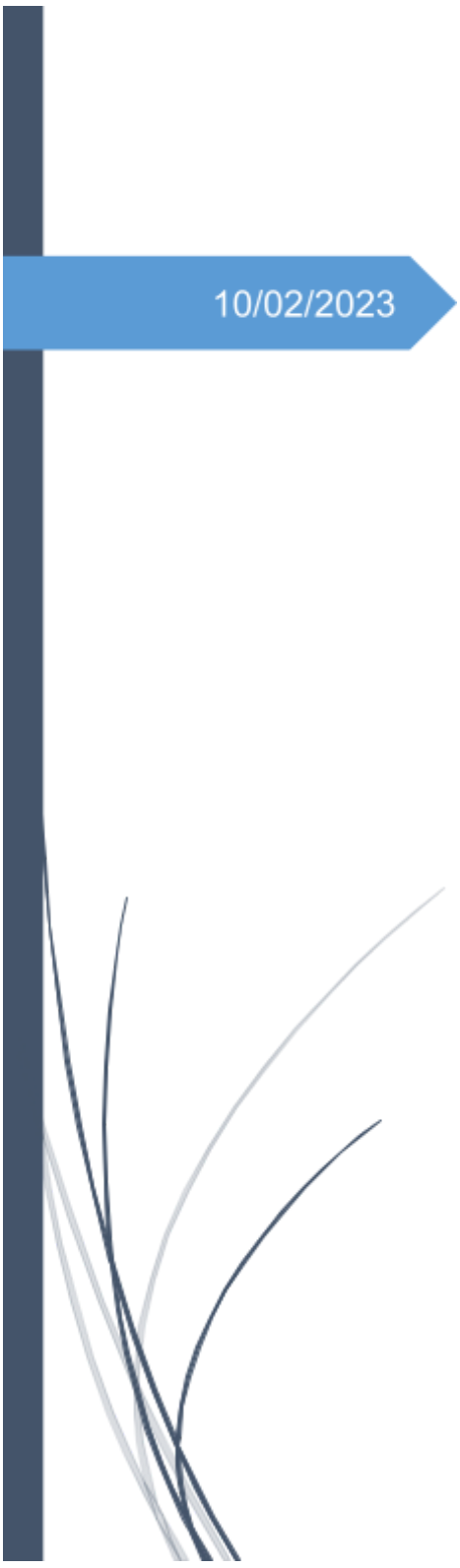
JSON-NTV Format

Presentation

10/02/2023



Environmental Sensing

Table of Contents

1. Abstract

This document describes a set of simple rules for unambiguously and concisely encoding data-names and data-types into JSON Data Interchange Format (RFC 8259).

These rules and framework, called JSON-NTV (JSON with Named and Typed Values), relies on the rules defined in the JSON-ND project (https://github.com/glenkleidon/JSON-ND).

2 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].

3 Terminology

The terms

> **JSON-type, JSON-text, JSON-name, JSON-value, JSON-object, JSON-member, JSON-element, JSON-array, JSON-number, JSON-string, JSON-false, JSON-null, JSON-true**

are defined as

> type, text, name, value, object, member, element, array, number, string, false, null, true

in the JSON grammar.

The **JSON-primitive** term represents a JSON-number, JSON-string, JSON-false, JSON-true or JSON-null.

The **JSON-unnamed** term represents a JSON-object without a single member.

The **JSON-named** term represents a JSON-object with a single member.

4 Introduction

Today, the semantic level of shared data remains low. It is very often limited to the type of data defined in the exchange formats (strings for CSV formats; numbers, strings, arrays and objects for JSON formats).

JSON-NTV proposes to increase the semantic level of the JSON entities by adding two OPTIONAL pieces of information to a JSON entity :

- name: interpretation of the value in human language or detailed information (e.g., "beginning of the observation") or link to external information(e.g., "https://github.com/loco-philippe/Environmental-Sensing/tree/main"),
- type: interpretation of the value in a data standard (e.g. GeoJSON, datetime) or in a data catalog or in a software language.

The NTV entity is thus a triplet with a mandatory element (value) and two OPTIONAL elements (name, type).

For example, Paris location can be represented by :

- *a name : "paris",*
- *a type : geoJSON Point coordinates,*
- *a value : [2.3522, 48.8566]*

The easiest way to add those informations is to use a JSON-object with a single member using the syntax JSON-ND for the first term of the member and the JSON-value for the second term of the member.

