

INFORMATION TECHNOLOGIES FOR SHIFT2RAIL

D1.3 – Semantic Discovery Engine

Due date of deliverable: 30/10/2017

Actual submission date: 20/11/2017

Leader of this Deliverable: CEFRIEL

Reviewed: Y

Document status		
Revision	Date	Description
1	26/09/2017	First issue
2	18/10/2017	Revision after ThalesPT comments
3	20/11/2017	Final Review

Project funded from the European Union's Horizon 2020 research and innovation programme		
Dissemination Level		
PU	Public	X
CO	Confidential, restricted under conditions set out in Model Grant Agreement	
CI	Classified, information as referred to in Commission Decision 2001/844/EC	

Start date of project: 01/05/2015

Duration: 36 months

EXECUTIVE SUMMARY

The semantic discovery engine of the IT²RAIL Interoperability framework aims at supporting the discovery and selection of assets descriptions stored in semantic registries through queries submitted by functional applications. The semantic discovery engine extends the IT²RAIL Semantic Assets Manager (described in details in D1.2) and enhances the discoverability and the reusability of its assets. Moreover, since the structured asset descriptions are expressed using first order logic, the semantic discovery engine can exploit inferred data to improve the quality of the discovery.

The solution defined by the IT²RAIL project for the development of the semantic discovery engine aims to improve and facilitate the sharing of data, limiting the risk of exposing business-sensitive data on the Web. The solution consists in defining parametric queries that can be invoked by external applications by calling an API. In this way, the access to exposed data is controlled and governed through authorisations to use specific APIs. To support this controlled solution to access data, the IT²RAIL Semantic Assets Manager manages a new asset type, called SPARQL-API Service that represents a service exposing an API for the execution of a parametric query.

This deliverable:

- Describe the new asset type SPARQL-API Service managed by the IT²RAIL Semantic Assets Manager (Section 2);
- Provide a detailed description of the architecture of the IT²RAIL Semantic Assets Manager enriched with components supporting the semantic discovery of RDF data (Section 3);
- Describe examples of parametric queries in order to show how they are managed within the IT²RAIL Semantic Assets Manager as SPARQL-API Service assets and how they can be used by external applications (Section 4).

TABLE OF CONTENTS

Executive Summary	2
List of Figures	4
List of TABLES	5
List of Abbreviations.....	6
1. Referenced documents	7
2. SPARQL-API Services.....	8
3. The Enhanced Architecture.....	9
3.1 The SPARQL – REST Server.....	10
4. Examples of parametric Queries	15
4.1 Query #1: Assets by Author.....	16
4.1.1 Template of Query #1	16
4.1.2 Swagger of Query #1	17
4.2 Query #2: Assets by Institution.....	19
4.2.1 Template of Query #2	19
4.2.2 Swagger of Query #2	20
4.3 Query #3: Expiring Assets.....	22
4.3.1 Template of Query #3	22
4.3.2 Swagger of Query #3	23
4.4 Query #4: Assets by Author and Date.....	25
4.4.1 Template of Query #4	25
4.4.2 Swagger of Query #4	26
4.5 Query #5: Assets by Author with Multiple Choice	28
4.5.1 Template of Query #5	28
4.5.2 Swagger of Query #5	29

LIST OF FIGURES

Figure 1: The form for editing a <i>SPARQL-API Service</i> asset.....	9
Figure 2: The enhanced architecture of the IT ² RAIL Semantic Assets Manager	10
Figure 3: The SPARQL – REST Server	11
Figure 4: An example of <i>SPARQL-API Service</i> asset description.....	13
Figure 5: UML Sequence Diagram showing the execution of a query through Exec API	15
Figure 6: SPARQL-API Service assets available in the IT ² RAIL Semantic Assets Manager.....	16
Figure 7: The SPARQL-API Service “Assets by Author” in the Assets Manager	18
Figure 8: Structured Swagger API description of the SPARQL-API Service representation of query “Assets by Author”.....	19
Figure 9: The SPARQL-API Service “Assets by Institution” in the Assets Manager	21
Figure 10: Structured Swagger representation of query “Assets by Institution”	22
Figure 11: The SPARQL-API Service “Expiring Assets” in the Assets Manager	24
Figure 12: Structured Swagger representation of query “Expiring Assets”	25
Figure 13: The SPARQL-API Service “Assets by Author and Date” in the Assets Manager	27
Figure 14: Structured Swagger representation of query “Assets by Author and Date”	28
Figure 15: The SPARQL-API Service “Assets by Author with Multiple Choice” in the Assets Manager	30
Figure 16: Structured Swagger representation of query “Assets by Author with Multiple Choice” ..	31

LIST OF TABLES

Table 1: An example of SPARQL Template.....	8
Table 2: The Single Swagger Template used by the SPARQL-REST Server.....	12
Table 3: The Multiple Swagger Template used by the SPARQL-REST Server.....	14
Table 4: SPARQL Template of the query “Assets by Author”.....	17
Table 5: SPARQL Template of the query “Asset by Institution”.....	20
Table 6: SPARQL Template of the query “Expiring Assets”.....	23
Table 7: SPARQL Template of the query “Assets by Author and Date”.....	26
Table 8: SPARQL Template of the query “Assets by Author with Multiple Choice”	29

LIST OF ABBREVIATIONS

API	A pplication P rogramming I nterface
JSON	J ava S cript O bject N otation
RDBMS	R ational D atabase M anagement S ystem
RDF	R esource D escription F ramework
REST	R Epresentational S tate T ransfer
SPARQL	S PARQL Protocol and RDF Query Language
UML	U nified M odeling L anguage
UNDEF	U ndefined

1. REFERENCED DOCUMENTS

This section lists the document reference number, title, revision, and date of all documents referenced in the specifications document.

Reference Number	Title	Revision	Date
D1.2	Semantic Web Services Registry	2	17/10/2017

2. SPARQL-API SERVICES

In order to support the discovery of RDF data, the IT²RAIL Semantic Assets Manager (described in D1.2 Semantic Web Services Registry) manages a new asset type, called **SPARQL-API Service** (SPARQLEST). This asset represents a service exposing an API for the execution of a parametric query. Basically, by invoking the API and specifying the parameters, the service returns the results of a SPARQL query on RDF data.

The Figure 1 below shows the form allowing the editing of a SPARQL-API Service asset description. The form is divided into two main sections. In the *Overview* section, the following attributes can be specified:

- **Name**: the name given to the SPARQL-API Service;
- **Version**: a specific historical description of the asset;
- **Author, author email** and **institution**: the entity responsible for making the asset available and his/her contacts;
- **Description**: a brief description of the asset;
- **Expected validity**: the date until when the asset is supposed to be valid.

The *Content* section contains the **SPARQL Template** and the **Swagger API description** (see Figure 1).

The SPARQL Template is a text box to be filled with a parametric query template in SPARQL. An example of template is shown in Table 1 where *s*, *p* and *o* are the parameters to be quantified with values or UNDEF in order to make the query executable.

```
SELECT ?s ?p ?o
WHERE { ?s ?p ?o
VALUES (?s ?p ?o) {{{s or 'UNDEF'}} {{p or 'UNDEF'}} {{o or 'UNDEF'}} }}
```

Table 1: An example of SPARQL Template

SPARQL Templates are defined using the Jinja2 syntax¹. A template contains variables and/or expressions, which are replaced with values when a template is rendered; and tags, which control the logic of the template. The default Jinja2 delimiters are the following:

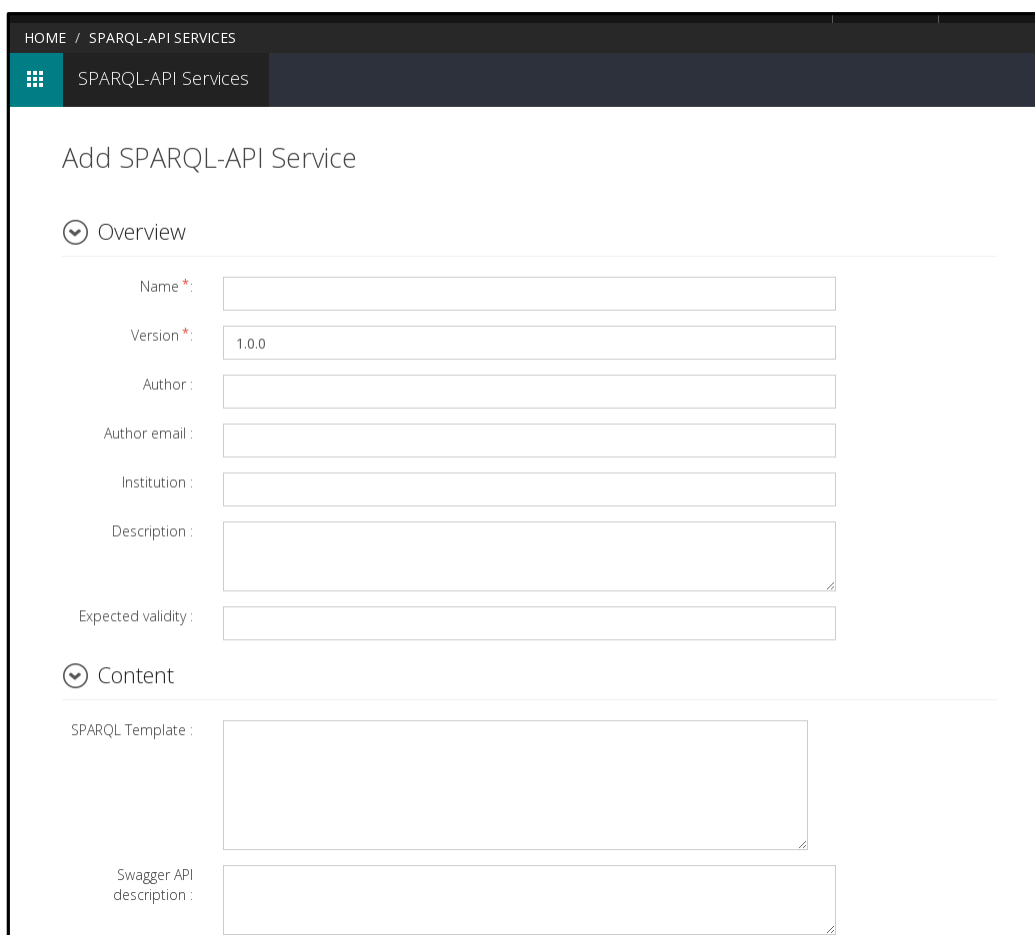
- {% ... %} for Statements;
- {{ ... }} for Expressions to print to the template output;
- {# ... #} for Comments not included in the template output;
- # ... ## for Line Statements;
- {{ foo }}: prints a variable name;

¹ jinja.pocoo.org

- `{{ foo.bar }}` or `{{ foo['bar'] }}`: prints an attribute of a variable.

