PARTHENOS

Pooling Activities, Resources and Tools
for Heritage E-research Networking,
Optimization and Synergies

D6.3 Report on the implementation of the Joint Resource Registry (interim)

AUTHORS: Luca Frosini
Massimiliano Assante
Andrea Dell'Amico
Lucio Lelii
Leonardo Candela
Pasquale Pagano

DATE   27 November 2017

HORIZON 2020 - INFRADEV-4-2014/2015:

Grant Agreement No. 654119

PARTHENOS
Pooling Activities, Resources and Tools for Heritage E-research Networking, Optimization and Synergies

Report on the implementation of the Joint Resource Registry (interim)

Deliverable Number	D6.3
Dissemination Level	Public
Delivery date	27 November 2017
Status	Final
Author(s)	Luca Frosini Massimiliano Assante Andrea Dell'Amico Lucio Lelii Leonardo Candela Pasquale Pagano

Project Acronym	PARTHENOS
Project Full title	Pooling Activities, Resources and Tools for Heritage E-research Networking, Optimization and Synergies
Grant Agreement nr.	654119

Deliverable/Document Information

Deliverable nr./title	D6.3
Document title	Report on the implementation of the Joint Resource Registry (interim)
Author(s)	Luca Frosini Massimiliano Assante Andrea Dell'Amico Lucio Lelii Leonardo Candela Pasquale Pagano
Dissemination level/distribution	Public

Document History

Version/date	Changes/approval	Author/Approved by
V 0.1 04.10.17	Section 2 and Section 3	Luca Frosini, Massimiliano Assante
V 0.2 12.10.17	Section 2 Revision, Section 4	Luca Frosini, Lucio Lelii
V 0.3 18.10.17	Section 4 Revision, Section 5.1 and Section 5.2	Luca Frosini, Lucio Lelii, Andrea dell'Amico
V 0.4 27.10.17	Section 5, Section 6	Luca Frosini
Final 17.11.17	Revision of all sections	Leonardo Candela, Pasquale Pagano
22/11/2017	Reviewed	Sheena Bassett, PIN

Table of Contents

List of Tables

1 Executive Summary

The *D6.3 Report on the implementation of the Joint Resource Registry* documents the interim implementation of the Joint Resource Registry (JRR). It complements the *D5.2 Report on the design of the Joint Resource Registry* deliverable by providing details on how to interact with and exploit the functionalities it provides.

The JRR hosts the PARTHENOS entities represented according to the PARTHENOS Entities Model defined in WP5. As such, it represents an information system for the PARTHENOS community and the PARTHENOS universe of tools and services designed for and released in the PARTHENOS infrastructure.

This deliverable presents, in Section 2, the principles and guidelines that govern the implementation of the JRR which has been designed to support the persistence of the PARTHENOS Entities. The JRR is implemented as a tailored information system capable of satisfying the evolution of the model itself, the main features of which are described in section 3 since the facet based resource model is extensively referred to throughout this report. Entities, Resources, Facets and Relations are described in detail. Section 4 describes the set of technical components comprising the JRR and APIs, covering the architecture which includes the Resource Registry Service, Context Client, Schema Client, Publisher and Client. Section 5 provides information regarding how to interact with the Resource Registry Service by exploiting the Context and Schema Port Types. The REST APIs are also presented for each functionality. Section 6 covers the backend database, OrientDB, a Multi-Model Open Source NoSQL DBMS that brings together the power of graphs and the flexibility of documents into one scalable high-performance operational database. The final section provides information on the Studio GUI used by the Content Administrator for searching between the created types and inspection of their schema.

2 Introduction

The Joint Resource Registry (JRR) has been designed to support the persistence of the PARTHENOS Entities. It is implemented as a tailored information system capable of satisfying the evolution of the model itself. Moreover, it contributes to the large gCube open-source framework as presented in the deliverable D6.1 PARTHENOS Cloud Infrastructure. In this Section, the role of this tailored information system is first clarified and then the functional and non-functional requirements are illustrated.

2.1 Definition

Several definitions of Information System (IS) exist. Each definition aims to capture either a specific role or a specific behaviour in systems managing some kind of information. It is quite common to define an IS as *"any organized system for the collection, organization, storage and communication of information"*. The Encyclopaedia Britannica defines an IS as *"an integrated set of components for **collecting**, **storing**, and **processing data** and for **providing information**, knowledge, and digital products"*.

All the definitions convey the characteristics of Information. Information consists of data that:
- is **accurate** and **timely**,
- is specific and **organized for a purpose**,
- is presented **within a context** that gives it meaning and relevance,
- can increase understanding and **decrease uncertainty.**

According to the Business Dictionary, an information system is *"a combination of hardware, software, infrastructure and **trained personnel** organized to facilitate planning, control, coordination, and **decision making in an organization"**.*
In this context, trained personnel are illustrated as:
- human resources
- procedures for using, operating, and maintaining the information system
- set of basic principles and associated guidelines, a.k.a. policies, formulated and enforced to direct and limit actions in pursuit of long-term goals.

Looking at the MIT Press, an information system is *"a software system to capture, transmit, store, retrieve, and manipulate data **produced by software systems** to provide access to information, thereby supporting people, organizations, or **other software systems"**.*
This definition makes it evident that software systems are both producers and consumers of the Information System making it the core of their business activities.

In the context of the research infrastructures[1] and system of systems, we can define an information system (IS) as:

---

[1] *The term 'research infrastructures' refers to facilities, resources and related services used by the scientific community to conduct top-level research in their respective fields, ranging from social sciences to astronomy, genomics to nanotechnologies* https://ec.europa.eu/research/infrastructures/index_en.cfm?pg=about

A software system

- to capture, transmit, store, retrieve, and manipulate data **produced by software systems**
- to provide access to information - *organized for a purpose and within a contextual domain* - that are used, accessed, and maintained according to **well-known procedures** operated under the limit of the (evolving) **organization policies**
- to support people within an organization and **other software systems.**

2.2 Requirements

The analysis of the requirements of an information system capable of providing support for a Research Infrastructure led to identification of the functionality the system has to provide (functional requirements) and the constraints and performances it has to respect (non-functional requirements).

2.2.1 Functional Requirements

IEEE has defined **Functional Requirements** as "*A requirement that specifies a function that a system or system component must be able to perform*"[2]
According to this definition, the following requirements have been identified:

- **Data Definition Language** (**DDL**) for schemas definition (entities and relations);
- **Entity and Relation** instances must be:
  - Univocally identifiable;
  - Selective/Partial updatable;
  - Validated against the Schema.
- **Referential Integrity** is a property of data stating references within it are valid[3]. A referential integrity constraint is defined as part of an association between two entity types. The purpose of referential integrity constraints is to ensure that valid associations always exist[4];
- **Dynamic Query** (no pre-defined query): Capabilities of a system allowing clients to build their own query and submit it to the system with no long-term impact on the information system. With regard to relational databases, this characteristic seems obvious (provided by SQL). Unfortunately, especially with the new trend of NoSQL, this same functionality is not supported by some types of NoSQL databases;
- **Standard Abstraction** (desiderata) as far as the relational databases respect SQL standard dialect, it is a desiderata that the information system supports a standard family of query language;

---

[2] IEEE (1990). Standard Glossary of Software Engineering Terminology. IEEE Standard 610.12-1990.
[3] https://en.wikipedia.org/wiki/Referential_integrity
[4] https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/referential-integrity-constraint

- **Subscription Notification** support allows *"full decoupling of the communicating entities in time, space, and synchronization"* [5] which reflect the nature of loosely coupled nature of distributed interaction in large-scale applications (such as a Research Infrastructure). By providing this functionality, the possibility to construct event-based services and to improve the scalability of the system will be ensured.

2.2.2 Non-Functional Requirements

Commonly Non-Functional Requirements are identified as *"requirements that specify criteria that can be used to judge the operation of a system, rather than specific behaviours"*[6]. Unfortunately, there is no consensus in the scientific community on a non-functional requirements definition. Martin Glinz [7] has defined taxonomy to identify non-functional requirements. In particular, a non-functional requirement can be:

- An attribute: is a performance requirement or a specific quality requirement;
  - A performance requirement is a requirement that pertains to a performance concern;
  - A specific quality requirement is a requirement that pertains to a quality concern other than the quality of meeting the functional requirements.
- A constraint: is a requirement that constrains the solution space beyond what is necessary for meeting the given functional, performance, and specific quality requirements.

Under the above-mentioned definition and the taxonomy fall:
- **High Availability (HA)** is a characteristic of a system, which aims to ensure an agreed level of operational performance, usually uptime, for a higher than normal period [8];
- **Eventual Consistency** is a consistency model used in distributed computing to achieve high availability that informally guarantees that, if no new updates are made to a given data item, eventually all accesses to that item will return the last updated value [9] ;
- **Horizontal Scalability**. Scalability is the capability of a system, network, or process to handle a growing amount of work, or its potential to be enlarged to accommodate that growth [10]. Horizontally scalability (or scale out/in) means adding more nodes to

---

[5] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. 2003. The many faces of publish/subscribe. ACM Comput. Surv. 35, 2 (June 2003), 114-131. DOI=http://dx.doi.org/10.1145/857076.857078

[6] https://en.wikipedia.org/wiki/Non-functional_requirement

[7] M. Glinz. On non-functional requirements. In Proc. 15th IEEE Int. Requirements Eng. Conf., 2007.

[8] https://en.wikipedia.org/wiki/High_availability

[9] Werner Vogels. 2009. Eventually consistent. Commun. ACM 52, 1 (January 2009), 40-44. DOI: https://doi.org/10.1145/1435417.1435432

[10] André B. Bondi. 2000. Characteristics of scalability and their impact on performance. In Proceedings of the 2nd international workshop on Software and performance (WOSP '00). ACM, New York, NY, USA, 195-203. DOI=http://dx.doi.org/10.1145/350391.350432

(or remove nodes from) a system, such as adding a new computer to a distributed software application.

- **Multi-Tenancy**, i.e. a single instance of the technology should be able to serve many "independent" contexts (between the same Application Domain) [11];
- **EUPL** license **compatibility** of all its components.

2.3 Architecture

The architecture of this information system comprises several components. It includes the software components dealing with the generic and the tailored entities models; the services components implementing the capabilities to interact with those entities; the backend database used to persist the entities; and finally the graphical user interface oriented for human exploitation and visualization of the entities.

The architecture of the information system is, therefore, composed of the following software components:

- Facet Based Resource Model libraries
  - Information System Model library
  - gCube Model library
  - PARTHENOS Model library

- Joint Resource Registry
  - Resource Registry Service
  - Resource Registry Context Client
  - Resource Registry Schema Client
  - Resource Registry Publisher
  - Resource Registry Client
- Backend Database (i.e. OrientDB as Graph Database)
- Information System Subscription Notification Service
- Graphical User Interface (GUI)

---

[11] Please note that different Application domain must be managed by completely separated instances of the whole IS.

3 Facet Based Resource Model

The PARTHENOS Joint Resource Registry Data Model is extensively presented in Section 6 of the deliverable *D5.2 Design of the Joint Resource Registry*. In the following sections, some basic information about the Resource Model is reported since this is largely used in the remaining part of this document.

3.1 Information System Model

3.1.1 Basic Concept

Two typologies of **Entities** are envisaged:
- **Resources**, i.e. entities representing a description of "thing" to be managed;
  - Every Resource is characterized by a number of Facets.
- **Facets**, i.e. entities contributing to "build" a description of a Resource. Every facet, once attached to a Resource profile, captures a certain aspect / characterization of the resource. Every facet is characterized by a number of properties;

Two typologies of **Relations** are envisaged:
- **isRelatedTo**, i.e. a relation linking any two Resources.
- **consistsOf**, i.e. a relation connecting each Resource with one of the Facets characterizing it;
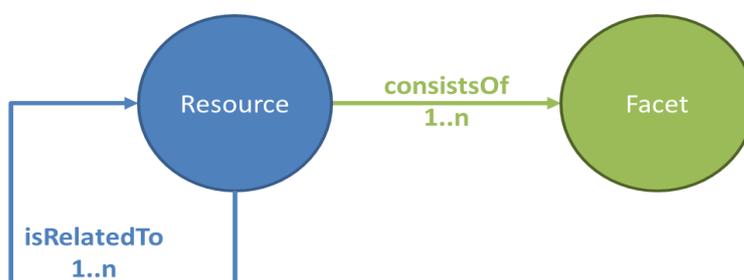
Each **Entity** and **Relation**
- has an **header** automatically generated for the sake of identification and provenance of the specific information;
- can be **specialized**
  - A number of specializations are identified below. Such specializations are managed by the gCube Core services, i.e. Core services builds upon these specializations to realize its management tasks;
  - Other specializations can be defined by clients, the system make it possible to store these additional typologies of relations and facets and to discover them.

