JSON-NTV

Philippe Thomy
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

Choosing the right format for open data

When it comes to open data formats, start with CSV.

A comma-separated values (CSV) file is simply lines of data, with each data point separated from the next by a comma. CSV is perfect for tabular data and can be easily loaded into and saved from applications like Excel, making it accessible to users.

Although CSV doesn’t maintain formatting and graphs like Excel formats, it is an open, machine-readable format. CSV represents the simplest format that still supports broad reuse of open data. In other words, CSV is the ‘lowest common denominator’ for open data – open data should be made available in this format wherever possible.
JSON existing improvement

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

Extension defines data types in specific domains

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

Examples of Data schemas

- JSON schema, JSON Type Definition, Table Schema

Examples of extensions

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

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

**Single entity (leaf node)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equinox</td>
<td>date</td>
<td>20-03-2023</td>
</tr>
</tbody>
</table>

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

**Structured entity (inner node)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>First</td>
<td>date</td>
<td>20-03-2023</td>
</tr>
<tr>
<td>Second</td>
<td>date</td>
<td>23-09-2023</td>
</tr>
</tbody>
</table>

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

“Name:type” JSON Notation defined in JSON-ND project in 2020 (Glen Kleidon)
3 - JSON-NTV is a superset of JSON

**Single entity**

- **Name**: None
  - **Type**: json
  - **Value**: 25

- **Name**: number
  - **Type**: json
  - **Value**: 25

**Structured entity**

- **Name**: First
  - **Type**: json
  - **Value**: 10

- **Name**: Second
  - **Type**: json
  - **Value**: 20

- **Name**: None
  - **Type**: json
  - **Value**: 10

- **Name**: None
  - **Type**: json
  - **Value**: 20

- **Name**: None
  - **Type**: None
  - **Value**: [ 10, 20 ]

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