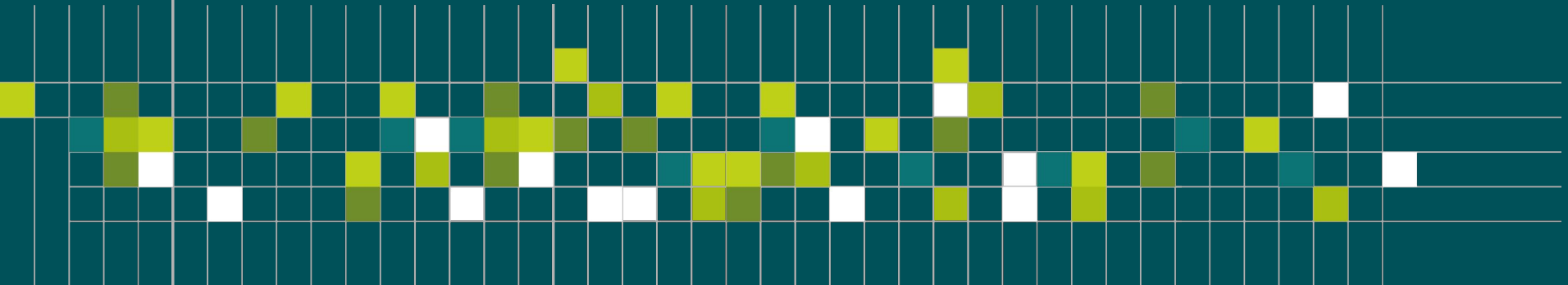
loco

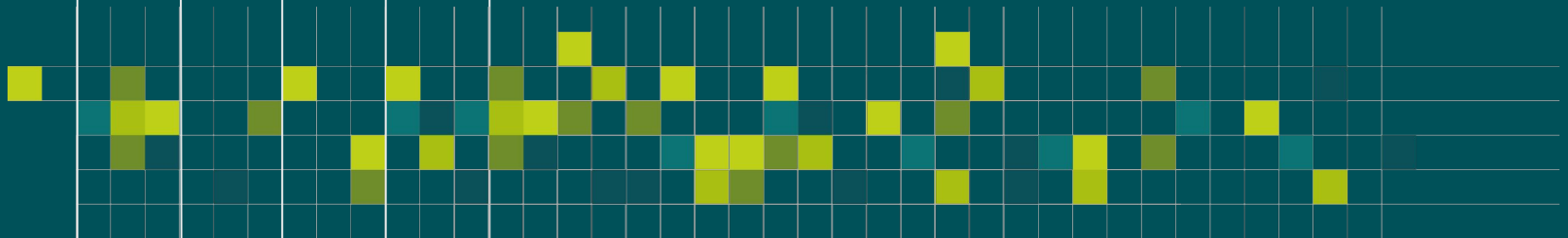
{ NTV Semantic JSON }

JSON-NTV format

A semantic format for interoperability



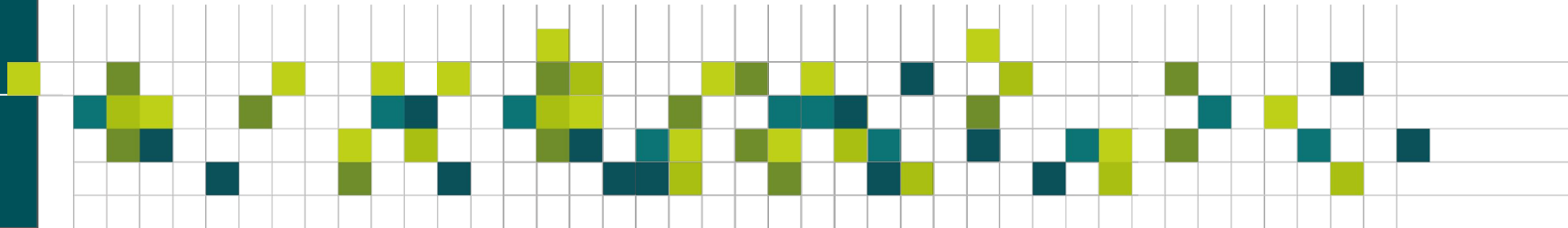
Sommaire



Introduction

Pourquoi étendre le format JSON ?