For the list of the control structures and the available filters, please refer to Jinja2 documentation².

In addition to the SPARQL Template, the *Content* section of the SPARQL-API Service description contains the Swagger API description that defines the API to be invoked to execute the parametric SPARQL query. This description is automatically created by the IT²RAIL Semantic Assets Manager invoking the SPARQL-REST Server (see Section 3.1).



HOME / SPARQL-API SERVICES

SPARQL-API Services

Add SPARQL-API Service

Overview

Name *:

Version *:

Author :

Author email :

Institution :

Description :

Expected validity :

Content

SPARQL Template :

Swagger API description :

Figure 1: The form for editing a SPARQL-API Service asset

3. THE ENHANCED ARCHITECTURE

The Figure 2 below represents the architecture of the IT²RAIL Semantic Assets Manager enriched with components supporting the semantic discovery of RDF data. In addition to what has been described in D1.2 Semantic Web Services Registry, the architecture of the IT²RAIL Semantic Assets Manager presents a SPARQL-REST endpoint supporting the user in the selection of SPARQL queries. The SPARQL-REST endpoint invokes the external SPARQL-REST Server (see Section 3.1) through APIs. The SPARQL-REST Server executes the SPARQL queries on the RDF repository

² jinja.pocoo.org/docs/2.9/templates/

populated by the transformation server (see D1.2 Semantic Web Services Registry) and/or on external RDF repository using the provided query parameters.

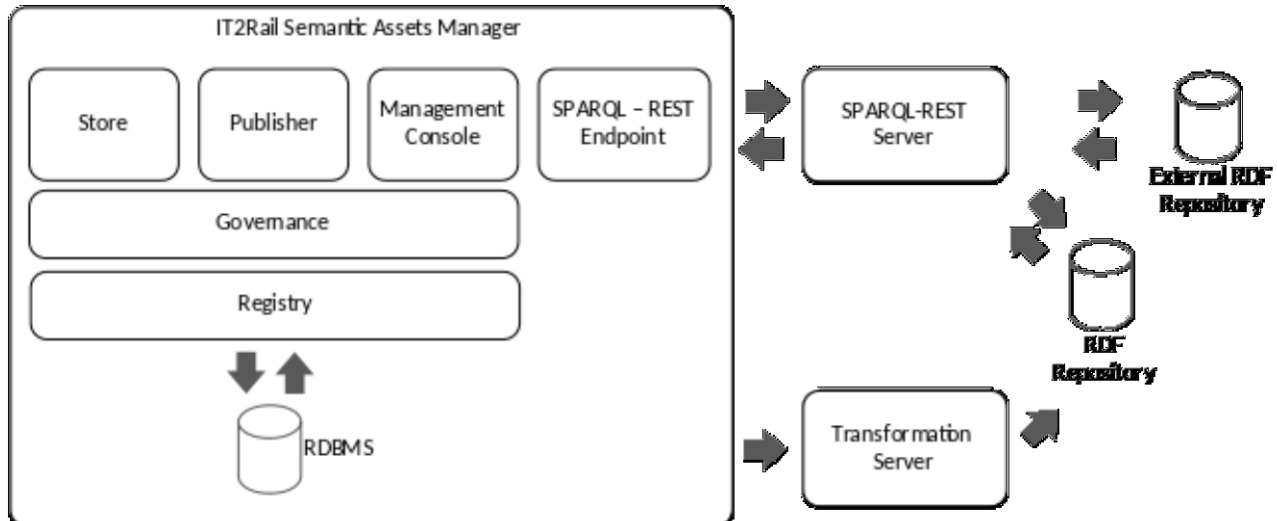


Figure 2: The enhanced architecture of the IT²RAIL Semantic Assets Manager

3.1 THE SPARQL – REST SERVER

As shown in the Figure 3 below, the SPARQL-REST Server exposes three APIs. The *Get Swagger* API is used to create the Swagger API description to be included into the form of the related SPARQL-API Service (See Figure 1).

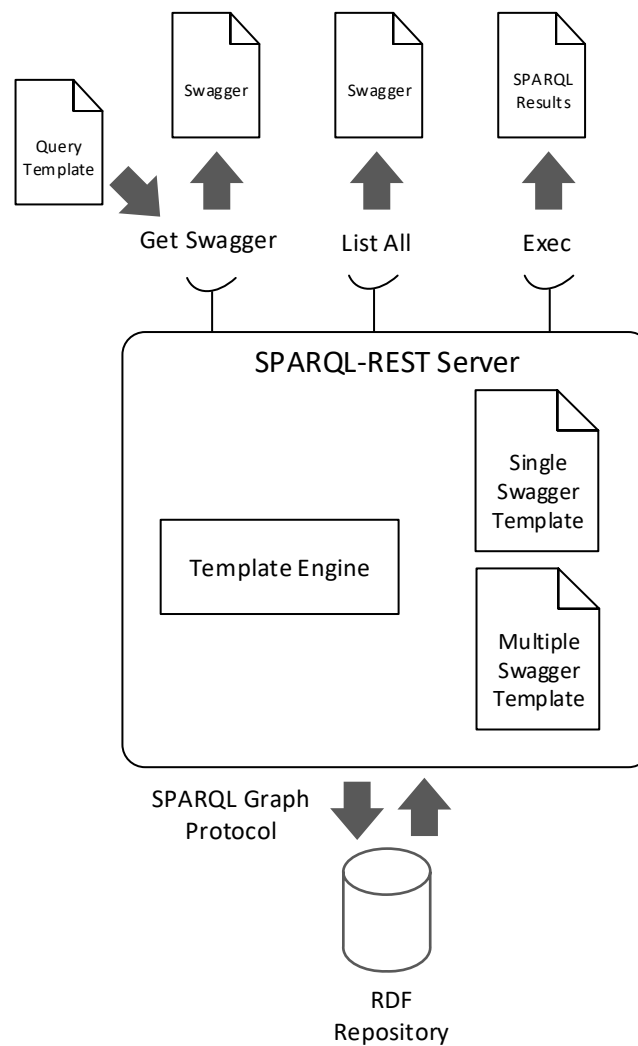


Figure 3: The SPARQL – REST Server

Basically, when the status of SPARQL-API Service asset is promoted to “Published” (See Asset lifecycle in D1.2 Semantic Web Services Registry), the template is sent to the SPARQL-REST Server through the *Get Swagger* API. The SPARQL-REST Server uses the Single Swagger Template (see Table 2) to define and associate an API to the SPARQL-API Service and create its Swagger description. This description is sent back to the SPARQL-REST endpoint and linked to the SPARQL-API Service asset description.

The resulting SPARQL-API Service asset description related to the SPARQL template proposed in Table 1 is shown in the Figure 4.

```
{
  "swagger": "2.0",
  "info": {
    "title": "SPARQLest {{query_name}} API",
    "version": "1.0"
  },
  "paths": {
    "/discovery/exec/{{query_name}}": {
      "get": {
        "summary": "Execute SPARQLest query {{query_name}}",
        "description": "",
        "parameters": [
          {
            "name": "parameters",
            "in": "body",
            "description": "API parameters",
            "required": true,
            "schema": {
              "$ref": "#/definitions/APIParams"
            }
          },
          {
            "name": "wso2_session_id",
            "in": "body",
            "description": "WSO2 Session id obtained with
https://docs.wso2.com/display/ES200/Authenticate+Credentials+via+the+Store+REST+API",
            "required": true,
            "schema": {
              "type": "string"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "SPARQL query results",
            "schema": {
              "type": "string"
            }
          }
        }
      }
    }
  },
  "definitions": {
    "APIParams": {
      "type": "object",
      "properties": {
        {%for param in variables%}
          "{{param}}": {
            "type": "string"
          }{{ " ", " if not loop.last }}
        {%endfor%}
      }
    }
  }
}
```


Table 2: The Single Swagger Template used by the SPARQL-REST Server

HOME / SPARQL-API SERVICES / EXAMPLE2 / 1.0.0

SPARQL-API Services

EDIT
DELETE
LIFECYCLE
VERSION
ASSOCIATIONS
PERMISSIONS

0
OPTIONS



Example2
Version : 1.0.0
SparqlApiLifeCycle : Published
Thu, 27 Jul 2017 08:34:08 GMT

Show Dependencies

Overview

Name : Example2
Version : 1.0.0
Author : Alessio Carenini
Author email : carenini@gmail.com
Institution : Cefriel
Description :
Expected validity :

Content

SPARQL Template : SELECT ?s ?p ?o WHERE { ?s ?p ?o . VALUES (?s ?p ?o) { ({ {s or 'UNDEF'}} {p or 'UNDEF'}} {o or 'UNDEF'}} } }
Swagger API description : { "definitions": { "APIParams": { "properties": { "o": { "type": "string" }, "p": { "type": "string" }, "s": { "type": "string" } }, "type": "object" }, "info": { "title": "SPARQLest example2-1-0-0 API", "version": "1.0" }, "paths": { "/discovery/exec/example2-1-0-0": { "get": { "description": "", "parameters": [{ "description": "API parameters", "in": "body", "name": "api_params", "required": true, "schema": { "\$ref": "#/definitions/APIParams" } }], "responses": { "200": { "description": "SPARQL query results", "schema": { "type": "string" } } }, "summary": "Execute SPARQLest query example2-1-0-0" } } }, "swagger": "2.0" }

Figure 4: An example of SPARQL-API Service asset description.

