



Project Acronym: **DELTA**

Project Full Title: **Future tamper-proof Demand rEsponse framework through seLf-configured, self-opTimized and collAborative virtual distributed energy nodes**

Grant Agreement: **773960**

Project Duration: **36 months (01/05/2018 – 30/04/2021)**

DELTA ONTOLOGY AND DATA MODELLING FRAMEWORK SPECIFICATION D1.3_v1

Work Package **WP1 – Delta Requirements & System Architecture
Definition**

Task **T1.3 – DELTA Ontology and Data Modelling Framework
Definition**

Document Status: **Final**

File Name: **DELTA_D1.3_v1.0.docx**

Due Date: **April 2019**

Lead Beneficiary: **UPM**

Dissemination Level

Public X

Confidential, only for members of the Consortium (including the Commission Services)

Authors List

Leading Author				
First Name		Last Name	Beneficiary	Contact e-mail
Alba		Fernández-Izquierdo	UPM	albafernandez@fi.upm.es
Co-Author(s)				
#	First Name	Last Name	Beneficiary	Contact e-mail
1	Andrea	Cimmino	UPM	cimmino@fi.upm.es
2	Raúl	García-Castro	UPM	rgarcia@fi.upm.es
3	María	Poveda-Villalón	UPM	mpoveda@fi.upm.es
4	Sofia	Terzi	CERTH	sterzi@iti.gr
5	Christos	Patsonakis	CERTH	cpatsonakis@iti.gr

Reviewers List

Reviewers			
First Name	Last Name	Beneficiary	Contact e-mail
Christof	Amann	e7	christof.amannqe-sieben.at
Venizelos	Efthymiou	FOSS	Efthymiou.venizelos@ucy.ac.cy
Ian	Cole	FOSS	cole.ian@ucy.ac.cy
Ioannis	Moschos	CERTH	imoschos@iti.gr

Legal Disclaimer

The DELTA has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 773960. The sole responsibility for the content of this publication lies with the authors. It does not necessarily reflect the opinion of the Innovation and Networks Executive Agency (INEA) or the European Commission (EC). INEA or the EC are not responsible for any use that may be made of the information contained therein.

Copyright

© Universidad Politécnica de Madrid, Madrid, Spain. Copies of this publication – also of extracts thereof – may only be made with reference to the publisher.

Executive Summary

This document describes the first release of the DELTA ontology and the Common Information Model, which is the component responsible of the data exchange among different main DELTA components. Additionally, the JSON-LD documents associated to the DELTA interfaces for each component, which are describe to exchange data between components in the DELTA platform have been also specified, including some example of use to clarify the data exchange. Such JSON-LD interfaces are designed around the DELTA ontology context to link the terms in the JSON-LD document to concepts in the DELTA ontology.

An analysis of state-of-the-art standards and ontologies dealing with to Demand-Response is presented to analyse how other institutions define Demand-Response platforms, as well as to analyse possible reuse of standards and alignment with already published Demand-Response ontologies.

In this document the presented DELTA ontology is focused on satisfying the requirements of the first release of the DELTA architecture. The ontology and the artefacts produced during the ontology development are available online and continuously improving. If new requirements appear in the DELTA architecture, they will be included in the DELTA ontology.

Finally, the OpenADR DELTA Repository CIM is presented, which implements the requirements specified in the OpenADR standard to exchange information between different DELTA sub-components and the DELTA Repository.

Table of Contents

1. Introduction	11
1.1 Scope and objectives of the deliverable.....	11
1.2 Structure of the deliverable.....	11
1.3 Relation to Other Tasks and Deliverables	12
2. Relevant standards and ontologies for DELTA	13
2.1 Standards.....	13
2.1.1 OpenADR	13
2.1.2 Smart Grid Architecture Model (SGAM)	13
2.1.3 IEC CIM	13
2.1.4 Facility Smart Grid Information Model (FSGIM)	14
2.1.5 Enery@home	14
2.1.6 NAESB Energy Usage Information Model	14
2.1.7 OASIS Energy Market Information Exchange (eMIX).....	14
2.1.8 USEF (Universal Smart Energy Framework).....	15
2.1.9 Energy Flexibility Interface (EFI)	15
2.1.10 IEC 62746	15
2.1.11 IEC/EN 61970	15
2.1.12 IEC 62056 COSEM.....	15
2.1.13 CENELEC EN 50491-11 Smart Metering	16
2.1.14 CENELEC EN 50631-1 (SPINE)	16
2.1.15 CENELEC EN 50090 (KNX)	16
2.1.16 CEN EN 16836 (ZigBee SEP2)	17
2.2 Ontologies	17
2.2.1 SAREF4ENER	17
2.2.2 MAS2TERING	17
2.2.3 OEMA Ontology Network	17
2.2.4 CIM ontology for Smart Grids	18
2.2.5 OntoENERGY	18
2.2.6 ThinkHome	18
2.2.7 BOnSAI	18
2.2.8 ProSGV3.....	18
2.2.9 Mirabel ontology	19
2.2.10 SEMANCO	19
2.2.11 Sesame demonstrator.....	19
3. Common Information Model Requirements	20
3.1 Information Model.....	20
3.1.1 Smart Meters.....	20
3.1.2 Assets/DERs	20
3.1.3 Building Information Model (BIM).....	21
3.1.4 User comfort constraints.....	21
3.1.5 Gamification	21
4. Ontology development methodology	22
4.1 Ontological requirements specification.....	22
4.1.1 Purpose and scope identification	23
4.1.2 Data exchange identification	23
4.1.3 Ontological requirements proposal.....	23
4.1.4 Ontological requirements completion and validation.....	23
4.1.5 Ontological requirements prioritization.....	23

4.2	Ontology implementation	23
4.2.1	Ontology conceptualization.....	24
4.2.2	Ontology implementation	24
4.2.3	Ontology evaluation	24
4.3	Ontology publication	25
4.3.1	Documentation generation.....	25
4.3.2	Propose release candidate	25
4.3.3	Online publication	25
4.4	Ontology maintenance	25
4.4.1	Bug detection.....	26
4.4.2	New requirements proposal	26
5.	Ontology development infrastructure	5-27
5.1	Infrastructure to support the requirements specification activity	5-27
5.2	Infrastructure to support the ontology implementation activity	5-28
5.3	Infrastructure to support the ontology publication activity	5-29
5.3.1	Ontologies.....	5-29
5.3.2	Ontology testing	5-29
5.3.3	How we work.....	5-30
5.4	Infrastructure to support the ontology maintenance activity	5-30
6.	DELTA ontology overview	6-32
7.	Specification of DELTA data interfaces.....	7-35
7.1	DELTA Virtual Node	7-35
7.1.1	Consumer/Prosumer Flexibility Data Monitoring and Profiling	7-35
7.1.2	Generation/Consumption Optimal Dispatch.....	7-38
7.1.3	Load Forecasting	7-40
7.1.4	Inter/Intra Node Energy Matchmaking.....	7-41
7.1.5	Consumer/Prosumer Clustering.....	7-43
7.2	Aggregator	7-46
7.2.1	Grid Stability Simulation Engine	7-46
7.2.2	Self Portfolio Energy Balancing	7-47
7.2.3	Energy Portfolio Segmentation & Classification.....	7-49
7.2.4	Energy Market Price Forecast	7-52
7.2.5	DR & Flexibility Forecasting	7-53
7.2.6	Asset Handling Optimization	7-56
7.2.7	Node Flexibility Data Monitoring and Profiling	7-58
7.3	DELTA Fog Enable Agent	7-59
7.3.1	Lightweight Toolkit (FEID) per Customer	7-59
7.4	Demand Response Settings to market stakeholders	7-64
7.4.1	DR Services to market stakeholders	7-64
7.5	Innovative Customer Engagement Tools.....	7-65
7.5.1	Demand Response Visualization Kit	7-65
7.5.2	Social Interaction and Cooperation	7-71
7.5.3	Award-enabled Energy Behavioral Platform.....	7-72
7.6	DELTA Cyber Security Services	7-73
7.6.1	DELTA Blockchain	7-73
8.	Common Information Model	8-75
8.1	CIM and OpenADR.....	8-76
8.2	CIM communication protocols	8-77
8.3	OpenADR Aggregator CIM.....	8-78

8.4	OpenADR Virtual Node CIM	8-79
8.5	OpenADR FEID CIM.....	8-80
8.6	OpenADR Repository CIM.....	8-81
8.7	OpenADR Security Services CIM	8-82
8.8	Developing and Deploying CIM sub-components.....	8-82
9.	Conclusions	9-83
10.	References	10-84
Annex 1	10-85
A-1.	Aggregated profiling	10-85
A-2.	Bids.....	10-88
A-3.	Comfort settings	10-93
A-4.	Customer information	10-95
A-5.	DR signals	10-97
A-6.	DVN Clusters.....	10-100
A-7.	Energy Price profiling	10-102
A-8.	FEIDs Clusters	10-107
A-9.	Flexibility forecast.....	10-110
A-10.	Forecasted profiling	10-113
A-11.	Historical consumption.....	10-117
A-12.	Historical generation	10-120
A-13.	KPIs.....	10-123
A-14.	Market settlement	10-125
A-15.	Node Profiling.....	10-127
A-16.	Rewards	10-131
A-17.	Smart contracts	10-133
A-18.	Status signals	10-138
A-19.	System constraints	10-140
A-20.	Transactions	10-142
A-21.	Voltage & Frequency	10-144

List of Figures

Figure 1	Relation between DELTA platform and DELTA ontology	11
Figure 2	DELTA ontology development process (García-Castro et al., 2017)	22
Figure 3	Workflow for ontology requirements specification	22
Figure 4	Workflow for ontology implementation (García-Castro et al., 2017)	24
Figure 5	Workflow for ontology publication (García-Castro et al., 2017).....	25
Figure 6	Workflow for ontology maintenance.....	26
Figure 7	DELTA ontology requirements in a Google Spreadsheet	5-27
Figure 8	Excerpt of the HTML documentation of the requirements	5-28

Figure 9 DELTA ontology portal.....	5-29
Figure 10 Ontology testing section.....	5-30
Figure 11 GitHub issue tracker.....	5-31
Figure 12 Ontology overview	6-33
Figure 13 DELTA Signals.....	6-33
Figure 14 Interfaces for Consumer/Prosumer Flexibility Data Monitoring and Profiling	7-35
Figure 15 Ontology concepts related to Flexibility forecast	7-36
Figure 16 Ontology concepts related to Historical Consumption.....	7-36
Figure 17 Ontology concepts related to Historical Generation.....	7-37
Figure 18 Ontology concepts related to Node Profiling.....	7-37
Figure 19 Ontology concepts related to the Voltage & Frequency Constraints.....	7-38
Figure 20 Interfaces for Generation/Consumption Optimal Dispatch	7-38
Figure 21 Ontology concepts related to the FEIDs Clusters	7-39
Figure 22 Ontology concepts related to DR Signals.....	7-39
Figure 23 Ontology concepts related to DELTA Node Profiling Interface	7-39
Figure 24 Ontology concepts related to Forecasted Profiling Interface	7-40
Figure 25 Ontology concepts related to Transactions.....	7-40
Figure 26 Interfaces for Load Forecasting	7-40
Figure 27 Ontology concepts related to Node Profiling.....	7-41
Figure 28 Ontology concepts related to Forecasted Profiling	7-41
Figure 29 Interfaces for Inter/Intra Node Energy Matchmaking	7-42
Figure 30 Ontology concepts related to the FEIDs Clusters	7-42
Figure 31 Ontology concepts related to DVN Clusters	7-42
Figure 32 Ontology concepts related to Status Signal	7-43
Figure 33 Ontology concepts related to DR Signal	7-43
Figure 34 Interfaces for Consumer/Prosumer Clustering.....	7-44
Figure 35 Ontology concepts related to DVN Clusters	7-44

Figure 36 Ontology concepts related to KPIs	7-44
Figure 37 Ontology concepts related to DR Signal	7-45
Figure 38 Ontology concepts related to Node Profiling.....	7-45
Figure 39 Ontology concepts related to the FEIDs Clusters	7-46
Figure 40 Interfaces for Grid Stability Simulation Engine	7-46
Figure 41 Ontology concepts related to the Aggregated Profiling.....	7-46
Figure 42 Ontology concepts related to the Voltage & Frequency Constraints.....	7-47
Figure 43 Interfaces for the Self Portfolio Energy Balancing.....	7-47
Figure 44 Ontology concepts related to Forecasted Flexibility.....	7-48
Figure 45 Ontology concepts related to the Aggregated Profiling.....	7-48
Figure 46 Ontology concepts related to Energy Price Profiling	7-49
Figure 47 Ontology concepts related to Bids	7-49
Figure 48 Ontology concepts related to the Market Settlement	7-49
Figure 49 Interfaces for the Energy Portfolio Segmentation & Classification.....	7-50
Figure 50 Ontology concepts related to the Customers Information	7-50
Figure 51 Ontology concepts related to the Rewards	7-51
Figure 52 Ontology concepts related to Forecasted Flexibility.....	7-51
Figure 53 Ontology concepts related to the Aggregated Profiling.....	7-52
Figure 54 Ontology concepts related to DVN Clusters.....	7-52
Figure 55 Interfaces for Energy Market Price Forecast	7-52
Figure 56 Ontology concepts related to the Voltage & Frequency Constraints Interface	7-53
Figure 57 Ontology concepts related to Energy Price Profile.....	7-53
Figure 58 Interfaces for DR & Flexibility Forecasting	7-54
Figure 59 Ontology concepts related to the Aggregated Profile	7-54
Figure 60 Ontology concepts related to the Forecasted profiling.....	7-55
Figure 61 Ontology concepts related to the Customers Information	7-55
Figure 62 Ontology concepts related to Forecasted Flexibility	7-56

Figure 63 Interfaces for the Assets Handling Optimization.....	7-56
Figure 64 Ontology concepts related to DR Signal	7-57
Figure 65 Ontology concepts related to the Aggregate Profiling.....	7-57
Figure 66 Ontology concepts related to Transactions.....	7-58
Figure 67 Ontology concepts related to the Market Settlement	7-58
Figure 68 Interfaces for the Node Flexibility Data Monitoring and Profiling	7-58
Figure 69 Ontology concepts related to DELTA Node Profiling	7-59
Figure 70 Ontology concepts related to the Aggregated Profile	7-59
Figure 71 Interfaces for Lightweight Toolkit per Customer.....	7-60
Figure 72 Ontology concepts related to Comfort Settings.....	7-60
Figure 73 Ontology concepts related to System Constraints.....	7-60
Figure 74 Ontology concepts related to Smart Contracts	7-61
Figure 75 Ontology concepts related to DR Signal	7-61
Figure 76 Ontology concepts related to Status Signal	7-62
Figure 77 Ontology concepts related to Flexibility Forecast.....	7-62
Figure 78 Ontology concepts related to Historical Consumption.....	7-62
Figure 79 Ontology concepts related to Historical Generation.....	7-63
Figure 80 Ontology concepts related to the Voltage & Frequency Constraints.....	7-63
Figure 81 Ontology concepts related to Transactions.....	7-64
Figure 82 Interfaces for DR Services to market stakeholders	7-64
Figure 83 Ontology concepts related to Bids	7-64
Figure 84 Ontology concepts related to DR Signal	7-65
Figure 85 Interfaces for the Demand Response Visualization Kit (Aggregator Level)	7-66
Figure 86 Interfaces for the Demand Response Visualization Kit (Customer Level)...	7-66
Figure 87 Ontology concepts related to the Customers Information	7-67
Figure 88 Ontology concepts related to Historical Consumption.....	7-67
Figure 89 Ontology concepts related to Historical Generation.....	7-68

Figure 90 Ontology concepts related to Flexibility forecast	7-68
Figure 91 Ontology concepts related to DR Signal	7-69
Figure 92 Ontology concepts related to Bids	7-69
Figure 93 Ontology concepts related to the Rewards	7-70
Figure 94 Ontology concepts related to Energy Price Profiling	7-70
Figure 95 Ontology concepts related to DVN Clusters	7-70
Figure 96 Ontology concepts related to Node Profiling.....	7-71
Figure 97 Ontology concepts related to the Aggregated Profiling.....	7-71
Figure 98 Interfaces for the Social Interaction and Cooperation.....	7-72
Figure 99 Ontology concepts related to the Rewards	7-72
Figure 100 Interfaces for the Award-enabled Energy Behavioral Platform	7-72
Figure 101 Ontology concepts related to DVN Clusters.....	7-73
Figure 102 Ontology concepts related to the Rewards	7-73
Figure 103 Interfaces for DELTA Blockchain	7-74
Figure 104 Ontology concepts related to Smart Contracts	7-74
Figure 105 Ontology concepts related to Transactions.....	7-74

List of Tables

Table 1 – OpenADR specification.....	8-76
---	-------------

List of Acronyms and Abbreviations

Term	Description
VPP	Virtual Power Plant
OWL	Web Ontology Language
DVN	Delta Virtual Node
API	Application Programming Interface
DR	Demand Response
CIM	Common Information Model

1. Introduction

1.1 Scope and objectives of the deliverable

The DELTA interoperability approach relies on ontologies (i.e., semantic data models) that will be exploited throughout the DELTA infrastructure. In computer science, ontologies are defined as “formal, explicit specifications of a shared conceptualization”. The DELTA ontologies will be implemented in the W3C Web Ontology Language standard OWL¹ reusing existing resources and standards whenever possible. The conceptualization will be shared among the DELTA components being of interest for every DELTA component. This document will describe the process followed to build such ontologies and the resulting models in detail.

The goal of this deliverable is to detail the DELTA ontologies developed for the release 0.1 and the DELTA Common Information Model (CIM). The scope of this deliverable includes the description of the resulting ontologies, the processes followed, the infrastructure deployed during the ontology development, the interfaces for data exchange following this ontology and the description of the CIM. Such CIM, as shown in Figure 1, relies on the DELTA ontology to create the interfaces that defines the data that is going to be exchanged between the DELTA components. In addition, some standard and ontologies are reviewed for reused and examples of how to use the developed DELTA ontology are provided.

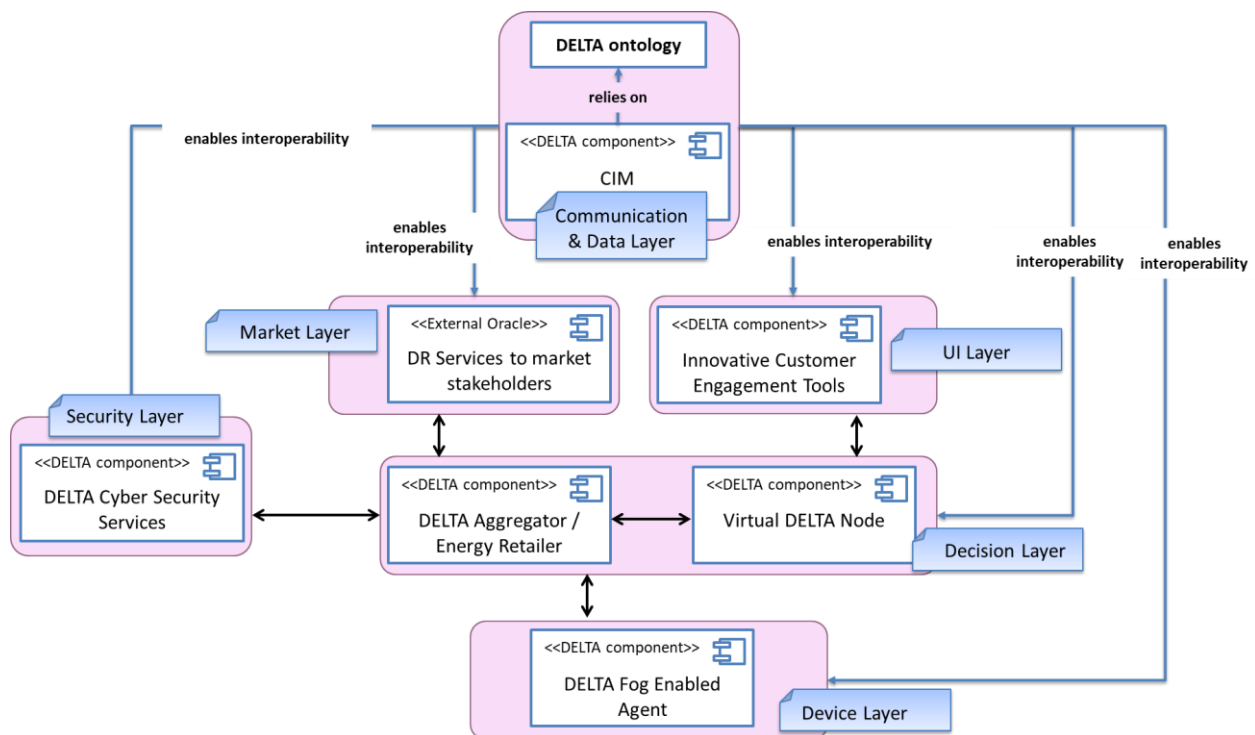


Figure 1 Relation between DELTA platform and DELTA ontology

1.2 Structure of the deliverable

The deliverable is structured as follows:

- **Section 2** provides an overview of the main standards and ontologies related to the scope of DELTA ontology network.
- **Section 3** provides the methodological guidelines followed during the ontology development.

¹ <https://www.w3.org/TR/owl-ref>

- **Section 4** describes the infrastructure deployed during the ontology development process.
- **Section 5** is dedicated to the description of DELTA ontology network.
- **Section 6** provides the specification of the DELTA data interchange interfaces.
- **Section 7** describes the DELTA Common Information Model.
- **Section 8** includes a number of examples about how to use and instantiate the DELTA ontology network and the interfaces.
- **Section 9** provides some conclusions and future lines of work.

1.3 Relation to Other Tasks and Deliverables

This deliverable depends on the DELTA architecture defined in D1.2 and T1.2 *Architectural Design, Functional & Technical Specification*.

2. Relevant standards and ontologies for DELTA

The DELTA ontology network will be based on existing standards and ontologies. Therefore, before the DELTA ontology network is developed, the set of standards and ontologies related to the Demand Response domain were analysed.

2.1 Standards

The following list enumerates the standards that we consider more relevant for Demand Response domain:

2.1.1 OpenADR

Name	OpenADR
Author and License	OpenADR Alliance
URL	https://www.openadr.org/
Description	The OpenADR standard conceptualizes demand response (a subset of DSM) through a data and communication specification. It provides a data model to facilitate information exchange between electricity service providers, aggregators and end users. This standards support the definition of server, which publish information (Virtual Top Nodes or VTNs) to the automated clients, which subscribe the information (Virtual End Nodes, or VENs). Additionally, OpenADR support several services such as registration of devices or events.
Scope	Events, measurements, Equipment

2.1.2 Smart Grid Architecture Model (SGAM)

Name	Smart Grid Architecture Model (SGAM)
Author and License	NIST SGIP/SGAC
URL	https://sgam-toolbox.org/
Description	The SGAM framework aims to present the design of smart grid use cases in an architectural viewpoint. The SGAM framework consists of five layers representing business objectives and processes, functions, information exchange and models, communication protocols and components. This model intents to represent the zones of information management where interactions between zones can occur.
Scope	Measurements, Equipment, Products

2.1.3 IEC CIM

Name	IEC CIM
Author and License	International Electrotechnical Commission (IEC)
URL	https://www.entsoe.eu/digital/common-information-model/
Description	<p>CIM represents the main resources for the management of the electric system. Expressing the knowledge associated to the electrical domain, CIM represent its base ontology.</p> <p>The CIM is a 3-layer broad domain model that aims to facilitate power management processes such as outage management, asset management and customer information management. The CIM is arguably not well suited to DSM due to its lack of modeling at the last mile of the supply chain.</p> <p>IEC TC57 family of standards, including IEC 61850, 61968, 61970 that support distribution, transmission and substation automation, collectively referred to as the Common Information Model (CIM).</p>
Scope	Events, Measurements, Equipment, Products

2.1.4 Facility Smart Grid Information Model (FSGIM)

Name	Facility Smart Grid Information Model (FSGIM)
Author and License	BSR/ASHRAE
URL	https://www.entsoe.eu/digital/common-information-model/
Description	The Facility Smart Grid Information Model (FSGIM) standard is one part of a larger ecosystem of standards that support the development and implementation of a smart electric grid. FSGIM represents an abstract information model of what the Smart Grid looks like from the perspective of a facility. It provides a common framework to guide the development of the control technologies within a facility so that they can meet the control needs of a smart grid environment.
Scope	Events, Measurements, Equipment

2.1.5 Energy@home

Name	Energy@Home
Author and License	Energy@Home Association
URL	http://www.energy-home.it/SitePages/Home.aspx
Description	The Energy@home data model specifies a representation model for home area networks, including smart appliances, power profiles, renewable energy generation, smart meters and smart user interfaces. It is based on the CIM approach and is broadly aligned with the OpenADR schema. It formalizes a method of describing devices energy consumption profiles in terms of energy phases, modes, power profiles and extended profiles
Scope	Events, Products, Equipment

2.1.6 NAESB Energy Usage Information Model

Name	NAESB Energy Usage Information Model
Author and License	North American Energy Standards Board (NAESB)
URL	https://www.naesb.org/pdf4/naesb_energy_usage_information_model.pdf
Description	The NAESB Energy Usage Information Model is a data standard that allow utilities and customers to exchange information about electricity usage. The energy usage information model, where possible, uses classes, information elements and attribute names drawn from the CIM. This standard describes usage points, which identifies the customer, the location, and the physical asset, and meter readings, which composes information about a particular measurement. The usage points may also be associated with summary information on load, usage and also power quality.
Scope	Measurements

2.1.7 OASIS Energy Market Information Exchange (eMIX)

Name	OASIS Energy Market Information Exchange (eMIX)
Author and License	OASIS (Open standards)
URL	https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=emix
Description	Energy Market Information Exchange (eMIX) is a standard for price and product definition. It is an information model designed to express common characteristics of market including price. It also describes ratchet tariffs, where exceeding demand charge thresholds may affect

	price for months.
Scope	Products

2.1.8 USEF (Universal Smart Energy Framework)

Name	USEF (Universal Smart Energy Framework)
Author and License	Alliander, ABB, DNV GL, IBM, ICT, RWE-Essent and Stedin
URL	https://ict.eu/markets/energy/usef/
Description	A framework that describes the market for flexibility, enabling commoditization and market trading of flexible energy use. USEF also specifies all stakeholder roles involved in the energy system and describes how they interact. The framework also describes service capabilities, connectivity, data exchange and control features.
Scope	Measurements

2.1.9 Energy Flexibility Interface (EFI)

Name	Energy Flexibility Interface (EFI)
Author and License	Flexible Power Alliance Network
URL	https://flexible-energy.eu/efi/
Description	Communication interface between smart devices and Demand Side Management solutions. One common language for energy flexibility. It does not model the smart devices, but only the available energy flexibility and the ways it is used by the Smart Grid technology.
Scope	Measurements

2.1.10 IEC 62746

Name	IEC 62746- 2
Author and License	International Electrotechnical Commission (IEC)
URL	https://webstore.iec.ch/publication/22279
Description	Systems interface between customer energy management system and the power management system between “Grid” and “Active Customers” will be based on current OpenADR 2 specifications. It defines an architecture and associated data model to support flexibility
Scope	Measurements

2.1.11 IEC/EN 61970

Name	IEC/EN 61970
Author and License	International Electrotechnical Commission (IEC)
URL	https://webstore.iec.ch/publication/22279
Description	Define data models to support information exchange between systems that supports business functions and that are directly involved with operation and planning of the overall interconnected electric grid. This standards includes the IEC 61970 CIM which lays down the common information model (CIM). The CIM is an abstract model that represents all the major objects in an electric utility enterprise involved in utility operations. It provides a standard representation of power system resources.
Scope	Events, measurements, Equipment, Products

2.1.12 IEC 62056 COSEM

Name	IEC 62056 COSEM
Author and License	International Electrotechnical Commission (IEC)
URL	https://webstore.iec.ch/publication/22279

Description	COSEM standard specifies smart meter functionality. It set the rules for the exchange of data for electric power measuring equipment. COSEM achieves this by using object modelling techniques to model all functions of the meter, without making any assumptions about which functions need to be supported, how those functions are implemented and how the data are transported.
Scope	Equipment, Measurements

2.1.13 CENELEC EN 50491-11 Smart Metering

Name	CENELEC EN 50491-11 Smart Metering
Author and License	CENELEC
URL	https://standards.globalspec.com/std/9931626/cenelec-en-50491-11
Description	CENELEC EN 50491-11 Smart Metering standard specifies a data model to abstract the metering world towards a simple external consumer display. The data model, as described by means of functional blocks contained in EN 50491-11, lays down the format of metering data accessible by a simple external consumer display. This data interface would be typically part of the meter communication functions and be accessed by a simple external consumer display via the H1 interface of the CEN/CLC/ETSI TR 50572 between the display and the meter communication functions (however, note that the same data model can be used also on the H2 interface).
Scope	Measurements, Equipment, Products

2.1.14 CENELEC EN 50631-1 (SPINE)

Name	CENELEC EN 50631-1 (SPINE)
Author and License	CENELEC
URL	https://standards.globalspec.com/std/9931626/cenelec-en-50491-11
Description	SPINE defines a neutral layer to connect different technologies to build a smart home/ smart grid system. SPINE defines messages and procedures on application level and is independent from the used transport protocol. Any technology that supports the bi-directional exchange of arbitrary data can be used directly, e.g., SmartHome IP (SHIP), which is also created by the EEBUS Initiative, or Thread, which is very much used by the Energy@home association. For other communications technologies, a mapping is needed. SPINE covers use cases concerning control and monitoring of smart appliances like White Goods, HVAC systems and adjacent devices like batteries, e-cars, etc. with a focus on the sectors of smart energy, smart home and building, connected devices and E-Mobility.
Scope	Measurements, Equipment, Products

2.1.15 CENELEC EN 50090 (KNX)

Name	CENELEC EN 50090 (KNX)
Author and License	CENELEC
URL	https://www2.knx.org/uy/knx/tecnologia/estandarizacion/index.php
Description	The KNX standard is designed for the control of applications in industrial, commercial and residential buildings worldwide. It supports lighting and shutter control, security systems, heating, ventilation and monitoring, among others. KNX is now including in their standards a KNX IoT ontology, which is available at https://knxiot.org . The access to the KNX ontology requires credentials that can be obtained contacting the

	KNX association.
Scope	Measurements, Equipment, Products

2.1.16 CEN EN 16836 (ZigBee SEP2)

Name	CEN EN 16836 (ZigBee SEP2)
Author and License	European Committee for Standardization (CEN)
URL	https://standards.globalspec.com/std/10058688/en-16836-2
Description	The CEN EN 16836 (ZigBee SEP2) defines devices and interfaces for Smart Energy applications in residential or light commercial environments. ZigBee can also be used in networks for sub-metering within a home or for communication of devices within such home. This standard defines various types of metering, measurements types, real-time readings, historical information, status indicators, etc. Among devices types, the standard includes ESI (Energy Services Interface), In-Premises Display, PCT (Programmable Communicating Thermostat). Regarding the functions that are supported, it includes pricing, demand response and load control, messaging and billing.
Scope	Measurements, Equipment, Products

2.2 Ontologies

The following list enumerates the ontologies that we consider more relevant for Demand Response domain:

2.2.1 SAREF4ENER

Name	SAREF4ENER
Author	European Telecommunications Standards Institute (ETSI)
URL	https://w3id.org/saref4ee
Description	SAREF4ENER is an extension of SAREF created in collaboration with the EEBus and Energy@Home industry associations to interconnect their data models. SAREF4ENER focuses on demand response systems, where customers can offer flexibility to the Smart Grid to manage their smart home devices.
Scope	Location, Stakeholders, Products, Equipment, Measurements

2.2.2 MAS2TERING

Name	MAS2TERING (FP7-ICT project)
Author	-
URL	- (not available online)
Description	MAS2TERING ontology describes the message exchange between the agents for the smart grid, the agents and their behaviours, and the constraints.
Scope	Equipment, Measurements

2.2.3 OEMA Ontology Network

Name	OEMA Ontology Network
Author	Javier Cuenca, Felix Larrinaga and Edward Curry
URL	https://innoweb.mondragon.edu/ontologies/oema/index-en.html
Description	OEMA is an ontology network to unify existing heterogeneous ontologies that represent different energy-related data, such as equipment or infrastructure.

Scope	Location, Stakeholders, Products, Equipment, Measurements
-------	---

2.2.4 CIM ontology for Smart Grids

Name	CIM ontology for Smart Grids
Author	TNO
URL	http://ontology.tno.nl/cerise/cim-profile/
Description	This ontology, developed by Cerise-SG project, is a profile of the IEC Common Information Model for Smart Grids.
Scope	Equipment, Measurements

2.2.5 OntoENERGY

Name	OntoENERGY
Author	Helmut Schmidt University and Siemens AG
URL	(not available online)
Description	OntoENERGY is an ontology for support energy-efficiency tasks. It aims to define the fundamental physical quantities and their interrelations in the energy domain. OntoENERGY also distinguished three main interpretations of energy: physical, industrial and automation.
Scope	Measurements

2.2.6 ThinkHome

Name	ThinkHome
Author	TU Vienna
URL	https://www.sciencedirect.com/science/article/pii/S0378778811005901#sec0025
Description	Ontology that allows to describe the energy produced by an energy plant and consumed by a device according to its state. It includes the energy provider, the type of energy produced, and the tariff.
Scope	Equipments, Measurements, Products

2.2.7 BOnSAI

Name	BOnSAI
Author	International Hellenic University
URL	https://www.researchgate.net/publication/254006761_BOnSAI_A_smart_building_ontology_for_ambient_intelligence
Description	Ontology that model different aspects of a service-oriented smart building system: (1) it includes concepts modelling all functionalities such as services, operations, inputs, outputs, logic, parameters and environmental conditions; (2) QoS such as resources and QoS parameters; (3) hardware such as smart devices, sensors and actuators, appliances and servers; (4) users and (5) context such as user profiles, moods, location and rooms.
Scope	Location, Measurements, Equipment

2.2.8 ProSGV3

Name	ProSGV3
Author	Université Jean Monnet
URL	http://data-satin.telecom-st-etienne.fr/ontologies/smartgrids/proSGV3/ProSG.html
Description	Ontology that models infrastructure, electrical appliances, energy

	generation and storage systems, weather report, events, energy production and consumption and information about energy producers and consumers.
Scope	

2.2.9 Mirabel ontology

Name	Mirabel
Author	TNO
URL	https://sites.google.com/site/smartappliancesproject/ontologies/mirabel-ontology
Description	The Mirabel ontology defines how actors can express their energy flexibility for a specific device. This flexibility is related to to amount, time and price in user preferences. Each device has an energy profile that describes the amount of energy consumed or produced over a time interval. A flexibility offer combines the user preferences with the corresponding device energy profile.
Scope	Measurements, Equipement, Products, Events

2.2.10 SEMANCO

Name	SEMANCO
Author	Politecnico di Torino
URL	http://www.semanco-project.eu/ontology.htm
Description	Ontology that contains the terms and attributes that describe regions, cities, neighbourhoods and buildings. It also includes energy consumption and CO2 emission indicators, as well as climate and socio-economic factors that influence energy consumption.
Scope	Equipment, Products, Measurements, Location

2.2.11 Sesame demonstrator

Name	SESAME-S Smart Building Ontology
Author	Research Centre for Telecommunication (FTW, http://www.ftw.at/), Austria
URL	https://commodatastorage.googleapis.com/ckannet-storage/2012-08-20T165445/SmartBuildingv3.owl
Description	Ontology that includes concepts related to devices, tariffs, energy usage profiles and activities. It describes an energy-aware home and the relationships between the objects and actors within the control scenario.
Scope	Location, Products, Measurements, Equipment

3. Common Information Model Requirements

The CIM provides the common data models that are going to be used from the DELTA components in the backend environment. Most of the components should interact with the CIM either for requesting information or for storing information. Thus, there should be a CIM API providing functionality for data access and management (e.g. import, export, search, access etc.). The following chapters are dedicated to the identification of possible data structures flowing between DELTA's components. As presented below, the data structure specification in CIM will be based on JSON schema. The elements described in the following sub-sections can be considered as common and shared vocabulary within the DELTA system.

3.1 Information Model

The information models of the pilot sites elements have to be imported into the CIM at the beginning of the pilot activities. The complete information for the pilot elements contains much information that is not required directly by the DELTA components. Thus, a simplified information model is stored into the CIM in JSON format. The pilot site elements, whose information model should be considered as part of the CIM, can be considered the following:

- Smart Meters;
- Assets/DERs;
- Building Information Model (BIM);
- User comfort constraints.

3.1.1 Smart Meters

Smart meters are electronic devices that records energy consumption with a standard and frequent rate unlike the conventional metering devices. These devices enable also bi-directional communication between the meter and the central system. Thus, the electricity providers are able to monitor and charge customers with different prices for consumption according to the time of day and the season. The information provided by the smart meters could be used by the majority of the DELTA components (such as Load Forecasting, Consumption/Generation Optimal Dispatch etc.). The features that are associated with the Smart meter model are:

- Smart meter ID;
- Smart meter list;
- Type;
- Communication type;
- Specific space/location;
- Measurement recording time interval.

3.1.2 Assets/DERs

The information related to assets/DERs concerns the energy resources that are available on each pilot site. The distributed energy resources (DERs) refer to distributed/on-site (district/decentralized) generation and includes the electrical generation and storage resources. Each energy resource can be considered as an asset. Each prosumer of the aggregator's portfolio has a combination of assets. Multiple DERs can be combined and managed within a smart grid framework. The features that are related to the asset/DER model are presented below:

- Asset/DER ID;
- Asset/DER list;
- List of metering/sensoring devices (e.g. IoT devices etc.) belonging to each DER;
- Energy profile type.

3.1.3 Building Information Model (BIM)

The building information model contains much information that is not directly required by the platform's components. Thus, the following simplified structure is presented:

- Space reference ID;
- Devices related to energy consumption;
- Devices related to energy production;
- CO₂ emissions metric.

3.1.4 User comfort constraints

The concept of customer satisfaction should be considered during the design framework of DR programs strategies. Thus, during the selection of the appropriate DR program for each customer, the habits related to energy consumption should be taken into account. Another key factor is the remuneration of the customer. As main features of user comfort constraints can be considered the following:

- Comfort limits concerning temperature;
- Comfort limits concerning luminance.

3.1.5 Gamification

All the information related to the gamification activities within DELTA will be collected and stored in the CIM during the pilot site testing activities. The gamification points will be produced by a game mechanics for the Demand Response framework of the DELTA energy platform. The type of award to customers will be based on their participation in DR services through fun DR related activities, such as DR games based on actual DR scenarios. The main features of the Gamification system are:

- DR signal ID;
- DR signal type;
- DR signal duration;
- Gamification statistics;
- Gamification actions;
- Gamification highlights;
- Awards.

