Linked Data

How to Make Machines Understand APIs by Linking to UN/CEFACT Semantics

Nis Jespersen
nis.jespersen@maerskgtd.com
TradeLens.com

ISCO

A Typical JSON

Here's what a typical JSON response might look like, returned from, say, a carrier's tracking API:

```
{
  "shipment": {
  "bookingNumber": "123456789",
  "goods": [{
      "goodsItem": {
        "information": "Mangos and bananas",
        "grossWeight": {"Value": "12000", "Unit": "Kgs"}
      }
    }],
  "containers": [{
      "container": {
        "boxNb": "MSKU0134962",
      }
    }]
}
```

We kinda get this, right? The usual story: carriers call it "shipment", but seeing that it's got a bookingNb, we know what is meant. We also guess that "container" means some kind of big, multimodal metal box, and the geeks among us even know who owns the ones starting with MSKU. We apply a ton of implicit context, and we're pretty sure we get it.

UN/CEFACT Alignment

```
{
  "consignment": {
    "identification": "123456789",
    "includedConsignmentItem": [{
        "consignmentItem": {
          "information": "Mangos and bananas",
          "grossWeight": {"Value": "12000", "Unit": "Kgs"}
        }
      }],
    "utilizedTransportEquipment": [{
        "transportEquipment": {
          "identification": "MSKU0134962",
          "affixedSeal": {"identification": "54234398"}
        }
      }]
  }
}
```

This is much better. Someone have clearly peeked at UN/CEFACT while constructing this. It uses the semantics which we know and love, and we can almost see the UML diagrams in play here!

A good start, and clear to us (humans), which recognize the terms and are creative enough to mentally make the connection. But to a computer, ”consignment” has no meaning. It’s just a string.

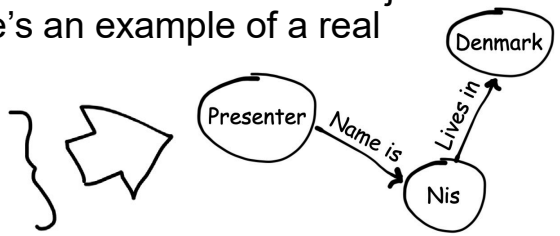
Also, this is a massive breaking change affecting anyone implementing the earlier API version.

Enter Linked Data

Machines build “knowledge graphs” by linking data together.

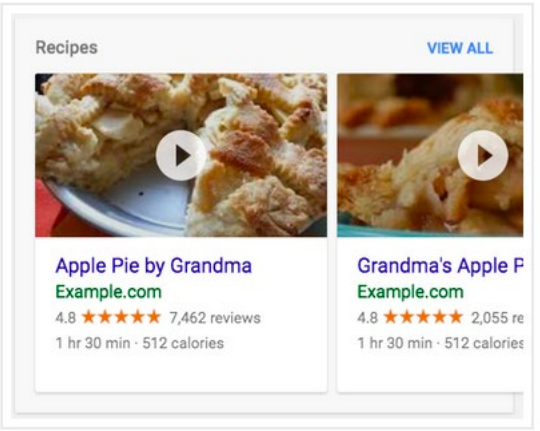
Resource Description Framework “RDF” defines subject-predicate-object triplets. Here’s an example of a real simple knowledge graph:

- Presenter’s name is Nis.
- Nis lives in Denmark



From this, a machine would be able to determine that the presenter lives in Denmark.

More tangibly, when Google presents something like this, it is based on a knowledge graph built from input like this.




```
<html>
<head>
<title>Grandma's Holiday Apple Pie</title>
<script type="application/ld+json">
{
  "@context": "https://schema.org/",
  "@type": "Recipe",
  "name": "Grandma's Holiday Apple Pie",
  "author": "Elaine Smith",
  "image": "http://images.edge-generalmills.com/564592
  "description": "A classic apple pie.",
  "aggregateRating": {
    "@type": "AggregateRating",
    "ratingValue": "4",
    "reviewCount": "276",
    "bestRating": "5",
    "worstRating": "1"
  },
  "prepTime": "PT30M",
  "totalTime": "PT1H",
```

(Example from https://developers.google.com/search/docs/guides/intro-structured-data)

Further, in this web-linked world, everything is identified by a URI – “uniform resource identifier”. An URI *identifies* *the thing*, as opposed to a URL which *locates a page describing* the *the thing*.

There can be a bit of a clash here between what’s appropriate for humans vs machines. For example `<https://schema.org/instructor> <https://schema.org/givenName> ”Nis” .` is hardly as readable as ”Presenter’s name is Nis”.

Luckily, RDF comes in many flavors and JSON-LD is quite useful for both humans and machines.

JSON-LD

First off, JSON-LD is based on JSON, which practically is the grammar of APIs.