The JSON value of the previous example is:

{ "paris:point" : [2.3522, 48.8566] }

With this approach, three NTV entities are defined :

- a primitive entity which is not composed of any other entity,
- two structured entities: an unordered collection of NTV entities and an ordered sequence of NTV entities.

as well as two JSON formats depending on the presence of the OPTIONAL elements :

- simple format when name and type are not present
- named format when name or type is present

Example (entity composed of two other primitive entities):

{ "cities::point": [[2.3522, 48.8566], [4.8357, 45.7640]] }

{"cities::point": {"paris":[2.3522, 48.8566], "lyon":[4.8357, 45.7640]}}

A JSON-NTV generator produces a JSON-value from a NTV entity and vice versa a JSON-NTV parser transforms a JSON-value into a NTV entity.

The conversion between NTV entity and software object is not the subject of this note.

5 NTV STRUCTURE

5.1 NTV ENTITIES

The NTV entity is a triplet (NTVvalue, NTVtype, NTVname):

- the NTVvalue is the JSON representation of the NTV entity
- the NTVtype defines the conversion rules between entity and NTVvalue
- the NTVname is a string

The triplet contains all the data needed to reconstruct the NTV entity.

NTVtype and NTVname are OPTIONAL and have None as default value.

Three categories of entities (one primitive and two structured) are defined:

- NTV-single for the primitive entity,
- NTV-set for an unordered collection of NTV entities
- NTV-list for an ordered sequence of NTV entities

Abbreviations list:

- *NV-single, NV-list, NV-set : entity without NTVtype*
- *TV-single, TV-list, TV-set : entity without NTVname*
- *V-single, V-list, V-set : entity without NTVtype and without NTVname*

5.2 NTVTYPE

The NTVtype is defined in a Namespace. Namespaces may be nested.
A Namespace is represented by a string followed by a point.
The global Namespace is represented by an empty string.
The Namespace representations are added to the NTVtype to have an absolute representation of an NTVtype.

Example for a representation of an NTVtype defined in two nested Namespace :
"ns1.ns2.type"
where:
ns1. is a Namespace defined in the global Namespace,
ns2. is a Namespace defined in the ns1. Namespace,
type is a NTVtype defined in the ns2. Namespace

Example for a NVTtype defined in the global Namespace :
"type"
where:
type is a NTVtype defined in the global Namespace

The NTVtype for NTV-single entities defines the type of entity and the conversion rules between the NTV entity and the NTVvalue.

The NTVtype for structured entities is a Namespace or a default NTVtype to apply to the NTV entities included. This NTVtype avoids having to include a type (if default NTVtype) or reduce the length (if Namespace) in the JSON representation of the included NTV entities.

Three categories of NTVtype are defined (None, Simple, Generic).

- If NTVtype is None, the conversion rules are the JSON conversion rules.
- If NTVtype is Simple, the rules associated with the NTVtype are used for the conversion between an entity and a NTVvalue.
- The Generic NTVtype is equivalent to a set of Simple NTVtype. It can be converted into a Simple NTVtype when the NTVvalue is decoded.

  Examples :

  If a JSON-value is { "::dat" : ["2022-01-28T18-23-54", 123456.78] }, the parsers deduce that the NTVvalue of included entities has a "dat" NTVtype. The parser of included NTV entities deduces that the first object has an equivalent "datetime" NTVtype and the second an equivalent "timeposix" NTVtype.

  If JSON-value is { "::dat" : ["2022-01-28T18-23-54", {":point": [1, 2]] }, the parsers deduce that the first NTVvalue has a "dat" NTVtype and the second a "point" NTVtype.

This structuring of type makes it possible to reference any type of data that has a JSON representation and to consolidate all the shared data structures within the same tree of types.

5.3 NTVVALUE

The NTVvalue for an NTV-single entity is the JSON-value of the NTV entity.

The NTVvalue for an NTV-list or an NTV-set entity is the list of included NTV entities.

6 JSON-NTV REPRESENTATION

6.1 JsonNTVtype

The JsonNTVtype is identical to the NTVtype for the NTV entities not included in another entity.

For NTV entities included in another entity, the JsonNTVtype MAY be set to :

- None if the NTVtype of the structured NTV entity is identical (Simple type) or associated (Generic type) to the NTVtype of the NTV entity,
- relative NTVtype if the NTVtype of the structured NTV entity is a Namespace shared with the NTVtype of the NTV entity.
- absolut NTVtype in the other cases.

6.2 JsonName

For NTV-single, JsonName is :

- NTVname:JsonNTVtype *(if NTVname and JsonNTVtype are present),*
- NTVname *(if JsonNTVtype is None),*
- :JsonNTVtype *(if NTVname is None),*

For NTV-set or NTV-list, JsonName is :

- NTVname::JsonNTVtype *(if NTVname and JsonNTVtype are present),*
- NTVname *(if JsonNTVtype is None),*
- ::JsonNTVtype *(if NTVname is None),*

If the JsonName contains one colon, the entity is a NTV-single.

If the JsonName contains two adjacent colons, the entity is an NTV-set or an NTV-list.

6.3 JsonValue

For an NTV-single, the JsonValue is the NTVvalue.

