

MICHEL STORNEBRINK | TNO

Sr. Advisor Semantic Interoperability and Data Spaces

# ▶ Vocabulary Hub for semantic interoperability in data spaces

A concept, method and implementation

▶ **ENDORSE**

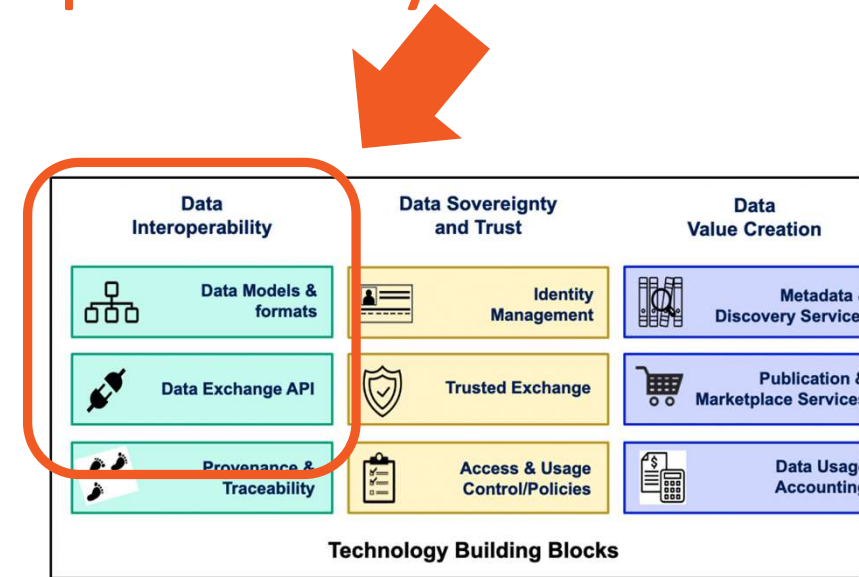
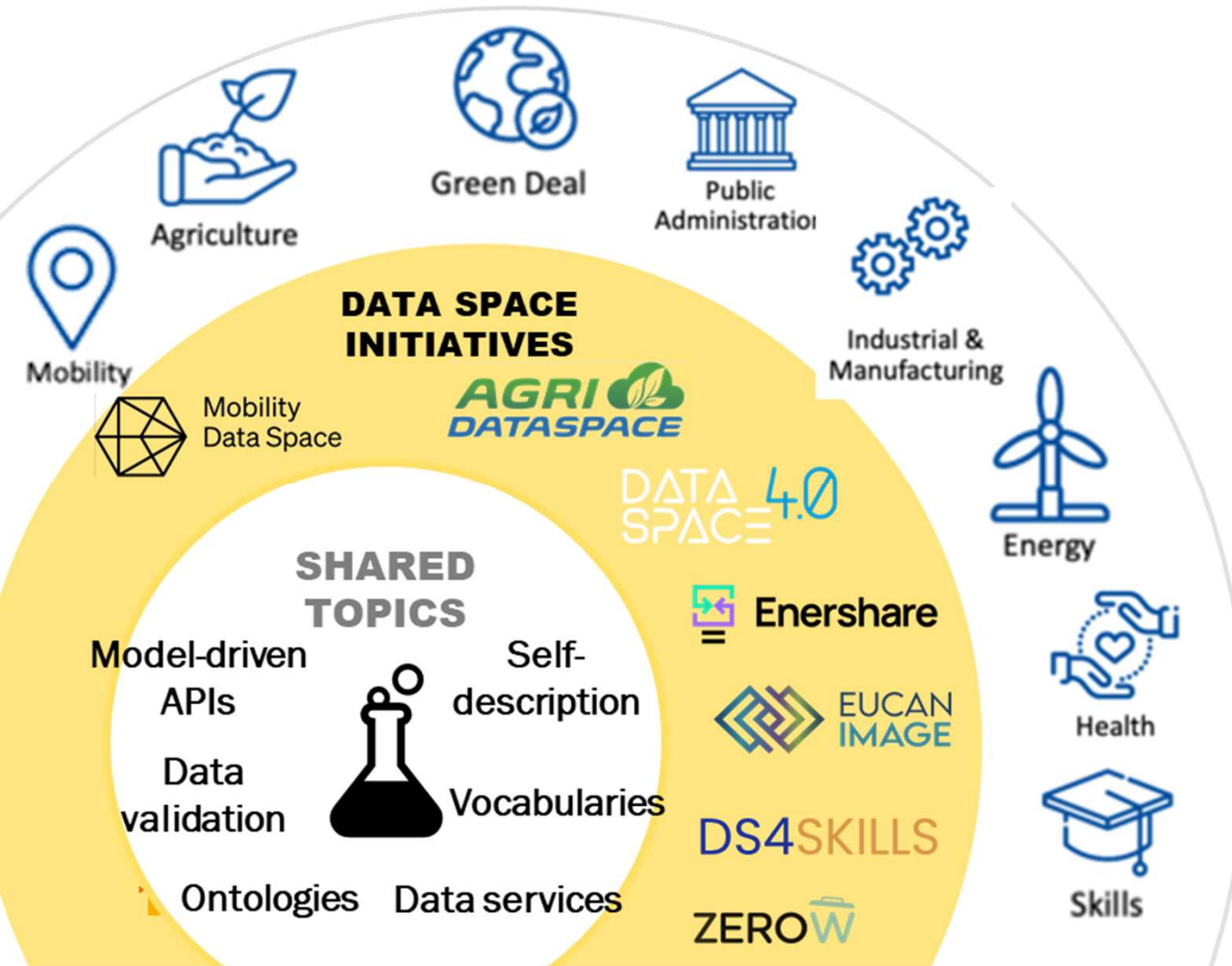
THE EUROPEAN DATA CONFERENCE ON REFERENCE DATA AND SEMANTICS

