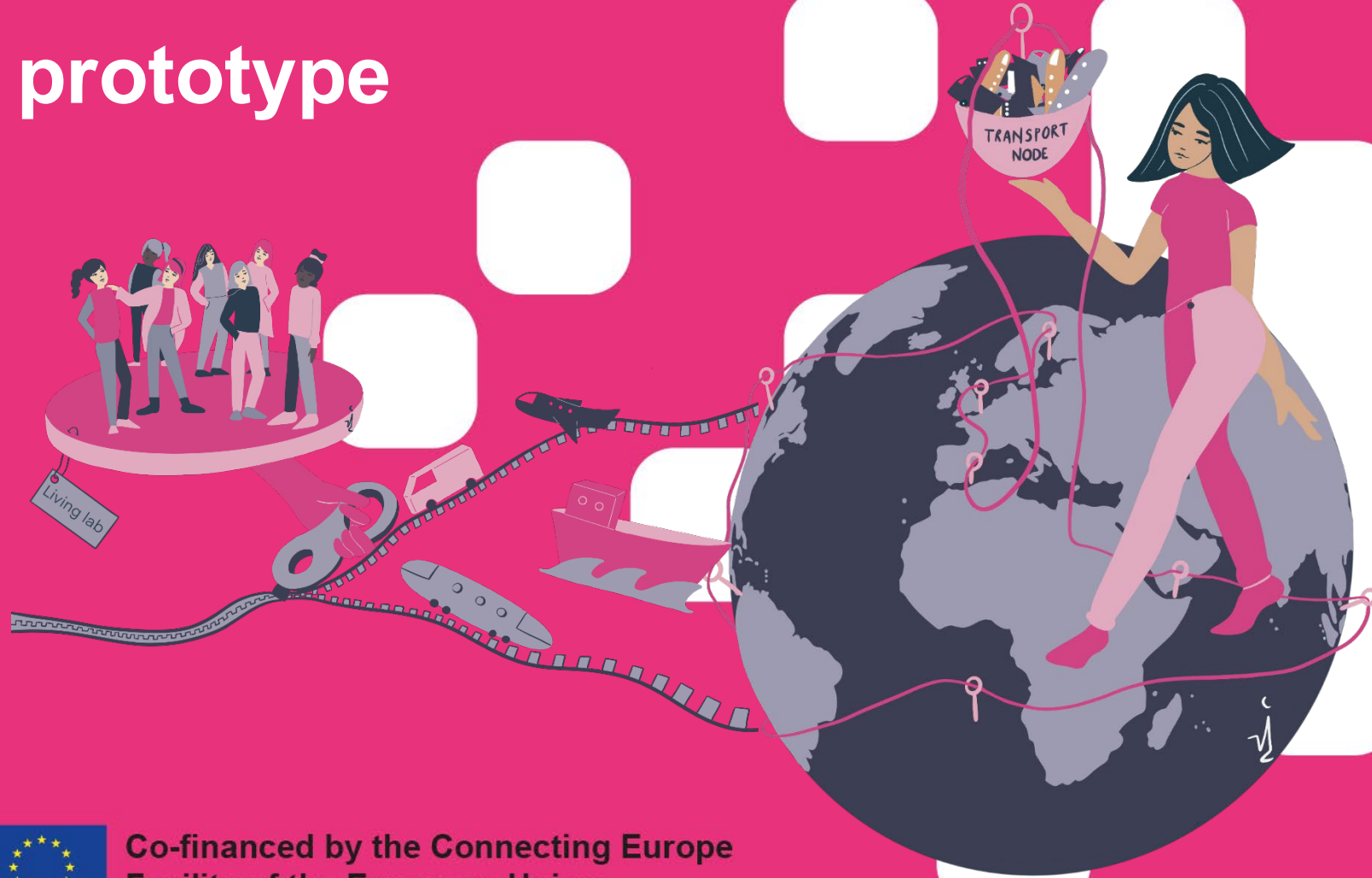


FEDeRATED node prototype development

From concept to implementation

Wout Hofman
(wout.hofman@tno.nl)