- Principaux formats d'échange interopérables (hors binaire)
  - Format CSV pour les données tabulaires
    - Obsolète, niveau sémantique très faible (chaînes de caractères)
  - Format JSON
    - Langage simple de description, peu de types de données (object, array, string, number, false, true, null)
  - Format XML
    - Langage de balisage "extensible" mais plus complexe
- Difficultés
  - Le type des données (sémantique) n'est pas pris en compte
  - Formats non autoporteur -> Schémas de données complémentaires (Table schema, JSON schema, XML schema)
  - Formats CSV et XML anciens (CSV : 2005, XML : 2008)



Présentation

Format NTV

- Origine
  - Format JSON-ND défini en 2018 (*JSON with Named Datatypes*)
- Structure
  - Entité NTV
    - Value : Donnée échangée
    - Name : Interprétation ou complément utile à la compréhension
    - Type : Nature de la donnée dans un standard, catalogue ou logiciel
  - Format JSON-NTV (JSON augmenté)
    - Primitive : Donnée unique (Value est une "JSON-value")
    - Structure : Données composées (Value est une liste d'entités NTV)

43

Value : 43
Type : 'json' (défaut)
Name : None (défaut)

{ ':point' : [2.3, 48.9] }

Value : [2.3, 48.9]
Type : 'point'
Name : None (défaut)

Donnée unique

{ 'Paris:point' : [2.3, 48.9] }

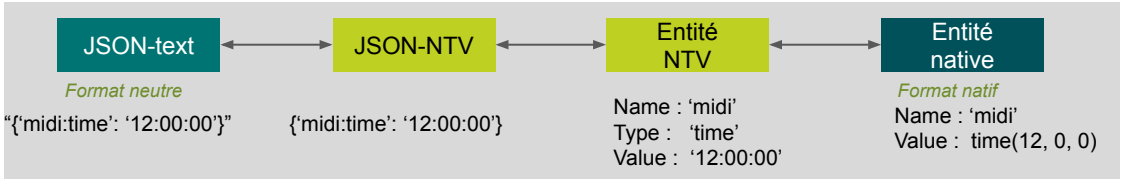
Value : [2.3, 48.9]
Type : 'point'
Name : Paris

Donnée composée

{ '::point' : [[2.3, 48.9], [4.8, 45.8]]

Value : [2.3, 48.9] et [4.8, 45.8]
Type : liste de 'point'
Name : None (défaut)

{ 'city::point' : { 'paris: [2.3, 48.9], 'lyon': [4.8, 45.8] } }
ou
{ 'city' : { 'paris:point': [2.3, 48.9], 'lyon:point': [4.8, 45.8] } }



loco

Présentation

Propriétés

- Format universel et réversible
  - Objets exprimables sous forme Json
- JSON unifié
  - Mixité et compatibilité JSON / JSON-NTV
  - Enrichissement

  [3, 'patates'] { 'NTV example' : [3, 'patates'] }

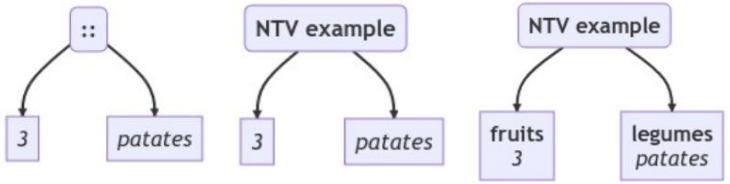
  { 'NTV example' : { 'fruits': 3, 'legumes': 'patates' } }

  { 'NTV example' : [{'fruits': 3}, {'legumes': 'patates' }] }
- Structure NTV
  - Arborescence (rooted tree)
- Type NTV
  - Typage arborescent et customisable