Facet and **Relation** instances can have additional properties which are not defined in the schema (henceforth schema-mixed mode).
Relation properties:
- Any relation has a direction, i.e. a "source" (**out** bound of the relation) and a "target" (**in** bound of the relation). Anyway, the relation can be also navigated in the opposite direction;
- It is not permitted to define a **Relation** having a **Facet** as "source". In other words:
  - It is not permitted to define a **Relation** connecting a **Facet** with another one;
  - It is not permitted to define a **Relation** connecting a **Facet** with a **Resource** (as target);
- A **Facet** instance can be linked (by **consistsOf** or any specialization of it) from different **Resources**.

Any Property can be enriched with the following attributes:
- **Name**: Property Name
- **Type**: The Type of the Property (e.g. String, Integer, ...). See **Property Type**
- **Description**: The description of the Property. Default=null.
- **Mandatory** (**M**): Indicates if the Property is mandatory. Default=false.
- **ReadOnly** (**RO**): The Property cannot change its value. Default=false.
- **NotNull** (**NN**): Whether the property must assume a value diverse from 'null' or not. Default=false
- **Max** (**Max**): Default=null
- **Min** (**Min**): Default=null
- **Regexpr** (**Reg**)): A Regular Expression to validate the property value, default=null.

Table 1: Basic Property Types

Type	Java type	Description
Boolean	java.lang.Boolean or boolean	Handles only the values *True* or *False*.
Integer	java.lang.Integer or int or java.math.BigInteger	32-bit signed Integers.
Short	java.lang.Short or short	Small 16-bit signed integers.
Long	java.lang.Long or long	Big 64-bit signed integers.
Float	java.lang.Float or float	Decimal numbers.
Double	java.lang.Double or double	Decimal numbers with high precision.
Date	java.util.Date	Any date with the precision up to milliseconds.
String	java.lang.String	Any string as alphanumeric sequence of chars.
Embedded	? extends org.gcube.informationsystem.model.embedded.Embedded	This is an Object contained inside the owner Entity and has no Header. It is reachable only by navigating the owner Entity.
Embedded list	List<? extends org.gcube.informationsystem.model.embedded.Embedded>	List of Objects contained inside the owner Entity and have no Header. They are reachable only by navigating the owner Entity.
Embedded set	Set<? extends org.gcube.informationsystem.model.embedded.Embedded>	Set (no duplicates) of Objects contained inside the owner Entity and have no Header. They are reachable only by navigating the owner Entity.
Embedded map	Map<String, ? extends org.gcube.informationsystem.model.embedded.Embedded>	Map of Objects contained inside the owner Entity and have no Header. They are reachable only by navigating the owner Entity.
Byte	java.lang.Byte or byte	Single byte. Useful to store small 8-bit signed integers.
Binary	java.lang.Byte[] or byte[]	Can contain any value as byte array.

Table 2: Derived Property Types

Type	Java type	Description
Enum	java.lang.Enum or enum	By default, it is represented using the String representation of the Enum so that the primitive type used will be String. The enumeration is checked by setting the Regexpr property. The Regular Expression is auto-generated and it will be something like ^(FIRST-ENUM-STRING_REPRESENTATION|SECOND-ENUM-STRING_REPRESENTATION|...|LAST_ENUM_STRING_REPRESENTATION)$. Otherwise (if indicated using an annotation), it can be represented using the Integer value of the Enum so that the primitive type used will be Integer. The enumeration is checked using Max and Min properties.
UUID	java.util.UUID	String representation of the UUID. The check is obtained using the regular expression ^([a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}){1}$
URL	java.net.URL	String representation of the URL. No check actually.
URI	java.net.URI	String representation of the URI. No check actually.

Table 3: Header

Name	Type	Attributes	Description
uuid	UUID	Mandatory=true NotNull=true ReadOnly=true	This uuid can be used to univocally identify the Entity or the Relation
creator	String	Mandatory=true NotNull=true ReadOnly=true	Filled at creation time. The creator is retrieved using the authorization token
creationTime	Date	Mandatory=true NotNull=true ReadOnly=true	Creation time in milliseconds. Represent the difference, measured in milliseconds, between the creation time and midnight, January 1, 1970 UTC
lastUpdateTime	Date	Mandatory=true NotNull=true	Last Update time in milliseconds. Represent the difference, measured in milliseconds, between the last update time and midnight, January 1, 1970 UTC

Table 4: Propagation Constraints

Name	Type	Attributes	Description
remove	Enum	Mandatory=true NotNull=true Regex=(cascadeWhenOrphan|cascade|keep)	Indicate the behaviour to Resource Registry to be applied to the target Entity when the source Entity is remove from context or deleted
add	Enum	Mandatory=true NotNull=true Regex=(propagate|unpropagate)	Indicate the behaviour to Resource Registry to be applied to the target Entity when the source Entity is added to Context

Any Relation contains such a property. If the values are not specified at creation time, the system will initialize it following the following rules:

- IsRelatedTo Relation: remove=keep, add=unpropagate
- ConsistsOf Relation: remove=cascadeWhenOrphan, add=propagate

3.1.2 Entity

The resource entity is conceived to describe every "main thing" to be registered in and discovered through the Joint Resource Registry.

Table 5: Resource Entity

Scope:				
Source	**Relation**	**Multiplicity**	**Target**	**Description**
Facets				
Resource	isIdentifiedBy	1..n	Facet	Any Resource has at least one Facet which in some way allow to identify the Resource per se.
Resource	consistsOf	0..n	Facet	Any Resource consist of zero or more Facets which describes the different aspects of the facet.
Relations				
Resource	isRelatedTo	0..n	Resource	Any Resource can be related to any other resource.

3.1.3 Facet

Facets are collections of attributes conceived to capture a certain feature / aspect of the resource they are associated with.
Every Facet has:
- A Header automatically generated to capture identification- and provenance-related aspects of the facet once it is instantiated;
- Zero or more properties. Besides the per-facet envisaged properties, clients can add new ones.

3.1.4 Relation

Every relation has:
- A Header
- A PropagationConstraint
- Zero or More properties (not necessarily predefined, similarly to Facets).

Table 6: isRelatedTo

Source	Relation	Multiplicity	Target	Description
Resource	isRelatedTo	0..n	Resource	A relation linking any two Resources.

Default PropagationConstraint has the following values: remove=keep, add=unpropagate

Table 7: consistOf

Source	Relation	Multiplicity	Target	Description
Resource	consistsOf	1..n	Facet	A relation connecting each Resource with one of the Facet characterizing it.

Default PropagationConstraint has the following values: remove=cascadeWhenOrphan, add=propagate

Table 8: isIdentifiedBy

Source	Relation	Multiplicity	Target	Description
Definition				
Resource	isIdentifiedBy	1..n	Facet	A relation connecting each Resource with one of the Facet which can be used to identify the Resource.

4 Joint Resource Registry

The **Joint Resource Registry** is designed to support the following operations:
- To capture, transmit, store, retrieve and manipulate data from any software system enabled on the infrastructure, including:
  - Location and properties
  - Status, load, exploitation usage, and accounting data
- To provide access to information, organized to enable:
  - Monitoring, validation, and reporting
  - Elasticity and pooling of resources
- To support any software system to:
  - Discover services and infrastructure resources.

The Joint Resource Registry enables:
- **a set of resource management functions**
  - enabling functions
    - publication, discovery
    - monitoring, deployment
    - contextualization, security, execution
  - data management functions
    - access, store
    - index, search
    - transfer, transform
- **a set of applications**
  - built around those functions
- **an abstract view over functions**
  - defined by specifications
  - multiple implementations, over time / concurrently
- **secure and consistent entities evolution**
  - tailored support for facet and resource definition
  - implementations produce/consume different facets, independently
- **dynamic resource semantics**
  - no longer predefined in class hierarchies
  - implicitly captured by current facets
  - changes over time / across "similar" resources

4.1 Architecture

The constituent software components are:
- Resource Registry Service
- Resource Registry Context Client
- Resource Registry Schema Client
- Resource Registry Publisher
- Resource Registry Client

4.1.1 Resource Registry Service

The Resource Registry Service is a web service running on SmartGears responsible for storing information, in particular the global and partial view of:
- the resources (e.g. computing, storage, services, software, datasets);
- their current status (e.g. up and running, available);
- their relationships with other resources;
- the policies governing their exploitation.

The Resource Registry is developed only by using the concepts defined in the Information System Model and it provides the capabilities to enrich its knowledge by creating new types of entities and relations and their schemas. The Resource Registry is capable of serving different applications domains (i.e. Context). To achieve this goal the Resource Registry provides capabilities for managing Contexts (the contexts are hierarchical) and associating the entities and relations to one or more of the Contexts as requested by the different clients. The Resource Registry is also responsible for notifying any update to or creation of any entity or relation to **Information System Subscription Notification Service**.

To reach its goals, the Resource Registry offers four port types:
- **Context Management**: manage hierarchical Context;
- **Schema Management**: register and define Entities and Relations schema;
- **Entities and Relations Instances Management**: manage instances of registered Entity and Relation type;
- **Query & Access**: query instances and get the schema definition of registered types.

Every Port type is exposed as REST[12] API.
Every REST API is JSON[13] based. This means that any content present in an HTTP request is formatted using the JSON standard.

4.1.2 Resource Registry Context Client

The Resource Registry Context Client is a java library providing RPC facilities to interact with the **Context Management** port type. The library hides all the complexity of marshalling and un-marshalling of requests and results. By using this library, any client can manage java classes instead of JSON objects.

4.1.3 Resource Registry Schema Client

The Resource Registry Schema Client is a java library providing RPC facilities to interact with the **Schema Management** port type. The library hides all the complexity of marshalling and un-marshalling of requests and results. By using this library, any client can manage java classes instead of JSON objects.

---

[12] https://en.wikipedia.org/wiki/Representational_state_transfer
[13] https://en.wikipedia.org/wiki/JSON

4.1.4 Resource Registry Publisher

The Resource Registry Publisher is a java library providing RPC facilities to interact with **the Entities and Relations Instances Management** port type. The library hides all the complexity of marshalling and un-marshalling of requests and result. By using this library any client can manage java classes instead of JSON objects.

4.1.5 Resource Registry Client

The Resource Registry Client is a java library providing RPC facilities to interact with the **Query & Access** port type. The library hides all the complexity of marshalling and un-marshalling of requests and result. By using this library any client manages java classes instead of JSON objects.

5 Interacting with Resource Registry Service

This section provides information regarding how to interact with the Resource Registry Service by exploiting the Context and Schema Port Types. The REST APIs are also presented for each functionality. Please note that the provided examples can intentionally hide some details in the response to avoid unneeded complexity.

5.1 Context Management

It is responsible for managing Context belonging to the same Application Domain. The security configuration based on the Authorization Framework makes this port type accessible only from the Resource Manager. In other words, no others client is allowed to manage Context other than the Resource Manager. See *D6.1 PARTHENOS Cloud infrastructure* for details about the Resource Manager and the Authorization Framework. Context requirements are:
- No predefined number of levels.
- Possibility to change the name of the Context with no impact for any component.
- Possibility to move a Context from a parent Context to another.

Available Methods:
- Create
- Read
- Rename
- Move
- Delete

Any action made to Contexts succeeds if the following requirements are guaranteed:
- Two Contexts with same name can exist but only if they have different parents. The operation which will try to obtain a Context with the same name to the same parent will fail with no effect.
- Any operation made in any Context has an effect on only the Context. In other words, there will be no effect on the associated Entity and Relations.

At time of writing this document, this port type is only accessible by using REST API. The Resource Registry Context Client (java client) is under testing and validation.

