe:Count>
                      </swe:field>
                      <swe:field name="corrections">
                        <swe:DataRecord>
                          <swe:field name="corrections">
                            <swe:Boolean definition="urn:nswcdd:epocha:corrections"/>
                          </swe:field>
                          <swe:field name="cycle">
                            <swe:Boolean
definition="urn:nswcdd:epocha:corrections.cycle"/>
                          </swe:field>
                          <swe:field name="future">
                            <swe:Boolean
definition="urn:nswcdd:epocha:corrections.future"/>
                          </swe:field>
                        </swe:DataRecord>
                      </swe:field>
                      <swe:field name="range">
                        <swe:DataRecord>
                          <swe:field name="value">
                            <swe:Quantity definition="urn:nswcdd:epocha:range.value">
                              <swe:label>Pseudo Range Value</swe:label>
                              <swe:uom code="m"/>
                            </swe:Quantity>
                          </swe:field>
                          <swe:field name="correction">
                            <swe:Quantity definition="urn:nswcdd:epocha:range.value">
                              <swe:label>Pseudo Range Correction</swe:label>
                              <swe:uom code="m"/>
                            </swe:Quantity>
                          </swe:field>
                          <swe:field name="sttddev">
                            <swe:Quantity definition="urn:nswcdd:epocha:range.value">
                              <swe:label>Pseudo Range Standard Deviation</swe:label>
                              <swe:uom code="m"/>
                            </swe:Quantity>
                          </swe:field>
                        </swe:DataRecord>
                      </swe:field>
                      <swe:field name="phase">
                        <swe:DataRecord>
                          <swe:field name="value">
                            <swe:Quantity definition="urn:nswcdd:epocha:range.value">
                              <swe:label>Phase Value</swe:label>
                              <swe:uom code="m"/>
                            </swe:Quantity>
                          </swe:field>
                          <swe:field name="correction">
                            <swe:Quantity definition="urn:nswcdd:epocha:range.value">
                              <swe:label>Phase Correction</swe:label>
                              <swe:uom code="m"/>
                            </swe:Quantity>
                          </swe:field>
                          <swe:field name="sttddev">
                            <swe:Quantity definition="urn:nswcdd:epocha:range.value">
                              <swe:label>Phase Standard Deviation</swe:label>
                              <swe:uom code="m"/>
                            </swe:Quantity>
                          </swe:field>
                        </swe:DataRecord>
                      </swe:field>
                    </swe:DataRecord>
                  </swe:elementType>
                </swe:DataArray>
              </swe:field>
            </swe:DataRecord>
          </swe:item>
        </swe:DataChoice>
      </swe:field>
      <!-- crc is required BINEX field and common to all messages -->
      <swe:field name="crc">
        <swe:Count definition="urn:nswcdd:epocha:crc">
          <swe:label>Checksum</swe:label>
        </swe:Count>
      </swe:field>
    </swe:DataRecord>
  </swe:elementType>
  <!-- -->
  <swe:encoding>
    <swe:BinaryEncoding byteOrder="bigEndian" byteEncoding="raw">
      <!-- common message encodings -->
      <swe:member>
        <swe:Component ref="msncc/sync"
dataType="http://www.opengis.net/def/dataType/OGC/0/unsignedByte"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="msncc/crc"
dataType="http://www.opengis.net/def/dataType/OGC/0/unsignedShort"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="msncc/msg/choiceValue"
dataType="http://www.opengis.net/def/dataType/OGC/0/unsignedShort"/>
      </swe:member>
      <!-- wx_obs encodings -->
      <swe:member>
        <swe:Component ref="msncc/msg/wx_obs/mess_len"
dataType="http://www.opengis.net/def/dataType/OGC/0/unsignedShort"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="msncc/msg/wx_obs/obs_time"
dataType="http://www.opengis.net/def/dataType/OGC/0/double"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="msncc/msg/wx_obs/transfer_time"
dataType="http://www.opengis.net/def/dataType/OGC/0/float32"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="msncc/msg/wx_obs/IODW"
dataType="http://www.opengis.net/def/dataType/OGC/0/unsignedShort"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="msncc/msg/wx_obs/IODM"
dataType="http://www.opengis.net/def/dataType/OGC/0/unsignedShort"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="msncc/msg/wx_obs/temp"
dataType="http://www.opengis.net/def/dataType/OGC/0/float32"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="msncc/msg/wx_obs/press"
dataType="http://www.opengis.net/def/dataType/OGC/0/float32"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="msncc/msg/wx_obs/dewpt"
dataType="http://www.opengis.net/def/dataType/OGC/0/float32"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="msncc/msg/wx_obs/temp_flag"
dataType="http://www.opengis.net/def/dataType/OGC/0/unsignedShort"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="msncc/msg/wx_obs/press_flag"
dataType="http://www.opengis.net/def/dataType/OGC/0/unsignedShort"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="msncc/msg/wx_obs/dewpt_flag"
dataType="http://www.opengis.net/def/dataType/OGC/0/unsignedShort"/>
      </swe:member>
      <!-- smo_obs encodings -->
      <swe:member>
        <swe:Component ref="smo_obs/mess_len"
dataType="http://www.opengis.net/def/dataType/OGC/0/unsignedShort"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="smo_obs/obs_time"
dataType="http://www.opengis.net/def/dataType/OGC/0/double"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="smo_obs/transfer_time"
dataType="http://www.opengis.net/def/dataType/OGC/0/float32"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="smo_obs/IODS"
dataType="http://www.opengis.net/def/dataType/OGC/0/unsignedShort"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="smo_obs/IODR"
dataType="http://www.opengis.net/def/dataType/OGC/0/unsignedShort"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="smo_obs/IODW"
dataType="http://www.opengis.net/def/dataType/OGC/0/unsignedShort"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="smo_obs/num_svs"
dataType="http://www.opengis.net/def/dataType/OGC/0/unsignedByte"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="smo_obs/obs_array/obs/channel"
dataType="http://www.opengis.net/def/dataType/OGC/0/unsignedByte"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="smo_obs/obs_array/obs/prn"
dataType="http://www.opengis.net/def/dataType/OGC/0/unsignedByte"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="smo_obs/obs_array/obs/flags/lli"
dataType="http://www.opengis.net/def/dataType/OGC/0/boolean" bitLength="1"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="smo_obs/obs_array/obs/flags/edit"
dataType="http://www.opengis.net/def/dataType/OGC/0/boolean" bitLength="1"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="smo_obs/obs_array/obs/flags/codeless"
dataType="http://www.opengis.net/def/dataType/OGC/0/boolean" bitLength="1"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="smo_obs/obs_array/obs/flags/biased_pass"
dataType="http://www.opengis.net/def/dataType/OGC/0/boolean" bitLength="1"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="smo_obs/obs_array/obs/flags/future"
dataType="http://www.opengis.net/def/dataType/OGC/0/boolean" bitLength="12"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="smo_obs/obs_array/obs/num_good"
dataType="http://www.opengis.net/def/dataType/OGC/0/unsignedShort"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="smo_obs/obs_array/obs/corrections"
dataType="http://www.opengis.net/def/dataType/OGC/0/unsignedByte"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="smo_obs/obs_array/obs/corrections/cycle"
dataType="http://www.opengis.net/def/dataType/OGC/0/boolean" bitLength="1"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="smo_obs/obs_array/obs/corrections/future"
dataType="http://www.opengis.net/def/dataType/OGC/0/boolean" bitLength="7"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="smo_obs/obs_array/obs/range/value"
dataType="http://www.opengis.net/def/dataType/OGC/0/double"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="smo_obs/obs_array/obs/range/correction"
dataType="http://www.opengis.net/def/dataType/OGC/0/double"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="smo_obs/obs_array/obs/range/stddev"
dataType="http://www.opengis.net/def/dataType/OGC/0/float32"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="smo_obs/obs_array/obs/phase/value"
dataType="http://www.opengis.net/def/dataType/OGC/0/double"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="smo_obs/obs_array/obs/phase/correction"
dataType="http://www.opengis.net/def/dataType/OGC/0/double"/>
      </swe:member>
      <swe:member>
        <swe:Component ref="smo_obs/obs_array/obs/phase/stddev"
dataType="http://www.opengis.net/def/dataType/OGC/0/float32"/>
      </swe:member>
    </swe:BinaryEncoding>
  </swe:encoding>
  <swe:values xlink:href="http://myser.org/epocha/stream"/>
</swe:DataStream>
```

Annex C
UML Profile used for the SWE Common data stream