# 1. Positioning the Vocabulary Hub

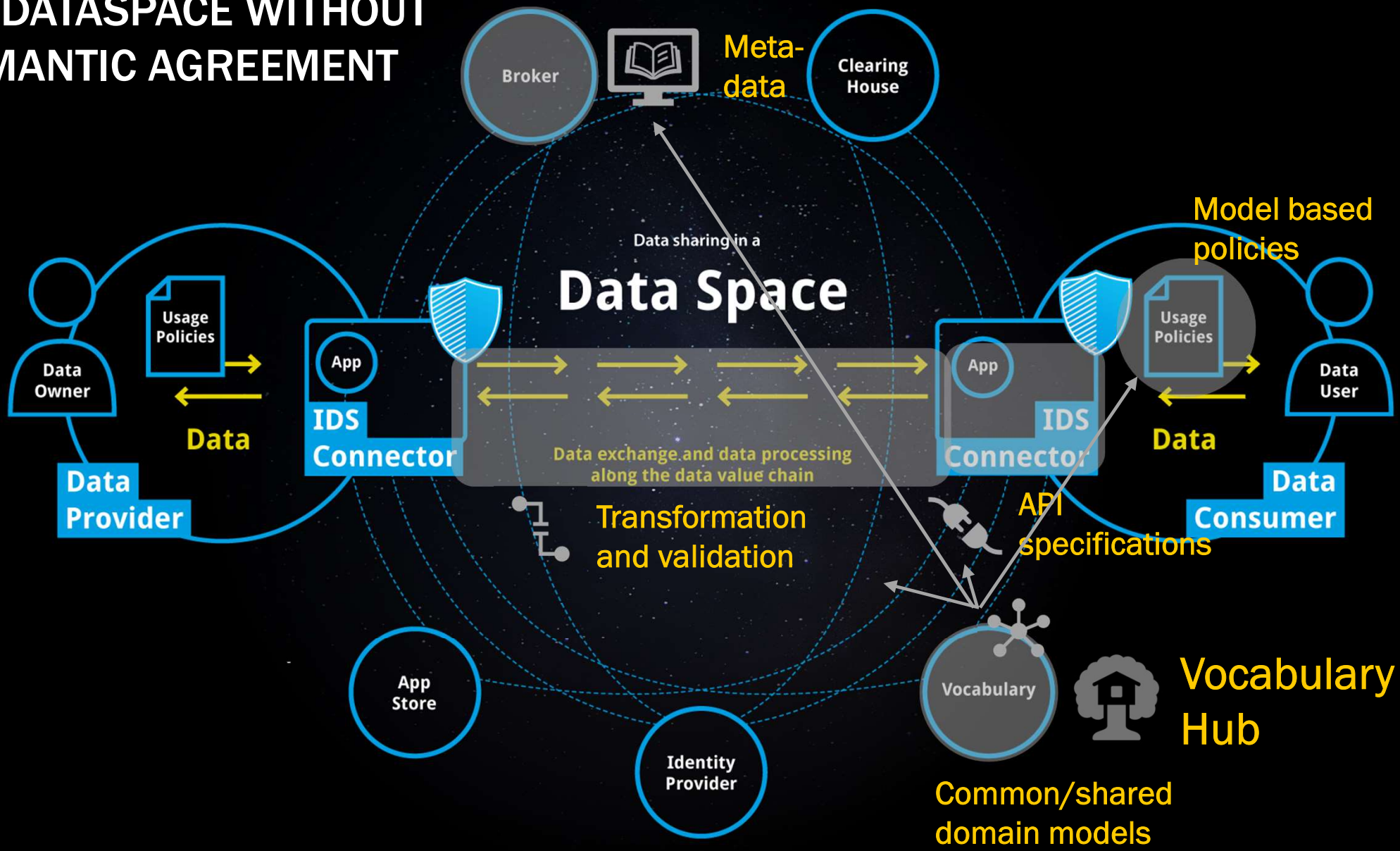
Supporting component for semantic interoperability in data spaces

