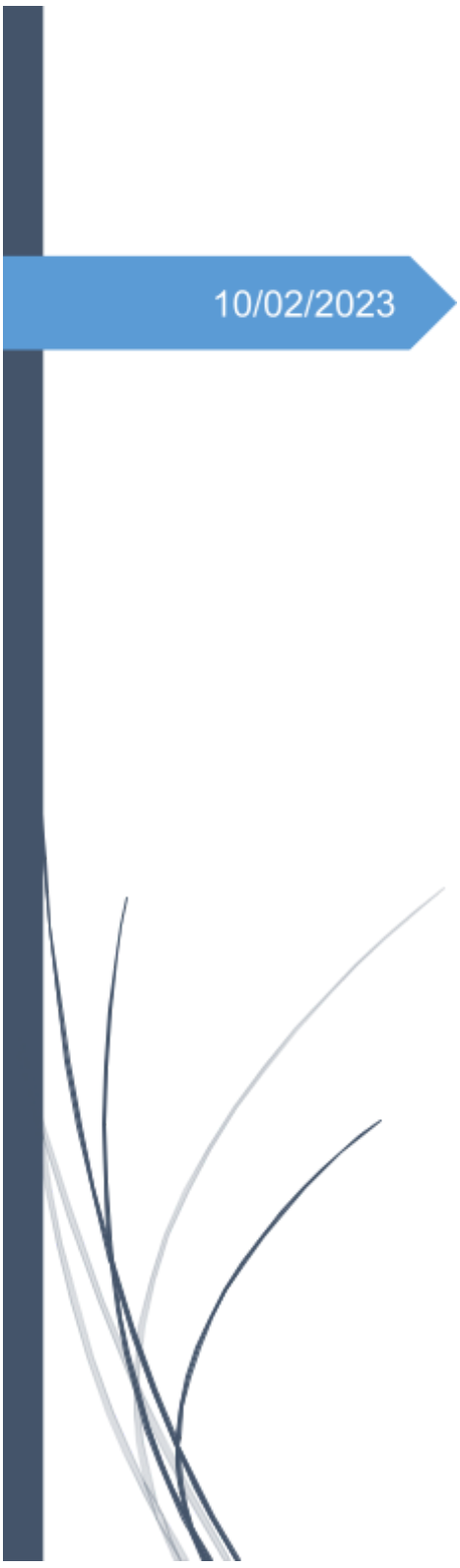
JSON-NTV Format

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 represent a JSON-object without single member

The JSON-named term represent a JSON-object with a single member

4 Introduction

The JSON grammar is based on simple entities: array, object, number, string, true, false, null.

JSON-NTV proposes to increase the semantic level of the 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 simplest way to add those informations is to use a JSON-object with a single member where:

- JSON-ND syntax is used for the member name
- JSON representation of the value is used for the member value.

The JSON value of the previous example is:

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

With this approach, three NTV entities are defined :

- one primitive entity not composed of other entities
- two structured entities: an unordered collection of NTV entities and an ordered sequence of NTV entities.

and two JSON formats depending on the presence of the OPTIONAL elements :

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

Example (entity composed with two 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

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 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 the Namespace or default NTVtype to apply to the NTV entities included.

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 available only for structured entities. Generic NTVtype avoids having to include a type in the JSON representation of the included NTV entities.

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 :

- absolut NTVtype,
- 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.

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.

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]}]

[4.8357, 45.7640]

["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]}] }

{ "lyon" : [4.8357, 45.7640] }

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 Generic, the NTVtype is set to the Simple NTVtype consistent with the JSON-value.

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

Example :

If a JSON-value is { "::dat" : ["2022-01-28T18-23-54", 123456.78] }, the parsers deduce that the first NTVvalue has a "datetime" NTVtype and the second a "timeposix" NTVtype.

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

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 table below shows the conversion rules.

JSON-value			**NTV entity**	
JSON-type	**JsonName**	**JsonValue**	**NTV category**	**JSON-NTV format**
JSON-primitive	None	JSON-value	NTV-single	simple
JSON-unnamed	None	JSON-value	NTV-set	simple
JSON-array	None	JSON-value	NTV-list	simple

JSON-named	without colon	JSON-primitive	NTV-single	named
JSON-named	without colon	JSON-object	NTV-set	named
JSON-named	without colon	JSON-array	NTV-list	named
JSON-named	with one colon	JSON-value	NTV-single	named
JSON-named	with two adjacent colon	JSON-object	NTV-set	named
JSON-named	with two adjacent colon	JSON-array	NTV-list	named
JSON-named	with two adjacent colon	JSON-primitive	None	None

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

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	**NTVvalue**	**example NTVvalue**
timeposix	Json-Number	123456.78
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
date	Json-string (Calendar date, extended format - ISO8601)	“2022-01-28”
time	Json-string (time of day, extended format - ISO8601)	“T18:23:54”, “18:23”, “T18”
datetime	Json-string (date and time of day, extended format - ISO8601)	“2022-01-28T18-23-54Z” “2022-01-28T18-23-54+0400”
timearray	Json-Array	[date1, date2] date1, date2 are date, datetime or timeposix
timeslot	Json-Array	[array1, array2] array1, array2 are timearray

DURATION

Duration has a generic type : duration (or dur)