{ 'city' : { 'paris: [2.3, 48.9], 'lyon': [4.8, 45.8] } }

{ 'city::point' : { 'paris: [2.3, 48.9], 'lyon': [4.8, 45.8] } }

{ 'city:point' : { 'paris: [2.3, 48.9], 'lyon': [4.8, 45.8] } }



{ "paris airport:iata": "CDG" }

{ "::time":{"morning": "08:00:00","midnight":"00:00:00"} }

{ "::uint16": [1, 2, 3, 4] }

{ ":schemaorg.propertyID": "NO2" }

{ ":darwincore.acceptedNameUsage": "Tamias minimus" }

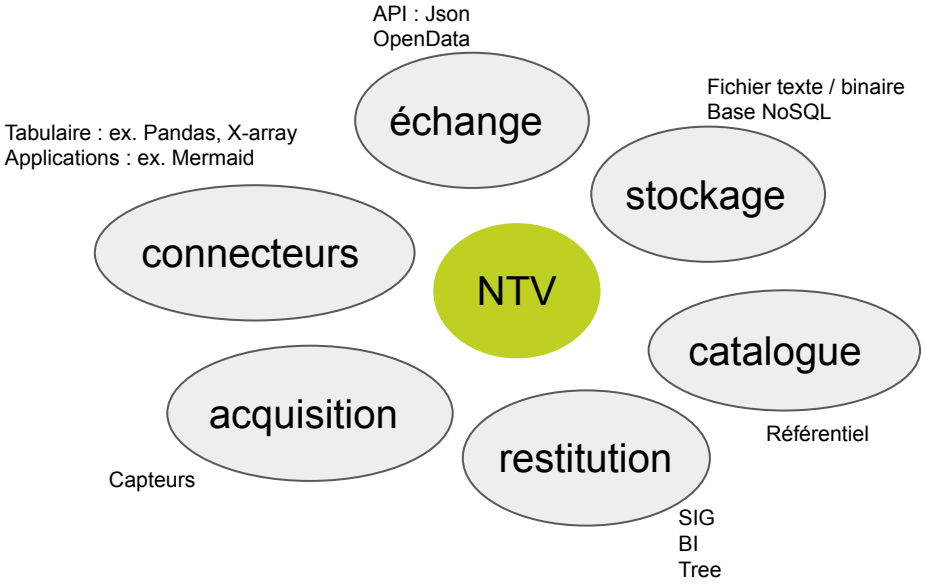
{ "oise:fr.dep": 60 }

{"name": ["john", "Anna"], "age": [25, 36] } ***(tab object)***

{":$sensor": [3.51, 4.2, "mg/m3", [4.1, 45.2]]}

Présentation

Usage



- Intérêt
  - JSON

    lisible, simple, web-native, universel, compact
  - Sémantique

    nombreux types prédéfinis, types customisables
  - Compatibilité

    équivalence et mixité JSON / JSON-NTV,

    conservation des types existants et intégration des catalogues de données
  - Neutralité

    indépendance logicielle et matérielle des types et structures,

    concept NTV applicable à toute structure de données

Annexe

Exemple tabulaire

```
produit,quantite,aliment,validite,id,disponibilite,prix
piment,sachet 1 kg,legume,2022-01-01,15,oui,1.5
piment,carton 10 kg,legume,2022-01-01,16,fin 2022,15
banane,sachet 1 kg,fruit,2022-01-01,17,oui,0.5
banane,carton 10 kg,fruit,2022-01-01,18,oui,5
pomme,sachet 1 kg,fruit,2022-01-01,11,oui,1
pomme,carton 10 kg,fruit,2022-01-01,12,oui,10
orange,sachet 1 kg,fruit,2022-01-01,13,fin 2022,2
orange,carton 10 kg,fruit,2022-01-01,14,fin 2022,20
```