4. Ontology development methodology

This section presents the ontology development methodology, including ontology requirements specification, implementation, publication and maintenance, used for the development of DELTA ontologies. The development methodology described is based on NeOn methodology (Suárez-Figueroa, 2012) and adapts previous ontology development processes from the European VICINITY project (García-Castro et al., 2017) to the particularities of DELTA. Figure 2 shows an overview of the processes that have to be performed and of the products resultant of them.

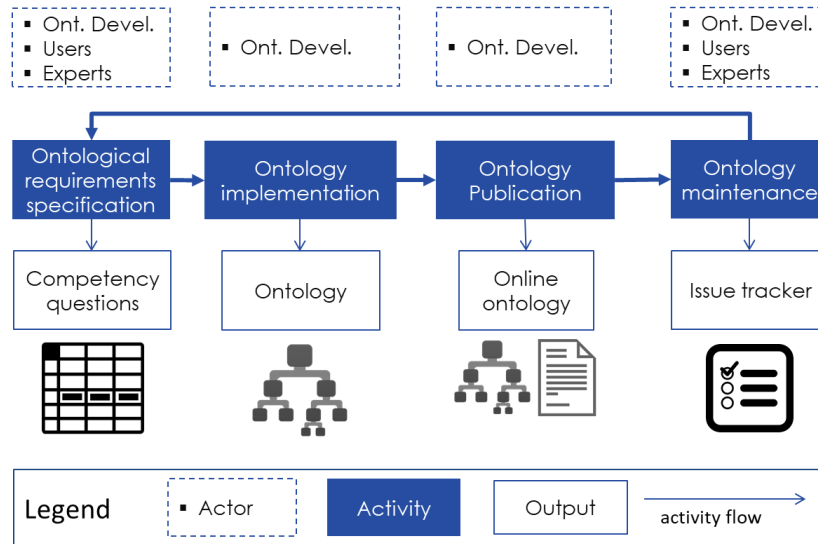


Figure 2 DELTA ontology development process (García-Castro et al., 2017)

4.1 Ontological requirements specification

The aim of the requirements specification activity is to identify and define the requirements the ontology to be created need to fulfil. During this activity, involvement and commitment by experts in the specific domain at hand is required to generate the appropriate industry perspective and knowledge. Figure 3 shows the steps needed to carry out this requirements specification activity.

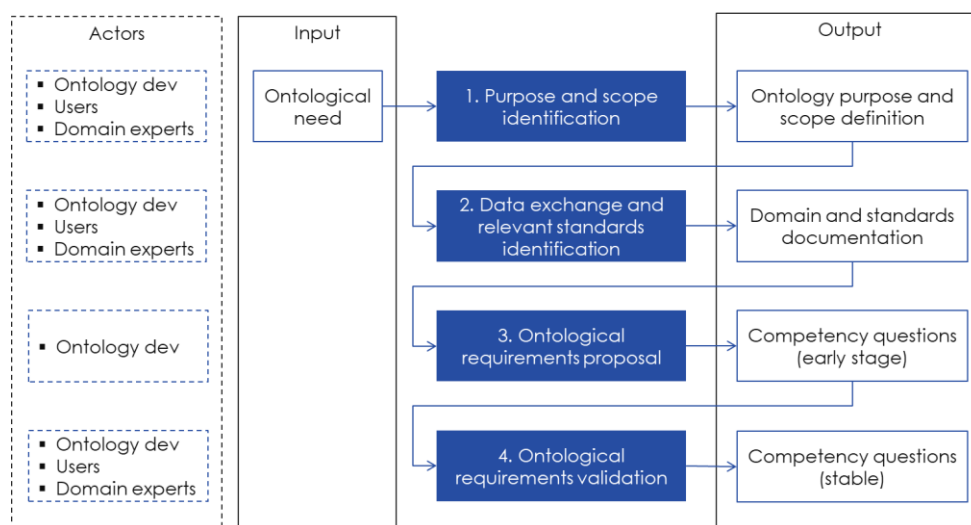


Figure 3 Workflow for ontology requirements specification

4.1.1 Purpose and scope identification

The goal of this step is to define the purpose and scope of the given ontology or module.

The ontology development team works in collaboration with users and domains experts to define the purpose and scope of each ontology or module to be developed.

4.1.2 Data exchange identification

During this activity the ontology development team needs to gather the necessary documentation about the domain to be modelled. This documentation might correspond to:

- Standards
- Datasets
- API specifications

This documentation needs to be collected by both the ontology development team and the domain experts.

4.1.3 Ontological requirements proposal

Taking as input the documentation and data provided by domain experts and users, the ontology development team generates a first proposal of ontological requirements written in the form of Competency Questions (Gruninger, 1995) or of sentences.

The format used for this requirements proposal follows a tabular approach and includes the following fields:

- Requirement identifier, which needs to be unique for each requirement
- Partner related to the definition of the requirement
- Component of the DELTA architecture related to the requirement
- Sprint in which the requirement is planned to be implemented
- Competency question or sentence
- Status of the requirement, which can be: (1) Proposed, (2) Accepted, (3) Rejected, (4) Pending or (5) Deprecated.
- In case the requirement is deprecated, the identifier of the updated requirement.
- Comments about the requirement
- Provenance of the requirement
- Priority of the requirement, which can be: (1) High, (2) Medium or (3) Low.

The ontological requirements are completed iteratively, as the followed ontology development process is incremental. In each iteration or sprint several components of the DELTA architecture are analysed.

4.1.4 Ontological requirements completion and validation

During this activity domain experts and users in collaboration with the ontology development team validate whether the ontology requirements defined in the previous step are correct and complete.

4.1.5 Ontological requirements prioritization

This prioritization will be performed if there is the need for prioritizing functional requirements. This prioritization allows the development team to plan and schedule the development of the ontology in sprints. This prioritization would be present in the backlog driving therefore the ontology development process.

To carry out this prioritization the ontology development team works with the domain experts to identify which requirements need to be fulfilled first.

4.2 Ontology implementation

During the implementation activity, the ontology is built using a formal language based on the requirements identified in the previous activity.

After defining the set of requirements, though modification and addition of requirements is allowed during the development, the ontology implementation phase is carried out through a number of sprints. The ontology developers schedule the ontology development according to the requirements that were identified. The ontology development team builds the ontology iteratively, implementing only a certain number of requirements in each iteration. The output of each iteration is a new version of the ontology.

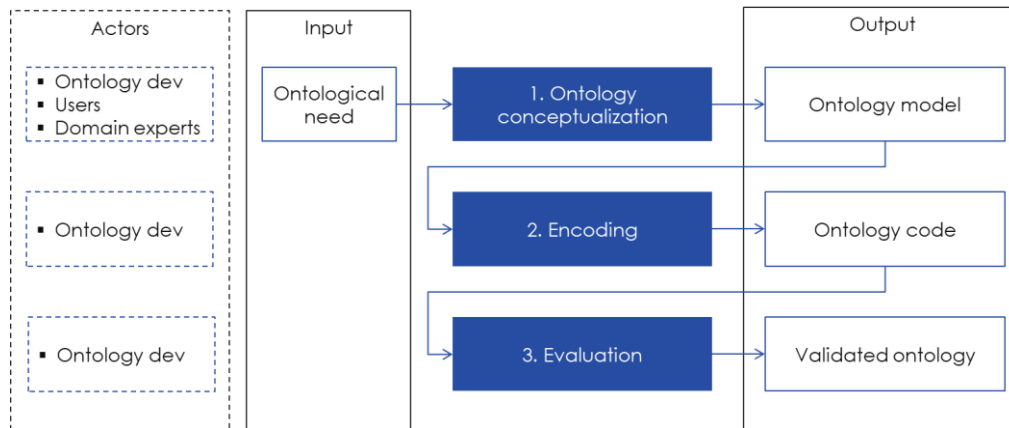


Figure 4 Workflow for ontology implementation (García-Castro et al., 2017)

4.2.1 Ontology conceptualization

The aim of this activity is to build an ontology model from the ontological requirements identified in the requirements specification process. During the ontology conceptualization, the domain knowledge obtained from the ORSD document is organized and structured into a model by the ontology developers.

4.2.2 Ontology implementation

During this activity, the ontology development team generates computable models in the OWL language from the ontology conceptualization. The ontology code resultant from this activity includes metadata, such as creator, title, publisher, license and version of the ontology.

During the development of the DELTA ontology a following version convention, based on software versioning² has been adopted. Following this convention, each release will follow the pattern v.major.minor.fix, where each field follows the rules:

- **major:** The field is updated when the ontology covers the complete domain it intends to model. That is, it is a complete product and covers the final goal of the development.
- **minor:** The field is updated when:
 - All the requirements of a subdomain are covered.
 - Documentation is added to the ontology.
- **fix:** The field is updated when:
 - Typos or bugs are corrected in the ontology.
 - Classes, relationships, axioms, individuals or annotations are added, deleted or modified but the domain is not covered.

In each iteration the minor and fix fields might be changed from zero to several times

4.2.3 Ontology evaluation

Before publishing a new version of the ontology, the ontology development team needs to evaluate the ontology to guarantee that:

- The ontology does not have syntactic, modelling or semantic errors.
- The ontology fulfils the requirements scheduled for the ontology, in order to assure that the ontology is completed regarding the domain experts needs.

² <https://semver.org>

4.3 Ontology publication

During this activity the ontology development team provides an online ontology which is accessible both as a human-readable document and a machine-readable file from its URI. Figure 5 shows the steps that needs to be carried out to publish the ontology.

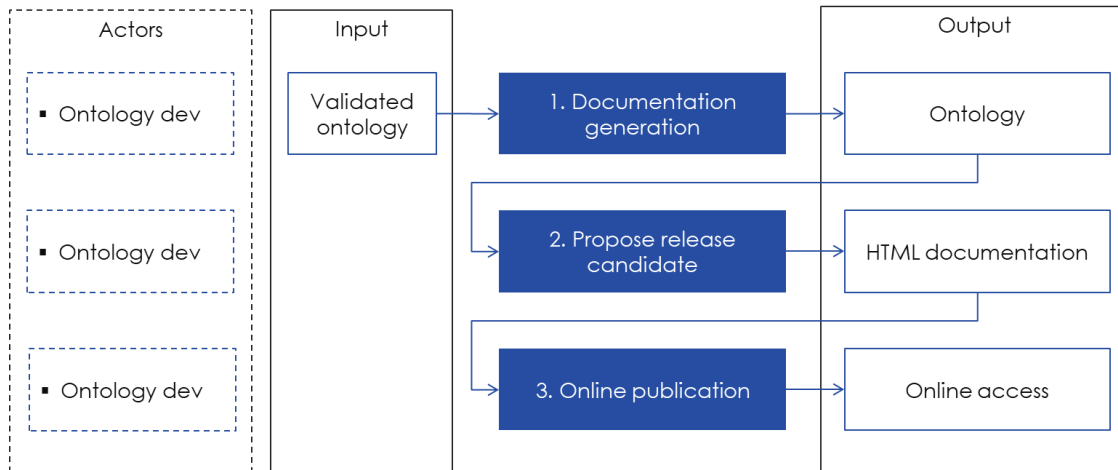


Figure 5 Workflow for ontology publication (García-Castro et al., 2017)

4.3.1 Documentation generation

Taking as input the ontology generated in the previous activities, the ontology development team in collaboration with the domain experts generates the ontology documentation. This documentation includes:

- An HTML description of the ontology which describes the classes, properties and data properties of the ontology, the license URI and title being used. The domain experts have to collaborate with the ontology development team to describe the classes and the properties. This description also includes metadata, such as creator, publisher, date of creation, last modification or version number. It also includes links to different formats of serialization of the ontology, such as TTL, JSON-LD or RDF/XML.
- Diagrams which store the graphical representation of the ontology, including taxonomy and class diagrams.

4.3.2 Propose release candidate

Once the ontology developers have implemented and validated the ontology, they propose a release version of the ontology to be published on the Web. This version tagging needs to follow the convention version convention described Section 4.2.2.

4.3.3 Online publication

During this activity the ontology, which should be already validated and documented, is published on the Web. This ontology is accessible through its URI as a machine-readable and human-readable file by using content negotiation.

4.4 Ontology maintenance

The goal of this activity is to update and add new requirements to the ontology, as well as to identify and correct errors in the ontology.

During the ontology development process, the domain experts and ontology developers can propose new requirements to the ontology. These requirements need to be approved by the ontology development team.

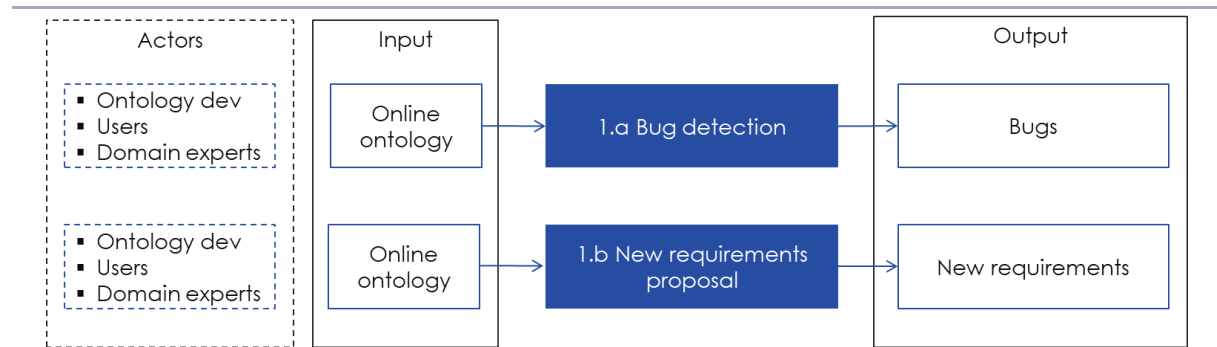


Figure 6 Workflow for ontology maintenance

4.4.1 Bug detection

Once the ontology developers have published the ontology, any user, developer or domain expert can detect and inform about bugs. This notification should be done by means of an issue tracker, so all the information related to the bug, as well as the actor that identifies it, is stored.

4.4.2 New requirements proposal

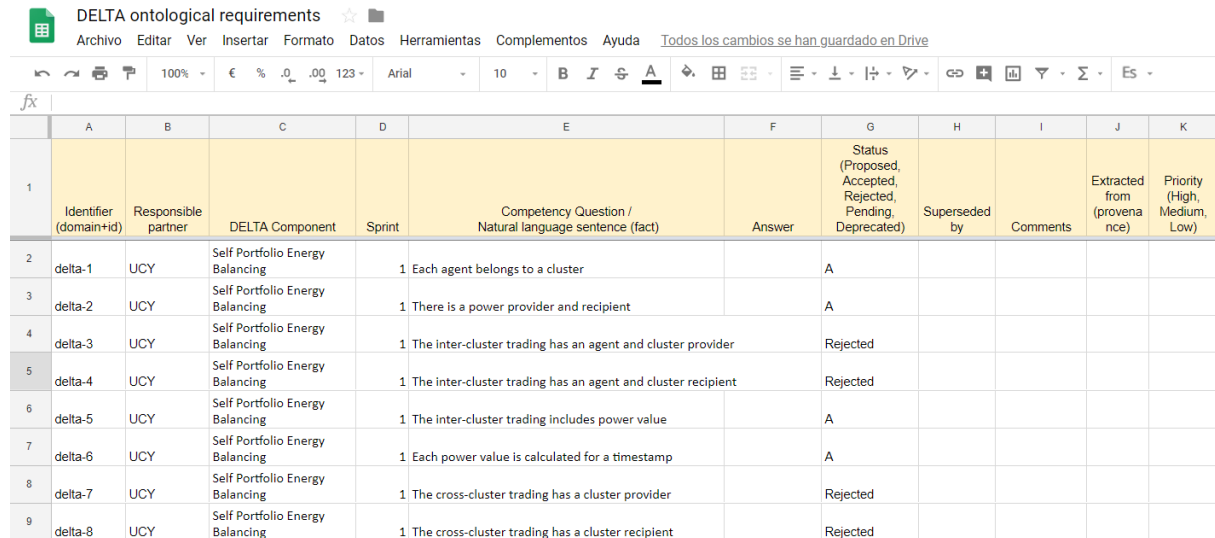
During this activity new requirements can be proposed to the ontology. The ontology developers, domain experts or users can propose modifications in order to improve the published ontology version. This proposal should be done by means of an issue tracker, so all the information related to it is stored.

5. Ontology development infrastructure

This section describes the ontology development infrastructure used to support the activities in the ontology development process.

5.1 Infrastructure to support the requirements specification activity

To support the requirements specification activity, the ontology development team use Google Spreadsheets³ to store the requirements. Figure 7 shows an excerpt of the requirements for DELTA ontology.



	A	B	C	D	E	F	G	H	I	J	K
	Identifier (domain+id)	Responsible partner	DELTA Component	Sprint	Competency Question / Natural language sentence (fact)	Answer	Status (Proposed, Accepted, Rejected, Pending, Deprecated)	Superseded by	Comments	Extracted from (provena nce)	Priority (High, Medium, Low)
1											
2	delta-1	UCY	Self Portfolio Energy Balancing	1	Each agent belongs to a cluster		A				
3	delta-2	UCY	Self Portfolio Energy Balancing	1	There is a power provider and recipient		A				
4	delta-3	UCY	Self Portfolio Energy Balancing	1	The inter-cluster trading has an agent and cluster provider		Rejected				
5	delta-4	UCY	Self Portfolio Energy Balancing	1	The inter-cluster trading has an agent and cluster recipient		Rejected				
6	delta-5	UCY	Self Portfolio Energy Balancing	1	The inter-cluster trading includes power value		A				
7	delta-6	UCY	Self Portfolio Energy Balancing	1	Each power value is calculated for a timestamp		A				
8	delta-7	UCY	Self Portfolio Energy Balancing	1	The cross-cluster trading has a cluster provider		Rejected				
9	delta-8	UCY	Self Portfolio Energy Balancing	1	The cross-cluster trading has a cluster recipient		Rejected				

Figure 7 DELTA ontology requirements in a Google Spreadsheet

These Google Spreadsheets are converted to an HTML file⁴ with the most relevant information for the users to facilitate visualization. Figure 8 shows an excerpt of the HTML documentation of the DELTA requirements.

³ goo.gl/y2x6Nz

⁴ <http://delta.iot.linkeddata.es/requirements/report.html>



Here you can find the list of the requirements identified for DELTA ontology and their main features.

Identifier	Sprint	Competency Question	Answer	Status	Superseeded by	Extracted from	Priority
delta-9	1	The cross-cluster trading includes power value		Pending			
delta-8	1	The cross-cluster trading has a cluster recipient		Pending			
delta-7	1	The cross-cluster trading has a cluster provider		Pending			
delta-6	1	Each power value is calculated for a					

Figure 8 Excerpt of the HTML documentation of the requirements

5.2 Infrastructure to support the ontology implementation activity

To support the implementation activity, the ontology development team use several tools to edit, store and evaluate the ontology.

- For ontology edition the ontology development team uses Protégé.⁵ This tool allows the creation, visualization and manipulation of ontologies.
- For ontology storage, the ontology development team uses GitHub. The GitHub repository created to store DELTA ontology⁶ includes:
 - A folder with the implementation of the ontology.
 - A folder with the ontology modelling diagrams.
 - A folder with the documentation of the ontology.
 - A folder with the requirements and tests of the ontology.

The ontology developers should use Git propose workflow to develop the ontology⁷, which can be summarized in the following steps:

1. Create a new branch for the master branch
2. Add changes to the ontology and commit them
3. Open a pull request to start a discussion over the proposed changes
4. If the pull request is accepted, then merge the new branch into the master branch

The development team uses the tool OnToology to generate the documentation and to evaluate the ontology. OnToology, which integrates the tools Widoco (Garijo, 2017), OOPS! (Poveda-Villalón, 2012) and AR2DTool,⁸ generates automatically a folder in the GitHub repository with includes the resources: diagrams, documentation and evaluation report.

⁵ <https://protege.stanford.edu>

⁶ <https://github.com/oeg-upm/delta-ontology>

⁷ <https://git-scm.com/book/en/v2/Git-Branching-Branching-Workflows>

⁸ <https://github.com/idafensp/ar2dtool>

Additionally, the development uses the tool Themis⁹ in order to validate that the requirements defined for the DELTA ontology are fulfilled. After each pull request, GitHub will inform the developer by means of a message in the pull request if there are some requirements that are not covered in the ontology. If this occurs, then a new issue is created with more details about these requirements.

5.3 Infrastructure to support the ontology publication activity

To support the publication activity, the ontology development team creates an online ontology portal in order to make accessible the ontology and all the associated information (repository, requirements, tests, releases, etc.) to the users. This ontology portal has different sections:

1. Ontologies
2. Ontology testing
3. How we work

The following subsections will describe each of these sections.

5.3.1 Ontologies

The Ontologies section¹⁰ is the main section of the portal, which shows the main information about the ontology created. Figure 9 shows an overview of the information exposed in this section of the portal. The section follows a tabular approach which includes:

- Link to the ontology documentation published on the Web
- Ontology description;
- Link to each Github repository
- Links to each GitHub issue tracker
- HTML description of the requirements identified by the domain experts
- Link to each ontology releases.

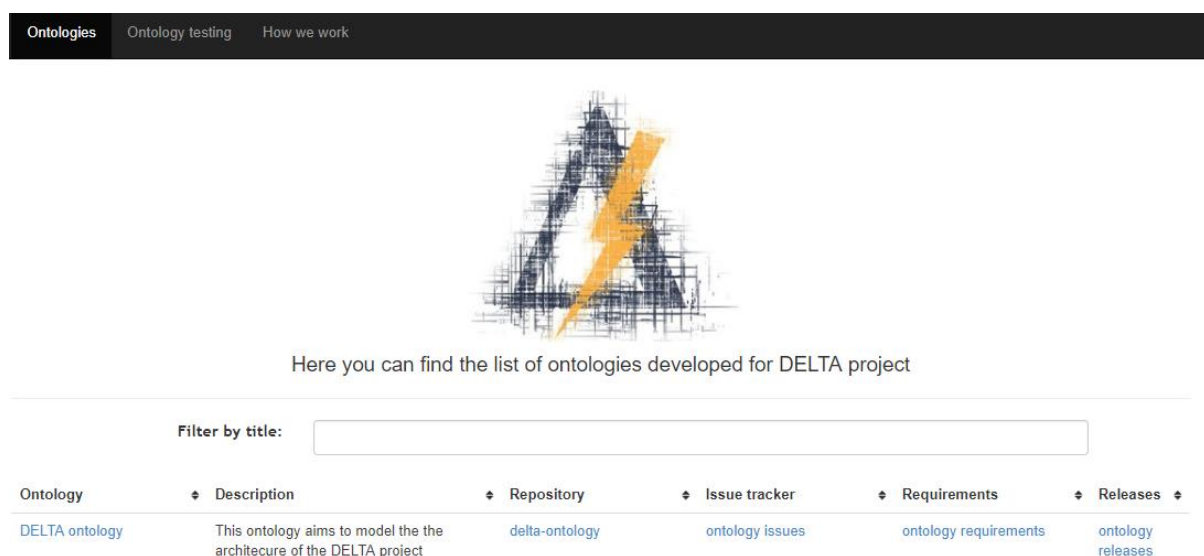


Figure 9 DELTA ontology portal

5.3.2 Ontology testing

The portal also has a section of ontology testing¹¹ that allows to validate the ontology. This testing service, which integrates Themis, allows the users or ontology developers to execute tests on the

⁹ <http://themis.linkeddata.es>

¹⁰ <http://delta.iot.linkeddata.es/>

¹¹ <http://delta.iot.linkeddata.es/validationontology.html>

DELTA ontology to verify if it models all the requirements that the domain experts and users need. Figure 10 shows an overview of this testing service together with an example of test executed and its result.

[Ontologies](#) [Ontology testing](#) [How we work](#)

Validate the ontology network

Below you can add tests to validate the DELTA ontology

Test:

[See supported tests](#)

[Check test](#)

Test	Ontology	Result
Prosumer subclassOf Customer	delta	Passed

Figure 10 Ontology testing section

The tool informs the user if the ontology passed the test, if the terms in the test are not defined in the ontology or if the ontology does not pass the test.

5.3.3 How we work

Regarding the “how we work” section,¹² it provides a brief overview of the proposed process for developing ontologies and some guidelines, which should be anyone who wants to contribute to this ontology. This section includes information about:

- How the repository should be structured
- The tools recommended to be used during the ontology development process
- Ontology versioning
- Issues management

5.4 Infrastructure to support the ontology maintenance activity

To provide support for the maintenance activity, the ontology developers use the GitHub issue tracker¹³ which manages and maintain the list of issues identified by the domain experts and ontology developers. The GitHub issue tracker provides status of the issue, assignee, and description and allows to add comments to the issue to discuss about it.

All the proposal of changes and improvements over the ontology need to be agreed by all the ontology development team. If domain experts, users or ontology developers want to add, delete or modify concepts in the ontology they must create a new issue in the GitHub issue tracker, which will be used to discuss about the approval or rejection of the proposal. Figure 11 shows the GitHub issue tracker for the DELTA ontology.

¹² <http://delta.iot.linkeddata.es/guide.html>

¹³ <https://github.com/oeg-upm/delta-ontology/issues>

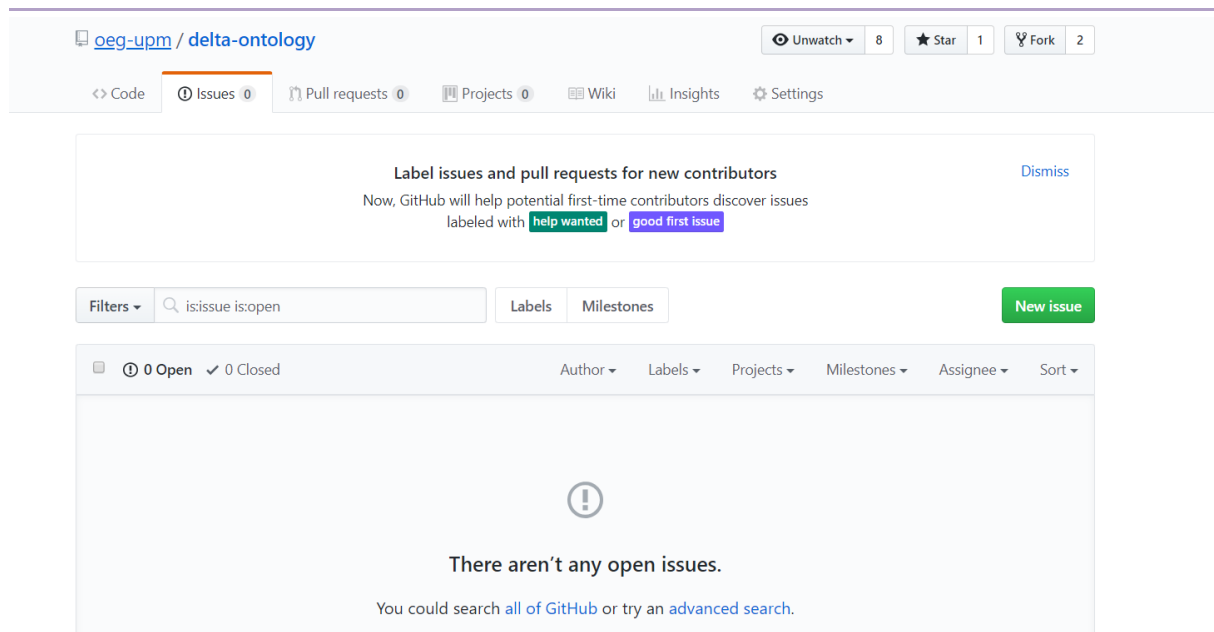


Figure 11 GitHub issue tracker

The ontology developers should label each issue according to its topic in order to organize the different types of issues. GitHub provides by default a few labels: bug, duplicate, enhancement, invalid, question, and won't fix. Those issues that represent new requirements for the ontology should be labelled as a “requirement” issue. In addition, ontology developers can create new labels if they think they can improve issue management. The ontology developer team is also responsible for closing the issues created that have already been addressed.

6. DELTA ontology overview

This section describes the ontology developed for DELTA platform. As mentioned in previous sections, the DELTA ontology portal is available online at <http://delta.iot.linkeddata.es> and provides the following information:

- A link to the documentation
- A link to the GitHub repository in which the code is stored
- A link to the issue tracker
- A link to the online information of requirements
- A link to the list of its releases

Figure 12 shows an overview of the concepts and relations modelled in the DELTA ontology. The figure was created using the following conventions:

- Green rectangles are used to denote classes created in the DELTA ontology being described while white rectangles represent reused classes.
- Plain arrows with white triangles represent the *rdfs:subClassOf* relation between two classes. The origin of the arrow is the class to be declared as subclass of the class at the destination of the arrow.
- Plain arrows between two classes indicate that an object property has been declared. The origin of the arrow represents the domain and the destination indicates the range. The identifier of the object property is indicated within the arrow.
- Datatype properties are represented by rectangles attached to the classes.

The complete and up to date documentation of each ontology module is provided online and is accessible through each of the ontology URI. In the rest of this document only the main concepts and modelling decisions are detailed.

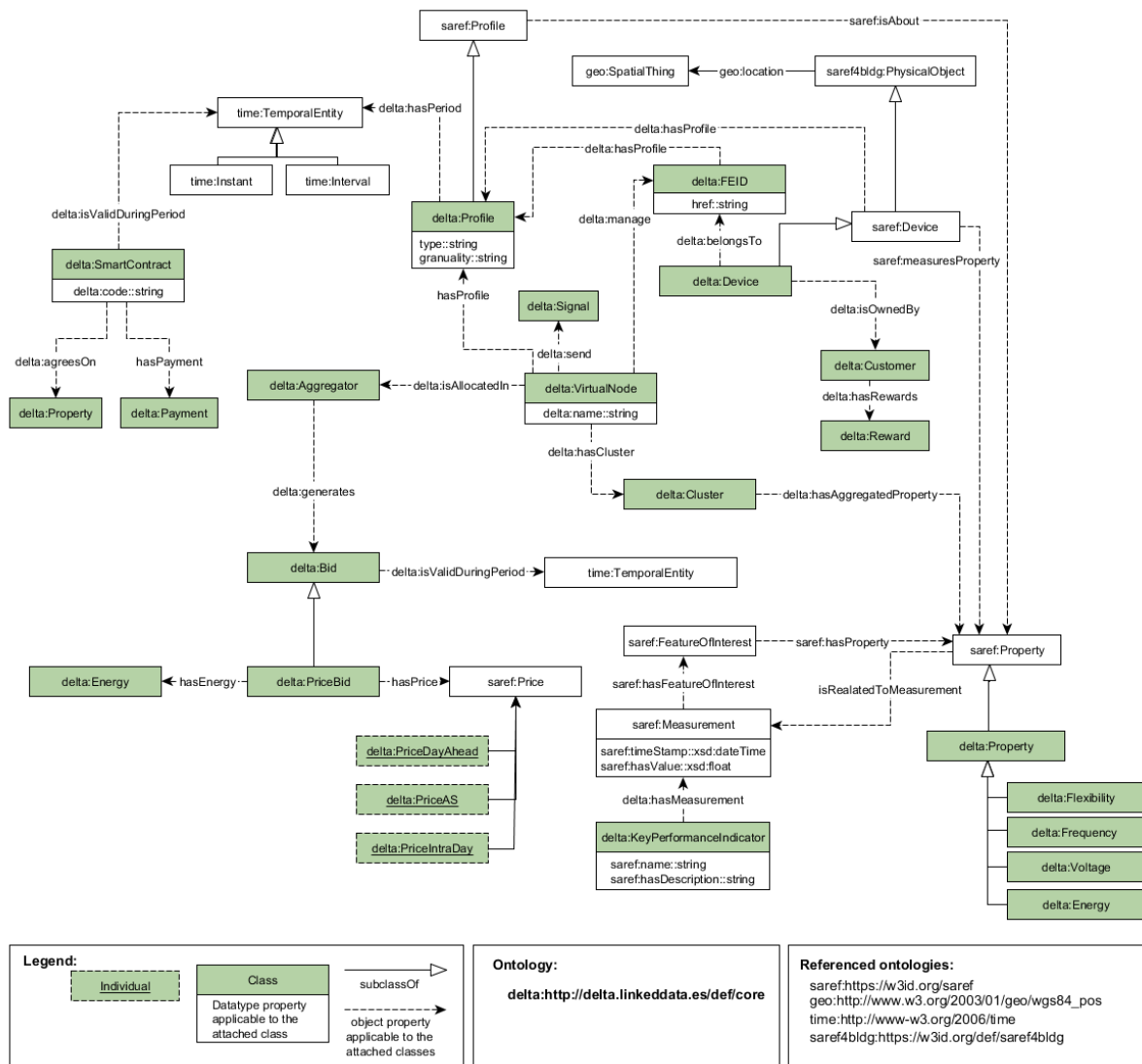


Figure 12 Ontology overview

There are two types of signals exchanged in the DELTA platform: (1) Status signals, which send the status of an entity and (2) DR signals, which represent actions, e.g., request energy price, that need to be carried out by the entities. These latter signals are aligned with OpenADR standard. Figure 13 shows the different type of signals.

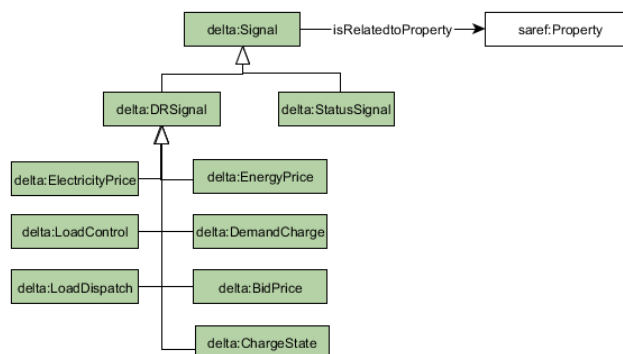


Figure 13 DELTA Signals

The DELTA ontology is based on the following non-functional requirements:

1. **Reuse:** existing ontologies and standards will be reused when possible in order to increase interoperability with external systems.
2. **Good practices:** the ontology is developed following methodologies and good practices commonly used in ontological engineering.

Additionally, a set of functional requirements were defined in order to develop the ontology. These requirements identify the knowledge that has to be modelled in the ontology. The domain experts and the ontology developers worked together in order to identify these functional requirements. In total, 65 requirements were identified, and from those 36 requirements were accepted, 25 requirements were rejected and 7 requirements are pending.

DELTA ontology has been developed to model concepts related to energy market, smart grids and demand-response platforms within the DELTA demand response platform. The main concepts in this ontology are:

- **Virtual Node.** The Virtual DELTA Node is meant to handle several DELTA Fog Enabled Agents, which includes devices and are in charge of controlling and monitoring physical infrastructures.
- **Aggregator.** The aggregator is the component that links and combine the data coming from most of the components that belong to DELTA.
- **Flexibility.** Flexibility is a property that indicates the ability to increase or reduce the production of power plants or the consumption of demand processes.
- **Customer.** A DELTA customer represents a device that consumes and produces energy.
- **Property.** This concept all the properties that are associated to a Device, such as the consumed energy, power or price.

According to the model, a `delta:VirtualNode` belongs to a particular `delta:Aggregator` and manages a set of `saref:Device`. This `saref:Device` belongs to a `delta:Prosumer` or `delta:Consumer`, and they will measure several `saref:Property` such as `delta:Energy` or `delta:Power`. The `delta:Aggregator` will use the `saref:Property` of the `delta:Device` managed by the `delta:VirtualNode` associated to it in order to generate `delta:Bid`. This `delta:Bid` will be used by the `delta:EnergyMarket` to estimate the expected flexibility and energy price.

Finally, the `delta:Customer` will be rewarded based on its behaviour. This reward is represented in the model as `delta:Reward`.

It should be mentioned that the presented ontology is under development and new concepts might be included or extended.

7. Specification of DELTA data interfaces

This section is devoted to the specification of the DELTA data interfaces. Here the interfaces for input and output data associated to each DELTA component will be identified, as well as the concepts in the ontology associated to each of these interfaces. The exchange of data between components is carried out using JSON-LD interfaces, which is lightweight Linked Data format based on JSON. These JSON-LD interfaces will be based on the JSON schemas, which are also exposed in this section in order to explain the structure of the data.

It is worth mentioning that some of the information included in this section, such as the descriptions of the components, were also included in the deliverable D1.2. In addition, there are interfaces that, even though they have different names, they exchange the same type of data, e.g. Node Profiling and Forecasted Profiling. Therefore, the ontology concepts related to them and the JSON-LD structure are similar in this scenario. All figures in this section follow UML conventions¹⁴ for the representation of components, subcomponents and interfaces.

In the following subsections, the following issues are described for each component:

1. Interfaces
2. Ontology concepts related to each interface

For the sake of clarity, the JSON schemas associated to each interface, and the JSON-LD interface examples are described in Annex A.

7.1 DELTA Virtual Node

This subsection describes the interfaces associated to the DELTA Virtual Node subcomponents.

7.1.1 Consumer/Prosumer Flexibility Data Monitoring and Profiling

The goal of this component is to provide a real-time overview of the assets assigned to a specific Virtual DELTA Node. To achieve this goal, the component has three interfaces: (1) Flexibility forecast, (2) Historical consumption (3) Historical generation, (4) Node Profiling and (5) Voltage & Frequency. Figure 11 shows the interfaces that are related to this component.

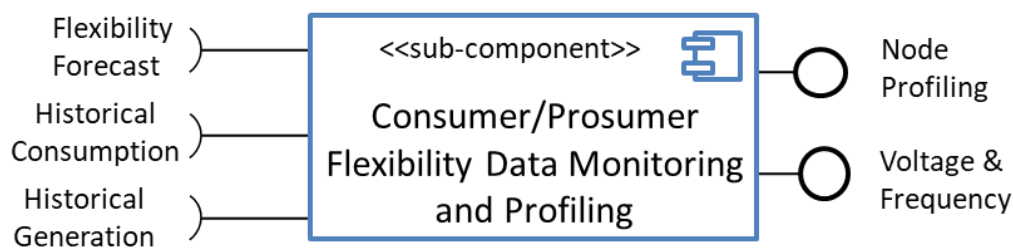


Figure 14 Interfaces for Consumer/Prosumer Flexibility Data Monitoring and Profiling

7.1.1.1 Flexibility forecast

This interface will provide the component with the profiling related to the flexibility forecast.

Regarding the ontology model, the information exchanged by this component in this interface corresponds to the following concepts that are summarized in Figure 15.

