JSON-NTV

Philippe Thomy

ID: draft-thomy-json-ntv

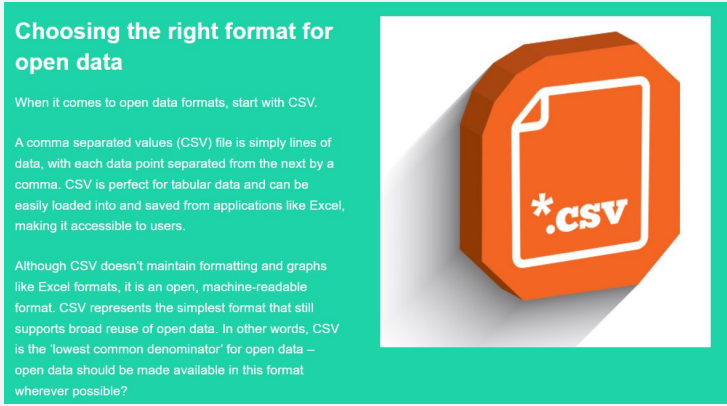
What is the problem ?

The most widely used data exchange formats have a remarkably low semantic level:

- JSON : numbers, string, literals / object, array structure
- CSV : only ASCII text / table structure

This makes the reuse of exchanged information complex:

- inference to translate data into information
- data sniffing tools
- meaningful naming
- low level of automation



EU open data recommendation

JSON existing improvement

Data schema defines rules and data types to apply in a JSON data

Extension defines data types in specific domains

Examples of Data schemas

- JSON schema, JSON Type Definition, *Table Schema*

Examples of extensions

- GeoJSON, jCal, jCard, JWA, JSON IODEF, JSContact

but the problem remains because there is no standard for using data type in JSON data

CBOR is a good example of taking data type (tags) into account

Objective

To extend the JSON format to semantic types with the following goals:

1. Take into account most common data formats used in Internet standards
2. No schema description needed to decode Data
3. The format is a superset of the existing JSON format
4. The format representation is either text or binary, with strict reversibility (round-trip) and no-ambiguity
5. The format is simple and reasonably compact
6. The format is extensible

JSON-NTV proposal (draft-thomy-json-ntv)

1 - Common formats
2 - Without schema
5 - Compact and simple

"Name:type" JSON Notation defined in JSON-ND project in 2020 (Glen Kleidon)

Single entity
(leaf node)

Name : Equinox
Type : date
Value : 20-03-2023

{ "equinox:date": "2023-03-20"}

Structured entity
(inner node)

Name : Equinox ***Type*** *: date* **Value** :

Name : First
Type : date
Value : 20-03-2023

Name : Second
Type : date
Value : 23-09-2023

{ "equinox::date": { "first": "2023-03-20", "second": "2023-09-23"}
{ "equinox": { "first:date": "2023-03-20", "second:date": "2023-09-23"}

3 - JSON-NTV is a superset of JSON

"json" type is the default type of NTV single entities

Single entity

Name : None
Type : json
Value : 25

25

Name : number
Type : json
Value : 25

{ "number": 25 }

Structured entity

Name: None **Type**: None **Value** :

Name : First
Type : json
Value : 10

Name : Second
Type : json
Value : 20

{ "first": 10, "second": 20 }

Name: None **Type**: None **Value** :

Name : None
Type : json
Value : 10

Name : None
Type : json
Value : 20

[10, 20]

4 - text and binary representation
6 - extensible

Reversible text representation

- JSON encoding

Reversible binary representation

- CBOR encoding

Defined type (most common)

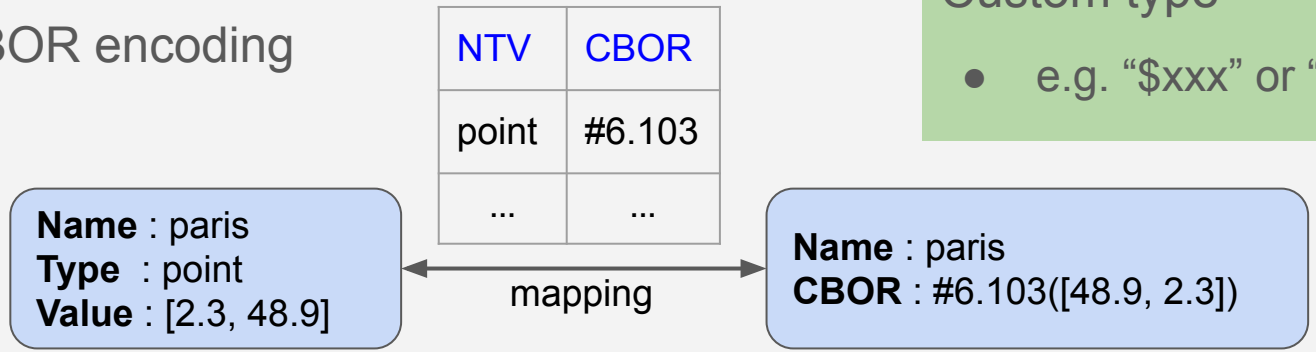
- e.g. “point”, “date”, “URI”

Nested type (namespace)

- e.g. “fr.dpt”

Custom type

- e.g. “$xxx” or “#xxx”



6.1. Converting from CBOR to JSON *(RFC8949)*

- *For all other tags (major type 6, any other tag number), the tag content is represented as a JSON value; the tag number* ***is ignored.***

Solution relevance

A typed JSON enables the inclusion of any type of object defined by a name, a type and a JSON value (most common data structure in programming languages).

Tabular data is a declinaison of this format. It could be an alternative to the obsolete CSV format (e.g. tabular JSON-NTV python implementation is referenced in the pandas ecosystem)

The approach is simple and easy to implement. A first python implementation is available.

Why is it relevant for IETF ?

IETF covers JSON format and his extensions

JSON-NTV is compatible with JSON pointer, JSON patch, JSON path

JSON-NTV is complementary to CBOR (Answer to: How to convert a tag into JSON ?)

It seems to me that the study of a JSON alternative to CSV format falls under the IETF

This is an opportunity to converge approaches to data schemas (e.g. JSON type definition - RFC8927), tags management (e.g. draft "notable CBOR tags") and semantic structures.

So, include it in an existing or a new WG