Format CSV

id	produit	aliment	quantité	prix	validité	disponibilité
11	pomme	fruit	sachet 1 kg	1	2022-01-01	oui
12	pomme	fruit	carton 10 kg	9	2022-01-01	oui
13	orange	fruit	sachet 1 kg	2	2022-01-01	fin 2022
14	orange	fruit	carton 10 kg	18	2022-01-01	fin 2022
15	piment	légume	sachet 1 kg	1.5	2022-01-01	oui
16	piment	légume	carton 10 kg	13	2022-01-01	fin 2022
17	banane	fruit	sachet 1 kg	0.5	2022-01-01	oui
18	banane	fruit	carton 10 kg	4	2022-01-01	oui

```
{'produit': ['piment', 'piment', 'banane', 'banane', 'pomme', 'pomme', 'orange', 'orange'],
 'quantite': ['sachet 1 kg', 'carton 10 kg', 'sachet 1 kg', 'carton 10 kg', 'sachet 1 kg',
'carton 10 kg', 'sachet 1 kg', 'carton 10 kg'],
 'aliment': ['legume', 'legume', 'fruit', 'fruit', 'fruit', 'fruit', 'fruit', 'fruit'],
 'validite::date': ['2022-01-01', '2022-01-01', '2022-01-01', '2022-01-01', '2022-01-01', '2
022-01-01', '2022-01-01', '2022-01-01'],
'id::int': [15, 16, 17, 18, 11, 12, 13, 14],
'disponibilite': ['oui', 'fin 2022', 'oui', 'oui', 'oui', 'oui', 'fin 2022', 'fin 2022'],
'prix': [1.5, 15, 0.5, 5, 1, 10, 2, 20]}
```

Format JSON “type CSV”

Format optimisé

```
{'produit': [['piment', 'banane', 'pomme', 'orange'], [2]],
 'quantite': [['sachet 1 kg', 'carton 10 kg'], [1]],
 'aliment': [['fruit', 'legume'], 0, [1, 0, 0, 0]],
 'validite::date': '2022-01-01',
 'id::int': [15, 16, 17, 18, 11, 12, 13, 14],
 'disponibilite': [['fin 2022', 'oui'], [1, 0, 1, 1, 1, 1, 0, 0]],
 'prix', [1.5, 15, 0.5, 5, 1, 10, 2, 20]}
```

Format annoté

```
{'produit': [['piment', 'banane', 'pomme', {'provenance espagne': 'orange'}], [2]],
 'quantite': [['sachet 1 kg', 'carton 10 kg'], [1]],
 'aliment': [['fruit', 'legume'], 0, [1, 0, 0, 0]],
 'validite::date': '2022-01-01',
 'id::int': [15, 16, 17, 18, 11, 12, 13, 14],
 'disponibilite': [['fin 2022', 'oui'], [1, 0, 1, 1, 1, 1, 0, 0]],
 'prix', [1.5, 15, 0.5, {'à confirmer': 5}, 1, 10, 2, 20]}
```

Avantages

- Pas de contrainte sur la nature des données
- Nom et type des colonnes explicite
- Taille de fichier équivalente ou réduite
- Données annotables
- Partage identique par API ou par fichier (texte ou binaire)

Exemple Mermaid

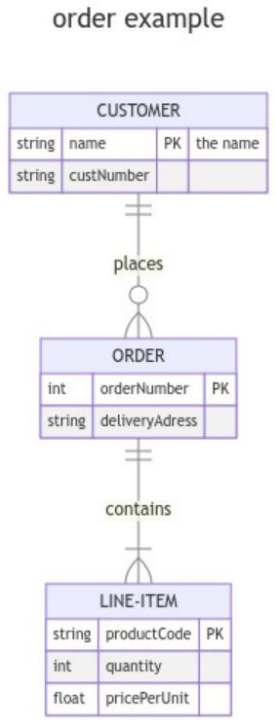
Annexe

```
order_example = {
    'order example:$erDiagram' : {
        'relationship::': [
            [ 'CUSTOMER', 'exactly one', 'identifying', 'zero or more', 'ORDER',     'places'],
            [ 'ORDER',    'exactly one', 'identifying', 'one or more',  'LINE-ITEM', 'contains']
        ],
        'entity::': {
            'CUSTOMER':  [
                ['string', 'name',         'PK', 'the name'],
                ['string', 'custNumber']
            ],
            'ORDER': [
                ['int',    'orderNumber',  'PK'],
                ['string', 'deliveryAdress']
            ],
            'LINE-ITEM': [
                ['string', 'productCode',  'PK'],
                ['int',    'quantity'],
                ['float',  'pricePerUnit']
            ]
        },
    } }
```