5.1.1 Create

Create new Context as child of another Context (if any).
PUT /resource-registry/context

Table 9: Create Request Parameters

Name	Type	Required	Description
name	String	true	The name of the context.
parentContextId	String (UUID)	false	The UUID of the parent Context if any

Table 10: Create Response Type

Code	Type	Description
200	String	The JSON representation of the Context.
400	String	HTTP error code

Example 1
Create a new Context with named **gcube** with no parent. It is a ROOT Context.
Request URL
PUT /resource-registry/context?name=gcube

Response Body
```
{
        "@class": "Context",
        "name": "gcube",
        "header": {
                "@class": "Header",
                "uuid": "2705dd32-c857-444b-818a-3ec69e339e5d",
                "creator": "luca.frosini",
                "modifiedBy": "luca.frosini",
                "creationTime": "2017-03-1711:47:55",
                "lastUpdateTime": "2017-03-17 11:47:55"
        }
}
```

Example 2
Create a new Context with named **devsec** as child of Context with UUID **2705dd32-c857-444b-818a-3ec69e339e5d (gcube)**

Request URL
PUT /resource-registry/context?name=devsec&parentContextId=2705dd32-c857-444b-818a-3ec69e339e5d
Response Body

```
        "@class": "Context",
        "name": "devsec",
        "header": {
                "@class": "Header",
                "uuid": "30f6254c-c87a-451e-bc0f-7cfcbd94a84a",
                "creator": "luca.frosini",
                "modifiedBy": "luca.frosini",
                "creationTime": "2017-03-17 11:47:56",
                "lastUpdateTime": "2017-03-17 11:47:56"
        }
}
```

Example 3
Create a new Context with named **devVRE** as child of Context with UUID **30f6254c-c87a-451e-bc0f-7cfcbd94a84a (devsec)**
Request URL
PUT /resource-registry/context?name=devVRE&parentContextId=30f6254c-c87a-451e-bc0f-7cfcbd94a84a
Response Body
```
{
        "@class": "Context",
        "name": "devVRE",
        "header": {
                "@class": "Header",
                "uuid": "9d73d3bd-1873-490c-b0a7-e3c0da11ad52",
                "modifiedBy": "luca.frosini",
                "creator": "luca.frosini",
                "creationTime": "2017-03-17 11:47:57",
                "lastUpdateTime": "2017-03-17 11:47:57"
        }
}
```

Example 4
If you try to create again a Context named **gcube** without specifying the parent a **400 Bad Request** HTTP error is reported.
Request URL
PUT /resource-registry/context?name=gcube
Response Body
```
{
    "@class": "ContextCreationException",
    "message": "A root context with the same name (gcube) already exist"
}
```

This will also happen anytime a Context with the same name, having the same parent, of an existing Context is requested to be created.

5.1.2 Read

Return the definition of the Context identified by the UUID provided as path parameter.
Request URL
GET /resource-registry/context/{UUID}

Table 11: Read Request Parameters

Name	Type	Required	Description
UUID	String (UUID)	true	The UUID of the target context.

Table 12: Read Response Type

Code	Type	Description
200	String	The JSON representation of the context.
400	String	HTTP error code

Examples

Read the Context having UUID **9d73d3bd-1873-490c-b0a7-e3c0da11ad52**

Request URL

GET /resource-registry/context/9d73d3bd-1873-490c-b0a7-e3c0da11ad52

Response Body

```
{
        "@class": "Context",
        "name": "devVRE",
        "header": {
                "@class": "Header",
                "uuid": "9d73d3bd-1873-490c-b0a7-e3c0da11ad52",
                "creator": "luca.frosini",
                "modifiedBy": "luca.frosini",
                "creationTime": "2017-03-17 11:47:56",
                "lastUpdateTime": "2017-03-17 11:52:56"
        }
}
```

5.1.3 Rename

Rename a Context identified by the UUID provided as path parameter to the new name provided as query parameter.

Request URL

POST /resource-registry/context/rename/{UUID}

Table 13: Rename Request Parameters

Name	Type	Required	Description
UUID	String (UUID)	true	The UUID of the target context.
name	String	true	The new name of the target context.

Table 14: Rename Response Type

Code	Type	Description
200	String	The JSON representation of the context.
400	String	HTTP error code

Example

Rename a Context **9d73d3bd-1873-490c-b0a7-e3c0da11ad52** (was **devVRE**) to the new name **devNext**.

Request URL

POST /resource-registry/context/rename/9d73d3bd-1873-490c-b0a7-e3c0da11ad52?name=devNext

Response Body

```
{
        "@class": "Context",
        "name": "devNext",
        "header": {
                "@class": "Header",
                "uuid": "9d73d3bd-1873-490c-b0a7-e3c0da11ad52",
                "creator": "luca.frosini",
                "modifiedBy": "luca.frosini",
                "creationTime": "2017-03-17 11:47:56",
                "lastUpdateTime": "2017-03-17 11:52:56"
        }
}
```

5.1.4 Move

Move a Context identified by the UUID provided as path parameter as child of the Context provided as query parameter.

Request URL

POST /resource-registry/context/move/{UUID}

Table 15: Move Request Parameters

Name	Type	Required	Description
UUID	String (UUID)	true	The UUID of the target Context
parentContextId	String (UUID)	true	The parent Context UUID

Table 16: Move Response Type

Code	Type	Description
200	String	The JSON representation of the context.
400	String	HTTP error code

Example

Rename a Context **9d73d3bd-1873-490c-b0a7-e3c0da11ad52** as child of the Context **761d9e99-a4dc-4838-9b16-4bf73813b625**

Request URL

POST /resource-registry/context/move/9d73d3bd-1873-490c-b0a7-
e3c0da11ad52?parentContextId=761d9e99-a4dc-4838-9b16-4bf73813b625

Response Body

```
{
        "@class": "Context",
        "name": "devNext"
        "header": {
                "@class": "Header",
                "uuid": "9d73d3bd-1873-490c-b0a7-e3c0da11ad52",
                "creator": "luca.frosini",
                "modifiedBy": "luca.frosini",
                "creationTime": "2017-03-17 11:47:56",
                "lastUpdateTime": "2017-03-17 11:52:56"
```

```
        }
}
```

5.1.5 Delete

Delete the Context identified by the UUID provided as path parameter.
Request URL
DELETE /resource-registry/context/{{UUID}}

Table 17: Delete Request Parameter

Name	Type	Required	Description
UUID	String (UUID)	true	The UUID of the target Context

Table 18: Delete Response Type

Code	Type	Description
200	String	Empty content
400	String	HTTP error code

Example
Delete the Context having UUID **9d73d3bd-1873-490c-b0a7-e3c0da11ad52**
Request URL
DELETE /resource-registry/context/9d73d3bd-1873-490c-b0a7-e3c0da11ad52

5.2 Schema Management

This port type is only accessible by using REST API.

5.2.1 Type Definition

Any Type is described by the following attributes:
- **name** (String): Type Name
- **description** (String): The description of the Type. Default=null.
- **abstractType** (Boolean): Indicate if the type is abstract so that it cannot be instantiated. In other words, only subtypes of this type can be instantiated. Default=false.
- **superclasses** (List<String>): The list of all super types of this type. Multiple Inheritance is supported.
- Zero or more Properties

Property
Any Property is described by the following attributes:
- **name**: Property Name
- **type**: The Type of the Property (e.g. String, Integer, ...). See Property Type

- **description**: The description of the Property. Default=null.
- **mandatory**: Indicate if the Property is mandatory. Default=false.
- **readOnly**: The Property cannot change its value. Default=false.
- **notNull**: Whether the property must assume a value diverse from 'null' or not. Default=false
- **max**: Default=null
- **min**: Default=null
- **regexpr**: A Regular Expression[14] to validate the property value, default=null. A good online tool for regex is available at https://regex101.com.

Property Type are mapped to and integer to be used in property definition[15].
Table 19: Property Type Mapping

Type	Integer Mapping	Java type	Description
Boolean	0	java.lang.Boolean or boolean	Handles only the values *True* or *False*.
Integer	1	java.lang.Integer or int or java.math.BigInteger	32-bit signed Integers.
Short	2	java.lang.Short or short	Small 16-bit signed integers.
Long	3	java.lang.Long or long	Big 64-bit signed integers.
Float	4	java.lang.Float or float	Decimal numbers
Double	5	java.lang.Double or double	Decimal numbers with high precision.
Date	6	java.util.Date	Any date with the precision up to milliseconds.
String	7	java.lang.String	Any string as alphanumeric sequence of chars.
Binary	8	java.lang.Byte[] or byte[]	Can contain any value as byte array.
Embedded	9	? extends org.gcube.informationsystem.model.embedded.Embedded	This is an Object contained inside the owner Entity and has no Header. It is reachable only by navigating the owner Entity.
Embedded list	10	List<? extends org.gcube.informationsystem.model.embedded.Embedded>	List of Objects contained inside the owner Entity and have no Header. They are reachable only by navigating the owner Entity.
Embedded set	11	Set<? extends org.gcube.informationsystem.model.embedded.Embedded>	Set (no duplicates) of Objects contained inside the owner Entity and have no Header. They are reachable only by navigating the owner Entity.

---

[14] https://en.wikipedia.org/wiki/Regular_expression
[15] Type binder is defined in the class **Type** available at
http://svn,research-infrastructures.eu/public/d4science/gcube/trunk/information-system/information-system-model/src/main/java/org/gcube/informationsystem/types/Type.java

Embedded map	12	Map<String, ? extends org.gcube.informationsystem.model.embedded.Embedded>	Map of Objects contained inside the owner Entity and have no Header. They are reachable only by navigating the owner Entity.
Byte	17	java.lang.Byte or byte	Single byte. usesful to store small 8-bit signed integers.

5.2.2 Type Creation

Allow to create new Entity or Relation or Embedded Type.
Request URL
PUT /resource-registry/schema/{TypeName}

Table 20: Create Parameters

Name	Type	Required	Description
TypeName	String	true	The name of the new type to create

Table 21: Create Response Type

Code	Type	Description
200	String	The JSON representation of the newly created type (which is the same of the request)
400	String	HTTP error code

Example: Resource Type Creation
PUT /resource-registry/schema/Actor

Request Body
```
{
        "name": "Actor",
        "description": "Any entity (human or machine) playing an active role.",
        "abstractType": true, /* If the Resource cannot be instantiated */
        "superclasses": ["Resource"], /* Resource or any registered specialization. */
        "properties": null /* MUST be null. The Resource cannot have any property. */
}
```

Example: Facet Type Creation
PUT /resource-registry/schema/ContactFacet
Request Body
```
{
        "name": "ContactFacet",
        "description": "This facet is expected to capture contact information",
        "abstractType": false,
```

```
            "superclasses":["Facet"],
            "properties":[
                    {
                            "name": "name",
                            "description": "First Name",
                            "mandatory": true,
                            "readonly": false,
                            "notnull": true,
                            "max": null,
                            "min": null,
                            "regexpr": null,
                            "linkedType": null,
                            "linkedClass": null,
                            "type": 7 /* String*/
                    },{
                            "name": "eMail",
                            "description": "A restricted range of RFC-822 compliant email address. ... ",
                            "mandatory": true,
                            "readonly": false,
                            "notnull": true,
                            "max": null,
                            "min": null,
                            "regexpr":"^[a-z0-9._%+-]{1,128}@[a-z0-9.-]{1,128}$",
                            "linkedType": null,
                            "linkedClass": null,
                            "type":7 /* String */
                    }
            ]
}
```

Example: IsRelatedTo Type Creation

PUT /resource-registry/schema/Hosts

Request Body

```
{
        "name": "Hosts",
        "description": "…",
        "abstractType": false,
        "superclasses": ["IsRelatedTo"],
        "properties": []
}
```

Example: ConsistsOf Type Creation

PUT /resource-registry/schema/HasContact

Request Body

```
{
        "name": "HasContact",
        "description":"",
        "abstractType": true,
        "superclasses": ["ConsistsOf"],
        "properties": []
}
```