NTVtype	NTVvalue	example NTVvalue
timeinterval	Json-string (Time interval with start and end, extended format- ISO8601)	"2007-03-01T13:00:00Z/2008-05-11T15:30:00Z"
durationiso	Json-string (Time interval by alternative format duration and context, extended format - ISO8601)	"P0002-10- 15T10:30:20"
durposix	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	NTVvalue	example NTVvalue
point	Json-Array (Point coordinates - RFC7946)	[5.12, 45.256] *(lon, lat)*
line	Json-Array (LineString coordinates - RFC7946)	[point1, point2, point3] pointxx is point
ring	Json-Array (linear ring coordinates - RFC7946)	[point1, point2, point3] pointxx is point
multiline	Json-Array (MultiLineString coordinates - RFC7946)	[line1, line2, line3] linexx is line
polygon	Json-Array (Polygon coordinates - RFC7946)	[ring1, ring2, ring3] ringxx is ring
multipolygon	Json-Array (MultiPolygon coordinates - RFC7946)	[poly1, poly2, poly3] polyxx is polygon
bbox	Json-Array (bbox coordinates - RFC7946	[-10.0, -10.0, 10.0, 10.0]
geojson	Json-Object (geoJSON object -	{“type”: “point”,

	RFC7946)	"coordinates": [40.0, 0.0] }
codeolc	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	Definition
row	NTV-list of entities
field	NTV-list of entities (following JSON-TAB format)
table	NTV-list of "field" entities 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": "PM10" } { ":schemaorg.unitText":"µg/m3"}
darwincore.	{ ":darwincore.acceptedNameUsage": "Tamias minimus" }

(to be completed)

Appendix : Example of .fr NTVtype namespace

Identifier

Ils pourraient correspondre à des identifiants définis dans des jeux de données référencés (via un schéma de données ou un modèle de données).

Par exemple :

NTVtype	NTVvalue	définition	example NTVvalue
fr.reg	Json-number	code région	93
fr.dep	Json-number	code département	60
fr.com	Json-number	code INSEE commune	77284
fr.cp	Json-number	code postal	76450
fr.iris	Json-number	code IRIS ilôt	977010101
fr.sirenc	Json-number	code SIREN commune	217702737
fr.epci	Json-number	code EPCI	244301107
fr.arm	Json-number	code arrondissement municipal	842
fr.can	Json-number	code canton	8208
fr.ctcd	Json-string	code collectivité territoriale	6AE
fr.cog	Json-number	code officiel géographique COG	99114
fr.cov	Json-string	code zone covoiturage	35238-C-012
fr.zfe	Json-string	code ZFE	200046977-ZFE-001
fr.bnls	Json-string	code lieu stationnement	04070-P-001
fr.uic	Json-number	code UIC gare	8757449
fr.iata	Json-string	code IATA aéroport	CDG
fr.naf	Json-number	code NAF	23
fr.siret	Json-number	code SIRET enterprise	41844736300015
fr.siren	Json-number	code SIREN enterprise	418447363
fr.fantoir	Json-string	code FANTOIR voie	4500023086F
fr.parcel	Json-number	code parcelle	670010320165

fr.iua	Json-string	code id adresse	27115_0110_00017_bis
fr.struc	Json-string	code SP+ structure	3Tn8gzTdcz
fr.synop	Json-number	code SYNOP station météo	07130
fr.lcsqa	Json-string	code LCSQA station mesure	FR01021
fr.uai	Json-string	code UAI établissement	0951099D
fr.aca	Json-number	code académies	22
fr.circo	Json-number	code circonscription	69002
fr.finessej	Json-number	code FINESS entité juridique	790001606
fr.finesset	Json-number	code FINESS établissement	790010375
fr.rna	Json-string	code WALDEC association	843S0843004860
fr.spi	Json-number	code SPI numéro fiscal	1899582886173
fr.nir	Json-number	code NIR sécurité sociale	164026005705953

NAMESPACE

Les espaces de noms pourraient correspondre à des catalogues ou des jeux de données dont les types de données sont identifiés dans des modèles de données ou bien dans des schémas de données.

Par exemple :

NTVtype	**example JSON-NTV**
fr.sandre.	{ ":fr.sandre.CdStationHydro": K163 3010 01 } { ":fr.sandre.TypStationHydro": "standard" }
fr.synop.	{ ":fr.synop.numer_sta": 07130 } { ":fr.synop.t": 300, ":fr.synop.ff": 5 }
fr.IRVE.	{ ":fr.IRVE.nom_station": "M2026" } { ":fr.IRVE.nom_operateur": "DEBELEC" }
fr.BAN.	{ ":fr.BAN.numero": 54 } { ":fr.BAN.lon": 3.5124 }

ENTITY

Ils pourraient correspondre à des assemblages de données associées à un type défini.

Par exemple :

NTVtype	NTVvalue	example JSON-NTV
fr.parcelle	Json-array	{"monAdresse:fr.parcelle": [84500, 0, I, 97]} *(fr.cp, fr.cadastre.préfixe, fr.cadastre.section, fr.cadastre.numéro)*
fr.adresse	Json-array	{"monAdresse:fr.adresse": [54, bis, rue de la mairie, 78730] *(fr.BAN.numero, fr.BAN.rep, fr.BAN.nom_voie, fr.cp)*