**ENDORSE**

# Data spaces require data interoperability

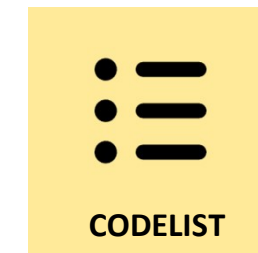
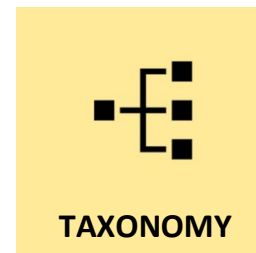
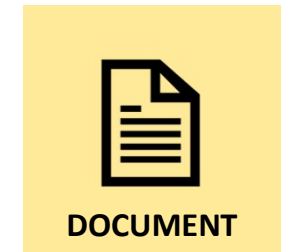
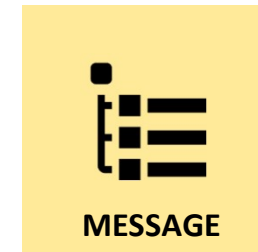
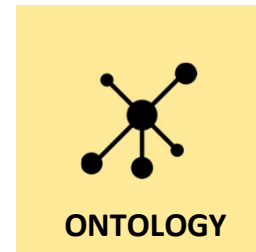


# NO DATASPACE WITHOUT SEMANTIC AGREEMENT



# Broad range of vocabularies and semantic interoperability specifications

- It's about more than ontologies and a glossary
- It includes all (in)formal specifications for semantic interoperability
- Those specifications need to be designed, documented, published, shared and maintained



# Vocabulary Hub explained



Vocabulary Hub

**A registry service providing facilities for publishing, editing, browsing and maintaining vocabularies and related documentation.**

- Vocabularies incl. ontologies, reference data models, schema specifications, mappings and API specifications that can be used to annotate and describe data sets and data services.
- The vocabulary hub can mirror a set of third-party vocabularies ensuring availability and resolution for participants in a data space.

# Related components



Transformation  
Engine

**Service that provides semantic transformation/conversion between data formats.** It uses vocabularies and mapping specification as provided by the vocabulary hub. The component can be integrated at the data consumer or -provider implementation or offered as a service in a dataspace.



Dataspace Connector  
Configurator

**Service to configure the semantic interoperability of dataspace connector implementations.**

- Creating ontology based API-specifications to specify the semantic interface between data provider and -consumer.
- Additionally the dataspace connector configurator can assist in creating mapping specifications if needed. These can be used in the Semantic Transformation Engine.



Semantic Validation  
and Certification

**Validation service to determine if implementations comply to the specifications.**

The service uses formal schema/constraint specification languages to perform the validation steps. Examples of such specification languages are: XML schema, Schematron, JSON schema, SHACL and CSV schema.

