Cellar

The semantic repository of the Publications Office

Publications Office

Manuscript completed in June 2018.

More information on the European Union is available on the internet (http://europa.eu).

Luxembourg: Publications Office of the European Union, 2018

Print	ISBN 978-92-78-41543-3	doi:10.2830/03072	OA-01-17-789-EN-C
PDF	ISBN 978-92-78-41542-6	doi:10.2830/064688	OA-01-17-789-EN-N

Cellar

The semantic repository of the Publications Office

2018 EDITION

Contents

1. Introduction

Following the decision on reuse [1], the Publications Office (OP) put in place the Cellar, a repository to incorporate its publications. Currently, EUR-Lex [2], the EU-Bookshop [3] and others are included in the Cellar.

The Cellar is based on semantic technologies: a framework of several standards to share and reuse data. They normalise named resources in controlled vocabularies, which allows computers to talk and link to one another.

1.1. Objective and target audience

This document provides information and examples on how to access content files and metadata from the Cellar, the digital dissemination repository of the OP.

Though it is open to all citizens, this document is aimed mainly at reusers and companies who want to automatise their access to OP publications. If you are interested in regular, but not automatised access, you may find better, more relevant information in their respective portals.

1.2. Structure of the document

This document is structured in three parts.

Part I: how to access the Cellar, with real use case examples.

Part II: technical references, with all the possibilities of the application programming interface (API), ontologies and authority tables.

Part III: appendixes, with a glossary and references.

1.3. Contact us

The OP is responsible for the operational management and improvement of the Cellar. Please send us an email at op-Cellar-helpdesk@publications.europa.eu if you are aware of any of the following items, or for any other problem or enhancement you would like to communicate to us:

- an error in the data or the metadata;
- a useful use-case that might be of interest;
- a Sparql query that does not work;
- an error in the Cellar interfaces or in its content;
- an error in this document.

[1] European Commission, 'Commission decision on the reuse of Commission documents', Official Journal of the European Union (OJ L 330, 14.12.2011, pp. 39-42). Retrieved from http://eur-lex.europa.eu/legal-content/en/TXT/?uri=CELEX:32011D0833 on 03/10/2016.

[2] http://eur-lex.europa.eu/

[3] http://bookshop.europa.eu/

1.4. Update of this document

This document will be updated as often as needed or possible, always on a best-effort basis, usually when errors are spotted and corrected, or when a new Cellar version is deployed or any of its interfaces change.

1.5. Service level agreement

Access to the Cellar API is given on a best-effort basis. No level of service performance can be guaranteed at this stage.

If you are planning to try to access the Cellar interfaces repeatedly, there is a chance you might get an access error after a few hits. This is set up on the EU firewalls: there is no designed limitation on the Cellar. Please remember that resources are (always) limited, be respectful of that limited availability and do not try to download the whole Cellar (this is explained further in section 4.3).

The Sparql endpoint in particular might show some stability issues due to its nature as state-of-the-art technology, i.e. it may suffer from lack of maturity (1). We know this and we reassure you: we are continuing to work on it and hope and thank you for your patience.

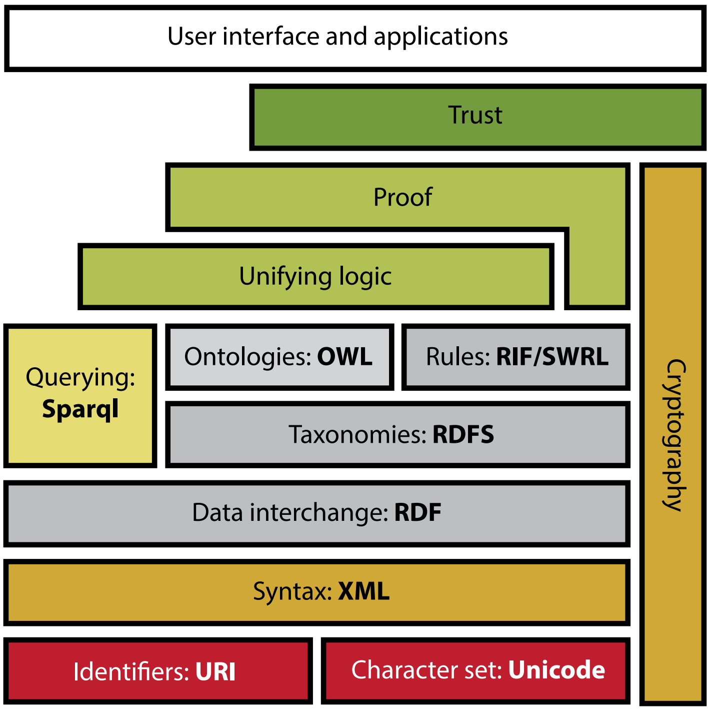
(1) A list of other Sparql endpoints is given by W3c at https://www.w3.org/wiki/SparqlEndpoints.

PART I:
Semantic access and use cases

2. Semantic web technologies

The semantic web is an evolution of the World Wide Web that allows for automating tasks on the web, letting computers talk to each other. In order to do so, data must follow common formats, structures and standards known by all participating systems. That group of standards is called the semantic web and a few are shown in Figure 1.

Figure 1: Semantic web technologies stack



Source: Wikipedia (Semantic web stack.svg, user Marobi1; *https://en.wikipedia.org/wiki/Semantic_Web_Stack#/media/File:Semantic_web_stack.svg*)

Some semantic concepts are explained in the following sections. More detailed information on the semantic web can be found on the World Wide Web Consortium (W3C) website [1].

[1] https://www.w3.org/2013/data/

2.1. Naming things with URIs

The first thing needed to understand the semantic web is the standard in naming things with common vocabularies and names (identifiers). In order to get universal identifiers that were common to all world computers, a very simple approach was taken: using the existing web. These identifiers are called uniform resource identifiers (URIs) and look very much like web addresses or uniform resource locators (URLs), though they are not the same. A URI can have a web page or URL, but its main use is as a universal identifier. For example,

- **`<http://publications.europa.eu/resource/oj/JOA_1952_001_R>`** is both a URI and a valid URL, because when typed in a web browser, it returns some content or metadata associated with it. Here, we say that the URI is 'dereferenceable'.
- **`<http://publications.europa.eu/ontology/cdm#act_body_agreement_international>`** is a URI (it identifies a type of document in the ontology) for the moment; it could also become a URL if it is mapped to a web page.

Other websites in or outside the Cellar can use the same identifiers and even refer to them. In the rest of the document, the term URI will be used, and it may or may not point to a website.

The important part here is that URIs ***do not change***.

'What makes a cool URI?
A cool URI is one which does not change.
What sorts of URI change?
URIs don't change: people change them.'

Tim Berners-Lee, 1998
https://www.w3.org/Provider/Style/URI

URIs in the Cellar will not change.

The OP will ensure this with data modelling and a number of technologies so you don't need to worry, or to search every time, or to download anything to compare or map to previous identifiers.

More information on best practices on naming things in the semantic web can be found in the excellent free online book *Linked Data Patterns* by Leigh Dodds and Ian Davis [1].

[1] http://patterns.dataincubator.org/book/

2.2. Dereferencing URIs

Machines and humans do not see information in the same way. When accessing a URI, a computer receives a different version of what people see in the browser.

When the Cellar is asked for a URI, it returns information depending on who is asking.

- When it is a computer (in general), the Cellar responds with a resource description framework (RDF — see next section) with the description of the URI, with all available metadata at that level.
- When it is a human, the Cellar responds by default with the content related to the URI — either PDF, or HTML, or any other format available. (This depends on the browser, especially for formats which need plugins, like PDF).

This is done via a mechanism called content negotiation. You already know and use it: when you browse on the internet, a request for a webpage usually returns it correctly (code ok: 200), or with an error (code 404). This is content negotiation. The Cellar can respond with these or other codes, such as a redirection (code 303), to one of the two modes above, either the RDF or the content.

Of course, both humans and machines can simulate both ways of obtaining content or metadata. In order to do so, one would need to specify the HTTP headers, with tools like the following.

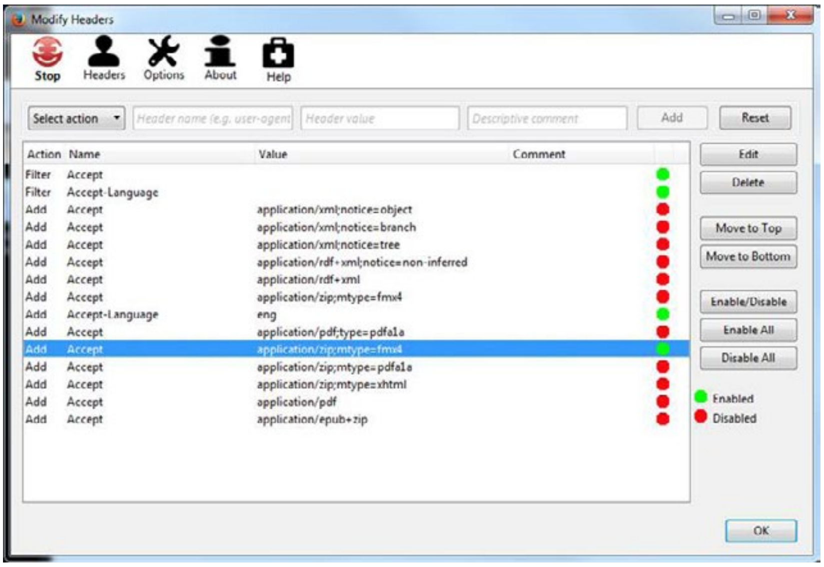
- Client URL Request Library (cURL) (1): with the –H option for the headers, and the –L to follow the Cellar redirects. Example:

```
curl -L -H “Accept:application/xml;notice=tree” -H
“Accept-Language:eng” http://publications.europa.eu/
resource/oj/JOA_1952_001_R
```

- Modify headers (2) plugin for the Firefox (3) browser: see Figure 2 for a screen capture with a configuration example. A similar plugin (4) exists for the browser Google Chrome (5).

(1) https://curl.haxx.se/
(2) http://addons.mozilla.org/firefox/addon/modify-headers/
(3) http://www.mozilla.org/firefox/
(4) https://chrome.google.com/webstore/detail/modify-headers-for-google/innpjfdalfhpcoinfnehdnbkglpmogdi
(5) http://www.google.com/chrome/

Figure 2: Screen capture with a configuration example of the modify headers plugin for Mozilla Firefox.



Action	Name	Value	Comment	
Filter	Accept			🟢
Filter	Accept-Language			🟢
Add	Accept	application/xml;notice=object		🔴
Add	Accept	application/xml;notice=branch		🔴
Add	Accept	application/xml;notice=tree		🔴
Add	Accept	application/rdf+xml;notice=non-inferred		🔴
Add	Accept	application/rdf+xml		🔴
Add	Accept	application/zip;mtype=fmx4		🔴
Add	Accept-Language	eng		🟢
Add	Accept	application/pdf;type=pdfa1a		🔴
Add	Accept	application/zip;mtype=fmx4		🟢
Add	Accept	application/zip;mtype=pdfa1a		🔴
Add	Accept	application/zip;mtype=xhtml		🔴
Add	Accept	application/pdf		🔴
Add	Accept	application/epub+zip		🔴

🟢 Enabled
🔴 Disabled

The full specifications for the headers accepted by the Cellar can be found in chapter *5*.

2.3. Semantic modelling

Once entities have been named (2.1) and there is a negotiation mechanism to access them (2.2), we can introduce the technologies that can be used to model the relationships. Some definitions follow [1].

Triple: a minimal unit of information composed of three parts: subject, predicate or property, and object or value. For example, see Table 1. Triples can also be expressed in RDF format.

Triplestore: a database of triples, also known as an RDF triplestore.

Subject: any entity upon which we can define properties. The subject is always a URI.