¹⁴ <https://www.ibm.com/developerworks/rational/library/dec04/bell/index.html>

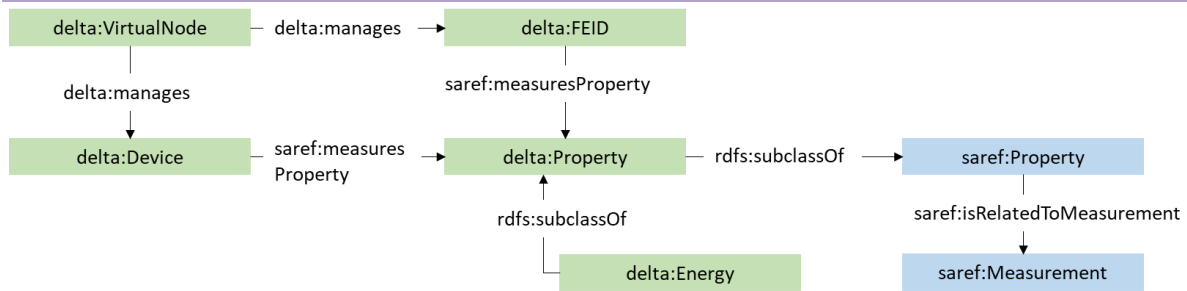


Figure 15 Ontology concepts related to Flexibility forecast

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-9.

7.1.1.2 Historical consumption

This interface will provide the component with the historical energy consumption, either related to each device or aggregated in FEIDS.

Regarding the ontology model, the information exchanged by this component in this interface corresponds to the following concepts that are summarized in Figure 16.

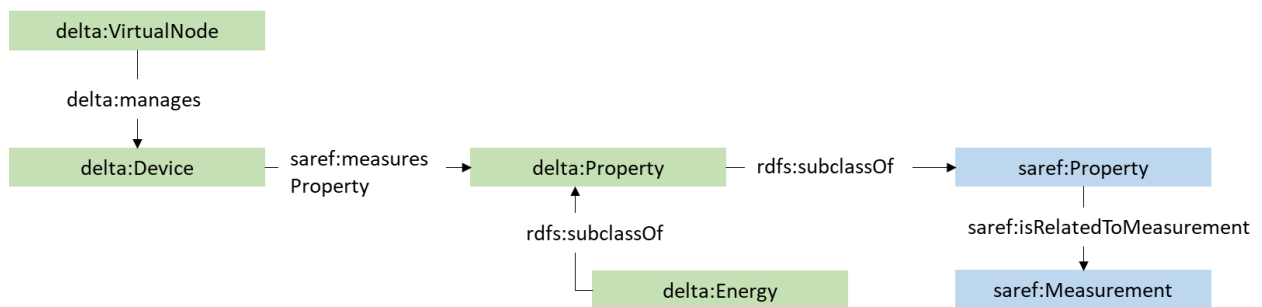


Figure 16 Ontology concepts related to Historical Consumption

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-11.

7.1.1.3 Historical generation

This interface will provide the component with the historical energy generation, either related to each device or aggregated in FEIDS.

Regarding the ontology model, the information exchanged by this component in this interface corresponds to the following concepts that are summarized in Figure 17.

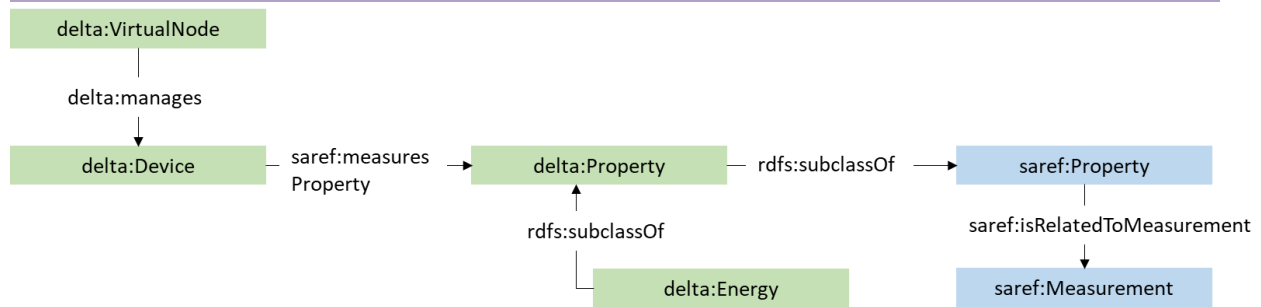


Figure 17 Ontology concepts related to Historical Generation

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-12.

7.1.1.4 Node Profiling

This interface will provide other components with the DELTA Node Profiling. This profiling will be used to assess the DVN and the devices. An example of possible profile could be related with the reliability of the device. Figure 18 summarizes the concepts in the ontology model that are related to this interface.

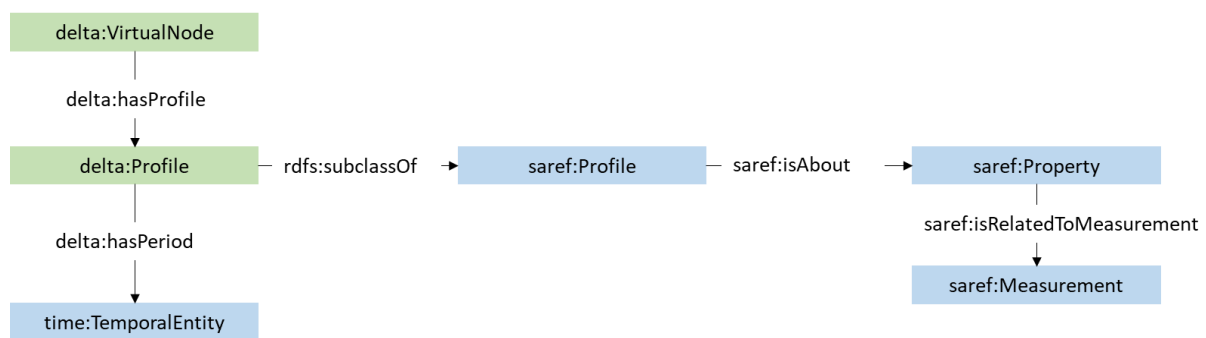


Figure 18 Ontology concepts related to Node Profiling

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-15.

7.1.1.5 Voltage & Frequency

This interface represents the constraints of the grid, i.e., the voltage and frequency that are calculated by a device. Figure 19 summarizes the ontology concepts related to this interface

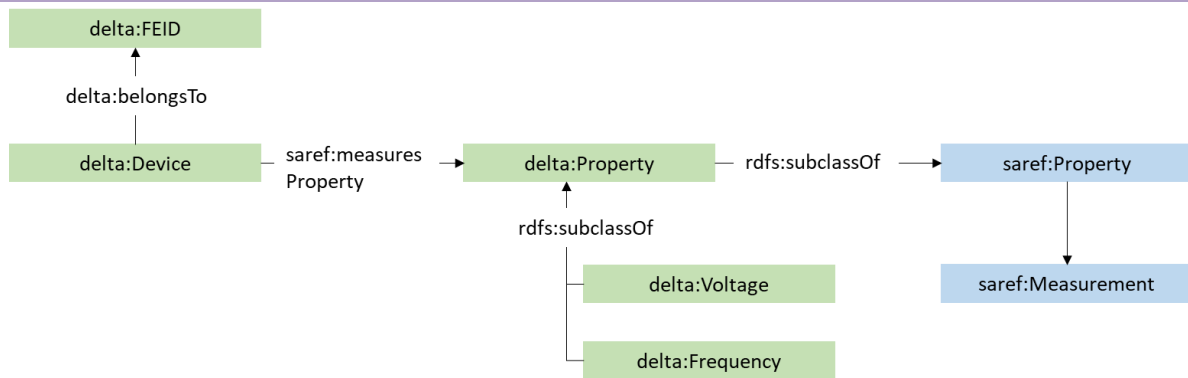


Figure 19 Ontology concepts related to the Voltage & Frequency Constraints

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-21.

7.1.2 Generation/Consumption Optimal Dispatch

The goal of this component is to establish energy decisions that the DELTA Fog Enabled Agents must fulfil by evaluating the reliability for each FEID and re-distributing the DR signals from aggregators to FEIDs. To achieve this goal, the component has five interfaces: (1) FEIDs clusters, (2) DR Signals, (3) Node Profiling and (4) Forecasted profiling and (5) Transactions. Figure 20 shows the interfaces that are related to this component. The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex A.

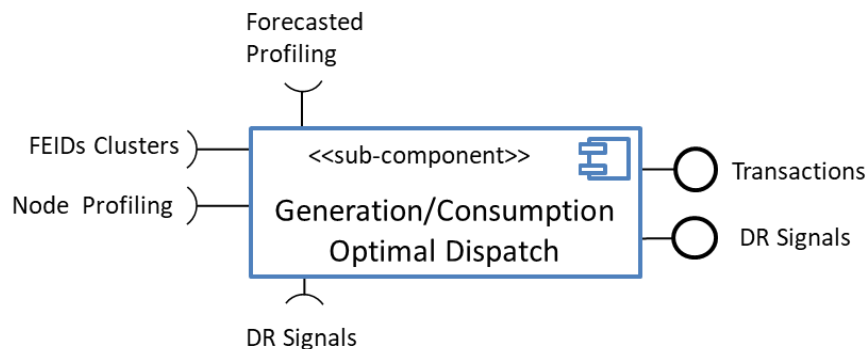


Figure 20 Interfaces for Generation/Consumption Optimal Dispatch

7.1.2.1 FEIDs Clusters

This interface provides the components with the devices associated a FEID. Figure 21 summarizes the ontology concepts that are related to this interface.

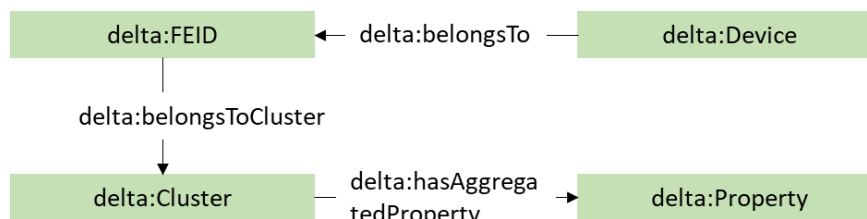


Figure 21 Ontology concepts related to the FEIDs Clusters

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-8.

7.1.2.2 DR signals

This interface sends a set of DR Signals in order to request and offer energy or to inform about the consensus related to a smart contract. Figure 22 shows the ontology concepts related to this DR signals. It is worth noting that these signals are aligned with OpenADR signals.

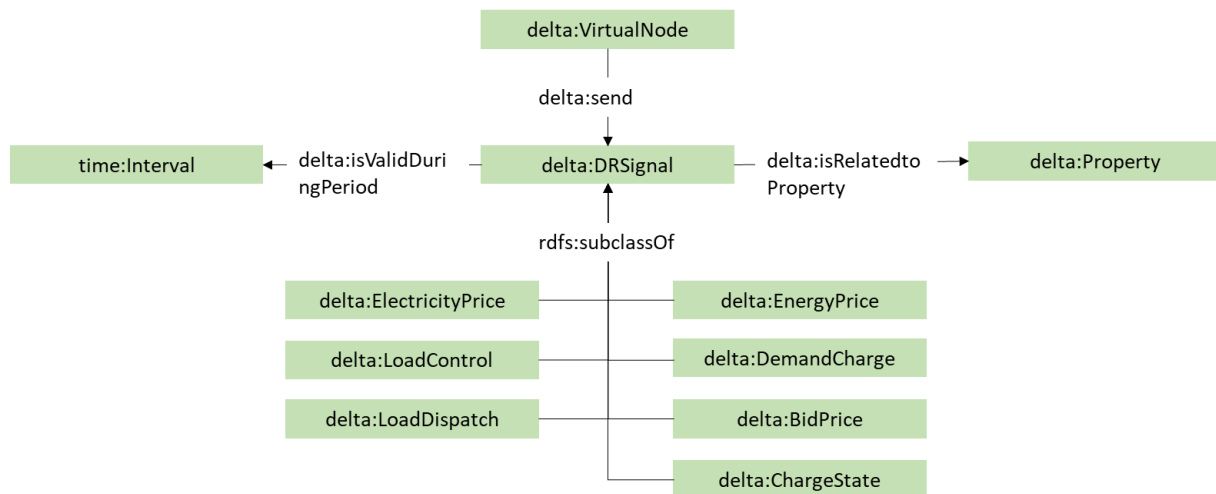


Figure 22 Ontology concepts related to DR Signals

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-5.

7.1.2.3 Node Profiling

This interface will provide other components with the Node Profiling. This profile will be used to assess the virtual node and the devices. An example of possible profile could be related with the reliability of the device. Figure 23 summarizes the concepts in the ontology model that are related to this interface.

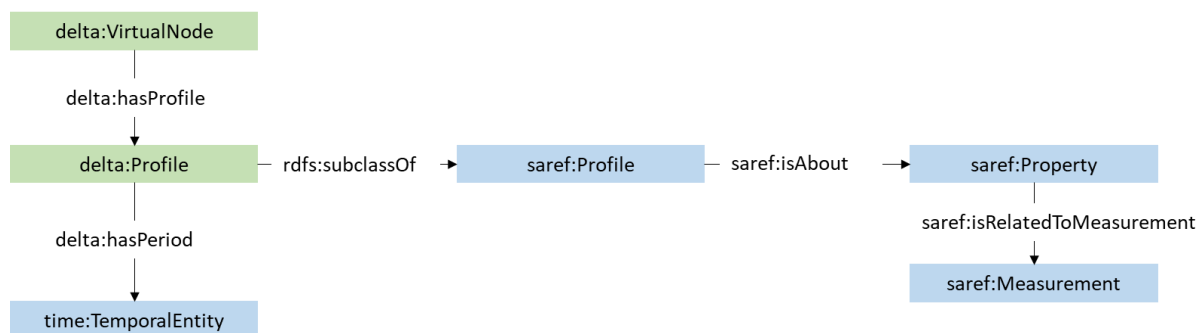


Figure 23 Ontology concepts related to DELTA Node Profiling Interface

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-15.

7.1.2.4 Forecasted profiling

This interface exchanges the forecasted profile of a Virtual Node. This profile is also related to a property measured by a device associated to the Virtual Node. Figure 24 summarizes the ontological concepts related to this interface.

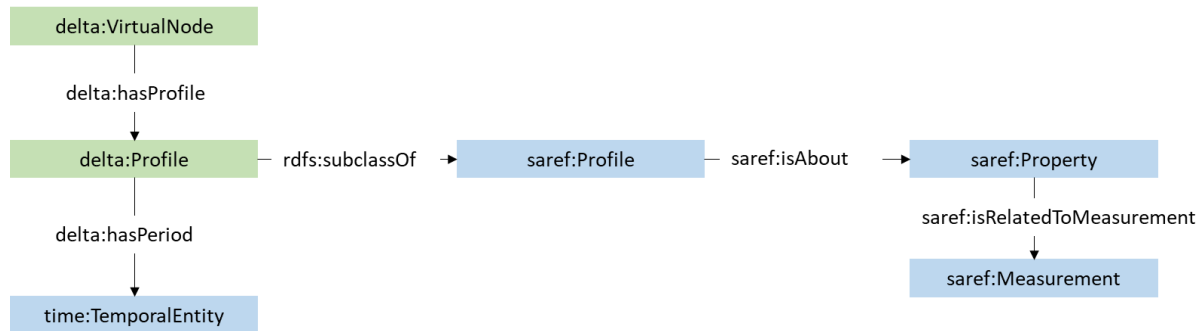


Figure 24 Ontology concepts related to Forecasted Profiling Interface

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-10.

7.1.2.5 Transactions

This interface stores the transactions made by the Virtual Node. Figure 25 summarizes the ontological concepts related to this interface.

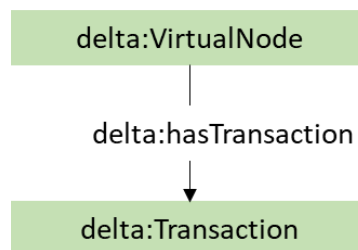


Figure 25 Ontology concepts related to Transactions

7.1.3 Load Forecasting

The goal of this component is to forecast the values that allows maximizing the availability of the assets in order to fulfil the energy promises established between the Demand Response Services to Market Stakeholders and the DELTA Aggregator/Energy Retailer. To achieve this goal two interfaces are defined: (1) Node Profiling and (2) Forecasted Profiling. Figure 26 summarizes such interfaces. The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex A.

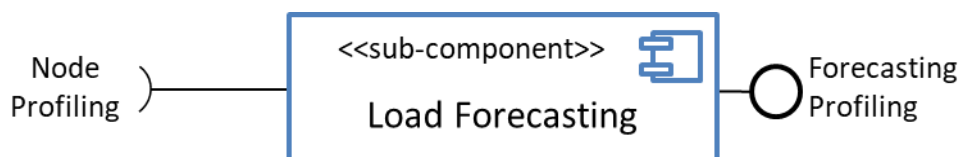


Figure 26 Interfaces for Load Forecasting

7.1.3.1 Node Profiling

This interface will provide other components with the DELTA Node Profiling. This profiling will be used to assess the DVN and the devices. Figure 27 summarizes the concepts in the ontology model that are related to this interface.

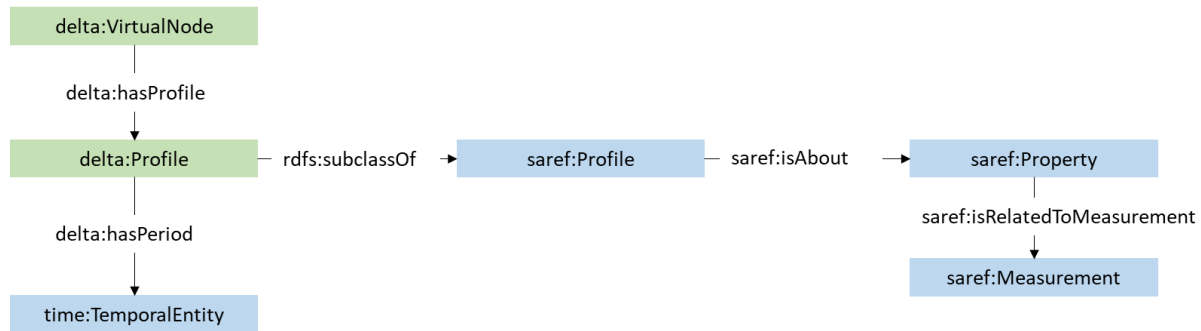


Figure 27 Ontology concepts related to Node Profiling

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-15.

7.1.3.2 Forecasted Profiling

This component sends the forecasted profiling to another component. This forecasted profiling, which is associated to a particular property in a time interval, is calculated based on previous profiles. Figure 28 summarizes the ontology concepts related to this interface.

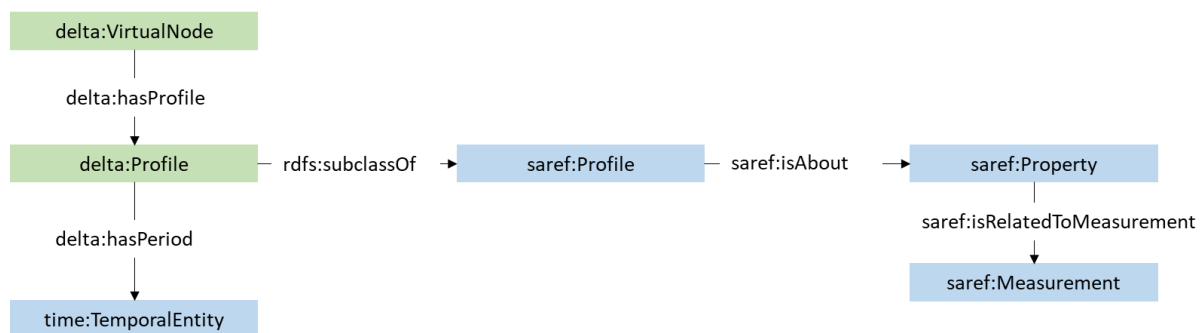


Figure 28 Ontology concepts related to Forecasted Profiling

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-10.

7.1.4 Inter/Intra Node Energy Matchmaking

The goal of this component aims at managing the DELTA Fog Enabled Agents of a certain Virtual DELTA Node by sending DR Signals to the underneath DELTA Fog Enabled Agents or to other Virtual DELTA Node. To achieve such goal, four interfaces are defined: (1) FEIDs Clusters, (2) DVN Clusters, (3) Status signals and (4) DR Signals. Figure 29 depicts these interfaces. The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex A.

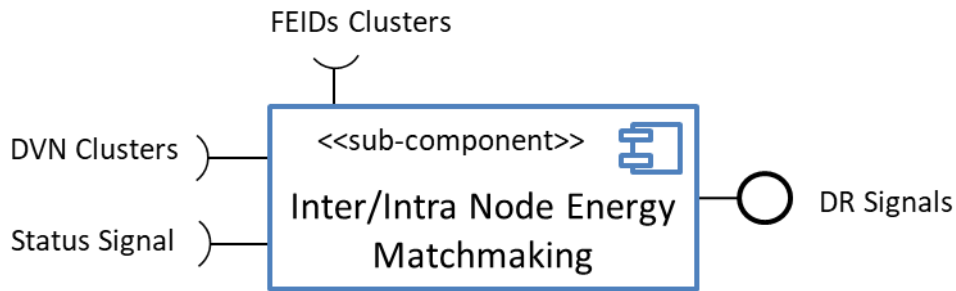


Figure 29 Interfaces for Inter/Intra Node Energy Matchmaking

7.1.4.1 FEIDs Clusters

This interface provides the components with the clusters and devices that belongs to a FEID. Figure 30 summarizes the ontology concepts that are related to this interface.

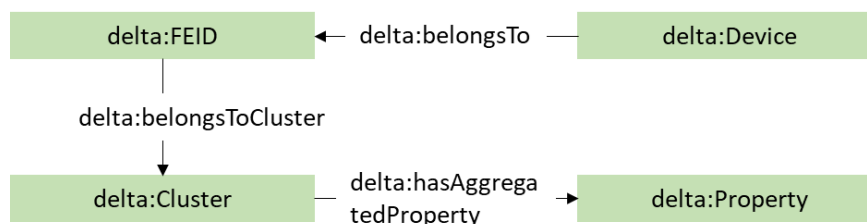


Figure 30 Ontology concepts related to the FEIDs Clusters

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-8

7.1.4.2 DVN Clusters

This interface provides the components with the clusters of each virtual node. Figure 31 summarizes the ontology concepts that are related to this interface.



Figure 31 Ontology concepts related to DVN Clusters

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-6.

7.1.4.3 Status Signals

This interface sends the status signals associated to a Virutal Node. Figure 32 shows the ontology concepts related to such signals.

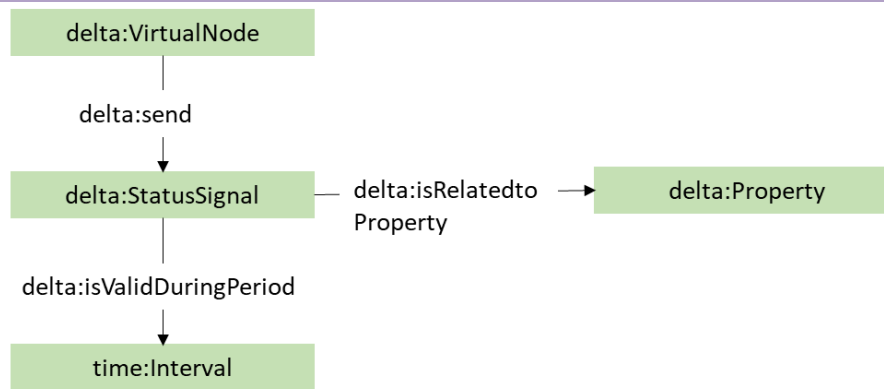


Figure 32 Ontology concepts related to Status Signal

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-15 1.A-18.

7.1.4.4 DR Signals

This interface sends a set of DR Signals in order to request and offer energy or to inform about the consensus related to a smart contract. Figure 33 shows the ontology concepts related to this DR signals. It is worth note that these signals are align with OpenADR signals.

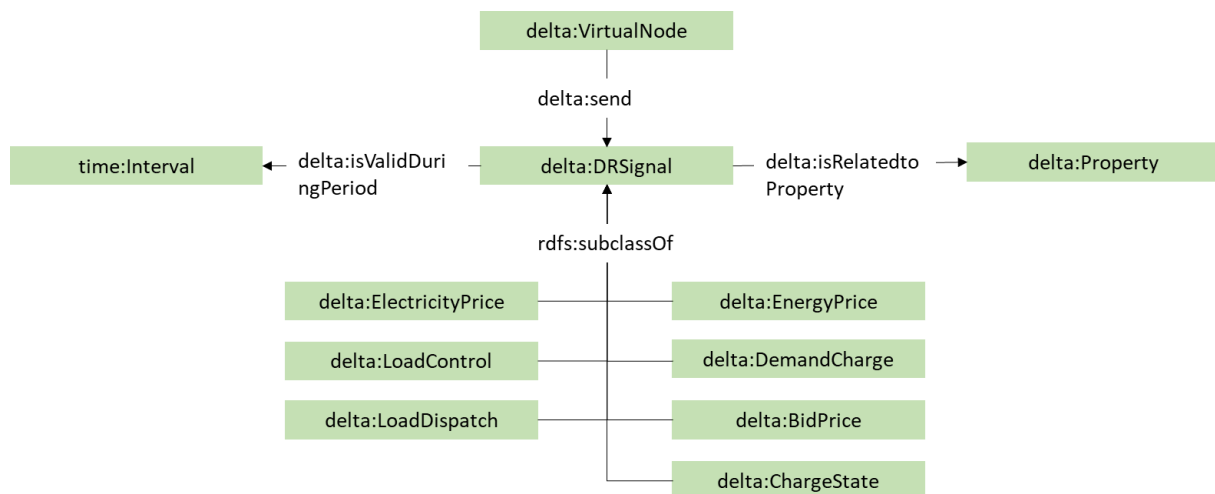


Figure 33 Ontology concepts related to DR Signal

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-5.

7.1.5 Consumer/Prosumer Clustering

This component implements techniques to cluster and group in segments the different DELTA Fog Enabled Agents allocated in the same DELTA Virtual Node Platform. The goal of this clustering is to assign each DELTA Fog Enabled Agent to a different Virtual DELTA Node. As a result, the clustering will produce suitable relationships for the scopes of the platform. The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex A.

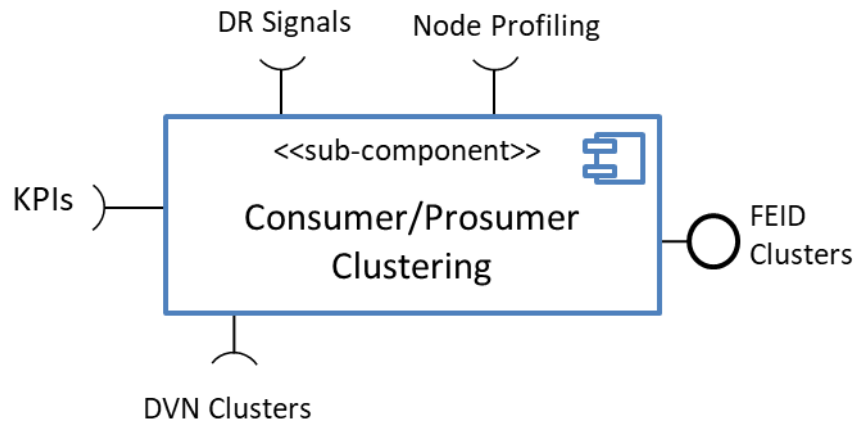


Figure 34 Interfaces for Consumer/Prosumer Clustering

7.1.5.1 DVN Clusters

This interface provides the components with the clusters of each virtual node. Figure 35 summarizes the ontology concepts that are related to this interface.



Figure 35 Ontology concepts related to DVN Clusters

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-6.

7.1.5.2 KPIs

This interface provides the components with the key performance indicators (KPIs) for different entities. Figure 36 summarizes the ontology concepts related to this interface.

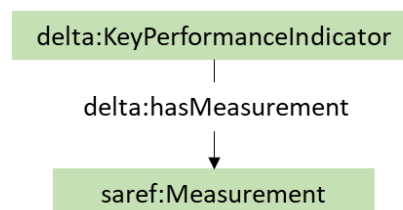


Figure 36 Ontology concepts related to KPIs

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-13.

7.1.5.3 DR Signals

This interface sends a set of DR Signals in order to request and offer energy or to inform about the consensus related to a smart contract. Figure 37 shows the ontology concepts related to this DR signals. It is worth note that these signals are align with OpenADR signals.

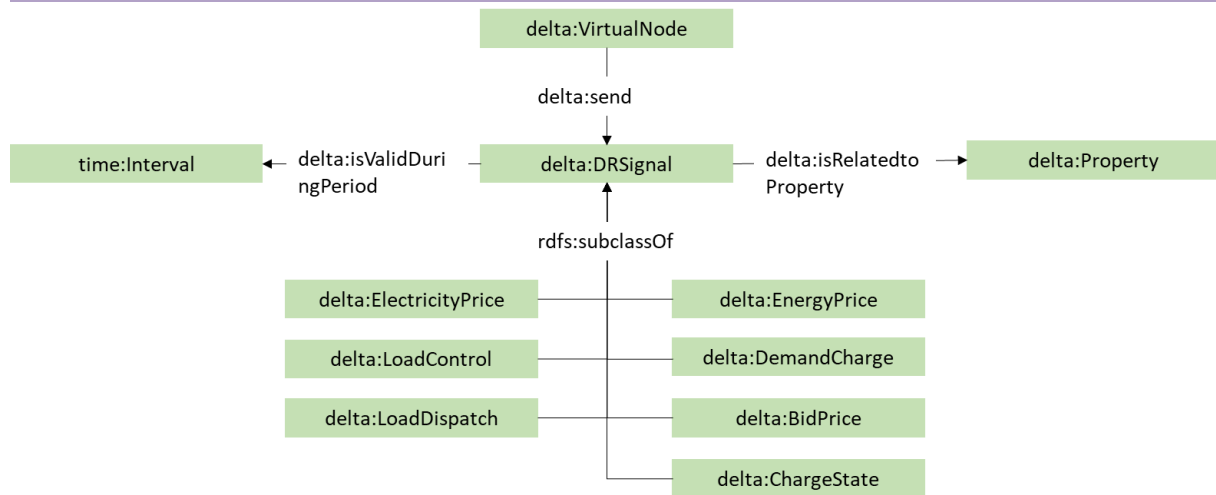


Figure 37 Ontology concepts related to DR Signal

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-5.

7.1.5.4 Node Profiling

This interface will provide other components with the DELTA Node Profiling. This profiling will be used to assess the DVN and the devices. Figure 38 summarizes the concepts in the ontology model that are related to this interface.

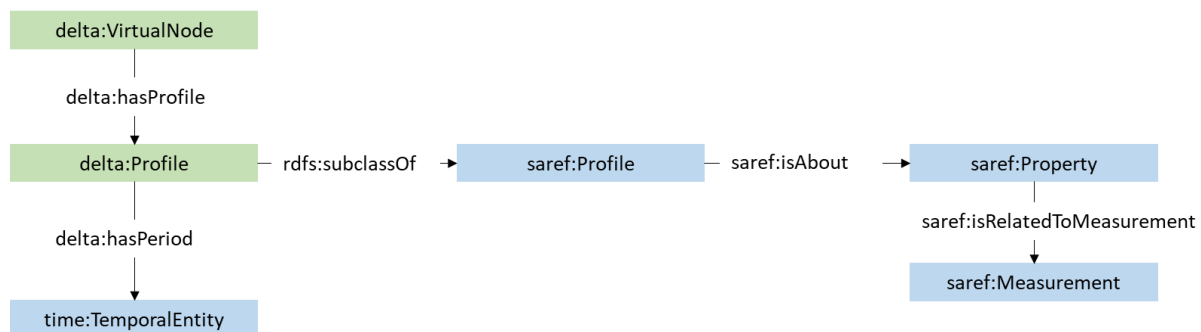


Figure 38 Ontology concepts related to Node Profiling

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-15.

7.1.5.5 FEIDs Clusters

This interface provides the components with the FEIDs cluster associated to a particular VirtualNode. Figure 39 summarizes the ontology concepts that are related to this interface.

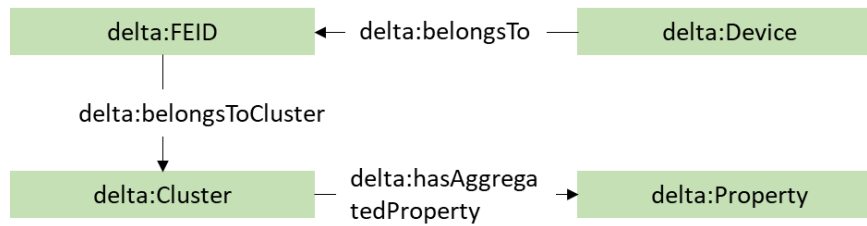


Figure 39 Ontology concepts related to the FEIDs Clusters

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-8.

7.2 Aggregator

This subsection describes the input and outputs interfaces for each subcomponent in the Aggregator.

7.2.1 Grid Stability Simulation Engine

This component is in charge of identifying issues in the physical level, voltage and frequency fluctuations and offers predictions of such physical constraints. To receive the needed information to reach such goal and to provide the obtained information to other components, two interfaces are defined: (1) Aggregated profiling and (2) Voltage & Frequency Constraints. The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex A.

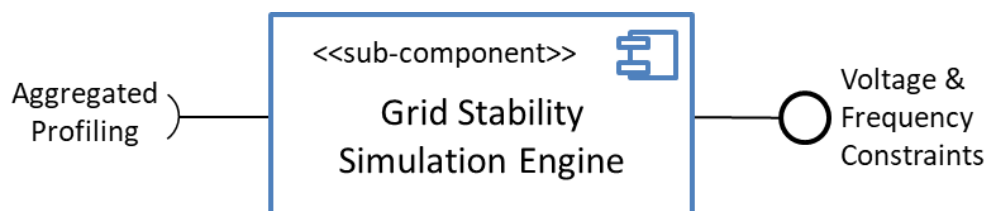


Figure 40 Interfaces for Grid Stability Simulation Engine

7.2.1.1 Aggregated Profiling

This interface provides the profile related to an aggregator. This profile, as the DVN profile, is associated to a property measurement in a time interval. Figure 41 represents the ontology concepts related to this interface.

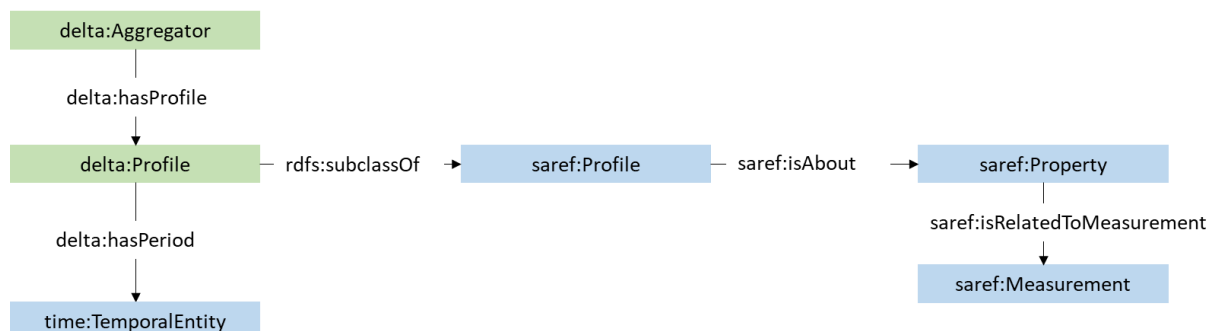


Figure 41 Ontology concepts related to the Aggregated Profiling

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-1.

7.2.1.2 Voltage & Frequency Constraints

This interface represents the constraints of the grid, i.e., the voltage and frequency that are calculated by a device. Figure 42 summarizes the ontology concepts related to this interface

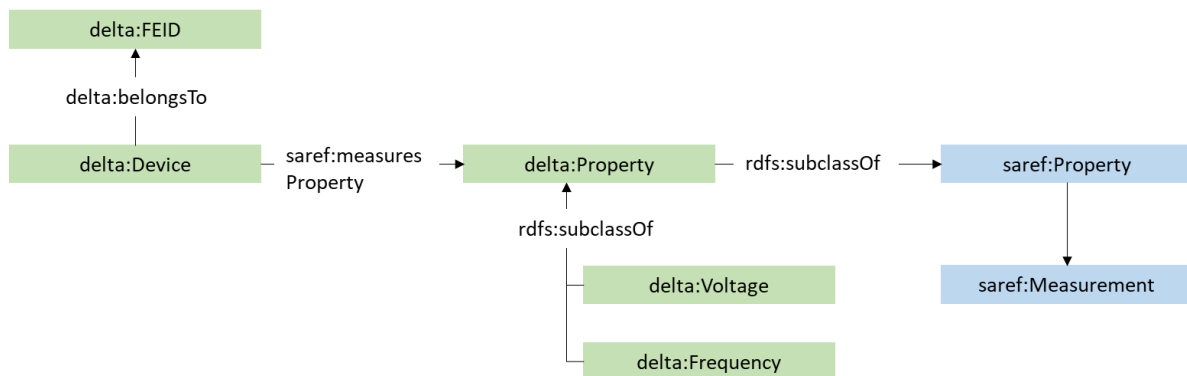


Figure 42 Ontology concepts related to the Voltage & Frequency Constraints

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-21.

7.2.2 Self Portfolio Energy Balancing

The Self Portfolio Energy Balancing component aims at considering the energy market behavior into account to produce assessments in the DELTA Aggregator/Energy Retailer portfolio. Its goal is to generate Bids that are sent to the external oracle Demand Response Services, and to produce a market settlement that specifies a certain behavior that the DELTA Aggregator/Energy Retailer should consider having. Therefore, this component has three interfaces, which are summarized in Figure 43: (1) Forecasted Flexibility, (2) Aggregated profiling, (3) Energy price profiling, (4) Bids and (5) Market settlement. The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex A.

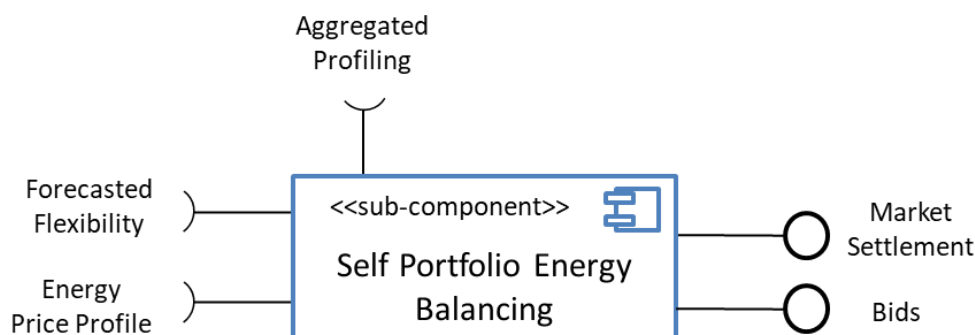


Figure 43 Interfaces for the Self Portfolio Energy Balancing

7.2.2.1 Forecasted Flexibility

This interface will provide the component with the profiling related to the flexibility forecast.

