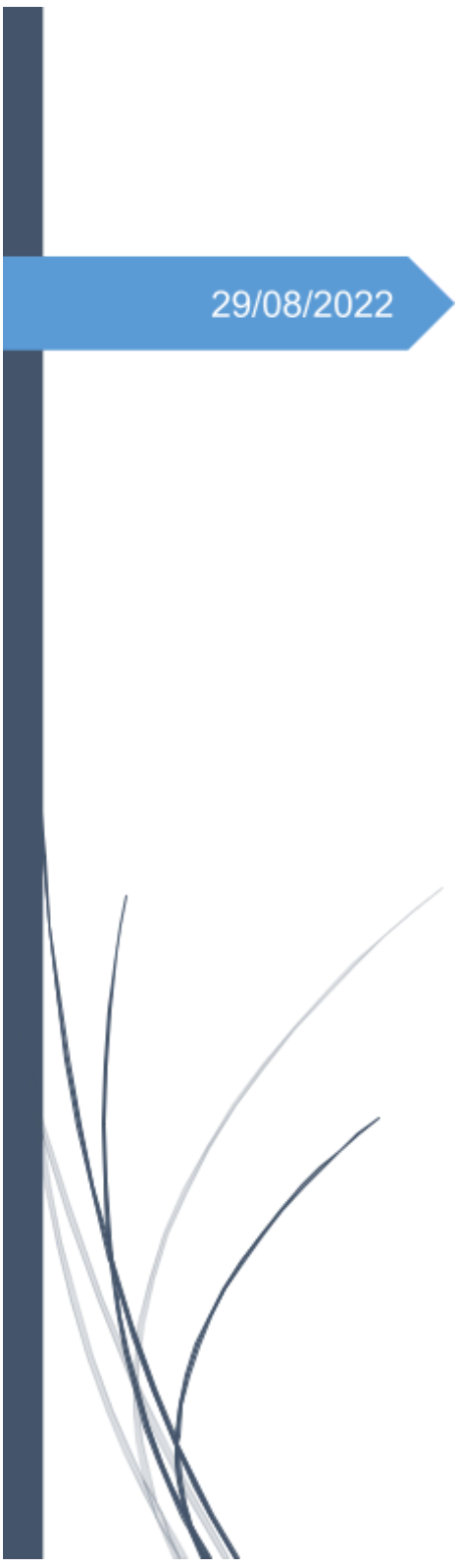
Ilist JSON Format

Présentation

29/08/2022



Environmental Sensing

TABLE OF CONTENTS

1 Introduction **3**

1.1 Conventions used 3

1.2 Terminology 3

1.3 Rules 3

2 Structure **4**

3 IlistJSON **5**

4 IindexJSON **6**

4.1 ContextElement 6

4.2 CodecElement 6

4.3 KeysElement 7

5 Examples **8**

Appendix : reserved values **9**

Appendix : CBOR format **10**

Appendix : Codec values **12**

1 INTRODUCTION

IlistJSON is a text format for the Indexed-list data (Ilist object).

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)

1.1 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].

1.2 TERMINOLOGY

The terms Json-Text, Json-Value (Value), Object, Member, Element, Array, Number, String, False, Null, True are defined 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.

1.3 RULES

An Ilist Json-Text SHOULD be unambiguous (i.e., parsers CAN deduce the Ilist object).

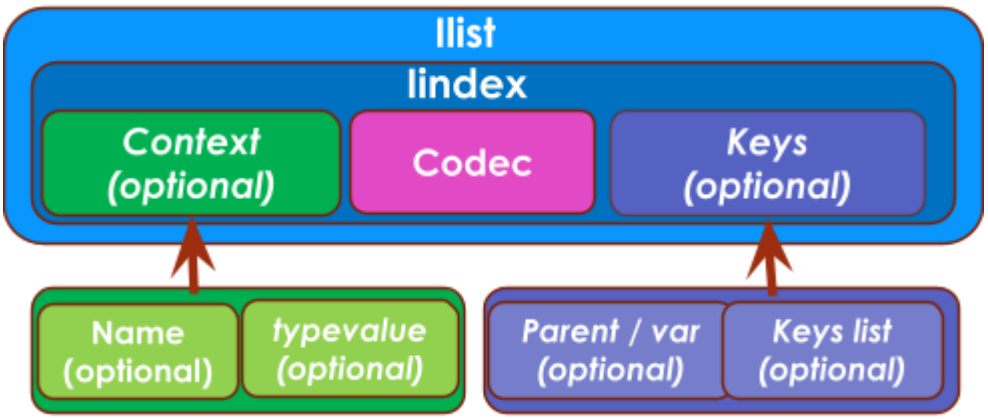
Values in Array are ordered and independent from the other Values.

Members in Objects are not ordered.

2 STRUCTURE

An Ilist Object is composed of Iindex Objects.

The figure below shows the Json structure of Ilist objects.



Ilist:	JSON Array (or JSON value if only one value)
Iindex:	JSON Array (or JSON value if only one value)
Context:	JSON Object (or JSON string if only one value)
Codec:	JSON Array (or JSON value if only one value)
Keys:	JSON Array (or JSON value if only one value)
Name, Typevalue:	string
keys list:	JSON Array
Parent / variable:	integer (-1 for variable)

3 IlistJSON

Description

The Ilist object consists of a list of Iindex objects.