Co-financed by the Connecting Europe
Facility of the European Union

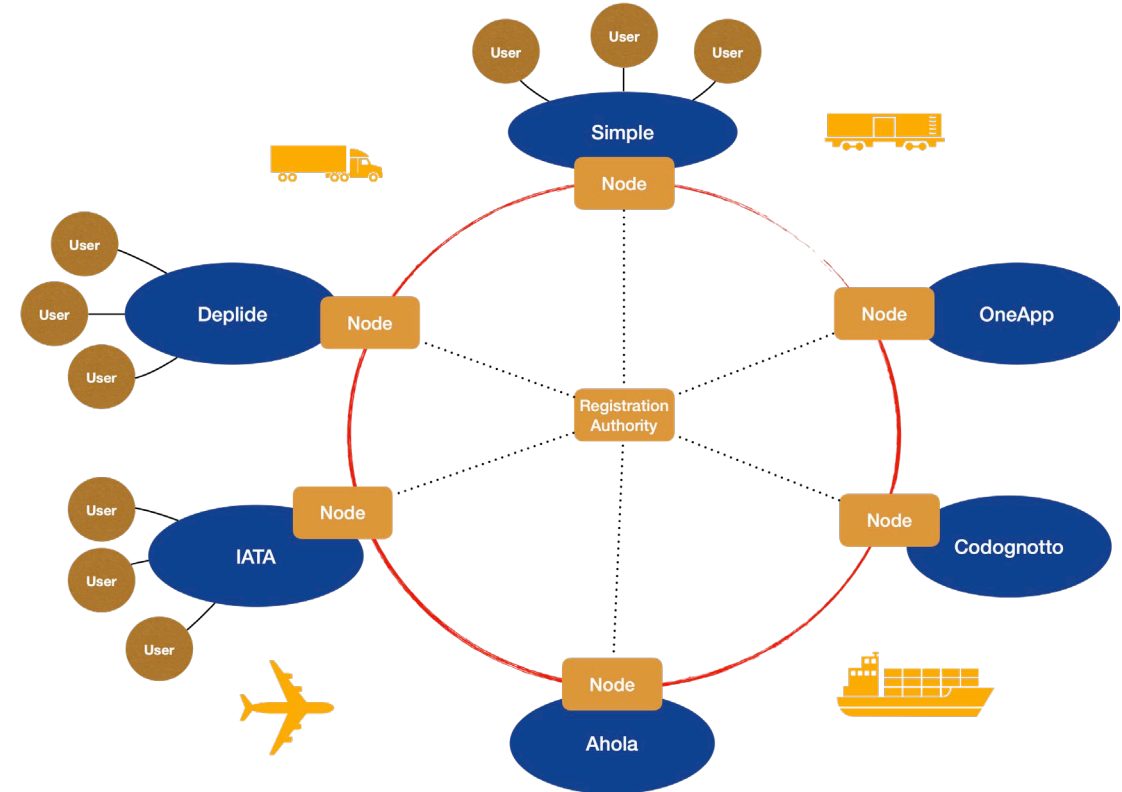
How to become part of the network with a node

Each participant:

