Open Geospatial Consortium

Submission Date: 2015-09-15

Approval Date: 2015-09-17

Publication Date: 2015-12-09

External identifier of this OGC® document: http://www.opengis.net/doc/dp/om-json/

Internal reference number of this OGC® document: 15-100r1

Category: OGC® Discussion Paper

Editors: Simon J D Cox, Peter Taylor

OGC Observations and Measurements – JSON implementation

Warning

This document is not an OGC Standard. This document is an OGC Discussion Paper and is therefore not an official position of the OGC membership. This document is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, an OGC Discussion Paper should not be referenced as required or mandatory technology in procurements.

Document type: OGC® Discussion Paper
Document subtype: Not applicable
Document stage: Approved for Public Release
Document language: English

Contents

i. Abstract

This Discussion Paper specifies a potential OGC Candidate Standard for a JSON implementation of the OGC and ISO Observations and Measurements (O&M) conceptual model (*OGC Observations and Measurements v2.0* also published as ISO/DIS 19156). This encoding is expected to be useful in RESTful implementations of observation services.

More specifically, this Discussion Paper defines JSON schemas for observations, and for features involved in sampling when making observations. These provide document models for the exchange of information describing observation acts and their results, both within and between different scientific and technical communities.

ii. Keywords

The following are keywords to be used by search engines and document catalogues.

ogcdoc, ogc documents, O&M, observations, measurements, sampling, RESTful API, JSON encoding

iii. Preface

The OGC Observations and Measurements JSON standard defines a new implementation of O&M compatible with current expectations for web data delivery, in particular REST APIs.

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.

iv. Submitting organizations

The following organizations submitted this Document to the Open Geospatial Consortium (OGC):

CSIRO Australia

v. Submitters

All questions regarding this submission should be directed to the editor or the submitters:

Name	Affiliation
Simon J D Cox	CSIRO
Peter Taylor	CSIRO

vi. Changes to the OGC® Abstract Specification

The encoding described in this Discussion Paper uses a property *samplingStrategy* for observations to link to sampling features independently of the featureOfInterest property, which should be used strictly for a feature with which the observed property is associated.

This encoding includes an Observation Collection class, to enable common properties of a collection of observations to be (optionally) associated with a container object, instead of repeated on every member observation.

This encoding adds some properties of individual sampling features to the Sampling Feature Collection class, to enable common properties of a collection of sampling features to be (optionally) associated with a container object, instead of repeated on every member sampling feature.

This encoding adds an optional property *samplingElevation* to the description of Specimens, to support a common use-case where the vertical offset from a 2-D location needs to be specified.

Changes to the UML model in OGC® Abstract Specification – Topic 20 should be considered to match these.

vii. Future work

Harmonization with JSON Schema implementation from 52-North.

Harmonization/adoption by SensorThings.

1. Scope

This Discussion Paper defines a JSON implementation of schemas for observations, and for features involved in sampling when making observations. This provides document models for the exchange of information describing observation acts and their results, both within and between different scientific and technical communities.

The implementation is derived from a conceptual model defined in *OGC Observations and Measurements v2.0* (also published as ISO/DIS 19156) with some modifications to support a more streamlined encoding of collections, and to improve the handling of sampling strategies. The conversion to JSON was done manually, but provides a data-point for potential standardization of a model→JSON conversion rule.

This document does not describe the full O&M information models or XML/GML encodings. The OGC/ISO standards (above) should be referred to for these details.

2. Conformance

2.1 Overview

The proposed standard described in this Discussion Paper defines seven requirements classes, covering the most commonly used Observation types (generic observation, measurement, category observation, count observation, truth observation, geometry observation, temporal observation, time-series observation) and all sampling feature types specified in Topic 20 except sampling solid.

Requirements for one standardization target type is considered:

- instances of observation data encoded in JSON

Since data *producing* applications should generate conformant data instances, the requirements and tests described in this standard also apply to this standardization target.

> NOTE: Requirements and tests for a second standardization target type (data consuming applications, i.e. data processing software that accepts observation data as input) are also highly desirable. However, a general solution to specifying this target this is more challenging and has been deferred.

2.2 Conformance classes

The framework, concepts, and methodology for testing, and the criteria to be achieved to claim conformance are specified in the OGC Compliance Testing Policies and Procedures and the OGC Compliance Testing web site1.

Annex A defines a set of tests and conformance classes that will support various applications with a range of different requirements. Seven conformance classes are

1 www.opengeospatial.org/cite

distinguished. Testing is based on data validation using the JSON Schema representation of O&M. In order to conform to this draft standard, an implementation shall choose to implement any one of the conformance classes specified in Annex A (normative).

Table 1 — Conformance classes related Observations and Measurements instances

Conformance class	Description	Clause
/conf/base	Base types and objects	A.2
/conf/geometry	Geometry types and objects	A.3
/conf/time-series	Time series content	A.4
/conf/sampling	Sampling feature data	A.5
/conf/specimen	Specimen data	A.6
/conf/spatial-sampling	Spatial sampling feature data	A.7
/conf/sampling-collection	Collections of sampling features	A.8
/conf/observation	Observation data	A.9
/conf/observation-collection	Collections of observations	A.10

3. Normative references

The following normative documents contain provisions that, through reference in this text, constitute provisions of 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.

ECMA-404, The JSON Data Interchange Format, (2013) 7pp. http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf (accessed August 19, 2015).

F. Galiegue, K. Zyp, G. Court, eds., JSON Schema: core definitions and terminology, (2013) 14pp. https://tools.ietf.org/html/draft-zyp-json-schema-04 (accessed August 21, 2015).

ISO 8601- Data elements and interchange formats – Information interchange – Representation of dates and times

OGC 08-131r3, The Specification Model — A Standard for Modular specifications, (2008). https://portal.opengeospatial.org/files/34762 (accessed August 21, 2015).

OGC Abstract Specification Topic 20 – OGC Observations and Measurements v2.0 OGC Document 10-004r1 http://www.opengis.net/doc/AS/Topic20 (also published as ISO/DIS 19156:2010, Geographic information — Observations and Measurements)

OGC Sensor Observation Service (SOS) v2.0 - OGC Document 12-006. http://www.opengis.net/doc/IS/SOS/2.0.

G. Schadow, C.J. McDonald, UCUM - The Unified Code for Units of Measure, (1998). http://unitsofmeasure.org/ucum.html (accessed July 12, 2015).

4. Terms and Definitions

This document uses the terms defined in Sub-clause 5.3 of [OGC 06-121r8], which is based on the ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this standard.

For the purposes of this document, the following additional terms and definitions apply.

4.1
GeoJSON
a geospatial data interchange format based on JavaScript Object Notation (JSON)

4.2
JSON
a lightweight, text-based, language-independent data interchange format, based on the Javascript programming language

4.3
JSON Schema
JSON document to describe a JSON data structure, providing for documentation and structural validation

5. Conventions

5.1 Abbreviated terms

JSON	Javascript Object Notation
O&M	Observations and Measurements
OM-JSON	Observations and Measurements JSON Implementation
OMXML	Observations and Measurements XML Implementation
OGC	Open Geospatial Consortium
SOS	Sensor Observation Service
SWE	Sensor Web Enablement
UML	Unified Modeling Language
XSD	W3C XML Schema Definition Language

5.2 Schema language

The JSON implementation specified in this draft standard is described using JSON Schema[2]. Version 4 of JSON Schema supports composition of a schema from elements defined in multiple documents, which we take advantage of here.