Regarding the ontology model, the information exchanged by this component in this interface corresponds to the following concepts that are summarized in Figure 44.

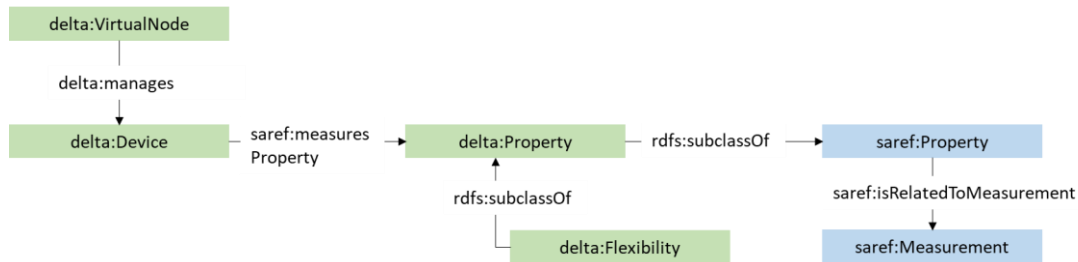


Figure 44 Ontology concepts related to Forecasted Flexibility

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-9.

7.2.2.2 Aggregated Profiling

This interface will provide other components with the aggregated profile. This profiling will be associated to a particular property and is available for a given time interval. Figure 45 summarizes the concepts in the ontology model that are related to this interface.

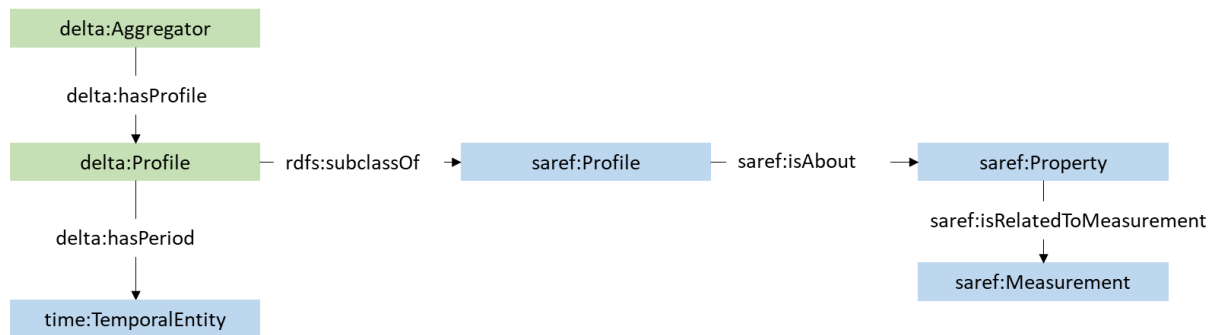


Figure 45 Ontology concepts related to the Aggregated Profiling

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-1.

7.2.2.3 Energy price Profiling

This interface will provide other components with the Node Profiling related to energy price. This profiling will be used to calculate the price bids. Figure 46 summarizes the concepts in the ontology model that are related to this interface.

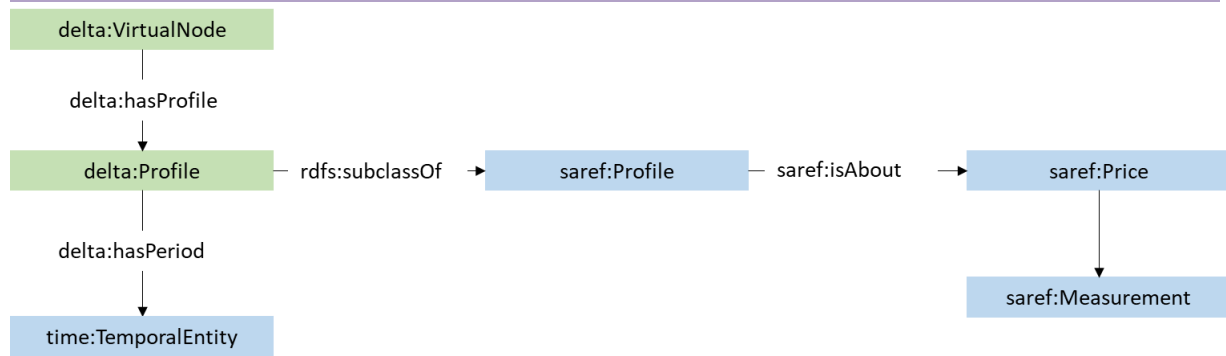


Figure 46 Ontology concepts related to Energy Price Profiling

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-7.

7.2.2.4 Bids

This interface will exchange the price bid for an amount of energy calculated by the aggregator. Figure 47 summarizes the concepts in the ontology model that are related to this interface.

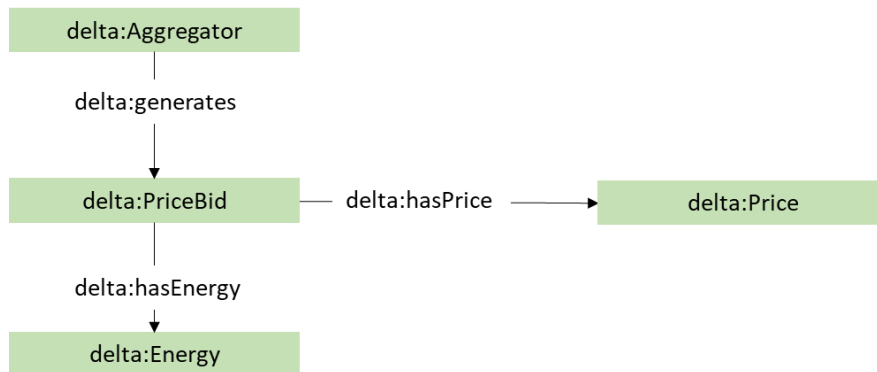


Figure 47 Ontology concepts related to Bids

7.2.2.5 Market settlement

This interface allows to provide the aggregator with the market settlement. Figure 48 summarizes the concepts in the ontology model that are related to this interface.

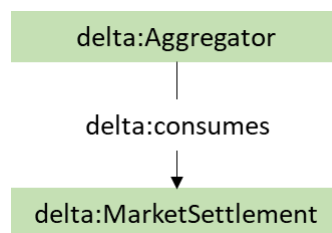


Figure 48 Ontology concepts related to the Market Settlement

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-141.A-1.

7.2.3 Energy Portfolio Segmentation & Classification

The aim of this component is to establish the allocation of the Virtual DELTA Nodes underneath the DELTA Aggregator/Energy Retailer. This classification is based on a clustering algorithm performed

by the Aggregator at their premises. After this first step, the Nodes would re-act and re-arrange themselves autonomously, based on indicators infused by the Aggregator. To carry out such clustering, this component needs: (1) the customers information, (2) the rewards of the consumer of each virtual node, (3) the forecasted flexibility and (4) the aggregated profiling. Figure 49 provides an overview of the interfaces that exchange this data. The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex A.

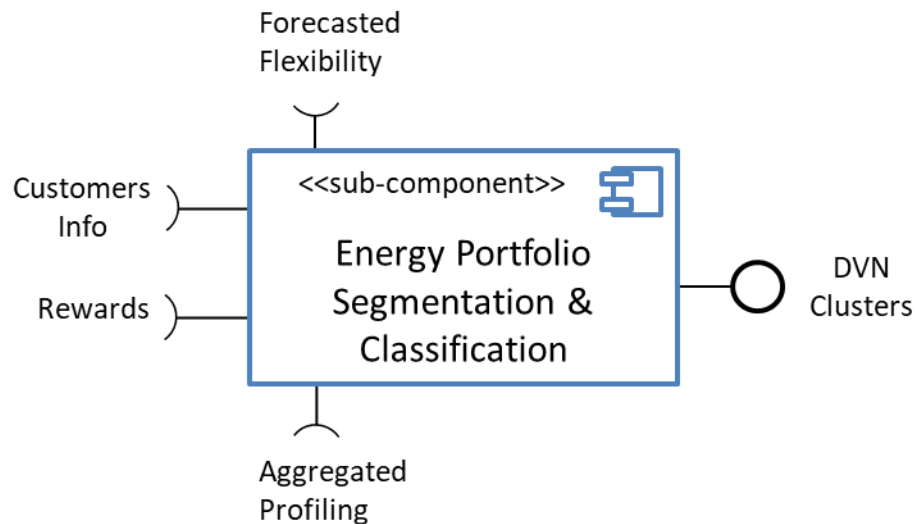


Figure 49 Interfaces for the Energy Portfolio Segmentation & Classification

7.2.3.1 Customers information

This interface provides all the information related to each DELTA customer, including the managed devices. Figure 50 represents the ontology concepts related to this interface.

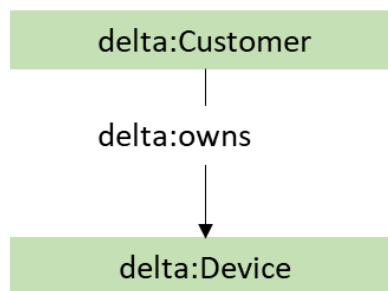


Figure 50 Ontology concepts related to the Customers Information

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-4.

7.2.3.2 Rewards

This interface provides the rewards associated to each consumer, which were previously obtained during the gamification process. Figure 51 summarizes the ontology concepts related to this interface.

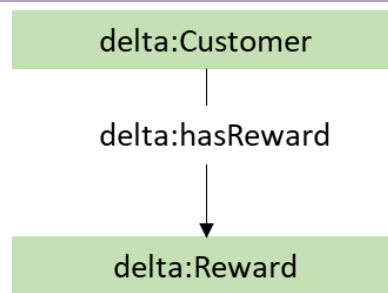


Figure 51 Ontology concepts related to the Rewards

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-16.

7.2.3.3 Forecasted Flexibility

This interface will provide the component with the profiling related to the flexibility forecast

Regarding the ontology model, the information exchanged by this component in this interface corresponds to the following concepts that are summarized in Figure 52.

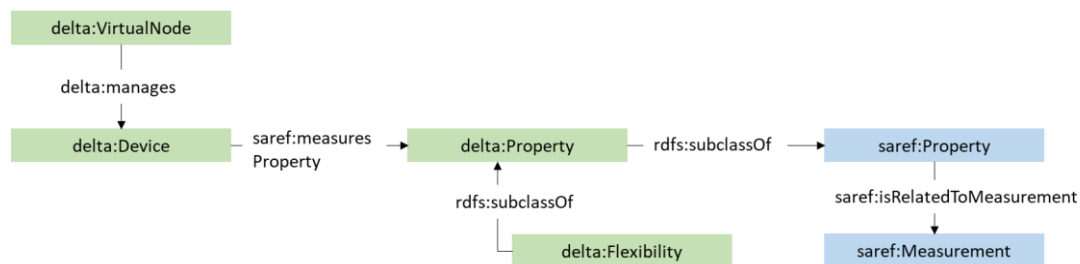


Figure 52 Ontology concepts related to Forecasted Flexibility

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-10.

7.2.3.4 Aggregated Profiling

This interface will provide other components with the aggregated profiles. This profiling will be associated to a particular property and is available for a given time interval. Figure 53 summarizes the concepts in the ontology model that are related to this interface.

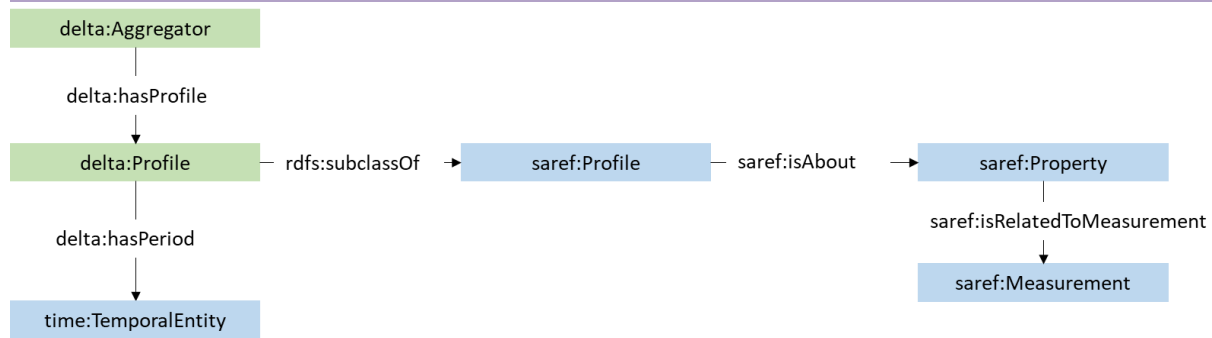


Figure 53 Ontology concepts related to the Aggregated Profiling

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-1.

7.2.3.5 DVN Clusters

This interface provides the components with the clusters of each virtual node. Figure 54 summarizes the ontology concepts that are related to this interface.



Figure 54 Ontology concepts related to DVN Clusters

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-6.

7.2.4 Energy Market Price Forecast

This component aims at predicting the energy market prices. To achieve that goal, it has to receive the voltage and frequency constraints of the grid. Therefore, this component has two interfaces: (1) the voltage & frequency constraints interface and (2) the energy price profile. The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex A.

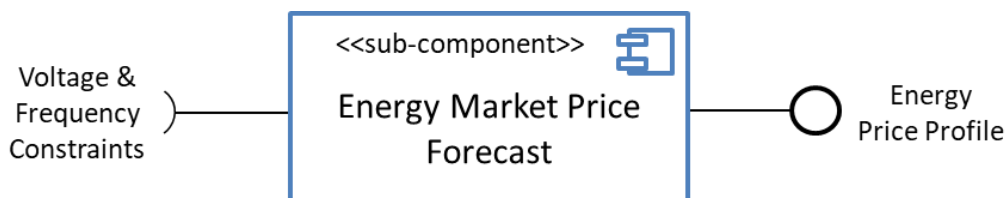


Figure 55 Interfaces for Energy Market Price Forecast

7.2.4.1 Voltage & Frequency Constraints

This interface represents the constraints of the grid, i.e., the voltage and frequency that are calculated by a device. Figure 56 summarizes the ontology concepts related to this interface

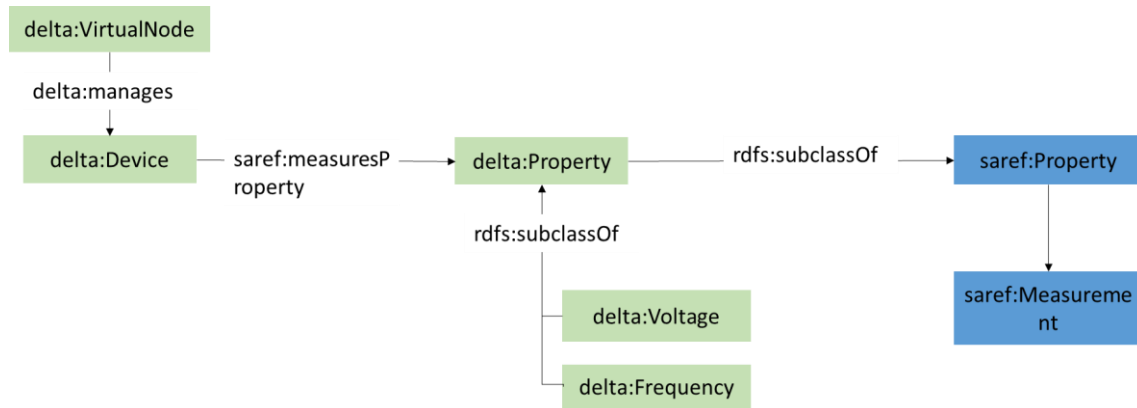


Figure 56 Ontology concepts related to the Voltage & Frequency Constraints Interface

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-21.

7.2.4.2 Energy Price Profile

This interface will provide other components with the Node Profiling related to energy price. This profiling will be used to calculate the price bids. Figure 57 summarizes the concepts in the ontology model that are related to this interface.

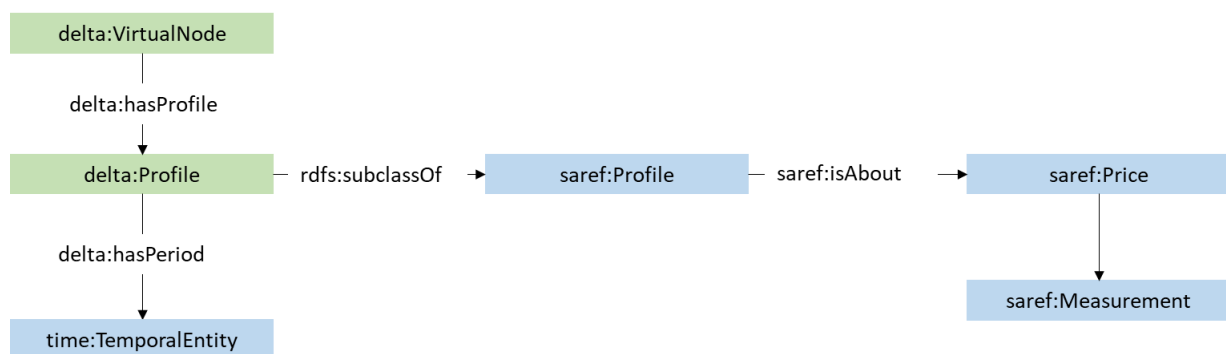


Figure 57 Ontology concepts related to Energy Price Profile

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-7.

7.2.5 DR & Flexibility Forecasting

The Node Flexibility Data Monitoring and Profiling component is in charge of providing the balance responsibility that a DELTA Aggregator/Energy Retailer may count with. To achieve its goal, this component has two interfaces: (1) Aggregate profiling, (2) Customer information, (3) Forecasting Profiling and (4) Forecasted flexibility. The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex A.

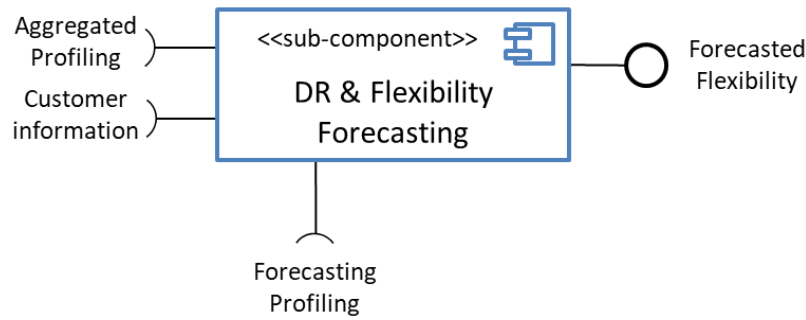


Figure 58 Interfaces for DR & Flexibility Forecasting

7.2.5.1 Aggregated Profiling

This interface will provide other components with the aggregated Profiling. This profiling will be associated to a particular property and is available for a given time interval. Figure 59 summarizes the concepts in the ontology model that are related to this interface.

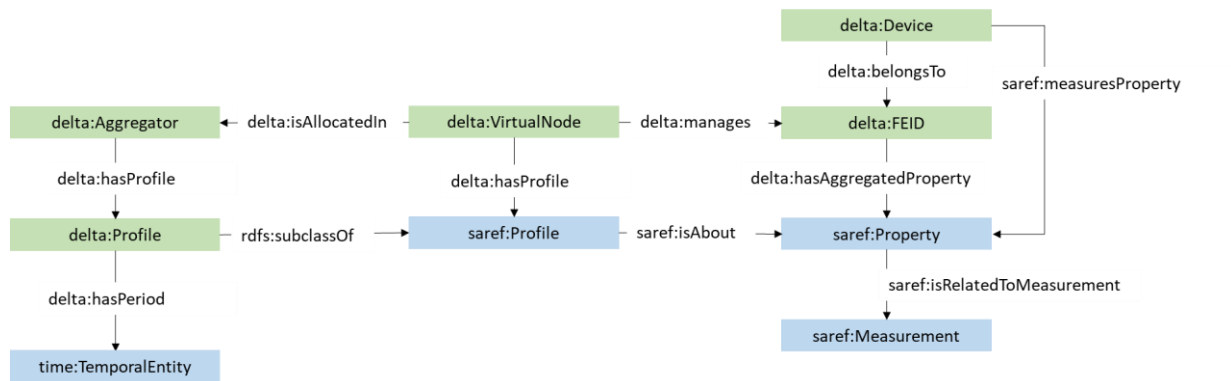


Figure 59 Ontology concepts related to the Aggregated Profile

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-1.

7.2.5.2 Forecasting Profiling

This interface will provide other components with the aggregated Profiling. This profiling will be associated to a particular property and is available for a given time interval. Figure 60 summarizes the concepts in the ontology model that are related to this interface.

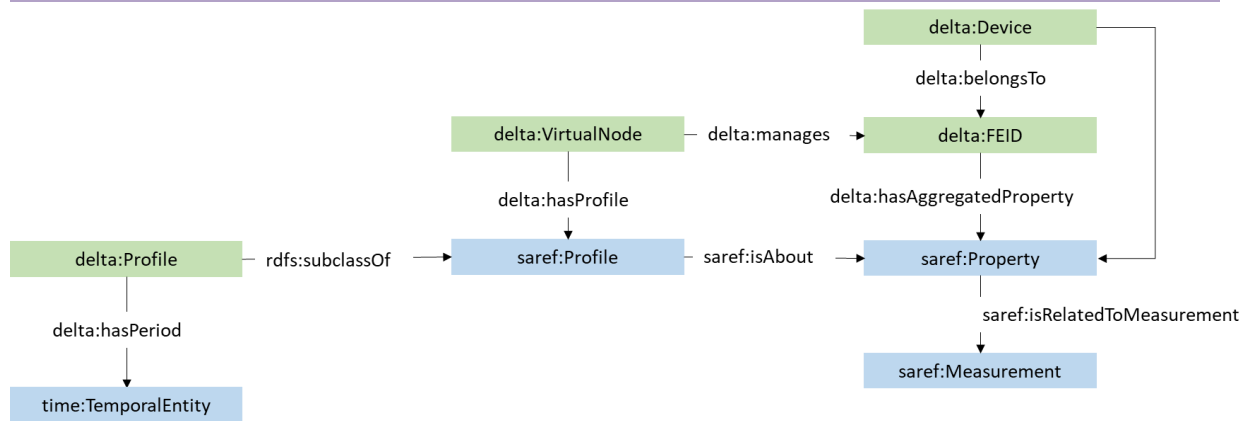


Figure 60 Ontology concepts related to the Forecasted profiling

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-10.

7.2.5.3 Customers information

This interface provides all the information related to each customer, including the managed devices. Figure 61 represents the ontology concepts related to this interface.

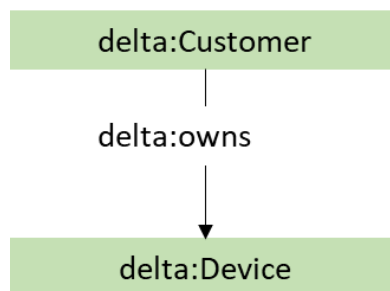


Figure 61 Ontology concepts related to the Customers Information

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-4.

7.2.5.4 Forecasted Flexibility

This component sends the forecasted profiling to another component. This interface will provide the component with the profiling related to the flexibility forecast.

Regarding the ontology model, the information exchanged by this component in this interface corresponds to the following concepts that are summarized in Figure 62.

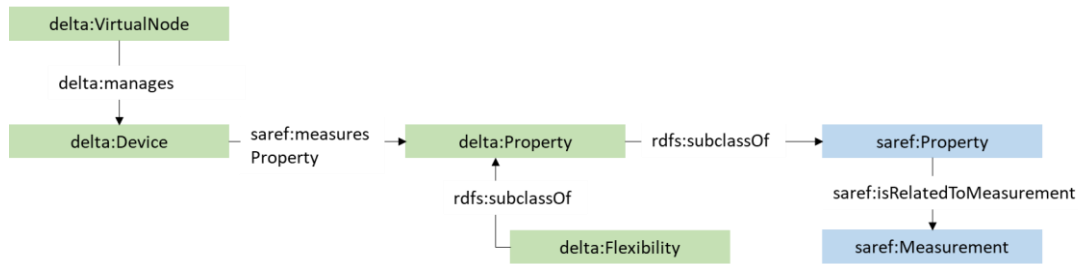


Figure 62 Ontology concepts related to Forecasted Flexibility

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-9.

7.2.6 Asset Handling Optimization

This component aims at processing incoming DR signals sent by the external oracle DR Services to Market Stakeholders. It re-distributes the assets of a certain DELTA Aggregator / Energy Retailer, or create DR signals to forward to the underneath Virtual DELTA Nodes. In addition, this component generates Smart Contract that reflect the compromise established between the DR Services to Market Stakeholders and the DELTA Aggregator / Energy Retailer. The Asset Handling Optimization needs three interfaces, which are depicted in Figure 63: (1) DR Signal, (2) Aggregated Profile, (3) Transactions and (4) Market Settlement. The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex A.

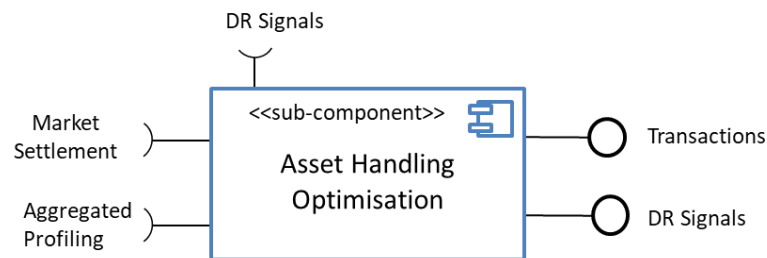


Figure 63 Interfaces for the Assets Handling Optimization

7.2.6.1 DR Signal

This interface sends a set of DR Signals in order to ask or send information to components. Figure 64 shows the ontology concepts related to this DR signals. It is worth note that these signals are align with OpenADR signals.

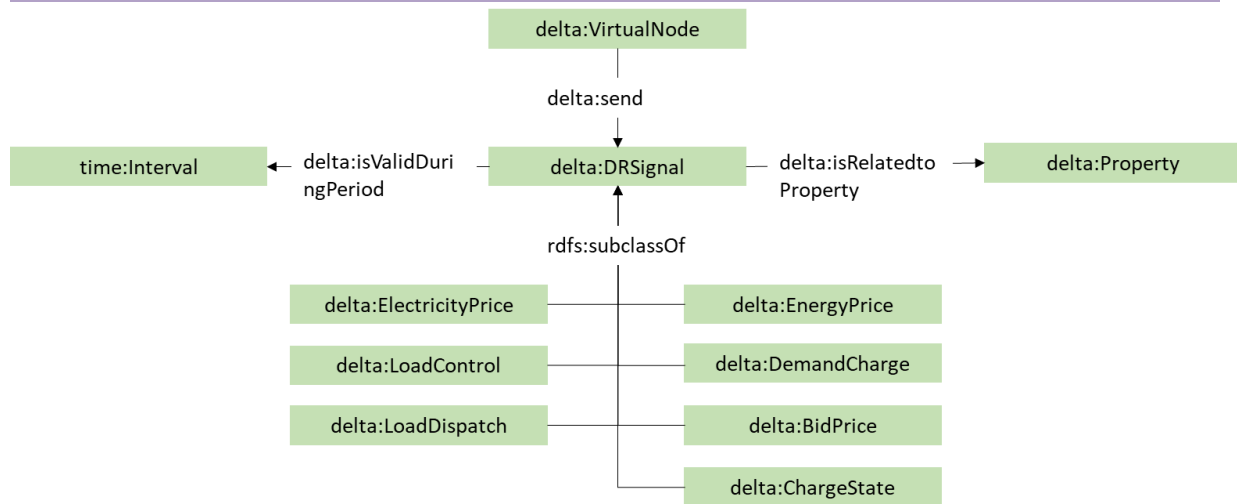


Figure 64 Ontology concepts related to DR Signal

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-5.

7.2.6.2 Aggregated Profiling

This interface will provide other components with the aggregated profile. This profiling will be associated to a particular property and is available for a given time interval. Figure 65 summarizes the concepts in the ontology model that are related to this interface.

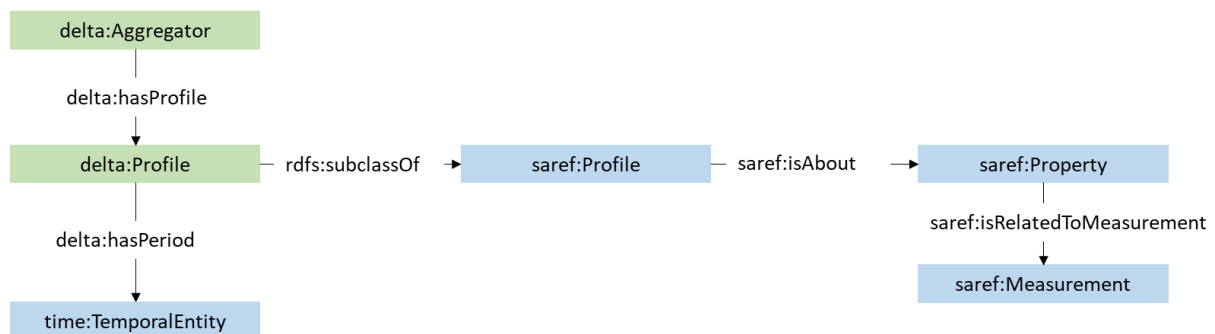


Figure 65 Ontology concepts related to the Aggregate Profiling

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-1.

7.2.6.3 Transactions

This interface stores the transactions made by the Aggregator. Figure 66 summarizes the ontological concepts related to this interface.

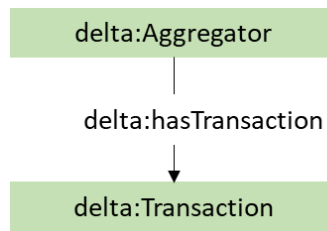


Figure 66 Ontology concepts related to Transactions

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-20.

7.2.6.4 Market settlement

This interface allows to provide the aggregator with the market settlement. Figure 67 summarizes the concepts in the ontology model that are related to this interface.

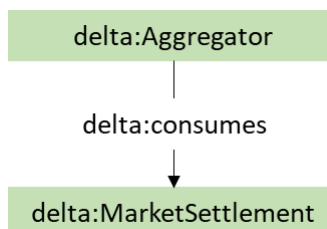


Figure 67 Ontology concepts related to the Market Settlement

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-14.

7.2.7 Node Flexibility Data Monitoring and Profiling

This component gathers data related to the underneath elements and aggregates them in order to provide a unified value of flexibility and profiling for the aggregator. To achieve such goal, this component has two interfaces: (1) Node Profiling and (3) Aggregated Profiling. The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex A.

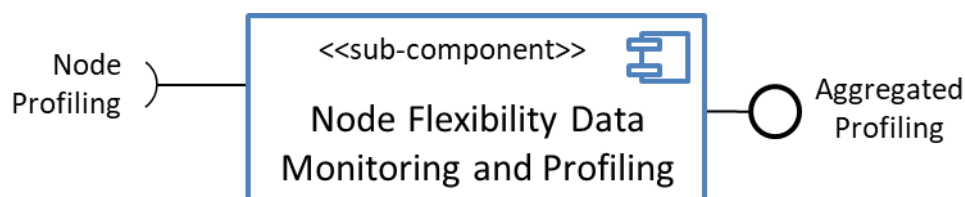


Figure 68 Interfaces for the Node Flexibility Data Monitoring and Profiling

7.2.7.1 Node Profiling

This interface will provide other components with the DELTA Node Profiling. This profiling will be used to assess the DVN and the devices. Figure 69 summarizes the concepts in the ontology model that are related to this interface.

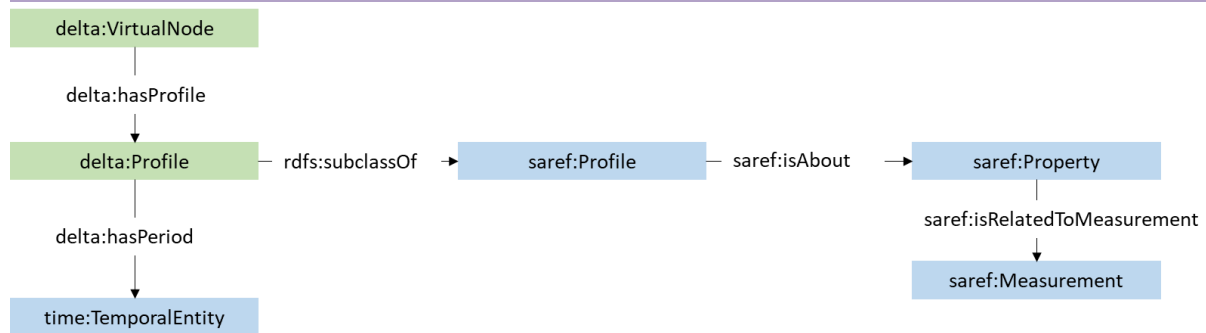


Figure 69 Ontology concepts related to DELTA Node Profiling

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-15.

7.2.7.2 Aggregated Profile

The Aggregated Profile interface represents several metrics related to the properties associated to the Aggregator. Each profile is available in a time interval. Figure 70 shows the ontological concepts related to this profiling.

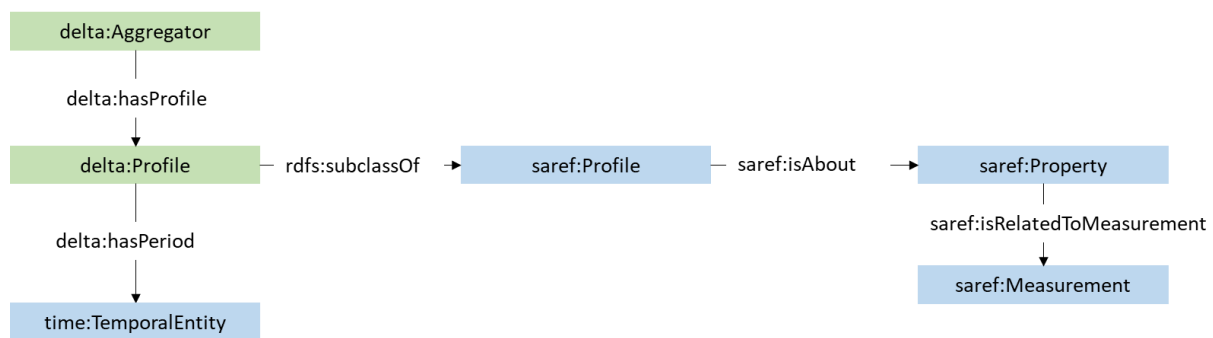


Figure 70 Ontology concepts related to the Aggregated Profile

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-1.

7.3 DELTA Fog Enable Agent

This subsection specifies the interfaces related to the DELTA Fog Enable Agent subcomponents.

7.3.1 Lightweight Toolkit (FEID) per Customer

The Lightweight Toolkit per Customer is in charge of processing incoming DR Signals and, at the same time, forwarding to upper layers the observed values and computed metrics. To achieve such goal, it needs ten interfaces: (1) Comfort Setting, (2) System Constraints, (3) Smart contracts, (4) DR Signals, (5) Status Signal, (6) Flexibility Forecast, (7) Historical Consumption, (8) Historical Generation, (9) Voltage & Frequency and (10) Transactions. Figure 71 summarizes all these interfaces.

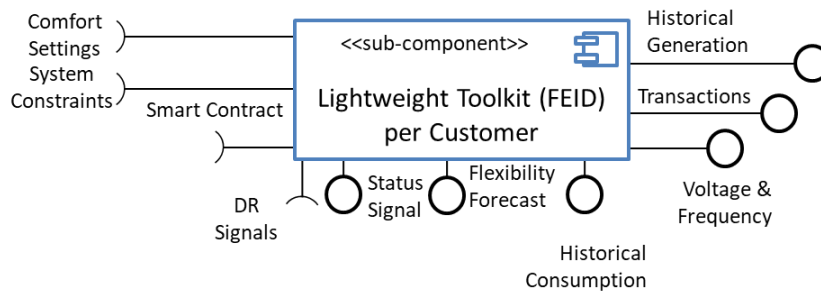


Figure 71 Interfaces for Lightweight Toolkit per Customer

7.3.1.1 Comfort Settings

This interface receives some comfort settings specified by the owner of the Lightweight Toolkit (FEID) per Customer. These settings modify the regular behaviour of this asset. Figure 72 summarizes the ontology concepts related to this comfort settings interface.

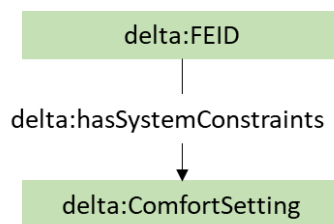


Figure 72 Ontology concepts related to Comfort Settings

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-3.

7.3.1.2 System Constraints

This interface receives some system constraints specified by the owner of the Lightweight Toolkit (FEID) per Customer, which will modify the regular behaviour of this asset. Figure 73 summarizes the ontology concepts related to this interface.

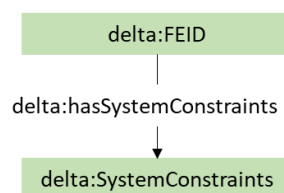


Figure 73 Ontology concepts related to System Constraints

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-19.

7.3.1.3 Smart Contracts

This interface exchanges the smart contracts that are stored in the blockchain. Such contract includes the value of the expected DELTA property and the agreed payment. Figure 74 summarizes the ontological concepts related to this interface.

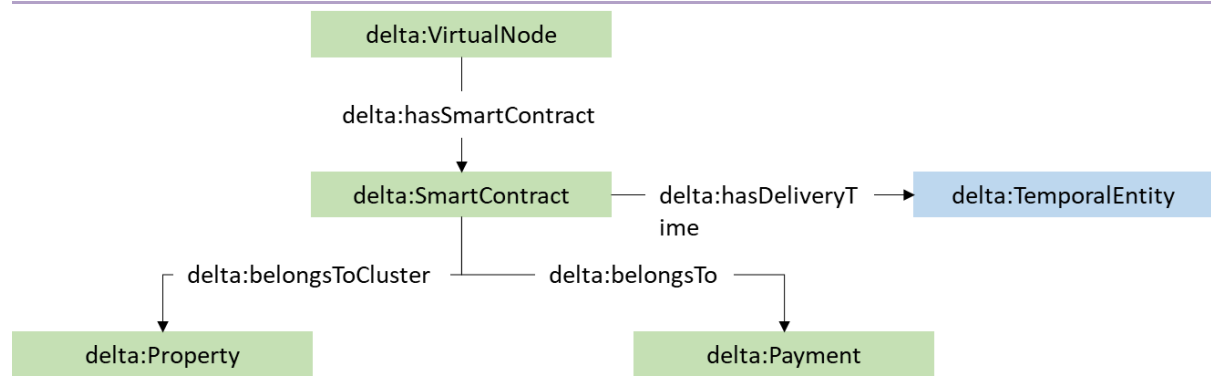


Figure 74 Ontology concepts related to Smart Contracts

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-17.