[1] For more formals definitions you may refer to http://www.w3.org/TR/ld-glossary/

Table 1: Examples of triples

Subject	Property	Object
`oj:JOA_1952_001_R ( )`	`cdm:work_date_document`	`1952-12-30`
`oj:JOA_1952_001_R`	`rdf:type`	`cdm:official-journal`
`celex:31952S0003`	`cdm:resource_legal_published_in_official-journal`	`oj:JOA_1952_001_R`
`celex:31952S0003`	`owl:sameAs`	`eli:dec/1952/3/oj`

Property: or predicate, it is a verb explaining how the object is related to the subject. The property is always a URI. Please note that properties can also be subjects in a triple.

Object: can be either an entity (expressed with a URI) or a literal value with its data type. Please note that objects with URIs can also be subjects in a triple.

Ontology: a formal model that allows knowledge to be represented for a specific domain. An ontology describes the types of things that exist (classes), the relationships between them (properties), and the logical ways those classes and properties can be used together (axioms).

RDF: resource description framework. A family of international standards for data interchange on the web. RDF is based on the idea of identifying things using web identifiers or HTTP URIs and describing resources in terms of simple properties and property values. It is the model in which the triples (subject, property, and object) are coded. Some vocabularies are predefined, like RDFS, OWL or SKOS, all used at the OP. Externally, this looks like the extended markup language (XML). An example of an RDF file is shown in Table 2, with similar contents as the triples in Table 1.

[1] This is an abbreviated form of `<http://publications.europa.eu/resource/oj/JOA_1952_001_R>` as if a prefix was declared. Please note that works like this OJ have several identifiers. The triplestore is normalised to have all properties to the same identifier type, so this result may only be achieved via `owl:sameAs` property.

Table 2: Example of an RDF file with some of the triples from Table 1

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF …>
<rdf:RDF …>
<onto:official-journal rdf:about= "&resource;oj/JOA_1952_001_R">
<onto:work_id_document rdf:datatype="&xsd;string">oj:JOA_1952_0001_R</
onto:work_id_document>
<onto:work_date_document rdf:datatype="&xsd;date">1952-12-30</
onto:work_date_document>
<onto:work_created_by_agent>
<onto:institution rdf:about="&Cellar-authority;corporate-body/EC">
<skos:inScheme … rdf:resource= "&Cellar-authority;corporate-body/"/>
</onto:institution>
</onto:work_created_by_agent>
…
```

RDFS: RDF schema. It provides description of vocabularies to structure RDF resources.

OWL: ontology web language. It uses classes, entities and properties to define more complex structures and restrictions, like cardinality, value restrictions or transitive property. The ontology used at the OP is called common data model (CDM) and is further explained in section 6.1.

SKOS: simple knowledge organisation system. It provides a framework for defining thesauri, taxonomies and other vocabularies.

Sparql: Sparql protocol and RDF query language. It is the language used for querying the RDF triple store via an endpoint. Some examples can be found in Section 4.

Sparql endpoint: access service to a triplestore, receives Sparql queries and responds with results in a variety of formats, including XML, JSON, HTML, etc.

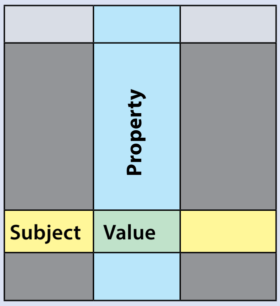
Further information about semantic technologies can be found in many books and online resources (1).

(1) Popular resources include Heath, T and Bizer, C, 'Linked Data: Evolving the Web into a Global Data Space (1st edition)', *Synthesis Lectures on the Semantic Web: Theory and Technology*, Vol. 1, No 1, Morgan & Claypool, 2011, pp. 1-136, (http://linkeddatabook.com/). Other books include DuCharme, B, *Learning Sparql*, O'Reilly, Sebastopol, 2013. In addition, there are numerous online resources.

2.4. From SQL to Sparql

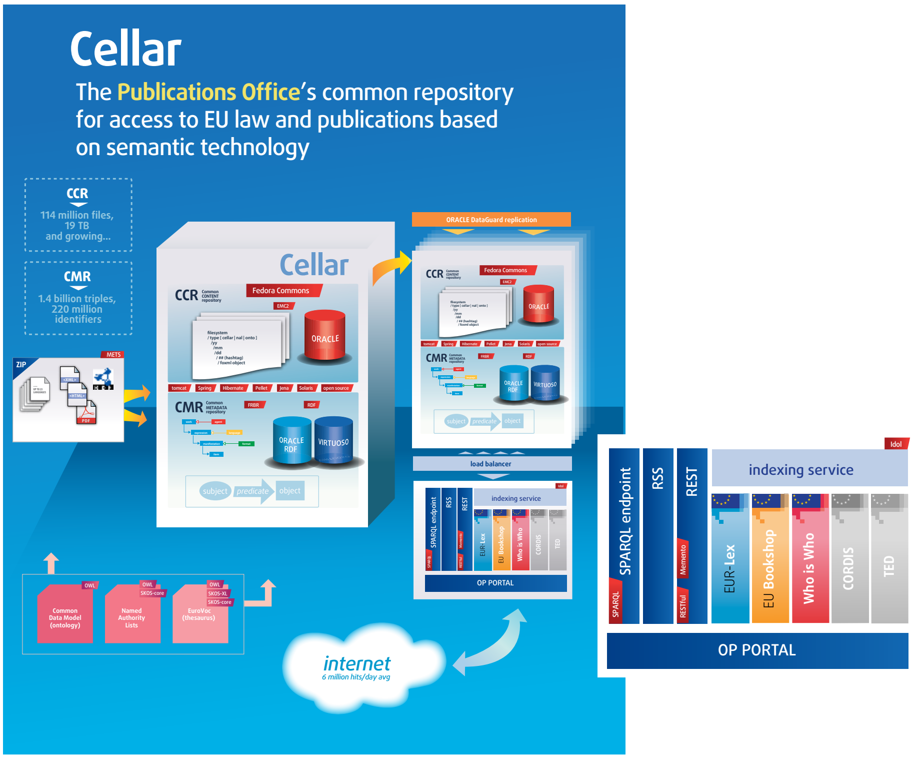
In case you already have some knowledge on relational databases, semantic databases are similar. In Table 3 you can find some differences between them.

Table 3: Differences between relational and semantic databases

	Relational database	Semantic database
Storage	Several tables with fixed columns, primary keys, foreign keys, etc.	Standard scheme: triple: { subject, property, object/value } 
Queries	SQL	Sparql
Performance	Very high, mature technology	Variable, new technology
Model	Rigid: constraints, keys, triggers, external application	Flexible: graphs, ontology
Compliance	Often vendor-specific, generally not directly portable	Core Sparql largely portable
Federation	Very difficult or impossible. Syntactic differences	Integrated
Identifiers	Own	Normalised with URIs
Common use tables	Must be imported and maintained	Can be linked as authorities
Reasoning	External tools	Integrated in the ontology (e.g. inference)
Discovery	External tools	Integrated Sparql endpoint
Standard	ISO	W3C recommendation

3. Access possibilities

Figure 3: Cellar architecture with detail in interfaces.


Cellar
The Publications Office's common repository for access to EU law and publications based on semantic technology
CCR
114 million files, 19 TB and growing...
CMR
1.4 billion triples, 220 million identifiers
ZIP
METS
Cellar
CCR Common CONTENT repository
Fedora Commons
ORACLE
CMR Common METADATA repository
ORACLE RDF
VIRTUOSO
subject predicate object
ORACLE DataGuard replication
load balancer
Common Data Model (ontology)
Named Authority Lists
EuroVoc (thesaurus)
OWL
SKOS-core
SKOS-XL
internet
6 million hits/day avg
SPARQL endpoint
RSS
REST
Memento
RESTful
SPARQL
indexing service
Idol
EUR-Lex
EU Bookshop
Who is Who
CORDIS
TED
OP PORTAL

There are several levels on which access is possible; see Table 4 for a high-level classification:

Table 4: Access possibilities to Cellar notifications, content and metadata

	RSS notifications	Metadata	Metadata and content
EUR-Lex (meant for humans)	Custom	Web services	EUR-Lex website
Cellar (meant for machines)	Full	Sparql endpoint	RESTful interface

3.1. EUR-Lex RSS

EUR-Lex and the OP Portal are front-ends for the Cellar repository. The EUR-Lex RSS can be useful when one needs a summary of what is being published. It has several categories to choose from.

For the specifications, please refer to the EUR-Lex RSS webpage (http://eur-lex.europa.eu/predefined-rss.html).

An example of the content of one of these RSSs is shown in Table 5.

Table 5: EUR-Lex OJ L Complete Edition RSS extract

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" media="screen" href="./../screen-rss.xsl"?>
<rss version="2.0">
<channel
  xmlns:dc="http://purl.org/dc/elements/1.1/">
  <title>OJ L Complete Edition</title>
  <pubDate>Thu, 08 Jun 2017 15:15:43 +0200</pubDate>
  <description>Retrieves the complete edition of the L series (Legislation) of the
Official Journal of the European Union, and not the acts.</description>
  <language>en</language>
  <link>http://eur-lex.europa.eu/homepage.html</link>
  <image>
     <title>OJ L Complete Edition</title>
     <url>http://eur-lex.europa.eu/images/eurlex_logo.png</url>
     <link>http://eur-lex.europa.eu/homepage.html</link>
  </image>
  <item>
     <title>OJ:L:2017:145:FULL: Official Journal of the European Union, L 145, 8 June
     2017</title>
     <link>http://eur-lex.europa.eu/./legal-content/AUTO/?uri=OJ:L:2017:145:FULL</link>
     <guid>http://eur-lex.europa.eu/./legal-content/AUTO/?uri=OJ:L:2017:145:FULL</guid>
     <category>Lex Alerts</category>
     <dc:creator>European Union</dc:creator>
  </item>
  <item>
     <title>OJ:L:2017:144:FULL: Official Journal of the European Union, L 144, 7 June
     2017</title>
     <link>http://eur-lex.europa.eu/./legal-content/AUTO/?uri=OJ:L:2017:144:FULL</link>
     <guid>http://eur-lex.europa.eu/./legal-content/AUTO/?uri=OJ:L:2017:144:FULL</guid>
     <category>Lex Alerts</category>
     <dc:creator>European Union</dc:creator>
  </item>
  <item>(…)
```

The concrete use of RSS is outside the scope of this document. Information and examples on how to use RSS can be found in Internet [1].

3.2. Cellar RSS

The Cellar has its own RSS, which has the same content as EUR-Lex plus other publications (from EU-Bookshop, for example), detailed to the most granular extent possible. Every modification in the metadata for every language, every update of an authority table or ontology, everything will be shown in it.

(1) For example: http://www.whatisrss.com/

The Cellar RSS cannot be filtered manually as it is meant to be used by machines. For the full specifications, please refer to the RSS API in section 5.

3.3. EUR-Lex web services

If the RSS is not enough or you have more specific needs (like the documents of a specific sector), we recommend you get web-services notifications.