For an NTV-list, the JsonValue is a JSON-array where JSON-elements are the JSON-NTV formats of included NTV entities.

For an NTV-set, the JsonValue is a JSON-object where the JSON-members are the JSON-members of the JSON-NTV formats of included NTV entities.

Note:

A JsonValue MUST NOT be a JSON-object with a single member.

NTV entities included in an NTV-set MUST have a JsonName and all the JsonName MUST be different *(e.g. {":point" : [2.3522, 48.8566], ":point" : [4.8357, 45.7640]} is not a valid JSON-value)*

6.4 JSON-NTV FORMAT

The JSON-NTV format is the JSON representation of an NTV entity. The JSON-NTV format is built with the NTVname, NTVvalue and the JsonNTVtype.

Two JSON-NTV formats are defined:

- named format (*if NTVname or JsonNTVtype are present)*:
  - { JsonName : JsonValue }
- simple format *(if NTVname and JsonNTVtype are None)*:
  - JsonValue

where:

- JsonName is a string representing a combination of JsonNTVtype and NTVname
- JsonValue is the NTVvalue.

This format allows full compatibility with existing JSON structures:

- a JSON-number, JSON-string, JSON-null or JSON-boolean is the representation of an V-single entity,
- a JSON-object with a single member is the representation of an NTV entity with NTVname or NTVtype
- a JSON-array is the representation of an V-list entity,
- a JSON-object without a single member is the representation of a V-set entity.

7 EXAMPLES

simple JSON-NTV format of a NTV-single entity :

"lyon"

52.5

{ }

named JSON-NTV format of a NTV-single entity:

{ "paris:point" : [2.3522, 48.8566] }

{ ":point" : [4.8357, 45.7640] }

{ "city" : "paris" }

simple JSON-NTV format of a NTV-list entity :

[[2.3522, 48.8566], {"lyon" : [4.8357, 45.7640]}]

[{ ":point" : [2.3522, 48.8566]}, {":point" : [4.8357, 45.7640]}]

[4, 45]

["paris"]

[]

named JSON-NTV format of a NTV-list entity :

{ "cities::point" : [[2.3522, 48.8566], {"lyon": [4.8357, 45.7640]}] }

{ "::point" : [[2.3522, 48.8566], {"lyon" : [4.8357, 45.7640]}] }

{ "simple list" : [4, 45.7] }

{ "generic date::dat" : ["2022-01-28T18-23-54Z", "2022-01-28", 1234.78] }

simple JSON-NTV format of a NTV-set entity :

{ "name": "white", "firstname":"walter", "surname":"heisenberg" }

{ "paris:point" : [2.3522, 48.8566] , "lyon" : "france"}

{ "paris" : [2.3522, 48.8566], "" : [4.8357, 45.7640]}

named JSON-NTV format of a NTV-set entity :

{ "cities::point": { "paris": [2.352, 48.856], "lyon": [4.835, 45.764]}}

{ "cities" : { "paris:point" : [2.3522, 48.8566] , "lyon" : "france"} }

{ "city" : { "paris" : [2.3522, 48.8566] } }

8 PARSING A JSON-VALUE

8.1 NTVTYPE

The NTVtype is identical to the JsonNTVtype for the NTV entities not included in another entity.

For NTV entities included in another entity, the NTVtype is set to :

- the JsonNTVtype if it is an absolute NTVtype and the entity is a NTV-single,
- the concatenation of the NTVtype of the structured NTV entity and the JsonNTVtype if it is a relative NTVtype,
- the NTVtype of the structured NTV entity if the JsonNTVtype is None.

If the resulting NTVtype is inconsistent, the NTVtype is set to None.

8.2 NTV ENTITY

An NTV entity is deduced from the JSON-value according to its JSON-type, JsonName (if present) and JsonValue (if different).

The tables below show the conversion rules.

JSON primitive, unnamed and array :

JSON-value	NTV entity
JSON-primitive	V-single
JSON-unnamed	V-set
JSON-array	V-list

JSON named :

JSON-value		NTV entity
JsonName	**JsonValue**	**NTV category**
without colon	JSON-primitive	NV-single
without colon	JSON-named	NV-single
without colon	JSON-unnamed	NV-set

without colon	JSON-array	NV-list
with one colon	JSON-value	NV-single or TV-single or NTV-single
with a pair of colons	JSON-unnamed	NV-set or TV-set or NTV-set
with a pair of colons	JSON-array	NV-list or TV-list or NTV-list

The NTVvalue of NTV-single is the JsonValue.

The NTVname is deduced from the JsonName.

The NTVtype is deduced from the JsonName or if None from the inherited NTVtype of the “parent” NTV entity (if exists).

Appendix : Global NTVtype

The structure of types by namespace makes it possible to have types corresponding to recognized standards at the global level.