## 2. Configuring data space connectors using semantic technology

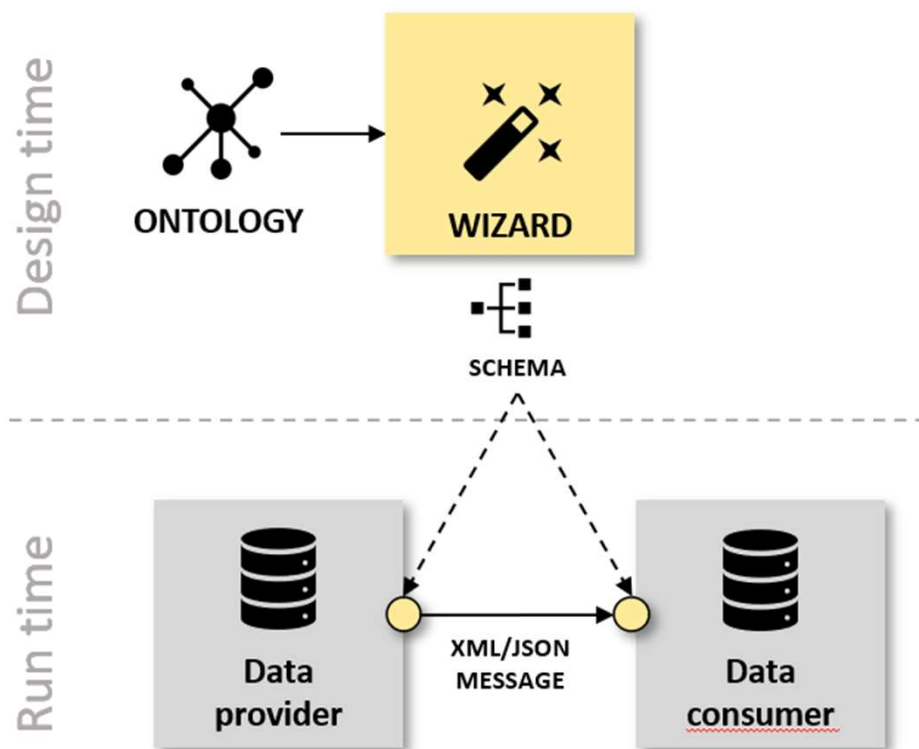
**ENDORSE**



# Overview



Dataspace Connector  
Configurator

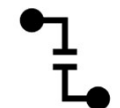


**ENDORSE**

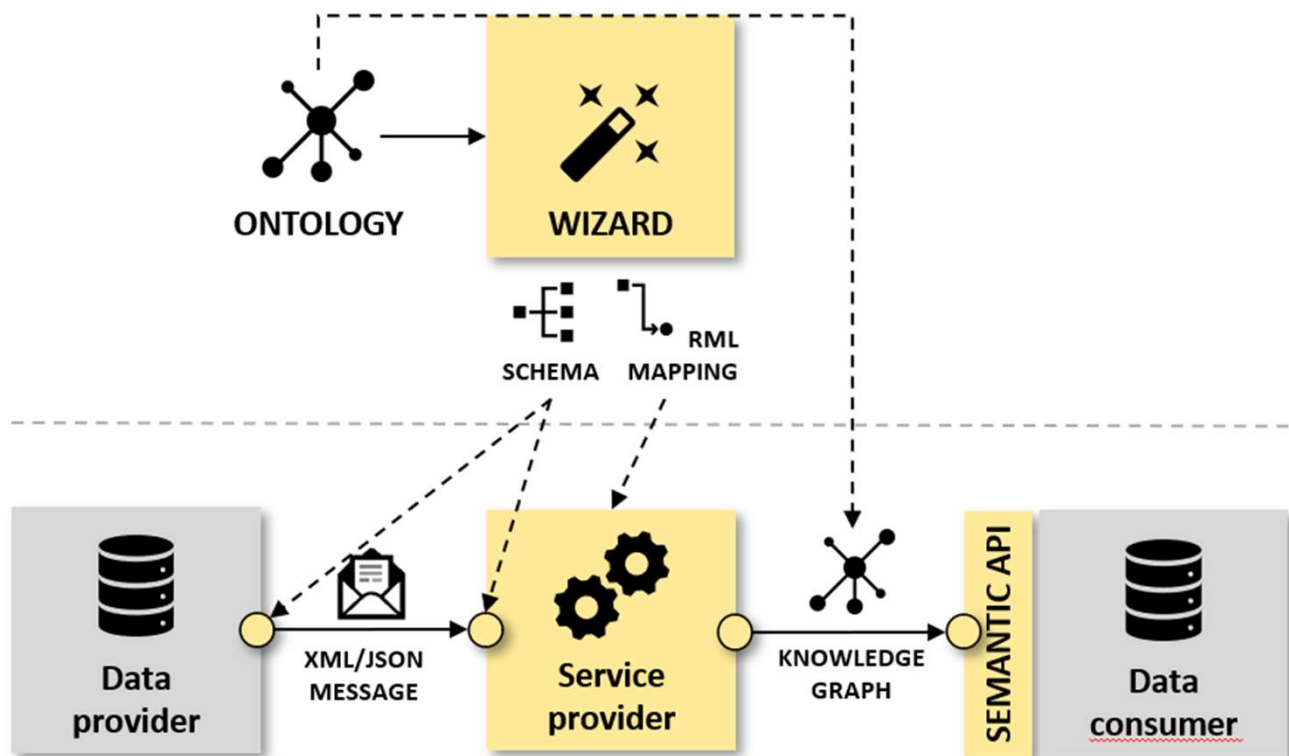