Example: Embedded Type Creation

PUT /resource-registry/schema/AccessPolicy

Request Body

```
{
        "name": "AccessPolicy",
        "description": "",
        "abstractType": false,
        "superclasses": ["Embedded"],
        "properties":[
                {
                        "name": "policy",
                        "description": "",
                        "mandatory": false,
```

```
                    "readonly": false,
                    "notnull": false,
                    "max": null,
                    "min": null,
                    "regexpr": null,
                    "linkedType": null,
                    "linkedClass": "ValueSchema",
                    "type": 9  /* Embedded */
            },{
                    "name": "note",
                    "description": "",
                    "mandatory": false,
                    "readonly": false,
                    "notnull": false,
                    "max": null,
                    "min": null,
                    "regexpr": null,
                    "linkedType": null,
                    "linkedClass": null,
                    "type":7 /* String */
            }
        ]
}
```

5.2.3 Read Type Definition

Allow to read Type Definition
Request URL
GET /resource-registry/schema/{TypeName}

Table 22 Read Parameters

Name	Type	Required	Description
TypeName	String	true	The name of the type you want to retrieve the definition

Table 23 Read Response Type

Code	Type	Description
200	String	The JSON representation of the newly created type
400	String	HTTP error code

Example: Resource Type
GET /resource-registry/schema/Actor
Response
```
{
        "name": "Actor",
        "description": "Any entity (human or machine) playing an active role.",
        "abstractType": true,
        "superclasses": ["Resource"],
        "properties": null
}
```

Example: Facet Type
GET /resource-registry/schema/ContactFacet
Response

```
{
        "name": "ContactFacet",
        "description": "This facet is expected to capture contact information",
        "abstractType": false,
        "superclasses": ["Facet"],
        "properties":[
                        {
                                "name": "name",
                                "description": "First Name",
                                "mandatory": true,
                                "readonly": false,
                                "notnull": true,
                                "max": null,
                                "min": null,
                                "regexpr": null,
                                "linkedType": null,
                                "linkedClass": null,
                                "type": 7 /* String*/
                        },
                {
                                "name": "eMail",
                                "description": "A restricted range of RFC-822 compliant email address. ... ",
                                "mandatory": true,
                                "readonly": false,
                                "notnull": true,
                                "max": null,
                                "min": null,
                                "regexpr": "^[a-z0-9._%+-]{1,128}@[a-z0-9.-]{1,128}$",
                                "linkedType": null,
                                "linkedClass": null,
                                "type": 7 /* String */
                }
        ]
}
```

Example: IsRelatedTo Type
GET /resource-registry/schema/Hosts
Response

```
{
        "name": "Hosts",
        "description": "…",
        "abstractType": false,
        "superclasses": ["IsRelatedTo"],
        "properties": []
}
```

Example: ConsistsOf Type
GET /resource-registry/schema/HasContact
Response

```
{
        "name": "HasContact",
        "description": "",
        "abstractType": true,
        "superclasses": ["ConsistsOf"],
        "properties": []
}
```

Example: Embedded Type
GET /resource-registry/schema/AccessPolicy
Response

```
{
        "name": "AccessPolicy",
        "description": "",
        "abstractType": false,
```

```
        "superclasses": ["Embedded"],
        "properties":[{
                "name": "policy",
                "description": "",
                "mandatory": false,
                "readonly": false,
                "notnull": false,
                "max": null,
                "min": null,
                "regexpr": null,
                "linkedType": null,
                "linkedClass": "ValueSchema",
                "type": 9  /* Embedded */
        },{
                "name": "note",
                "description": "",
                "mandatory": false,
                "readonly": false,
                "notnull": false,
                "max": null,
                "min": null,
                "regexpr": null,
                "linkedType": null,
                "linkedClass": null,
                "type": 7 /* String */
        }]
}
```

5.3 Entities and Relations Instances Management

This section provides information regarding how to interact with Resource Registry Service for Entities and Relations Management. REST and JAVA API are presented for each functionality. Please note that the provided examples can intentionally hide some details in the response to avoid unneeded complexity. Apart from the REST API, this port type can be used also by using Resource Registry Publisher java client.

Resource Registry Publisher has the following maven coordinates:
```
<dependency>
        <groupId>org.gcube.information-system</groupId>
        <artifactId>resource-registry-publisher</artifactId>
        <version>[1.0.0-SNAPSHOT, 2.0.0-SNAPSHOT)</version>
</dependency>
```

To use the client, you need first get a ResourceRegistryPublisher instance.
By using ResourceRegistryPublisherFactory.create() method the library discovers the correct endpoint to interact with the Resource Registry for the current context.
```
SecurityTokenProvider.instance.set("Your-Token-Here"); // If not already set
ResourceRegistryPublisher resourceRegistryPublisher = ResourceRegistryPublisherFactory.create();
```

5.4 Facet Instances APIs

5.4.1 Create Facet Instance

REST API
PUT /resource-registry/er/facet/{FacetType}

Example
PUT /resource-registry/er/facet/CPUFacet
Request Body
```
{
        "@class": "CPUFacet",
        "header": null,
```

```
        "model": "Opteron",
        "vendor": "AMD",
        "clockSpeed": "1 GHz"
}
```

Response Body
```
{
        "@class": "CPUFacet",
        "header": {
                "uuid": "69f0b376-38d2-4a85-bc63-37f9fa323f82",
                "creator": "luca.frosini",
                "modifiedBy": "luca.frosini",
                "creationTime": "2016-10-05 11:16:24",
                "lastUpdateTime": "2016-10-05 11:16:24
        },
        "model": "Opteron",
        "vendor": "AMD",
        "clockSpeed": "1 GHz"
}
```

Java API
public <F extends Facet> F createFacet(F facet) throws FacetAlreadyPresentException,
ResourceRegistryException;

Example
CPUFacet cpuFacet = new CPUFacetImpl();
cpuFacet.setClockSpeed("1 GHz");
cpuFacet.setModel("Opteron");
cpuFacet.setVendor("AMD");

CPUFacet createdCpuFacet = resourceRegistryPublisher.createFacet(cpuFacet);
UUID uuid = createdCpuFacet.getHeader().getUUID(); // 69f0b376-38d2-4a85-bc63-37f9fa323f82

Alternative JAVA API
There are also two other equivalent methods with the following signature:
public String createFacet(String facet) throws FacetAlreadyPresentException, ResourceRegistryException;

public String createFacet(String facetType, String facet) throws FacetAlreadyPresentException,
ResourceRegistryException;

The first method gets the Facet to be created as JSON string instead of as Java class. The
second gets also the **facetType** as parameter (which has to be specified as PATH
PARAMETER in the request) avoiding to force client to retrieve it from the string. The
second method is more efficient but you have to be sure that the **facetType** is the same
specified in the header of the serialized facet.

5.4.2 Update Facet Instance

REST API
POST /resource-registry/er/facet/{FacetInstanceUUID}

Example
POST /resource-registry/er/facet/69f0b376-38d2-4a85-bc63-37f9fa323f82

Request Body
```
{
        "@class": "CPUFacet",
        "header": { "uuid":"69f0b376-38d2-4a85-bc63-37f9fa323f82" }, /* if you pass the header only the
UUID is checked and must be the same of the one provided in the URL*/
        "model": "Opteron",
        "vendor": "AMD",
        "clockSpeed":"2 GHz"
}
```

Response Body

```
{
        "@class": "CPUFacet",
        "header": {
                "uuid": "69f0b376-38d2-4a85-bc63-37f9fa323f82",
                "creator": "luca.frosini",
                "modifiedBy": "luca.frosini",
                "creationTime": "2016-10-05 11:16:24",
                "lastUpdateTime": "2016-10-05 11:17:32"
        },
        "model": "Opteron",
        "vendor": "AMD",
        "clockSpeed": "2 GHz"
}
```

Java API
public <F extends Facet> F updateFacet(F facet) throws FacetNotFoundException,
ResourceRegistryException;

Example
createdCpuFacet.setClockSpeed("2 GHz");
CPUFacet updatedCpuFacet = resourceRegistryPublisher.updateFacet(createdCpuFacet);

Alternative JAVA API
There are also two other equivalent methods with the following signature:
public String updateFacet(String facet) throws FacetNotFoundException, ResourceRegistryException;
public String updateFacet(UUID uuid, String facet) throws FacetNotFoundException,
ResourceRegistryException;

The first method gets the Facet to be created as a JSON string instead of as a Java class.
The second get also the **uuid** as a parameter (which has to be specified as PATH
PARAMETER in the request) avoiding forcing the client to retrieve it from the string. The
second method is more efficient but you have to be sure that the provided **uuid** is the
same specified in the header of the serialized facet.

5.4.3 Delete Facet Instance

REST API
DELETE /resource-registry/er/facet/{FacetInstanceUUID}

Example
DELETE /resource-registry/er/facet/69f0b376-38d2-4a85-bc63-37f9fa323f82

Java API
public <F extends Facet> boolean deleteFacet(F facet) throws FacetNotFoundException,
ResourceRegistryException;

Example
boolean deleted = resourceRegistryPublisher.deleteFacet(createdCpuFacet);

Alternative JAVA API
There is also another equivalent method with the following signature:
public boolean deleteFacet(UUID uuid) throws FacetNotFoundException, ResourceRegistryException;

The method just requires the UUID of the Facet to be deleted.

5.5 Resource Instances APIs

5.5.1 Create Resource Instance

REST API
PUT /resource-registry/er/resource/{ResourceType}

Example
PUT /resource-registry/er/resource/HostingNode
Request Body
```
{
        "@class": "HostingNode",
        "consistsOf":[
                {
                        "@class": "ConsistsOf",
                        "target":{
                                "@class": "CPUFacet",
                                "model": "Opteron",
                                "vendor": "AMD",
                                "clockSpeed": "3 GHz"
                        }
                },{
                        "@class": "IsIdentifiedBy",
                        "target":{
                                "@class": "NetworkingFacet",
                                "header":{ "uuid":"59617b01-5856-4d8e-b85c-590a42039933" },
/* In this example we suppose that the NetworkingFacet was already created, so the UUID is enough to
attach it by using IsIdentifiedBy relation */
                        }
                }
        ],
        "isRelatedTo":[
                {
                        "@class": "Hosts",
                        "propagationConstraint": {
                                "add": "unpropagate",
                                "remove": "cascade"
                        },
                        "target": {
                                "@class":" EService",
                                "header": { "uuid":"9bff49c8-c0a7-45de-827c-accb71defbd3" }
/* The EService was already created, so the UUID is enough to attach it by using Hosts relation */
                        }
                }
        ]
}
```
Response
```
{
        "@class": "HostingNode",
        "header": { "uuid": "670eeabf-76c7-493f-a449-4e6e139a2e84", ...},
        "consistsOf":[
                {
                        "@class": "ConsistsOf",
                        "header": { "uuid": "9d0b1b2b-ac4e-40a9-8dea-bec90076e0ca", ...},
                        "target":{
                                "@class": "CPUFacet",
                                "header": { "uuid": "1daef6a8-5ca4-4700-844b-2a2d784e17b0", ...},
                                "model": "Opteron",
                                "vendor": "AMD",
                                "clockSpeed":"1 GHz"
                        }
                },{
                        "@class": "IsIdentifiedBy",
                        "header": { "uuid": "02a7072c-4f72-4568-945b-9ddccc881e9f", ...},
                        "target":{
                                "@class": "NetworkingFacet",
                                "header":{ "uuid": "59617b01-5856-4d8e-b85c-590a42039933", ...},
                                "ipAddress": "146.48.87.183",
```

```
                              "hostName": "pc-frosini.isti.cnr.it",
                              "domainName": "isti.cnr.it",
                              "mask": "255.255.248.0",
                              "broadcastAddress": "146.48.87.255"
                      }
              }
      ],
      "isRelatedTo":[
              {
                      "@class": "Hosts",
                      "header": { "uuid": "47494ad0-e606-4630-9def-4c607761ae14", ...},
                      "propagationConstraint":{
                              "add": "unpropagate",
                              "remove": "cascade"
                      },
                      "target":{
                              "@class": "EService",
                              "header": { "uuid":"9bff49c8-c0a7-45de-827c-accb71defbd3", ...}
                      }
              }
      ]
}
```

Java API

```
public <R extends Resource> R createResource(R resource) throws ResourceAlreadyPresentException,
ResourceRegistryException;
```