The *List All* API exposed by the SPARQL–REST Server supports users who need to know the full set of APIs that can be invoked for e.g., design and implement new applications. Basically, a user through the SPARQL–REST Endpoint can invoke the *List All* API of the SPARQL–REST Server. The SPARQL–REST Server retrieves from the IT2RAIL Semantic Assets Manager all the SPARQL-API Service asset descriptions available for that user. Then, it extracts the Swagger API description from each retrieved asset and uses the Multiple Swagger Template (See Table 3) to create a single Swagger description of all the queries that can be invoked by the user through APIs.

```
{
  "swagger": "2.0",
  "info": {
    "title": "SPARQLest APIs",
    "version": "1.0"
  },
  "paths": {
    {%for query_name in queries %}
    "/discovery/exec/{{query_name}}": {
      "get": {
        "summary": "Execute SPARQLest query {{query_name}}",
        "description": "",
        "parameters": [
          {
            "name": "parameters",
            "in": "body",
            "description": "API parameters",
            "required": true,
            "schema": {
              "$ref": "#/definitions/APIParams_{{query_name}}"
            }
          },
          {
            "name": "wso2_session_id",
            "in": "body",
            "description": "WSO2 Session id obtained with
https://docs.wso2.com/display/ES200/Authenticate+Credentials+via+the+Store+REST+API",
            "required": true,
            "schema": {
              "type": "string"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "SPARQL query results",
            "schema": {
              "type": "string"
            }
          }
        }
      }
    }
    }{{ "," if not loop.last }}
  {%endfor%}
}
"definitions": {
  {%for query_name in queries %}
  "APIParams_{{query_name}}": {
    "type": "object",
    "properties": {
      {%for param in variables%}
      "{{param}}": {
        "type": "string"
      }{{ "," if not loop.last }}
      {%endfor%}
    }
  }
  }{{ "," if not loop.last }}
}
```

Table 3: The Multiple Swagger Template used by the SPARQL-REST Server

The *Exec* API exposed by the SPARQL-REST Server supports the execution of SPARQL queries on RDF data. Basically, the user, by means of the SPARQL – REST Endpoint, selects a parametric query and sets its parameters. The selected parametric query and the parameters (in JSON format) are sent to the SPARQL – REST server invoking the *Exec* API. The UML Sequence Diagram in the

Figure 5 shows the interaction between SPARQL – REST Endpoint, SPARQL – REST Server, Assets Manager and RDF Repository for the execution of a parametric query.

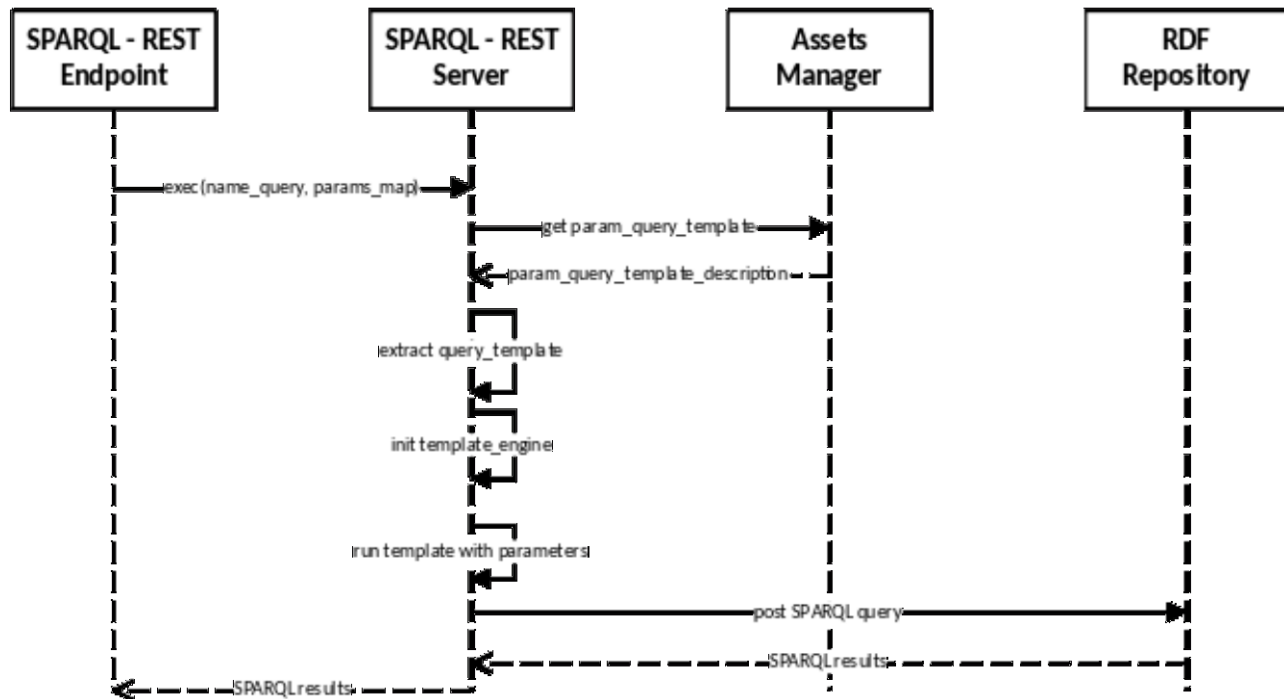


Figure 5: UML Sequence Diagram showing the execution of a query through Exec API

4. EXAMPLES OF PARAMETRIC QUERIES

In this section, examples of parametric queries are proposed in order to show how they are managed within the IT²RAIL Semantic Assets Manager as SPARQL-API Service assets. Each parametric query is described in terms of:

- A textual description in natural language;
- Its SPARQL template;
- The associated Swagger description generated by invoking the *Get Swagger* API of the SPARQL-REST Server;
- Its SPARQL-API Service description managed by the Assets Manager.

As documented in D1.2 Semantic Web Services Registry, all the asset types managed by the Assets Manager present a common set of metadata (*name*, *version*, *author*, *author e-mail*, *institution*, *description* and *expected validity*) in the *Overview* section of their descriptions. The usage of the same metadata set for all the asset types supports the comparison between different assets through queries like the ones proposed in this section.

The SPARQL-API Service assets related to the parametric queries described in the following are available in the Store of the IT²RAIL Semantic Assets Manager (See Figure 6).

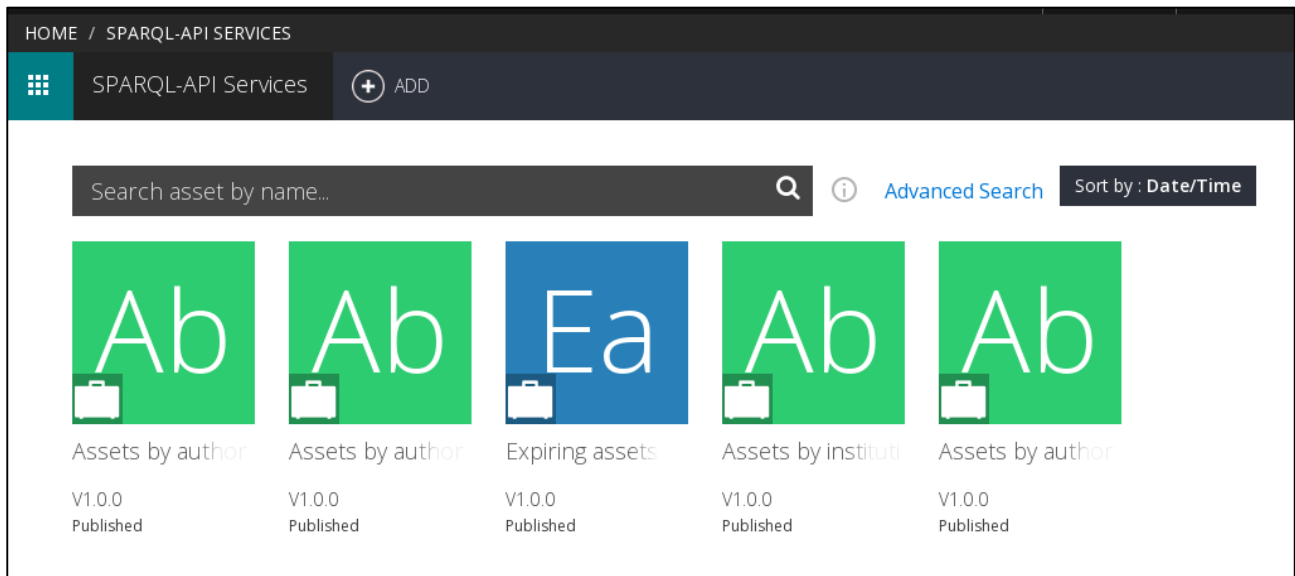


Figure 6: SPARQL-API Service assets available in the IT²RAIL Semantic Assets Manager

4.1 QUERY #1: ASSETS BY AUTHOR

The first proposed parametric query, named “Assets by author”, is the following:

“Show me all the assets provided by the author X”

The query acts on the metadata *author* and targets *all the asset types*.