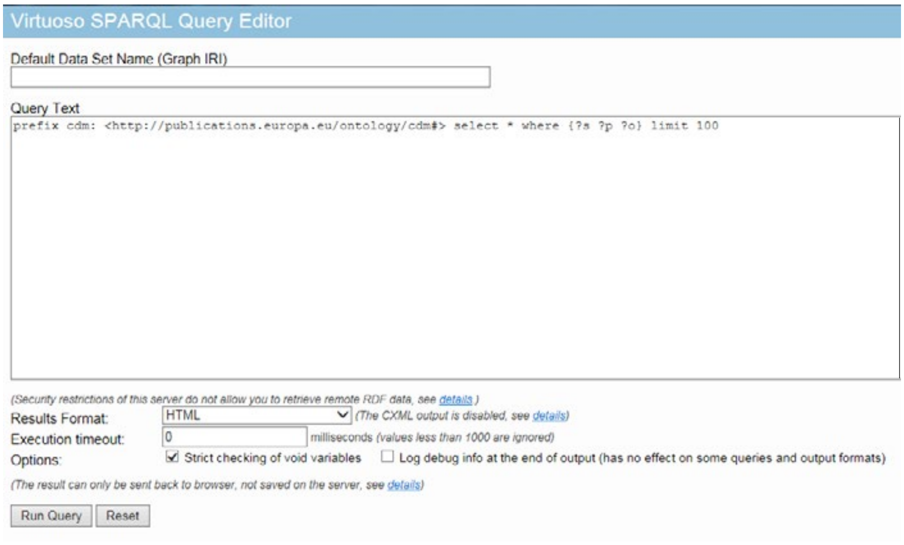
More information is available on the web services help page (http://eur-lex.europa.eu/content/help/faq/intro.html#help15) and in the manual (http://eur-lex.europa.eu/content/tools/webservices/SearchWebServiceUserManual_v2.00.pdf).

Please note that the names of fields in EUR-Lex web services may be different from those present in the ontology and the Sparql endpoint. Therefore, if you implement a service based on EUR-Lex web-services and later want to move to the Cellar, you will need to reimplement or remap field names.

3.4. Cellar Sparql endpoint

All metadata in the repository can be consulted via the Sparql endpoint. A complete syntax and semantics specification of Sparql can be found at the W3C website (¹). Queries can be done in the OP's Sparql endpoint (http://publications.europa.eu/webapi/rdf/sparql) as shown in Figure 4, though a step-by-step wizard exists as well (http://publications.europa.eu/en/linked-data) as shown in Figure 5.

Figure 4: Cellar Sparql endpoint


Virtuoso SPARQL Query Editor
Default Data Set Name (Graph IRI)
Query Text

```
prefix cdm: <http://publications.europa.eu/ontology/cdm#> select * where {?s ?p ?o} limit 100
```

(Security restrictions of this server do not allow you to retrieve remote RDF data, see details)

Results Format: HTML (The CXML output is disabled, see details)

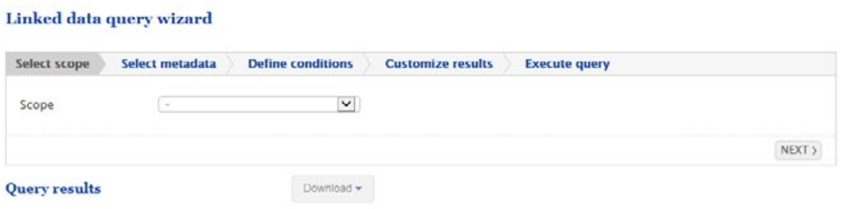
Execution timeout: 0 milliseconds (values less than 1000 are ignored)

Options: [✓] Strict checking of void variables [ ] Log debug info at the end of output (has no effect on some queries and output formats)

(The result can only be sent back to browser, not saved on the server, see details)

(¹) https://www.w3.org/TR/rdf-sparql-query/

Figure 5: OP Portal linked data query wizard



Several examples can be found in the use cases (see section 4).

3.5. Cellar RESTful interface

All content and metadata can be accessed directly via a RESTful interface, which dereferences the requests upon the headers, as shown in section 2.2.

For the full specifications, please refer to the RESTful API in section 5.

4. Use cases

4.1. Presentation at Diplohack, Brussels, March 2016

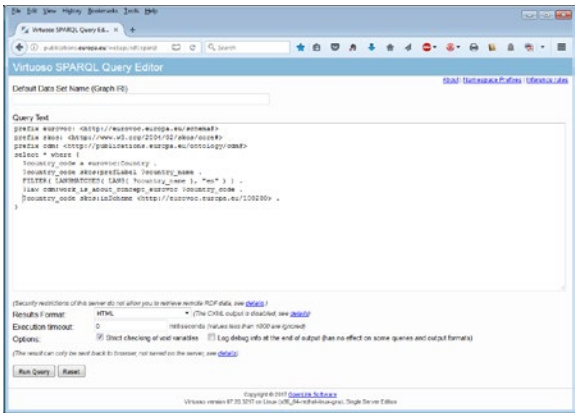
Scenario: the objective of this hackathon(1) was to present European citizens with new ways of using EU data to enhance transparency. The following use case was presented there as a trigger for further ideas. Imagine someone might want to list the African countries by the number of EU laws related to them.

Solution: this can be done with the query in Table 6. Copy all lines from the central column of that table to the input box of the Sparql endpoint from section 3.4 as shown in Figure 4. Click the 'Run Query' button. The results can be seen in Table 7.

Table 6: Commented Sparql query to get the list of countries in English, with the EU laws related to them.

01	`prefix eurovoc: <http://eurovoc.europa.eu/schema#>`	Prefixes are used to save space in URIs (see 2.1 and footnote 14) when writing queries
02	`prefix skos: <http://www.w3.org/2004/02/skos/core#>`	
03	`prefix cdm: <http://publications.europa.eu/ontology/cdm#>`	
04	`select * where {`	We select everything that fulfils the following criteria (all must finish with a point)
05	`?country_code a eurovoc:Country.`	Is type country
06	`?country_code skos:prefLabel?country_name.`	Link of country code and name
07	`FILTER(LANGMATCHES(LANG(?country_name), "en")).`	Country name in English
08	`?law cdm:work_is_about_concept_eurovoc?country_code.`	Country name related to a Eurovoc concept
09	`?country_code skos:inScheme <http://eurovoc.europa.eu/100280> .`	Eurovoc concept is Africa
10	`}`	End of the selection criteria

Figure 6: Sparql endpoint with the query from Table 6 copied and ready to be run.


File Edit View History Bookmarks Tools Help
Virtuoso SPARQL Query Ed...
Virtuoso SPARQL Query Editor
About | Namespace Prefixes | Inference rules
Run Query
Reset

Default Data Set Name (Graph IRI)

Query Text

```
prefix eurovoc: <http://eurovoc.europa.eu/schema#>
prefix skos: <http://www.w3.org/2004/02/skos/core#>
prefix cdm: <http://publications.europa.eu/ontology/cdm#>
select * where {
  ?country_code a eurovoc:Country .
  ?country_code skos:prefLabel ?country_name .
  FILTER( LANGMATCHES( LANG( ?country_name ), "en" ) ) .
  ?law cdm:work_is_about_concept_eurovoc ?country_code .
  ?country_code skos:inScheme <http://eurovoc.europa.eu/100280> .
}
```

(Security restrictions of this server do not allow you to retrieve remote RDF data, see details.)

Results Format: HTML (The CXML output is disabled, see details)

Execution timeout: 0 milliseconds (values less than 1000 are ignored)

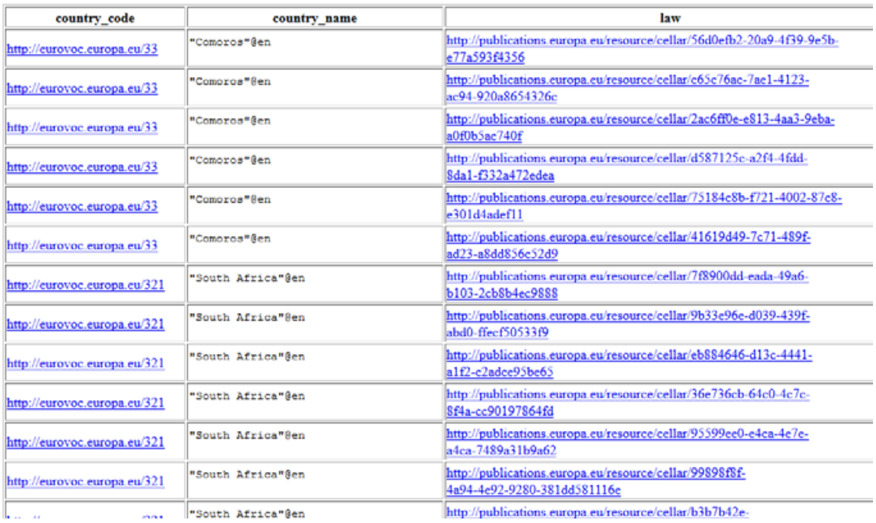
Options: [✓] Strict checking of void variables [ ] Log debug info at the end of output (has no effect on some queries and output formats)

(The result can only be sent back to browser, not saved on the server, see details)

Copyright © 2017 OpenLink Software
Virtuoso version 07.20.3217 on Linux (x86_64-redhat-linux-gnu), Single Server Edition

(1) http://diplohack.brussels/

Table 7: Some results of the Sparql query in Table 6.



country_code	country_name	law
http://eurovoc.europa.eu/33	"Comoros"@en	http://publications.europa.eu/resource/cellar/56d0efb2-20a9-4f39-9e5b-e77a593f4356
http://eurovoc.europa.eu/33	"Comoros"@en	http://publications.europa.eu/resource/cellar/c65c76ac-7ae1-4123-ac94-920a8654326c
http://eurovoc.europa.eu/33	"Comoros"@en	http://publications.europa.eu/resource/cellar/2ac6ff0e-e813-4aa3-9eba-a0f0b5ac740f
http://eurovoc.europa.eu/33	"Comoros"@en	http://publications.europa.eu/resource/cellar/d587125c-a2f4-4fdd-8da1-f332a472edea
http://eurovoc.europa.eu/33	"Comoros"@en	http://publications.europa.eu/resource/cellar/75184c8b-f721-4002-87c8-e301d4adef11
http://eurovoc.europa.eu/33	"Comoros"@en	http://publications.europa.eu/resource/cellar/41619d49-7c71-489f-ad23-a8dd856c52d9
http://eurovoc.europa.eu/321	"South Africa"@en	http://publications.europa.eu/resource/cellar/7f8900dd-eada-49a6-b103-2cb8b4ec9888
http://eurovoc.europa.eu/321	"South Africa"@en	http://publications.europa.eu/resource/cellar/9b33e96e-d039-439f-abd0-ffecf50533f9
http://eurovoc.europa.eu/321	"South Africa"@en	http://publications.europa.eu/resource/cellar/eb884646-d13c-4441-a1f2-c2adce95be65
http://eurovoc.europa.eu/321	"South Africa"@en	http://publications.europa.eu/resource/cellar/36e736cb-64c0-4c7c-8f4a-cc90197864fd
http://eurovoc.europa.eu/321	"South Africa"@en	http://publications.europa.eu/resource/cellar/95599ee0-e4ca-4e7e-a4ca-7489a31b9a62
http://eurovoc.europa.eu/321	"South Africa"@en	http://publications.europa.eu/resource/cellar/99898f8f-4a94-4e92-9280-381dd581116e
[illegible]	"South Africa"@en	http://publications.europa.eu/resource/cellar/b3b7b42e-

A more advanced query could get just the African country names in English, plus the total number of EU laws related to each. The query is shown in Table 8, and after executing it in the Sparql endpoint, the result is shown in Table 9. Please note that the order may vary; and also new laws are published every day, so the numbers may increase over time.