5.3 Layout and identifiers

This standard follows the structures defined in the OGC Policy [*The Specification Model – A Standard for Modular specifications*]. All normative material is organized as requirements, requirements classes, conformance tests and conformance classes. Each is identified with a URI, and the content and dependencies are described in tables whose structure matches the specification model.

The normative provisions for OM-JSON are denoted by the URI http://www.opengis.net/spec/om-json/1.0. All requirements and conformance tests that appear in this document are denoted by relative URIs which are relative to this base URI.

[2] http://json-schema.org/documentation.html

6. Overview

This draft standard contains requirements relating to a single standardization target: data documents. Specifically these are documents containing observation data conforming to the O&M model and encoded in JSON.

This is a draft implementation standard based on OGC Observations and Measurements (OGC Abstract Specification – Topic 20), complementing OMXML (OGC 10-025r1) which provides a GML-based XML encoding. OWL implementations have also been proposed [Cox 2013, 2015].

This draft standard provides a JSON encoding of O&M that is independent of a web service or API. This allows a common encoding to be used across services, leading to increased interoperability. JSON encodings of O&M have been specified in some existing services, including:

1. The Sensor Observation Service (SOS) implementation from 52North provides a JSON encoding, also described using JSON Schema[3].
2. The candidate OGC standard, SensorThings API, has a component that handles JSON encoding of observations.[4]
3. A CSIRO developed 'SensorCloud' has a partial implementation of an O&M JSON encoding.

A partial review of these has occurred; further harmonization is recommended.

6.1 Use of JSON

There is currently no formalised UML to JSON mapping rule for the GML UML profile. The engineering report "OWS-9: UML-to-GML-Application Schema (UGAS) Conversion Engineering Report" investigates some approaches. The ShapeChange tool[5] implements the rules outlined in the OWS-9 report. This tool was used within the WaterML2.0 part 2 Interoperability Experiment to generate JSON Schema, with some minor modifications[6].

The "Testbed 11 Implementing JSON/GeoJSON in an OGC Standard" provides recommendations for the use of JSON, JSON-LD and GeoJSON in OGC standards.

The schema follows a set of UML to JSON encoding rule as follows:

- An instance of a class is a JSON object. The value of a name/value pair with the name 'type'[7] maps to the UML class name.

[3] https://github.com/52North/SOS/tree/master/coding/json-common/src/main/resources/schema

[4] http://ogc-iot.github.io/ogc-iot-api/

[5] http://shapechange.net/targets/json/

[6] https://portal.opengeospatial.org/files/?artifact_id=61224

[7] This convention is taken from GeoJSON, and is also consistent with OMXML.

- Properties (attributes and association roles) are name/value pairs. The ‘name’ matches the UML attribute name or association role name
- Cardinality/obligation of properties are generally relaxed compared to the abstract/conceptual model. While values for all mandatory properties are required to be available, in some cases they are recorded against a container class (collection) rather than the atomic object.
- Values are JSON objects, except where the type corresponds with one of the built-in JSON types.
- A set of basic object types required for OM-JSON is described in the /req/base requirements class, and defined rigorously in the Common.json and Temporal.json JSON-Schema documents.
- GeoJSON geometry encoding is used, and defined in a JSON-Schema for geometry.

JSON does not have a formal class model - JSON Objects are just sets of properties. However, similar to GeoJSON, the JSON encoding described in this draft standard features a “type” property on each JSON object, to support explicit alignment with the underlying conceptual model. The value of this type is matched to O&M v2.0 class names is provided in Table 2.

Table 2 — Map of UML class names from O&M v2.0 to values for the type property in OM-JSON

Class in O&M v2.0	Type in OM-JSON	Class in O&M v2.0	Type in OM-JSON
OM_Observation	Observation	SF_SamplingFeature	SamplingFeature
OM_Measurement	Measurement	SF_Specimen	Specimen
OM_CategoryObservation	CategoryObservation	SF_SpatialSamplingFeature	SpatialSamplingFeature
OM_CountObservation	CountObservation	SF_SamplingPoint	SamplingPoint
OM_TruthObservation	TruthObservation	SF_SamplingCurve	SamplingCurve
OM_GeometryObservation	GeometryObservation	SF_SamplingSurface	SamplingSurface
OM_TemporalObservation	TemporalObservation	SF_SamplingSolid	Not implemented
OM_TimeSeriesObservation	TimeSeriesObservation		

6.2 Conformance with O&M model

OM-JSON satisfies the 'Conformance Classes related to Application Schemas including Observations and Measurements' listed in Clause 2.2 of *OGC Observations and Measurements v2.0*. The map from O&M v2.0 Conformance Classes to OM-JSON is given in Table 3.

Table 3 —Map of O&M v2.0 Conformance Classes to OM-JSON.

O&M v2.0 Conformance Class	OM-JSON Conformance Class
Sampling feature interchange	/conf/sampling
Specimen interchange	/conf/specimen
Spatial sampling feature interchange	/conf/spatial-sampling
Sampling point interchange	/conf/spatial-sampling[1]
Sampling curve interchange	/conf/spatial-sampling[1]
Sampling surface interchange	/conf/spatial-sampling[1]
Sampling solid interchange	/conf/spatial-sampling[1]
Generic observation interchange	/conf/observation
Measurement interchange	/conf/observation[2]
Category observation	/conf/observation[2]
Count observation	/conf/observation[2]
Truth observation	/conf/observation[2]
Geometry observation	/conf/observation[2]
Temporal observation	/conf/observation[2]
Category observation	/conf/observation[2]
Temporal coverage observation interchange	/conf/observation[2]

[1] The value of the “type” property, and the type of the “shape” property determine the spatial sampling feature type

[2] The value of the “type” property, and the type of the “result” property determine the observation type.

7. Requirements for JSON encoded instances of Observations

7.1 Introduction

An Observations and Measurements data document will include one or more of the JSON objects listed in Table 2. The basic requirements for data instances are, therefore, formalized in terms of these elements. The corresponding conformance tests use document validation using various combinations of schema documents.

Nine requirements classes are described in this clause. Figure 1 provides a graphical (informative) summary of the dependencies of the requirements classes.