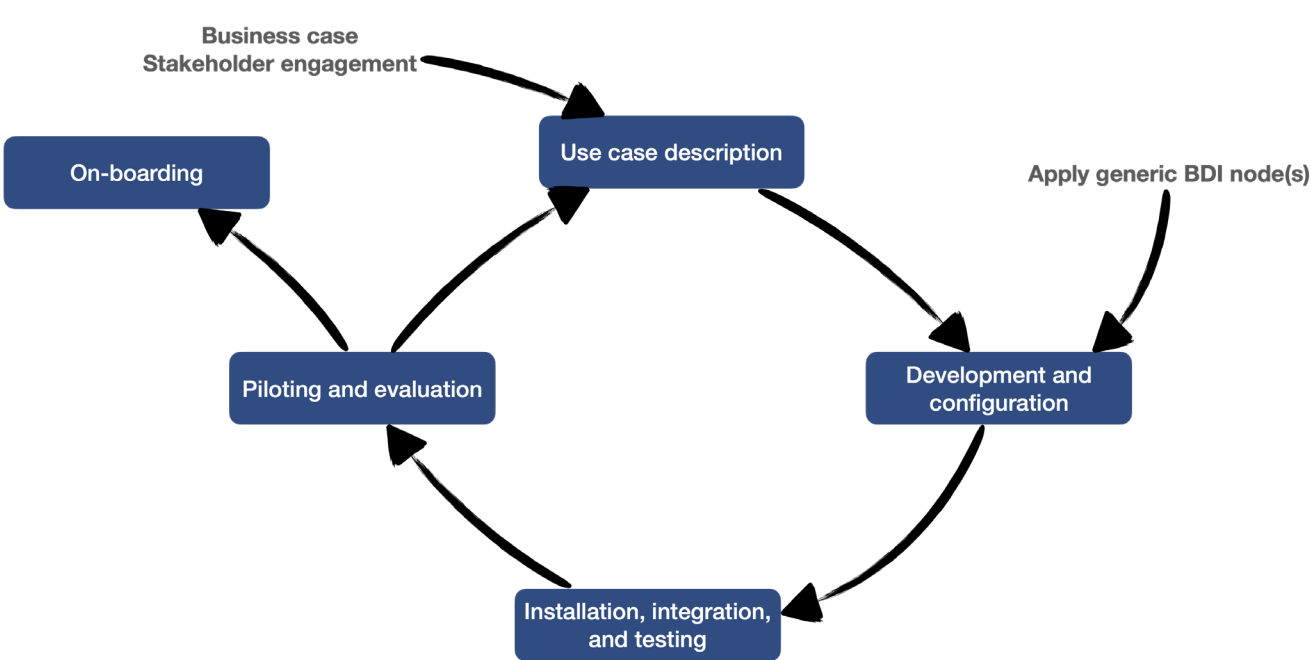
- Downloading and installing a 'node'
- Integrate (open)APIs of a node with internal IT systems

Use case driven Semantic Adapter and APIs:

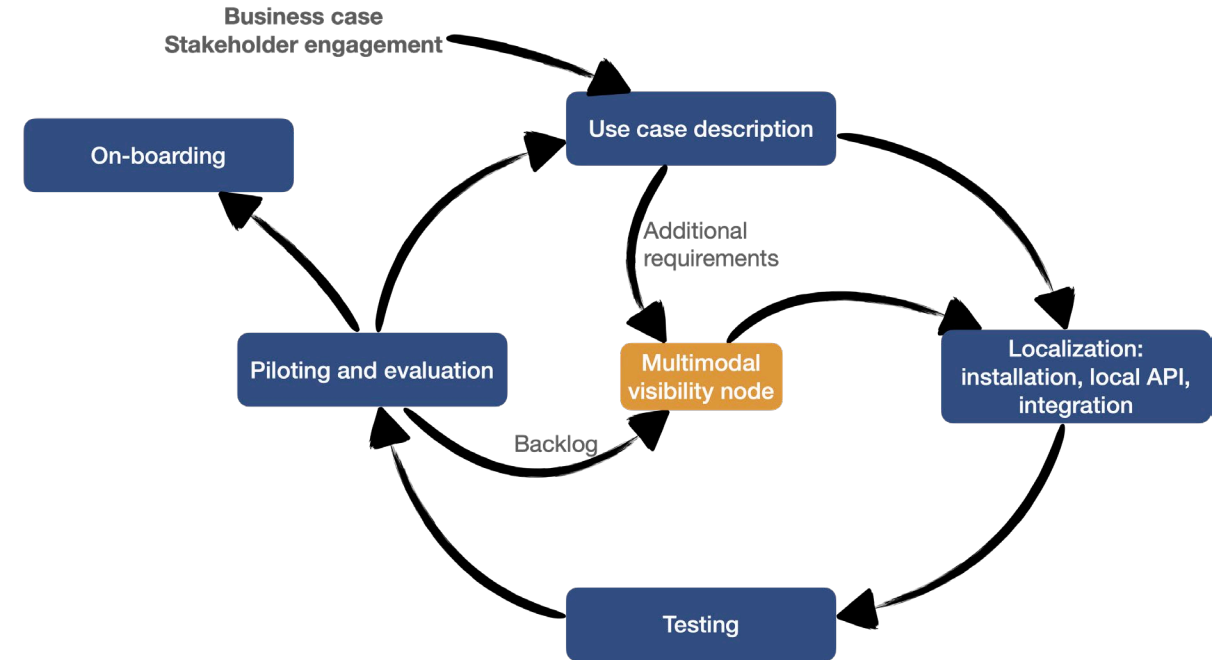
1. Common to all participants for their use case
 - Identical APIs for a use case
 - Use case specific semantic adapter
2. Integration of a common 'multimodal visibility' use case (not yet supported):
 - Localization by each participant (local APIs) to cover its modality/cargo type/...
 - A common multimodal semantic adapter.