Table 8: Commented Sparql query to get the list of countries in English, with the EU laws related to them

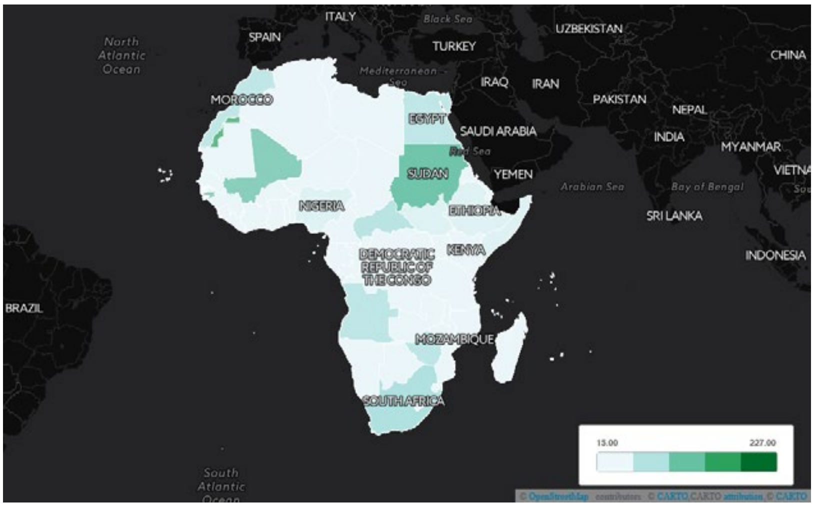
```
prefix eurovoc: <http://eurovoc.europa.eu/schema#>
prefix skos: <http://www.w3.org/2004/02/skos/core#>
prefix cdm: <http://publications.europa.eu/ontology/cdm#>
select count(?law)?country_name
where {
?country_code skos:prefLabel?country_name.
FILTER(LANGMATCHES(LANG(?country_name), "en")).
?law cdm:work_is_about_concept_eurovoc?country_code.
?country_code skos:inScheme <http://eurovoc.europa.eu/100280> .
}
group by?country_name
```

Table 9: Results of the Sparql query in Table 8 showing a list of African countries with the number of EU laws related to them.

country_name	# of laws
“Equatorial Guinea”@en	107
“Burkina Faso”@en	39
“Nigeria”@en	324
“Togo”@en	45
“Ghana”@en	52
“São Tomé and Príncipe”@en	103
“South Sudan”@en	68
“Botswana”@en	154
“Comoros”@en	121
“Guinea-Bissau”@en	243
“Mozambique”@en	242
“Namibia”@en	316
“Niger”@en	84
“Western Sahara”@en	335
“Gambia”@en	39
“Zambia”@en	106
“Egypt”@en	1066
“Ethiopia”@en	402
…	…

Finally, with a copy and paste to an external map service (outside the scope of this document), an image like the one shown in Figure 7 can be obtained.

Figure 7: Map with the results from Table 9



Source: Carto.com with OpenStreetMap cartography

4.2. Getting the table of contents, acts and signature related to the authentic OJ

Scenario: you want to get all the work URIs (table of contents, acts and signature) related to a specific, authentic Official Journal.

Solution: this can be done in Sparql with the query shown in Table 10. There are as many result rows as acts, shown in Table 11.

Table 10: Sparql query for getting the URIs of TOC, acts and signature of an Official Journal

```
prefix cdm: <http://publications.europa.eu/ontology/cdm#>
select distinct <http://publications.europa.eu/resource/oj/JOL_2017_001_R>
?act?toc?sig
where {
?c_oj owl:sameAs <http://publications.europa.eu/resource/oj/JOL_2017_001_R> .

?c_act cdm:resource_legal_published_in_official-journal?c_oj.
?c_act owl:sameAs?act.
FILTER (regex(?act,'/oj/')).

?c_toc cdm:work_part_of_work?c_oj.
?c_toc owl:sameAs?toc.

?c_exp cdm:expression_belongs_to_work?c_oj.
?c_manif cdm:manifestation_manifests_expression?c_exp.
?c_sig cdm:signature_digital_signs_manifestation?c_manif.
?c_sig owl:sameAs?sig.
}
```

Table 11: Results of Sparql query of Table 10, with the URIs of TOC, acts and signature of the Official Journal `<http://publications.europa.eu/resource/oj/JOL_2017_001_R>`

OJ	Acts	TOC	Signature
`http://publications.europa.eu/resource/oj/JOL_2017_001_R`	`http://publications.europa.eu/resource/oj/JOL_2017_001_R_0002`	`http://publications.europa.eu/resource/oj/JOL_2017_001_R_TOC`	`http://publications.europa.eu/resource/oj/JOL_2017_001_R_SIG`
`http://publications.europa.eu/resource/oj/JOL_2017_001_R`	`http://publications.europa.eu/resource/oj/JOL_2017_001_R_0001`	`http://publications.europa.eu/resource/oj/JOL_2017_001_R_TOC`	`http://publications.europa.eu/resource/oj/JOL_2017_001_R_SIG`

4.3. Download a part or the whole of the repository

Scenario: you want to download all or part of the Cellar.

Solution: given the possibilities of Cellar interfaces, it is possible to download everything from it, but be warned, it is big. As of July 2017, there were:

- 1.6 billion triples (that is, 1.6 x 10^9),
- 246 million identifiers,
- 127 million files,
- Occupying 27.5 TB in disk and 4.1 TB in the database,
- And getting an average of 9 million hits per day, that is, more than 100 hits per second.

The point of having semantic technologies is precisely not needing to download anything and trusting the author, who will keep everything online and updated. You can model your data and link to the Cellar (see section 2).

There is no single list or catalogue on the publications, except by collections, which can be consulted in the corresponding portals, like EUR-Lex or the EU-Bookshop. However, you can make general queries in the Sparql endpoint, but more general queries take much more time to process.

From either source, and with some links ready, you may download documents one by one, or automate the process. In any case, please take into account Section 1.5: service level agreement.

Often, we receive questions on filtering or customising some extractions. It is impossible for the OP to fulfil all kinds of requests. Nonetheless we will be happy to extend this document to answer the most common needs and provide examples.

4.4. Old license holder extractions at the OP

Scenario: in the past, the OP had contracts with license holders who paid for specific extraction notices, made available in an FTP server. As the reuse Decision came into force (see footnote 1), the contracts were terminated. These extractions were phased out for the following reasons.

- The extractions were not complete, or at least not as complete as the Sparql or the RESTful API. The reason: there are updates after the extractions to the FTP.
- The extractions were a non-standard interface.
- The insecurity inherent to an FTP server.

Solution: do the following steps:

1. Get RSS notifications (see section 5.2.3) with wemiClasses=work, other parameters as you need. This will retrieve references like:

Table 12: Example of an RSS extraction of a work

```
<item>
<guid isPermaLink="false">cellar:b768d303-9301-11e7-b92d-01aa75ed71a1</guid>
<notifEntry:type>CREATE</notifEntry:type>
<notifEntry:classes>
<notifEntry:class>http://publications.europa.eu/ontology/cdm#evolutive_
work</notifEntry:class>
<notifEntry:class>http://publications.europa.eu/ontology/cdm#serial_work</
notifEntry:class>
<notifEntry:class>http://publications.europa.eu/ontology/cdm#resource_legal</
notifEntry:class>
<notifEntry:class>http://publications.europa.eu/ontology/cdm#work</noti
fEntry:class>
</notifEntry:classes>
<notifEntry:identifiers>
<notifEntry:identifier>celex:62016CC0244</notifEntry:identifier>
<notifEntry:identifier>ecli:ECLI%3AEU%3AC%3A2017%3A635</notifEntry:iden
tifier>
</notifEntry:identifiers>
<notifEntry:date>2017-09-06T14:48:48+02:00</notifEntry:date>
</item>
```

2. For every item above, take the <guid> tag (in the example, cellar:b768d303-9301-11e7-b92d-01aa75ed71a1) and get the XML branch notice of the publication in the language required (see section 5.2.1.2).

The result will be as follows:

Table 13: Example of a branch notice extraction

```
<?xml version="1.0" encoding="UTF-8"?>
<NOTICE decoding="eng" type="branch">
<WORK>
<URI>
<VALUE>http://publications.europa.eu/resource/cellar/b84f49cd-750f-11e3-
8e20-01aa75ed71a1</VALUE>
<IDENTIFIER>b84f49cd-750f-11e3-8e20-01aa75ed71a1</IDENTIFIER>
<TYPE>cellar</TYPE>
</URI>
<SAMEAS>
…
```

4.5. Case-law

Scenario: how can you get the case-law collection? Filter: title (when available) in English, European case-law identifier (ECLI), divided into four classes:

- Documents CJ EU: contains the whole sector (1) 6+8 (documents from the Court, from the National Courts, notices published in the OJ, evolutive works).
- Summary Case Law: short version of the documents from the Court.
- Summary Case Law Jure: short version of the documents from the National Courts.
- Case reports.

Solution: use the Sparql query in Table 14. Also, if you are only interested in one of the classes, you may remove the others from the FILTER query line.

Table 14: Sparql query to get case-law in English, with class and ECLI

```
prefix cdm: <http://publications.europa.eu/ontology/cdm#>
select distinct?w?class?ecli?title
where {
?w a?class.
OPTIONAL{?w cdm:case-law_ecli?ecli.}
OPTIONAL{?e cdm:expression_belongs_to_work?w.
?e cdm:expression_uses_language <http://publications.europa.eu/resource/authority/
language/ENG>.
?e cdm:expression_title?title.}
FILTER(?class in (<http://publications.europa.eu/ontology/cdm#document_cjeu>,
<http://publications.europa.eu/ontology/cdm#court-report>,
<http://publications.europa.eu/ontology/cdm#summary_case-law>,
<http://publications.europa.eu/ontology/cdm#summary_case-law_jure>))
}
order by?class
```

The startup OpenLaws is enriching (2) justice documents targeting citizens, businesses and legal experts with a new platform called <http://openlaws.eu>. It uses Sparql to get the notice references, and then the RESTful interface to get the metadata in RDF and the contents.

(1) For more information on sectors, please refer to the FAQ from EUR-Lex: http://eur-lex.europa.eu/content/help/faq/intro.html#help9

(2) https://zenodo.org/record/158999/files/D4.4.d3%20BOLD%20Vision.pdf

4.6. EU treaties

Scenario: how can you get all EU treaties?

Solution: use the Sparql query in Table 15.

Table 15: Sparql query to get all EU treaties

```
prefix cdm: <http://publications.europa.eu/ontology/cdm#>
select distinct?work?doc_id
where
{
?work a cdm:treaty;
  cdm:work_id_document?doc_id.
FILTER not exists{?work a cdm:fragment_resource_legal}.
}
```

4.7. EU legislation in force about climate change

Scenario: what is the EU legislation in force about the Eurovoc concept 'climate change'?

Solution: use the Sparql query in Table 16.

Table 16: Sparql query to get all EU legislation in force about climate change

```
prefix cdm: <http://publications.europa.eu/ontology/cdm#>
select distinct?work
where
{
?work cdm:resource_legal_in-force "true"^^<http://www.w3.org/2001/
XMLSchema#boolean> ;
 a cdm:legislation_secondary;
 cdm:work_is_about_concept_eurovoc <http://eurovoc.europa.eu/5482> .
}
```

Other examples are the projects Jurion [1] and Aligned [2], which are enriching the EU legislation. They use Sparql to get the notice references, and then the RESTful interface to get the metadata in RDF and the contents.

The project api.epdb.eu [3] is also reusing EU legislation and pre-legislation in the same way, but for providing a new API to access it. Check its website for examples on code used and visualisation.

[1] http://jurion.de/
[2] http://aligned-project.eu/
[3] http://api.epdb.eu/

PART II: Technical reference

5. RESTful and RSS API

5.1. Main concepts

Here follows a description of the main concepts on which the CELLAR data model is built upon:

- Functional Requirements for Bibliographic Records (FRBR) – paragraph 5.1.1
- Types of notices – paragraph 5.1.2
- Content streams – paragraph 5.1.3
- NALs – paragraph 5.1.4
- EUROVOC – paragraph 5.1.5
- Resource URI – paragraph 5.1.6.

5.1.1. FUNCTIONAL REQUIREMENTS FOR BIBLIOGRAPHIC RECORDS (FRBR)

Functional Requirements for Bibliographic Records (FRBR) is a conceptual entity-relationship model developed by the International Federation of Library Associations and Institutions (IFLA) that relates user tasks of retrieval and access in online library catalogues and bibliographic databases from a user's perspective.