7.3.1.4 DR Signals

This interface sends a set of DR Signals in order to ask or send information to components. Figure 75 shows the ontology concepts related to this DR signals. It is worth note that these signals are align with OpenADR signals.

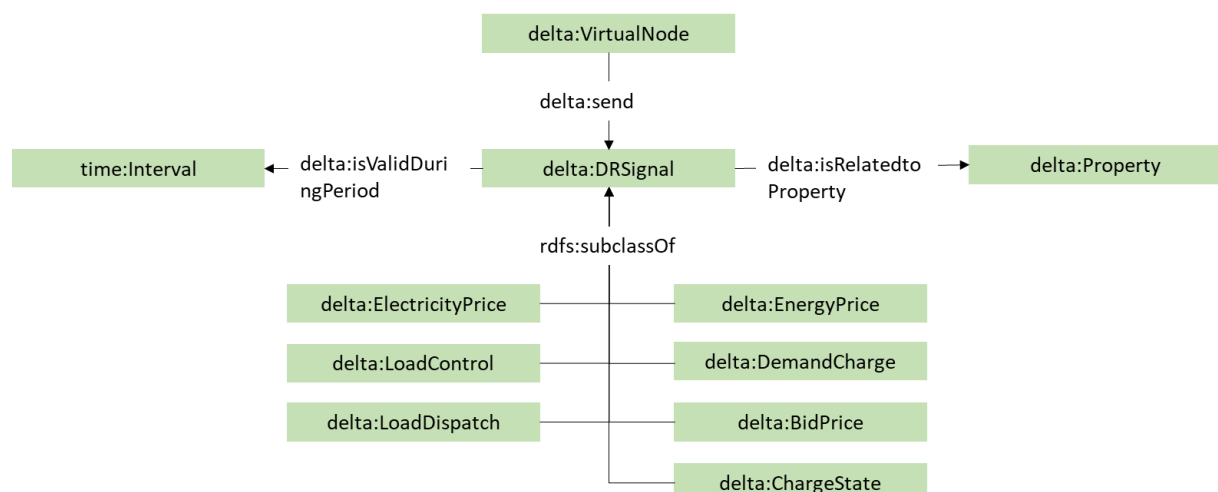


Figure 75 Ontology concepts related to DR Signal

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-5.

7.3.1.5 Status Signal

This interface sends the status signals associated to a Virtual Node. Figure 76 shows the ontology concepts related to such signals.



Figure 76 Ontology concepts related to Status Signal

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-18.

7.3.1.6 Flexibility Forecast

This component sends the forecasted profiling to another component. This interface will provide the component with the profiling related to the flexibility forecast

Regarding the ontology model, the information exchanged by this component in this interface corresponds to the following concepts that are summarized in Figure 77.

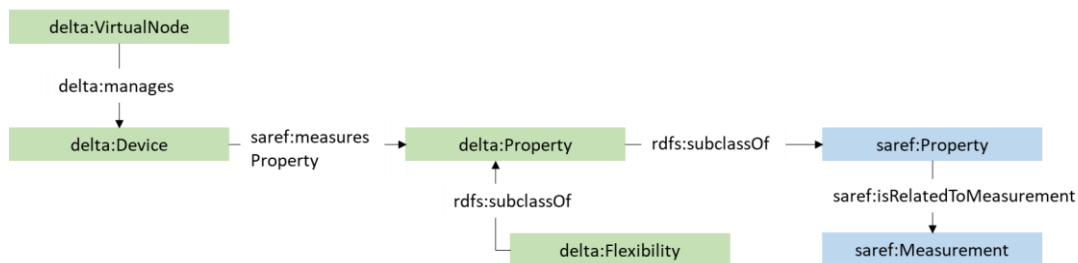


Figure 77 Ontology concepts related to Flexibility Forecast

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-9.

7.3.1.7 Historical Consumption

This interface will provide the component with the historical energy consumption, either related to each device or aggregated in FEIDS.

Regarding the ontology model, the information exchanged by this component in this interface corresponds to the following concepts that are summarized in Figure 78.

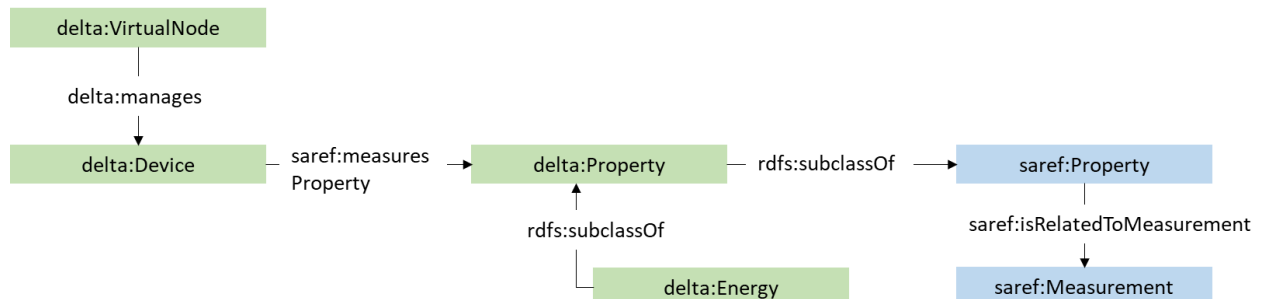


Figure 78 Ontology concepts related to Historical Consumption

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-11.

7.3.1.8 Historical generation

This interface will provide the component with the historical energy generation, either related to each device or aggregated in FEIDS.

Regarding the ontology model, the information exchanged by this component in this interface corresponds to the following concepts that are summarized in Figure 79.

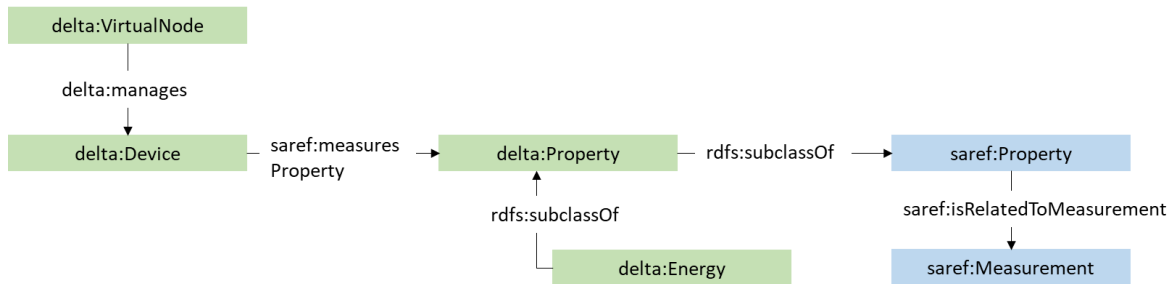


Figure 79 Ontology concepts related to Historical Generation

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-12.

7.3.1.9 Voltage & Frequency

This interface represents the constraints of the grid, i.e., the voltage and frequency that are calculated by a device. Figure 80 summarizes the ontology concepts related to this interface

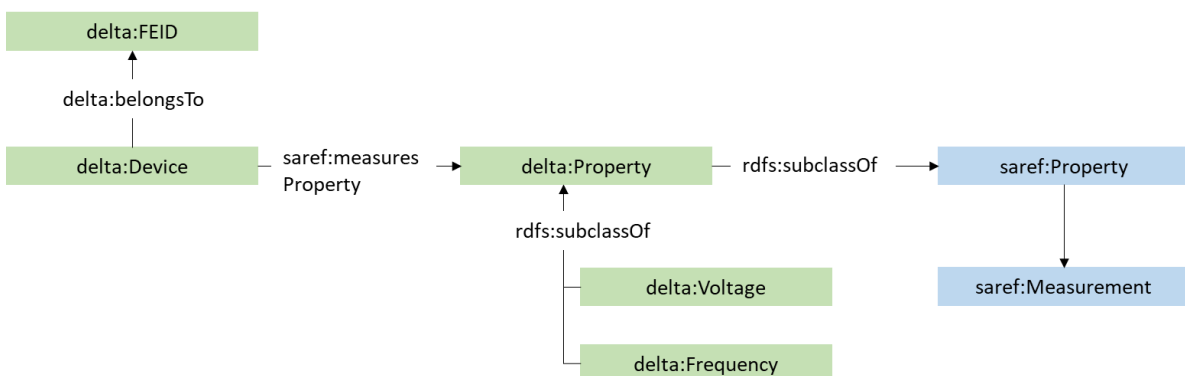


Figure 80 Ontology concepts related to the Voltage & Frequency Constraints

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-21.

7.3.1.10 Transactions

This interface stores the transactions made by the FEID. Figure 81 summarizes the ontological concepts related to this interface.

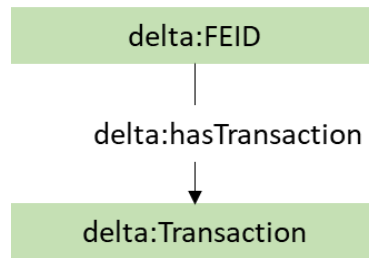


Figure 81 Ontology concepts related to Transactions

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-20.

7.4 Demand Response Settings to market stakeholders

This subsection describes the interfaces for each subcomponent related to the Demand Response Setting to market stakeholders.

7.4.1 DR Services to market stakeholders

This component represents the potential DSOs, TSOs or BRPs that would be able to interact with the DELTA platform by sending demand response signals to the main component DELTA Aggregator/Energy Retailers; which will process these requests and will respond with computed Bids. To achieve such goal, it needs two interfaces: (1) Bids and (2) DR Signals. Figure 82 summarizes these interfaces.

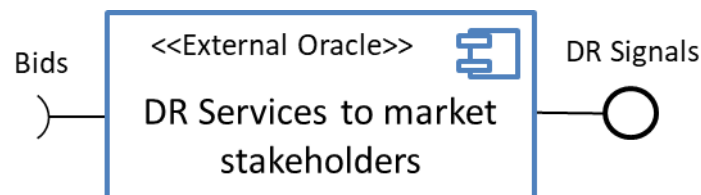


Figure 82 Interfaces for DR Services to market stakeholders

7.4.1.1 Bids

This interface will exchange the price bid for an amount of energy calculated by the aggregator. Figure 83 summarizes the concepts in the ontology model that are related to this interface.

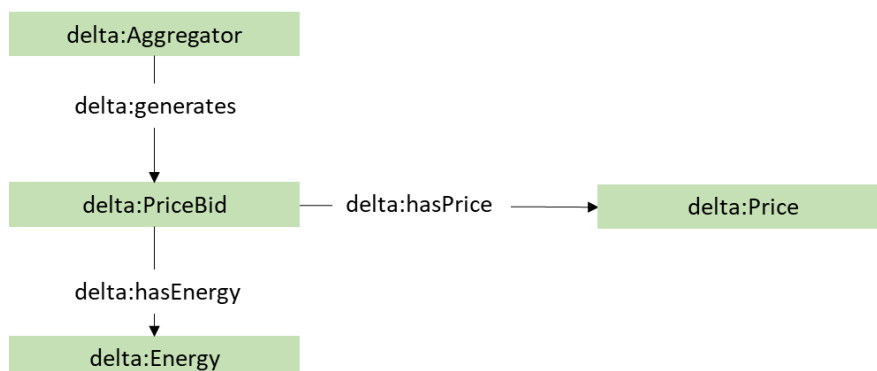


Figure 83 Ontology concepts related to Bids

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-2.

7.4.1.2 DR Signals

This interface sends a set of DR Signals in order to ask or send information to components. Figure 84 shows the ontology concepts related to this DR signals. It is worth note that these signals are align with OpenADR signals.

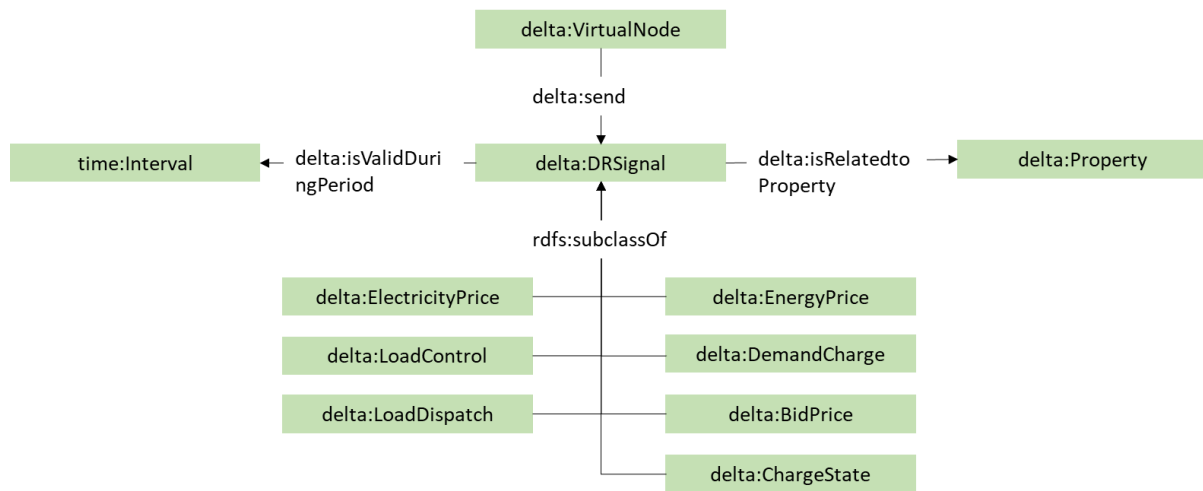


Figure 84 Ontology concepts related to DR Signal

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-5.

7.5 Innovative Customer Engagement Tools

This subsection describes the interfaces for each subcomponent related to the Innovative Customer Engagement Tools.

7.5.1 Demand Response Visualization Kit

This component is a web-based tool that will visualize real-time and historical energy information including consumption, generation, appropriate DR strategies for the Virtual Node, as well as financial and environmental data. Its purpose is to present useful information for the effective application of DR programs. Since it is a component that provides visualization information, it has no output interfaces. This module is considered as a holistic tool for visualizing Demand Response information to all involved system stakeholders. Thus, there will be two type of visualizations, one for the Aggregator level and one for the Customer level. The DR Visualization Kit for the Aggregator level has eleven input interfaces that are presented in Figure 65: (1) Customers information, (2) Historical Consumption, (3) Historical Generation, (4) Forecasted Flexibility, (5) DR Signals, (6) Bids, (7) Rewards, (8) Energy price Profiling, (9) DVN Clusters, (10) Node Profiling and (11) Aggregated Profiling. The respective DR Visualization Kit for the Customer level has three input interfaces, as depicted in Figure 66. These input interfaces are: (1) Rewards, (2) DR Signals and (3) FEID Energy Profile.

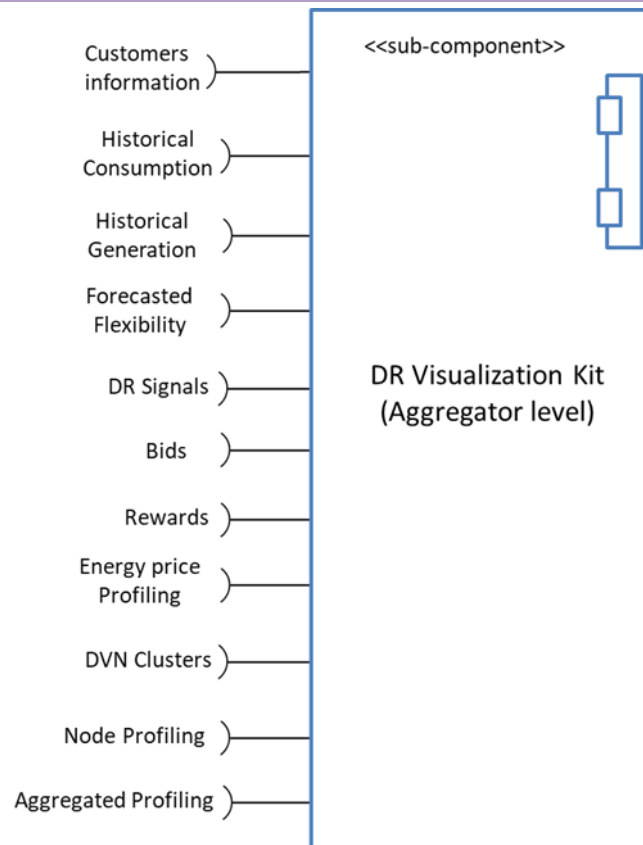


Figure 85 Interfaces for the Demand Response Visualization Kit (Aggregator Level)

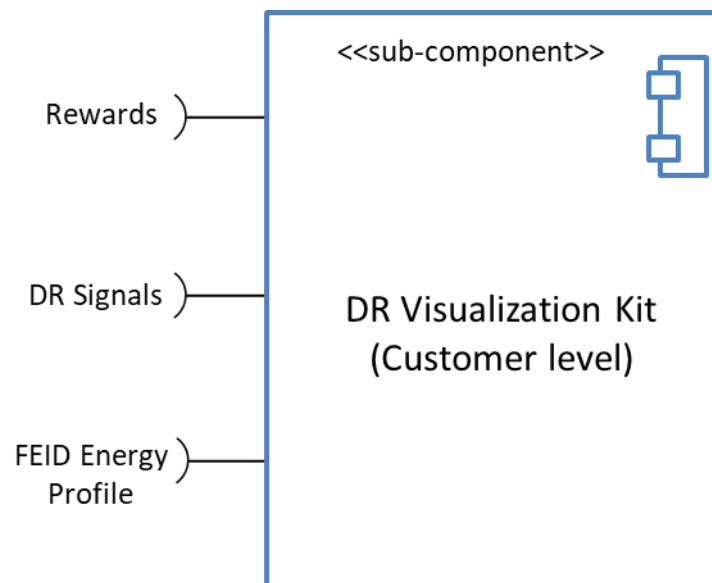


Figure 86 Interfaces for the Demand Response Visualization Kit (Customer Level)

7.5.1.1 Customers information

This interface provides all the information related to each DELTA customer, including the managed devices. Figure 87 represents the ontology concepts related to this interface.

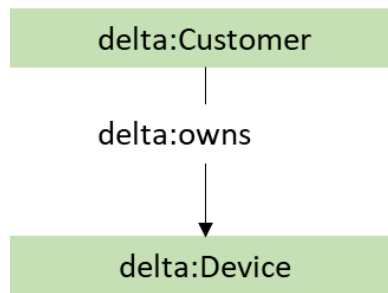


Figure 87 Ontology concepts related to the Customers Information

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-4.

7.5.1.2 Historical consumption

This interface will provide the component with the historical energy consumption, either related to each device or aggregated in FEIDS.

Regarding the ontology model, the information exchanged by this component in this interface corresponds to the following concepts that are summarized in Figure 88.

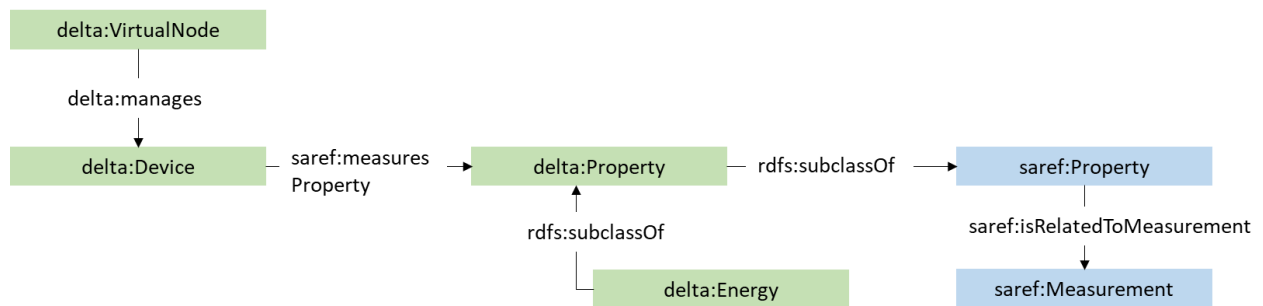


Figure 88 Ontology concepts related to Historical Consumption

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-11.

7.5.1.3 Historical generation

This interface will provide the component with the historical energy generation, either related to each device or aggregated in FEIDS.

Regarding the ontology model, the information exchanged by this component in this interface corresponds to the following concepts that are summarized in Figure 89.

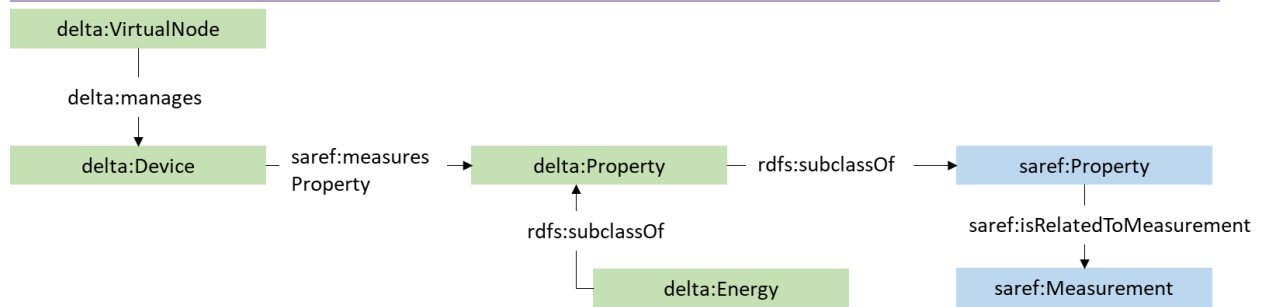


Figure 89 Ontology concepts related to Historical Generation

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-12.

7.5.1.4 Flexibility forecast

This interface will provide the component with the profiling related to the flexibility forecast.

Regarding the ontology model, the information exchanged by this component in this interface corresponds to the following concepts that are summarized in Figure 90.

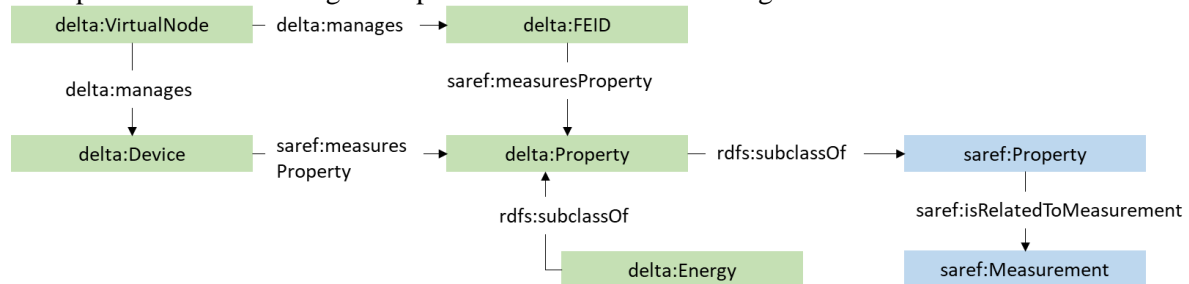


Figure 90 Ontology concepts related to Flexibility forecast

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-9.

7.5.1.5 DR Signals

This interface sends a set of DR Signals in order to ask or send information to components. Figure 91 shows the ontology concepts related to this DR signals. It is worth note that these signals are align with OpenADR signals.

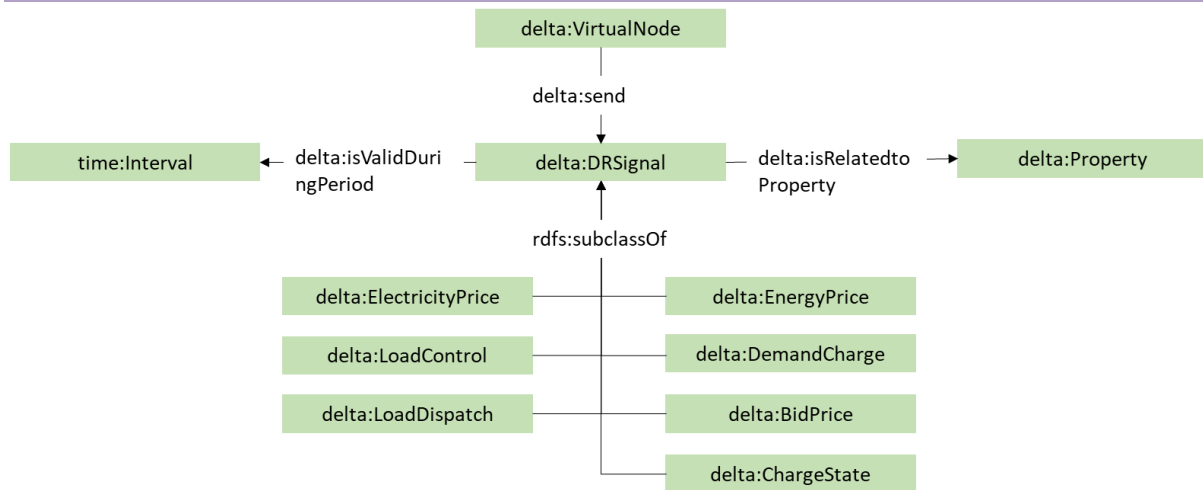


Figure 91 Ontology concepts related to DR Signal

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-5.

7.5.1.6 Bids

This interface will exchange the price bid for an amount of energy calculated by the aggregator. Figure 92 summarizes the concepts in the ontology model that are related to this interface.

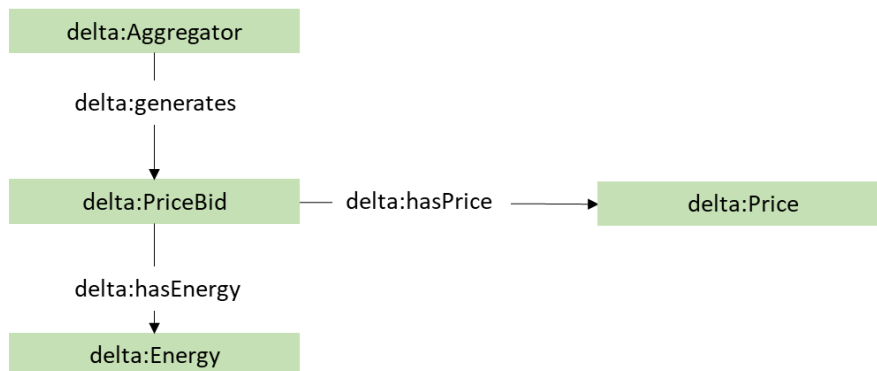


Figure 92 Ontology concepts related to Bids

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-2.

7.5.1.7 Rewards

This interface provides the rewards associated to each consumer, which were previously obtained during the gamification process. Figure 93 summarizes the ontology concepts related to this interface.

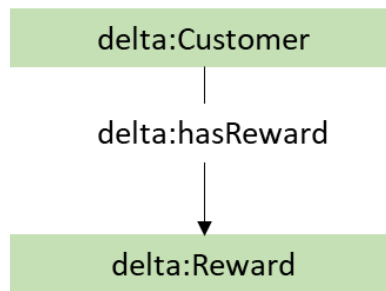


Figure 93 Ontology concepts related to the Rewards

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-16.

7.5.1.8 Energy price Profiling

This interface will provide other components with the Node Profiling related to energy price. This profiling will be used to calculate the price bids. Figure 94 Figure 46 summarizes the concepts in the ontology model that are related to this interface.

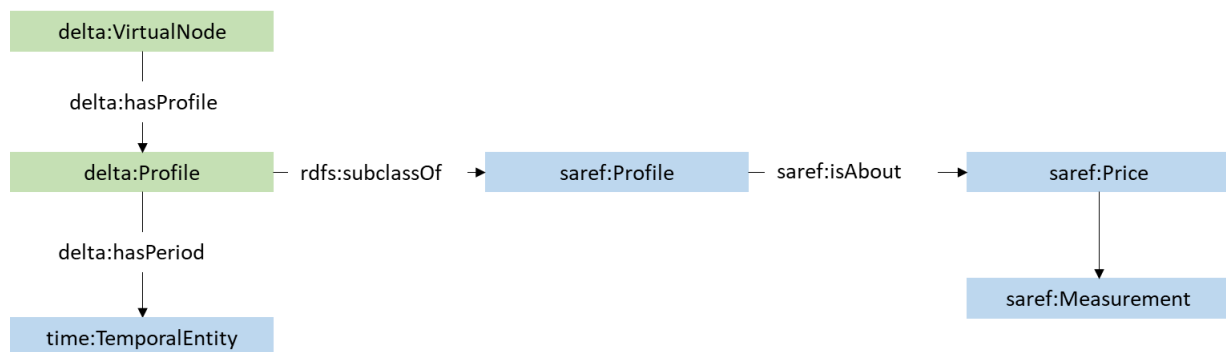


Figure 94 Ontology concepts related to Energy Price Profiling

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-7.

7.5.1.9 DVN Clusters

This interface provides the components with the clusters of each virtual node. Figure 95 summarizes the ontology concepts that are related to this interface.



Figure 95 Ontology concepts related to DVN Clusters

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-6.

7.5.1.10 Node Profiling

This interface will provide other components with the DELTA Node Profiling. This profiling will be used to assess the DVN and the devices. An example of possible profile could be related with the reliability of the device. Figure 96 summarizes the concepts in the ontology model that are related to this interface.

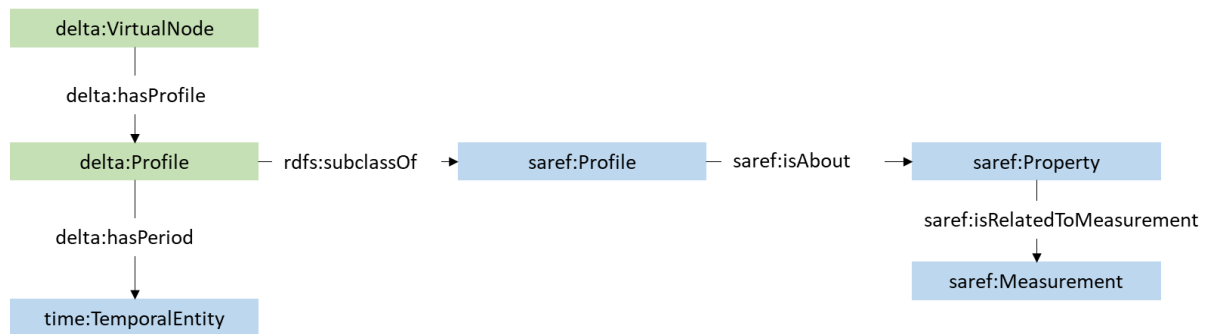


Figure 96 Ontology concepts related to Node Profiling

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-15.

7.5.1.11 Aggregated Profiling

This interface provides the profile related to an aggregator. This profile, as the DVN profile, is associated to a property measurement in a time interval. Figure 97 Figure 97 represents the ontology concepts related to this interface.

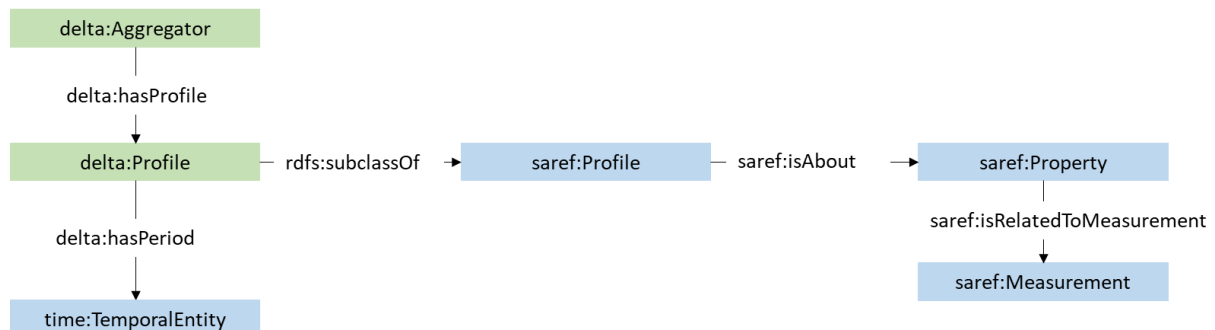


Figure 97 Ontology concepts related to the Aggregated Profiling

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-1.

7.5.2 Social Interaction and Cooperation

This component will enable the discussion and knowledge diffusion among end-users through a platform that will provide a large portfolio of useful information related to their participation in DR programs. This module will allow end-users interaction between them and the platform. It is also related to the other two components of the toolset, the Demand Response Visualization Kit and the Award-enabled Energy Behavioral Platform. This platform receives as input the Rewards generated from the Award-enabled Energy Behavioral Platform. This component has not a specific output interface for the DELTA platform. Its purpose is to send the data to the end-users' UIs.

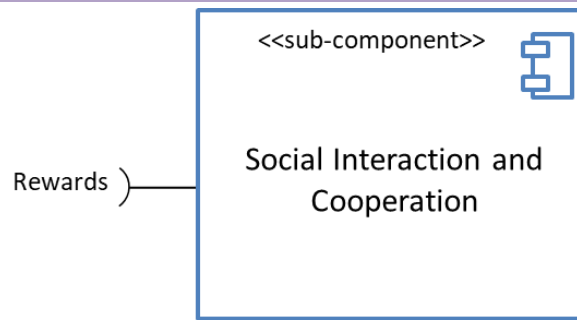


Figure 98 Interfaces for the Social Interaction and Cooperation

7.5.2.1 Rewards

This interface provides the rewards associated to each consumer, which were previously obtained during the gamification process. Figure 99 summarizes the ontology concepts related to this interface.

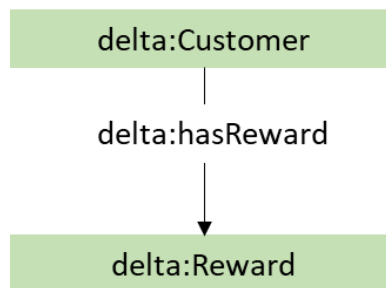


Figure 99 Ontology concepts related to the Rewards

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-16.

7.5.3 Award-enabled Energy Behavioral Platform

The aim of this component is to establish a reward point system that is going to be integrated into the smart contract framework. The main function of this module to reward customers for their participation in DR programs. This will be achieved through DR games based on actual DR scenarios. The main indicators, that are included in each smart contract and will be taken into account, are the points earned/kW of DR-enabled capacity. These points will be translated in realistic respective rewards. This process will improve the end-user engagement. Thus, this module has one input interface that is the: (1) DVN Clusters.

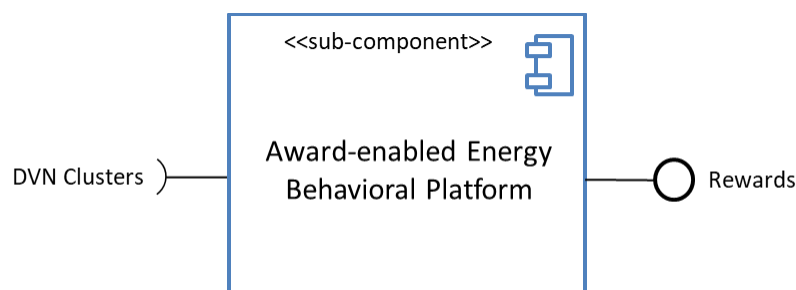


Figure 100 Interfaces for the Award-enabled Energy Behavioral Platform

7.5.3.1 DVN Clusters

This interface provides the components with the clusters of each virtual node. Figure 101 summarizes the ontology concepts that are related to this interface.



Figure 101 Ontology concepts related to DVN Clusters

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-6.

7.5.3.2 Rewards

This interface provides the rewards associated to each consumer, which were previously obtained during the gamification process. Figure 102 summarizes the ontology concepts related to this interface.

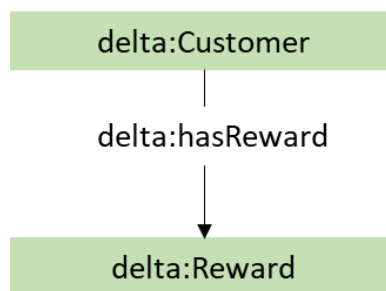


Figure 102 Ontology concepts related to the Rewards

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-16.

7.6 DELTA Cyber Security Services

This subsection describes the interfaces for each subcomponent related to the DELTA Cyber Security Services.

7.6.1 DELTA Blockchain

This component aims at storing the Transactions made between the main DELTA components, namely: DELTA Aggregator/Energy Retailer, Virtual DELTA Node, and DELTA Fog Enabled Agent.

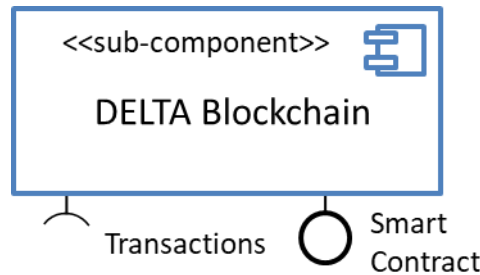


Figure 103 Interfaces for DELTA Blockchain

7.6.1.1 Smart contracts

This interface exchanges the smart contracts that are stored in the blockchain. Such contract includes the value of the expected DELTA property and the agreed payment. Figure 104 summarizes the ontological concepts related to this interface.

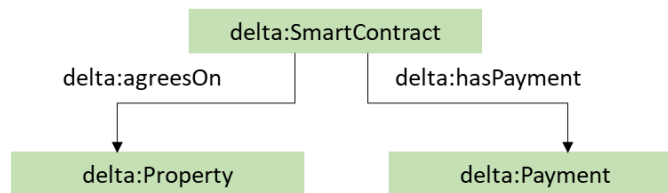


Figure 104 Ontology concepts related to Smart Contracts

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-17.

7.6.1.2 Transactions

This interface stores the transactions made by the Aggregator, the Virtual Node and the FEID. Figure 105 summarizes the ontological concepts related to this interface.