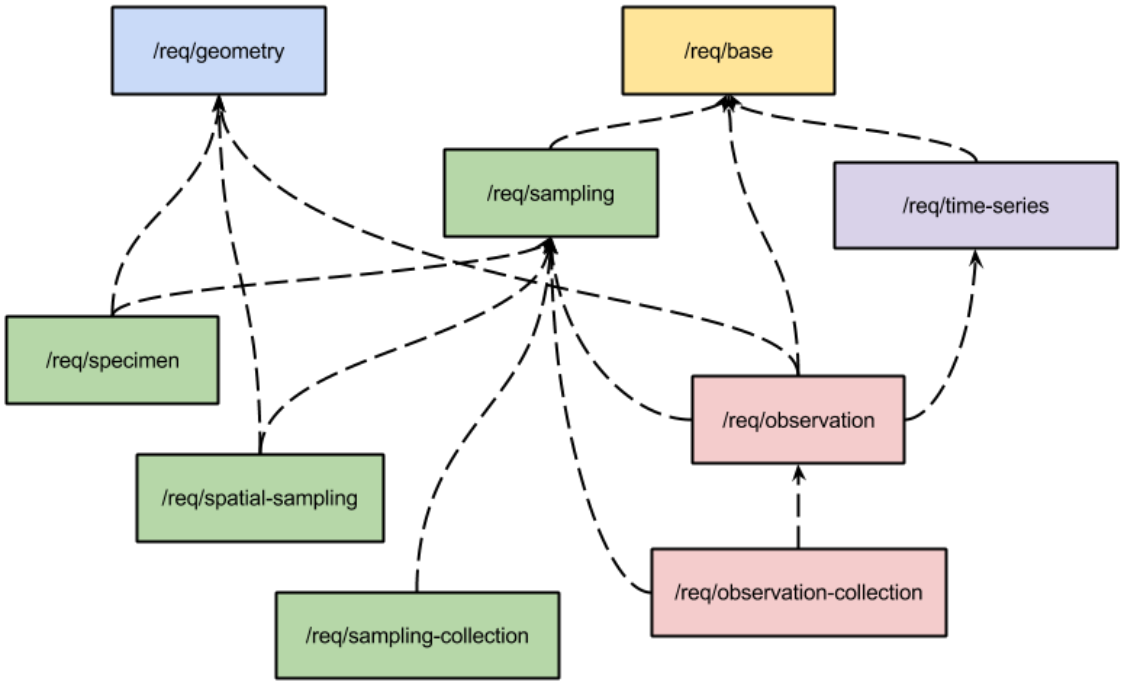

Figure 1. (informative) Dependencies of OM-JSON requirements classes.

7.2 Requirements class: JSON base types

This requirements class defines the base requirements for JSON encodings. It includes definitions of common types used in a number of applications.

NOTE: The JSON objects defined in this requirements class are not specific to Observations and Measurements applications. This requirements class may be suitable for use in JSON encodings for other geospatial applications.

Requirements Class	
/req/base	
Target type	Data instance
Dependency	**JSON**
Dependency	**GeoJSON**
Dependency	**XML Schema – Part 2**
Requirement	**/req/base/json**
Requirement	**/req/base/time-position**
Requirement	**/req/base/time-instant**
Requirement	**/req/base/time-interval**
Requirement	**/req/base/duration**
Requirement	**/req/base/measure**
Requirement	**/req/base/vocab-term**
Requirement	**/req/base/link**

The first requirement is that an OM-JSON document is a valid JSON document.

Requirement	**/req/base/json** A data instance shall be a conformant JSON document, as defined in ECMA-404

JSON has a limited range of built-in types (http://json.org/). The next seven requirements provide standard JSON representations of additional types required across all requirements within this specification.

ISO 8601 defines a standard 7-element encoding for time position expressed as a date in the Gregorian calendar and time on the 24-hour clock with timezone. A number of sub-types from ISO 8601 are described in W3C XML-Schema – Part 2 – Datatypes, which provides both an accessible definition and convenient identifiers for the individual types.

Temporal position is encoded as a text string matching a subset of the relevant XML Schema types.

Requirement	**/req/base/time-position** Each date-time position used in a data instance shall be encoded as a

	character string matching one of the following XML Schema types: http://www.w3.org/TR/xmlschema11-2/#dateTime http://www.w3.org/TR/xmlschema11-2/#date http://www.w3.org/TR/xmlschema11-2/#gYearMonth http://www.w3.org/TR/xmlschema11-2/#gYear.

Examples:

```
“2015-05-12T15:05:00.00+10:00”
“2015-05-12T05:05:00.00Z”
“2015-05-12”
“2015-05”
“2015”
```

A time instant is encoded as a JSON object with a single property whose value is a temporal position.

Requirement	**/req/base/time-instant** Each date-time position used in a data instance shall be encoded as a JSON object with a single property “instant” whose value is a temporal position.

Examples:

```
{   “instant”: “2015-05-12T15:05:00.00+10:00” }

{   “instant”: “2015-05-12T05:05:00.00Z” }

{   “instant”: “2015-05-12” }

{   “instant”: “2015-05” }

{   “instant”: “2015” }
```

A time interval is encoded as a JSON object with explicit ends.

Requirement	**/req/base/time-interval** Each date-time interval used in a data instance shall be encoded as a JSON object, with properties “begin” and “end”, whose value is a temporal position. An open-ended interval (i.e. in which an end is not specified) shall use the same JSON object, omitting the open end.

Examples:

```
{    “begin”: “2015-05-10” , “end”: “2015-05-15” }
```

```
{    “begin”: “1788-01-26” }

{    “end”: “1918-11-11” }
```

A time duration is encoded as a text string matching the relevant XML Schema type, or as a measure.

Requirement	**/req/base/duration** Each temporal duration used in a data instance shall be encoded either as a text string matching the XML Schema type http://www.w3.org/TR/xmlschema11-2/#duration, or as a measure object, encoded following /req/base/measure

Examples:

```
“P1Y3M16DT2H35M14.53S”

{
        “value”: 60,
        “uom”: “s”
}
```

A quantity value or measure is encoded as a JSON object with properties for the value and unit of measure.

Requirement	**/req/base/measure** A measure value (scaled number or quantity) used in a data instance shall be encoded as a JSON object containing an amount, denoted “value”, and an optional unit of measure, denoted “uom”. If present, the value of “uom” shall be a symbol from UCUM, or a URI denoting a unit-of-measure defined in a web resource.

Examples:

```
{
        “value”: 76.50 ,
        “uom”: “kg”
}

{
        “value”: 4.567 ,
        “uom”: “http://www.opengis.net/def/uom/ogc/0/My”
}
```

A value from a controlled vocabulary encoded as a JSON object with properties for the term and vocabulary.

Requirement	**/req/base/vocab-term** A controlled term used in a data instance shall be encoded as a JSON

	object containing the text value in a property “term”, and an optional source vocabulary in a property “vocabulary”. If present, the value of “vocabulary” shall be a URI denoting a controlled vocabulary.

Example:

```
{
        “term”: “Geochemistry” ,
        “vocabulary”: “http://ns.nature.com/subjects/”
}
```

A link object is provided for reference to an external web resource

Requirement	**/req/base/link** A hyperlink used in a data instance shall be encoded as a JSON object, with a property “href” carrying the URI of the external resource, an optional “rel” property providing the semantics of the reference, and an optional “title” property providing a human readable label for the reference.

Example:

```
{
        “title”: “OGC Observations and Measurements”,
        “href”: “http://www.opengeospatial.org/standards/om”,
        “rel”: “http://www.opengis.net/def/doc-type/as”
}
```

The vocab-term object is particularly useful where a vocabulary is published as a whole, but without separate URIs for each member. On the other hand some vocabularies are published with a URI for each vocabulary item, in which case a link object could be used instead of a vocab-term.

7.3 Requirements class: Geometry types

This requirements class defines the requirements for JSON encoding of geometry.

Requirements Class	
/req/geometry	
Target type	Data instance
Dependency	**JSON**
Dependency	**GeoJSON**
Dependency	**XML Schema – Part 2**
Requirement	**/req/geometry/geojson**

The encoding of geometry follows GeoJSON for Position, Point, MultiPoint, LineString, MultiLineString, Polygon, MultiPolygon, Geometry Collection.