Représentation Mermaid simple

Représentation Mermaid annotée

```
order_example2 = {
    ':$erDiagram' : {
        'relationship::': [
            [ {'1st entity':'CUSTOMER'}, 'exactly one', 'identifying', 'zero or more', {'2nd entity':'ORDER'}, 'places'],
            [ 'ORDER', {'to be confirmed': 'exactly one'}, 'identifying', 'one or more',  'LINE-ITEM', {'label': 'contains'}]
        ],
        'entity::': {
            'CUSTOMER':  [
                ['string', 'name', 'PK', {'comments': 'the name'}],
                ['string', 'custNumber']
            ],
            'ORDER': [
                {'type_att':'int', 'name_att':'orderNumber',  'key_att': 'PK'},
                {'this attribute is not yet valid' : ['string', 'deliveryAdress']}
            ],
            'LINE-ITEM': {
                'first attribute': ['string', 'productCode',  'PK'],
                'second attribute': ['int',    'quantity'],
                'third attribute': ['float',  'pricePerUnit']
            }
        },
    } }
```

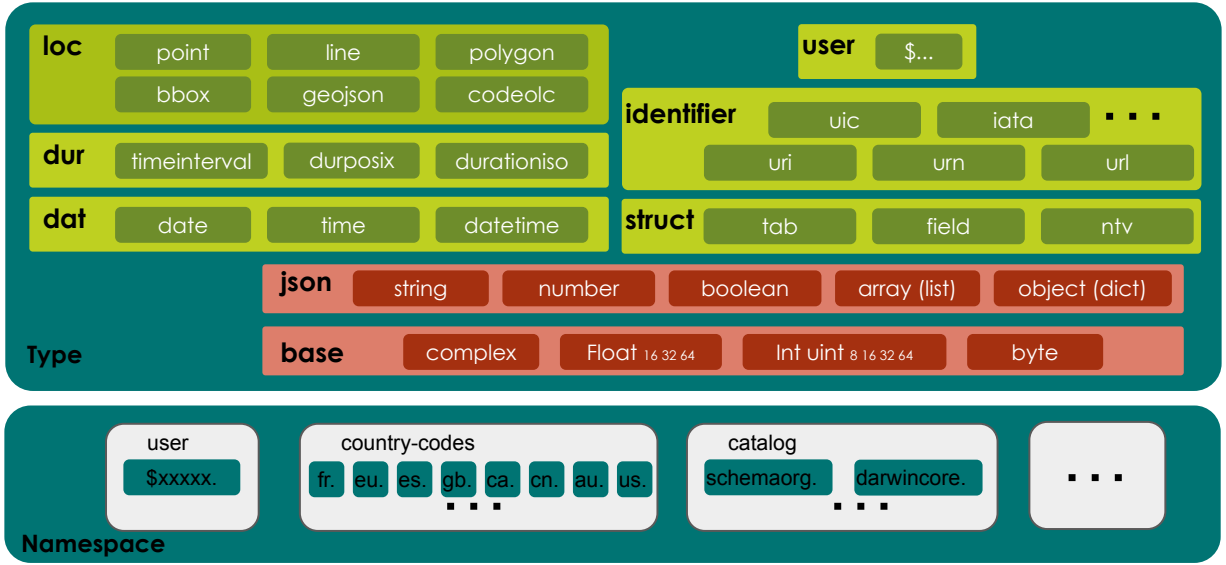


loco

Annexe

Augmentation du niveau sémantique

Global Namespace



Exemples :