JSON-LD works by injecting a context and some other aspects (all prefixed with an “@” indicating JSON-LD keyword) into your normal JSON. Let’s look at the example from earlier:

```
{"@context": {
    "shipment": {
      "@id": "https://edi3.org/specs/edi3-transport/develop/vocab/Consignment",
      "@type": "@id" },
    "goods": "https://edi3.org/specs/edi3-transport/develop/vocab/Consignment#ConsignmentItem",
    "goodsItem": "https://edi3.org/specs/edi3-transport/develop/vocab/ConsignmentItem",
    "containers": "https://edi3.org/specs/edi3-transport/develop/vocab/Consignment#utilizedTransportEquipment",
    "container": {
      "@id": "https://edi3.org/specs/edi3-transport/develop/vocab/TransportEquipment",
      "@type": "@id" }},
  "shipment": {
    "bookingNumber": "123456789",
    "@id": "https://www.maersk.com/tracking/123456789",
    "goods": [{
      "goodsItem": {
        "information": "Mangos and bananas",
        "grossWeight": {"Value": "12000", "Unit": "Kgs"}}}],
    "containers": [{
      "container": {
        "boxNb": "MSKU0134962",
        "@id": "https://app.bic-boxtech.org/containers?search=MSKU0134962"}}]}}
```

Here, the `@context` adds machine-understandable RDF to the human-understandable JSON. Also, `@id` adds the necessary URIs for the parts which need identification, here leveraging Maersk and BIC APIs.

All this is retrofitted “on top” – the legacy JSON still works! Your APIs don’t break!

A Machines Interpretation

Quick demo.

But in case something doesn't work out as planned, here's the result I hope to get to:

```
{
  "https://edi3.org/specs/edi3-transport/develop/vocab/Consignment": {
    "@id": "https://www.maersk.com/tracking/123456789",
    "https://edi3.org/specs/edi3-transport/develop/vocab/Consignment#ConsignmentItem": {
      "https://edi3.org/specs/edi3-transport/develop/vocab/ConsignmentItem": {}
    },
    "https://edi3.org/specs/edi3-transport/develop/vocab/Consignment#utilizedTransportEquipment": {
      "https://edi3.org/specs/edi3-transport/develop/vocab/TransportEquipment": {
        "@id": "https://app.bic-boxtech.org/containers?search=MSKU0134962"
      }
    }
  }
}
```

or

```
<https://www.maersk.com/tracking/123456789> <https://edi3.org/specs/edi3-transport/develop/vocab/Consignment#ConsignmentItem> _:b1 .
<https://www.maersk.com/tracking/123456789> <https://edi3.org/specs/edi3-transport/develop/vocab/Consignment#utilizedTransportEquipment>
_:b3 .
_:b0 <https://edi3.org/specs/edi3-transport/develop/vocab/Consignment> <https://www.maersk.com/tracking/123456789> .
_:b1 <https://edi3.org/specs/edi3-transport/develop/vocab/ConsignmentItem> _:b2 .
_:b3 <https://edi3.org/specs/edi3-transport/develop/vocab/TransportEquipment> <https://app.bic-boxtech.org/containers?search=MSKU0134962>
.
```

or

```
[
  {
    "https://edi3.org/specs/edi3-transport/develop/vocab/Consignment": [
      {
        "@id": "https://www.maersk.com/tracking/123456789",
        "https://edi3.org/specs/edi3-transport/develop/vocab/Consignment#utilizedTransportEquipment": [
          {
            "https://edi3.org/specs/edi3-transport/develop/vocab/TransportEquipment": [
              {
                "@id": "https://app.bic-boxtech.org/containers?search=MSKU0134962"
              }
            ]
          }
        ],
        "https://edi3.org/specs/edi3-transport/develop/vocab/Consignment#ConsignmentItem": [
          {
            "https://edi3.org/specs/edi3-transport/develop/vocab/ConsignmentItem": [
              {}
            ]
          }
        ]
      }
    ]
  }
]
```

These are different ways to serialize the same model, which was built from importing our earlier example. The key here is that there's a model constructed from our data; a model, which can be extended to tons of other linked data, potentially from a ton of other data sources.