# Scenario with transformation to RDF



Dataspace Connector  
Configurator



Transformation  
Engine

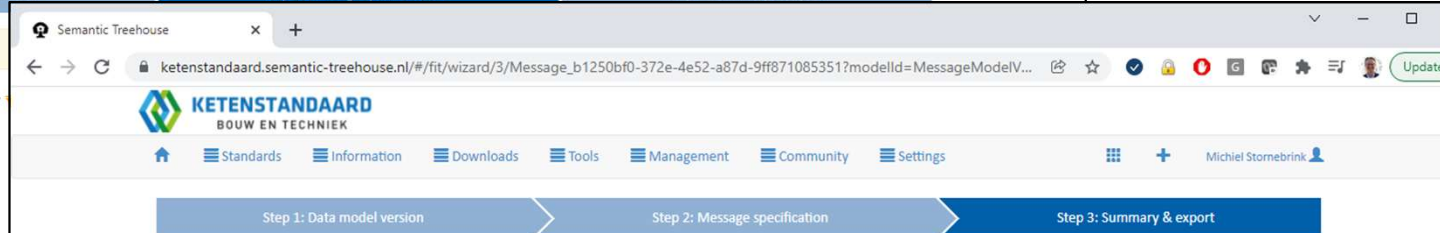
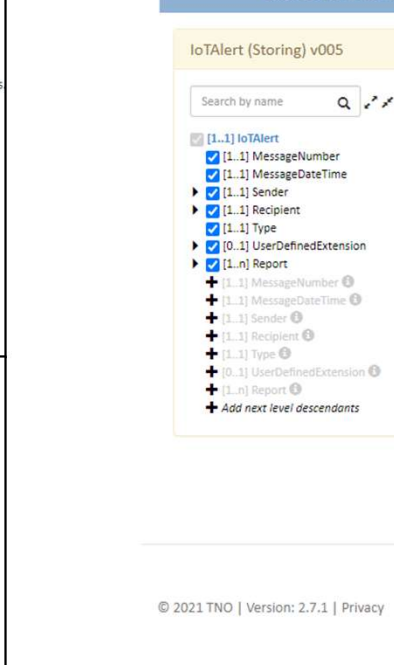
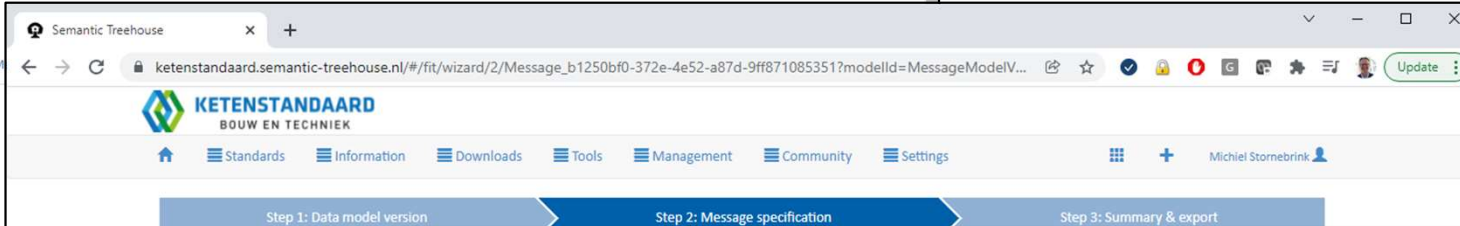
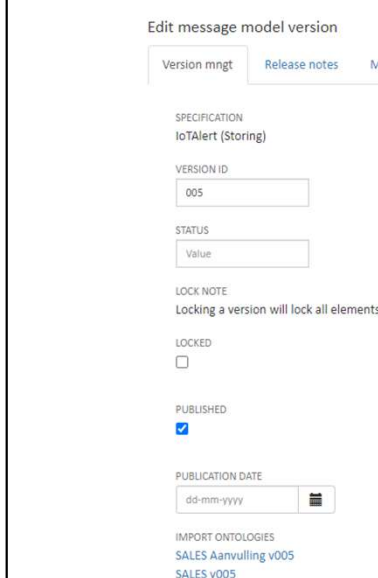
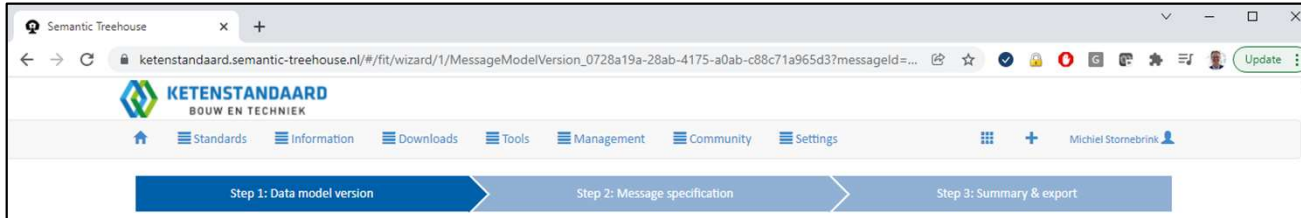