{ ":schemaorg.propertyID": "NO2" }

{ ":darwincore.acceptedNameUsage": "Tamias minimus" }

{ "oise:fr.dep": 60 }

{ "paris airport:iata": "CDG" }

{ "uri syntax:url": "https://www.ietf.org/rfc/rfc3986.txt" }

{ "marseille:point": [5.37, 43.3] }

{ "cities::loc": [[5.37, 43.3], "8FW4V75V+8F6"] }

{ "::time":{"morning": "08:00:00","midnight":"00:00:00"} }

[1, 2, 3, 4] *(default type : json)*

{ *"::uint16":* [1, 2, 3, 4] }

{"name": ["john", "Anna"], "age": [25, 36] } *(tab object)*

{":float16": 4096 }

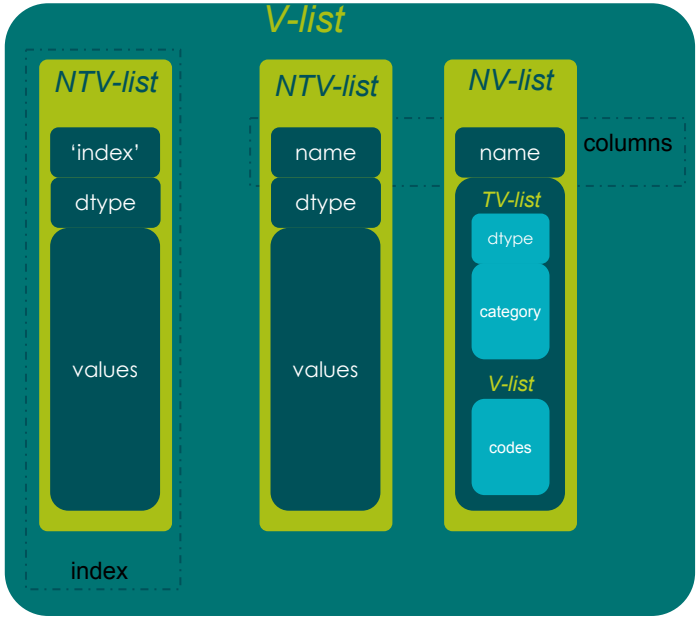
{":$sensor": [3.51, 4.2, "mg/m3", [4.1, 45.2]]}

loco

Annexe

Pandas mapping

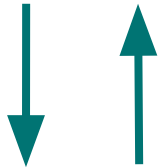
JSON-TAB



Examples :

index	dates::date	value	value32	res	coord::point	names	unique
100	1964-01-01	10	12	10	POINT (1 2)	john	True
200	1985-02-05	10	12	20	POINT (3 4)	eric	True
300	2022-01-21	20	22	30	POINT (5 6)	judith	True
400	1964-01-01	20	22	10	POINT (7 8)	mila	True
500	1985-02-05	30	32	20	POINT (3 4)	hector	True
600	2022-01-21	30	32	30	POINT (5 6)	maria	True

```
dates::date      object
value             int64
value32           int32
res               int64
coord::point     object
names            string
unique             bool
dtype: object
```



```
{':tab': {'coord::point': [[1.0, 2.0], [3.0, 4.0], [5.0, 6.0], [7.0, 8.0], [3.0, 4.0], [5.0, 6.0]],
          'dates::date': ['1964-01-01', '1985-02-05', '2022-01-21', '1964-01-01', '1985-02-05', '2022-01-21'],
          'index': [100, 200, 300, 400, 500, 600],
          'names::string': ['john', 'eric', 'judith', 'mila', 'hector', 'maria'],
          'res': [10, 20, 30, 10, 20, 30],
          'unique': [True, True, True, True, True, True],
          'value': [10, 10, 20, 20, 30, 30],
          'value32::int32': [12, 12, 22, 22, 32, 32]}}
```