From case description to on-boarding



Use case specific node



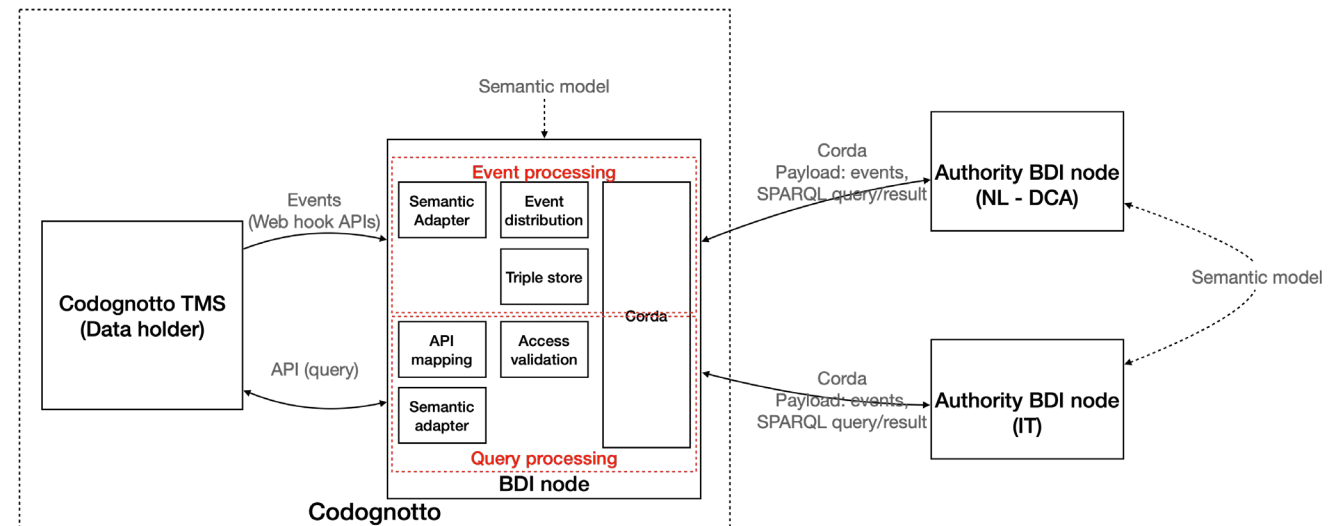
Applying a multimodal visibility node



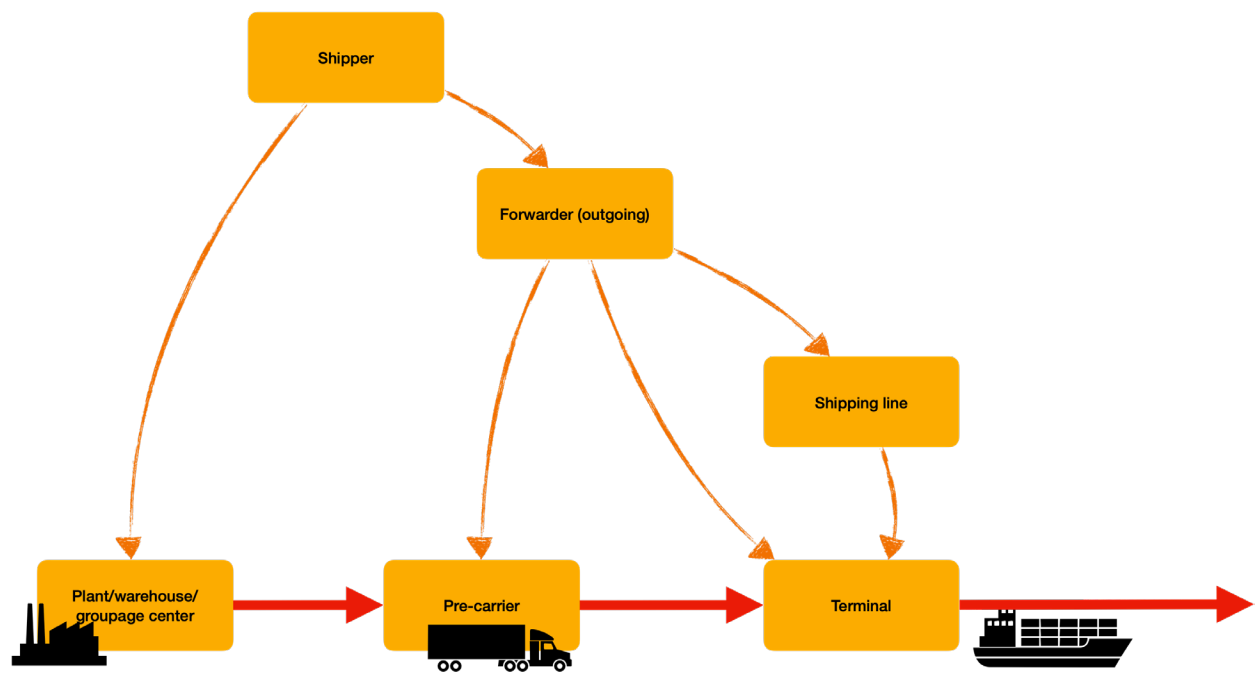
Case description– simple – and complex use cases

- Complex use case (next slide):
 - Hierarchy of commercial relations and compliance (transaction hierarchy)
 - Event sequencing (e.g. UML sequence diagrams) → event logic and - distribution
- Functional specification of events
- Functional specification of linked data (e.g. eCMR data set) → SPARQL query
- Assign data holder/data user roles
- Specify technical setting (systems and platforms to be integrated with a BDI node)