A Global NTVtype is a NTVtype defined in the Global Namespace.

NTVtype and the rules to code or decode NTVvalues MUST be understood by data producers and data consumers.

So Global NTVtype and rules associated have to be defined in a specification shared by a large community.

The Global NTVtype are listed below (to be completed).

Datation

Datation has a generic Type : datation (or dat)

NTVtype (generic)	**NTVvalue**	**example NTVvalue**
year	Json-Number (Year - ISO8601)	1998
month	Json-Number (Month - ISO8601)	10
day	Json-Number (Day - ISO8601)	21
week	Json-Number (Week - ISO8601)	38
hour	Json-Number (Hour - ISO8601)	20
minute	Json-Number (Minute - ISO8601)	18
second	Json-Number (Second - ISO8601)	54
timeposix (dat)	Json-Number	123456.78
date (dat)	Json-string (Calendar date, extended format - ISO8601)	“2022-01-28”
time (dat)	Json-string (time of day, extended format - ISO8601)	“T18:23:54”, “18:23”, “T18”
datetime (dat)	Json-string (date and time of day, extended format - ISO8601)	“2022-01-28T18-23-54Z” “2022-01-28T18-23-54+0400”
timearray (dat)	Json-Array	[date1, date2] date1, date2 are date, datetime or timeposix
timeslot (dat)	Json-Array	[array1, array2]

		array1, array2 are timearray

Duration

Duration has a generic type : duration (or dur)

NTVtype (generic)	**NTVvalue**	**example NTVvalue**
timeinterval (dur)	Json-string (Time interval with start and end, extended format- ISO8601)	"2007-03-01T13:00:00Z/2008-05-11T15:30:00Z"
durationiso (dur)	Json-string (Time interval by alternative format duration and context, extended format - ISO8601)	"P0002-10- 15T10:30:20"
durposix (dur)	Json-Number	123456.78

Location

Location has a generic type : location (or loc)

The CRS (Coordinate Reference Systems) is geographic, using the World Geodetic System 1984 (WGS 84) datum, with longitude and latitude units of decimal degrees (EPSG:4326).

NTVtype (generic)	**NTVvalue**	**example NTVvalue**
point (loc)	Json-Array (Point coordinates - RFC7946)	[5.12, 45.256] *(lon, lat)*
line (loc)	Json-Array (LineString coordinates - RFC7946)	[point1, point2, point3] pointxx is point
multipoint	Json-Array (MultiPoint coordinates - RFC7946)	[point1, point2, point3] pointxx is point
multiline	Json-Array (MultiLineString coordinates - RFC7946)	[line1, line2, line3] linexx is line
polygon (loc)	Json-Array (Polygon coordinates - RFC7946)	[ring1, ring2, ring3] ringxx is line
multipolygon (loc)	Json-Array (MultiPolygon coordinates - RFC7946)	[poly1, poly2, poly3] polyxx is polygon

bbox (loc)	Json-Array (bbox coordinates - RFC7946	[-10.0, -10.0, 10.0, 10.0]
geojson (loc)	Json-Object (geoJSON object - RFC7946)	{"type": "point", "coordinates": [40.0, 0.0] }
codeolc (loc)	Json-string (Open Location Code)	"8FW4V75V+8F6"

Tabular data

Several types can be defined following the "Model for Tabular Data and Metadata on the Web - W3C Recommendation 17 December 2015"

NTVtype	**NTVvalue**
row	JSON-array of JSON-NTV
field	JSON-array of NTVvalue (following JSON-TAB format)
table	JSON-array of NTVvalue of fields with the same length

Normalized String

Normalized String doesn't have a generic Type.

NTV type	**NTVvalue**	**example NTVvalue**
uri	Json-string (URI - RFC3986)	"https://www.ietf.org/rfc/rfc3986.txt" "https://gallica.bnf.fr/ark:/12148/bpt6k107371t" "urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6" "ni:///sha-256;UyaQV-Ev4rdLoHyJJWCi11OHfrYv9E1aGQAlMO2X_-Q" "geo:13.4125,103.86673" (RFC5870) "info:eu-repo/dai/nl/12345" "mailto:John.Doe@example.com" "news:comp.infosystems.www.servers.unix" "urn:oasis:names:specification:docbook:dtd:xml:4.1.2"

(to be completed)

Appendix : Global NTVtype namespace

Country

The ISO 3166-1 alpha-2 codes are Namespace defined in the Global Namespace.

Example :

"fr." is the Namespace defined in the Global Namespace for France country NTVtypes.

Catalog

NTVtype	**example JSON-NTV**
schemaorg.	{ ":schemaorg.propertyID": "NO2" } { ":schemaorg.unitText":"µg/m3"}
darwincore.	{ ":darwincore.acceptedNameUsage": "Tamias minimus" }

(to be completed)