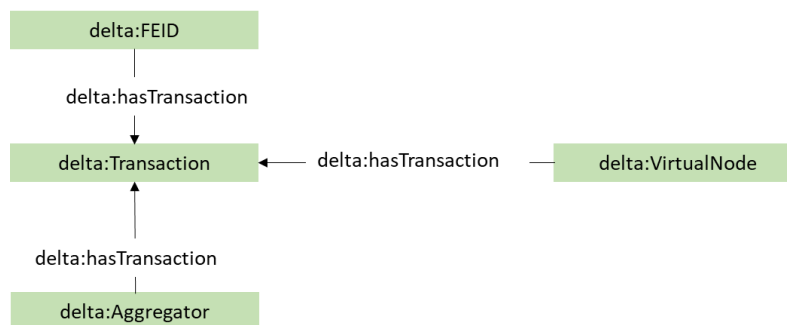
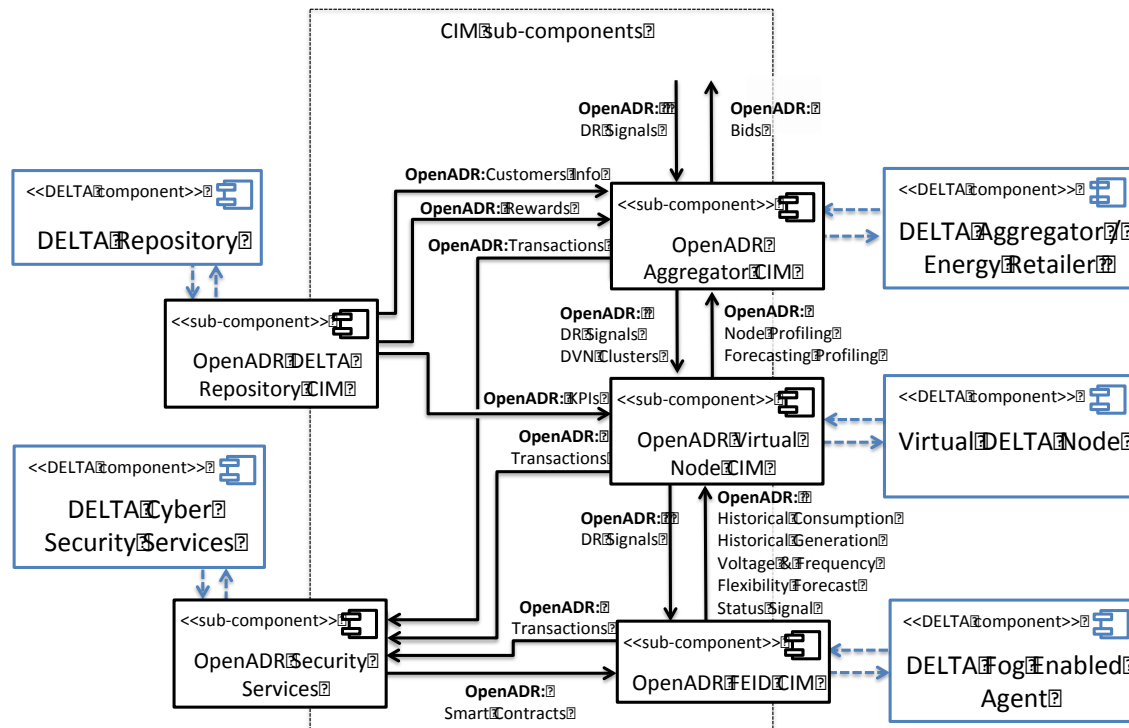


Figure 105 Ontology concepts related to Transactions

The JSON Schema that describes the structure of the data that need to be contained in each interface and an example of JSON-LD interface that exchange the flexibility forecast are shown in Annex 1.A-20.

8. Common Information Model

The Common Information Model (CIM) in the DELTA architecture is the component responsible of the data exchange among different main DELTA components, i.e., it allows a transparent data exchange between DELTA components. To achieve its goal, the CIM consist of several sub-components that assist each of the main DELTA components deployed. These sub-components send and receive data following the OpenADR standard, they are namey: OpenADR Aggregator CIM, OpenADR Virtual Node CIM, OpenADR FEID CIM, OpenADR DELTA Repository CIM, and the OpenADR Security Services CIM. These sub-components are Software artefacts through which data flows, as depicted in the figure below.



The data exchanged feeds specific sub-components within a main DELTA component, the CIM sub-components are in charge to exchange the data following the OpenADR specification, but in addition they distribute and deliver the data to the right sub-components, as we explain in the following sections. The following table exposes the project tasks in which the different software artefacts will be developed:

CIM sub-component	DELTA Task
OpenADR DELTA Repository CIM	T1.3 DELTA Ontology and Data Modelling Framework Definition
OpenADR FEID CIM	T3.2 DELTA Virtual Node Multi Agent System
OpenADR Virtual Node CIM	T3.2 DELTA Virtual Node Multi Agent System
OpenADR Aggregator CIM	T4.1 DELTA Aggregator Infrastructure, Design and Implementation
OpenADR Security Services CIM	T5.2 Log-oriented architecture design and Implementation of the DELTA blockchain

In following sections we introduce a short version of OpenADR and its requirements, then we aim at showing how we implemented the different OpenADR requirements in DELTA, following we show all the data exchange interactions expected to occur in the DELTA platform, and finally we provide an overview of the software requirements to implement the different CIM sub-components.

8.1 CIM and OpenADR

The DELTA platform counts on different components that exchange information following the OpenADR communication protocol and the data models described in this deliverable. The OpenADR is a standard meant to model demand response domains, which counts with two profiles that can be implemented. Table 1 shows the technological requirements that each of the OpenADR profiles entail.

Services and Functions to Support	Profile B	Profile A
EiEvent interface	Mandatory	Mandatory (limited implementation)
EiOpt interface	Mandatory	Not available
EiReport interface	Mandatory	Not available
EiRegisterParty interface	Mandatory	Not available
HTTP transport protocol	Mandatory	Mandatory
XMPP communication protocol	Mandatory	Optional
Data encryption	Mandatory	Mandatory
Exchanged data signed with XML signatures	Optional	Optional

Table 1 – OpenADR specification

The OpenADR requirements are namely: data interfaces such as EiEvent, EiOpt, EiReport, and EiRegisterParty, communication requirements such as HTTP and XMPP, and security such as data encryption and XML signatures. To have a detailed specification of each of the requirements refer to the OpenADR profile B specification document (O. Alliance, 2013), following we provide a short overview of every requirement:

- EiEvent interface: this interface aims at receiving and exposing information regarding potential signals that are involved in a demand response interaction.
- EiOpt interface: this interface aims at scheduling the communication between two components, for instance by reporting a time of un-availability of one of the components involved in the exchange.
- EiReport interface: this interface exposes *reports* that in OpenADR are namely historical data and real-time data gathered by physical devices.
- EiRegisterParty: this interface manages the components to which a certain component can communicate with.
- HTTP and XMPP: in the case of profile B OpenADR demands the use of both protocols, therefore the communication is hold by a secured Peer-to-Peer network.
- Data encryption: cyphers the information that is being exchanged, so even if such data is captured by a malicious third part it will not be readable.
- XML signatures: are meant to entrust that original information sent is correct and ensure the integrity of the information.

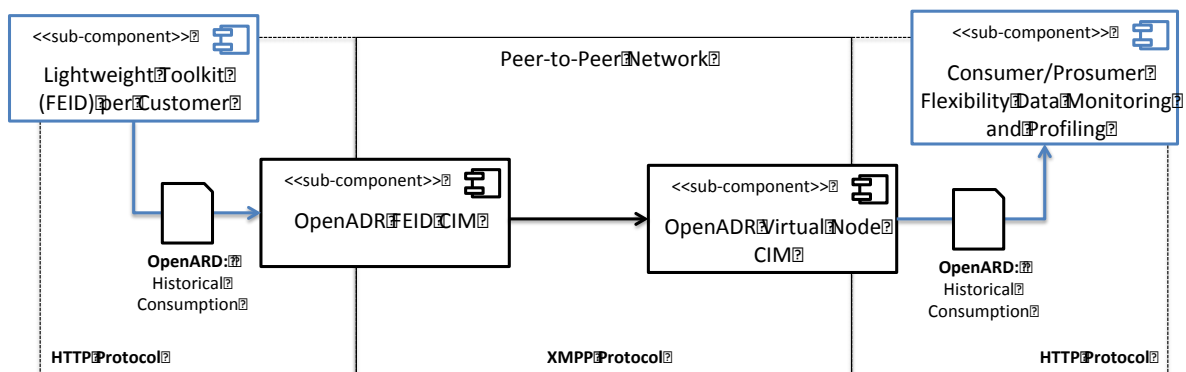
Unfortunately, OpenADR is not enough to model the DELTA domain since contains elements out of the OpenADR scope, for instance the profiles of devices or software components related to energy domain; such as aggregators or DSOs, smart contracts, market bids, or KPIs. Due to this limitation we have extended the range of the data model in OpenADR with the models presented in this deliverable. Notice that OpenADR is a standard that allows interoperability, which in our case is still valid since the entry point of DELTA for external agents (the ones with which potentially the platform needs to be interoperable with) follow the pure OpenADR interfaces, i.e., the DELTA component *DELTA Aggregator/Energy Retailer*.

To ensure the requirements of the OpenADR standard DELTA relies on the CIM, which is a software component that allows the exchange of data following the requirements of OpenADR and the

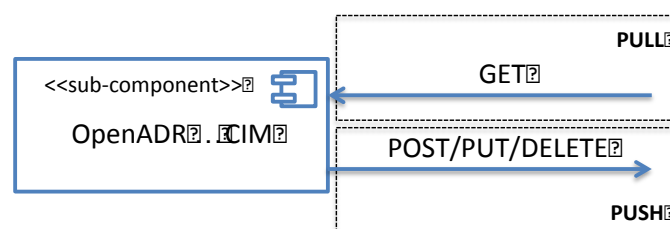
extension done in DELTA. The CIM relies on five main sub-components, i.e., the OpenADR Aggregator CIM, OpenADR Virtual Node CIM, OpenADR FEID CIM, OpenADR DELTA Repository CIM, and the OpenADR Security Services CIM. Each of these subcomponents assists one of the main DELTA components to communicate following the specifications of the next sections.

8.2 CIM communication protocols

The CIM sub-components deployed in DELTA must support the XMPP and HTTP protocols, due to this reason DELTA relies on a Peer-to-Peer network. As depicted in the figure below, anytime a sub-component in one DELTA component needs to send or receive information to other sub-component within a different DELTA component such communication is held through the P2P network thanks to the CIM sub-components.



The REST standard is used by the delta sub-components to request or send information. OpenADR specifies two different ways of working PULL and PUSH. The former is when one sub-component requests information to another, the later when one sub-component directly sends the information to another sub-component without a prior request of such data. The figure below shows how we map the different REST methods to the PULL/PUSH approach of OpenADR.



Notice that the PULL/PUSH approach demands to the components to have URLs to which data is requested or submitted. OpenADR specifies a URL pattern that follows the indications below:

`https://<hostname>(:port)/(prefix)/(oid)/OpenADR2/Simple/2.ob/<service>`

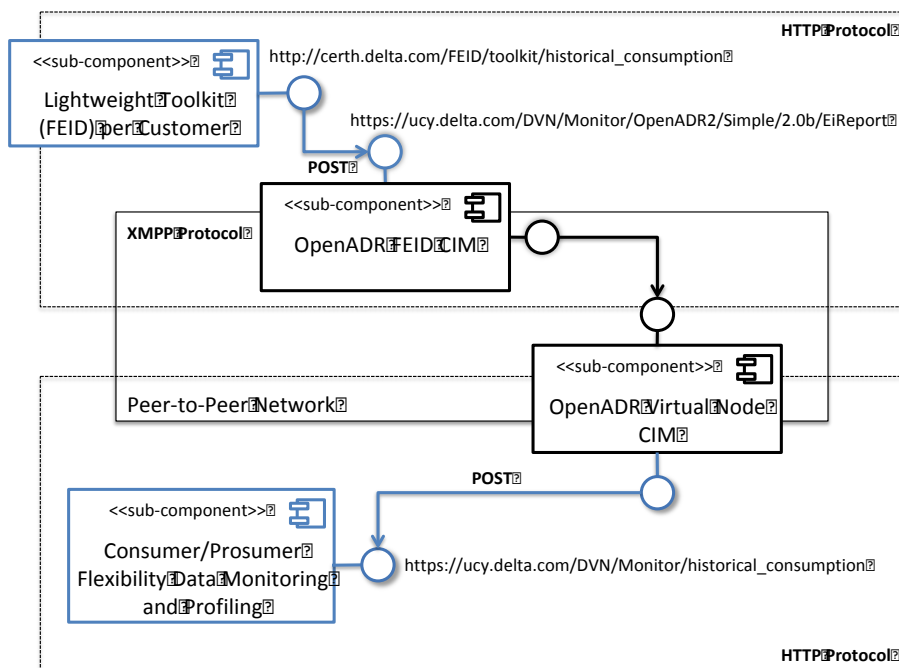
- Prefix: is an optional path to distinguish different OpenADR services that may be allocated in the same hostname.
- Oid: is the oid of the component that exposes the required data through the P2P network.
- Service: is the name of the interface implemented, i.e., EiEvent, EiReport, EiOpt, EiRegisterParty.

Notice that the DELTA sub-components that do not belong to the CIM are not required to implement OpenADR. Therefore the requirements that they must fulfil are the following ones: expose/receive data following the REST standard, follow the data models defined in this deliverable (that includes OpenADR data models), and expose data using the HTTP/HTTPS protocol and URLs that follow this pattern

`http://<hostname>(:port)/<owner>/<component>/<sub-component>/<service>`

- Owner: is the organization or entity that owns the component.
- Component: is the name of a DELTA main component, namely Aggregator, Virtual Node (DVN), FEID, Repository, or Security Services (Security).
- Sub-component: is the name of a sub-component that belongs to the main component. For instance the LoadForecaster in the Virtual Delta Node. Notice that spaces are not coded instead of replacing them for %20.
- Service: is the name of the interface that exposes the data, for instance for the LoadForecaster it would be the forecasted_values. Notice that spaces are coded with the char '_' instead of usual %20.

Considering these restrictions the figure below shows an example of how the flow of a data exchange is defined in DELTA using a PUSH approach.

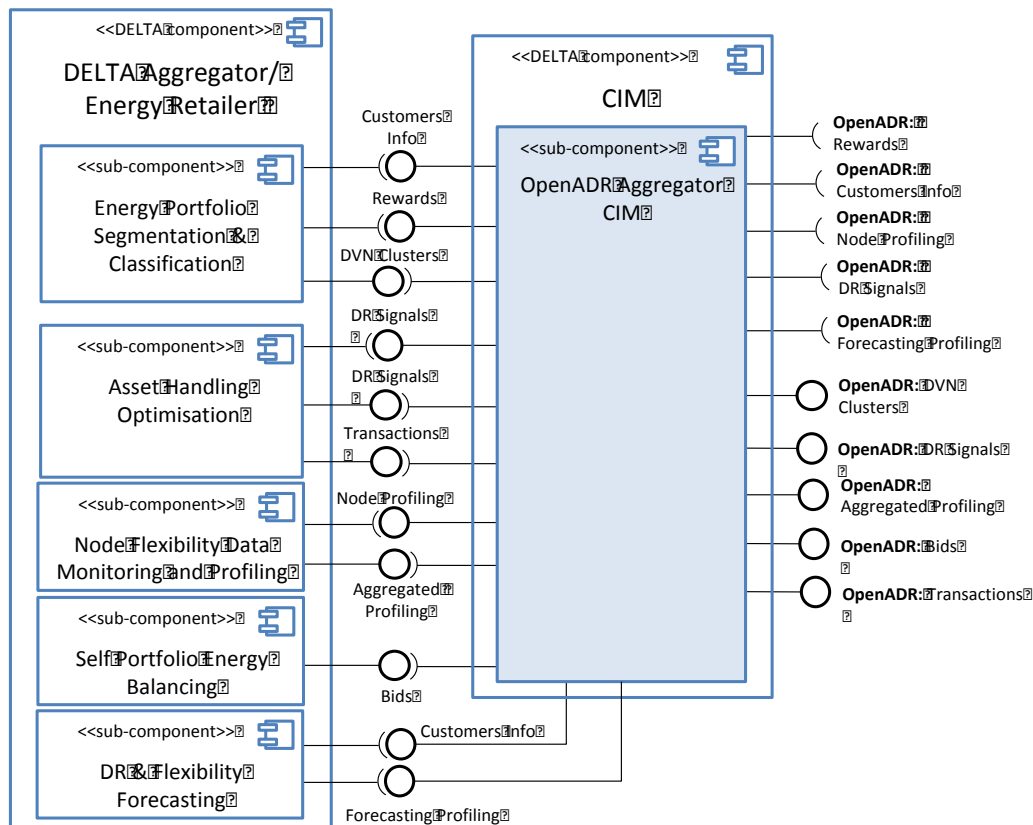


In the example figure it can be observed how the CIM sub-components allow the communication between two different sub-components that belong to different DELTA components. In addition, this communication is ensured to follow the OpenADR communication protocol specification. In the following subsections we provide an overview of the different CIM sub-components, their URLs, interfaces, and the data that is exchanged (which has been detailed in Deliverable D1.1).

8.3 OpenADR Aggregator CIM

The *OpenADR Aggregator CIM* allows the DELTA component *DELA Aggregator/Energy Retailer* to communicate on the one hand with potential DSOs or TSOs and on the other hand with the DELTA

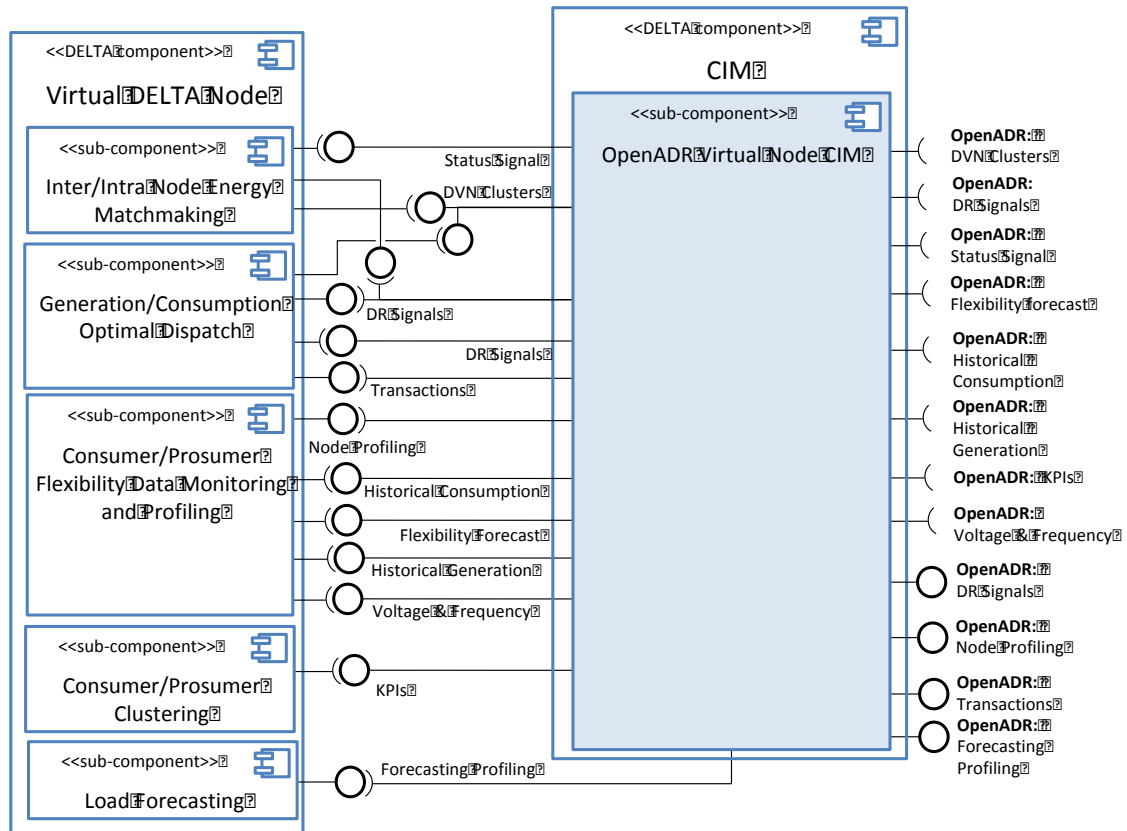
components *Virtual DELTA Node*. Following figure depict the data exchange between the different sub-components of the *DELA Aggregator/Energy Retailer* and the *OpenADR Aggregator CIM*.



Notice that the *OpenADR Aggregator CIM* internally routes the data to the right components, for instance when a demand response signal is received through the **OpenADR: DR Signals** interface this CIM sub-component automatically forwards the signal to the Asset Handling Optimisation. All the data interfaces shown in the previous figure has being defined in this deliverable. All the data interfaces related to non Peer-to-Peer communications are bounded to different sub-components of the *DELTA Aggregator / Energy Retailer* however, the OpenADR interfaces are available to any other CIM sub-component (or software component that implements this standard) to read or write information.

8.4 OpenADR Virtual Node CIM

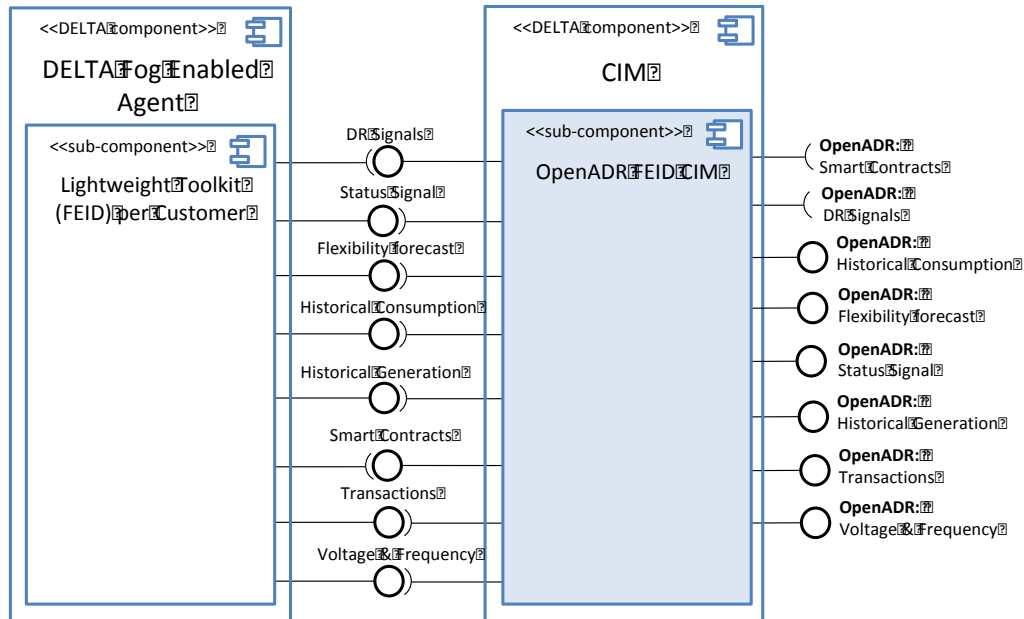
The *OpenADR Virtual Node CIM* allows the DELTA component *Virtual DELTA Node* to communicate with the *DELTA Fog Enabled Agent* and with the *DELTA Aggregator / Energy Retailer*, respectively. We present below a figure that depicts the data exchange between the different sub-components of the *Virtual DELTA Node* and the *OpenADR Virtual Node CIM*. The data interfaces shown in the figure have been explained during this deliverable.



The *OpenADR Virtual Node CIM* bounds all the non-OpenADR data interfaces (the ones outside the Peer-to-Peer network) with different Virtual DELTA Node sub-components, as shown in figure above. On the contrary the OpenADR data interfaces are available to any other CIM sub-component (of software component that implements OpenADR) to interact with.

8.5 OpenADR FEID CIM

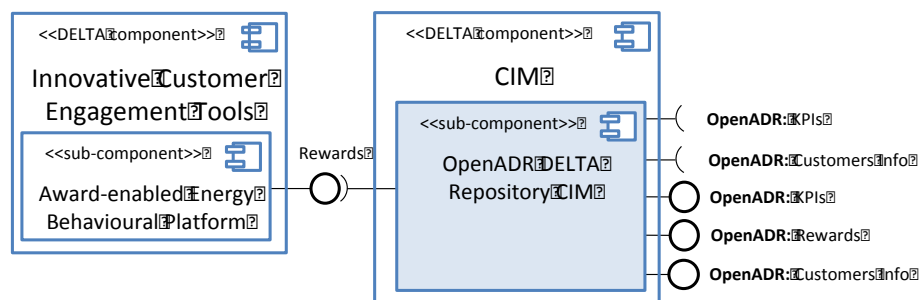
The *OpenADR FEID CIM* allows the *DELTA Fog Enabled Agent* to exchange data with the *Virtual DELTA Nodes*. Following we introduce a figure that shows how data can be exchanged with the sub-component *Lightweight Toolkit (FEID) per Customer* of the *DELTA Fog Enabled Agent*. The data interfaces shown in the figure have been explained during this deliverable.



Notice that the *OpenADR FEID CIM* namely forwards the demand response signal meant for the FEIDs, and on the other hand allows reading some of their features, e.g., Status Signal. Although the non-OpenADR data interfaces are all bounded to the *Lightweight Toolkit (FEID) per Customer* the OpenADR interfaces are available to any other CIM sub-component (of software component that implements OpenADR) to interact with.

8.6 OpenADR Repository CIM

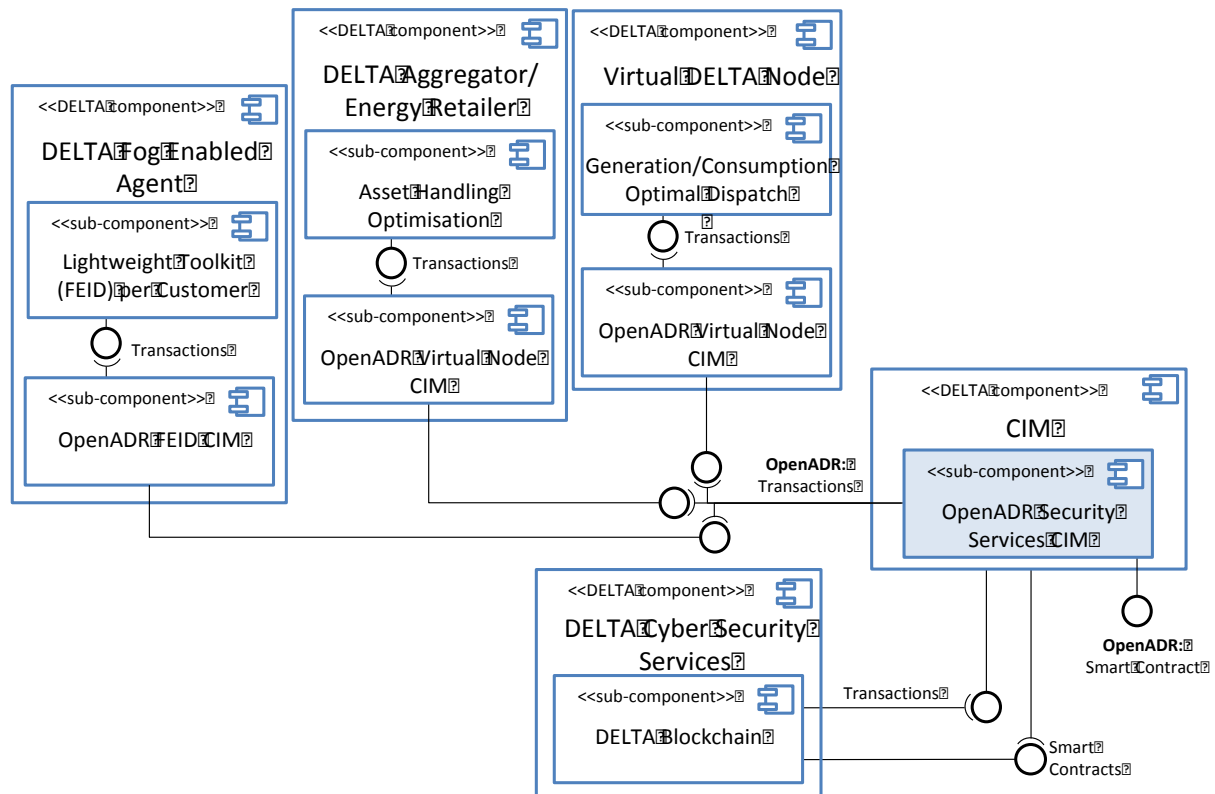
The *OpenADR Repository CIM* exposes data from a repository to other DELTA sub-components, which require such information in order to properly work. Below we present a figure showing the interactions of these sub-components with the *OpenADR DELTA Repository CIM*; every interface shown in the picture exposes data that has already been introduced in this deliverable.



Notice that the *DELTA Repository CIM* only has non-OpenADR data interfaces, and all of them are bounded to different sub-components, namely: *Award-enabled energy behavioural Platform*, *Energy Portfolio Segmentation & Classification*, and the *Consumer/Prosumer Clustering*. It should be remarked that the *KPIs* and *Customers Info* are provided by experts in the former case, and by a third-application in the later.

8.7 OpenADR Security Services CIM

The *OpenADR Security Services CIM* allows different DELTA components to send Smart Contracts to the DELTA sub-component in charge of storing them, i.e., the *DELTA Blockchain*. Following we introduce a figure that shows how the *Asset Handling Optimisation* and the *Generation/Consumption Optimal Dispatch* send *Smart Contracts*, and these are received by the DELTA Blockchain thanks to the *Security Services CIM*. All the data shown in the figure has been explained during this deliverable.



8.8 Developing and Deploying CIM sub-components

The CIM sub-components are services that follow the REST standard and the HTTP protocol to receive and send data to any DELTA sub-component that is not a CIM sub-component. On the other hand these sub-components follow the XMPP protocol to exchange data among them. The data expected to be received or sent must follow the specifications introduced in this deliverable, otherwise the correct working of the platform cannot be ensured.

The different CIM sub-components will be developed relaying on the SPRING technology to build the services, and the smack library to connect to the P2P network, which is built using the ejabber P2P server. The use of these libraries entails that all the code developed will be in Java 8.0; which will be stored in the version control list currently used by the project.

The different CIM sub-components must be deployed in the same infrastructure where the sub-components of a main DELTA component are, otherwise the correct communication will not be ensured. The deployment technology will be either a jar file with an embedded tomcat distributed with a starting script, or a docker image; in both cases this software will not have any problems with the OS where will be deployed. In addition all the different CIM sub-components will be provided with the necessary documentation to understand how to deploy them, and their JAVADOC in case a third-part would need to extend some piece of code.

9. Conclusions

In this document the ontology and the Common Information Model (CIM) have been detailed, along with a review of existing standards and ontologies related to Demand-Response domain. Additionally, the JSON-LD interfaces needed to exchange data between components in the DELTA platform have been specified.

This deliverable presents the first release of the DELTA ontology, which is focused on satisfying the requirements of the first release of the DELTA architecture. In future releases, new requirements will appear and be included in the DELTA ontology. Both the ontology and the artefacts produced during the ontology development are available online in the DELTA ontology portal.

Additionally, the first release of the CIM is also presented, i.e., the OpenADR DELTA Repository CIM; which implements the requirements specified in the OpenADR standard to exchange information between different DELTA sub-components and the DELTA Repository. As a conclusion of this deliverable, the CIM is one of the pillar software artefacts in the DELTA platform to exchange information by following the OpenADR data model already included in the DELTA ontology and the OpenADR communication protocols.

10. References

- García-Castro, R., Fernández-Izquierdo, A., Heinz, C., Kostelnik, P., Poveda-Villalón, M., & Serena, F. (2017). *D2.2 Detailed Specification of the Semantic Model*.
- Garijo, D. (2017). WIDOCO: a wizard for documenting ontologies. *International Semantic Web Conference*.
- Gruninger, M. a. (1995). Methodology for the design and evaluation of ontologies.
- O. Alliance. (2013). *OpenADR 2.0 Profile Specification B Profile*. Raportti.
- Poveda-Villalón, M. a.-F.-P. (2012). Validating ontologies with oops!
- Smith, J. (2016). inteGRIDy sample reference. *EU*, 1-2.
- Suárez-Figueroa, M. C.-P.-L. (2012). The NeOn methodology for ontology engineering. In *Ontology engineering in a networked world* (pp. 9-34).

Annex 1

In this annex all the JSON Schemas related to the interfaces described in the document are defined, together with a JSON-LD example of how to use such schema.

A-1. Aggregated profiling

The following JSON Schema describes the structure of the data that need to be contained in the interface.

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "@context": {
      "$id": "#/properties/@context",
      "type": "object",
      "properties": {
        "core": {
          "$id": "#/properties/@context/properties/core",
          "type": "string"
        },
        "saref": {
          "$id": "#/properties/@context/properties/saref",
          "type": "string"
        },
        "xsd": {
          "$id": "#/properties/@context/properties/xsd",
          "type": "string"
        },
        "VirtualNode": {
          "$id": "#/properties/@context/properties/VirtualNode",
          "type": "object",
          "properties": {
            "@id": {
              "$id":
                "#/properties/@context/properties/VirtualNode/properties/@id",
              "type": "string"
            }
          }
        }
      },
    },
    "hasMeasurement": {
      "$id": "#/properties/@context/properties/hasMeasurement",
      "type": "object",
      "properties": {
        "@id": {
          "$id":
            "#/properties/@context/properties/hasMeasurement/properties/@id",
          "type": "string"
        }
      }
    },
    "hasValue": {
      "$id": "#/properties/@context/properties/hasValue",
      "type": "object",
      "properties": {
        "@id": {
          "$id":
            "#/properties/@context/properties/hasValue/properties/@id",
          "type": "string"
        }
      }
    }
  }
}
```

```
    }
  }
},
"@graph": {
  "$id": "#/properties/@graph",
  "type": "array",
  "items": {
    "$id": "#/properties/@graph/items",
    "type": "object",
    "properties": {
      "@id": {
        "$id": "#/properties/@graph/items/properties/@id",
        "type": "string"
      },
      "@type": {
        "$id": "#/properties/@graph/items/properties/@type",
        "type": "string"
      },
      "hasMeasurement": {
        "$id": "#/properties/@graph/items/properties/hasMeasurement",
        "type": "object",
        "properties": {
          "@id": {
            "$id":
"#/properties/@graph/items/properties/hasMeasurement/properties/@id",
            "type": "string"
          }
        }
      }
    }
  }
}
```

Based on the JSON Schema and the ontology concepts, the following code depicts an example of JSON-LD interface..

```
{
  "@context": {
    "core": "http://delta.linkeddata.es/def/core#",
    "saref": "https://w3id.org/saref#",
    "om": "http://www.wurvoc.org/vocabularies/om-1.8#",
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "TemporalEntity": {
      "@id": "time:TemporalEntity"
    },
    "Aggregator": {
      "@id": "core: Aggregator"
    },
    "hasDuration": {
      "@id": "core:hasDuration"
    },
    "hasProfile": {
      "@id": "core:hasProfile"
    },
    "Energy": {
      "@id": "core:Energy"
    },
    "hasBeginning": {
      "@id": "core:hasBeginning"
    }
  },
}
```

```
"hasEnd": {
  "@id": "core:hasEnd"
},
"isAbout": {
  "@id": "saref:isAbout"
}
},
"@graph": [
  {
    "@id": "core:Aggregator01",
    "@type": "Aggregator",
    "hasProfile": {
      "@id": "core:Profile1"
    }
  },
  {
    "@id": "core:Profile1",
    "@type": "Profile",
    "hasDuration": {
      "@id": "core:TimeInterval1"
    },
    "isAbout": {
      "@id": "core:Energy01"
    }
  },
  {
    "@id": "core: Energy01",
    "@type": "Energy",
    "hasMeasurement": {
      "@id": "core:Measurement1"
    }
  },
  {
    "@id": "core: Measurement1",
    "@type": "Load",
    "hasValue": {
      "@type": "xsd:float",
      "@value": "78.2"
    },
    "saref:isMeasuredIn": {
      "@id": "om:watt"
    }
  },
  {
    "@id": "core:TimeInterval1",
    "@type": "TemporalEntity",
    "hasBeginning": {
      "@id": "Instant01"
    },
    "hasEnd": {
      "@id": "Instant02"
    }
  },
  {
    "@id": "core:Instant01",
    "@type": "Instant",
    "inXSDDate": {
      "@type": "xsd:date",
      "@value": "2019-01-05"
    }
  }
],
```

```
{
  "@id": "core:Instant02",
  "@type": "Instant",
  "inXSDDate": {
    "@type": "xsd:date",
    "@value": "2019-01-06"
  }
}
]
```

A-2. Bids

The following JSON Schema describes the structure of the data that need to be contained in the interface.

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "http://example.com/root.json",
  "type": "object",
  "properties": {
    "@context": {
      "$id": "#/properties/@context",
      "type": "object",
      "properties": {
        "core": {
          "$id": "#/properties/@context/properties/core",
          "type": "string"
        },
        "saref": {
          "$id": "#/properties/@context/properties/saref",
          "type": "string"
        },
        "xsd": {
          "$id": "#/properties/@context/properties/xsd",
          "type": "string"
        },
        "TemporalEntity": {
          "$id": "#/properties/@context/properties/TemporalEntity",
          "type": "object",
          "properties": {
            "@id": {
              "$id":
                "#/properties/@context/properties/TemporalEntity/properties/@id",
              "type": "string"
            }
          }
        }
      },
    },
    "VirtualNode": {
      "$id": "#/properties/@context/properties/VirtualNode",
      "type": "object",
      "properties": {
        "@id": {
          "$id":
            "#/properties/@context/properties/VirtualNode/properties/@id",
          "type": "string"
        }
      }
    }
  }
}
```



```
    },
    "hasDuration": {
      "$id": "#/properties/@context/properties/hasDuration",
      "type": "object",
      "properties": {
        "@id": {
          "$id":
"#/properties/@context/properties/hasDuration/properties/@id",
          "type": "string"
        }
      }
    },
    "hasProfile": {
      "$id": "#/properties/@context/properties/hasProfile",
      "type": "object",
      "properties": {
        "@id": {
          "$id":
"#/properties/@context/properties/hasProfile/properties/@id",
          "type": "string"
        }
      }
    },
    "hasPrice": {
      "$id": "#/properties/@context/properties/hasPrice",
      "type": "object",
      "properties": {
        "@id": {
          "$id":
"#/properties/@context/properties/hasPrice/properties/@id",
          "type": "string"
        }
      }
    },
    "hasEnergy": {
      "$id": "#/properties/@context/properties/hasEnergy",
      "type": "object",
      "properties": {
        "@id": {
          "$id":
"#/properties/@context/properties/hasEnergy/properties/@id",
          "type": "string"
        }
      }
    },
    "isAbout": {
      "$id": "#/properties/@context/properties/isAbout",
      "type": "object",
      "properties": {
        "@id": {
          "$id":
"#/properties/@context/properties/isAbout/properties/@id",
          "type": "string"
        }
      }
    },
    "hasMeasurement": {
      "$id": "#/properties/@context/properties/hasMeasurement",
      "type": "object",
      "properties": {
        "@id": {
```

```
        "$id":
"#/properties/@context/properties/hasMeasurement/properties/@id",
        "type": "string"
    }
}
},
"Measurement": {
    "$id": "#/properties/@context/properties/Measurement",
    "type": "object",
    "properties": {
        "@id": {
            "$id":
"#/properties/@context/properties/Measurement/properties/@id",
            "type": "string"
        }
    }
},
"isMeasuredIn": {
    "$id": "#/properties/@context/properties/isMeasuredIn",
    "type": "object",
    "properties": {
        "@id": {
            "$id":
"#/properties/@context/properties/isMeasuredIn/properties/@id",
            "type": "string"
        }
    }
},
"hasValue": {
    "$id": "#/properties/@context/properties/hasValue",
    "type": "object",
    "properties": {
        "@id": {
            "$id":
"#/properties/@context/properties/hasValue/properties/@id",
            "type": "string"
        }
    }
},
"Profile": {
    "$id": "#/properties/@context/properties/Profile",
    "type": "object",
    "properties": {
        "@id": {
            "$id":
"#/properties/@context/properties/Profile/properties/@id",
            "type": "string"
        }
    }
}
},
"@graph": {
    "$id": "#/properties/@graph",
    "type": "array",
    "items": {
        "$id": "#/properties/@graph/items",
        "type": "object",
        "properties": {
            "@id": {
                "$id": "#/properties/@graph/items/properties/@id",
```

```
        "type": "string"
      },
      "@type": {
        "$id": "#/properties/@graph/items/properties/@type",
        "type": "string"
      },
      "generates": {
        "$id": "#/properties/@graph/items/properties/generates",
        "type": "object",
        "properties": {
          "@id": {
            "$id":
"#/properties/@graph/items/properties/generates/properties/@id",
            "type": "string"
          }
        }
      }
    }
  }
}
}
```

Based on the JSON Schema and the ontology concepts, the following code depicts an example of JSON-LD interface that exchange the bids.

```
{
  "@context": {
    "core": "http://delta.linkeddata.es/def/core#",
    "saref": "https://w3id.org/saref#",
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "TemporalEntity": {
      "@id": "time:TemporalEntity"
    },
    "VirtualNode": {
      "@id": "core:VirtualNode"
    },
    "hasDuration": {
      "@id": "core:hasDuration"
    },
    "hasProfile": {
      "@id": "core:hasProfile"
    },
    "hasPrice": {
      "@id": "core:hasPrice"
    },
    "hasEnergy": {
      "@id": "core:hasEnergy"
    },
    "isAbout": {
      "@id": "saref:isAbout"
    },
    "hasMeasurement": {
      "@id": "saref:hasMeasurement"
    },
    "Measurement": {
      "@id": "saref:Measurement"
    },
    "isMeasuredIn": {
      "@id": "saref:isMeasuredIn"
    },
    "hasValue": {
      "@id": "saref:hasValue"
    }
  }
}
```

```
    },
    "Profile": {
      "@id": "core:Profile"
    }
  },
  "@graph": [
    {
      "@id": "core:Aggregator01",
      "@type": "Aggregator",
      "generates": {
        "@id": "core:PriceBid01"
      }
    },
    {
      "@id": "core:PriceBid01",
      "@type": "PriceBid",
      "hasEnergy": {
        "@id": "core:Price01"
      },
      "hasPrice": {
        "@id": "core:Energy01"
      }
    },
    {
      "@id": "core: Energy01",
      "@type": "Energy",
      "hasMeasurement": {
        "@id": "core:Measurement1"
      }
    },
    {
      "@id": "core: Measurement1",
      "@type": "Measurement",
      "hasValue": {
        "@type": "xsd:float",
        "@value": "26.2"
      },
      "saref:isMeasuredIn": {
        "@id": "om:watt"
      }
    },
    {
      "@id": "core:Price01",
      "@type": "Price",
      "hasMeasurement": {
        "@id": "core:Measurement1"
      }
    },
    {
      "@id": "core: Measurement2",
      "@type": "Measurement",
      "hasValue": {
        "@type": "xsd:float",
        "@value": "268.3"
      },
      "isMeasuredIn": {
        "@id": "om:euro"
      }
    }
  ]
}
```

A-3. Comfort settings

The following JSON Schema describes the structure of the data that need to be contained in the interface.