```
<UMLProfile>

    <Documentation id="OWS-9" name="UML Profile for OWS-9 SWE Common data streams"
version="0.1" notes="This profile defines a set of stereotypes and tagged values for
defining SWE Common data streams."/>

    <Content>

        <Stereotypes>

            <Stereotype name="dataStream" notes="A SWE Common DataStream.">
                <AppliesTo>
                    <Apply type="class"/>
                </AppliesTo>
                <TaggedValues>
                    <Tag name="identifier" description="URI of the data stream" />
                    <Tag name="label" description="Human readable label for the data
stream"/>
                    <Tag name="byteEncoding" description="Byte encoding in a binary
encoding"/>
                    <Tag name="byteOrder" description="Byte order in a binary encoding"/>
                    <Tag name="name" description="Name of the element type in the data
stream"/>
                </TaggedValues>
            </Stereotype>

            <Stereotype name="dataRecord" notes="A SWE Common DataRecord.">
                <AppliesTo>
                    <Apply type="class"/>
                </AppliesTo>
                <TaggedValues>
                </TaggedValues>
            </Stereotype>

            <Stereotype name="dataChoice" notes="A SWE Common DataChoice.">
                <AppliesTo>
                    <Apply type="class"/>
                </AppliesTo>
                <TaggedValues>
                    <Tag name="binaryDataType" description="Data type of the choiceValue
field in a binary encoding"/>
                </TaggedValues>
            </Stereotype>

            <Stereotype name="enumeration" notes="An enumeration.">
                <AppliesTo>
                    <Apply type="class"/>
                </AppliesTo>
                <TaggedValues>
                </TaggedValues>
            </Stereotype>

            <Stereotype name="field" notes="A SWE Common field.">
                <AppliesTo>
                    <Apply type="attribute"/>
                </AppliesTo>
                <TaggedValues>
                    <Tag name="definition" description="URI of the field definition"/>
                    <Tag name="label" description="Human readable label for the field"/>
                    <Tag name="uom" description="Unit of measure of the field. Only for
Quantity and Time fields."/>
```

```
                    <Tag name="referenceTime" description="Reference time attribute. Only
for Time fields."/>
                    <Tag name="binaryDataType" description="Data type in a binary
encoding"/>
                </TaggedValues>
            </Stereotype>

            <Stereotype name="item" notes="A SWE Common item in a DataChoice.">
                <AppliesTo>
                    <Apply type="attribute"/>
                </AppliesTo>
                <TaggedValues>
                    <Tag name="token" description="Token identifying the item in the
choice."/>
                </TaggedValues>
            </Stereotype>

        </Stereotypes>
    </Content>
</UMLProfile>
```