Requirement	**/req/geometry/geojson**

	Objects in a data instance that describe 0-D, 1-D, or 2-D geometries with positions in the WGS84 system shall be encoded using the GeoJSON geometry encoding.

NOTE: GeoJSON supports 0-D, 1-D and 2-D geometries and collections, in a single Coordinate Reference System (CRS). If it is required to describe a geometry that falls outside the scope of GeoJSON, then another encoding must be used.

Examples:

```
{
    "type": "Point",
    "coordinates": [-170, 10]
}

{
    "type": "LineString",
    "coordinates": [
        [-170, 10],
        [170, 11]
    ]
}

{
    "type": "Polygon",
    "coordinates": [ [
            [-180.0, 10.0],
            [20.0, 90.0],
            [180.0, -5.0],
            [-30.0, -90.0],
            [-180.0, 10.0]
    ] ]
}
```

7.4 Requirements class: Time Series time-value-pair encoding

This requirements class defines a JSON encoding for time-series, organized as a sequence of time-value-pairs. The encoding is based on the current version of TimeseriesML, which is a draft OGC standard. This encoding should be updated when an endorsed standard is available. Time-series may be defined as a separate encoding specification.

Requirements Class	
/req/time-series	
Target type	Data instance
Dependency	**/req/base**
Requirement	**/req/time-series/series**
Requirement	**/req/time-series/metadata**
Requirement	**/req/time-series/tvp**
Requirement	**/req/time-series/tvp-metadata**

The first requirement defines the basic structure of time-value (tvp) encoding for time-series.

Requirement	/req/time-series/tvp-series The time-series shall implement the properties shown in Table 4.

Table 4 - time-value-pair time-series properties

Names	Definition	Data type and values	Obligation
id	unique identifier for the time-series	Character string.	Mandatory
points	array of time-value pairs	Time-value pairs as defined in Table 6.	Mandatory
metadata	descriptive data for the time-series object	An object that follows the time-series metadata, defined in Table .	Optional
defaultPointMetadata	the feature which the sampling feature was designed to sample	Default metadata for each point in the timeseries (can be overridden on a per-point basis). Implements properties from Table 6.	Optional

Example:

```
{
  "metadata": {
    "intendedObservationSpacing": "P1D",
    "status": {
      "term”: "checked",
      "vocabulary”: "http://www.example.org/vocabs"
    }
  },
  "defaultPointMetadata": {
    "interpolationType": {
      "term”: "Continuous",
      "vocabulary”: "http://www.opengis.net/def/waterml/2.0/interpolationType"
    },
    "quality": {
      "term”: "good",
      "vocabulary”: "http://www.opengis.net/def/waterml/2.0/quality"
    },
    "uom”: "http://www.qudt.org/qudt/owl/1.0.0/unit/Instances.html#Meter"
  },
  "points": [
    {
      "time": {
        "instant": "2010-01-01T00:00:00"
      },
      "value": 3.2
    },
    {
      "time": {
        "instant": "2010-01-02T00:00:00"
      },
      "value": 3.6
```

```
    }
  ]
}
```

Requirement	/req/time-series/metadata The time-series metadata type shall implement the properties shown in Table 5.

Table 5 - time-series metadata

Names	Definition	Data type and values	Obligation
temporalExtent	the time bounds of the time-series	Date-time interval (/req/base/time-interval)	Optional.
baseTime	the starting time for definition of a monotonic time-series	Date-time position (/req/base/time-position)	Optional
spacing	the time distance between monotonically increasing time-series points	Duration (/req/base/duration)	Optional (required if baseTime is defined)
commentBlocks	comments that span a time-period	Array of date-time position, string tuples.	Optional
intendedObservationSpacing	The intended temporal spacing of the time series points. This may vary from the actual spacing.	Duration (/req/base/duration)	Optional
status	Indicates the statues of the observation. E.g. unreleased, verified etc.	Vocab-term (/req/base/vocab-term)	Optional
cumulative	This boolean property indicates whether the series is sequentially increasing and accumulates over time.	Boolean	Optional

Names	Definition	Data type and values	Obligation
accumulationAnchorTime	Defines the time at which accumulation begins. e.g. 9am.	Date-time position (/req/base/time-position)	Optional
startAnchorPoint	Specifies a 'ghost' point to allow the first value of the time-series to be interpolated correctly.	Date-time position (/req/base/time-position)	Optional
endAnchorPoint	Specifies a 'ghost' point to allow the last value of the time-series to be interpolated correctly.	Date-time position (/req/base/time-position)	Optional
maxGapPeriod	Defines whether it is possible to interpolate between any two adjoining points. If the join period between two adjoining points is greater than the maxGapPeriod then the series should not be interpolated between these points.	Duration (/req/base/duration)	Optional

Example:

```
{
    "intendedObservationSpacing": "P1D",
    "commentBlocks": [
        {
            "applicablePeriod": {
                "begin": "2015-01-05T00:00:00",
                "end": "2015-01-25T00:00:00"
            },
            "comment": "Period of communication disruption"
        }
    ],
    "cumulative": true
}
```

Requirement	/req/time-series/tvp The time-value instances shall implement the properties defined in Table 6.

Table 6 - time-value pair properties

Names	Definition	Data type and values	Obligation
time	the time of the time-series point	Either a date-time position (/req/base/time-position) -or- date-time interval (/req/base/time-interval)	Mandatory
value	the value of the time-series point	Either a JSON number or a vocab term (/req/base/vocab-term)	Mandatory
metadata	descriptive data for the time-series point.	An object that implements the properties defined in Table .	Optional

Example:

```
{
    "time": {
        "instant": "2010-01-02T00:00:00"
    },
    "value": 3.6
}
```

Requirement	/req/time-series/tvp-metadata The time-value metadata shall implement the properties defined in Table 7.

Table 7 - time-value pair metadata properties

Names	Definition	Data type and values	Obligation
interpolationType	the interpolation type of the value	A vocab-term (/req/base/vocab-term). Example vocabularies: http://www.opengis.net/def/waterml/2.0/interpolationType/	Mandatory

Names	Definition	Data type and values	Obligation
quality	subjective classification of the quality of a single value	A vocab-term (/req/base/vocab-term).	Optional
uom	the unit of measure of the value	A vocab-term (/req/base/vocab-term).	Optional
nilReason	indicates the reason for nil value	Vocab-term (/req/base/vocab-term).	Optional
censoredReason	indicates the value is censored and the reason	Vocab-term (/req/base/vocab-term).	Optional
comment	free text comment associated with the point	String	Optional
accuracy	specifies the accuracy of the measurement	Measure (/req/base/measure)	Optional
aggregationDuration	duration over which the value has been aggregated	Duration (/req/base/duration)	Optional
qualifier	a vocab term that further qualifies the value	Vocab-term (/req/base/vocab-term).	Optional
processing	a vocab term indicating any processing that has occurred to the value	Vocab-term (/req/base/vocab-term).	Optional

Example (showing use of time-value pair and metadata):

```
{
    "time": {
        "instant": "2010-01-03T00:00:00"
    },
    "value": null,
    "metadata": {
        "nilReason": {
            "term": "missing",
            "vocabulary": "http://www.opengis.net/def/nil/OGC/0"
        }
    }
}
```

7.5 Requirements class: Sampling feature data

This requirements class defines a JSON encoding of the O&M Sampling feature.