The FRBR comprises 3 groups of entities.

The group 1 entities are the Work, Expression, Manifestation, and Item (WEMI): they represent the products of intellectual or artistic endeavour, and are the foundation of the FRBR model.

Here follows a description of each:

- the Work is generally defined as a distinct intellectual or artistic creation. Example: Beethoven's Ninth Symphony apart from all ways of expressing it is a work
- the Expression is the specific intellectual or artistic form that a work takes each time it is 'realized'. Example: an expression of Beethoven's Ninth might be the musical score he wrote down
- the Manifestation is the physical embodiment of an expression of a work. As an entity, manifestation represents all the physical objects that bear the same characteristics, in respect to both intellectual content and physical form. Example: the recording the London Philharmonic made of the Ninth in 1996 is a manifestation
- the Item is a single exemplar of a manifestation. The entity defined as item is a concrete entity. Example: each of the 1996 pressings of that 1996 recording is an item.

The group 2 entities are Person and Corporate body, responsible for the custodianship of Group 1's intellectual or artistic endeavor.

The group 3 entities are subjects of Group 1 or Group 2's intellectual endeavour, and include Concepts, Objects, Events and Places.

5.1.1.1. FRBR in Cellar's context

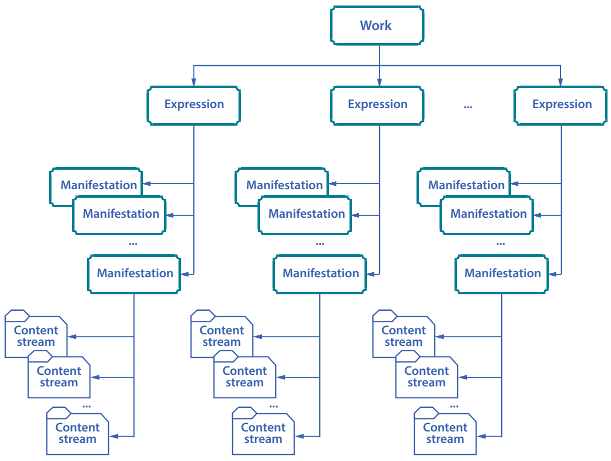
For what concerns its use in the Cellar, the essential idea of FRBR is to present a publication at different levels of abstraction. In order to accomplish this, the Cellar realizes the WEMI pattern through three different hierarchies, each with its own levels of abstraction.

A. Hierarchy work-expression-manifestation-content stream

The work-expression-manifestation-content stream hierarchy (see Figure 7) is composed by:

- a work, which covers the W role of the WEMI pattern. A work may embed:
- several expressions. An expression covers the E role of the WEMI pattern, and is defined as the realization of a work in a specific language. It may embed:
- several manifestations. A manifestation covers the M role of the WEMI pattern, and is defined as the instantiation of a work in the language defined by the embedding expression, and in a specific format. Finally, a manifestation may embed:
- several content streams. A content stream covers the I role of the WEMI pattern, and is defined as the entity that physically carries the information of the manifestation. The content stream is typically a document written in the language and format defined by the embedding manifestation.

Figure 8: The work-expression-manifestation-content stream hierarchy



The Cellar contains works from the OP's primary domains of work:

- Legislative data, currently published primarily in the EUR-Lex portal
- General publications, currently published in EU Bookshop

In the future, also:

- Tender documents and related works (OJ-S), currently published in TeD portal
- Research documents, currently published in the CORDIS portal

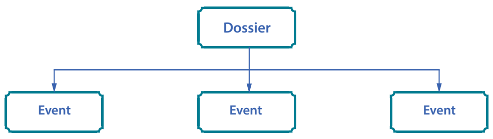
The WEM model is applied consistently throughout for works from all domains. However, the abstract classes such as works are then concretized for the various domains in the Cellar's Common Data Model (CDM) by subclassing these abstract classes. The full set of subclasses is documented in the CDM's web page. See section 6.1.

B. Hierarchy dossier-event

The dossier-event hierarchy (see Figure 8) is composed by:

- a dossier, which covers the W role of the WEMI pattern. A dossier may embed:
- several events, which cover the E role of the WEMI pattern.

Figure 9: The dossier-event hierarchy



As for works, dossiers can have specializations for each of the domains. At present there are such specializations for legislative procedures with and without inter-institutional codes to classify legislative procedures. There are also classifications for different types of events that can occur in a procedure.

C. Hierarchy event

The event also called top level event hierarchy is solely composed by an event, which covers the W role of the WEMI pattern. It's a new hierarchy: now an event can be a top level entity and not only a child of a "Dossier".

D. Hierarchy agent

The agent hierarchy is solely composed by an agent, which covers the W role of the WEMI pattern.

5.1.2. TYPES OF NOTICES

We present hereby the concept of notice, which can be subsequently divided into 5 types: tree-, branch-, object-, identifier- and rdf-notice.

For the sake of simplicity, the explanations below refer to the work-expression-manifestation-content stream hierarchy, but they can be considered valid also for the dossier-event, top level event and agent hierarchy.

5.1.2.1. Tree notice

A Tree notice is an XML document including:

– the work's metadata

– all available expressions' metadata

– all available manifestations' metadata for each expression.

All metadata is decoded in the given decoding language, that is, the language used for notices to decode NAL and EUROVOC concepts into the specific natural language. For more information about NAL and EUROVOC concepts, please consult paragraphs 5.1.4 and 5.1.5.

For more information about how to retrieve a tree notice and its format, please see paragraph 5.2.1.1.

5.1.2.2. Branch notice

A Branch notice is a content language specific XML document including:

– the work's metadata

– the metadata of the expression in the given content language

– all available manifestations' metadata for that expression.

All metadata is decoded in the given decoding language.

It is a subset of the Tree Notice.

For more information about how to retrieve a branch notice and its format, please see paragraph 5.2.1.2.

5.1.2.3. Object notice

An Object notice is a content language specific XML document with the metadata for a specific resource (work/expression/manifestation).

The metadata is decoded in the given decoding language.

It is a subset of the Tree Notice because only one object is in scope, while hierarchically dependent objects are not included (e.g. an expression, but not its manifestations).

For more information about how to retrieve an object notice and its format, please see paragraphs 5.2.1.3, 5.2.1.4 and 5.2.1.5.

5.1.2.4. Identifier notice

An Identifier notice is an XML document containing the synonyms of a list of resource URIs.

For a definition of resource URI, please see paragraph 5.1.6.

For more information about how to retrieve an identifier notice and its format, please see paragraph 5.2.1.6.

5.1.2.5. RDF-Object notice

An RDF-Object notice is the RDF/XML notice format for a specific resource (work/expression/manifestation/dossier/event/topLevelEvent/agent).

For more information about how to retrieve an RDF-Object notice and its format, please see paragraph 5.2.1.7.

5.1.2.6. RDF-Tree notice

An RDF-Tree notice is the RDF/XML notice format for the tree whose root is a specific resource (work/dossier/topLevelEvent/agent).

For more information about how to retrieve an RDF-Tree notice and its format, please see paragraph 5.2.1.7.

5.1.3. CONTENT STREAMS

The content stream physically carries the information of the manifestation that embeds it. It realizes the item of the WEMI pattern (see also paragraph 5.1.1.1).

Typically, it is a document written in the content language and format defined by the embedding manifestation: for instance, it may represent the PDF document Official Journal of the European Union C 318, Volume 52, English edition.

For more information about how to retrieve a content stream, please see paragraph 5.2.1.9.

5.1.4. NALS

The NALs (Named Authority List) are a preloaded, not modifiable, decoded-by-language set of data meant to be used by the Cellar ontology's concepts. The NAL itself is a concept defined with the resource URI:

`http://publications.europa.eu/resource/authority/*`

where * is the NAL specific class.

One exception is EUROVOC, which is defined at:

`http://eurovoc.europa.eu/100141`

5.1.5. EUROVOC

Eurovoc is the multilingual thesaurus maintained by the Publications Office of the European Union.

It exists in all the official languages of the European Union. Eurovoc is used by:

– the European Parliament

– the Publications Office of the European Union

– the national and regional parliaments in Europe

– some national government departments and European organisations.

This thesaurus serves as the basis for the domain names used in the European Union's terminology database: Inter-Active Terminology for Europe.

As stated in previous paragraph, the EUROVOC is one specific type of NAL.

5.1.6. RESOURCE URI

Each resource in the CELLAR is globally identified by a URI composed as follows:

```
http://publications.europa.eu/resource/{ps-name}/{ps-id}
```

From now on, we will refer to this URI as the resource URI.

Here follows a description of each part of the resource URI (paragraphs 5.1.6.1 and 5.1.6.2), with some examples depicted in paragraph 5.1.6.3. Finally, paragraph 5.1.6.4 describes the CURIE format.

5.1.6.1. {PS-NAME}

It identifies the name of the production system.

The CELLAR currently uses the following production system names: **`cellar, celex, oj, com, genpub, ep, jurisprudence, dd, mtf, consolidation, eurostat, eesc, cor, nim, pegase, agent, uriserv, join, swd, comnat,mdr, legissum, ecli, procedure, procedure-event, eli, immc and planjo.`**

5.1.6.2. {PS-ID}

It is the resource's unique identifier, and it has a structure that depends on the value of **`{ps-name}.`**

5.1.6.2.1. If {ps-name} is 'cellar'

`cellar` is the only production system's name reserved to the CELLAR application, and its identifiers follow the following conventions:

Table 17: Identifier's conventions for production system name cellar

Type	{ps-id}	Example
work dossier event agent	`{work-id}`	`b84f49cd-750f-11e3-8e20-01aa75ed71a1`
expression	`{work-id}.{expr-id}`	`b84f49cd-750f-11e3-8e20-01aa75ed71a1.0001`
manifestation	`{work-id}.{expr-id}.{man-id}`	`b84f49cd-750f-11e3-8e20-01aa75ed71a1.0001.03`
content stream	`{work-id}.{expr-id}.{man-id}/{cs-id}`	`b84f49cd-750f-11e3-8e20-01aa75ed71a1.0001.03/DOC_1`

where:

– **`{work-id}`** is a valid Universally Unique Identifier (UUID)

– **`{expr-id}`** is a 4-chars numeric value

– **`{man-id}`** is a 2-chars numeric value

– **`{cs-id}`** is an alphanumeric value with following pattern: DOC_x, where x is an incremental numeric value that identifies the content stream.

5.1.6.2.2. If {ps-name} is other than 'cellar'

For all other production system's names, the following conventions are used:

Table 18: Identifier's conventions for production system names other than cellar

Type	{ps-id}	Example
work dossier agent	`{work-id}`	`32006D0241`
expression	`{work-id}.{expr-id}`	`32006D0241.FRA`
manifestation	`{work-id}.{expr-id}.{man-id}`	`32006D0241.FRA.fmx4`
content stream	`{work-id}.{expr-id}.{man-id}.{cs-id}`	`32006D0241.FRA.fmx4.L_2006088FR.01006402.xml`
event	`{work-id}.{event-id}`	`11260.12796`

where:

– **`{work-id}`** is an alphanumeric value

– **`{expr-id}`** is a 3-chars ISO_639-3 language code. For the exhaustive list of supported ISO_639-3 codes, please refer to paragraph 5.4.1.

– **`{man-id}`** is an alphanumeric value identifying a file format (FORMEX, PDF, HTML, XML, etc.)
– **`{cs-id}`** is an alphanumeric value identifying the name of the content stream
– **`{event-id}`** is a numeric value.