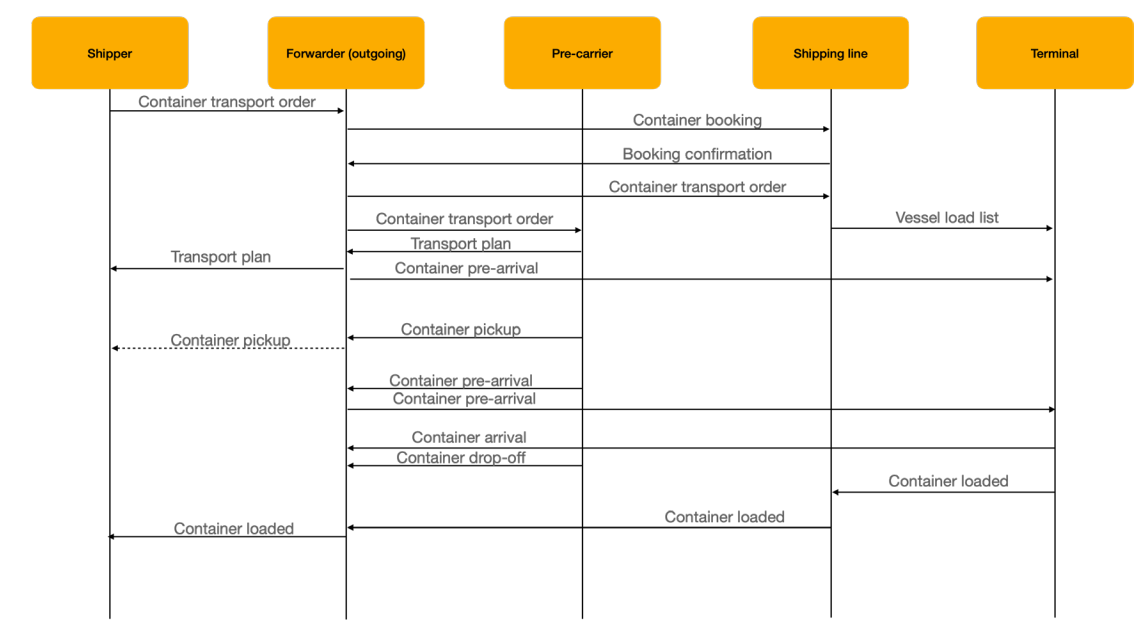
	Trip	Border crossing	Arrival	Load	Arrival	Load	arrival
Event (trip)							
UUID (event)	✓		✓	✓	✓	✓	✓
UUID (sender organisation)	✓		✓	✓	✓	✓	✓
milestone	start / end		start	start	end	start	end
estimated date/time							
actual date/time	✓		✓	✓	✓	✓	✓
UUID (location)	✓		✓ (rail terminal)	✓ (rail terminal)	✓ (ferry terminal (R'dam))	✓ (ferry terminal (R'dam))	✓ (ferry terminal (UK))
External reference	✓		✓				
Reference type	trip number		eCMR data set				
UUID Digital Twin (truck)	✓						
UUID Digital Twin (trailer)	✓		truck or trailer	train	train	ferry	truck or trailer
UUID Digital Twin (goods)	✓		✓	✓	✓	✓	✓
Event (border crossing)		✓					
UUID (event)	✓						
UUID (sender organisation)	✓						
milestone	start	end					
estimated date/time	✓						
actual date/time	✓	✓					
UUID (location)	✓	✓					
Organisation							
UUID organization	✓						
Organization name	✓						
Infrastructure (location)							
UUID	✓						
Name and address (example)	✓						
Digital twin - goods							
UUID Digital Twin	✓						
consignment ID	✓						
item number	✓						
Digital twin - transport means - truck/trailer							
UUID Digital Twin	✓						
License plate	✓						
Country of registration	✓						
Digital Twin - equipment (trailer)							
UUID	✓						
Equipment ID (trailer license plate)	✓						
Event (truck composition)							
UUID (event)	✓						
UUID (sender organisation)	✓						
milestone	start						
estimated date/time							
actual date/time	✓						
UUID Digital Twin (truck)	✓						
UUID Digital Twin (trailer)	✓						



Example of a transaction hierarchy and sequence diagram



Transaction hierarchy



Sequence diagram



Development

- Map functional specifications to the FEDeRATED ontology (initial version of Service Registry)
 - Events and SPARQL queries
 - Generate SHACL (SHACL validation)
 - Generate technical specification openAPIs
 - Generate RML (Rule Markup Language) for mapping JSON to RDF
 - Generate API code and semantic adapter → basis for integration with back office system(s)
- Configure event distribution – change code in API code
- Potentially include event logic (not yet supported) → requires specific code, inclusion of a 'pipeline' like Apache Camel

Slide with screendumps of STH from Theodor specifying the events
Example various steps and example APIcode generated by Stephan



STH FIT Wizard: Step 1

Step 1: Data model version

Step 2: Message specification

Step 3: Summary & export

Edit message model version

Version mngt

Release notes

Message model

Namespaces

Validator syntax

Documentation

Acknowledgements

IMPORT ONTOLOGIES

testfirst v1

Add Ontology/Version

MESSAGE DEFINITION

MESSAGE NAME

LoadEvent

BASED ON CLASS (FULL URI)

https://ontology.tno.nl/logisti

NAMESPACE URI

https://ontology.tno.nl/logisti

PREDECESSOR

Add Element

SYNTAX BINDING (OLD STYLE)