4.1.1 Template of Query #1

The Table 4 shows the SPARQL Template of the query “Assets by Author”. The query has a parameter identified by the variable *author* that is quantified by the caller or set to UNDEF (line 9 in Table 4). The query returns *asset_id*, *asset_url* and *asset_title* of each asset (line 1 in Table 4) that satisfies the condition of having the internal representation of the author (*dcat:contactPoint.vcard:fn*) equals to the value assumed by the variable *author* (lines 6-7 in Table 4).

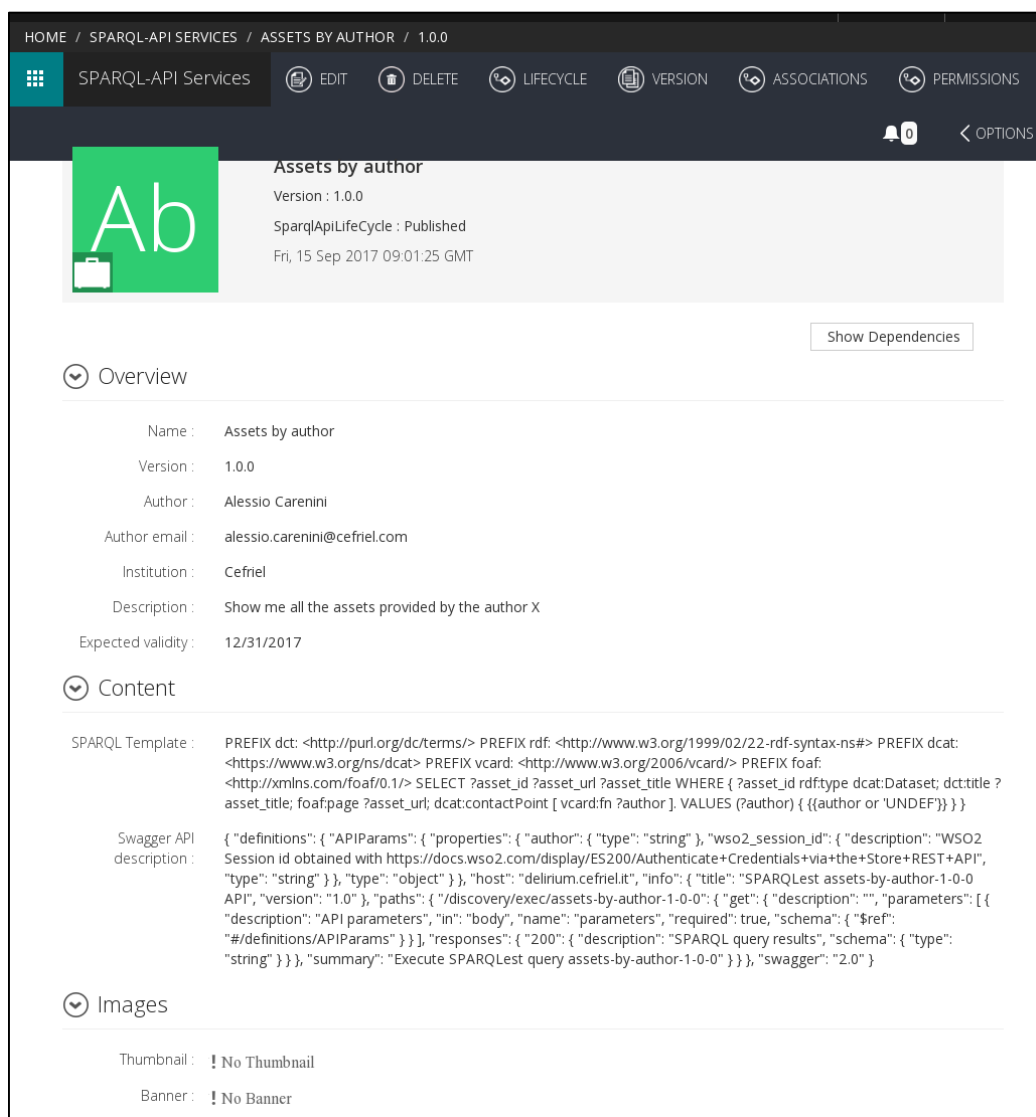

```
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dcat: <https://www.w3.org/ns/dcat>
PREFIX vcard: <http://www.w3.org/2006/vcard/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

```
1 SELECT ?asset_id ?asset_url ?asset_title
2 WHERE {
3   ?asset_id rdf:type dcat:Dataset;
4   dct:title ?asset_title;
5   foaf:page ?asset_url;
6   dcat:contactPoint [
7     vcard:fn ?author
8   ].
9 VALUES (?author) { {{author or 'UNDEF'}} }
10 }
```

Table 4: SPARQL Template of the query “Assets by Author”

4.1.2 Swagger of Query #1

The Swagger description of the API to be invoke for executing the query “Assets by Author” on the RDF Repository is automatically obtained by the IT²RAIL Semantic Assets Manager by invoking the *Get Swagger* API of the SPARQL–REST Server. The Figure 7 shows the complete description of the SPARQL-API Service “Assets by Author” managed by the Assets Manager that contains the metadata description of the asset, its SPARQL template and the related Swagger API description.



HOME / SPARQL-API SERVICES / ASSETS BY AUTHOR / 1.0.0

SPARQL-API Services EDIT DELETE LIFECYCLE VERSION ASSOCIATIONS PERMISSIONS

Assets by author
Version : 1.0.0
SparqlApiLifeCycle : Published
Fri, 15 Sep 2017 09:01:25 GMT

Show Dependencies

Overview

Name : Assets by author
Version : 1.0.0
Author : Alessio Carenini
Author email : alessio.carenini@cefriel.com
Institution : Cefriel
Description : Show me all the assets provided by the author X
Expected validity : 12/31/2017

Content

SPARQL Template : PREFIX dct: <http://purl.org/dc/terms/> PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX dcat: <https://www.w3.org/ns/dcat> PREFIX vcard: <http://www.w3.org/2006/vcard/> PREFIX foaf: <http://xmlns.com/foaf/0.1/> SELECT ?asset_id ?asset_url ?asset_title WHERE { ?asset_id rdf:type dcat:Dataset; dct:title ?asset_title; foaf:page ?asset_url; dcat:contactPoint [vcard:fn ?author]. VALUES (?author) { {{author or 'UNDEF'}} } }

Swagger API description : { "definitions": { "APIParams": { "properties": { "author": { "type": "string" }, "wso2_session_id": { "description": "WSO2 Session id obtained with https://docs.wso2.com/display/ES200/Authenticate+Credentials+via+the+Store+REST+API", "type": "string" }, "type": "object" }, "host": "delirium.cefriel.it", "info": { "title": "SPARQLest assets-by-author-1-0-0 API", "version": "1.0", "paths": { "/discovery/exec/assets-by-author-1-0-0": { "get": { "description": "", "parameters": [{ "description": "API parameters", "in": "body", "name": "parameters", "required": true, "schema": { "\$ref": "#/definitions/APIParams" } }], "responses": { "200": { "description": "SPARQL query results", "schema": { "type": "string" } }, "summary": "Execute SPARQLest query assets-by-author-1-0-0" }, "swagger": "2.0" }

Images

Thumbnail : ! No Thumbnail
Banner : ! No Banner

Figure 7: The SPARQL-API Service “Assets by Author” in the Assets Manager

The Figure 8 shows a structured representation of the Swagger API description of the SPARQL-API Service “Assets by Author”. It shows (1) the path `/discovery/exec/assets-by-author-1-0-0` associated to the API, (2) the required parameters that are the `author` and the `wso2_session_id` (i.e., an `id` identifying the requestor), and (3) the output of the API invocation (i.e., a string containing the query results).



Figure 8: Structured Swagger API description of the SPARQL-API Service representation of query “Assets by Author”

4.2 QUERY #2: ASSETS BY INSTITUTION

The second proposed parametric query, named “Assets by Institution”, is the following:

“How many assets have been provided by the authors from the institution X”

This query acts on the metadata *institution* and targets *all the asset types*. This is an example of query to define statistics on the available assets.

4.2.1 Template of Query #2

The Table 5 shows the SPARQL Template of the query “Assets by Institution”. The query has a parameter identified by the variable *institution* that is quantified by the caller or set to UNDEF (line 10 in Table 5). The query returns the *count* of the distinct assets (line 1 in Table 5) that satisfies the condition of having the internal representation of the author institution (*dcat: publisher.name*) equals to the value assumed by the variable *institution* (lines 6-8 in Table 5).

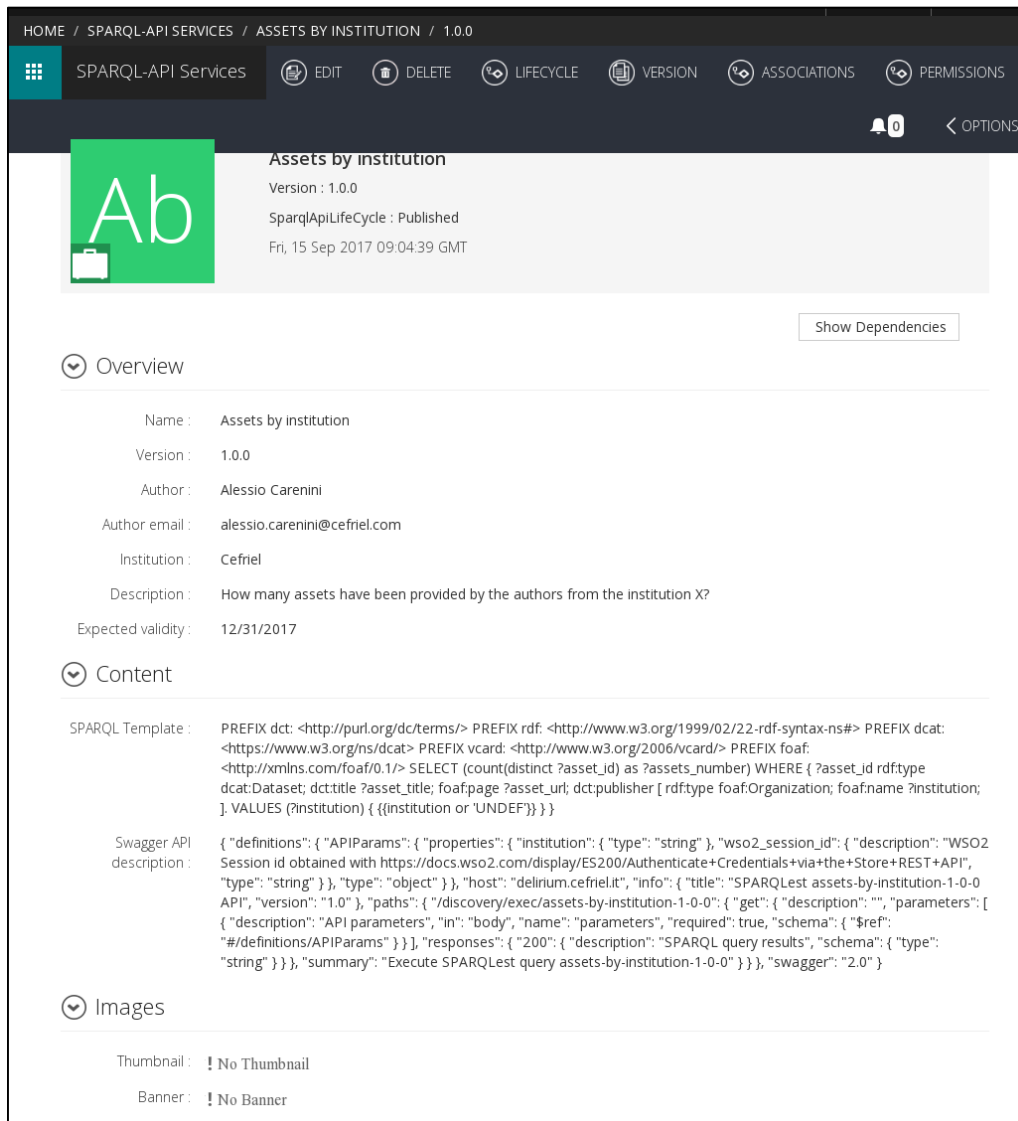
```
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dcat: <https://www.w3.org/ns/dcat>
PREFIX vcard: <http://www.w3.org/2006/vcard/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

```
1 SELECT (count(distinct ?asset_id) as ?assets_number)
2 WHERE {
3   ?asset_id rdf:type dcat:Dataset;
4   dct:title ?asset_title;
5   foaf:page ?asset_url;
6   dct:publisher [
7     rdf:type foaf:Organisation;
8     foaf:name ?institution;
9   ].
10 VALUES (?institution) { {{institution or 'UNDEF'}} }
11 }
```