Example

```
NetworkingFacet networkingFacet = new NetworkingFacetImpl();
networkingFacet.setIPAddress("146.48.87.183");
networkingFacet.setHostName("pc-frosini.isti.cnr.it");
networkingFacet.setDomainName("isti.cnr.it");
networkingFacet.setMask("255.255.248.0");
networkingFacet.setBroadcastAddress("146.48.87.255");

networkingFacet = resourceRegistryPublisher.createFacet(networkingFacet);

HostingNode hostingNode = new HostingNodeImpl();

CPUFacet cpuFacet = new CPUFacetImpl();
cpuFacet.setClockSpeed("1 GHz");
cpuFacet.setModel("Opteron");
cpuFacet.setVendor("AMD");
hostingNode.addFacet(cpuFacet);

isIdentifiedBy = new IsIdentifiedByImpl<Resource, Facet>(hostingNode, networkingFacet, null);
hostingNode.attachFacet(isIdentifiedBy);

PropagationConstraint propagationConstraint = new PropagationConstraintImpl();
propagationConstraint.setRemoveConstraint(RemoveConstraint.cascade);
propagationConstraint.setAddConstraint(AddConstraint.unpropagate);

Hosts<HostingNode, EService> hosts = new HostsImpl<HostingNode, EService>(hostingNode, eService,
propagationConstraint);
hostingNode.attachResource(hosts);

hostingNode = resourceRegistryPublisher.createResource(hostingNode);
```

Alternative JAVA API

There are also two other equivalent methods with the following signature:
```
public String createResource(String resource) throws ResourceAlreadyPresentException,
ResourceRegistryException;
```

```
public String createResource(String resourceType, String resource) throws
ResourceAlreadyPresentException, ResourceRegistryException;
```

The first method gets the Resource to be created as a JSON string instead of as a Java class. The second method gets also the **resourceType** as a parameter (which has to be

30

specified as PATH PARAMETER in the request) avoiding forcing the client to retrieve it from the string. The second method is more efficient but you have to be sure that the **resourceType** is the same specified in the header of the serialized resource.

5.5.2 Update Resource Instance

REST API
POST /resource-registry/er/resource/{Resource Instance UUID}

Example
POST /resource-registry/er/resource/670eeabf-76c7-493f-a449-4e6e139a2e84
Request Body
```
{
        "@class": "HostingNode",
        "header": { "uuid": "670eeabf-76c7-493f-a449-4e6e139a2e84", ...},
        "consistsOf": [
                {
                        "@class": "ConsistsOf",
                        "header": { "uuid": "9d0b1b2b-ac4e-40a9-8dea-bec90076e0ca", ...},
                        "target": {
                                "@class": "CPUFacet",
                                "header": { "uuid": "1daef6a8-5ca4-4700-844b-2a2d784e17b0", ...},
                                "model": "Opteron",
                                "vendor": "AMD",
                                "clockSpeed": "1 GHz"
                        }
                },{
                        "@class": "IsIdentifiedBy",
                        "header": { "uuid": "02a7072c-4f72-4568-945b-9ddccc881e9f", ...},
                        "target": {
                                "@class": "NetworkingFacet",
                                "header": { "uuid": "59617b01-5856-4d8e-b85c-590a42039933", ...},
                                "ipAddress": "146.48.87.183",
                                "hostName": "pc-frosini.isti.cnr.it",
                                "domainName": "isti.cnr.it",
                                "mask": "255.255.248.0",
                                "broadcastAddress": "146.48.87.255",
                                "username": "luca.frosini" /* Added this property */
                        }
                }
        ]
}
```

Response
```
{
        "@class": "HostingNode",
        "header": { "uuid": "670eeabf-76c7-493f-a449-4e6e139a2e84", ...},
        "consistsOf": [
                {
                        "@class": "ConsistsOf",
                        "header": { "uuid": "9d0b1b2b-ac4e-40a9-8dea-bec90076e0ca", ...},
                        "target": {
                                "@class": "CPUFacet",
                                "header": { "uuid": "1daef6a8-5ca4-4700-844b-2a2d784e17b0", ...},
                                "model": "Opteron",
                                "vendor": "AMD",
                                "clockSpeed": "1 GHz"
                        }
                },{
                        "@class": "IsIdentifiedBy",
                        "header": { "uuid": "02a7072c-4f72-4568-945b-9ddccc881e9f", ...},
                        "target": {
                                "@class": "NetworkingFacet",
                                "header": { "uuid": "59617b01-5856-4d8e-b85c-590a42039933", ...},
                                "ipAddress": "146.48.87.183",
                                "hostName": "pc-frosini.isti.cnr.it",
                                "domainName": "isti.cnr.it",
                                "mask": "255.255.248.0",
```

31

```
                              "broadcastAddress": "146.48.87.255",
                              "username": "luca.frosini"
                      }
              }
      ]
}
```

Java API
public <R extends Resource> R updateResource(R resource) throws ResourceNotFoundException,
ResourceRegistryException;

Example
```
/* This is just a code example, here we suppose that there is only one identification Facet of the type
(NetworkingFacet). This could not be true in real scenario*/
networkingFacet = (NetworkingFacet) hostingNode.getIdentificationFacets().get(0);
networkingFacet.setAdditionalProperty("username", "luca.frosini");

hostingNode = resourceRegistryPublisher.updateResource(hostingNode);
```

Alternative JAVA API
There are also two other equivalent methods with the following signature:
public String updateResource(String resource) throws ResourceNotFoundException,
ResourceRegistryException;
public String updateResource(UUID uuid, String resource) throws ResourceNotFoundException,
ResourceRegistryException;

The first method gets the Resource to be created as a JSON string instead of as a Java
class. The second get also the **uuid** as a parameter (which has to be specified as PATH
PARAMETER in the request) avoiding forcing the client to retrieve it from the string. The
second method is more efficient but you have to be sure that the **uuid** is the same
specified in the header of the serialized resource.

5.5.3 Delete Resource Instance

REST API
DELETE /resource-registry/er/resource/{Resource Instance UUID}

Example
DELETE /resource-registry/er/resource/670eeabf-76c7-493f-a449-4e6e139a2e84

Java API
public <R extends Resource> boolean deleteResource(R resource) throws ResourceNotFoundException,
ResourceRegistryException;

Example
boolean deleted = resourceRegistryPublisher.deleteResource(hostingNode);

Alternative JAVA API
There is also another equivalent method with the following signature:
public boolean deleteResource(UUID uuid) throws ResourceNotFoundException,
ResourceRegistryException;

The method just requires the UUID of the Resource to be deleted.

5.6 ConsistsOf

5.6.1 Create ConsistsOf Instance

REST API
PUT /resource-registry/er/consistsOf/{ConsistOfType}

Example 1
PUT /resource-registry/er/consistsOf/IsIdentifiedBy

In this example the target Facet already exists. The Service set automatically the Facet_Based_Resource_Model#PropagationConstraint Propagation Constraint to default values (i.e. remove=cascadeWhenOrphan, add=propagate)

Request Body
```
{
        "@class": "IsIdentifiedBy",
        "source": {
                "@class": "HostingNode",
                "header": {"uuid": "670eeabf-76c7-493f-a449-4e6e139a2e84" } // The HostingNode must be
already created. The header with UUID is enough.
        }
        "target": {
                "@class": "NetworkingFacet",
                "header": { "uuid": "59617b01-5856-4d8e-b85c-590a42039933" },
                /* In this example we suppose that the NetworkingFacet already exists, so the UUID is
enough to attach it by using IsIdentifiedBy relation */
        }
}
```

Response
```
{
        "@class": "IsIdentifiedBy",
        "propagationConstraint": {
                "add": "propagate",
                "remove": "cascadeWhenOrphan"
        },
        "header": { "uuid": "02a7072c-4f72-4568-945b-9ddccc881e9f", ...},
        "target":{
                "@class": "NetworkingFacet",
                "header": { "uuid": "59617b01-5856-4d8e-b85c-590a42039933", ...},
                "ipAddress": "146.48.87.183",
                "hostName": "pc-frosini.isti.cnr.it",
                "domainName": "isti.cnr.it",
                "mask": "255.255.248.0",
                "broadcastAddress": "146.48.87.255"
        }
}
```

Example 2
PUT /resource-registry/er/consistsOf/ConsistsOf

In this example the target Facet is created contextually with the ConsistsOf relation. Moreover, the Facet_Based_Resource_Model#PropagationConstraint Propagation Constraint are explicitly set (i.e. remove=cascade, add=propagate).

Request Body
```
{
        "@class": "ConsistsOf",
        "propagationConstraint": {
                "add": "propagate",
                "remove": "cascade"
        },
        "source": {
                "@class": "HostingNode",
                "header": { "uuid": "670eeabf-76c7-493f-a449-4e6e139a2e84" } // The HostingNode must be
```

already created. The header with UUID is enough.

```
                }
        "target": {
                "@class": "CPUFacet",
                "model": "Opteron",
                "vendor": "AMD",
                "clockSpeed": "3 GHz"
        }
}
```

Response

```
{
        "@class": "ConsistsOf",
        "header": { "uuid": "9bff49c8-c0a7-45de-827c-accb71defbd3", ...},
        "propagationConstraint": {
                "add": "propagate",
                "remove": "cascade"
        },
        "target": {
                "@class": "CPUFacet",
                "header": { "uuid": "1daef6a8-5ca4-4700-844b-2a2d784e17b0", ...},
                "model": "Opteron",
                "vendor": "AMD",
                "clockSpeed": "1 GHz"
        }
}
```

Java API
public <C extends ConsistsOf<? extends Resource, ? extends Facet>> C createConsistsOf(C consistsOf)
throws FacetNotFoundException, ResourceNotFoundException, ResourceRegistryException;

Example 1
IsIdentifiedBy isIdentifiedBy = new IsIdentifiedByImpl<Resource, Facet>(hostingNode, networkingFacet, null);
resourceRegistryPublisher.createConsistsOf(isIdentifiedBy);

Example 2
CPUFacet cpuFacet = new CPUFacetImpl();
cpuFacet.setClockSpeed("1 GHz");
cpuFacet.setModel("Opteron");
cpuFacet.setVendor("AMD");

PropagationConstraint propagationConstraint = new PropagationConstraintImpl();
propagationConstraint.setRemoveConstraint(RemoveConstraint.cascade);
propagationConstraint.setAddConstraint(AddConstraint.propagate);

ConsistsOf consistsOf = new ConsistsOfImpl<Resource, Facet>(hostingNode, cpuFacet, propagationConstraint);
consistsOf = resourceRegistryPublisher.createConsistsOf(consistsOf);

Alternative JAVA API
There are also two other equivalent methods with the following signature:
public String createConsistsOf(String consistsOfType, String consistsOf) throws FacetNotFoundException, ResourceNotFoundException, ResourceRegistryException;

public String createConsistsOf(String consistsOf) throws FacetNotFoundException, ResourceNotFoundException, ResourceRegistryException;

The first method gets the **consistsOfType** to be created as a JSON string instead of as a Java class. The second get also the **consistsOfType** as a parameter (which has to be specified as PATH PARAMETER in the request) avoiding forcing the client to retrieve it from the string. The second method is more efficient but you have to be sure that the **consistsOfType** is the same specified in the header of the serialized resource.

34

5.6.2 Delete ConsistsOf Instance

REST API
DELETE /resource-registry/er/consistsOf/{ConsistsOf Instance UUID}

Example 1
DELETE /resource-registry/er/consistsOf/02a7072c-4f72-4568-945b-9ddccc881e9f

Example 2
DELETE /resource-registry/er/consistsOf/9bff49c8-c0a7-45de-827c-accb71defbd3

Java API
public <C extends ConsistsOf<? extends Resource, ? extends Facet>> boolean deleteConsistsOf(C consistsOf) throws ResourceRegistryException;

Example 1
boolean deleted = resourceRegistryPublisher.deleteConsistsOf(isIdentifiedBy);

Example 2
boolean deleted = resourceRegistryPublisher.deleteConsistsOf(consistsOf);

Alternative JAVA API
There is also another equivalent method with the following signature:
public boolean deleteConsistsOf(UUID uuid) throws ResourceRegistryException;

The method just requires the UUID of the ConsistsOf relation to be deleted.