Requirements Class	
/req/sampling	
Target type	Data instance
Dependency	**/req/base**

Requirement	/req/sampling/properties The sampling feature JSON object shall implement the properties shown in Table 8, with values matching the type shown, and with the obligation shown.

Table 8 – Sampling feature properties

Names	Definition	Data type and values	Obligation
id	unique identifier for the sampling feature	Character string.	Mandatory
type	type of the sampling feature. For spatial sampling features this is defined by the type of the shape property	A value from Table 2, column 4.	Mandatory
sampledFeature	the feature which the sampling feature was designed to sample	Array of: Links (each to an individual feature) /req/base/link	May be omitted **only** if member of a collection with a common sampled feature.
complex	related sampling features	Array of links /req/base/link; within each - value of "href" property identifies a related sampling feature - value of "rel" property indicates relationship of target to source.	Optional

The properties listed in Table 8 correspond with attributes and association-roles from the conceptual model for sampling features provided in O&M.

The next two requirements classes specify specialized sampling feature types. /req/sampling is a formal dependency of each of these, so an instance of any of these MUST include all the properties associated with a generic sampling feature, with additional properties as noted.

7.6 Requirements class: Specimen data

This requirements class defines a JSON encoding of the O&M Specimen. A specimen is a specialized sampling feature so carries all the properties of the parent type.

Requirements Class	
/req/specimen	
Target type	Data instance

Dependency	**/req/base**
Dependency	**/req/geometry**
Dependency	**/req/sampling**
Requirement	**/req/specimen/properties** In addition to the properties inherited from the model for sampling features, the specimen JSON object SHALL implement the properties shown in Table 9.

Table 9 – Specimen properties

Names	Definition	Data type and values	Obligation
samplingTime	the time the specimen was taken from the sampled feature	Time-instant /req/base/time-instant -or- time-interval /req/base/time-interval	Mandatory
samplingMethod	Process used to take the specimen from the sampled feature	Link /req/base/link (to a description of a sampling process)	Optional
samplingLocation	Location from which the specimen was taken within the sampled feature	Text description -or- Geometry /req/geometry/geojson -or- Link /req/base/link (to a description of a location)	Optional
samplingElevation	Elevation from where the specimen was taken at the given sampling location	JSON Object with mandatory property “elevation” whose value is a scaled number /req/base/measure, and optional property “verticalDatum” whose value is a link /req/base/link denoting a vertical data definition	Optional
currentLocation	Location where the specimen is currently	Text description -or- GeoJSON geometry /req/geometry/geojson -or- Link /req/base/link (to a description of a location)	Optional
size	Physical extent of the specimen (length, mass, etc)	Measure /req/base/measure	Optional

```
{
  "id": "spec1",
  "type": "Specimen",
  "sampledFeature": {
    "href": "http://example.org/featureA",
    "title": "feature A"
  },
  "complex": [
    { "rel": "http://example.org/parent", "href": "http://example.org/sample2" },
    { "rel": "http://example.org/child",  "href": "http://example.org/sample3" }
  ],
  "samplingTime": { "instant": "2015-05-14" },
  "samplingLocation": {
    "type": "Point",
    "coordinates": [  -30,  -90 ]
  },
  "currentLocation": {
    "href": "http://example.org/locations/store",
    "title": "bottom drawer"
  },
  "size": {
    "value": 23.1,
    "uom": "http://www.opengis.net/def/uom/ogc/0/m"
  }
}
```


7.7 Requirements class: Spatial sampling feature data

This requirements class defines a JSON encoding of the O&M Spatial sampling feature. A spatial sampling feature is a specialized sampling feature so carries all the properties of the parent type.

Requirements Class	
/req/spatial-sampling	
Target type	Data instance
Dependency	**/req/geometry**
Dependency	**/req/sampling**
Requirement	**/req/spatial-sampling/properties** In addition to the properties inherited from the model for sampling features, the spatial sampling feature JSON object SHALL implement the properties shown in Table 10.
Requirement	**/req/spatial-sampling/type** The value of the shape property shall be consistent with the sampling feature type if present, according to the map in Table 11.

Table 10– Spatial sampling feature properties

Names	Definition	Data type and values	Obligation
shape	Spatial location and extent of the sampling feature	GeoJSON geometry /req/geometry/geojson (recommended where possible)	Mandatory
hostedProcedure	Observation procedure (instrument, sensor, observer) associated with or hosted by this spatial sampling feature	Link /req/base/link (to a description of an observation process)	Optional

Table 11 – Map of sampling-feature type and shape value

Value of sampling feature "type"	GeoJSON object in "shape"
SamplingPoint	Point
SamplingCurve	LineString
SamplingSurface	Polygon
SamplingSolid	n/a

```
{
  "id": "sample1",
  "type": "SamplingSurface",
  "sampledFeature": {
    "href": "http://example.org/featureA",
    "title": "test feature A"
  },
  "complex": [
    { "rel": "http://example.org/child",  "href": "http://example.org/pixel3" }
  ],
  "shape": {
    "type": "Polygon",
    "coordinates": [ [
        [ -180,   10 ],
        [   20,   90 ],
        [  180,   -5 ],
        [  -30,  -90 ]
    ] ]
  },
  "hostedProcedure": [
    { "href": "http://example.org/sensor99" },
    { "href": "http://example.org/observer98" }
  ]
}
```

7.8 Requirements class: Sampling feature collection

This requirements class defines a JSON encoding of the O&M Sampling feature collection.

If the `sampledFeature` property is present, then its value is the default for all members of the collection.

Requirements Class	
/req/sampling-feature-collection	
Target type	Data instance
Dependency	**/req/base**
Requirement	**/req/sampling-feature-collection/properties** The sampling feature collection JSON object shall implement the properties shown in Table 12.

Table 12 – Sampling feature collection properties

Names	Definition	Data type and values	Obligation
id	A unique identifier for the sampling feature collection	A string identifier string.	Mandatory
sampledFeature	the feature which the sampling features in the collection were designed to sample	Array of: Links (each to an individual feature) /req/base/link	Optional
member	sampling feature that is a member of this collection	Array of: SamplingFeature object (or specialization) /req/sampling -or- Link /req/base/link (to a description of a sampling feature)	Mandatory

Example:

```
{
  "id": "sampling feature collection 77",
  "member": [
    { "href": "http://example.org/sample23" },
    {
      "id": "spec56",
      "type": "Specimen",
      "samplingTime": { "instant": "2015-05-14" },
      "samplingLocation": {
        "type": "Point",
        "coordinates": [  -30,  -90 ]
      },
      "size": {
        "value": 2,
        "uom": "l"
      }
    }
  ],
```

```
  "sampledFeature": [
    { "href": "http://example.org/feature45", "title": "Derwent River" }
  ]
}
```

7.9 Requirements class: Observation data

This requirements class defines a JSON encoding of the O&M Observation.

Requirements Class	
/req/observation	
Target type	Data instance
Dependency	**/req/base**
Dependency	**/req/geometry**
Dependency	**/req/sampling**
Requirement	**/req/observation/properties** An Observation JSON object shall have the properties shown in Table 13, with values matching the type shown, and with the obligation shown.
Requirement	**/req/observation/type** The value of the "result" property shall be consistent with the observation "type" if present, according to the map in Table 14.