Table 5: SPARQL Template of the query “Asset by Institution”

4.2.2 Swagger of Query #2

The shows the complete description of the SPARQL-API Service “Assets by Institution” managed by the Assets Manager that contains the metadata description, its SPARQL template and the related Swagger API description.



HOME / SPARQL-API SERVICES / ASSETS BY INSTITUTION / 1.0.0

SPARQL-API Services EDIT DELETE LIFECYCLE VERSION ASSOCIATIONS PERMISSIONS

Assets by institution
Version : 1.0.0
SparqlApiLifeCycle : Published
Fri, 15 Sep 2017 09:04:39 GMT

Show Dependencies

Overview

Name : Assets by institution
Version : 1.0.0
Author : Alessio Carenini
Author email : alessio.carenini@cefriel.com
Institution : Cefriel
Description : How many assets have been provided by the authors from the institution X?
Expected validity : 12/31/2017

Content

SPARQL Template : PREFIX dct: <http://purl.org/dc/terms/> PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX dcat: <https://www.w3.org/ns/dcat> PREFIX vcard: <http://www.w3.org/2006/vcard/> PREFIX foaf: <http://xmlns.com/foaf/0.1/> SELECT (count(distinct ?asset_id) as ?assets_number) WHERE { ?asset_id rdf:type dcat:Dataset; dct:title ?asset_title; foaf:page ?asset_url; dct:publisher [rdf:type foaf:Organization; foaf:name ?institution;]. VALUES (?institution) { {(institution or "UNDEF")} }

Swagger API description : { "definitions": { "APIParams": { "properties": { "institution": { "type": "string" }, "wso2_session_id": { "description": "WSO2 Session id obtained with https://docs.wso2.com/display/ES200/Authenticate+Credentials+via+the+Store+REST+API", "type": "string" }, "type": "object" }, "host": "delirium.cefriel.it", "info": { "title": "SPARQLest assets-by-institution-1-0-0 API", "version": "1.0" }, "paths": { "/discovery/exec/assets-by-institution-1-0-0": { "get": { "description": "", "parameters": [{ "description": "API parameters", "in": "body", "name": "parameters", "required": true, "schema": { "\$ref": "#/definitions/APIParams" } }], "responses": { "200": { "description": "SPARQL query results", "schema": { "type": "string" } } }, "summary": "Execute SPARQLest query assets-by-institution-1-0-0" } } }, "swagger": "2.0" }

Images

Thumbnail : ! No Thumbnail
Banner : ! No Banner

Figure 9: The SPARQL-API Service “Assets by Institution” in the Assets Manager

The Figure 10 shows a structured representation of the Swagger API description of the SPARQL-API Service “Assets by Institution”. It represents (1) the path */discovery/exec/assets-by-institution-1-0-0* associated to the API, (2) the required parameters that are the *institution* and the *wso2_session_id* (i.e., an id identifying the requestor), and (3) the output of the API invocation (i.e., a string containing the query results).

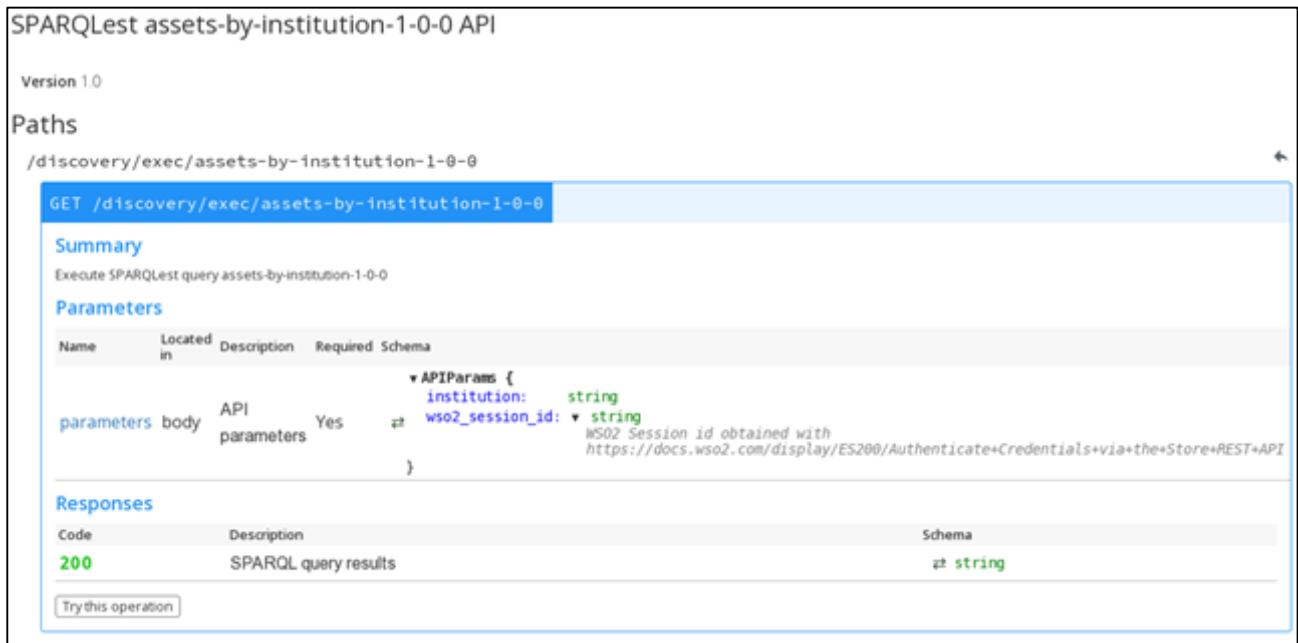


Figure 10: Structured Swagger representation of query “Assets by Institution”

4.3 QUERY #3: EXPIRING ASSETS

The third proposed parametric query, named “Expiring Assets”, is the following:

“Show me all the assets that will expire after date X”

This query acts on the metadata *expected validity* and targets *all the asset types*. This query is useful for managerial activity.