5.1.6.3. Examples of valid resource URIs

Here follows a non-exhaustive list of examples of resource URIs that match the patterns described above:

1. The following resource URI identifies a work with ps-name of type cellar and the given ps-id:

   `http://publications.europa.eu/resource/cellar/b84f49cd-750f-11e3-8e20-01aa75ed71a1`

2. The following resource URI identifies an expression – belonging to the work at point 1) – with ps-name of type cellar and the given ps-id:

   `http://publications.europa.eu/resource/cellar/b84f49cd-750f-11e3-8e20-01aa75ed71a1.0006`

3. The following resource URI identifies a manifestation – belonging to the expression at point 2) - with ps-name of type cellar and the given ps-id:

   `http://publications.europa.eu/resource/cellar/b84f49cd-750f-11e3-8e20-01aa75ed71a1.0006.03`

4. The following resource URI identifies a content stream – belonging to the manifestation at point 3) – with ps-name of type cellar and the given ps-id:

   `http://publications.europa.eu/resource/cellar/b84f49cd-750f-11e3-8e20-01aa75ed71a1.0006.03/DOC_1`

5. The following resource URI identifies a work with ps-name of type oj and the given ps-id:

   `http://publications.europa.eu/resource/oj/JOL_2014_001_R_0001_01`

6. The following resource URI identifies a work with ps-name of type celex and the given ps-id:

   `http://publications.europa.eu/resource/celex/32014R0001`

7. The following resource URI identifies an expression – belonging to the work at point 6) - with ps-name of type celex and the given ps-id:

   `http://publications.europa.eu/resource/celex/32014R0001.FRA`

8. The following resource URI identifies a manifestation – belonging to the expression at point 7) - with ps-name of type oj and the given ps-id:

   `http://publications.europa.eu/resource/oj/JOL_2014_001_R_0001_01.FRA.fmx4`

9. The following resource URI identifies a content stream – belonging to the manifestation at point 8) - with ps-name of type oj and the given ps-id:

   ```
   http://publications.europa.eu/resource/oj/
   JOL_2014_001_R_0001_01.FRA.fmx4.L_2014001FR.01000302.xml
   ```

10. The following resource URI identifies a work with ps-name of type pegase and the given ps-id:

    ```
    http://publications.europa.eu/resource/pegase/11260
    ```

11. The following resource URI identifies an event with ps-name of type pegase and the given ps-id:

    ```
    http://publications.europa.eu/resource/pegase/11260.12796
    ```

5.1.6.4. CURIE format of a resource URI

For practical reasons, resource URIs are abbreviated onto a CURIE (Compact URI) format. This is done by making the production system name the alias of the system base URI.

For example, by declaring the namespace

```
xmlns:celex=http://publications.europa.eu/resource/celex/
```

we can abbreviate

```
http://publications.europa.eu/resource/celex/1234R5678
```

onto

```
celex:1234R5678
```

This CURIE format is important as it is massively used for identifying objects in Cellar's notices (for more info about Cellar's notices' format, please see paragraph 5.2.1).

5.2. Available services

The CELLAR API allows performing different operations on the CELLAR. Such API encapsulates all the HTTP calls to the CELLAR and exposes convenience methods allowing the user to easily retrieve the requested content.

It is hereby described how to invoke services on WEMI objects, namely:

- retrieve the tree notice of a work – see paragraph 5.2.1.1
- retrieve the branch notice of a work – see paragraph 5.2.1.2
- retrieve the object notice of an object (work, expression or manifestation) – see paragraphs 5.2.1.3, 5.2.1.4 and 5.2.1.5.
- retrieve all the identifiers of a specific document (synonyms) – paragraph 5.2.1.6
- retrieve the RDF/XML formatted metadata for a given resource – paragraph 5.2.1.7
- retrieve the RDF/XML formatted metadata for the tree whose root is a given resource – paragraph 5.2.1.8

– retrieve content streams of a work given a specific language and format – paragraph 5.2.1.9

and how to invoke services on NAL/EUROVOC objects, namely:

– retrieve a dump – paragraph 5.2.2.1
– retrieve the supported languages – paragraph 5.2.2.2
– retrieve a concept scheme – paragraph 5.2.2.3
– retrieve the concept schemes – paragraph 5.2.2.4
– retrieve a concept – paragraph 5.2.2.5
– retrieve the concept relatives – paragraph 5.2.2.6
– retrieve the top concepts – paragraph 5.2.2.7
– retrieve the domains – paragraph 5.2.2.8.

The next sections explain how to use these services, each of which is described through the following sections:

– description: a short description of what the service is supposed to do
– request, where are described:
  - the URL to invoke and its type (GET or POST)
  - the URL parameters, if any. Please note that all parameters representing an HTTP URL themselves must be URL-encoded, for example: **`http%3A%2F%2Fpublications.europa.eu%2Fresource%2Fauthority%2Ffd_330`**

    If not specified otherwise, a parameter is always mandatory
  - the HTTP headers, if any
  - a list of examples of valid requests.
– response: what the response is supposed to contain, its format, and an example of it.

5.2.1. WEMI SERVICES

We describe hereby the available services for retrieving the information related to the WEMI objects. For simplicity, they are described for the work-expression-manifestation-content stream hierarchy, but they can be considered valid also for the dossier-event, topLevelEvent and agent hierarchy (see paragraph 5.2.1.1).

Dissemination service uses a global negotiation system that returns always a "303 - See other" response. The client must enable the follow-redirect option.

5.2.1.1. Retrieve the tree notice

Description

This service allows the user to search for a complete tree notice of a given work, decoded in the given decoding language.

The returned notice will contain the work metadata, the metadata of all the expressions associated to the work, and the metadata of all the manifestations associated to the expressions.

Request

The user must fire a GET request to the following URL:

```
http://publications.europa.eu/resource/{ps-name}/{ps-id}?language={dec-lang}&filter={in_notice-only}
```

where:

- `{ps-name}` is a valid production system name (see also paragraph 5.1.6.1)
- `{ps-id}` is a valid production system id identifying a work, and compatible with its {ps-name} (see also paragraph 5.1.6.2)
- `{dec-lang}` is a 3-chars ISO_639-3 language code identifying the decoding language to use: this is the language used for decoding the NALs associated to the notice. If decoding language is not available, the default value defined in the configuration is used.
- `{in_notice-only}` is an optional boolean that indicates if the notice contains only the properties annotated with in_notice.

Please note: no matter what the request specifies, the response notice is always the filtered one. The filter parameter will stay for a transition period due to legacy reasons.

The following HTTP headers must be set on the request:

- `Accept:application/xml;notice=tree`

Here follows some examples of valid requests using cURL:

- `curl -H 'Accept:application/xml;notice=tree' http://publications.europa.eu/resource/cellar/b84f49cd-750f-11e3-8e20-01aa75ed71a1?language=eng -L`
- `curl -H 'Accept:application/xml;notice=tree' http://publications.europa.eu/resource/oj/JOL_2014_001_R_0001_01?language=eng -L`
- `curl -H 'Accept:application/xml;notice=tree' http://publications.europa.eu/resource/celex/32014R0001?language=eng -L`

Please note that the 3 requests use different production system names and identifiers, but actually retrieve the same work. These 3 synonyms are related to the same cellar id.

Response

The response is an XML-formatted tree notice containing the full hierarchy of the work, here included all the expressions of the work and all the manifestations associated to the expressions.

Here follows an example of returned notice (only the relevant information is reported):

```
<NOTICE decoding="eng" type="tree">
   <WORK>
      <URI>
         <VALUE>http://publications.europa.eu/resource/
cellar/b84f49cd-750f-11e3-8e20-01aa75ed71a1</VALUE>
         <IDENTIFIER>b84f49cd-750f-11e3-8e20-01aa75ed71a1</
IDENTIFIER>
         <TYPE>cellar</TYPE>
      </URI>
      <SAMEAS>
         <URI>
            <VALUE>http://publications.europa.eu/resource/
celex/32014R0001</VALUE>
            <IDENTIFIER>32014R0001</IDENTIFIER>
            <TYPE>celex</TYPE>
         </URI>
      </SAMEAS>
      <SAMEAS>
         <URI>
            <VALUE>http://publications.europa.eu/resource/
oj/JOL_2014_001_R_0001_01</VALUE>
            <IDENTIFIER>JOL_2014_001_R_0001_01</
IDENTIFIER>
            <TYPE>oj</TYPE>
         </URI>
      </SAMEAS>
      [...]
   </WORK>
   [...]
   <EXPRESSION>
      [content of expression 0001]
   <EXPRESSION>
   <MANIFESTATION>
      [content of manifestation 0001.01]
   <MANIFESTATION>
   <MANIFESTATION>
      [content of manifestation 0001.02]
   <MANIFESTATION>
   [...]
   <MANIFESTATION>
      [content of manifestation 0001.M]
   <MANIFESTATION>
   <EXPRESSION>
      [content of expression 0002]
   <EXPRESSION>
   <MANIFESTATION>
      [content of manifestation 0001.01]
   <MANIFESTATION>
```

```
    <MANIFESTATION>
        [content of manifestation 0002.02]
    <MANIFESTATION>
    [...]
    <MANIFESTATION>
        [content of manifestation 0002.M]
    <MANIFESTATION>
    [...]
    <EXPRESSION>
        [content of expression N]
    <EXPRESSION>
    <MANIFESTATION>
        [content of manifestation N.01]
    <MANIFESTATION>
    <MANIFESTATION>
        [content of manifestation N.02]
    <MANIFESTATION>
    [...]
    <MANIFESTATION>
        [content of manifestation N.M]
    <MANIFESTATION>
</NOTICE>
```

5.2.1.2. Retrieve the branch notice

Description

This service allows the user to search for a complete branch notice of a given work, decoded in the given decoding language.

The returned notice will contain the work metadata, the metadata of the expression in the given accept language, and the metadata of all manifestations associated to the expression.

Request

The user must fire a GET request to the following URL:

```
http://publications.europa.eu/resource/{ps-name}/{ps-id}?language={dec-lang}&filter={in_notice-only}
```

where:

– `{ps-name}` is a valid production system name

– `{ps-id}` is a valid production system id identifying a work, and compatible with its {ps-name}

– `{dec-lang}` is a 3-chars ISO_639-3 language code identifying the decoding language to use: this is the language used for decoding the NALs associated to the notice. If decoding language is not available, the default value defined in the configuration is used.

- **{in_notice-only}** is an optional boolean that indicates if the notice contains only the properties annotated with in_notice.

Please note: no matter what the request specifies, the response notice is always the filtered one. The filter parameter will stay for a transition period due to legacy reasons.

The following HTTP headers must be set on the request:

- **Accept:application/xml;notice=branch**
- **Accept-Language:{acc-lang}**, where **{acc-lang}** is a 3-chars ISO_639-3 language code identifying the accept language to use: this will be used for retrieving the correct expression.

Here follows some examples of valid requests that retrieve the same object, using cURL:

- `curl -H 'Accept:application/xml;notice=branch' -H 'Accept-Language:fra' http://publications.europa.eu/resource/cellar/b84f49cd-750f-11e3-8e20-01aa75ed71a1?language=eng -L`
- `curl -H 'Accept:application/xml;notice=branch' -H 'Accept-Language:fra' http://publications.europa.eu/resource/oj/JOL_2014_001_R_0001_01?language=en -L`
- `curl -H 'Accept:application/xml;notice=branch' -H 'Accept-Language:fra' http://publications.europa.eu/resource/celex/32014R0001?language=eng -L`

Response

The response is an XML-formatted branch notice containing the work, within the expression in the given accept language, and all the associated manifestations.

