Observation JSON Format

Présentation

16/04/2022

Environmental Sensing

Table of Contents

[1 Introduction 2](#_Toc101042540)

[1.1 Conventions used 2](#_Toc101042541)

[1.2 Terminology 2](#_Toc101042542)

[1.3 Rules 2](#_Toc101042543)

[2 Environmental Sensing – Observation 3](#_Toc101042544)

[3 Data Model 4](#_Toc101042545)

[4 ObsJSON objects 5](#_Toc101042546)

[4.1 ESAtt 5](#_Toc101042547)

[4.2 ESObservation 5](#_Toc101042548)

[4.3 ESType 6](#_Toc101042549)

[4.4 ESId 6](#_Toc101042550)

[4.5 ESIdxref 7](#_Toc101042551)

[4.6 ESOrder 7](#_Toc101042552)

[4.7 ESFeature 8](#_Toc101042553)

[4.8 ESValue 8](#_Toc101042554)

[4.8.1 LocationValue 9](#_Toc101042555)

[4.8.2 DatationValue 10](#_Toc101042556)

[4.8.3 PropertyValue 10](#_Toc101042557)

[4.8.4 ResultValue 11](#_Toc101042558)

[4.9 ESIndex 12](#_Toc101042559)

[4.10 ESData 12](#_Toc101042560)

[4.10.1 ESInformation 12](#_Toc101042561)

[4.10.2 ESParameter 13](#_Toc101042562)

[4.10.3 ESUserData 14](#_Toc101042563)

[5 Appendix : reserved values 15](#_Toc101042564)

[6 Appendix : Examples 16](#_Toc101042565)

# Introduction

ObsJSON is a text format for the ES-Observation data.

This format is an application of the JSON format (RFC 8259), GeoJSON format (RFC 7946), Date and Time format (RFC 3339).

A binary version is also defined (Appendix) with CBOR format (RFC 8949)

## Conventions used

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The grammatical rules in this document are to be interpreted as described in [RFC5234].

## Terminology

The terms Json-Text, Json-Value (Value), Object, Member, Array, Number, String, False, Null, True are define in the JSON grammar.

The terms Geometry-type, Point, MultiPoint, LineString, MultiLineString, Polygon, MultiPolygon, GeometryCollection, GeoJSON-Types are defined in GeoJSON grammar.

Timestamp is defined in Date and Time format.

## Rules

A Value in an ESValue SHOULD be unambiguous (i.e., parsers CAN deduce the Value type).

An ObsJSON-Text CAN contains all the ESObservation information (i.e., the ESObservation build from the ObsJSON-Text is identical to the initial ESObservation).

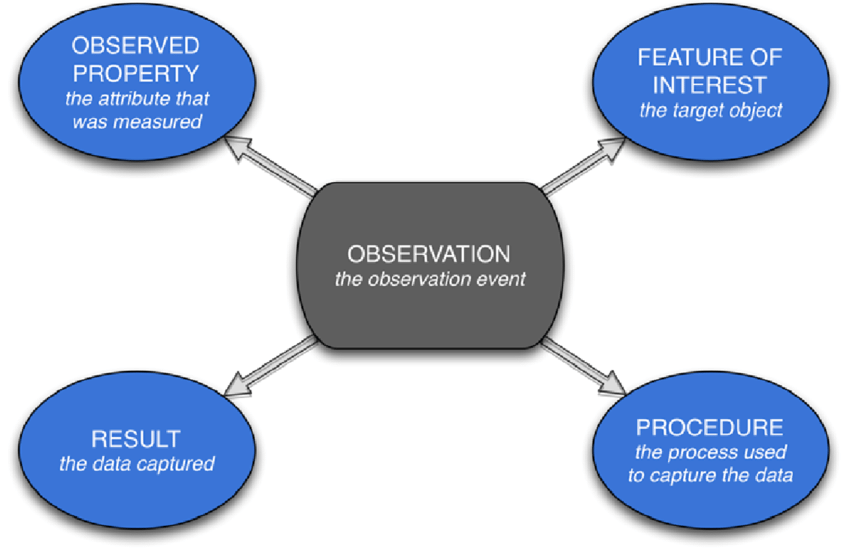
Values in Array are ordered and independent from the other Values.