4.3.1 Template of Query #3

The Table 6 shows the SPARQL Template of the query “Expiring Assets”. The query has a parameter identified by the variable *validity_date* that can be quantified by the caller (line 7 in Table 6). The query returns the *asset_id* of each asset (line 1 in Table 6) that satisfies the condition of having the internal representation of the expiring date (*dcat:temporal.endDate*) less than or equal to the value assumed by the variable *validity_date*. The condition is expressed using the construct FILTER that filters out all assets that are not expiring (lines 6-7 in Table 6).

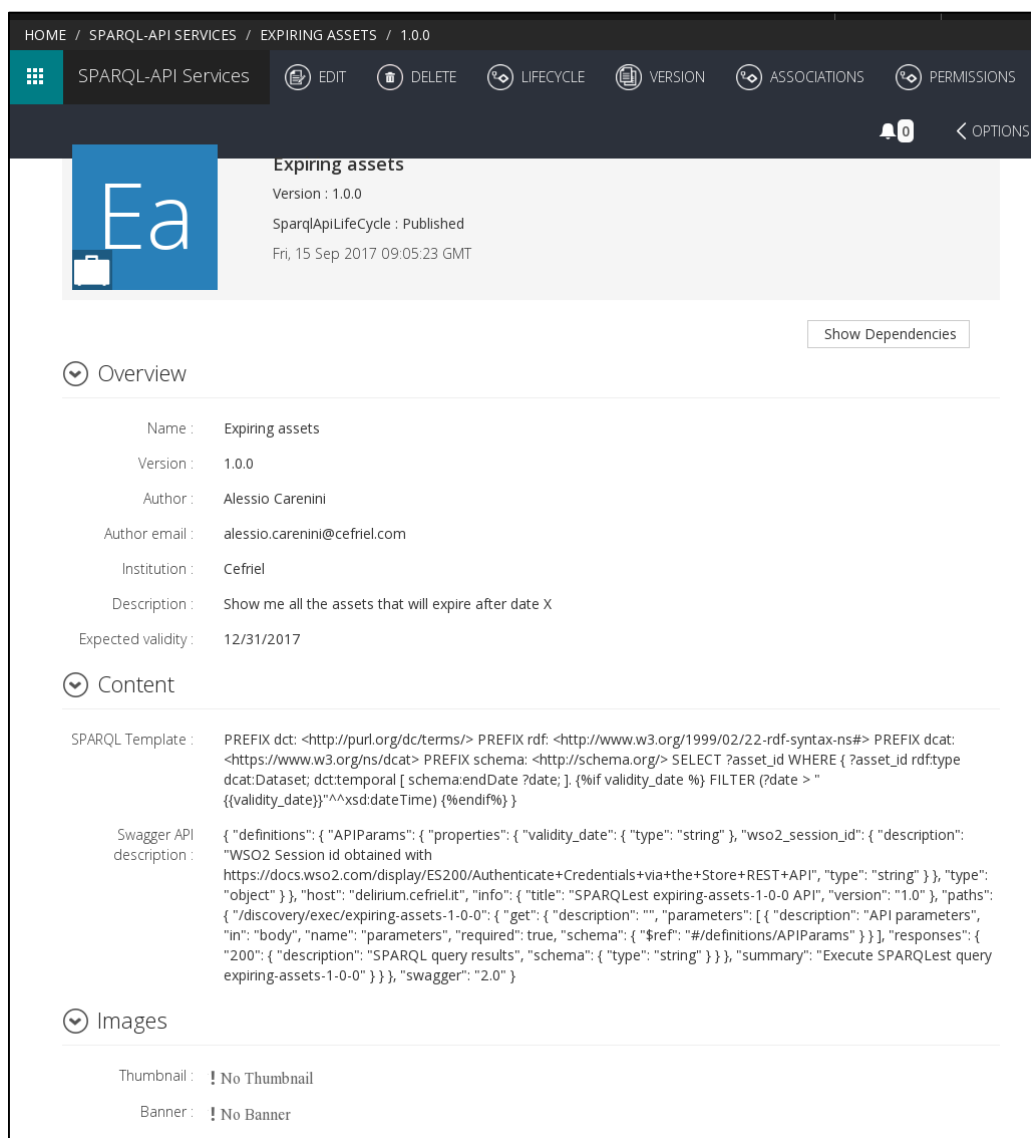
```
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dcat: <https://www.w3.org/ns/dcat>
PREFIX schema: <http://schema.org/>

1 SELECT ?asset_id
2 WHERE {
3   ?asset_id rdf:type dcat:Dataset;
4   dct:temporal [
5     schema:endDate ?date;
6   ].
7   {%if validity_date %}
8     FILTER (?date > "{{validity_date}}"^^xsd:dateTime)
9   {%endif%}
10 }
```

Table 6: SPARQL Template of the query “Expiring Assets”

4.3.2 Swagger of Query #3

The Figure 11 shows the complete description of the SPARQL-API Service “Expiring Assets” managed by the Assets Manager that contains the metadata description, its SPARQL template and the related Swagger API description.



HOME / SPARQL-API SERVICES / EXPIRING ASSETS / 1.0.0

SPARQL-API Services EDIT DELETE LIFECYCLE VERSION ASSOCIATIONS PERMISSIONS

Expiring assets
Version : 1.0.0
SparqlApiLifeCycle : Published
Fri, 15 Sep 2017 09:05:23 GMT

Show Dependencies

Overview

Name : Expiring assets
Version : 1.0.0
Author : Alessio Carenini
Author email : alessio.carenini@cefriel.com
Institution : Cefriel
Description : Show me all the assets that will expire after date X
Expected validity : 12/31/2017

Content

SPARQL Template : PREFIX dct: <http://purl.org/dc/terms/> PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX dcat: <https://www.w3.org/ns/dcat> PREFIX schema: <http://schema.org/> SELECT ?asset_id WHERE { ?asset_id rdf:type dcat:Dataset; dct:temporal [schema:endDate ?date;], {%if validity_date %} FILTER (?date > {%validity_date%})^^xsd:dateTime) {%endif%} }

Swagger API description : { "definitions": { "APIParams": { "properties": { "validity_date": { "type": "string" }, "wso2_session_id": { "description": "WSO2 Session id obtained with https://docs.wso2.com/display/ES200/Authenticate+Credentials+via+the+Store+REST+API", "type": "string" }, "type": "object" }, "host": "delirium.cefriel.it", "info": { "title": "SPARQLest expiring-assets-1-0-0 API", "version": "1.0" }, "paths": { "/discovery/exec/expiring-assets-1-0-0": { "get": { "description": "", "parameters": [{ "description": "API parameters", "in": "body", "name": "parameters", "required": true, "schema": { "\$ref": "#/definitions/APIParams" } },], "responses": { "200": { "description": "SPARQL query results", "schema": { "type": "string" } }, "summary": "Execute SPARQLest query expiring-assets-1-0-0" } }, "swagger": "2.0" }

Images

Thumbnail : ! No Thumbnail
Banner : ! No Banner

Figure 11: The SPARQL-API Service “Expiring Assets” in the Assets Manager

The Figure 12 shows the structured representation of the Swagger API for “Expiring Assets”. It represents (1) the path `/discovery/exec/expiring-assets-1-0-0` associated to the API, (2) the required parameters that are the *validity date* and the *wso2_session_id* (i.e., an *id* identifying the requestor), and (3) the output of the API invocation (i.e., a string containing the query results).

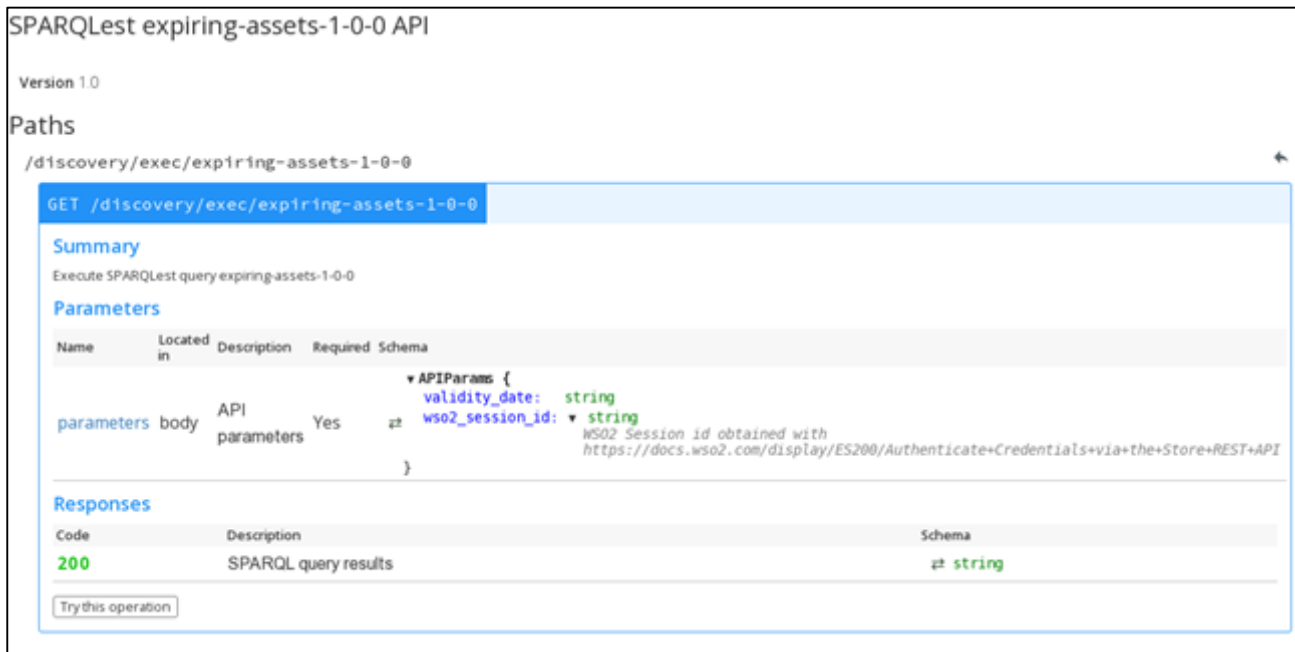


Figure 12: Structured Swagger representation of query “Expiring Assets”

4.4 QUERY #4: ASSETS BY AUTHOR AND DATE

The fourth proposed parametric query, named “Assets by Author and Date”, is the following:

*“Show me all the assets of type **X** provided by the author **Y** in the time period **Z**”*

This query presents multiple parameters and acts on (i) the metadata *author* available in the asset descriptions and (ii) on the metadata *asset type* and publishing date that are automatically created by the asset manager at publishing time. This query is useful for defining detailed statistics on the available assets.