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "@context": {
      "$id": "#/properties/@context",
      "type": "object",
      "properties": {
        "core": {
          "$id": "#/properties/@context/properties/core",
          "type": "string"
        },
        "saref": {
          "$id": "#/properties/@context/properties/saref",
          "type": "string"
        },
        "xsd": {
          "$id": "#/properties/@context/properties/xsd",
          "type": "string"
        },
        "FEID": {
          "$id": "#/properties/@context/properties/FEID",
          "type": "object",
          "properties": {
            "@id": {
              "$id":
                "#/properties/@context/properties/FEID/properties/@id",
              "type": "string"
            }
          }
        }
      },
    },
    "ComfortSetting": {
      "$id": "#/properties/@context/properties/ComfortSetting",
      "type": "object",
      "properties": {
        "@id": {
          "$id":
            "#/properties/@context/properties/ComfortSetting/properties/@id",
          "type": "string"
        }
      }
    },
    "hasComfortSetting": {
      "$id": "#/properties/@context/properties/hasComfortSetting",
      "type": "object",
      "properties": {
        "@id": {
          "$id":
            "#/properties/@context/properties/hasComfortSetting/properties/@id",
          "type": "string"
        }
      }
    }
  }
},
```

```
"@graph": {
  "$id": "#/properties/@graph",
  "type": "array",
  "items": {
    "$id": "#/properties/@graph/items",
    "type": "object",
    "properties": {
      "@id": {
        "$id": "#/properties/@graph/items/properties/@id",
        "type": "string"
      },
      "@type": {
        "$id": "#/properties/@graph/items/properties/@type",
        "type": "string"
      },
      "hasComfortSettings": {
        "$id":
"#/properties/@graph/items/properties/hasComfortSettings",
        "type": "object",
        "properties": {
          "@id": {
            "$id":
"#/properties/@graph/items/properties/hasComfortSettings/properties/@id",
            "type": "string"
          }
        }
      }
    }
  }
}
```

Based on the JSON Schema and the ontology concepts, the following code depicts an example of JSON-LD interface.

```
{
  "@context": {
    "core": "http://delta.linkeddata.es/def/core#",
    "saref": "https://w3id.org/saref#",
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "FEID": {
      "@id": "core:FEID"
    },
    "ComfortSetting": {
      "@id": "core:ComfortSetting"
    },
    "hasComfortSetting": {
      "@id": "core:hasComfortSetting"
    }
  },
  "@graph": [
    {
      "@id": "core:FEID01",
      "@type": "FEID",
      "hasComfortSettings": {
        "@id": "core:ComfortSetting01"
      }
    },
    {
      "@id": "core:ComfortSetting01",
      "@type": "ComfortSetting"
    }
  ]
}
```

```
]
}
```

A-4. Customer information

The following JSON Schema describes the structure of the data that need to be contained in the interface.

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "@context": {
      "$id": "#/properties/@context",
      "type": "object",
      "properties": {
        "core": {
          "$id": "#/properties/@context/properties/core",
          "type": "string"
        },
        "xsd": {
          "$id": "#/properties/@context/properties/xsd",
          "type": "string"
        },
        "Customer": {
          "$id": "#/properties/@context/properties/Customer",
          "type": "object",
          "properties": {
            "@id": {
              "$id":
                "#/properties/@context/properties/Customer/properties/@id",
              "type": "string"
            }
          }
        },
        "Device": {
          "$id": "#/properties/@context/properties/Device",
          "type": "object",
          "properties": {
            "@id": {
              "$id":
                "#/properties/@context/properties/Device/properties/@id",
              "type": "string"
            }
          }
        },
        "owns": {
          "$id": "#/properties/@context/properties/owns",
          "type": "object",
          "properties": {
            "@id": {
              "$id":
                "#/properties/@context/properties/owns/properties/@id",
              "type": "string"
            }
          }
        },
        "@graph": {
          "$id": "#/properties/@graph",
```

```
"type": "array",
"items": {
  "$id": "#/properties/@graph/items",
  "type": "object",
  "properties": {
    "@id": {
      "$id": "#/properties/@graph/items/properties/@id",
      "type": "string"
    },
    "@type": {
      "$id": "#/properties/@graph/items/properties/@type",
      "type": "string"
    },
    "owns": {
      "$id": "#/properties/@graph/items/properties/owns ",
      "type": "object",
      "properties": {
        "@id": {
          "$id": "#/properties/@graph/items/properties/owns
/properties/@id",
          "type": "string"
        }
      }
    }
  }
}
}
```

Based on the JSON Schema and the ontology concepts, the following code depicts an example of JSON-LD interface..

```
{
  "@context": {
    "core": "http://delta.linkeddata.es/def/core#",
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "Customer": {
      "@id": "core:Customer"
    },
    "Device": {
      "@id": "core:Device"
    },
    "owns": {
      "@id": "core:owns"
    }
  },
  "@graph": [
    {
      "@id": "core:Customer01",
      "@type": "Customer",
      "owns": {
        "@id": "core:Device1"
      }
    },
    {
      "@id": "core:Device1",
      "@type": "core:Device"
    }
  ]
}
```


A-5. DR signals

The following JSON Schema describes the structure of the data that need to be contained in the interface.

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "@context": {
      "$id": "#/properties/@context",
      "type": "object",
      "properties": {
        "core": {
          "$id": "#/properties/@context/properties/core",
          "type": "string"
        },
        "saref": {
          "$id": "#/properties/@context/properties/saref",
          "type": "string"
        },
        "xsd": {
          "$id": "#/properties/@context/properties/xsd",
          "type": "string"
        },
        "VirtualNode": {
          "$id": "#/properties/@context/properties/VirtualNode",
          "type": "object",
          "properties": {
            "@id": {
              "$id":
                "#/properties/@context/properties/VirtualNode/properties/@id",
              "type": "string"
            }
          }
        },
        "Price": {
          "$id": "#/properties/@context/properties/Price",
          "type": "object",
          "properties": {
            "@id": {
              "$id":
                "#/properties/@context/properties/Price/properties/@id",
              "type": "string"
            }
          }
        },
        "DRSignal": {
          "$id": "#/properties/@context/properties/DRSignal",
          "type": "object",
          "properties": {
            "@id": {
              "$id":
                "#/properties/@context/properties/DRSignal/properties/@id",
              "type": "string"
            }
          }
        },
        "isMeasuredIn": {
          "$id": "#/properties/@context/properties/isMeasuredIn",
          "type": "object",
          "properties": {
```

```
        "@id": {
          "$id":
"#/properties/@context/properties/isMeasuredIn/properties/@id",
          "type": "string"
        }
      }
    },
    "hasValue": {
      "$id": "#/properties/@context/properties/hasValue",
      "type": "object",
      "properties": {
        "@id": {
          "$id":
"#/properties/@context/properties/hasValue/properties/@id",
          "type": "string"
        }
      }
    },
    "Measurement": {
      "$id": "#/properties/@context/properties/Measurement",
      "type": "object",
      "properties": {
        "@id": {
          "$id":
"#/properties/@context/properties/Measurement/properties/@id",
          "type": "string"
        }
      }
    }
  },
  "@graph": {
    "$id": "#/properties/@graph",
    "type": "array",
    "items": {
      "$id": "#/properties/@graph/items",
      "type": "object",
      "properties": {
        "@id": {
          "$id": "#/properties/@graph/items/properties/@id",
          "type": "string"
        },
        "@type": {
          "$id": "#/properties/@graph/items/properties/@type",
          "type": "string"
        },
        "send": {
          "$id": "#/properties/@graph/items/properties/send",
          "type": "object",
          "properties": {
            "@id": {
              "$id":
"#/properties/@graph/items/properties/send/properties/@id",
              "type": "string"
            }
          }
        }
      }
    }
  }
}
```

}

Based on the JSON Schema and the ontology concepts, the following code depicts an example of JSON-LD interface.

```
{
  "@context": {
    "core": "http://delta.linkeddata.es/def/core#",
    "saref": "https://w3id.org/saref#",
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "VirtualNode": {
      "@id": "core:VirtualNode"
    },
    "Price": {
      "@id": "core:Price"
    },
    "DRSignal": {
      "@id": "core:DRSignal"
    },
    "isMeasuredIn": {
      "@id": "saref:isMeasuredIn"
    },
    "hasValue": {
      "@id": "saref:hasValue"
    },
    "Measurement": {
      "@id": "core:Profile"
    }
  },
  "@graph": [
    {
      "@id": "core:VirtualNode01",
      "@type": "VirtualNode",
      "send": {
        "@id": "core:EnergyPrice01"
      }
    },
    {
      "@id": "core: EnergyPrice01",
      "@type": "DRSignal",
      "isRelatedToProperty": {
        "@id": "core:Price01"
      },
      "isValidDuringPeriod": {
        "@id": "core:TimeInterval01"
      }
    },
    {
      "@id": "core:Price01",
      "@type": "Price",
      "hasMeasurement": {
        "@id": "core:Measurement1"
      }
    },
    {
      "@id": "core: Measurement1",
      "@type": "Measurement",
      "hasValue": {
        "@type": "xsd:float",
        "@value": "15.2"
      },
      "saref:isMeasuredIn": {
        "@id": "om:euro"
      }
    }
  ]
}
```

```
    }
  },
  {
    "@id": "core:TimeInterval",
    "@type": "TemporalEntity",
    "hasBeginning": {
      "@id": "Instant01"
    },
    "hasEnd": {
      "@id": "Instant02"
    }
  },
  {
    "@id": "core:Instant01",
    "@type": "Instant",
    "inXSDDate": {
      "@type": "xsd:date",
      "@value": "2019-04-11"
    }
  },
  {
    "@id": "core:Instant02",
    "@type": "Instant",
    "inXSDDate": {
      "@type": "xsd:date",
      "@value": "2019-04-12"
    }
  }
]
}
```

A-6. DVN Clusters

The following JSON Schema describes the structure of the data that need to be contained in the interface.

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "@context": {
      "$id": "#/properties/@context",
      "type": "object",
      "properties": {
        "core": {
          "$id": "#/properties/@context/properties/core",
          "type": "string"
        },
        "saref": {
          "$id": "#/properties/@context/properties/saref",
          "type": "string"
        },
        "xsd": {
          "$id": "#/properties/@context/properties/xsd",
          "type": "string"
        },
        "VirtualNode": {
          "$id": "#/properties/@context/properties/VirtualNode",
          "type": "object",
          "properties": {
            "@id": {
```

```
        "$id":
"#/properties/@context/properties/VirtualNode/properties/@id",
        "type": "string"
    }
}
},
"Transaction": {
    "$id": "#/properties/@context/properties/Transaction",
    "type": "object",
    "properties": {
        "@id": {
            "$id":
"#/properties/@context/properties/Transaction/properties/@id",
            "type": "string"
        }
    }
}
}
},
"@graph": {
    "$id": "#/properties/@graph",
    "type": "array",
    "items": {
        "$id": "#/properties/@graph/items",
        "type": "object",
        "properties": {
            "@id": {
                "$id": "#/properties/@graph/items/properties/@id",
                "type": "string"
            },
            "@type": {
                "$id": "#/properties/@graph/items/properties/@type",
                "type": "string"
            }
        }
    }
}
}
}
```

Based on the JSON Schema and the ontology concepts, the following code depicts an example of JSON-LD interface.

```
{
  "@context": {
    "core": "http://delta.linkeddata.es/def/core#",
    "dc": "http://purl.org/dc/elements/1.1/",
    "dcterms": "http://purl.org/dc/terms/",
    "owl": "http://www.w3.org/2002/07/owl#",
    "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
    "terms": "http://purl.org/dc/terms/",
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "Cluster": {
      "@id": "core:Cluster"
    },
    "hasCluster": {
      "@id": "core:hasCluster"
    },
    "VirtualNode": {
      "@id": "core:VirtualNode"
    }
  }
}
```

```
{,
"@graph": [
  {
    "@id": "core:VirtualNode01",
    "@type": "core:VirtualNode",
    "core:hasCluster": {
      "@id": "core:Cluster01"
    }
  },
  {
    "@id": "core:Cluster01",
    "@type": "core:Cluster",
    "hasAggregatedProperty": {
      "@id": "core:Flexibility01"
    }
  },
  {
    "@id": "core:Flexibility01",
    "@type": "core:Flexibility",
    "isRelatedToMeasurement": {
      "@id": "core:Measurement01"
    }
  },
  {
    "@id": "core: Measurement1",
    "@type": " Measurement ",
    "hasValue": {
      "@type": "xsd:float",
      "@value": "136.1"
    }
  }
]
}
```

A-7. Energy Price profiling

The following JSON Schema describes the structure of the data that need to be contained in the interface.

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "@context": {
      "$id": "#/properties/@context",
      "type": "object",
      "properties": {
        "core": {
          "$id": "#/properties/@context/properties/core",
          "type": "string"
        },
        "saref": {
          "$id": "#/properties/@context/properties/saref",
          "type": "string"
        },
        "xsd": {
          "$id": "#/properties/@context/properties/xsd",
          "type": "string"
        },
        "Price": {
          "$id": "#/properties/@context/properties/Price",
          "type": "object",

```

```
    "properties": {
      "@id": {
        "$id":
"#/properties/@context/properties/Price/properties/@id",
        "type": "string"
      }
    }
  },
  "VirtualNode": {
    "$id": "#/properties/@context/properties/VirtualNode",
    "type": "object",
    "properties": {
      "@id": {
        "$id":
"#/properties/@context/properties/VirtualNode/properties/@id",
        "type": "string"
      }
    }
  },
  "hasDuration": {
    "$id": "#/properties/@context/properties/hasDuration",
    "type": "object",
    "properties": {
      "@id": {
        "$id":
"#/properties/@context/properties/hasDuration/properties/@id",
        "type": "string"
      }
    }
  },
  "hasProfile": {
    "$id": "#/properties/@context/properties/hasProfile",
    "type": "object",
    "properties": {
      "@id": {
        "$id":
"#/properties/@context/properties/hasProfile/properties/@id",
        "type": "string"
      }
    }
  },
  "TemporalEntity": {
    "$id": "#/properties/@context/properties/TemporalEntity",
    "type": "object",
    "properties": {
      "@id": {
        "$id":
"#/properties/@context/properties/TemporalEntity/properties/@id",
        "type": "string"
      }
    }
  },
  "hasBeginning": {
    "$id": "#/properties/@context/properties/hasBeginning",
    "type": "object",
    "properties": {
      "@id": {
        "$id":
"#/properties/@context/properties/hasBeginning/properties/@id",
        "type": "string"
      }
    }
  }
```

```
    }
  },
  "hasEnd": {
    "$id": "#/properties/@context/properties/hasEnd",
    "type": "object",
    "properties": {
      "@id": {
        "$id":
"#/properties/@context/properties/hasEnd/properties/@id",
        "type": "string"
      }
    }
  },
  "isAbout": {
    "$id": "#/properties/@context/properties/isAbout",
    "type": "object",
    "properties": {
      "@id": {
        "$id":
"#/properties/@context/properties/isAbout/properties/@id",
        "type": "string"
      }
    }
  },
  "hasMeasurement": {
    "$id": "#/properties/@context/properties/hasMeasurement",
    "type": "object",
    "properties": {
      "@id": {
        "$id":
"#/properties/@context/properties/hasMeasurement/properties/@id",
        "type": "string"
      }
    }
  },
  "isMeasuredIn": {
    "$id": "#/properties/@context/properties/isMeasuredIn",
    "type": "object",
    "properties": {
      "@id": {
        "$id":
"#/properties/@context/properties/isMeasuredIn/properties/@id",
        "type": "string"
      }
    }
  }
},
"@graph": {
  "$id": "#/properties/@graph",
  "type": "array",
  "items": {
    "$id": "#/properties/@graph/items",
    "type": "object",
    "properties": {
      "@id": {
        "$id": "#/properties/@graph/items/properties/@id",
        "type": "string"
      },
      "@type": {
        "$id": "#/properties/@graph/items/properties/@type",
```



```
        "type": "string"
      },
      "hasProfile": {
        "$id": "#/properties/@graph/items/properties/hasProfile",
        "type": "object",
        "properties": {
          "@id": {
            "$id":
"#/properties/@graph/items/properties/hasProfile/properties/@id",
            "type": "string"
          }
        }
      }
    }
  }
}
}
```

Based on the JSON Schema and the ontology concepts, the following code depicts an example of JSON-LD interface.

```
{
  "@context": {
    "core": "http://delta.linkeddata.es/def/core#",
    "saref": "https://w3id.org/saref#",
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "Price": {
      "@id": "core:Price"
    },
    "VirtualNode": {
      "@id": "core:VirtualNode"
    },
    "hasDuration": {
      "@id": "core:hasDuration"
    },
    "hasProfile": {
      "@id": "core:hasProfile"
    },
    "TemporalEntity": {
      "@id": "time:TemporalEntity"
    },
    "hasBeginning": {
      "@id": "core:hasBeginning"
    },
    "hasEnd": {
      "@id": "core:hasEnd"
    },
    "isAbout": {
      "@id": "saref:isAbout"
    },
    "hasMeasurement": {
      "@id": "saref:hasMeasurement"
    },
    "isMeasuredIn": {
      "@id": "saref:isMeasuredIn"
    }
  },
  "@graph": [
    {
      "@id": "core:VirtualNode01",
      "@type": "VirtualNode",
      "hasProfile": {
```

```
      "@id": "core:Profile1"
    },
    {
      "@id": "core:Profile1",
      "@type": "Profile",
      "hasDuration": {
        "@id": "core:TimeInterval1"
      },
      "isAbout": {
        "@id": "core:Energy01"
      }
    },
    {
      "@id": "core:Price01",
      "@type": "Price",
      "hasMeasurement": {
        "@id": "core:Measurement1"
      }
    },
    {
      "@id": "core: Measurement1",
      "@type": " Measurement ",
      "hasValue": {
        "@type": "xsd:float",
        "@value": "16.6"
      },
      "saref:isMeasuredIn": {
        "@id": "om:euro"
      }
    },
    {
      "@id": "core:TimeInterval1",
      "@type": "TemporalEntity",
      "hasBeginning": {
        "@id": "Instant01"
      },
      "hasEnd": {
        "@id": "Instant02"
      }
    },
    {
      "@id": "core:Instant01",
      "@type": "Instant",
      "inXSDDate": {
        "@type": "xsd:date",
        "@value": "2019-06-01"
      }
    },
    {
      "@id": "core:Instant02",
      "@type": "Instant",
      "inXSDDate": {
        "@type": "xsd:date",
        "@value": "2019-07-02"
      }
    }
  ]
}
```

A-8. FEIDs Clusters

The following JSON Schema describes the structure of the data that need to be contained in the FEIDs Clusters interface.

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "@context": {
      "$id": "#/properties/@context",
      "type": "object",
      "properties": {
        "core": {
          "$id": "#/properties/@context/properties/core",
          "type": "string"
        },
        "dc": {
          "$id": "#/properties/@context/properties/dc",
          "type": "string"
        },
        "dcterms": {
          "$id": "#/properties/@context/properties/dcterms",
          "type": "string"
        },
        "owl": {
          "$id": "#/properties/@context/properties/owl",
          "type": "string"
        },
        "rdf": {
          "$id": "#/properties/@context/properties/rdf",
          "type": "string"
        },
        "rdfs": {
          "$id": "#/properties/@context/properties/rdfs",
          "type": "string"
        },
        "terms": {
          "$id": "#/properties/@context/properties/terms",
          "type": "string"
        },
        "xsd": {
          "$id": "#/properties/@context/properties/xsd",
          "type": "string"
        },
        "Cluster": {
          "$id": "#/properties/@context/properties/Cluster",
          "type": "object",
          "properties": {
            "@id": {
              "$id":
                "#/properties/@context/properties/Cluster/properties/@id",
              "type": "string"
            }
          }
        },
        "hasCluster": {
          "$id": "#/properties/@context/properties/hasCluster",
          "type": "object",
          "properties": {
            "@id": {
```

```
        "$id":
"#/properties/@context/properties/hasCluster/properties/@id",
        "type": "string"
    }
}
},
"VirtualNode": {
    "$id": "#/properties/@context/properties/VirtualNode",
    "type": "object",
    "properties": {
        "@id": {
            "$id":
"#/properties/@context/properties/VirtualNode/properties/@id",
            "type": "string"
        }
    }
},
"hasAggregatedProperty": {
    "$id": "#/properties/@context/properties/hasAggregatedProperty",
    "type": "object",
    "properties": {
        "@id": {
            "$id":
"#/properties/@context/properties/hasAggregatedProperty/properties/@id",
            "type": "string"
        }
    }
}
},
"@graph": {
    "$id": "#/properties/@graph",
    "type": "array",
    "items": {
        "$id": "#/properties/@graph/items",
        "type": "object",
        "properties": {
            "@id": {
                "$id": "#/properties/@graph/items/properties/@id",
                "type": "string"
            },
            "@type": {
                "$id": "#/properties/@graph/items/properties/@type",
                "type": "string"
            }
        },
        "core:hasCluster": {
            "$id": "#/properties/@graph/items/properties/core:hasCluster",
            "type": "object",
            "properties": {
                "@id": {
                    "$id":
"#/properties/@graph/items/properties/core:hasCluster/properties/@id",
                    "type": "string"
                }
            }
        }
    }
}
}
}
}
}
```

Based on the JSON Schema and the ontology concepts, the following code depicts an example of JSON-LD interface.

```
{
  "@context": {
    "core": "http://delta.linkeddata.es/def/core#",
    "dc": "http://purl.org/dc/elements/1.1/",
    "dcterms": "http://purl.org/dc/terms/",
    "owl": "http://www.w3.org/2002/07/owl#",
    "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
    "terms": "http://purl.org/dc/terms/",
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "Cluster": {
      "@id": "core:Cluster"
    },
    "hasCluster": {
      "@id": "core:hasCluster"
    },
    "VirtualNode": {
      "@id": "core:VirtualNode"
    },
    "hasAggregatedProperty": {
      "@id": "core:hasAggregatedProperty"
    }
  },
  "@graph": [
    {
      "@id": "core:VirtualNode01",
      "@type": "core:VirtualNode",
      "core:hasCluster": {
        "@id": "core:Cluster01"
      }
    },
    {
      "@id": "core:Cluster01",
      "@type": "core:Cluster",
      "hasAggregatedProperty": {
        "@id": "core:Flexibility01"
      }
    },
    {
      "@id": "core:Flexibility01",
      "@type": "core:Flexibility",
      "isRelatedToMeasurement": {
        "@id": "core:Measurement01"
      }
    },
    {
      "@id": "core: Measurement1",
      "@type": " Measurement ",
      "hasValue": {
        "@type": "xsd:float",
        "@value": "136.1"
      }
    }
  ]
}
```

A-9. Flexibility forecast

The following JSON Schema describes the structure of the data that need to be contained in the Flexibility forecast interface.

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "@context": {
      "$id": "#/properties/@context",
      "type": "object",
      "properties": {
        "core": {
          "$id": "#/properties/@context/properties/core",
          "type": "string"
        },
        "saref": {
          "$id": "#/properties/@context/properties/saref",
          "type": "string"
        },
        "xsd": {
          "$id": "#/properties/@context/properties/xsd",
          "type": "string"
        },
        "Energy": {
          "$id": "#/properties/@context/properties/Energy",
          "type": "object",
          "properties": {
            "@id": {
              "$id":
                "#/properties/@context/properties/Energy/properties/@id",
              "type": "string"
            }
          }
        },
        "VirtualNode": {
          "$id": "#/properties/@context/properties/VirtualNode",
          "type": "object",
          "properties": {
            "@id": {
              "$id":
                "#/properties/@context/properties/VirtualNode/properties/@id",
              "type": "string"
            }
          }
        },
        "manage": {
          "$id": "#/properties/@context/properties/manage",
          "type": "object",
          "properties": {
            "@id": {
              "$id":
                "#/properties/@context/properties/manage/properties/@id",
              "type": "string"
            }
          }
        },
        "Measurement": {
          "$id": "#/properties/@context/properties/Measurement",
          "type": "object",
          "properties": {
```

```
"@id": {
  "$id":
"#/properties/@context/properties/Measurement/properties/@id",
  "type": "string"
}
},
"Flexibility": {
  "$id": "#/properties/@context/properties/Flexibility",
  "type": "object",
  "properties": {
    "@id": {
      "$id":
"#/properties/@context/properties/Flexibility/properties/@id",
      "type": "string"
    }
  }
},
"isRelatedtoMeasurement": {
  "$id": "#/properties/@context/properties/isRelatedtoMeasurement",
  "type": "object",
  "properties": {
    "@id": {
      "$id":
"#/properties/@context/properties/isRelatedtoMeasurement/properties/@id",
      "type": "string"
    }
  }
},
"measuresProperty": {
  "$id": "#/properties/@context/properties/measuresProperty",
  "type": "object",
  "properties": {
    "@id": {
      "$id":
"#/properties/@context/properties/measuresProperty/properties/@id",
      "type": "string"
    }
  }
},
"hasValue": {
  "$id": "#/properties/@context/properties/hasValue",
  "type": "object",
  "properties": {
    "@id": {
      "$id":
"#/properties/@context/properties/hasValue/properties/@id",
      "type": "string"
    }
  }
}
},
"@graph": {
  "$id": "#/properties/@graph",
  "type": "array",
  "items": {
    "$id": "#/properties/@graph/items",
    "type": "object",
    "properties": {
      "@id": {
```

```
{
  "@context": {
    "core": "http://delta.linkeddata.es/def/core#",
    "saref": "https://w3id.org/saref#",
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "Energy": {"@id": "core:Energy"},
    "VirtualNode": {"@id": "core:VirtualNode"},
    "manage": {"@id": "core:manage"},
    "Measurement": {"@id": "core:Measurement"},
    "Flexibility": {"@id": "core:Flexibility"},
    "isRelatedtoMeasurement": {"@id": "core:isRelatedtoMeasurement"},
    "measuresProperty": {"@id": "core:measuresProperty"},
    "hasValue": {"@id": "core:hasValue"}
  },
  "@graph": [
    {
      "@id": "core:VirtualNode01",
      "@type": [
        "core:VirtualNode"
      ],
      "core:manage": {"@id": "core:Device1"}
    },
    {
      "@id": "core:Flexibility01",
      "@type": [
        "core:Flexibility"
      ],
      "isRelatedToMeasurement": {"@id": "core:Measurement01"}
    },
    {
      "@id": "core: Measurement1",
      "@type": " Measurement ",
      "hasValue": {
        "@type": "xsd:float",
        "@value": "136.1"
      }
    }
  ]
}
```



```
    }
  },
  {
    "@id": "core:Device1",
    "@type": [
      "core:Device"
    ],
    "measuresProperty": [
      { "@id": "core:Flexibility01" }
    ]
  }
]
}
```

A-10. Forecasted profiling

The following JSON Schema describes the structure of the data that need to be contained in the interface.

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "http://example.com/root.json",
  "type": "object",
  "properties": {
    "@context": {
      "$id": "#/properties/@context",
      "type": "object",
      "properties": {
        "core": {
          "$id": "#/properties/@context/properties/core",
          "type": "string"
        },
        "saref": {
          "$id": "#/properties/@context/properties/saref",
          "type": "string"
        },
        "xsd": {
          "$id": "#/properties/@context/properties/xsd",
          "type": "string"
        },
        "Energy": {
          "$id": "#/properties/@context/properties/Energy",
          "type": "object",
          "properties": {
            "@id": {
              "$id":
                "#/properties/@context/properties/Energy/properties/@id",
              "type": "string"
            }
          }
        },
        "VirtualNode": {
          "$id": "#/properties/@context/properties/VirtualNode",
          "type": "object",
          "properties": {
            "@id": {
              "$id":
                "#/properties/@context/properties/VirtualNode/properties/@id",
              "type": "string"
            }
          }
        }
      }
    }
  }
}
```

```
    },
    "manage": {
      "$id": "#/properties/@context/properties/manage",
      "type": "object",
      "properties": {
        "@id": {
          "$id":
"#/properties/@context/properties/manage/properties/@id",
          "type": "string"
        }
      }
    },
    "Measurement": {
      "$id": "#/properties/@context/properties/Measurement",
      "type": "object",
      "properties": {
        "@id": {
          "$id":
"#/properties/@context/properties/Measurement/properties/@id",
          "type": "string"
        }
      }
    },
    "isRelatedtoMeasurement": {
      "$id": "#/properties/@context/properties/isRelatedtoMeasurement",
      "type": "object",
      "properties": {
        "@id": {
          "$id":
"#/properties/@context/properties/isRelatedtoMeasurement/properties/@id",
          "type": "string"
        }
      }
    },
    "measuresProperty": {
      "$id": "#/properties/@context/properties/measuresProperty",
      "type": "object",
      "properties": {
        "@id": {
          "$id":
"#/properties/@context/properties/measuresProperty/properties/@id",
          "type": "string"
        }
      }
    }
  },
  "@graph": {
    "$id": "#/properties/@graph",
    "type": "array",
    "items": {
      "$id": "#/properties/@graph/items",
      "type": "object",
      "properties": {
        "@id": {
          "$id": "#/properties/@graph/items/properties/@id",
          "type": "string"
        },
        "@type": {
          "$id": "#/properties/@graph/items/properties/@type",
          "type": "string"
        }
      }
    }
  }
}
```

```
    },
    "manage": {
      "$id": "#/properties/@graph/items/properties/manage",
      "type": "object",
      "properties": {
        "@id": {
          "$id":
"#/properties/@graph/items/properties/manage/properties/@id",
          "type": "string"
        }
      }
    }
  }
}
}
```

Based on the JSON Schema and the ontology concepts, the following code depicts an example of JSON-LD interface.

```
{
  "@context": {
    "core": "http://delta.linkeddata.es/def/core#",
    "saref": "https://w3id.org/saref#",
    "om": "http://www.wurvoc.org/vocabularies/om-1.8#",
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "TemporalEntity": {
      "@id": "time:TemporalEntity"
    },
    "VirtualNode": {
      "@id": "core:VirtualNode"
    },
    "hasPeriod": {
      "@id": "core:hasPeriod"
    },
    "hasProfile": {
      "@id": "core:hasProfile"
    },
    "Energy": {
      "@id": "core:Energy"
    },
    "hasBeginning": {
      "@id": "core:hasBeginning"
    },
    "hasEnd": {
      "@id": "core:hasEnd"
    },
    "isAbout": {
      "@id": "saref:isAbout"
    },
    "hasMeasurement": {
      "@id": "saref:hasMeasurement"
    },
    "isMeasuredIn": {
      "@id": "saref:isMeasuredIn"
    },
    "hasValue": {
      "@id": "saref:hasValue"
    },
    "Profile": {
      "@id": "core:Profile"
    }
  },
}
```

```
"Instant": {
  "@id": "core:Instant"
},
"@graph": [
  {
    "@id": "core:VirtualNode01",
    "@type": "VirtualNode",
    "hasProfile": {
      "@id": "core:Profile1"
    }
  },
  {
    "@id": "core:Profile1",
    "@type": "Profile",
    "hasPeriod": {
      "@id": "core:TimeInterval1"
    },
    "isAbout": {
      "@id": "core:Energy01"
    }
  },
  {
    "@id": "core: Energy01",
    "@type": "Energy",
    "hasMeasurement": {
      "@id": "core:Measurement1"
    }
  },
  {
    "@id": "core: Measurement1",
    "@type": "Measurement",
    "hasValue": {
      "@type": "xsd:float",
      "@value": "78.2"
    },
    "saref:isMeasuredIn": {
      "@id": "om:watt"
    }
  },
  {
    "@id": "core:TimeInterval1",
    "@type": "TemporalEntity",
    "hasBeginning": {
      "@id": "Instant01"
    },
    "hasEnd": {
      "@id": "Instant02"
    }
  },
  {
    "@id": "core:Instant01",
    "@type": "Instant",
    "inXSDDate": {
      "@type": "xsd:date",
      "@value": "2019-01-01"
    }
  },
  {
    "@id": "core:Instant02",
    "@type": "Instant",
```

```
    "inXSDDate": {  
      "@type": "xsd:date",  
      "@value": "2019-01-02"  
    }  
  }  
]  
}
```

A-11. Historical consumption

The following JSON Schema describes the structure of the data that need to be contained in the Historical consumption interface.

```
{  
  "$schema": "http://json-schema.org/draft-07/schema#",  
  "type": "object",  
  "properties": {  
    "@context": {  
      "$id": "#/properties/@context",  
      "type": "object",  
      "properties": {  
        "core": {  
          "$id": "#/properties/@context/properties/core",  
          "type": "string"  
        },  
        "saref": {  
          "$id": "#/properties/@context/properties/saref",  
          "type": "string"  
        },  
        "xsd": {  
          "$id": "#/properties/@context/properties/xsd",  
          "type": "string"  
        },  
        "Energy": {  
          "$id": "#/properties/@context/properties/Energy",  
          "type": "object",  
          "properties": {  
            "@id": {  
              "$id":  
"#/properties/@context/properties/Energy/properties/@id",  
              "type": "string"  
            }  
          }  
        },  
        "VirtualNode": {  
          "$id": "#/properties/@context/properties/VirtualNode",  
          "type": "object",  
          "properties": {  
            "@id": {  
              "$id":  
"#/properties/@context/properties/VirtualNode/properties/@id",  
              "type": "string"  
            }  
          }  
        },  
        "manage": {  
          "$id": "#/properties/@context/properties/manage",  
          "type": "object",  
          "properties": {  
            "@id": {
```

```
        "$id":
"#/properties/@context/properties/manage/properties/@id",
        "type": "string"
    }
}
},
"Measurement": {
    "$id": "#/properties/@context/properties/Measurement",
    "type": "object",
    "properties": {
        "@id": {
            "$id":
"#/properties/@context/properties/Measurement/properties/@id",
            "type": "string"
        }
    }
},
"isRelatedtoMeasurement": {
    "$id": "#/properties/@context/properties/isRelatedtoMeasurement",
    "type": "object",
    "properties": {
        "@id": {
            "$id":
"#/properties/@context/properties/isRelatedtoMeasurement/properties/@id",
            "type": "string"
        }
    }
},
"measuresProperty": {
    "$id": "#/properties/@context/properties/measuresProperty",
    "type": "object",
    "properties": {
        "@id": {
            "$id":
"#/properties/@context/properties/measuresProperty/properties/@id",
            "type": "string"
        }
    }
}
},
"@graph": {
    "$id": "#/properties/@graph",
    "type": "array",
    "items": {
        "$id": "#/properties/@graph/items",
        "type": "object",
        "properties": {
            "@id": {
                "$id": "#/properties/@graph/items/properties/@id",
                "type": "string"
            },
            "@type": {
                "$id": "#/properties/@graph/items/properties/@type",
                "type": "string"
            },
            "manage": {
                "$id": "#/properties/@graph/items/properties/manage",
                "type": "object",
                "properties": {
                    "@id": {
```

```
        "$id":
"#/properties/@graph/items/properties/manage/properties/@id",
        "type": "string"
    }
}
}
}
}
}
}
}
```

Based on the JSON Schema and the ontology concepts, the following code depicts an example of JSON-LD interface.

```
{
  "@context": {
    "core": "http://delta.linkeddata.es/def/core#",
    "saref": "https://w3id.org/saref#",
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "Energy": {
      "@id": "core:Energy"
    },
    "VirtualNode": {
      "@id": "core:VirtualNode"
    },
    "manage": {
      "@id": "core:manage"
    },
    "Measurement": {
      "@id": "core:Measurement"
    },
    "isRelatedtoMeasurement": {
      "@id": "core:isRelatedtoMeasurement"
    },
    "measuresProperty": {
      "@id": "core:measuresProperty"
    }
  },
  "@graph": [
    {
      "@id": "core:VirtualNode01",
      "@type": "VirtualNode",
      "manage": {
        "@id": "core:Device1"
      }
    },
    {
      "@id": "core:Device1",
      "@type": "Device",
      "measuresProperty": {
        "@id": "core:Energy01"
      }
    },
    {
      "@id": "core:Energy01",
      "@type": "Energy",
      "isRelatedToMeasurement": {
        "@id": "core:Measurement01"
      }
    }
  ],
  {

```

```
"@id": "core: Measurement1",
"@type": "Measurement",
"hasValue": {
  "@type": "xsd:float",
  "@value": "-89.2"
},
"saref:isMeasuredIn": {
  "@id": "om:watt"
}
}
]
}
```

A-12. Historical generation

The following JSON Schema describes the structure of the data that need to be contained in the interface.