Table 13 – Observation properties

Names	Definition	Data type and values	Obligation
id	A unique identifier for the observation	Character string	Mandatory
type	type of the observation, defined by the type of the result property	A value from Table 2, column 2.	Mandatory
context	Other observations that provide the context for this observation	Array of links; within each - value of "href" property identifies a related observation - value of "rel" property indicates relationship of target to source.	Optional
phenomenonTime	time at which the result of the observation is associated with the feature of interest. If the observation result is a time-series, the value of phenomenon-time is the	time-instant -or- time-interval	May be omitted only if member of a collection with a common phenomenon time.

Names	Definition	Data type and values	Obligation
	time-interval that bounds the result.		
observedProperty	a phenomenon associated with the feature-of-interest	Link (to a definition of an observable property)	May be omitted only if member of a collection with a common observed property.
procedure	procedure used in making observation. Typically a sensor or sensor-system, algorithm, computational procedure	Link (to a description of an observation procedure)	May be omitted only if member of a collection with a common procedure.
featureOfInterest	subject of the observation, whose type may carry the observed property	Link (to a description of a feature)	May be omitted only if member of a collection with a common feature of interest.
samplingStrategy	a strategy, such as a sampling feature, which mediates between the procedure and the ultimate feature of interest	Sampling Feature -or- Link (to a description of the sampling strategy e.g. a sampling-feature)	Optional
resultTime	time at which the result became available, after all processing steps were completed	time-position	May be omitted only if member of a collection with a common time of availability.
result	estimate of the value of the observed property result from application of the procedure	Object or literal; one of (link, measure, vocab-term, boolean (true/false), integer, temporal-object, geometry-object, time-series)	Mandatory

Table 14 – Map of observation type and result value

Value of observation "type"	Type of observation "result"
Observation	Any type, or link /req/base/link
Measurement	Measure - /req/base/measure
CategoryObservation	Vocab-term - /req/base/vocab-term
CountObservation	integer
TruthObservation	True or false
GeometryObservation	Geometry - /req/base/geometry
TemporalObservation	Time object – one of

	/req/base/time-instant /req/base/time-interval /req/base/duration
TimeseriesObservation	Time series TVP object - /req/time-series

The properties listed in Table 13 correspond with attributes and association-roles from the O&M conceptual model. Several of the properties that are mandatory in the model are optional in the JSON encoding. These are where the value of the property may be available from a related object or data structure, in particular where the observation is a member of a collection that is homogeneous in this particular property, so its value is assigned by inheritance from the container collection.

The other variation from the O&M conceptual model is the explicit `samplingStrategy` property. In O&M a sampling-feature is allowed as the value of the feature-of-interest property, with the ultimate feature of interest found through its 'sampled-feature' property. This approach was found to confuse some of the community. In this implementation, the feature-of-interest refers to the *ultimate* feature of interest which has a type that may carry the observed property, while the sampling-strategy may refer to the *proximate* sampling feature.

Examples:

```
{
  "id":"measure-instance-test",
  "type”: "Measurement",
  "phenomenonTime": { "instant":"2011-05-11T00:00:00+10:00" },
  "observedProperty": {"href":"http://environment.data.gov.au/def/property/air_temperature"},
  "procedure": {"href":"http://www.opengis.net/def/waterml/2.0/processType/Sensor"},
  "featureOfInterest": {"href":"http://waterml2.csiro.au/rgs-api/v1/monitoring-point/419009/"},
  "resultTime": "2011-05-12T09:00:00+10:00",
  "result": {
      "value": 3.2,
      "uom": "http://qudt.org/vocab/unit#DegreeCelsius"
    }
}

{
  "id":"text-observation-instance",
  "type": "CategoryObservation",
  "phenomenonTime": { "instant":"2015-05-11"},
  "observedProperty": {"href": "http://environment.data.gov.au/def/property/taxon"},
  "procedure": {"href":"http://www.opengis.net/def/waterml/2.0/processType/Manual"},
  "featureOfInterest": {"href":"http://example.org/sighting345"},
  "resultTime": "2015-05-12T09:00:00+10:00",
  "result": {
    "term": "johnstons-crocodile" ,
    "vocabulary": "http://environment.data.gov.au/def/object/"
  }
}

{
  "id": "linestring-example",
  "type": "GeometryObservation",
  "phenomenonTime": { "instant":"2011-05-11"},
  "observedProperty": { "href":"http://environment.data.gov.au/def/property/shape"},
  "procedure": { "href":"http://www.opengis.net/def/waterml/2.0/processType/Sensor"},
  "featureOfInterest": {
      "href":"http://environment.data.gov.au/water/id/catchment/100862",
```

```
      "title":"Observed line through catchment"
  },
  "resultTime": "2011-05-12T00:00:00+10:00",
  "result": {
      "type": "LineString",
      "coordinates": [ [ 4e6, -2e6 ], [ 8e6, 2e6 ] ]
  }
}
```

7.10 Requirements class: collections of observations

This requirements class defines a JSON encoding for collections of observations. These are typically homogeneous in at least one of the observation properties.

If a value for any of (`procedure`, `featureOfInterest`, `samplingStrategy`, `observedProperty`, `resultTime`) is present, then its value is the default for that property on observations that are members of the collection.

If a value for `uom` is present, then its value is the default for any measure (quantity value) that is the result on observations that are members of the collection.

If a value for `vocabulary` is present, then its value is the default for any vocabulary term that is the result on observations that are members of the collection.

Requirements Class	
/req/observation-collection	
Target type	Data instance
Dependency	**/req/base**
Dependency	**/req/sampling**
Dependency	**/req/observation**
Requirement	**/req/observation-collection/properties** An Observation Collection JSON object shall have the properties shown in Table 15, with values matching the type shown, and with the obligation shown.

Table 15 – Observation collection properties

Names	Definition	Data type and values	Obligation
id	unique identifier for the collection of observations	Character string	Mandatory
procedure	procedure used in making the observations. Typically a sensor or sensor-system, algorithm, computational procedure	Link (to a description of an observation procedure)	Optional
featureOfInterest	subject of the observations,	Link (to a description of a	Optional

Names	Definition	Data type and values	Obligation
	whose type may carry the observed property	feature)	
samplingStrategy	strategy, such as a sampling feature, which mediates between the procedure and the ultimate feature of interest	Sampling Feature -or- Link (to a description of the sampling strategy e.g. a sampling-feature)	Optional
observedProperty	phenomenon associated with the feature-of-interest	Link (to a definition of an observable property)	Optional
phenomenonTime	time at which the result of the observation is associated with the feature of interest.	time-instant -or- time-interval	Optional.
resultTime	time at which the results became available, after all processing steps were completed	time-position	Optional
uom	units of measure for quantitative values in the results of the observations	A symbol from UCUM, or a URI denoting a unit-of-measure defined in a web resource.	Optional
vocabulary	vocabulary from which all terms in the results of observations are taken	Link (to a vocabulary)	Optional
member	Observation in the collection	Array of: Observations -or- Links (each to an observation)	Mandatory