Here follows an example of returned notice:

```
<NOTICE decoding="eng" type="branch">
   <WORK>
      <URI>
         <VALUE> http://publications.europa.eu/resource/
cellar/b84f49cd-750f-11e3-8e20-01aa75ed71a1</VALUE>
         <IDENTIFIER> b84f49cd-750f-11e3-8e20-01aa75ed71a1 </
IDENTIFIER>
         <TYPE>cellar</TYPE>
      </URI>
      <SAMEAS>
         <URI>
            <VALUE>http://publications.europa.eu/resource/
celex/32014R0001</VALUE>
            <IDENTIFIER>32014R0001</IDENTIFIER>
            <TYPE>celex</TYPE>
         </URI>
      </SAMEAS>
      <SAMEAS>
```

```
          <URI>
          <VALUE>http://publications.europa.eu/resource/oj/
JOL_2014_001_R_0001_01</VALUE>
             <IDENTIFIER>JOL_2014_001_R_0001_01</
IDENTIFIER>
             <TYPE>oj</TYPE>
          </URI>
       </SAMEAS>
       [...]
    </WORK>
    [...]
    <EXPRESSION>
       [content of expression X in given language {acc-
lang}]
    <EXPRESSION>
    <MANIFESTATION>
       [content of manifestation X.01]
    <MANIFESTATION>
    <MANIFESTATION>
       [content of manifestation X.02]
    <MANIFESTATION>
    [...]
    <MANIFESTATION>
       [content of manifestation X.M]
    <MANIFESTATION>
</NOTICE>
```

5.2.1.3. Retrieve the object work notice

Description

This service allows the user to search for the object notice of the given work, decoded in the given decoding language.

Only the metadata of the work are returned in the notice, with no expression or manifestation.

Request

The user must fire a GET request to the following URL:

`http://publications.europa.eu/resource/{ps-name}/{ps-id}?language={dec-lang}&filter={in_notice-only}`

where:

- **`{ps-name}`** is a valid production system name
- **`{ps-id}`** is a valid production system id identifying a work, and compatible with its {ps-name}
- **`{dec-lang}`** is a 3-chars ISO_639-3 language code identifying the decoding language to use: this is the language used for decoding the NALs associated to

the notice. If decoding language is not available, the default value defined in the configuration is used.

- **{in_notice-only}** is an optional boolean that indicates if the notice contains only the properties annotated with in_notice.

Please note: no matter what the request specifies, the response notice is always the filtered one. The filter parameter will stay for a transition period due to legacy reasons.

The following HTTP headers must be set on the request:

- **Accept:application/xml;notice=object**

Here follows some examples of valid requests that retrieve the same object, using cURL:

- **curl -H 'Accept:application/xml;notice=object' http://publications.europa.eu/resource/cellar/b84f49cd-750f-11e3-8e20-01aa75ed71a1?language=eng -L**
- **curl -H 'Accept:application/xml;notice=object' http://publications.europa.eu/resource/oj/JOL_2014_001_R_0001_01?language=eng -L**
- **curl -H 'Accept:application/xml;notice=object' http://publications.europa.eu/resource/celex/32014R0001?language=eng -L**

Response

The response is an XML-formatted object notice containing the metadata of the work only.

Here follows an example of returned notice:

```
<NOTICE decoding="eng" type="object">
   <WORK>
      <URI>
         <VALUE>http://publications.europa.eu/resource/
cellar/b84f49cd-750f-11e3-8e20-01aa75ed71a1</VALUE>
         <IDENTIFIER>b84f49cd-750f-11e3-8e20-01aa75ed71a1</
IDENTIFIER>
         <TYPE>cellar</TYPE>
      </URI>
      <SAMEAS>
         <URI>
            <VALUE>http://publications.europa.eu/resource/
celex/32014R0001</VALUE>
            <IDENTIFIER>32014R0001</IDENTIFIER>
            <TYPE>celex</TYPE>
         </URI>
      </SAMEAS>
      <SAMEAS>
         <URI>
```

```
              <VALUE>http://publications.europa.eu/resource/
oj/JOL_2014_001_R_0001_01</VALUE>
              <IDENTIFIER>JOL_2014_001_R_0001_01</
IDENTIFIER>
              <TYPE>oj</TYPE>
          </URI>
       </SAMEAS>
       [...]
    </WORK>
    [...]
</NOTICE>
```

5.2.1.4. Retrieve the object.expression notice

Descri ption

This service allows the user to search for the expression of the given work, decoded in the given decoding language.

The returned notice will contain the metadata of the expression in the given accept language, with no metadata of the work or manifestations.

Request

The user must fire a GET request to the following URL:

`http://publications.europa.eu/resource/{ps-name}/{ps-id}?language={dec-lang}&filter={in_notice-only}`

where:

– **`{ps-name}`** is a valid production system name

– **`{ps-id}`** is a valid production system id identifying a work, and compatible with its {ps-name}

– **`{dec-lang}`** is a 3-chars ISO_639-3 language code identifying the decoding language to use: this is the language used for decoding the NALs associated to the notice. If decoding language is not available, the default value defined in the configuration is used.

– **`{in_notice-only}`** is an optional boolean that indicates if the notice contains only the properties annotated with in_notice.

Please note: no matter what the request specifies, the response notice is always the filtered one. The filter parameter will stay for a transition period due to legacy reasons.

The following HTTP headers must be set on the request:

– **`Accept:application/xml;notice=object`**

– **`Accept-Language:{acc-lang},`** where **`{acc-lang}`** is a 3-chars ISO_639-3 language code identifying the accept language to use: this will be used for retrieving the correct expression.

Here follows some examples of valid requests that retrieve the same object, using cURL:

- `curl -H 'Accept:application/xml;notice=object' -H 'Accept-Language:fra' http://publications.europa.eu/resource/cellar/b84f49cd-750f-11e3-8e20-01aa75ed71a1?language=eng -L`
- `curl -H 'Accept:application/xml;notice=object' -H 'Accept-Language:fra' http://publications.europa.eu/resource/oj/JOL_2014_001_R_0001_01?language=eng -L`
- `curl -H 'Accept:application/xml;notice=object' -H 'Accept-Language:fra' http://publications.europa.eu/resource/celex/32014R0001?language=eng -L`

Response

The response is an XML-formatted object notice containing the metadata of the expression only.

Here follows an example of returned notice:

```
<NOTICE decoding="eng" type="object">
   <EXPRESSION>
      <URI>
         <VALUE>http://publications.europa.eu/resource/
cellar/b84f49cd-750f-11e3-8e20-01aa75ed71a1.0010</VALUE>
         <IDENTIFIER>b84f49cd-750f-11e3-8e20-
01aa75ed71a1.0010</IDENTIFIER>
         <TYPE>cellar</TYPE>
      </URI>
      <SAMEAS>
         <URI>
           <VALUE>http://publications.europa.eu/resource/
celex/32014R0001.FRA</VALUE>
            <IDENTIFIER>32014R0001.FRA</IDENTIFIER>
            <TYPE>celex</TYPE>
         </URI>
      </SAMEAS>
      <SAMEAS>
         <URI>
           <VALUE>http://publications.europa.eu/resource/
uriserv/OJ.L_.2014.001.01.0001.01.FRA</VALUE>
            <IDENTIFIER>OJ.L_.2014.001.01.0001.01.FRA</
IDENTIFIER>
            <TYPE>uriserv</TYPE>
         </URI>
      </SAMEAS>
      [...]
   </EXPRESSION>
</NOTICE>
```

5.2.1.5. Retrieve the object manifestation notice

Description

This service allows the user to search for the object notice of the given manifestation, decoded in the given decoding language.

Only the metadata of the manifestation are returned in the notice, with no work or expressions.

Request

The user must fire a GET request to the following URL:

`http://publications.europa.eu/resource/{ps-name}/{ps-id}?language={dec-lang}&filter={in_notice-only}`

where:

– **`{ps-name}`** is a valid production system name

– **`{ps-id}`** is a valid production system id identifying a manifestation, and compatible with its {ps-name}

– **`{dec-lang}`** is a 3-chars ISO_639-3 language code identifying the decoding language to use: this is the language used for decoding the NALs associated to the notice. If decoding language is not available, the default value defined in the configuration is used.

– **`{in_notice-only}`** is an optional boolean that indicates if the notice contains only the properties annotated with **`in_notice`**.

Please note: no matter what the request specifies, the response notice is always the filtered one. The filter parameter will stay for a transition period due to legacy reasons.

The following HTTP headers must be set on the request:

– **`Accept:application/xml;notice=object`**

The following HTTP header can be set on the request:

– **`Negotiate:vlist`**

If it is present, the response will include an Alternates header indicating all alternative representations of the returned object

Here follows some examples of valid requests that retrieve the same object, using cURL:

–
```
curl -H 'Accept:application/xml;notice=object' http://
publications.europa.eu/resource/cellar/b84f49cd-750f-
11e3-8e20-01aa75ed71a1.0010.03?language=eng -L
```

–
```
curl -H 'Accept:application/xml;notice=object'
http://publications.europa.eu/resource/oj/
JOL_2014_001_R_0001_01.FRA.xhtml?language=eng -L
```

```
– curl -H 'Accept:application/xml;notice=object' http://
  publications.europa.eu/resource/celex/32014R0001.FRA.
  print?language=eng -L
```

Response

The response is an XML-formatted object notice containing the metadata of the manifestation only.

Here follows an example of returned notice:

```
<NOTICE decoding="eng" type="object">
   <MANIFESTATION manifestation-type="xhtml">
      <URI>
         <VALUE>http://publications.europa.eu/resource/
cellar/b84f49cd-750f-11e3-8e20-01aa75ed71a1.0010.03</VALUE>
         <IDENTIFIER>b84f49cd-750f-11e3-8e20-
01aa75ed71a1.0010.03</IDENTIFIER>
         <TYPE>cellar</TYPE>
      </URI>
      <SAMEAS>
         <URI>
            <VALUE>http://publications.europa.eu/resource/
oj/JOL_2014_001_R_0001_01.FRA.xhtml</VALUE>
            <IDENTIFIER>JOL_2014_001_R_0001_01.FRA.xhtml</
IDENTIFIER>
            <TYPE>oj</TYPE>
         </URI>
      </SAMEAS>
      <SAMEAS>
         <URI>
            <VALUE>http://publications.europa.eu/resource/
uriserv/OJ.L_.2014.001.01.0001.01.FRA.xhtml</VALUE>
            <IDENTIFIER>OJ.L_.2014.001.01.0001.01.FRA.
xhtml</IDENTIFIER>
            <TYPE>uriserv</TYPE>
         </URI>
      </SAMEAS>
      <MANIFESTATION_TYPE type="data">
         <VALUE>xhtml</VALUE>
      </MANIFESTATION_TYPE>
      [...]
   </MANIFESTATION>
</NOTICE>
```

5.2.1.6. Retrieve the identifier notice

Description

This service allows the user to retrieve the synonyms of a given resource URI.