Elements in Objects are not ordered.

# Environmental Sensing – Observation

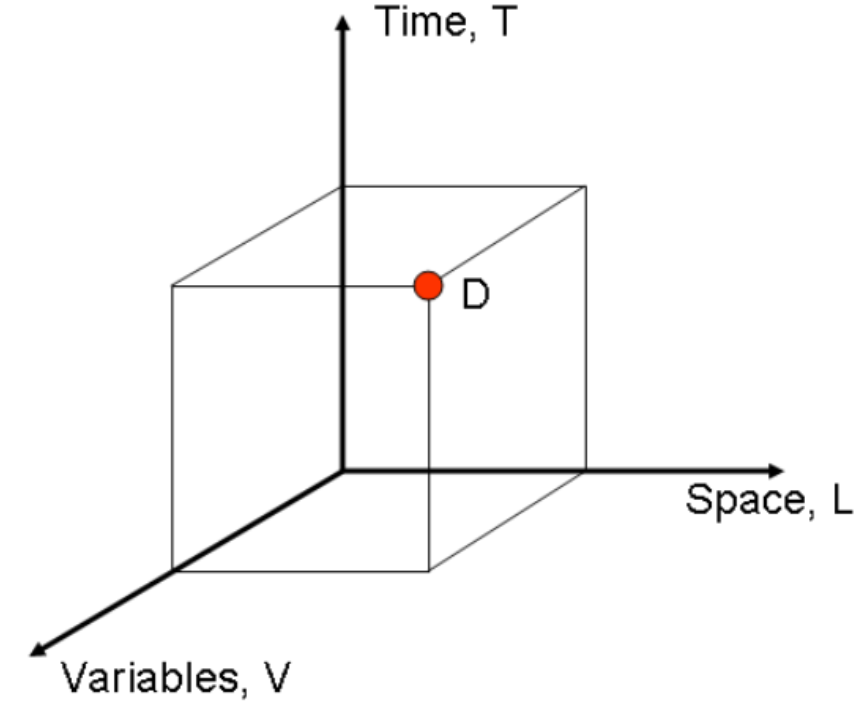
The concept of "Observation" is defined in the ISO19156 Standard. It allows to represent for example:

* Unit data from sensors,
* Modeling results,
* Geographical distributions,
* Temporal or trip histories,

In this Standard, an Observation is characterized by:

* "Observed property": the observed property,
* "Feature of interest": the object (usually a place) of the observation,
* "Procedure": the information acquisition mode (sensor, model, etc.)
* "result": result of the observation or measurement

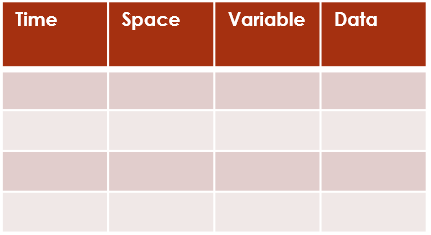
The result is a set of values referenced according to the 3 dimensions:



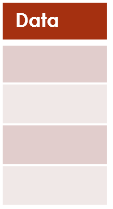
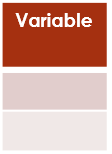
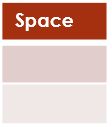
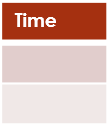
*CUAHSI Community Observations Data Model*

* Temporal,
* Spatial,
* Physical (observed property)

It can be converted into a 3-dimensional matrix, with each result indexed by temporal, spatial and physical values.



*interleaved*



*domain-range*

*Note: This “domain range” indexed representation is preferred to an “interleaved” tabular representation which associates temporal, spatial and physical values with each result value.*

Common properties (flags) are associated with each Observation. They make it possible to perform processing on the Observations without having to know their composition (e.g., bounding boxes, type of observation, volumetry, etc.).

*Note: If the indexed representation is identical to the original, the “domain range” representation is equivalent to the “interleaved” tabular representation.*

# Data Model

An Observation is composed of (domain range representation):

* A list of results data
* A list of datation data
* A list of location data
* A list of property data
* Several lists of variable data
* A link between result value and datation / location / property / variable value (indexing)

*Example:*



The representation of data is optimized (no duplication).