Add Message

MAPPINGS (NEW STYLE)

New MessageMapping



STH FIT Wizard : Step 2

Step 1: Data model version

Step 2: Message specification

Step 3: Summary & export

LoadEvent v

Search by name

☒ [1..1] LoadEvent

- ☒ [1..1] Has Submission Timestamp
- ☒ [0..n] Involves Digital Twin
 - ☒ [1..1] has goods description
 - ☒ [1..1] has goods weight
 - ☒ [1..n] Has Gross Mass
 - ☒ [1..n] has goods type code
 - + [0..n] Dangerous Goods Classificat
 - + [0..1] has gross mass
 - + [0..n] ID
 - + [0..n] billOfLadingNumber
 - + [0..n] Damage Remarks
 - + [0..n] Goods size
 - + [0..n] Gross Weight
 - + [0..n] Harmonized System Code
 - + [0..n] has AWB Number
 - + [0..n] Located At
 - + [0..n] is involved in
 - + Add all
 - + Add next level descendants
- ☒ [0..n] Involves Business Transaction
 - ☒ [0..n] Transport movement type
 - ☒ [0..n] Transport Movement ID
 - ☒ [0..n] Involved actor
 - ☒ [1..n] Actor website
 - ☒ [1..n] Actor address
 - ☒ [1..n] Actor name
- ☒ [1..1] has event type
- ☒ [0..n] Involves Physical Infrastructure
 - ☒ [1..n] City LoCode
 - ☒ [1..n] Postal Code
 - ☒ [1..n] Latitude
 - ☒ [1..n] Longitude
- ☒ [1..1] has milestone
- ☒ [1..1] has date

Involves Digital Twin v

Element Value constraints Usage notes Syntax binding Element mapping Development

Techn. details

LABEL

Involves Digital Twin

ELEMENT NAME

involvesDigitalTwin

NAMESPACE

https://ontology.tno.nl/logistics/federated/Event#

DEFINITION

This relation establishes an association between an Event and a Digital Twin. The exact meaning depends on the type of Event, which is inferred from this property and the second association the event has.

MIN MULTIPLICITY

0

MAX MULTIPLICITY

n

REF ELEMENT TO

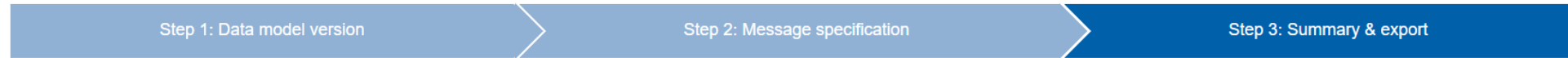
Add Message

SUB ELEMENTS

Visible Sort nr ▲ Min Max Name Is attribute



STH FIT Wizard : Step 3 (Open API Specification)



Configure your Semantic API using the following output

[XML syntax](#)
[JSON syntax](#)
[RDF syntax](#)
[CSV syntax](#)
[OpenAPI Specification syntax](#)

Open API Specification

☒ YAML ☐ JSON

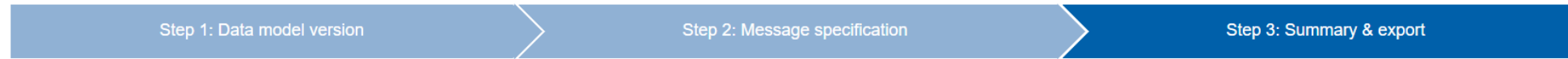
```
openapi: 3.0.0
info:
  title: LoadEvent
  version: undefined
  description: 'This OAS Specification was generated by Semantic Treeh
servers:
  -
    url: 'https://example.org/api/v1'
    description: 'The Example API server.'
paths:
  /LoadEvent:
    post:
      summary: 'Create a resource of type LoadEvent.'
      description: 'Create a resource of type LoadEvent.'
      requestBody:
```