Thus, an IlistJSON is an Array where Elements are IindexJSON.

Validity

An IlistJSON is valid if:

- It is an Array,
- The Array Elements are valid IindexJSON

Note:

The square brackets are mandatory even if the IlistJSON is empty or contains one Element.

4 IINDEXJSON

Description

The Iindex object is defined by attributes :

- Typevalue (string)
- Name (string),
- Codec (list)
- Keys (list)

Two informations from Ilist object are also associated to Iindex :

- parent or variable (integer)

The IindexJSON is an Array with three Elements :

- ContextElement (optional)
- CodecElement (mandatory)
- KeysElement (optional)

Validity

An IindexJSON is valid if:

- It is an Array,
- It contains at least the CodecElement
- each Element is valid

Note:

> *If the IindexJSON Array contains a single Element (CodecElement) even empty, the square brackets MAY be omitted in the JSON String.*

4.1 CONTEXTELEMENT

The ContextElement is composed with Name attribute and Typevalue attribute.

It has several formats :

- { Name : Typevalue } : Json-Object, if Name and Typevalue are present
- Name : Json-String, if Typevalue is not present
- Typevalue : Json-String, if Name is not present

If both Name and Typevalue aren't present, the ContextElement is not present in the IindexJSON.

4.2 CODECELEMENT

The Ilist Codec object consists of a list of values.

Thus, a CodecElement is an Array where Elements are Json-values.

If the Codec list is empty, the CodecElement is empty

Note:

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

4.3 KeysElement

The KeysElement is composed with Parent-Variable attribute (integer) and Keys attribute (list of integer).

It has several formats :

- [Parent-Variable, Keys]: Json-Array, if Parent-Variable is present and if Keys is present and not empty
- Parent-Variable: Json-number, if Keys is not present or empty
- Keys: Json-Array, if Parent-Variable is not present

If both Parent-Variable and Keys aren't present, the KeysElement is not present in the IindexJSON

5 EXAMPLES

Ilist :

[ ]	*Empty Ilist*
[25] or [[25]]	*Ilist with 1 Iindex with 1 codec value*
[2, 1] or [[2], [1]] or [2, [1]]	*Ilist with 2 Iindex with 1 codec value*
[[2, 1]]	*Ilist with 1 Iindex with 2 codec values*
[[2, 1], [4,3]]	*Ilist with 2 Iindex with 2 codec values*

Iindex :

[“age”, 25]	*Iindex with name and codec*
[“measure”, [2.4, 48.9]]	*Iindex with name and codec*
[[2.4, 48.9], [0, 0, 1, 1]]	*Iindex with codec and Keys*
[“measure”, [2.4, 48.9], [0, 0, 1, 1]]	*Iindex with name, codec and Keys*
[“locvalue”, [2.4, 48.9]]	*Iindex with typevalue and codec*
[{“locvalue”: “measure”}, [2.4, 48.9]]	*Iindex with typevalue, name and codec*
[{“locvalue”: “measure”}, [2.4, 8], [0,[0,0,1,1]]]	*Iindex with all data*

APPENDIX : RESERVED VALUES

to complete

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 TypeValue can be represented with a code value instead of string value
- The coordinates value can be described with integer instead of float (val_int = round(val_float)*10**7 : four bytes).

Example (Json format):

{"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"],

"coupled": {"datation": "location"}}

Example optimized (Cbor format):

{0 : [0,1,2],

1 : [[dt(2021, 1, 4, 10),[[dt(2021,1,5,8), dt(2021, 1, 5, 12]]],

[[2.4123456, 48.9123456], [[[2.4123456, 48.9123456], [4.8123456, 45.8123456], [5.4123456, 43.3123456], [2.4123456, 48.9123456]]]],

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

2 : [51.34375, {"low": 2.45703125}, 20.875, "high"],

3 : {0: 1} }

With :

- *Observation key codification:*

    - *0: "order"*
    - *1: "features"*
    - *2: "result"*
    - *3: "coupled"*
- *Order and coupled value codification:*
    - *0: "datation"*
    - *1: "location"*
    - *2: "property"*
- *Datation value: timestamp format*
- *Location value: integer representation (four bytes)*
- *Result value: half precision (two bytes)*

Length (bytes):

- JSON: 388
- CBOR: 298
- CBOR optimized: 133

APPENDIX : CODEC VALUES

The correspondence between value and Json-value depends on the Typevalue associated with the Iindex :

Object creation

\ Typevalue value	**None**	**ObjectValue**	**"ESValue"**	**ESValue**
TypedValue	Simple (value)	ObjectValue (value)	NamedValue (value)	ESValue (value)
Json-value				
{EStype: TypedValue}	EStype (TypedValue)			
{Objecttype: TypedValue}	Objecttype (TypedValue)		ExternValue (value)	

Json creation

\ Typevalue Objectvalue	**None**	**"ESValue"**	**ESValue**	**ObjectValue**
Simple	Json	JsonES (ESValue calculated)	JsonES	JsonObject
Object	JsonObject (Untyped)			JsonObject (Typed if same Object, Untyped else)
ES	JsonES(Typed if same ESValue, Untyped else)			