The indexing of the results can be done implicitly (ordered data) or explicitly with the rank of each index.

*Note: If the data is not complete, implicit indexing is not available*

# ObsJSON objects

An ObsJSON -Text is a JSON-text and consists of a single object ObsJSON

ObsJSON objects define in this document are:

* ESObservation,
* ESAtt (key /value Member),
* ESOrder,
* ESIdxref,
* ESFeature (ESDatation, ESLocation, ESProperty, ESResult, ESVariable),
* ESValue (DatationValue, LocationValue, PropertyValue, ResultValue),
* ESIndex,
* ESData (ESInformation, ESParameter, ESUserData),

## ESAtt

ESAtt are terminal key/value Members. The value SHALL not be another ObsJSON object.

Two kinds of ESAtt are defined:

* Defined ESAtt where the Key and the format of the Value are specified,
* User ESAtt where both Key and Value are unconstrained.

User ESAtt MAY be included in ESParameter or ESUserData Object.

## ESObservation

**Description**

ESObservation is an Object. The Members included SHALL be:

* ESAtt
* ESIdxref
* ESOrder
* ESFeature,
* ESIndex
* ESData

The Members are:

|  |  |  |  |
| --- | --- | --- | --- |
| Class | Member | Key | Value |
| ESAtt | ESName | « name » | String |
| ESAtt | ESType | « type » | « observation » |
| ESAtt | ESId | « id » | String |
| ESIdxref | ESIdxref | « idxref » | Object |
| ESOrder | ESOrder | « order » | Array |
| ESFeature | ESDatation | « datation » | Array or DatationValue |
| ESFeature | ESLocation | « location » | Array or LocationValue |
| ESFeature | ESProperty | « property » | Array or PropertyValue |
| ESFeature | ESResult | « result » | Array or ResultValue |
| ESFeature | ESVariable | String | Array or Object |
| ESIndex | ESIndex | « index » | Array or ESIdx |
| ESData | ESInformation | « information » | Object |
| ESData | ESParameter | « parameter » | Object |
| ESData | ESUserData | String | Object |

**Validity**

An ESObservation is valid if:

* it contains at least the ESType Member,
* it contains at most one ESDatation, one ESLocation, one ESProperty, one ESResult Member
* each Member is valid.

**Example**

{“type”: “observation”, “datation”: “morning”, “location”: “paris”, “property”: “air quality”, “result”: “good”}

{“type”: “observation”, “datation”: “2021-01-05T22:18:26”, “location”: [2.4, 48.9], “property”: {“prp”: “PM10”, “unit”: “µg/m3”}, “result”: 51.3}

## ESAtt

### ESType

**Description**

ESType is a single key/value Element :

* Key: “type”
* Value: “observation”

**Validity**

ESType is valid if key and value are as defined.

### ESId

**Description**

ESId is a single key/value Element:

* Key: “id”

The value is a string (e.g., Database Id or filename)

**Validity**

ESId is valid if key and value are as defined.

### ESName

**Description**

ESName is a single key/value Element:

* Key: “name”

The value is a string.

**Validity**

ESName is valid if key and value are as defined.

## ESIdxref

**Description**

ESIdxref is an Object. The members are key/value and represent coupled ESFeature:

* Key: ESFeature key
* Value: ESFeature key

If no one ESFeature is coupled, ESIdxref is not present.

**Validity**

ESIdxref is valid if at least one member is present and if ESFeature keys are present in ESObservation.

**Example**

“idxref”: {“datation”: “location”} *datation and location are coupled*

## ESDtype

**Description**

ESDtype is an Object. The members are key/value and represent the format of ESVariable data:

* Key: ESVariable key
* Value: ESValue used for the ESVariable

If ESDtype is not present, the default value ResultValue is used.

**Validity**

ESDtype is valid if ESFeature keys are present in ESObservation.

**Example**

“dtype”: {“campaign”: “datationvalue”} *the campaign is defined by datationvalue*

## ESOrder

**Description**

ESOrder is a String Array. The strings in the Array are the ESFeature keys ordered according to indexing.