ENDORSE



# FIT Wizard

Ontology based data exchange design  
Syntax independent



Configure your Semantic API using the following output

XML syntax JSON syntax CSV syntax

XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" >
  <!-- Root element definitions -->
  <xs:element name="IoTAlert" type="IoTAlertType" />
  <!-- ComplexType definitions -->
  <xs:complexType name="UserDefinedExtensionType" base="xs:string" />
  <xs:sequence base="UserDefinedExtensionType" />
  <xs:element name="any" minOccurs="0" maxOccurs="1" />
  <xs:annotation base="any" />
  <xs:documentation base="any" />
  <xs:element name="any" minOccurs="0" maxOccurs="1" />
  </xs:sequence>
</xs:schema>
```

RML

```
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
@prefix ql: <http://semweb.mmlab.be/ns/ql#> .
@prefix rr: <http://www.w3.org/ns/r2rml#> .

rml:logicalSource [
  rml:source "http://www.example.com/root" ;
  rml:referenceFormulation ql:XPath ;
  rml:iterator "/IoTAlert" ;
] ;
rr:subjectMap [
  rr:termType rr:BlankNode ;
  rr:class <http://ns.ketenstandaard.nl/sa> ;
] ;
rr:predicateObjectMap [
```

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<IoTAlert xmlns="http://www.ketenstandaard.nl/sa">
  <MessageNumber>some text here</MessageNumber>
  <MessageDateTime>2019-07-30T09:30:10+02:00</MessageDateTime>
  <Sender>
    <GLN>some text here</GLN>
  </Sender>
  <Recipient>
    <GLN>some text here</GLN>
  </Recipient>
  <Type>
  </Type>
  <UserDefinedExtension>
    <any/>
  </UserDefinedExtension>
  <Report>
```