Example 1
UUID uuid = UUID.fromString("02a7072c-4f72-4568-945b-9ddccc881e9f");
boolean deleted = resourceRegistryPublisher.deleteConsistsOf(uuid);

Example 2
UUID uuid = UUID.fromString("9bff49c8-c0a7-45de-827c-accb71defbd3");
boolean deleted = resourceRegistryPublisher.deleteConsistsOf(uuid);

5.7 IsRelatedTo

5.7.1 Create IsRelatedTo Instance

REST API
PUT /resource-registry/er/isRelatedTo/{IsRelatedToType}

Example
PUT /resource-registry/er/isRelatedTo/Hosts

In this example, the target Resource already exists. The Propagation Constraints are explicitly set. Please note that otherwise the service set the Propagation Constraint to default values (i.e. remove=keep, add=unpropagate)

Request Body
```
{
        "@class": "Hosts",
        "propagationConstraint": {
                "add": "unpropagate",
                "remove": "cascade"
        },
        "target": {
                "@class": "EService",
```

```
        "header": { "uuid": "9bff49c8-c0a7-45de-827c-accb71defbd3" }
/* The EService was already created, so the UUID is enough to attach it by using Hosts relation */
    }
}
```

Response
```
{
        "@class":"Hosts",
        "header":{"uuid":"47494ad0-e606-4630-9def-4c607761ae14", ...},
        "propagationConstraint":{
                "add":"unpropagate",
                "remove": "cascade"
        },
        "target":{
                "@class":"EService",
                "header":{"uuid":"9bff49c8-c0a7-45de-827c-accb71defbd3", ...}
        }
}
```

Java API
```
public <I extends IsRelatedTo<? extends Resource, ? extends Resource>> I createIsRelatedTo(I
isRelatedTo) throws ResourceNotFoundException, ResourceRegistryException;
```

Example
```
PropagationConstraint propagationConstraint = new PropagationConstraintImpl();
propagationConstraint.setRemoveConstraint(RemoveConstraint.cascade);
propagationConstraint.setAddConstraint(AddConstraint.propagate);

Hosts<HostingNode, EService> hosts = new HostsImpl<>(hostingNode, eService, propagationConstraint);
hosts = resourceRegistryPublisher.createIsRelatedTo(hosts);
```

Alternative Java API
There are also two other equivalent methods with the following signature:
```
public String createIsRelatedTo(String isRelatedToType, String isRelatedTo) throws
ResourceNotFoundException, ResourceRegistryException;

public String createIsRelatedTo(String isRelatedTo) throws ResourceNotFoundException,
ResourceRegistryException;
```

The first method gets the **isRelatedTo** to be created as a JSON string instead of as a Java
class. The second get also the **isRelatedTo** as a parameter (which has to be specified as
PATH PARAMETER in the request) avoiding forcing the client to retrieve it from the string.
The second method is more efficient but you have to be sure that the **isRelatedToType** is
the same specified in the header of the serialized resource.

5.7.2 Delete IsRelatedTo Instance

REST API
DELETE /resource-registry/er/isRelatedTo/{IsRelatedToInstanceUUID}

Example
DELETE /resource-registry/er/isRelatedTo/47494ad0-e606-4630-9def-4c607761ae14
Java API
```
public <I extends IsRelatedTo<? extends Resource, ? extends Resource>> boolean deleteIsRelatedTo(I
isRelatedTo) throws ResourceRegistryException;
```

Example
```
boolean deleted = resourceRegistryPublisher.deleteIsRelatedTo(hosts);
```

Alternative Java API
There is also another equivalent methods with the following signature:
public boolean deleteIsRelatedTo(UUID uuid) throws ResourceRegistryException;

Example
The method just requires the UUID of the ConsistsOf relation to be deleted.
UUID uuid = UUID.fromString("47494ad0-e606-4630-9def-4c607761ae14")
boolean deleted = resourceRegistryPublisher.deleteIsRelatedTo(uuid);

5.8 Query and Access

This sections provide information regarding on how to interact with Resource Registry
Service for Query and Access. The REST and JAVA API are presented for each
functionality. Please note that the provided examples can intentionally hide some details in
the response to avoid unneeded complexity. Apart from the REST API, this port type can
be used also by using Resource Registry Client java client.

The Resource Registry Client has the following maven coordinates
```
<dependency>
        <groupId>org.gcube.information-system</groupId>
        <artifactId>resource-registry-client</artifactId>
        <version>[1.0.0-SNAPSHOT, 2.0.0-SNAPSHOT)</version>
</dependency>
```

To use the client you need first to get a *ResourceRegistryClient* instance.
By using the *ResourceRegistryClientFactory.create()* method the library discovers the
correct endpoint to interact with the Resource Registry for the current context.
SecurityTokenProvider.instance.set("Your-Token-Here"); //If not already set
ResourceRegistryClient resourceRegistryClient = ResourceRegistryClientFactory.create();

5.8.1 Exists

REST API
HEAD /resource-registry/access/instance/{ERType}/{InstanceUUID}

Example
HEAD /resource-registry/access/instance/ContactFacet/4d28077b-566d-4132-b073-f4edaf61dcb9

Java API
public <ERType extends ER> boolean exists(Class<ERType> clazz, UUID uuid) throws
ERNotFoundException, ERAvailableInAnotherContextException, ResourceRegistryException;

Example
UUID uuid = UUID.fromString("4d28077b-566d-4132-b073-f4edaf61dcb9");
resourceRegistryClient.exists(ContactFacet.class, uuid);

Alternative Java API
public boolean exists(String type, UUID uuid) throws ERNotFoundException,
ERAvailableInAnotherContextException, ResourceRegistryException;

Example
UUID uuid = UUID.fromString("4d28077b-566d-4132-b073-f4edaf61dcb9");
resourceRegistryClient.exists("ContactFacet", uuid);

5.8.2 Get Instance

REST API
GET /resource-registry/access/instance/{ER Type}/{Instance UUID}

Example
GET /resource-registry/access/instance/CPUFacet/69f0b376-38d2-4a85-bc63-37f9fa323f82

Response Body
```
{
        "@class": "CPUFacet",
        "header": {
                "uuid": "69f0b376-38d2-4a85-bc63-37f9fa323f82",
                "creator": "luca.frosini",
                "lastUpdater": "luca.frosini",
                "creationTime": "2016-10-05 11:16:24",
                "lastUpdateTime": "2016-10-05 11:16:24
        },
        "model": "Opteron",
        "vendor": "AMD",
        "clockSpeed": "1 GHz"
}
```

Java API
public <ERType extends ER> ERType getInstance(Class<ERType> clazz, UUID uuid) throws
ERNotFoundException, ERAvailableInAnotherContextException, ResourceRegistryException;

Example
UUID uuid = UUID.fromString("69f0b376-38d2-4a85-bc63-37f9fa323f82");
CPUFacet cpuFacet = resourceRegistryClient.getInstance(CPUFacet.class, uuid);

Alternative Java API
public String getInstance(String type, UUID uuid) throws ERNotFoundException,
ERAvailableInAnotherContextException, ResourceRegistryException;

Example
UUID uuid = UUID.fromString("69f0b376-38d2-4a85-bc63-37f9fa323f82");
String cpuFacetJsonString = resourceRegistryClient.getInstance("CPUFacet" uuid);

5.8.3 Get All Instances of a Specific Type

REST API
GET /resource-registry/access/instances/{ERType}?[polymorphic=(true|false)]
Default: polymorphic=false

Example 1
GET /resource-registry/access/instances/EService
Response
```
[
        {
                "@class": "EService",
                "header": { "uuid": "0717b450-a698-11e2-900a-a46c6ff57f05", ...},
                "consistsOf": [
                        {
                                "@class": "IsIdentifiedBy",
                                "header": { "uuid": "aa1340d3-2229-497f-9eeb-cc7db93950fe", ...},
                                "target":{
                                        "@class": "SoftwareFacet",
                                        "header": { "uuid": "187bee6d-6742-49a7-be89-68bdb0f2f221", ...},
                                        "group": "DataStorage",
                                        "name": "StorageManager",
                                        "version": "2.3.0-0",
                                }
                },{
```

```
            "@class": "ConsistsOf",
            "header": { "uuid": "1f5b5608-4a91-4fe3-a7c2-edf3aeb5dbd7", ...},
            "target": {
                    "@class": "AccessPointFacet",
                    "header": { "uuid": "3b6061f9-e2ab-4c01-b3b2-48b470a5b8a ", ...},
                    "endpoint":  "mongo3-p-d4s.d4science.org",
                    "description": "MongoDB server",
                    "authorization": {
                            "@class": "ValueSchema",
                            "value": "d4sUser:Nxae6MegJrITUD6wyBTimw==",
                            "schema":       "USERNAME:PASSWORD"
                    }
            }
        }
    ]
},
....
]
```

Example 2
GET /resource-registry/access/instances/EService?polymorphic=true
Response
```
[
        {
                "@class": "RunningPlugin",
                "header": { "uuid": "66d69dab-203e-45ff-b49e-a8fa4126a392"},
                "consistsOf": [
                        {
                                "@class": "IsIdentifiedBy",
                                "header": { "uuid": "5e8cf3e3-8c75-49d4-98db-95d4d08d61f9", ...},
                                "target": {
                                    "@class": "SoftwareFacet",
                                    "header": { "uuid": "3715696d-796f-4e92-98a3-f9a38a2f8d5e", ...},
                                    "group": "Accounting",
                                    "name": "accounting-aggregator-se-plugin",
                                    "version": "1.3.0",
                                }
                        },
                        ...
                ]
        },
        {
                "@class": "EService",
                "header": { "uuid": "0717b450-a698-11e2-900a-a46c6ff57f05", ...},
                "consistsOf": [
                        {
                                "@class": "IsIdentifiedBy",
                                "header": { "uuid": "aa1340d3-2229-497f-9eeb-cc7db93950fe", ...},
                                "target": {
                                    "@class": "SoftwareFacet",
                                    "header": { "uuid": "187bee6d-6742-49a7-be89-68bdb0f2f221", ...},
                                    "group": "DataStorage",
                                    "name": "StorageManager",
                                    "version": "2.3.0-0",
                                }
                        },{
                                "@class": "ConsistsOf",
                                "header": { "uuid": "1f5b5608-4a91-4fe3-a7c2-edf3aeb5dbd7", ...},
                                "target": {
                                    "@class": "AccessPointFacet",
                                    "header": { "uuid": "3b6061f9-e2ab-4c01-b3b2-48b470a5b8a ", ...},
                                    "endpoint":  "mongo3-p-d4s.d4science.org"
                                    "description": "MongoDB server"
                                    "authorization": {
                                            "@class"          "ValueSchema"
                                            "value": "d4sUser:Nxae6MegJrITUD6wyBTimw=="
                                            "schema": "USERNAME:PASSWORD"
                                    }
                                }
                        },
```

```
        ....
    ]
},
....
]
```

Java API
public <ERType extends ER, R extends Resource> List<R> getInstances(Class<ERType> clazz, Boolean polymorphic) throws ResourceRegistryException;

Example 1
List<EService> eServices = resourceRegistryClient.getInstances(EService.class, false);

Example 2
List<EService> eServices = resourceRegistryClient.getInstances(EService.class, true);

Alternative Java API
public String getInstances(String type, Boolean polymorphic) throws ResourceRegistryException;

Example 1
String eServicesJsonString = resourceRegistryClient.getInstances("EService", false);

Example 2
String eServicesJsonString = resourceRegistryClient.getInstances("EService", true);

5.8.4 Get All Instances in relation with a specific entity instance

REST API
GET /resource-registry/access/instances/{Entity Type}?[polymorphic=(true|false)]&reference={Entity Instance UUID}&direction=(in|out|both)

Default: polymorphic: false, direction: both

Example
GET /resource-registry/access/instances/EService?polymorphic=true&reference=b0d15e45-62af-4221-b785-7d014f10e631&direction=out

In this example, we retrieve all EServices that are the target of a relation starting from the Entity identified by b0d15e45-62af-4221-b785-7d014f10e631