Example:

```
{
  "id": "observation collection a34",
  "phenomenonTime": { "instant":"2015-05-11" },
  "member": [
    { "href": "http://example.org/o23" },
    {
      "id":"o96",
      "type": "CategoryObservation",
      "observedProperty": { "href": "http://environment.data.gov.au/def/property/taxon" },
      "procedure": { "href":"http://www.opengis.net/def/waterml/2.0/processType/Manual" },
      "featureOfInterest": { "href":"http://example.org/sighting345" },
      "resultTime": "2015-05-12T09:00:00+10:00",
      "result": {
        "term": "johnstons-crocodile" ,
        "vocabulary": "http://environment.data.gov.au/def/object/"
      }
    }
  ]
}
```

8. Media Types for any data encoding(s)

When OM-JSON data is delivered using HTTP, the following MIME media-type shall be used in headers:

application/json

Annex A: Conformance Class Abstract Test Suite (Normative)

A.1 Introduction

Conformance is tested using a set of JSON Schema documents which formalize the requirements described above. Strictly, each object definition is a "JSON Schema", so a JSON schema document may include multiple, sometimes nested, "JSON Schemas", each providing the definition of one object. Using JSON References, JSON schema supports inclusion of schemas defined in a JSON schema document into new schemas defined in the same or another document, so standard JSON objects may be defined once and re-used in multiple contexts, and a set of related object definitions may be composed from multiple documents.

The OM-JSON schema is currently packaged in 11 JSON Schema documents. Two schema documents define generic datatypes, corresponding to /req/base, plus one schema document for geometry, corresponding to /req/geometry. Four schema documents define JSON objects for sampling features corresponding to /req/sampling, req/specimen, req/spatial-sampling, and req/sampling-collection. Four define JSON objects for observations, corresponding to /req/observation, /req/observation-collection, and /req/time-series. The dependencies of the OM-JSON schemas are shown in Figure 2.

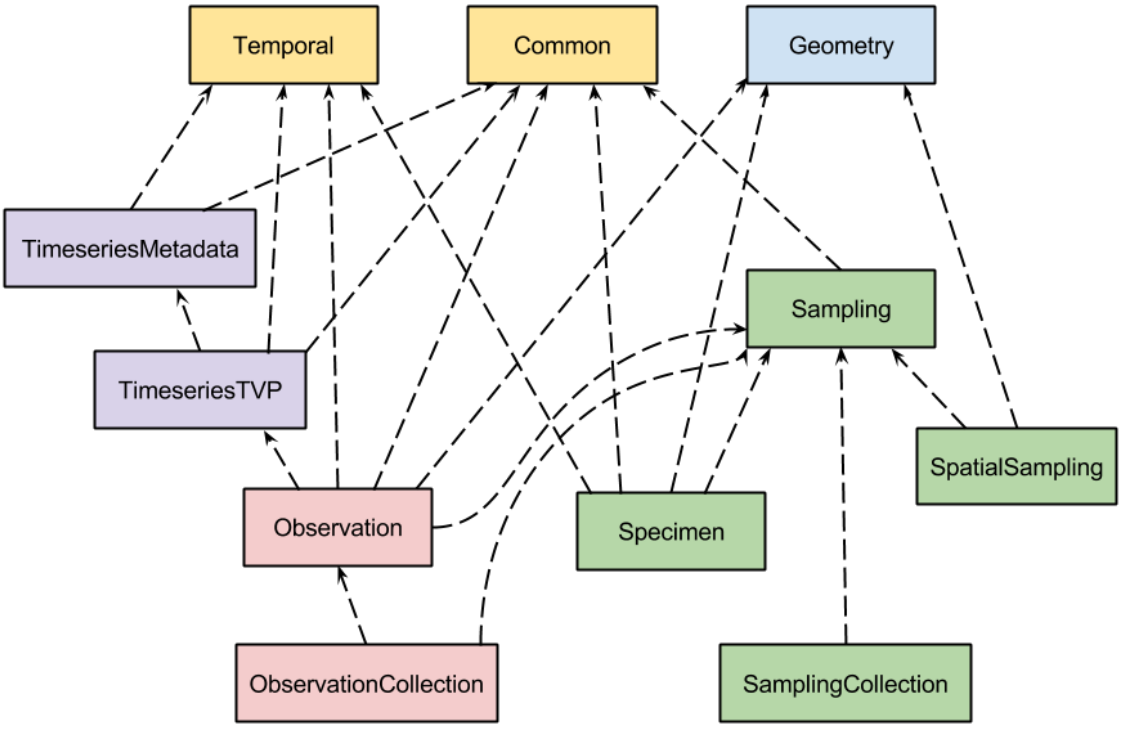

Figure 2. OM-JSON Schema dependencies. Arrows indicate dependencies, where an element from the dependency is included using a JSON Reference.

The OM-JSON schema documents are published at the web addresses indicated in Table 16.

Table 16 –

JSON Schema	Address of schema document PREFIX : <http://raw.githubusercontent.com/peterataylor/>[8]
Geometry objects	:om-json/master/Geometry.json
Temporal objects	:om-json/master/Temporal.json
Common types	:om-json/master/Common.json
Sampling features	:om-json/master/Sampling.json
Specimens	:om-json/master/Specimen.json
Spatial Sampling Features	:om-json/master/SpatialSampling.json
Collections of sampling features	:om-json/master/SamplingCollection.json
Observations	:om-json/master/Observation.json
Time-series metadata	:om-json/master/TimeseriesMetadata.json
Time-series data	:om-json/master/TimeseriesTVP.json
Collections of observations	:om-json/master/ObservationCollection.json

A.2 Conformance class: base types

This conformance class tests that occurrences of the base types (including geometry and temporal objects) are encoded according to the requirements.

The tests require first that an 'application' JSON Schema is constructed that includes one or more definitions of JSON types and objects from the base JSON Schema documents. Then that a document is valid according to this schema.

[8] This prefix is the development location of the JSON Schema documents, and is provided to enable testing. Upon publication by OGC the JSON schema documents may be published through a more formal location.

Conformance Class	/conf/base	
Requirements	/req/base	
Dependency	A JSON Schema Validator	
Test	/conf/base/json	
	Requirement	/req/base/json
	Test purpose	Verify that the document is well-formed JSON.
	Test method	Load the document in a JSON validator[9]. Pass if no errors reported. Fail otherwise.
	Test type	Capability
Test	/conf/base/temporal	
	Requirement	/req/base/time-position , /req/base/time-instant , /req/base/time-interval , /req/base/duration
	Test purpose	Verify that the time object is conforms to the relevant time encoding (e.g. ISO8601).
	Test method	Validate the JSON instance document using the appropriate object definition from the Temporal.json JSON Schema. Pass if no errors reported. Fail otherwise.
	Test type	Capability
Test	/conf/base/types	
	Requirement	/req/base/measure , /req/base/vocab-term , /req/base/link
	Test purpose	Verify the base types are encoded using the specified property names and structures.
	Test method	Validate the JSON instance document using the appropriate object definition from the Common.json JSON Schema. Pass if no errors reported. Fail otherwise.
	Test type	Capability

A.3 Conformance class: geometry

This conformance class tests that occurrences of the base types (including geometry and temporal objects) are encoded according to the requirements.

The tests require first that an 'application' JSON Schema is constructed that includes one or more definitions of JSON types and objects from the base JSON Schema documents. Then test that a document is valid according to this schema.