ENDORSE

# Ontology based generated REST APIs



The screenshot shows the Ketenstandaard web application interface. At the top, there is a navigation bar with links for Standards, DICO Information, Downloads, Tools, Management, Community, and Settings. Below this is a progress bar with three steps: Step 1: Data model version, Step 2: Message specification, and Step 3: Summary & export. The main content area is titled "Configure your Semantic API using the following output" and has three tabs: XML syntax, JSON syntax (selected), and CSV syntax. The JSON syntax tab is active, showing a JSON schema for an IoTAlert. The schema includes fields like MessageNumber, MessageDateTime, Sender, Recipient, Type, and Report. To the right of the schema is an "Example" section showing a JSON object with the same structure. Below the schema and example is a Swagger Editor window showing the generated Swagger JSON. The Swagger JSON includes metadata like title, version, license, and a list of API endpoints such as /pet, /pet/findByStatus, and /pet/findByTags. On the far right, there is a preview of the Swagger Petstore API documentation, showing the endpoints and their descriptions.

**JSON schema**

```
id: 'http://www.ketenstandaard.nl/storing...'
$schema: 'https://json-schema.org/draft/2020...
title: 'IoTAlert (Storing) version 005'
description: 'Generated by Semantic Treehous...'
required:
  - MessageNumber
  - MessageDateTime
  - Sender
  - Recipient
  - Type
  - Report
additionalProperties: false
properties:
  MessageNumber:
    type: string
```

**Example**

```
{
  "MessageNumber": "ec2bd1 6d9...",
  "MessageDateTime": "2019-07-...",
  "Sender": {
    "GLN": "d783e0 bcb06"
  },
  "Recipient": {
    "GLN": "11071b 338d"
  },
  "Type": {},
  "UserDefinedExtension": {
    "any": [
      {}
    ]
  }
}
```

**Swagger Editor**

```
1 swagger: "2.0"
2 info:
3   description: "This is a sample server Petstore server. You can find out more about Swagger at [http://swagger.io](http://swagger.io) or on [irc-freenode.net, #swagger](http://swagger.io/irc/). For this sample, you can use the api key 'special-key' to test the authorization filters."
4   version: "1.0.0"
5   title: "Swagger Petstore"
6   termsOfService: "http://swagger.io/terms/"
7   contact:
8     email: "apiteam@swagger.io"
9   license:
10    name: "Apache 2.0"
11    url: "http://www.apache.org/licenses/LICENSE-2.0.html"
12 host: "petstore.swagger.io"
13 basePath: "/v2"
14 tags:
15 - name: "pet"
16   description: "Everything about your Pets"
17 externalDocs:
18   description: "find out more"
19   url: "http://swagger.io"
20 - name: "store"
21   description: "Access to Petstore orders"
22 - name: "user"
23   description: "Operations about user"
24 externalDocs:
25   description: "find out more about our store"
26   url: "http://swagger.io"
27 schemes:
28 - https
29 - http
30 paths:
31 - /pet:
32   post:
33     tags:
34     - "pet"
35     summary: "Add a new pet to the store"
36     description: ""
37     operationId: "addPet"
38     consumes:
39     - "application/json"
40     - "application/xml"
41     produces:
42     - "application/xml"
```

**Swagger Petstore 1.0.0**

[ Base URL: petstore.swagger.io/v2 ]

This is a sample server Petstore server. You can find out more about Swagger at sample, you can use the api key [special-key](#) to test the authorization filters.

Terms of service  
Contact the developer  
Apache 2.0  
Find out more about Swagger

Schemes  
HTTPS

**pet** Everything about your Pets

- POST /pet Add a new pet to the store
- PUT /pet Update an existing pet
- GET /pet/findByStatus Finds Pets by status
- GET /pet/findByTags Finds Pets by tags

## 3. Next steps

**ENDORSE**

# Research & development

- Together with the IDSA community we are elaborating on the Vocabulary Hub functionality
- TNO is open sourcing its implementation: Semantic Treehouse
- Together with industry partners we are applying and experimenting with Vocabulary Hub in different sectoral data spaces

**INTERNATIONAL DATA  
SPACES ASSOCIATION**



**SEMANTIC  
TREEHOUSE**

**TNO** innovation  
for life

**ENDORSE**



Michiel Stornebrink – [michiel.stornebrink@tno.nl](mailto:michiel.stornebrink@tno.nl)

Find our position paper at

<https://www.semantic-treehouse.nl/vocabulary-hub>