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "http://example.com/root.json",
  "type": "object",
  "properties": {
    "@context": {
      "$id": "#/properties/@context",
      "type": "object",
      "properties": {
        "core": {
          "$id": "#/properties/@context/properties/core",
          "type": "string"
        },
        "saref": {
          "$id": "#/properties/@context/properties/saref",
          "type": "string"
        },
        "xsd": {
          "$id": "#/properties/@context/properties/xsd",
          "type": "string"
        },
        "Energy": {
          "$id": "#/properties/@context/properties/Energy",
          "type": "object",
          "properties": {
            "@id": {
              "$id":
                "#/properties/@context/properties/Energy/properties/@id",
              "type": "string"
            }
          }
        },
        "VirtualNode": {
          "$id": "#/properties/@context/properties/VirtualNode",
          "type": "object",
          "properties": {
            "@id": {
              "$id":
                "#/properties/@context/properties/VirtualNode/properties/@id",
              "type": "string"
            }
          }
        }
      }
    }
  },
}
```



```
"manage": {
  "$id": "#/properties/@context/properties/manage",
  "type": "object",
  "properties": {
    "@id": {
      "$id":
"#/properties/@context/properties/manage/properties/@id",
      "type": "string"
    }
  }
},
"Measurement": {
  "$id": "#/properties/@context/properties/Measurement",
  "type": "object",
  "properties": {
    "@id": {
      "$id":
"#/properties/@context/properties/Measurement/properties/@id",
      "type": "string"
    }
  }
},
"isRelatedtoMeasurement": {
  "$id": "#/properties/@context/properties/isRelatedtoMeasurement",
  "type": "object",
  "properties": {
    "@id": {
      "$id":
"#/properties/@context/properties/isRelatedtoMeasurement/properties/@id",
      "type": "string"
    }
  }
},
"measuresProperty": {
  "$id": "#/properties/@context/properties/measuresProperty",
  "type": "object",
  "properties": {
    "@id": {
      "$id":
"#/properties/@context/properties/measuresProperty/properties/@id",
      "type": "string"
    }
  }
}
},
"@graph": {
  "$id": "#/properties/@graph",
  "type": "array",
  "items": {
    "$id": "#/properties/@graph/items",
    "type": "object",
    "properties": {
      "@id": {
        "$id": "#/properties/@graph/items/properties/@id",
        "type": "string"
      },
      "@type": {
        "$id": "#/properties/@graph/items/properties/@type",
        "type": "string"
      }
    }
  },
  "type": "array"
}
```

```
    "manage": {
      "$id": "#/properties/@graph/items/properties/manage",
      "type": "object",
      "properties": {
        "@id": {
          "$id":
"#/properties/@graph/items/properties/manage/properties/@id",
          "type": "string"
        }
      }
    }
  }
}
}
```

Based on the JSON Schema and the ontology concepts, the following code depicts an example of JSON-LD interface that exchange the historical generation.

```
{
  "@context": {
    "core": "http://delta.linkeddata.es/def/core#",
    "saref": "https://w3id.org/saref#",
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "Energy": {
      "@id": "core:Energy"
    },
    "VirtualNode": {
      "@id": "core:VirtualNode"
    },
    "manage": {
      "@id": "core:manage"
    },
    "Measurement": {
      "@id": "core:Measurement"
    },
    "isRelatedtoMeasurement": {
      "@id": "core:isRelatedtoMeasurement"
    },
    "measuresProperty": {
      "@id": "core:measuresProperty"
    }
  },
  "@graph": [
    {
      "@id": "core:VirtualNode01",
      "@type": "VirtualNode",
      "manage": {
        "@id": "core:Device1"
      }
    },
    {
      "@id": "core:Device1",
      "@type": "Device",
      "measuresProperty": [
        {
          "@id": "core:Energy01"
        }
      ]
    },
    {
      "@id": "core:Energy01",
```

```
    "@type": "Energy",
    "isRelatedToMeasurement": {
      "@id": "core:Measurement01"
    }
  },
  {
    "@id": "core: Measurement1",
    "@type": "Measurement",
    "hasValue": {
      "@type": "xsd:float",
      "@value": "89.2"
    },
    "saref:isMeasuredIn": {
      "@id": "om:watt"
    }
  }
]
}
```

A-13. KPIs

The following JSON Schema describes the structure of the data that need to be contained in the interface.

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "@context": {
      "$id": "#/properties/@context",
      "type": "object",
      "properties": {
        "core": {
          "$id": "#/properties/@context/properties/core",
          "type": "string"
        },
        "saref": {
          "$id": "#/properties/@context/properties/saref",
          "type": "string"
        },
        "xsd": {
          "$id": "#/properties/@context/properties/xsd",
          "type": "string"
        },
        "VirtualNode": {
          "$id": "#/properties/@context/properties/VirtualNode",
          "type": "object",
          "properties": {
            "@id": {
              "$id":
                "#/properties/@context/properties/VirtualNode/properties/@id",
              "type": "string"
            }
          }
        }
      },
      "hasMeasurement": {
        "$id": "#/properties/@context/properties/hasMeasurement",
        "type": "object",
        "properties": {
          "@id": {
            "$id":
              "#/properties/@context/properties/hasMeasurement/properties/@id",
```

```

        "type": "string"
      }
    },
    "hasValue": {
      "$id": "#/properties/@context/properties/hasValue",
      "type": "object",
      "properties": {
        "@id": {
          "$id":
"#/properties/@context/properties/hasValue/properties/@id",
          "type": "string"
        }
      }
    }
  },
  "@graph": {
    "$id": "#/properties/@graph",
    "type": "array",
    "items": {
      "$id": "#/properties/@graph/items",
      "type": "object",
      "properties": {
        "@id": {
          "$id": "#/properties/@graph/items/properties/@id",
          "type": "string"
        },
        "@type": {
          "$id": "#/properties/@graph/items/properties/@type",
          "type": "string"
        },
        "hasMeasurement": {
          "$id": "#/properties/@graph/items/properties/hasMeasurement",
          "type": "object",
          "properties": {
            "@id": {
              "$id":
"#/properties/@graph/items/properties/hasMeasurement/properties/@id",
              "type": "string"
            }
          }
        }
      }
    }
  }
}

```

Based on the JSON Schema and the ontology concepts, the following code depicts an example of JSON-LD interface.

```

{
  "@context": {
    "core": "http://delta.linkeddata.es/def/core#",
    "saref": "https://w3id.org/saref#",
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "VirtualNode": {
      "@id": "core:VirtualNode"
    },
    "hasMeasurement": {
      "@id": "saref:hasMeasurement"
    }
  },

```

```
    "hasValue": {
      "@id": "saref:hasValue"
    }
  },
  "@graph": [
    {
      "@id": "core:KPI1",
      "@type": "KeyPerformanceIndicator",
      "hasMeasurement": {
        "@id": "core:Measurement1"
      }
    },
    {
      "@id": "core:Measurement1",
      "@type": "Measurement",
      "hasValue": {
        "@type": "xsd:float",
        "@value": "14.2"
      }
    }
  ]
}
```

A-14. Market settlement

The following JSON Schema describes the structure of the data that need to be contained in the interface.

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "@context": {
      "$id": "#/properties/@context",
      "type": "object",
      "properties": {
        "core": {
          "$id": "#/properties/@context/properties/core",
          "type": "string"
        },
        "dc": {
          "$id": "#/properties/@context/properties/dc",
          "type": "string"
        },
        "dcterms": {
          "$id": "#/properties/@context/properties/dcterms",
          "type": "string"
        },
        "owl": {
          "$id": "#/properties/@context/properties/owl",
          "type": "string"
        },
        "rdf": {
          "$id": "#/properties/@context/properties/rdf",
          "type": "string"
        },
        "rdfs": {
          "$id": "#/properties/@context/properties/rdfs",
          "type": "string"
        },
        "terms": {
```

```
    "$id": "#/properties/@context/properties/terms",
    "type": "string"
  },
  "xsd": {
    "$id": "#/properties/@context/properties/xsd",
    "type": "string"
  },
  "Cluster": {
    "$id": "#/properties/@context/properties/Cluster",
    "type": "object",
    "properties": {
      "@id": {
        "$id":
"#/properties/@context/properties/Cluster/properties/@id",
        "type": "string"
      }
    }
  },
  "hasCluster": {
    "$id": "#/properties/@context/properties/hasCluster",
    "type": "object",
    "properties": {
      "@id": {
        "$id":
"#/properties/@context/properties/hasCluster/properties/@id",
        "type": "string"
      }
    }
  },
  "VirtualNode": {
    "$id": "#/properties/@context/properties/VirtualNode",
    "type": "object",
    "properties": {
      "@id": {
        "$id":
"#/properties/@context/properties/VirtualNode/properties/@id",
        "type": "string"
      }
    }
  }
},
"@graph": {
  "$id": "#/properties/@graph",
  "type": "array",
  "items": {
    "$id": "#/properties/@graph/items",
    "type": "object",
    "properties": {
      "@id": {
        "$id": "#/properties/@graph/items/properties/@id",
        "type": "string"
      },
      "@type": {
        "$id": "#/properties/@graph/items/properties/@type",
        "type": "string"
      },
      "core:consumes": {
        "$id": "#/properties/@graph/items/properties/core:consumes",
        "type": "object",
        "properties": {
```

```

    "@id": {
      "$id":
"#/properties/@graph/items/properties/core:consumes/properties/@id",
      "type": "string"
    }
  }
}
}
}
}
}
}

```

Based on the JSON Schema and the ontology concepts, the following code depicts an example of JSON-LD interface.

```
{
  "@context": {
    "core": "http://delta.linkeddata.es/def/core#",
    "dc": "http://purl.org/dc/elements/1.1/",
    "dcterms": "http://purl.org/dc/terms/",
    "owl": "http://www.w3.org/2002/07/owl#",
    "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
    "terms": "http://purl.org/dc/terms/",
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "Cluster": {
      "@id": "core:Cluster"
    },
    "hasCluster": {
      "@id": "core:hasCluster"
    },
    "VirtualNode": {
      "@id": "core:VirtualNode"
    }
  },
  "@graph": [
    {
      "@id": "core:Aggregator01",
      "@type": [
        "core:Aggregator"
      ],
      "core:consumes": {
        "@id": "core:MarketSettlement01"
      }
    },
    {
      "@id": "core: MarketSettlement01",
      "@type": [
        "core:MarketSettlement"
      ]
    }
  ]
}
```

A-15. Node Profiling

The following JSON Schema describes the structure of the data that need to be contained in the interface.

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "http://example.com/root.json",
```

```
"type": "object",
"properties": {
  "@context": {
    "$id": "#/properties/@context",
    "type": "object",
    "properties": {
      "core": {
        "$id": "#/properties/@context/properties/core",
        "type": "string"
      },
      "saref": {
        "$id": "#/properties/@context/properties/saref",
        "type": "string"
      },
      "xsd": {
        "$id": "#/properties/@context/properties/xsd",
        "type": "string"
      },
      "Energy": {
        "$id": "#/properties/@context/properties/Energy",
        "type": "object",
        "properties": {
          "@id": {
            "$id":
"#/properties/@context/properties/Energy/properties/@id",
            "type": "string"
          }
        }
      },
      "VirtualNode": {
        "$id": "#/properties/@context/properties/VirtualNode",
        "type": "object",
        "properties": {
          "@id": {
            "$id":
"#/properties/@context/properties/VirtualNode/properties/@id",
            "type": "string"
          }
        }
      },
      "manage": {
        "$id": "#/properties/@context/properties/manage",
        "type": "object",
        "properties": {
          "@id": {
            "$id":
"#/properties/@context/properties/manage/properties/@id",
            "type": "string"
          }
        }
      },
      "Measurement": {
        "$id": "#/properties/@context/properties/Measurement",
        "type": "object",
        "properties": {
          "@id": {
            "$id":
"#/properties/@context/properties/Measurement/properties/@id",
            "type": "string"
          }
        }
      }
    }
  }
}
```



```
    },
    "isRelatedtoMeasurement": {
      "$id": "#/properties/@context/properties/isRelatedtoMeasurement",
      "type": "object",
      "properties": {
        "@id": {
          "$id":
"#/properties/@context/properties/isRelatedtoMeasurement/properties/@id",
          "type": "string"
        }
      }
    },
    "measuresProperty": {
      "$id": "#/properties/@context/properties/measuresProperty",
      "type": "object",
      "properties": {
        "@id": {
          "$id":
"#/properties/@context/properties/measuresProperty/properties/@id",
          "type": "string"
        }
      }
    }
  },
  "@graph": {
    "$id": "#/properties/@graph",
    "type": "array",
    "items": {
      "$id": "#/properties/@graph/items",
      "type": "object",
      "properties": {
        "@id": {
          "$id": "#/properties/@graph/items/properties/@id",
          "type": "string"
        },
        "@type": {
          "$id": "#/properties/@graph/items/properties/@type",
          "type": "string"
        },
        "manage": {
          "$id": "#/properties/@graph/items/properties/manage",
          "type": "object",
          "properties": {
            "@id": {
              "$id":
"#/properties/@graph/items/properties/manage/properties/@id",
              "type": "string"
            }
          }
        }
      }
    }
  }
}
```

Based on the JSON Schema and the ontology concepts, the following code depicts an example of JSON-LD interface that exchange the node profiling.

```
{
  "@context": {
    "core": "http://delta.linkeddata.es/def/core#",

```

```
"saref": "https://w3id.org/saref#",
"om": "http://www.wurvoc.org/vocabularies/om-1.8#",
"xsd": "http://www.w3.org/2001/XMLSchema#",
"TemporalEntity": {
  "@id": "time:TemporalEntity"
},
"VirtualNode": {
  "@id": "core:VirtualNode"
},
"hasPeriod": {
  "@id": "core:hasPeriod"
},
"hasProfile": {
  "@id": "core:hasProfile"
},
"Energy": {
  "@id": "core:Energy"
},
"hasBeginning": {
  "@id": "core:hasBeginning"
},
"hasEnd": {
  "@id": "core:hasEnd"
},
"isAbout": {
  "@id": "saref:isAbout"
},
"hasMeasurement": {
  "@id": "saref:hasMeasurement"
},
"isMeasuredIn": {
  "@id": "saref:isMeasuredIn"
},
"hasValue": {
  "@id": "saref:hasValue"
},
"Profile": {
  "@id": "core:Profile"
},
"Instant": {
  "@id": "core:Instant"
}
},
"@graph": [
  {
    "@id": "core:VirtualNode01",
    "@type": "VirtualNode",
    "hasProfile": {
      "@id": "core:Profile1"
    }
  },
  {
    "@id": "core:Profile1",
    "@type": "Profile",
    "hasPeriod": {
      "@id": "core:TimeInterval1"
    },
    "isAbout": {
      "@id": "core:Energy01"
    }
  }
],
```

```
{
  "@id": "core: Energy01",
  "@type": "Energy",
  "hasMeasurement": {
    "@id": "core:Measurement1"
  }
},
{
  "@id": "core: Measurement1",
  "@type": "Measurement",
  "hasValue": {
    "@type": "xsd:float",
    "@value": "78.2"
  },
  "saref:isMeasuredIn": {
    "@id": "om:watt"
  }
},
{
  "@id": "core:TimeInterval1",
  "@type": "TemporalEntity",
  "hasBeginning": {
    "@id": "Instant01"
  },
  "hasEnd": {
    "@id": "Instant02"
  }
},
{
  "@id": "core:Instant01",
  "@type": "Instant",
  "inXSDDate": {
    "@type": "xsd:date",
    "@value": "2019-01-01"
  }
},
{
  "@id": "core:Instant02",
  "@type": "Instant",
  "inXSDDate": {
    "@type": "xsd:date",
    "@value": "2019-01-02"
  }
}
]
```

A-16. Rewards

The following JSON Schema describes the structure of the data that need to be contained in the interface.

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "@context": {
      "$id": "#/properties/@context",
      "type": "object",
      "properties": {
        "core": {
          "$id": "#/properties/@context/properties/core",
```

```
        "type": "string"
      },
      "xsd": {
        "$id": "#/properties/@context/properties/xsd",
        "type": "string"
      },
      "Customer": {
        "$id": "#/properties/@context/properties/Customer",
        "type": "object",
        "properties": {
          "@id": {
            "$id":
"#/properties/@context/properties/Customer/properties/@id",
            "type": "string"
          }
        }
      },
      "Device": {
        "$id": "#/properties/@context/properties/Device",
        "type": "object",
        "properties": {
          "@id": {
            "$id":
"#/properties/@context/properties/Device/properties/@id",
            "type": "string"
          }
        }
      },
      "owns": {
        "$id": "#/properties/@context/properties/owns",
        "type": "object",
        "properties": {
          "@id": {
            "$id":
"#/properties/@context/properties/owns/properties/@id",
            "type": "string"
          }
        }
      },
      "@graph": {
        "$id": "#/properties/@graph",
        "type": "array",
        "items": {
          "$id": "#/properties/@graph/items",
          "type": "object",
          "properties": {
            "@id": {
              "$id": "#/properties/@graph/items/properties/@id",
              "type": "string"
            },
            "@type": {
              "$id": "#/properties/@graph/items/properties/@type",
              "type": "string"
            },
            "reward ": {
              "$id": "#/properties/@graph/items/properties/reward ",
              "type": "object",
              "properties": {
                "@id": {
```

```
        "$id": "#/properties/@graph/items/properties/reward  
/properties/@id",  
        "type": "string"  
    }  
    }  
    }  
    }  
    }  
    }  
    }  
}
```

Based on the JSON Schema and the ontology concepts, the following code depicts an example of JSON-LD interface.

```
{  
  "@context": {  
    "core": "http://delta.linkeddata.es/def/core#",  
    "xsd": "http://www.w3.org/2001/XMLSchema#",  
    "Customer": {  
      "@id": "core:Customer"  
    },  
    "Device": {  
      "@id": "core:Device"  
    },  
    "owns": {  
      "@id": "core:owns"  
    }  
  },  
  "@graph": [  
    {  
      "@id": "core:Customer01",  
      "@type": "Customer",  
      "reward": {  
        "@id": "core:Reward1"  
      }  
    },  
    {  
      "@id": "core:Reward1",  
      "@type": "core:Reward",  
      "hasValue": {  
        "@type": "xsd:float",  
        "@value": "6.2"  
      },  
      "isMeasuredIn": {  
        "@id": "om:euro"  
      }  
    }  
  ]  
}
```

A-17. Smart contracts

The following JSON Schema describes the structure of the data that need to be contained in the interface.

```
{  
  "$schema": "http://json-schema.org/draft-07/schema#",  
  "type": "object",  
  "properties": {  
    "@context": {  
      "$id": "#/properties/@context",  
      "type": "object",
```

```
"properties": {
  "core": {
    "$id": "#/properties/@context/properties/core",
    "type": "string"
  },
  "saref": {
    "$id": "#/properties/@context/properties/saref",
    "type": "string"
  },
  "xsd": {
    "$id": "#/properties/@context/properties/xsd",
    "type": "string"
  },
  "TemporalEntity": {
    "$id": "#/properties/@context/properties/TemporalEntity",
    "type": "object",
    "properties": {
      "@id": {
        "$id":
"#/properties/@context/properties/TemporalEntity/properties/@id",
        "type": "string"
      }
    }
  },
  "VirtualNode": {
    "$id": "#/properties/@context/properties/VirtualNode",
    "type": "object",
    "properties": {
      "@id": {
        "$id":
"#/properties/@context/properties/VirtualNode/properties/@id",
        "type": "string"
      }
    }
  },
  "hasDuration": {
    "$id": "#/properties/@context/properties/hasDuration",
    "type": "object",
    "properties": {
      "@id": {
        "$id":
"#/properties/@context/properties/hasDuration/properties/@id",
        "type": "string"
      }
    }
  },
  "hasProfile": {
    "$id": "#/properties/@context/properties/hasProfile",
    "type": "object",
    "properties": {
      "@id": {
        "$id":
"#/properties/@context/properties/hasProfile/properties/@id",
        "type": "string"
      }
    }
  },
  "Energy": {
    "$id": "#/properties/@context/properties/Energy",
    "type": "object",
    "properties": {
```

```
        "@id": {
            "$id":
"#/properties/@context/properties/Energy/properties/@id",
            "type": "string"
        }
    },
    "hasBeginning": {
        "$id": "#/properties/@context/properties/hasBeginning",
        "type": "object",
        "properties": {
            "@id": {
                "$id":
"#/properties/@context/properties/hasBeginning/properties/@id",
                "type": "string"
            }
        }
    },
    "hasEnd": {
        "$id": "#/properties/@context/properties/hasEnd",
        "type": "object",
        "properties": {
            "@id": {
                "$id":
"#/properties/@context/properties/hasEnd/properties/@id",
                "type": "string"
            }
        }
    },
    "isAbout": {
        "$id": "#/properties/@context/properties/isAbout",
        "type": "object",
        "properties": {
            "@id": {
                "$id":
"#/properties/@context/properties/isAbout/properties/@id",
                "type": "string"
            }
        }
    },
    "hasMeasurement": {
        "$id": "#/properties/@context/properties/hasMeasurement",
        "type": "object",
        "properties": {
            "@id": {
                "$id":
"#/properties/@context/properties/hasMeasurement/properties/@id",
                "type": "string"
            }
        }
    },
    "isMeasuredIn": {
        "$id": "#/properties/@context/properties/isMeasuredIn",
        "type": "object",
        "properties": {
            "@id": {
                "$id":
"#/properties/@context/properties/isMeasuredIn/properties/@id",
                "type": "string"
            }
        }
    }
}
```

```
    },
    "hasValue": {
      "$id": "#/properties/@context/properties/hasValue",
      "type": "object",
      "properties": {
        "@id": {
          "$id":
"#/properties/@context/properties/hasValue/properties/@id",
          "type": "string"
        }
      }
    },
    "Profile": {
      "$id": "#/properties/@context/properties/Profile",
      "type": "object",
      "properties": {
        "@id": {
          "$id":
"#/properties/@context/properties/Profile/properties/@id",
          "type": "string"
        }
      }
    }
  },
  "@graph": {
    "$id": "#/properties/@graph",
    "type": "array",
    "items": {
      "$id": "#/properties/@graph/items",
      "type": "object",
      "properties": {
        "@id": {
          "$id": "#/properties/@graph/items/properties/@id",
          "type": "string"
        },
        "@type": {
          "$id": "#/properties/@graph/items/properties/@type",
          "type": "string"
        },
        "agreesOn": {
          "$id": "#/properties/@graph/items/properties/agreesOn",
          "type": "object",
          "properties": {
            "@id": {
              "$id":
"#/properties/@graph/items/properties/agreesOn/properties/@id",
              "type": "string"
            }
          }
        }
      }
    },
    "hasPayment": {
      "$id": "#/properties/@graph/items/properties/hasPayment",
      "type": "object",
      "properties": {
        "@id": {
          "$id":
"#/properties/@graph/items/properties/hasPayment/properties/@id",
          "type": "string"
        }
      }
    }
  }
}
```



```
    }  
  }  
}  
}
```

Based on the JSON Schema and the ontology concepts, the following code depicts an example of JSON-LD interface.

```
{  
  "@context": {  
    "core": "http://delta.linkeddata.es/def/core#",  
    "saref": "https://w3id.org/saref#",  
    "xsd": "http://www.w3.org/2001/XMLSchema#",  
    "TemporalEntity": {  
      "@id": "time:TemporalEntity"  
    },  
    "VirtualNode": {  
      "@id": "core:VirtualNode"  
    },  
    "hasDuration": {  
      "@id": "core:hasDuration"  
    },  
    "hasProfile": {  
      "@id": "core:hasProfile"  
    },  
    "Energy": {  
      "@id": "core:Energy"  
    },  
    "hasBeginning": {  
      "@id": "core:hasBeginning"  
    },  
    "hasEnd": {  
      "@id": "core:hasEnd"  
    },  
    "isAbout": {  
      "@id": "saref:isAbout"  
    },  
    "hasMeasurement": {  
      "@id": "saref:hasMeasurement"  
    },  
    "isMeasuredIn": {  
      "@id": "saref:isMeasuredIn"  
    },  
    "hasValue": {  
      "@id": "saref:hasValue"  
    },  
    "Profile": {  
      "@id": "saref:Profile"  
    }  
  },  
  "@graph": [  
    {  
      "@id": "core:SmartContract01",  
      "@type": "SmartContract",  
      "agreesOn": {  
        "@id": "core:Energy01"  
      },  
      "hasPayment": {  
        "@id": "core:Payment01"  
      }  
    }  
  ],  
}
```

```
{
  "@id": "core: Energy01",
  "@type": "Energy",
  "hasMeasurement": {
    "@id": "core:Measurement1"
  }
},
{
  "@id": "core: Measurement1",
  "@type": "Load",
  "hasValue": {
    "@type": "xsd:float",
    "@value": "1.2"
  },
  "saref:isMeasuredIn": {
    "@id": "om:watt"
  }
},
{
  "@id": "core:Payment01",
  "@type": "Payment"
}
]
```

A-18. Status signals

The following JSON Schema describes the structure of the data that need to be contained in the interface.

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "@context": {
      "$id": "#/properties/@context",
      "type": "object",
      "properties": {
        "core": {
          "$id": "#/properties/@context/properties/core",
          "type": "string"
        },
        "saref": {
          "$id": "#/properties/@context/properties/saref",
          "type": "string"
        },
        "xsd": {
          "$id": "#/properties/@context/properties/xsd",
          "type": "string"
        },
        "VirtualNode": {
          "$id": "#/properties/@context/properties/VirtualNode",
          "type": "object",
          "properties": {
            "@id": {
              "$id":
                "#/properties/@context/properties/VirtualNode/properties/@id",
              "type": "string"
            }
          }
        }
      },
      "StatusSignal": {
```

Based on the JSON Schema and the ontology concepts, the following code depicts an example of JSON-LD interface.

Page 139

```
    "isValidDuringPeriod": {"@id": "core:TimeInterval"}
  },
  {
    "@id": "core:TimeInterval",
    "@type": "TemporalEntity",
    "hasBeginning": {"@id": "Instant01"},
    "hasEnd": {"@id": "Instant02"}
  },
  {
    "@id": "core:Instant01",
    "@type": "Instant",
    "inXSDDate": {
      "@type": "xsd:date",
      "@value": "2019-04-11"
    }
  },
  {
    "@id": "core:Instant02",
    "@type": "Instant",
    "inXSDDate": {
      "@type": "xsd:date",
      "@value": "2019-04-12"
    }
  }
]
}
```

A-19. System constraints

The following JSON Schema describes the structure of the data that need to be contained in the interface.

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "@context": {
      "$id": "#/properties/@context",
      "type": "object",
      "properties": {
        "core": {
          "$id": "#/properties/@context/properties/core",
          "type": "string"
        },
        "saref": {
          "$id": "#/properties/@context/properties/saref",
          "type": "string"
        },
        "xsd": {
          "$id": "#/properties/@context/properties/xsd",
          "type": "string"
        },
        "FEID": {
          "$id": "#/properties/@context/properties/FEID",
          "type": "object",
          "properties": {
            "@id": {
              "$id":
                "#/properties/@context/properties/FEID/properties/@id",
              "type": "string"
            }
          }
        }
      }
    }
  }
}
```

```
    }
  },
  "SystemConstraint": {
    "$id": "#/properties/@context/properties/SystemConstraint",
    "type": "object",
    "properties": {
      "@id": {
        "$id":
"#/properties/@context/properties/SystemConstraint/properties/@id",
        "type": "string"
      }
    }
  },
  "hasSystemConstraints": {
    "$id": "#/properties/@context/properties/hasSystemConstraints",
    "type": "object",
    "properties": {
      "@id": {
        "$id":
"#/properties/@context/properties/hasSystemConstraints/properties/@id",
        "type": "string"
      }
    }
  }
},
"@graph": {
  "$id": "#/properties/@graph",
  "type": "array",
  "items": {
    "$id": "#/properties/@graph/items",
    "type": "object",
    "properties": {
      "@id": {
        "$id": "#/properties/@graph/items/properties/@id",
        "type": "string"
      },
      "@type": {
        "$id": "#/properties/@graph/items/properties/@type",
        "type": "string"
      },
      "hasSystemConstraint": {
        "$id":
"#/properties/@graph/items/properties/hasSystemConstraint",
        "type": "object",
        "properties": {
          "@id": {
            "$id":
"#/properties/@graph/items/properties/hasSystemConstraint/properties/@id",
            "type": "string"
          }
        }
      }
    }
  }
}
```

Based on the JSON Schema and the ontology concepts, the following code depicts an example of JSON-LD interface.

```
{
```

```
"@context": {
  "core": "http://delta.linkeddata.es/def/core#",
  "saref": "https://w3id.org/saref#",
  "xsd": "http://www.w3.org/2001/XMLSchema#",
  "FEID": {
    "@id": "core:FEID"
  },
  "SystemConstraint": {
    "@id": "core:SystemConstraint"
  },
  "hasSystemConstraints": {
    "@id": "core:hasSystemConstraints"
  }
},
"@graph": [
  {
    "@id": "core:FEID01",
    "@type": "FEID",
    "hasSystemConstraint": {
      "@id": "core:SystemConstraint01"
    }
  },
  {
    "@id": "core: SystemConstraint01",
    "@type": "SystemConstraint"
  }
]
}
```

A-20. Transactions

The following JSON Schema describes the structure of the data that need to be contained in the interface.

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "@context": {
      "$id": "#/properties/@context",
      "type": "object",
      "properties": {
        "core": {
          "$id": "#/properties/@context/properties/core",
          "type": "string"
        },
        "saref": {
          "$id": "#/properties/@context/properties/saref",
          "type": "string"
        },
        "xsd": {
          "$id": "#/properties/@context/properties/xsd",
          "type": "string"
        },
        "VirtualNode": {
          "$id": "#/properties/@context/properties/VirtualNode",
          "type": "object",
          "properties": {
            "@id": {
              "$id":
                "#/properties/@context/properties/VirtualNode/properties/@id",
```

```
        "type": "string"
      }
    },
    "Transaction": {
      "$id": "#/properties/@context/properties/Transaction",
      "type": "object",
      "properties": {
        "@id": {
          "$id":
"#/properties/@context/properties/Transaction/properties/@id",
          "type": "string"
        }
      }
    }
  },
  "@graph": {
    "$id": "#/properties/@graph",
    "type": "array",
    "items": {
      "$id": "#/properties/@graph/items",
      "type": "object",
      "properties": {
        "@id": {
          "$id": "#/properties/@graph/items/properties/@id",
          "type": "string"
        },
        "@type": {
          "$id": "#/properties/@graph/items/properties/@type",
          "type": "string"
        }
      }
    }
  }
}
```

Based on the JSON Schema and the ontology concepts, the following code depicts an example of JSON-LD interface.

```
{
  "@context": {
    "core": "http://delta.linkeddata.es/def/core#",
    "saref": "https://w3id.org/saref#",
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "VirtualNode": {
      "@id": "core:VirtualNode"
    },
    "Transaction": {
      "@id": "core:Transaction"
    }
  },
  "@graph": [
    {
      "@id": "core:Transaction01",
      "@type": "Transaction"
    },
    {
      "@id": "core:VirtualNode01",
      "@type": "VirtualNode",
      "hasTransaction": {
        "@id": "core:Transaction01"
      }
    }
  ]
}
```

```
}  
}  
]  
}
```

A-21. Voltage & Frequency

The following JSON Schema describes the structure of the data that need to be contained in the interface.

```
{  
  "$schema": "http://json-schema.org/draft-07/schema#",  
  "$id": "http://example.com/root.json",  
  "type": "object",  
  "properties": {  
    "@context": {  
      "$id": "#/properties/@context",  
      "type": "object",  
      "properties": {  
        "core": {  
          "$id": "#/properties/@context/properties/core",  
          "type": "string"  
        },  
        "om": {  
          "$id": "#/properties/@context/properties/om",  
          "type": "string"  
        },  
        "saref": {  
          "$id": "#/properties/@context/properties/saref",  
          "type": "string"  
        },  
        "xsd": {  
          "$id": "#/properties/@context/properties/xsd",  
          "type": "string"  
        },  
        "Voltage": {  
          "$id": "#/properties/@context/properties/Voltage",  
          "type": "object",  
          "properties": {  
            "@id": {  
              "$id":  
                "#/properties/@context/properties/Voltage/properties/@id",  
              "type": "string"  
            }  
          }  
        },  
        "Frequency": {  
          "$id": "#/properties/@context/properties/Frequency",  
          "type": "object",  
          "properties": {  
            "@id": {  
              "$id":  
                "#/properties/@context/properties/Frequency/properties/@id",  
              "type": "string"  
            }  
          }  
        },  
        "VirtualNode": {  
          "$id": "#/properties/@context/properties/VirtualNode",  
          "type": "object",  
          "properties": {  
            "@id": {
```



```
        "$id":
"#/properties/@context/properties/VirtualNode/properties/@id",
        "type": "string"
    }
}
},
"manage": {
    "$id": "#/properties/@context/properties/manage",
    "type": "object",
    "properties": {
        "@id": {
            "$id":
"#/properties/@context/properties/manage/properties/@id",
            "type": "string"
        }
    }
},
"Measurement": {
    "$id": "#/properties/@context/properties/Measurement",
    "type": "object",
    "properties": {
        "@id": {
            "$id":
"#/properties/@context/properties/Measurement/properties/@id",
            "type": "string"
        }
    }
},
"Energy": {
    "$id": "#/properties/@context/properties/Energy",
    "type": "object",
    "properties": {
        "@id": {
            "$id":
"#/properties/@context/properties/Energy/properties/@id",
            "type": "string"
        }
    }
},
"isRelatedtoMeasurement": {
    "$id": "#/properties/@context/properties/isRelatedtoMeasurement",
    "type": "object",
    "properties": {
        "@id": {
            "$id":
"#/properties/@context/properties/isRelatedtoMeasurement/properties/@id",
            "type": "string"
        }
    }
},
"measuresProperty": {
    "$id": "#/properties/@context/properties/measuresProperty",
    "type": "object",
    "properties": {
        "@id": {
            "$id":
"#/properties/@context/properties/measuresProperty/properties/@id",
            "type": "string"
        }
    }
}
}
```

```
    }
  },
  "@graph": {
    "$id": "#/properties/@graph",
    "type": "array",
    "items": {
      "$id": "#/properties/@graph/items",
      "type": "object",
      "properties": {
        "@id": {
          "$id": "#/properties/@graph/items/properties/@id",
          "type": "string"
        },
        "@type": {
          "$id": "#/properties/@graph/items/properties/@type",
          "type": "string"
        },
        "belongsTo": {
          "$id": "#/properties/@graph/items/properties/belongsTo",
          "type": "object",
          "properties": {
            "@id": {
              "$id":
                "#/properties/@graph/items/properties/belongsTo/properties/@id",
              "type": "string"
            }
          }
        }
      }
    }
  }
}
```

Based on the JSON Schema and the ontology concepts, the following code depicts an example of JSON-LD interface that exchange the voltage and frequency constraints.

```
{
  "@context": {
    "core": "http://delta.linkeddata.es/def/core#",
    "om": "http://www.wurvoc.org/vocabularies/om-1.8#",
    "saref": "https://w3id.org/saref#",
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "Voltage": {
      "@id": "core:Voltage"
    },
    "Frequency": {
      "@id": "core:Frequency"
    },
    "VirtualNode": {
      "@id": "core:VirtualNode"
    },
    "manage": {
      "@id": "core:manage"
    },
    "Measurement": {
      "@id": "core:Measurement"
    },
    "Energy": {
      "@id": "core:Energy"
    },
    "isRelatedtoMeasurement": {
      "@id": "core:isRelatedtoMeasurement"
    }
  }
}
```

```
    },
    "measuresProperty": {
      "@id": "core:measuresProperty"
    }
  },
  "@graph": [
    {
      "@id": "core:FEID01",
      "@type": "FEID",
      "belongsTo": {
        "@id": "core:Device1"
      }
    },
    {
      "@id": "core: Measurement1",
      "@type": "Measurement",
      "hasValue": {
        "@type": "xsd:float",
        "@value": "16.2"
      },
      "saref:isMeasuredIn": {
        "@id": "om:volt"
      }
    },
    {
      "@id": "core: Measurement2",
      "@type": " Measurement ",
      "hasValue": {
        "@type": "xsd:float",
        "@value": "31.2"
      },
      "saref:isMeasuredIn": {
        "@id": "om:hertz"
      }
    },
    {
      "@id": "core:Voltage01",
      "@type":
        "Voltage",
      "isRelatedToMeasurement": {
        "@id": "core:Measurement01"
      }
    },
    {
      "@id": "core:Frequency01",
      "@type": "Frequency",
      "isRelatedToMeasurement": {
        "@id": "core:Measurement02"
      }
    },
    {
      "@id": "core:Device1",
      "@type": "Device",
      "measuresProperty":
        {
          "@id": "core:Energy01"
        }
    }
  ]
}
```