Message Example

```
{
  "hasSubmissionTimestamp": "2023-04-18T16:20:55+02:00",
  "involvesDigitalTwin": [
    {
      "goodsDescription": "b91af5 37562",
      "goodsWeight": 1,
      "hasGrossMass": [
        1
      ],
      "goodsTypeCode": [
        {}
      ]
    }
  ],
  "involvesBusinessTransaction": [
```



STH FIT Wizard : Step 3 (RML Mapping)



Configure your Semantic API using the following output

[XML syntax](#)
[JSON syntax](#)
[RDF syntax](#)
[CSV syntax](#)
[OpenAPI Specification syntax](#)

JSON schema

☒ YAML ☐ JSON

Example

RML

```
id: 'https://ontology.tno.nl/logistics/fec'
$schema: 'https://json-schema.org/draft/2020-12/schema'
title: 'LoadEvent version undefined'
description: 'Generated by Semantic Treehouse'
required:
  - hasSubmissionTimestamp
  - hasTimestamp
  - hasMilestone
  - hasDateTimeType
additionalProperties: false
properties:
  hasSubmissionTimestamp:
    type: string
    format: date-time
  involvesDigitalTwin:
```

```
{
  "hasSubmissionTimestamp": "2023-04-18T16:20:00Z",
  "involvesDigitalTwin": [
    {
      "goodsDescription": "05b05e 68d15",
      "goodsWeight": 1,
      "hasGrossMass": [
        1
      ],
      "goodsTypeCode": [
        {}
      ]
    }
  ],
  "involvesBusinessTransaction": [
```

```
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
@prefix ql: <http://semweb.mmlab.be/ns/ql#> .
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix ns0: <https://ontology.tno.nl/logistics/fec#> .

[]
  rml:logicalSource [
    rml:source "http://www.example.com/root"
    rml:referenceFormulation ql:JSONPath ;
    rml:iterator "$"
  ] ;
  rr:subjectMap [
    rr:termType rr:BlankNode ;
    rr:class ns0:LoadEvent
  ] ;
```



Step 4 – code generation

- OpenAPI code for testing
- Semantic Adapter:
 - JSON → RDF
 - RDF → JSON-LD

Step 5 – containerization (Docker, Kubernetes)



Where to find the code?



<https://github.com/tno/federated-bdi>

- Source code
- Technical documentation
- Unit and integration tests
- Gitlab CI pipeline
- Configured for a demonstration use case

<https://github.com/federated-bdi/docker-bdi-node>

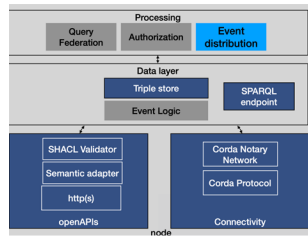
- Docker node

<https://github.com/federated-bdi/Kubernetes-bdi-node>

- Kubernetes node

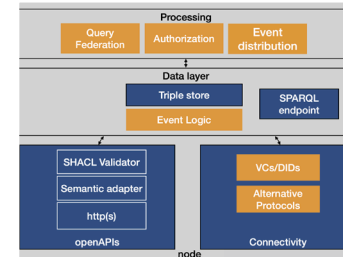


The node functionality



Existing

- openAPIs with internal IT systems
- Sharing of triples (RDF) between nodes (events with links to data)
- Simple semantic adapter (openAPIs to triples)
- Data validation (correctness and completeness according to specifications)
- Data storage with a triple store providing an endpoint for querying (SPARQL)
- Simple data distribution mechanism (type of 'smart contract')
- Connectivity: Corda-based
- On-boarding: Corda network manager
- Non-repudiation (log/audit trail): Corda Notary Network



Planned

- Improved data distribution mechanism
- Authorization based on links
- Access policy evaluation based on semantic model
- Event logic to validate states
- Query federation (data provenance)
- Support of other connectivity protocols
- VCs/DIDs for on-boarding
- Other types of non-repudiation (log/audit trail)
- Improved semantic adapter