- **HostingNode**(b0d15e45-62af-4221-b785-7d014f10e631) -> **Hosts** -> **EService**(4a7daacb-f13b-4685-b5c1-040c86806c16)
- **HostingNode**(b0d15e45-62af-4221-b785-7d014f10e631) -> **Hosts** -> **Eservice**(6e6442b9-37f1-479d-92eb-f935a983caba))

Response
```
[
    {
        "@class": "EService",
        "header": { "uuid": "4a7daacb-f13b-4685-b5c1-040c86806c16", ...},
        "consistsOf": [
            {
                "@class": "IsIdentifiedBy",
                "header": { "uuid": "6c6e7deb-a911-4612-989d-d770f30c0d69", ...},
                "target": {
                    "@class": "SoftwareFacet",
                    "header": { "uuid": "92f30e88-eff1-4282-b773-f3884b179d8d", ...},
                    "group": "VREManagement",
                    "name": "SmartExecutor",
                    "version": "1.7.0",
                }
            },
            ....
        ]
```

```
            },
            {
                    "@class": "EService",
                    "header": { "uuid": "6e6442b9-37f1-479d-92eb-f935a983caba", ...},
                    "consistsOf": [
                            {
                                    "@class": "IsIdentifiedBy",
                                    "header": { "uuid": "068f8d72-9495-4bd4-85b9-0ee234e99af6", ...},
                                    "target": {
                                            "@class": "SoftwareFacet",
                                            "header": { "uuid": "35546e56-6e76-4cc9-8c83-7320fd923597", ...},
                                            "group": "VREManagement",
                                            "name": "WhnManager",
                                            "version": "2.0.0",
                                    }
                            },
                            .....
                    ]
            },
            ....
]
```

Java API
public <ERType extends ER, E extends Entity, R extends Resource> List<R>
getInstancesFromEntity(Class<ERType> clazz, Boolean polymorphic, E reference, Direction direction)
throws ResourceRegistryException;

Example
UUID uuid = UUID.fromString("b0d15e45-62af-4221-b785-7d014f10e631");
HostingNode hostingNode = resourceRegistryClient.getInstance(HostingNode.class, uuid);
List<EService> eServices = resourceRegistryClient.getInstancesFromEntity(EService.class, true,
hostingNode, Direction.out);

Alternative Java API
public <ERType extends ER, R extends Resource> List<R> getInstancesFromEntity(Class<ERType> clazz,
Boolean polymorphic, UUID reference, Direction direction) throws ResourceRegistryException;

public String getInstancesFromEntity(String type, Boolean polymorphic, UUID reference, Direction direction)
throws ResourceRegistryException;

Example
UUID uuid = UUID.fromString("b0d15e45-62af-4221-b785-7d014f10e631");
List<EService> eServices = resourceRegistryClient.getInstancesFromEntity(EService.class, true, uuid,
Direction.out);

UUID uuid = UUID.fromString("b0d15e45-62af-4221-b785-7d014f10e631");
String eServicesJsonString = resourceRegistryClient.getInstancesFromEntity(EService.NAME, true, uuid,
Direction.out);

5.8.5 Get Filtered Resource Instances

This API allows retrieval of all resource of a given **Resource Type** described by a **Facet
Type** having certain *keys and values*. The relation between the **Resource** and the **Facet**
must be of the specified **ConsistsOf Type** (polymorphism is always used, use ConsistsOf
to be more generic).

REST API
GET /resource-registry/access/resourceInstances/{Resource Type}/{ConsistsOf Type}/{Facet
Type}?[polymorphic=(true|false)]&key1=value1&key2=value2&...
Default: polymorphic: false

Example
GET /resource-
registry/access/resourceInstances/EService/IsIdentifiedBy/SoftwareFacet?polymorphic=true&group=DataAc
cess&name=HomeLibraryWebapp

Response

```
{
        "@class": "EService",
        "header": { "uuid": "38823ddc-0713-4e83-9670-79e472408b0c", ...},
        "consistsOf": [
                {
                        "@class": "IsIdentifiedBy",
                        "header": { "uuid": "8829279d-70c1-4327-a817-169ed9a52467", ...},
                        "target": {
                                "@class": "SoftwareFacet",
                                "header": { "uuid": "747713c0-fcbf-42d9-8709-c782c2121f9d", ...},
                                "name": "HomeLibraryWebapp",
                                "group": "DataAccess",
                                "version": "1.5.0-SNAPSHOT",
                                "description": "home library webapp",
                                "qualifier": null,
                                "optional": false
                        }
                },
                ...
        ]
}
```

Java API
public <R extends Resource> List<R> getFilteredResources(Class<R> resourceClass, Class<? extends ConsistsOf> consistsOfClass, Class<? extends Facet> facetClass,boolean polymorphic, Map<String, Object> map) throws ResourceRegistryException

Example
```
Map<String, Object> map = new HashMap<>();
map.put("group", "DataAccess");
map.put("name", "HomeLibraryWebapp");

List<EService> eServices = resourceRegistryClient.getFilteredResources(EService.class,
IsIdentifiedBy.class, SoftwareFacet.class, true, map);
```

Alternative Java API
public List<Resource> getFilteredResources(String resourceType, String consistsOfType, String facetType, boolean polymorphic, Map<String, Object> map) throws ResourceRegistryException;

Example
```
Map<String, Object> map = new HashMap<>();
map.put("group", "DataAccess");
map.put("name", "HomeLibraryWebapp");

String eServiceJsonString = resourceRegistryClient.getFilteredResources(EService.NAME,
IsIdentifiedBy.NAME, SoftwareFacet.NAME, true, map);
```

5.8.6 Raw Query

This API provides a way to query the underlying database persistence by using the persistence query language dialect. This API does not provide any consistency with the Information System Model concepts. The result is related to how the service decides to represent the Information System Model concepts on persistence data model. At the time of writing, the underlying database persistence is OrientDB. It should be used only for development purposes because the way to represent the Information System Model concepts can change at any time or can change the database persistence. At time of writing the query language supported is OrientDB SQL Dialect[16]
GET /resource-registry/access?query=SELECT FROM Facet

REST API
GET /resource-registry/access?query={Query}

---

[16] http://orientdb.com/docs/last/SQL.html

Example 1
GET /resource-registry/access?query=SELECT FROM SoftwareFacet LIMIT 2

Response Body

```
{
  "result": [
    {
      "@type": "d",
      "@rid": "#99:5",
      "@version": 12,
      "@class": "SoftwareFacet",
      "header": {
        "@type": "d",
        "@version": 0,
        "@class": "Header",
        "uuid": "6b724a7c-9f51-4a4e-8e8e-1636ca2e9d29",
        "creator": "VREManagement:WhnManager:pc-frosini.isti.cnr.it_8080",
        "creationTime": "2017-10-05 16:09:02.618 +0200",
        "lastUpdateTime": "2017-10-05 17:23:44.191 +0200",
        "@fieldTypes": "creationTime=t,lastUpdateTime=t"
      },
      "name": "WhnManager",
      "description": "Web Hosting Node Service",
      "optional": false,
      "version": "2.0.0-SNAPSHOT",
      "group": "VREManagement",
      "_allow": [
        "#4:12",
        "#5:13",
        "#4:14"
      ],
      "_allowRead": [
        "#4:13",
        "#4:11"
      ],
      "in_IsIdentifiedBy": [
        "#168:5"
      ],
      "@fieldTypes": "_allow=n,_allowRead=n,in_IsIdentifiedBy=g"
    },
    {
      "@type": "d",
      "@rid": "#99:6",
      "@version": 5,
      "@class": "SoftwareFacet",
      "header": {
        "@type": "d",
        "@version": 0,
        "@class": "Header",
        "uuid": "bc98eec4-4365-49fd-83b3-2cacaf17f8bf",
        "creator": "VREManagement:SmartExecutor:pc-frosini.isti.cnr.it_8080",
        "creationTime": "2017-10-05 17:22:06.351 +0200",
        "lastUpdateTime": "2017-10-05 17:23:44.206 +0200",
        "@fieldTypes": "creationTime=t,lastUpdateTime=t"
      },
      "name": "SmartExecutor",
      "description": "Smart Executor Service",
      "optional": false,
      "version": "1.7.0-SNAPSHOT",
      "group": "VREManagement",
      "_allow": [
        "#4:12",
        "#5:15",
        "#4:14"
      ],
      "_allowRead": [
        "#4:13",
        "#4:11"
```

```
      ],
      "in_IsIdentifiedBy": [
        "#168:6"
      ],
      "@fieldTypes": "_allow=n,_allowRead=n,in_IsIdentifiedBy=g"
    }
  ],
  "notification": "Query executed in 0.147 sec. Returned 2 record(s)"
}
```

It is apparent that a lot of database specific information is returned but cannot be used or relied on to interact with registry.

Example 2
GET /resource-registry/access?query=SELECT FROM EService LIMIT 2

Response Body
```
{
  "result": [
    {
      "@type": "d",
      "@rid": "#138:2",
      "@version": 12,
      "@class": "EService",
      "header": {
        "@type": "d",
        "@version": 0,
        "@class": "Header",
        "uuid": "077a389f-6676-49bb-a925-16bea54c5f5d",
        "creator": "VREManagement:WhnManager:pc-frosini.isti.cnr.it_8080",
        "creationTime": "2017-10-05 16:09:02.604 +0200",
        "lastUpdateTime": "2017-10-05 17:23:44.191 +0200",
        "@fieldTypes": "creationTime=t,lastUpdateTime=t"
      },
      "_allow": [
        "#4:12",
        "#5:13",
        "#4:14"
      ],
      "_allowRead": [
        "#4:13",
        "#4:11"
      ],
      "out_IsIdentifiedBy": [
        "#168:5"
      ],
      "out_ConsistsOf": [
        "#165:32",
        "#166:32",
        "#164:33"
      ],
      "in_Hosts": [
        "#230:2"
      ],
      "@fieldTypes": "_allow=n,_allowRead=n,out_IsIdentifiedBy=g,out_ConsistsOf=g,in_Hosts=g"
    },
    {
      "@type": "d",
      "@rid": "#138:3",
      "@version": 5,
      "@class": "EService",
      "header": {
        "@type": "d",
        "@version": 0,
        "@class": "Header",
        "uuid": "cabca29a-59e4-463d-9932-02c6d68c8ce0",
        "creator": "VREManagement:SmartExecutor:pc-frosini.isti.cnr.it_8080",
        "creationTime": "2017-10-05 17:22:06.340 +0200",
        "lastUpdateTime": "2017-10-05 17:23:44.206 +0200",
```

```
          "@fieldTypes": "creationTime=t,lastUpdateTime=t"
      },
      "_allow": [
          "#4:12",
          "#5:15",
          "#4:14"
      ],
      "_allowRead": [
          "#4:13",
          "#4:11"
      ],
      "out_IsIdentifiedBy": [
          "#168:6"
      ],
      "out_ConsistsOf": [
          "#164:36",
          "#165:36",
          "#166:36"
      ],
      "in_Hosts": [
          "#230:3"
      ],
      "@fieldTypes": "_allow=n,_allowRead=n,out_IsIdentifiedBy=g,out_ConsistsOf=g,in_Hosts=g"
  }
  ],
  "notification": "Query executed in 0.197 sec. Returned 2 record(s)"
}
```

It should be noticed that only the Vertexes of **EServices** are returned which don't contain the facets.