UN/CEFACT's Role

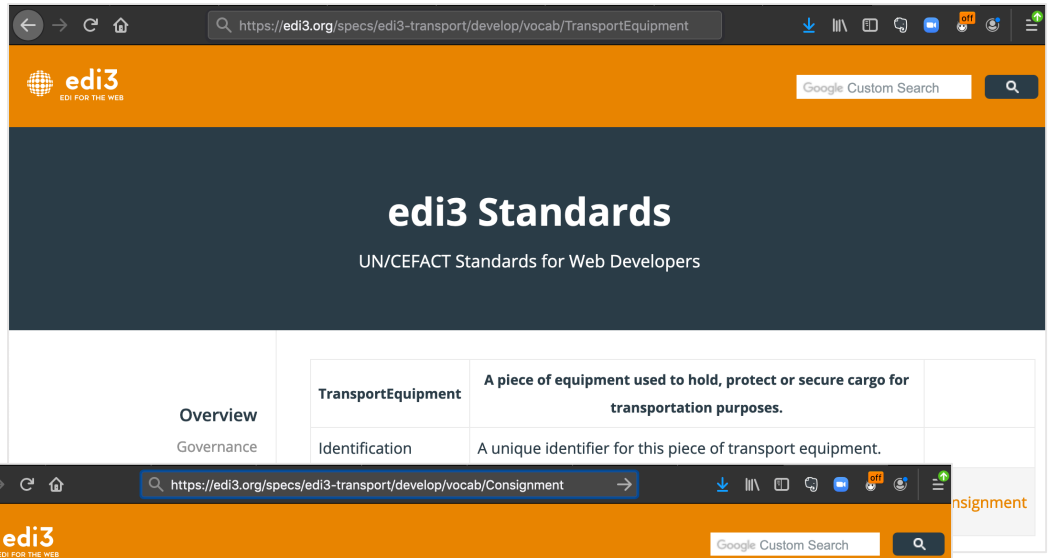
Common semantics are defined in many places throughout the internet. A very relevant example is https://schema.org/ which is extensively used for common stuff like defining Person, Address, Name and many other such fundamental aspects.

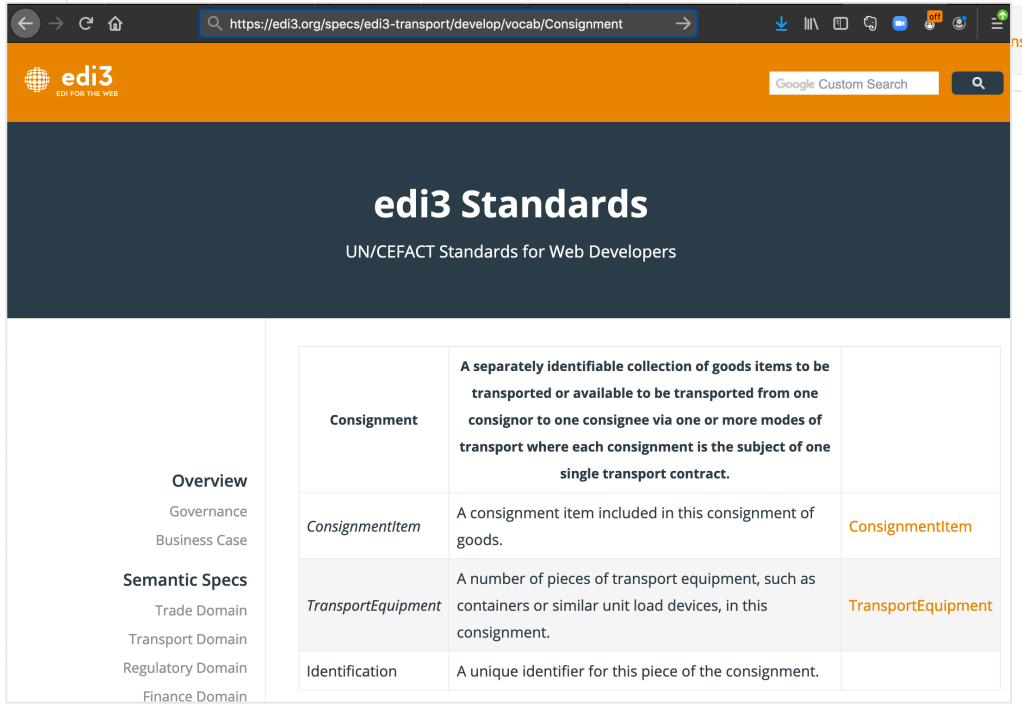
But more specialized semantics typically require specialized governance. In the case of trade and transport, that governance is *us*!

So, an informal initiative by UN/CEFACT experts aims to enable web developers of the world to utilize our semantics, part of the edi3 ambition is to expose our semantics in a referenceable way as a library of IRIs.

I've made some hand-coded examples of this, supporting the earlier examples:

- https://edi3.org/specs/edi3-transport/develop/vocab/Consignment
- https://edi3.org/specs/edi3-transport/develop/vocab/ConsignmentItem
- https://edi3.org/specs/edi3-transport/develop/vocab/TransportEquipment





Thanks…

Get involved on edi3.slack.com and github.com/edi3

Please refer to https://edi3.org/json-ld-ndr/ for more.

And don't hesitate to reach out to me on the edi3 slack or on:

nis.jespersen@maerskgtd.com

github.com/nissimsan/

linkedin.com/in/nis-jespersen-476542

TRADELENS

https://www.tradelens.com/