4.4.1 Template of Query #4

The Table 7 shows the SPARQL Template of the query “Assets by Author and Date”. The query has four parameters identified by the variables *asset_type*, *author*, *period_start* and *period_end* that are quantified by the caller or set to UNDEF (lines 9-11, 13 and 16 in Table 7). The query returns the *asset_id*, *asset_type*, *author* and *publishing_date* of each asset (line 1 in Table 7) that satisfies the conditions of having:

- the internal representation of the author (*dcat:contactPoint.vcard:fn*) equals to the value assumed by the variable *author* (lines 5-6 in Table 7);
- the internal representation of the asset type (*dct:type*) equals to the value assumed by the variable *asset_type* (line 4 in Table 7);
- the internal representation of the publishing_date (*dcat:issued*) in the time period defined by *start_period* and *end_period*. The condition is expressed using the construct FILTER (lines 14-17 in Table 7).

```

PREFIX dct: <http://purl.org/dc/terms/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dcat: <https://www.w3.org/ns/dcat>
PREFIX vcard: <http://www.w3.org/2006/vcard/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

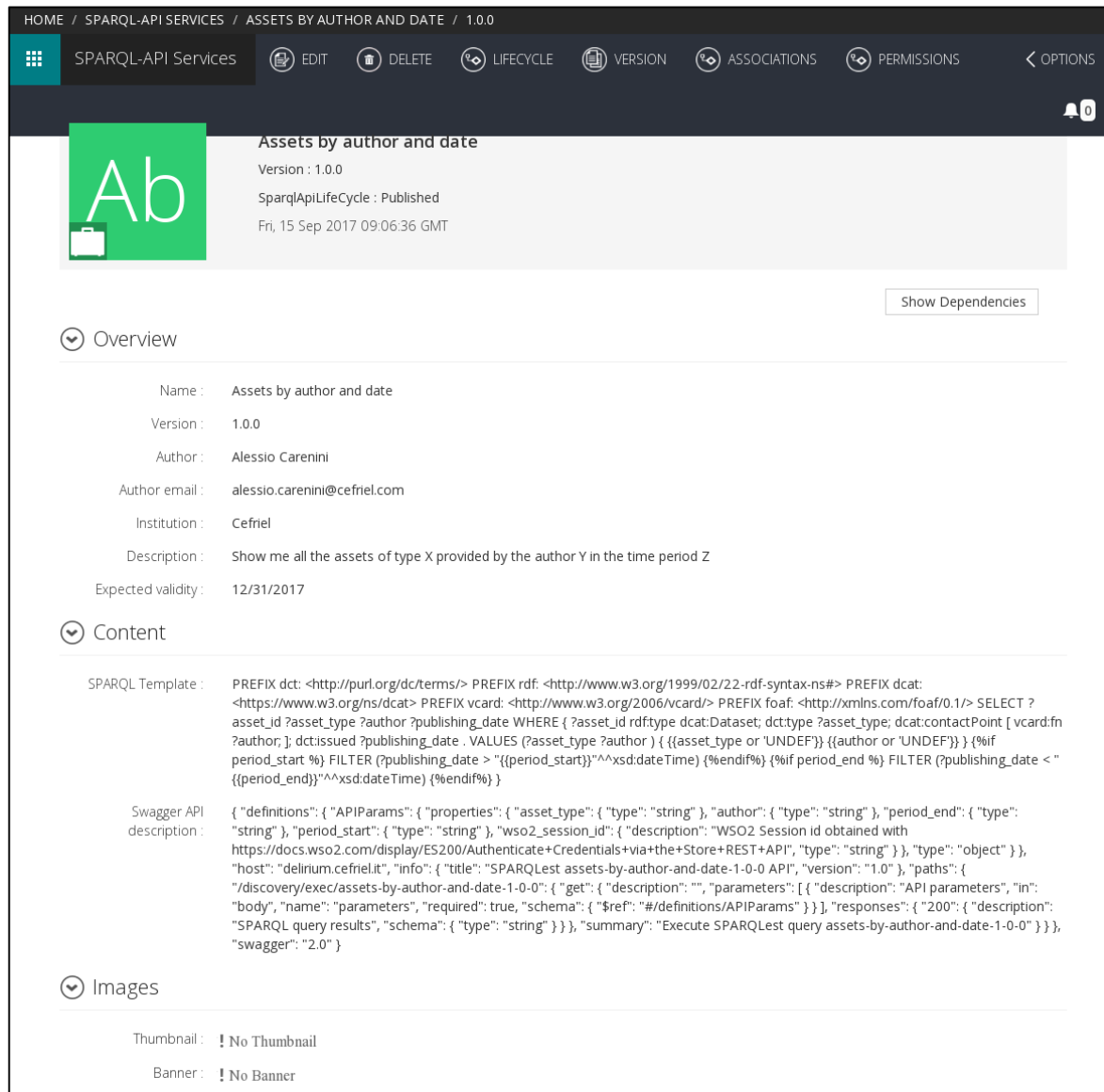
1 SELECT ?asset_id ?asset_type ?author ?publishing_date
2 WHERE {
3   ?asset_id rdf:type dcat:Dataset;
4     dct:type ?asset_type;
5     dcat:contactPoint [
6       vcard:fn ?author;
7     ];
8     dct:issued ?publishing_date
9   VALUES (?asset_type ?author) {
10     {{asset_type or 'UNDEF'}}
11     {{author or 'UNDEF'}}
12   }
13   {%if period_start %}
14     FILTER (?publishing_date > "{{period_start}}"^^xsd:dateTime)
15   {%endif%}
16   {%if period_end %}
17     FILTER (?publishing_date < "{{period_end}}"^^xsd:dateTime)
18   {%endif%}
19 }

```

Table 7: SPARQL Template of the query “Assets by Author and Date”

4.4.2 Swagger of Query #4

The Figure 13 shows the complete description of the SPARQL-API Service “Assets by Authors and Date” managed by the Assets Manager that contains the metadata description, its SPARQL template and the related Swagger API description.



HOME / SPARQL-API SERVICES / ASSETS BY AUTHOR AND DATE / 1.0.0

SPARQL-API Services EDIT DELETE LIFECYCLE VERSION ASSOCIATIONS PERMISSIONS OPTIONS

Assets by author and date
Version : 1.0.0
SparqlApiLifeCycle : Published
Fri, 15 Sep 2017 09:06:36 GMT

Show Dependencies

Overview

Name : Assets by author and date
Version : 1.0.0
Author : Alessio Carenini
Author email : alessio.carenini@cefriel.com
Institution : Cefriel
Description : Show me all the assets of type X provided by the author Y in the time period Z
Expected validity : 12/31/2017

Content

SPARQL Template : PREFIX dct: <http://purl.org/dc/terms/> PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX dcat: <https://www.w3.org/ns/dcat> PREFIX vcard: <http://www.w3.org/2006/vcard/> PREFIX foaf: <http://xmlns.com/foaf/0.1/> SELECT ? asset_id ?asset_type ?author ?publishing_date WHERE { ?asset_id rdf:type dcat:Dataset; dct:type ?asset_type; dcat:contactPoint [vcard:fn ?author;]; dct:issued ?publishing_date . VALUES (?asset_type ?author) { { {asset_type or 'UNDEF'} } { {author or 'UNDEF'} } } { %if period_start % } FILTER (?publishing_date > " {(period_start)}^^xsd:dateTime" { %endif% } { %if period_end % } FILTER (?publishing_date < " {(period_end)}^^xsd:dateTime" { %endif% } }

Swagger API description : { "definitions": { "APIParams": { "properties": { "asset_type": { "type": "string" }, "author": { "type": "string" }, "period_end": { "type": "string" }, "period_start": { "type": "string" }, "wso2_session_id": { "description": "WSO2 Session id obtained with https://docs.wso2.com/display/ES200/Authenticate+Credentials+via+the+Store+REST+API", "type": "string" }, "type": "object" } }, "host": "delinium.cefriel.it", "info": { "title": "SPARQLest assets-by-author-and-date-1-0-0 API", "version": "1.0" }, "paths": { "/discovery/exec/assets-by-author-and-date-1-0-0": { "get": { "description": "", "parameters": [{ "description": "API parameters", "in": "body", "name": "parameters", "required": true, "schema": { "\$ref": "#/definitions/APIParams" } }], "responses": { "200": { "description": "SPARQL query results", "schema": { "type": "string" } } }, "summary": "Execute SPARQLest query assets-by-author-and-date-1-0-0" } }, "swagger": "2.0" }

Images

Thumbnail : ! No Thumbnail
Banner : ! No Banner

Figure 13: The SPARQL-API Service “Assets by Author and Date” in the Assets Manager

The Figure 14 shows the structured representation of the Swagger API description. It represents:

1. The path */discovery/exec/assets-by-author-and-date-1-0-0* associated to the API;
2. The required parameters that are:
 - a. the *author*;
 - b. the *asset type*;
 - c. the *wso2_session_id* (i.e., an *id* identifying the requestor);
 - d. the time period identified by *period_start* and *period_end*;
3. The output of the API invocation (i.e., a string containing the query results).