5.8.7 Read Context

Allow to read the definition of a Context.
REST API
This API is also exposed in Context Port Type
GET /resource-registry/access/context/{Context UUID}

Example
Read the Context having UUID **9d73d3bd-1873-490c-b0a7-e3c0da11ad52**

Request URL
GET /resource-registry/context/9d73d3bd-1873-490c-b0a7-e3c0da11ad52

Response Body
```
{
      "@class": "Context",
      "name": "devVRE",
      "header": {
              "@class": "Header",
              "uuid": "9d73d3bd-1873-490c-b0a7-e3c0da11ad52",
              "creator": "luca.frosini",
              "modifiedBy": "luca.frosini",
              "creationTime": "2017-03-17 11:47:56",
              "lastUpdateTime": "2017-03-17 11:52:56"
      }
}
```

5.8.8 Read Type Definition

Allow to read Type Definition. This API is also exposed in Access Port Type.
REST API
GET /resource-registry/access/schema/{TypeName}

Example
GET /resource-registry/schema/ContactFacet

Response
```
{
        "name":"ContactFacet",
        "description":"This facet is expected to capture contact information",
        "abstractType": false,
        "superclasses":["Facet"],
        "properties":[
                {
                        "name":"name",
                        "description":"First Name",
                        "mandatory":true,
                        "readonly":false,
                        "notnull":true,
                        "max":null,
                        "min":null,
                        "regexpr":null,
                        "linkedType":null,
                        "linkedClass":null,
                        "type":7 /* String*/
                },{
                        "name":"eMail",
                        "description": "A restricted range of RFC-822 compliant email address. ... ",
                        "mandatory":true,
                        "readonly":false,
                        "notnull":true,
                        "max":null,
                        "min":null,
                        "regexpr":"^[a-z0-9._%+-]{1,128}@[a-z0-9.-]{1,128}$",
                        "linkedType":null,
                        "linkedClass":null,
                        "type":7 /* String */
                }
        ]
}
```

6 Backend Database (i.e. OrientDB as Graph Database)

OrientDB is a Multi-Model Open Source NoSQL DBMS that brings together the power of graphs and the flexibility of documents into one scalable high-performance operational database[17]. OrientDB engine supports **Graph**, **Document**, **Key/Value**, and **Object** models. A graph represents a network-like structure consisting of Vertices (also known as Nodes) interconnected by Edges (also known as Arcs).

OrientDB's graph model is represented by the concept of a property graph, which defines the following:

- **Vertex** - an entity that can be linked with other Vertices and has the following mandatory properties:
  o unique identifier
  o set of incoming Edges
  o set of outgoing Edges
- **Edge** - an entity that links two Vertices and has the following mandatory properties:
  o unique identifier
  o link to an incoming Vertex (also known as head)
  o link to an outgoing Vertex (also known as tail)

In addition to mandatory properties, each vertex or edge can also hold a set of custom properties. These properties can be defined by users, which can make vertices and edges appear similar to documents.[18] Given that, we can say that OrientDB, used as graph database, is de-facto a graph-document database. This peculiarity provides an excellent support for the Information System model which has been mapped on OrientDB concepts as following:

- Entities are modelled as Vertexes
- Relations are modelled as Edges

In both cases, the OrientDB internal ID has been hidden and, instead, the header property (embedded) is created which provides, among others, the ID to uniquely identify the Entity or Relation.

Another important characteristic is the native support of embedded properties. Embedded properties are structured properties inside a vertex or an edge. The Header is the only properties of resources.

OrientDB provides a simple referential integrity support guaranteeing that if a vertex is deleted then every attached edge (incoming or outgoing) is also deleted. The Resource Registry provides additional referential integrity support by using directives contained in each PropagationConstraint property attached to edges. It is responsibility of the Resource Registry to provide support for this.

---

[17] http://orientdb.com/docs/2.2.x/
[18] http://orientdb.com/docs/2.2.x/Tutorial-Introduction-to-the-NoSQL-world.html

7 The Studio GUI

The Content administrator is allowed to use the Web Graphical User Interface (GUI) provided with OrientDB called Studio.

Figure 1 shows the interface allowing browsing and searching of the content of the Joint Resource Registry. It also allows the inspection of the schema of the resources defined in the PARTHENOS Entity Model. At the top of the page the search bar is presented. The browsing is paginated and the types are divided into vertex and edge types.

Figure 1: Schema Manager

Moreover, two different interfaces to get the results of a query are provided. The first one, see Figure 2, provides textual results, while the second one, see Figures 3 and 4, provides a graphical representation of the graph results of the query.

The interface providing textual results also allows editing of any of the presented instances by clicking on the resulting row.

Figure 2: Textual Query Inspector

The interface providing the representation of the graph instead allows inspection of the content of the vertexes and edges by clicking on any one of them. The information is provided in the side panel on the left. The side panel has two tabs: the first shows the properties of the selected element; the second tab is used to change the presentation information of the element such as the colour of the circle for vertexes, and the attached label for edges and vertexes. The label can be either one of the attributes of the element or OrientDB internal information such as the internal id.
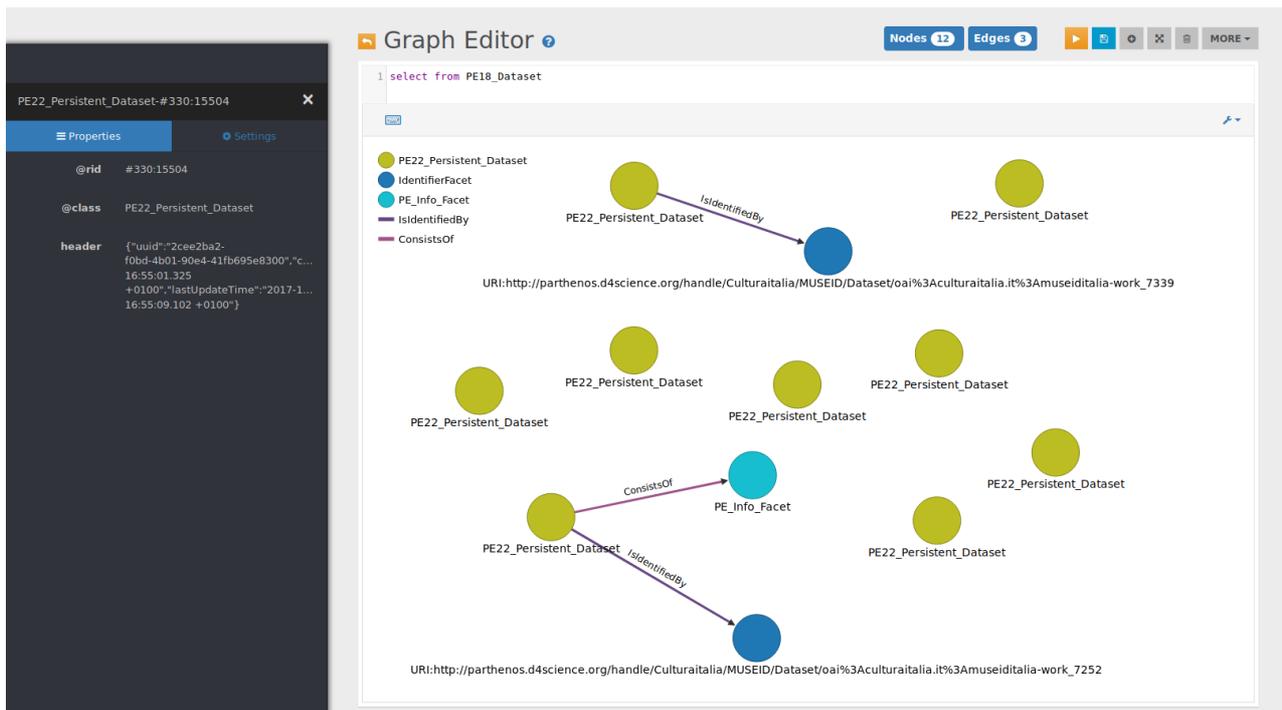
Figure 3: Graph Query Inspector

The Graph Query Inspector interface also allows iterative inspection by navigating the relations (edges) created between the entities (vertexes). By clicking on the element an overlay menu is presented. The menu directs the user to the valid options available for the navigation.
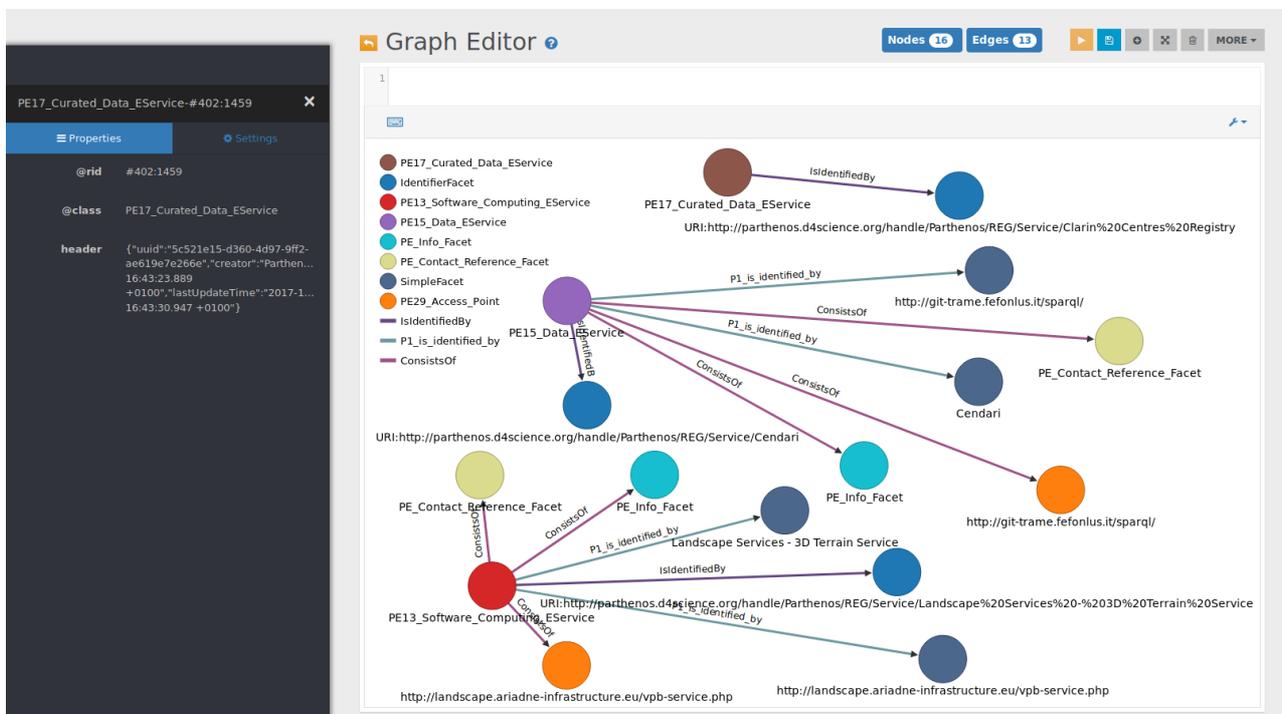
Figure 4: Graph Editor

The Content administrator graphical user interface will be complemented by an additional interface designed for end-users. This additional tool will present the content of the Joint Resource Registry as a catalogue of resources. The catalogue will be searchable and

browsable while faceted search will allow interactive inspections of the PARTHENOS entities.

The end-user graphical user interface is currently under testing and validation and its description will be added to the D6.5 Report on the Implementation of the Joint Resource Registry (final) deliverable due at month 48. A preliminary screenshot of this interface is shown in Figures 5 and 6. Figure 5 shows the welcome page allowing browsing between the types and the research infrastructures (i.e. groups), Figure 6 shows an example of a resource details

Welcome to the PARTHENOS Resource Catalogue!

Here you will find data and other resources of the PARTHENOS Infrastructure (http://www.parthenos-project.eu) and its research community.

In particular, the catalogue contains products from different disciplines: Linguistic Studies, Humanities, Cultural Heritage, History, Archaeology and related fields.

All the products are accompanied with rich descriptions capturing general attributes, e.g. title and creator(s), as well as usage policies and licences.

Items Search

Insert keywords here 🔍

See All

PARTHENOS Resource Catalogue statistics

5	1	9	5
items	organisation	groups	types

Browse by Types

E29_Design_or_Procedure (1) E39_Actor (1) E55_Type (1) E70_Thing (1) E7_Activity (1)

See All Types

Browse by Groups

Ariadne (1) HUMANUM (1) EHRI (1) CLARIN (1) METASHARE (1)

See All Groups

Popular Formats

Popular Tags

PE34_Team PE25_RI_Consortium E74_Group PE28_Curation_Plan PE15_Data_EService PE18_dataset PE7_Data_Hosting_Service PE37_Protocol_type PE22_Persistent_Dataset PE5_Digital_Hosting_Ser... PE2_Hosting_Service D1_Digital_Object PE1_Service

gCUBE ckan

Figure 5: End-user Graphical User Interface welcome page

Figure 6: End-user Graphical User Interface resource details

.