[9] E.g. http://jsonlint.com

Conformance Class	/conf/geometry	
Requirements	/req/geometry	
Dependency	A JSON Schema Validator	
Test	/conf/geometry/geojson	
	Requirement	/req/geometry/geojson
	Test purpose	Verify that any objects within the document with a “type” property that matches one of the types from GeoJSON conforms to the GeoJSON format.
	Test method	Validate the JSON instance document using the appropriate object definition from the Geometry.json JSON Schema. Pass if no errors reported. Fail otherwise.
	Test type	Capability

A.4 Conformance class: time series

This conformance class first requires that an ‘application’ JSON Schema is constructed that includes one or more definitions of JSON types and objects from the base JSON Schema documents.

Conformance Class	/conf/time-series	
Requirements	/req/time-series	
Dependency	A JSON Schema Validator	
Test	/conf/time-series/series	
	Requirement	/req/time-series/series
	Test purpose	Verify that the JSON instance document is a valid time-series object.
	Test method	Validate the JSON instance document using the appropriate object definition from the TimeseriesTVP.json JSON Schema. Pass if no errors reported. Fail otherwise.
	Test type	Capability

A.5 Conformance class: sampling features

This conformance class first requires that an ‘application’ JSON Schema is constructed that includes one or more definitions of JSON types and objects from the base JSON Schema documents.

Conformance Class	/conf/sampling

Requirements	/req/sampling	
Dependency	A JSON Schema Validator	
Test	/conf/sampling/properties	
	Requirement	/req/base/json
	Test purpose	Verify that the JSON instance document is a valid Sampling object.
	Test method	Validate the JSON instance document using the appropriate object definition from the Sampling.json JSON Schema. Pass if no errors reported. Fail otherwise.
	Test type	Capability

A.6 Conformance class: specimen features

This conformance class first requires that an 'application' JSON Schema is constructed that includes one or more definitions of JSON types and objects from the base JSON Schema documents.

Conformance Class	**/conf/specimen**	
Requirements	/req/specimen	
Dependency	A JSON Schema Validator	
Test	/conf/specimen/properties	
	Requirement	/req/specimen/properties
	Test purpose	Verify that the JSON instance document is a valid Specimen object.
	Test method	Validate the JSON instance document using the appropriate object definition from the Specimen.json JSON Schema. Pass if no errors reported. Fail otherwise.
	Test type	Capability

A.7 Conformance class: spatial sampling features

This conformance class first requires that an 'application' JSON Schema is constructed that includes one or more definitions of JSON types and objects from the base JSON Schema documents.

Conformance Class	**/conf/spatial-sampling**
Requirements	/req/spatial-sampling

Dependency	A JSON Schema Validator	
Test	/conf/spatial-sampling/properties	
	Requirement	/req/specimen/properties
	Test purpose	Verify that the JSON instance document is a valid Spatial Sampling object.
	Test method	Validate the JSON instance document using the appropriate object definition from the SpatialSampling.json JSON Schema. Pass if no errors reported. Fail otherwise.
	Test type	Capability
Test	/conf/spatial-sampling/type	
	Requirement	/req/specimen/type
	Test purpose	Verify that the shape property is a valid GeoJSON object type.
	Test method	Inspect the 'type' property within the 'shape' property and ensure the string value matches the mapping specified in Table 7. Pass if it matches; fail if it does not.
	Test type	Capability

A.8 Conformance class: sampling feature collection

This conformance class first requires that an 'application' JSON Schema is constructed that includes one or more definitions of JSON types and objects from the base JSON Schema documents.

Conformance Class	**/conf/spatial-sampling**	
Requirements	/req/sampling-feature-collection	
Dependency	A JSON Schema Validator	
Test	/conf/sampling-feature-collection/properties	
	Requirement	/req/sampling-feature-collection/properties
	Test purpose	Verify that the JSON instance document is a valid Sampling Feature Collection.
	Test method	Validate the JSON instance document using the appropriate object definition from the SamplingFeatureCollection.json JSON Schema. Pass if no errors reported. Fail otherwise.
	Test type	Capability

A.9 Conformance class: observations

This conformance class first requires that an 'application' JSON Schema is constructed that includes one or more definitions of JSON types and objects from the base JSON Schema documents.

Conformance Class	**/conf/observation**	
Requirements	/req/observation	
Dependency	A JSON Schema Validator	
Test	/conf/observation/properties	
	Requirement	/req/observation/properties
	Test purpose	Verify that the JSON instance document is a valid Observation object.
	Test method	Validate the JSON instance document using the appropriate object definition from the Observation.json JSON Schema. Pass if no errors reported. Fail otherwise.
	Test type	Capability
Test	/conf/observation/type	
	Requirement	/req/observation/type
	Test purpose	Verify that the *result* property value is consistent with the *type* property. .
	Test method	Inspect the type property of the Observation object. Pass if the value is consistent with the mappings in Table 10.
	Test type	Capability

A.10 Conformance class: collections of observations

This conformance class first requires that an 'application' JSON Schema is constructed that includes one or more definitions of JSON types and objects from the base JSON Schema documents.

Conformance Class	**/conf/observation-collection**	
Requirements	/req/observation-collection	
Dependency	A JSON Schema Validator	
Test	/conf/observation-collection/properties	
	Requirement	/req/observation-collection/properties
	Test purpose	Verify that the JSON instance document is a valid Observation Collection object.

	Test method	Validate the JSON instance document using the appropriate object definition from the ObservationCollection.json JSON Schema. Pass if no errors reported. Fail otherwise.
	Test type	Capability

Annex 9: Revision history

Date	Release	Author	Paragraph modified	Description
2015-08-27	0.1	Simon Cox, Peter Taylor	All	First complete document
2015-09-15	0.2	Simon Cox	7.1, 7.2, 7.3, 7.6, 7.7, 7.9, A.3	1. Refactored geometry classes out from base 2. Added samplingElevation 3. generalised geometry encoding requirement

Annex B: Bibliography

H. Butler, M. Daly, A. Doyle, S. Gillies, T. Schaub, S. Hagen, eds., The GeoJSON Format, (2015) 19pp. https://datatracker.ietf.org/doc/draft-butler-geojson/ (accessed August 19, 2015).

S.J.D. Cox, An explicit OWL representation of ISO/OGC Observations and Measurements, in: O. Corcho, C. Henson, P. Barnaghi (Eds.), Proc. 6th Int. Work. Semant. Sens. Networks Co-Located with 12th Int. Semant. Web Conf. (ISWC 2013), Sun SITE Central Europe, Sydney, Australia, October 22nd, 2013., 2013: pp. 1–18. http://ceur-ws.org/Vol-1063/paper1.pdf (accessed February 3, 2014).

S.J.D. Cox, Ontology for observations and sampling features, with alignments to existing models, Semant. Web J. (2015) Submitted. http://www.semantic-web-journal.net/content/ontology-observations-and-sampling-features-alignments-existing-models (accessed July 24, 2015).

Taylor P. WaterML2.0 part 2 - RESTful API and JSON encoding. OGC public discussion paper (accepted, in process of publication). OGC 15-033

Taylor P (ed.). WaterML2.0 part 2 – rating tables, gauging observations and cross-sections: Interoperability Experiment Results. OGC public engineering report. https://portal.opengeospatial.org/files/?artifact_id=61224

OGC 10-025r1, Observations and Measurements - XML Implementation, (2011) 66 + x. http://portal.opengeospatial.org/files/41510 (accessed September 16, 2014).