If ESOrder is not present, the default value [“datation”, “location”, “property”, “first ESVariable key”] is used.

**Validity**

ESOrder is valid if:

* The values are present in ESFeature list,
* The length of the array equals the number of ESFeature.

**Example**

“order”: [“datation”, “property”, “location”}

## ESFeature

**Description**

ESFeature is an Array. Value is specific for each ESFeature:

|  |  |  |
| --- | --- | --- |
| Member | Key | Value |
| ESDatation | « datation » | Array of DatationValue |
| ESLocation | « location » | Array of LocationValue |
| ESProperty | « property » | Array of PropertyValue |
| ESResult | « result » | Array of ResultValue |
| ESVariable | String | Array of value |

*Note:*

*If the Array contains only one value, the square brackets MAY be omitted in the JSON String.*

*The value of ESVariable is unconstrained.*

**Validity**

An ESFeature Member is valid if:

* it contains at least one value
* each value is valid

**Example**

“location”: “paris” *one ESValue*

“location”: [[2.4, 48.9], [4.8, 45.8], [5.4, 43.3]] *three ESValue*

“location”: [“paris”, “lyon”, “marseille”] *three ESValue*

## ESValue

**Description**

An ESValue contains two information: A Name, a Value

One of the three formats SHALL be used for ESValue (where Name is a String):

* Object format: {Name: Value}
* Value format: Value
* Name format: Name

**Validity**

An ESValue Member is valid if:

* One of the three formats is used,
* it contains at least a Value or a Name
* each Value is valid compared to ESFeature

*Note:*

*The Name string MAY be used to represent:*

* *detailed information (e.g., “beginning of the observation”),*
* *link to external information (e.g., “*[*https://loco-philippe.github.io/ES.html*](https://loco-philippe.github.io/ES.html)*”),*
* *id to link internal information (e.g., “res003” where “res003” is a key in a ESData Object),*

**Example**

“morning” *Name format*

{“morning”: “2021-01-05T10:00:00”} *Object format*

[[“2021-01-05T08:00:00”, “2021-01-05T12:00:00”]] *Value format*

### LocationValue

**Description**

The Value of a LocationValue is a representation of a Point or a Polygon. It is defined by a Coordinates Array (as specified in GeoJSON).

**Validity**

A Value is valid if the Coordinates Array is valid and represents a Point or a Polygon.

**Value example**

[2.4, 48.9] *Point*

[[[2.4, 48.9], [4.8, 45.8], [5.4, 43.3], [2.4, 48.9]]] *Polygon*

[[[0,0], [0,5], [5,5], [0,0]], [[1,1], [1,2], [2,2], [1,1]]] *Polygon with a hole*

*Note:*

*The other Geometry-type are not allowed because the Coordinates Array is ambiguous:*

* *LineString, Multipoint and Array of Point have the same representation*
* *Polygon and Array of LineString have the same representation*
* *MultiPolygon and Array of Polygon have the same representation*

*The LineString in a Polygon MAY be open (without the last Point)*

### DatationValue

**Description**

A Date is defined by a Timestamp (as specified in RFC 3339).

The Value of a DatationValue is a representation of a single Date or a Slot (MultiInterval):

* Date: String
* Slot: Array of one or multiple Interval (an Interval is an Array of two Date)

**Validity**

A Value is valid if the Date or Slot is valid.

**Value example**

“2021-01-05T10:00:00” *Date*

[[“2021-01-05T08:00:00”, “2021-01-05T12:00:00”]] *Interval*

[[“2021-01-05”, “2021-01-10”], [“2021-01-20”, “2021-01-25”]] *Slot*

*Note:*

*Intervals MUST be represented by a Slot to avoid ambiguities with an array of Dates*

*If the DatationValue consists of a unique String, and if the String represents a Date, the parser SHALL assign the String to the Date, otherwise to the Name.*

### PropertyValue

**Description**

The Value of a PropertyValue is an Object made up of ESAtt. The Defined ESAtt are:

|  |  |  |
| --- | --- | --- |
| Member | Key | Value |
| PropertyType | « prp » | String (mandatory) |
| Unit | « unit » | String |
| SamplingFunction | « sampling » | String |
| Application | « application » | String |
| SensorType | « sensor » | String |
| UpperValue | « uppervalue » | Float |
| LowerValue | « lowervalue » | Float |
| Period | « period » | Float |
| UpdateInterval | « updateinterval » | Float |
| Uncertainty | « uncertainty » | Float |

*Note:*

*If the PropertyValue consists of a single Member (the PropertyType), it’s not allowed to replace the Object by a string (the PropertyType value).*

*If the ESAtt PropertyDict is not defined, the default PropertyDict is used*

**Validity**

A Value is valid if it contains at least the PropertyType ESAtt.