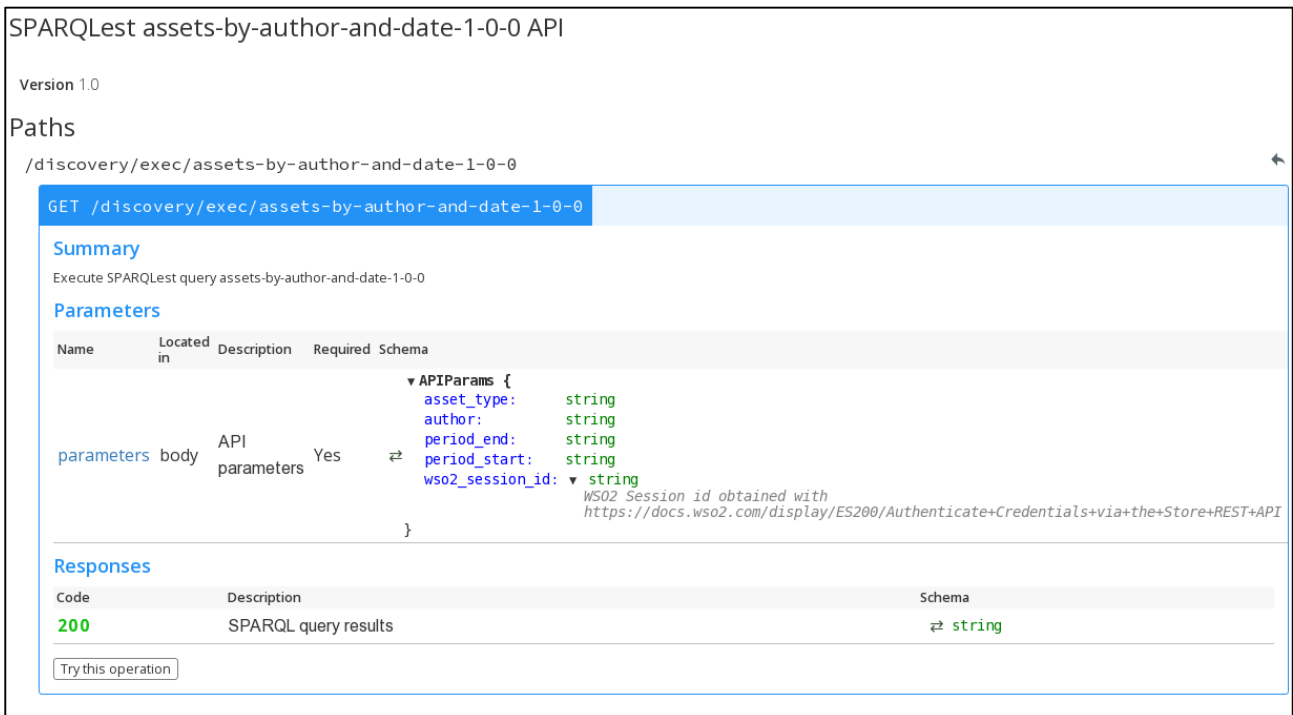


Figure 14: Structured Swagger representation of query “Assets by Author and Date”

4.5 QUERY #5: ASSETS BY AUTHOR WITH MULTIPLE CHOICE

The last proposed parametric query, named “Assets by Author with Multiple Choice”, is the following:

*“Show me all the assets provided by the author **X** or **Y**”*

The query acts on the metadata *author* and targets *all the asset types*. With respect to the query #1 (see Section 1), this query supports the specification of multiple values for the same target metadata (i.e., the author). This query simplifies the research of assets whose authors are known.

4.5.1 Template of Query #5

The Table 8 shows the SPARQL Template of the query “Assets by Author with Multiple Choice”. The query has a parameter identified by the variable *author* that can be quantified by the caller with the values assumed by the parameters *author1* and *author2* (lines 9-11 in Table 8). The query returns *asset_id* and *author* of each asset (line 1 in Table 8) that satisfies the condition of having the internal representation of the author (*dcat: contactPoint.vcard:fn*) equals to the one of the value assumed by the variable *author* (line 5-6 in Table 8).

```

PREFIX dct: <http://purl.org/dc/terms/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dcat: <https://www.w3.org/ns/dcat>
PREFIX vcard: <http://www.w3.org/2006/vcard/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

1 SELECT ?asset_id ?author
2 WHERE {
3   ?asset_id rdf:type dcat:Dataset;
4   dct:type ?asset_type;
5   dcat:contactPoint [
6     vcard:fn ?author;
7   ];
8   {%if author1 or author2 %}
9   VALUES ?author { {"\"author1\""} {"\"author2\""} }
10  {%endif%}
11 }

```

Table 8: SPARQL Template of the query “Assets by Author with Multiple Choice”


4.5.2 Swagger of Query #5

The Figure 15 shows the complete description of the SPARQL-API Service “Assets by Authors with Multiple Choice” managed by the Assets Manager that contains the metadata description, its SPARQL template and the related Swagger API description.

HOME / SPARQL-API SERVICES / ASSETS BY AUTHOR WITH MULTIPLE CHOICE / 1.0.0

SPARQL-API Services
EDIT
DELETE
LIFECYCLE
VERSION
ASSOCIATIONS
PERMISSIONS

0
OPTIONS



Assets by author with multiple choice
Version : 1.0.0
SparqlApiLifecycle : Published
Fri, 15 Sep 2017 09:07:33 GMT

Show Dependencies

Overview

Name :
Assets by author with multiple choice

Version :
1.0.0

Author :
Alessio Carenini

Author email :
alessio.carenini@cefriel.com

Institution :
Cefriel

Description :
Show me all the assets provided by the author X or Y

Expected validity :
12/31/2017

Content

SPARQL Template :

PREFIX dct: <http://purl.org/dc/terms/> PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX dcat: <https://www.w3.org/ns/dcat> PREFIX vcard: <http://www.w3.org/2006/vcard/> PREFIX foaf: <http://xmlns.com/foaf/0.1/> SELECT ?asset_id ?author WHERE { ?asset_id rdf:type dcat:Dataset; dct:type ?asset_type; dcat:contactPoint [vcard:fn ?author;] dct:issued ?publishing_date . {%if author1 or author2 %} VALUES ?author { {{\~author1~\~}} {{\~author2~\~}} } {%endif%} }

Swagger API description :

{ "definitions": { "APIParams": { "properties": { "author1": { "type": "string" }, "author2": { "type": "string" }, "wso2_session_id": { "description": "WSO2 Session id obtained with https://docs.wso2.com/display/ES200/Authenticate+Credentials+via+the+Store+REST+API", "type": "string" } }, "type": "object" }, "host": "delirium.cefriel.it", "info": { "title": "SPARQLest assets-by-author-with-multiple-choice-1-0-0 API", "version": "1.0" }, "paths": { "/discovery/exec/assets-by-author-with-multiple-choice-1-0-0": { "get": { "description": "", "parameters": [{ "description": "API parameters", "in": "body", "name": "parameters", "required": true, "schema": { "\$ref": "#/definitions/APIParams" } }], "responses": { "200": { "description": "SPARQL query results", "schema": { "type": "string" } } }, "summary": "Execute SPARQLest query assets-by-author-with-multiple-choice-1-0-0" } } }, "swagger": "2.0" }

Images

Thumbnail :
! No Thumbnail

Banner :
! No Banner

Figure 15: The SPARQL-API Service “Assets by Author with Multiple Choice” in the Assets Manager

The Figure 16 shows the structured representation of the Swagger API description. It represents (1) the path */discovery/exec/assets-by-authors-with-multiple-choice-1-0-0* associated to the API, (2) the required parameters that are the two authors (i.e., *author1* and *author2*) and the *wso2_session_id* (i.e., an *id* identifying the requestor), and (3) the output of the API invocation (i.e., a string containing the query results).

SPARQLest assets-by-author-with-multiple-choice-1-0-0 API

Version 1.0

Paths

/discovery/exec/assets-by-author-with-multiple-choice-1-0-0

GET /discovery/exec/assets-by-author-with-multiple-choice-1-0-0

Summary

Execute SPARQLest query assets-by-author-with-multiple-choice-1-0-0

Parameters

Name	Located in	Description	Required	Schema
parameters	body	API parameters	Yes	<pre> APIParams { author1: string author2: string wso2_session_id: string } </pre> <p><i>WSO2 Session id obtained with https://docs.wso2.com/display/ES200/Authenticate+Credential</i></p>

Responses

Code	Description	Schema
200	SPARQL query results	string

[Try this operation](#)

Figure 16: Structured Swagger representation of query “Assets by Author with Multiple Choice”