The PropertyType value MAY be present in a propertyDict how’s define the Unit value

**Example**

{“prp”: “Temp”} *Minimal Value*

{“prp”: “Temp”, “unit”: “°c”} *Defined Member*

{“prp”: “Temp”, “unit”: “°c”, “operation”: “phase 1”} *User Member*

*Note:*

*UserAtt MAY be used in the PropertyValue*

*For PropertyValue, the Name format is allowed (i.e., if the PropertyValue consists of a single string, this SHOULD be interpreted as the name).*

### ResultValue

**Description**

The Value of a ResultValue CAN be any JSON Object.

*Note:*

*For ResultValue, the Name format is not allowed (i.e., if the ResultValue consists of a single string, this SHOULD be interpreted as a Value).*

**Validity**

A Value is valid if it contains at least one Result.

**Example**

“

21.8 *Value format*

{“low temperature”: 2.4}  *Object format*

“https://loco-philippe.github.io/ES.html” *Value format*

*[21.8, {“test”: true}] Value format*

*Note:*

*If the ResultValue is composed by a unique String, the parser SHALL assign the String to the Result.*

## ESIndex

**Description**

ESIndex is an “indexed” ESResult Array. It’s a two-dimensional array following the order defined in ESOrder:

* One row for each ESFeature,
* One column for each ResultValue

The values in Array are integer.

*Note:*

*If the ESObservation contains only one ESFeature, the square brackets for row MAY be omitted.*

*If the ESObservation contains only one ResultValue, the square brackets for column MAY be omitted.*

*If only one level of square brackets is present, the parser must decide if columns or rows are present*

**Validity**

An ESIndex Value is valid if:

* it contains at least one integer value
* the number of rows equals the number of ESFeature
* the number of columns equals the number of ResultValue
* the integer values are positive and lower than the length of ESFeature

**Example**

“index”: [[0,2,1], [0,0,0]] two ESFeature, three ResultValue

“index”: [0,2,1] one ESFeature, three ResultValue (or the opposite)

## ESData

ESData are elements where the Value MAY be an Object.

The Object is made up of Defined ESAtt and User ESAtt.

### ESInformation

**Description**

The Defined ESAtt are:

|  |  |  |
| --- | --- | --- |
| Member | Key | Value |
| ObservationType | « typeobs » | String |
| LocationType | « typeloc » | String |
| DatationType | « typedat » | String |
| PropertyType | « typeprp » | String |
| ResultType | « typeres » | String |
| nValLocation | « nvalloc » | Integer |
| nValDatation | « nvaldat » | Integer |
| nValProperty | « nvalprp » | Integer |
| nValResult | « nvalres » | Integer |
| BoundingBox | « bbox » | Array (4 Float) |
| IntervalBox | « tbox » | Array (2 String) |
| Complet | « complet » | True / false |
| Score | « score » | Integer |
| Rate | « rate » | Float |
| Dimension | « dimension » | Integer |
| Axes | « axes » | Array (1 to 3 integers) |

**Validity**

An ESInformation is valid if the Value contains at least one ESAtt.

All the ESAtt are optional.

**Example**

{"typeobs": "areaObsrecord"} *Minimal Value*

{"typeobs": "areaObsrecord", "complet": false, "score": 226} *Defined Member*

*Note:*

*That information come from the other ESObervation elements.*

*A parser MAY ignore The ESInformation to build an ESObservation (User ESAtt are lost).*

### ESParameter

**Description**

The Defined ESAtt are:

|  |  |  |
| --- | --- | --- |
| Member | Key | Value |
| Reference | « reference » | String |
| ResultTime | « resulttime » | Timestamp |
| PropertyDict | « pdict » | String |
| UniqueIndex | « unicindex » | True/false |

**Validity**

An ESParameter is valid if the Value contains at least one ESAtt.

All the ESAtt are optional.

**Example**

{“unicindex”: true, “approbation”: true}

### ESUserData

The structure of ESUserData is totally free.

It MUST not contain any Defined ESAtt.

The keys used in ESUserData MUST be different from those defined in the Reserved list name (see Appendix.)

# Appendix : reserved values

|  |
| --- |
| « type » |
| « id » |
| « datation » |
| « location » |
| « property » |
| « result » |
| « information » |
| « parameter » |
| « observation » |
| « prp » |
| « unit » |
| « sampling » |
| « application » |
| « sensor » |
| « uppervalue » |
| « lowervalue » |
| « period » |
| « updateinterval » |
| « uncertainty » |
| « typeobs » |
| « typeloc » |
| « typedat » |
| « typeprp » |
| « typeres » |
| « nvalloc » |
| « nvaldat » |
| « nvalprp » |
| « nvalres » |
| « bbox » |
| « tbox » |
| « complet » |
| « score » |
| « rate » |
| « dimension » |
| « axes » |
| « reference » |
| « resulttime » |
| « order » |
| « propdict » |
| « unicindex » |

# Appendix : Examples

* {“type”: “observation”, “datation”: “morning”, “location”: “paris”, “property”: “air quality”, “result”: “good”}
* {“type”: “observation”, “datation”: “2021-01-05T22:18:26”, “location”: [2.4, 48.9], “property”: {“prp”: “PM10”}, “result”: 51.3}
* {“type”: “observation”, “datation”: [“2021-01-05T22:18:26”, “2021-01-05T22:18:26”], “property”: [“air quality PM10”, “air quality PM2.5”], “result”: [10.2, 21.5, 51.3, 48]}
* {“type”: “observation”, “name”: “example4”, “id”: “example4.obs”,

“parameter”: {“pdict”: “official”, “example”:4},

“datation”:

[“2021-01-04T10:00:00”,[[“2021-01-05T08:00:00”,“2021-01-05T12:00:00”]]],

“location”:

[[2.4, 48.9], [[[2.4, 48.9], [4.8, 45.8], [5.4, 43.3], [2.4, 48.9]]]],

“property”:

[{“prp”: “PM10”, “unit”: “µg/m3”}, {“prp”: “Temp”, “unit”: “°c”}]

“result”: [51.3, {“low temperature”: 2.4}, 20.8, “high temperature”]

“idxref”: {“datation”: “location”}}

*Note: another solution is to include index instead of idxref:*

“index”: [[0,0,1,1], [0,0,1,1], [0,1,0,1]]

# Appendix : CBOR format

The Concise Binary Object Representation (CBOR – RFC8949) is a data format whose design goals include the possibility of extremely small code size, small message size, and extensibility without the need for version negotiation.

CBOR is based on the JSON data model: numbers, strings, arrays, maps (called objects in JSON), and a few values such as false, true, and null.

The CBOR format can be used with different options to minimize length:

* The precision of float values is adjustable from half precision (two bytes) to double precision (eight bytes),
* The datetime can be described by a standard text string (RFC3339) or by a numerical value (Epoch-based: six bytes).
* The ESFeature name can be represented with code value instead of string value
* The coordinates value can be described with integer instead of float (val\_int = round(val\_float)\*10\*\*7 : for bytes).

**Example :**

*{"type": "observation",*

*"datation":["2021-01-04T10:00:00",[["2021-01-05T08:00:00","2021-01-5T12:00:00"]]],*

*"location":[[2.4123456, 48.9123456], [[[2.4123456, 48.9123456], [4.8123456, 45.8123456], [5.4123456, 43.3123456], [2.4123456, 48.9123456]]]],*

*"property": [{"prp": "PM10"}, {"prp": "Temp"}],*

*"result": [51.348, {"low": 2.457}, 20.88, "high"],*

*"idxref": {"datation": "location"}}*

Length (bytes):

* JSON: 388
* CBOR: 298
* CBOR optimized: 123