Request

The user must fire a GET request to the following URL:

`http://publications.europa.eu/resource/{ps-name}/{ps-id}`

where:

– `{ps-name}` is a valid production system name

– `{ps-id}` is a valid production system id identifying a work, an expression, a manifestation, an item, a dossier, an event, a top level event or an agent and is compatible with its {ps-name}

The following HTTP header must be set on the request:

– `Accept:application/xml;notice=identifiers`

Here follow some examples of valid requests that retrieve different objects, using cURL:

– `curl -H 'Accept:application/xml;notice=identifiers' http://publications.europa.eu/resource/cellar/b84f49cd-750f-11e3-8e20-01aa75ed71a1 -L`

– `curl -H 'Accept:application/xml;notice=identifiers' http://publications.europa.eu/resource/celex/32014R0001.FRA.print -L`

– `curl -H 'Accept:application/xml;notice=identifiers' http://publications.europa.eu/resource/oj/JOL_2014_001_R_0001_01.FRA.fmx4.L_2014001FR.01000101.xml -L`

– `curl -H 'Accept:application/xml;notice=identifiers' http://publications.europa.eu/resource/oj/JOL_2006_088_R_0063_01.FRA.fmx4.L_2006088FR.01006301.xml -L`

– `curl -H 'Accept:application/xml;notice=identifiers' http://publications.europa.eu/resource/pegase/11260.12796 -L`

Response

The response is an XML-formatted notice containing the URI of the cellar ID and its synonym(s).

```
<?xml version="1.0" encoding="UTF-8"?>
<NOTICE type="identifier">
    <URI>
        <VALUE>http://publications.europa.eu/resource/
cellar/32a58fc1-cffa-11e1-96ce-01aa75ed71a1.0003</VALUE>
        <TYPE>cellar</TYPE>
       <IDENTIFIER>32a58fc1-cffa-11e1-96ce-01aa75ed71a1.0003</
IDENTIFIER>
    </URI>
    <SAMEAS>
        <URI>
            <VALUE>http://publications.europa.eu/resource/
pegase/11260.12796</VALUE>
```

```
            <TYPE>pegase</TYPE>
            <IDENTIFIER>11260.12796</IDENTIFIER>
        </URI>
    </SAMEAS>
</NOTICE>
```

5.2.1.7. Retrieve the RDF/XML formatted metadata for a given resource

Description

This service allows the user to search for the RDF (Resource Description Framework) content of the given object. The object to search for can be a work, an expression, a manifestation, a dossier, an event, a top level event or an agent.

Request

The user must fire a GET request to the following URL:

`http://publications.europa.eu/resource/{ps-name}/{ps-id}?language={dec-lang}`

where:

– **`{ps-name}`** is a valid production system name

– **`{ps-id}`** is a valid production system id identifying a work, and compatible with its {ps-name}

The following HTTP headers may be set on the request:

– **`Accept:application/rdf+xml`**

In this case, the resulting RDF notice will contain the direct and inferred triples.

– **`Accept:application/rdf+xml;notice=non-inferred`**

In this case, the inferred triples will be excluded from the resulting RDF notice.

– **`Negotiate:vlist`**

If it is present, the response will include an Alternates header indicating all alternative representations of the returned object. Currently, this header is supported only for requests on manifestation level.

If the Accept header is not present, * or */* and the production identifier matches a WEM object, it will behave like if set to Accept:application/rdf+xml.

Here follows an example of valid request that retrieve the same RDF, using cURL:

– **`curl http://publications.europa.eu/resource/cellar/b84f49cd-750f-11e3-8e20-01aa75ed71a1 -L`**

– **`curl http://publications.europa.eu/resource/oj/JOL_2014_001_R_0001_01 -L`**

– **`curl http://publications.europa.eu/resource/celex/32014R0001 -L`**

- `curl -H 'Accept: application/rdf+xml' http://publications.europa.eu/resource/celex/32014R0001 -L`
- `curl -H 'Accept:' http://publications.europa.eu/resource/celex/32014R0001 -L`
- `curl -H 'Accept: *' http://publications.europa.eu/resource/celex/32014R0001 -L`
- `curl -H 'Accept:*/*' http://publications.europa.eu/resource/celex/32014R0001 -L`

Response

The response is an XML-formatted sheet containing the RDF metadata of the object.

Here follows an example of returned notice:

```
<rdf:RDF [...] >
   <rdf:Description rdf:about="http://publications.europa.
eu/resource/oj/JOL_2014_001_R_0001_01.ELL">
    <rdf:type rdf:resource="http://publications.europa.eu/
ontology/cdm#expression"/>
   </rdf:Description>
   <rdf:Description rdf:about="http://publications.
europa.eu/resource/cellar/b84f49cd-750f-11e3-8e20-
01aa75ed71a1.0022">
    <owl:sameAs rdf:resource="http://publications.europa.
eu/resource/oj/JOL_2014_001_R_0001_01.SLV"/>
   </rdf:Description>
   <rdf:Description rdf:about="http://publications.europa.
eu/resource/celex/32013R1421">
      <j.0:resource_legal_consolidated_by_act_
consolidated rdf:resource="http://publications.europa.eu/
resource/celex/02012R0978-20141001"/>
    <j.0:consolidated_by rdf:resource="http://publications.
europa.eu/resource/celex/02012R0978-20141001"/>
    <j.0:resource_legal_consolidated_by_act_consolidated
rdf:resource="http://publications.europa.eu/resource/
celex/02012R0978-20150101"/>
    <j.0:consolidated_by rdf:resource="http://publications.
europa.eu/resource/celex/02012R0978-20150101"/>
   </rdf:Description>
   [...]
</rdf:RDF>
```

5.2.1.8. Retrieve the RDF/XML formatted metadata of the tree whose root is a given resource

Description

This service allows the user to search for the RDF (Resource Description Framework) tree whose root is the given object. The object to search for can be a work, a dossier, a top level event or an agent.

Request

The user must fire a GET request to the following URL:

```
http://publications.europa.eu/resource/{ps-name}/{ps-
id}?language={dec-lang}
```

where:

– **`{ps-name}`** is a valid production system name
– **`{ps-id}`** is a valid production system id identifying a work, and compatible with its {ps-name}

The following HTTP headers may be set on the request:

– **`Accept:application/rdf+xml;notice=tree`**

In this case, the resulting RDF notice will contain the direct and inferred triples.

– **`Accept:application/rdf+xml;notice=non-inferred-tree`**

In this case, the inferred triples will be excluded from the resulting RDF notice.

Here follows some examples of valid requests that retrieve the same RDF, using cURL :

– `curl -H 'Accept:application/rdf+xml;notice=tree' http://publications.europa.eu/resource/cellar/b84f49cd-750f-11e3-8e20-01aa75ed71a1 -L`
– `curl -H 'Accept:application/rdf+xml;notice=tree' http://publications.europa.eu/resource/oj/JOL_2014_001_R_0001_01 -L`
– `curl -H 'Accept:application/rdf+xml;notice=tree' http://publications.europa.eu/resource/celex/32014R0001 -L`

Response

The response is an XML-formatted sheet containing the RDF metadata of the tree.

Here follows an example of returned notice:

```
<rdf:RDF [...] >
   <rdf:Description rdf:about="http://publications.europa.
eu/resource/authority/language/EST">
    <rdf:type rdf:resource="http://publications.europa.eu/
ontology/cdm#language"/>
    <rdf:type rdf:resource="http://www.w3.org/2004/02/skos/
```

```
core#Concept"/>
    <j.1:inScheme rdf:resource="http://publications.europa.
eu/resource/authority/language"/>
    <j.0:language_used_by_expression rdf:resource="http://
publications.europa.eu/resource/oj/JOL_2014_001_R_0001_01.
EST"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://publications.
europa.eu/resource/cellar/b84f49cd-750f-11e3-8e20-
01aa75ed71a1.0005.01">
    <j.2:metsStructSuperDiv rdf:resource="http://
publications.europa.eu/resource/cellar/b84f49cd-750f-11e3-
8e20-01aa75ed71a1.0005"/>
    <j.2:lastModificationDate rdf:datatype="http://
www.w3.org/2001/XMLSchema#dateTime">2014-01-
04T08:10:26.028+01:00</j.2:lastModificationDate>
    <owl:sameAs rdf:resource="http://publications.europa.
eu/resource/uriserv/OJ.L_.2014.001.01.0001.01.ELL.pdfa1a"/>
    <j.0:manifestation_has_item rdf:resource="http://
publications.europa.eu/resource/cellar/b84f49cd-750f-11e3-
8e20-01aa75ed71a1.0005.01/DOC_1"/>
    [...]
  </rdf:Description>
   [...]
</rdf:RDF>
```

5.2.1.9. Retrieve content streams

Description

This service allows the user to retrieve the content stream of the manifestation belonging to the given work and to the expression in the given accept language, and which contains at least 1 content stream of the given accept format.

Request

The user must fire a GET request to the following URL:

```
http://publications.europa.eu/resource/{ps-name}/{ps-
id}?language={dec-lang}
```

where:

- `{ps-name}` is a valid production system name
- `{ps-id}` is a valid production system id identifying a work, and compatible with its `{ps-name}`

The following HTTP headers must be set on the request:

- `Accept:{mime-type},` where `{mime-type}` is a valid (or a comma-separated list of) mimetype that identify the format of the content stream to return. Possible values are:

- application/epub+zip
- application/msword
- application/pdf
- application/pdf;type=pdf1x
- application/pdf;type=pdfa1a
- application/pdf;type=pdfa1b
- application/pdf;type=pdfx
- application/rdf+xml
- application/sparql-query
- application/sparql-results+xml
- application/vnd.amazon.ebook
- application/vnd.ms-excel
- application/vnd.ms-powerpoint
- application/vnd.openxmlformats-officedocument.presentationml.presentation
- application/vnd.openxmlformats-officedocument.presentationml.slideshow
- application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
- application/vnd.openxmlformats-officedocument.wordprocessingml.document.main+xml
- application/x-mobipocket-ebook
- application/xhtml+xml
- application/xhtml+xml;type=simplified
- application/xml
- application/xml;type=fmx2
- application/xml;type=fmx3
- application/xml;type=fmx4
- application/xslt+xml
- application/zip
- image/gif
- image/jpeg
- image/png
- image/tiff
- image/tiff-fx
- text/html
- text/html;type=simplified
- text/plain
- text/rtf
- text/sgml
- text/sgml;type=fmx2
- text/sgml;type=fmx3

- **`Accept-Language:{acc-lang}`,** where **`{acc-lang}`** is a 3-chars ISO_639-3 language code identifying the accept language to use: this will be used for retrieving the correct expression
- **`Accept-Max-Cs-Size:{size}`**, where **`{size}`** is a positive integer (max. value = 263-1) which specifies the max. content stream size in bytes. If the actual content stream size is bigger than specified, a "406 - Not Acceptable" response is given.

Here follows some examples of valid request that retrieve the same content stream, using cURL:

- **`curl -H 'Accept:application/xhtml+xml' -H 'Accept-Language:fra' http://publications.europa.eu/resource/cellar/b84f49cd-750f-11e3-8e20-01aa75ed71a1 -L`**
- **`curl -H 'Accept:application/xhtml+xml' -H 'Accept-Language:fra' http://publications.europa.eu/resource/oj/JOL_2014_001_R_0001_01 -L`**
- **`curl -H 'Accept:application/xhtml+xml' -H 'Accept-Language:fra' –H 'Accept-Max-Cs-Size:209715200' http://publications.europa.eu/resource/celex/32014R0001 -L`**

Response

The associated content stream.

5.2.1.10. Retrieve content stream collections

Description

This service allows the user to retrieve a collection (in zip or list format) of the content streams of the manifestation belonging to the given work and to the expression in the given accept language, and which contains at least 1 content stream of the given accept format.

Request

The user must fire a GET request to the following URL:

```
http://publications.europa.eu/resource/{ps-name}/{ps-id}?language={dec-lang}
```

where:

- **`{ps-name}`** is a valid production system name
- **`{ps-id}`** is a valid production system id identifying a work, and compatible with its **`{ps-name}`**

The following HTTP headers must be set on the request:

- **`Accept:{mime-type}`**, where **`{mime-type}`** is a valid (or a comma-separated list of) mimetype that identify the format of the content stream to return. Possible values are:

- **`application/list;mtype={manifestation-type}`**
- **`application/zip;mtype={manifestation-type}`**

The mtype token carries the **`{manifestation-type}`**, which must be set to the value of cdm:manifestation_type of the desired manifestation

– **`Accept-Language:{acc-lang}`**, where **`{acc-lang}`** is a 3-chars ISO_639-3 language code identifying the accept language to use: this will be used for retrieving the correct expression

Here follows some examples of valid request that retrieve the same content stream, using cURL:

– **`curl -H 'Accept:application/zip;mtype=fmx4' -H 'Accept-Language:fra' http://publications.europa.eu/resource/cellar/b84f49cd-750f-11e3-8e20-01aa75ed71a1 -L`**

– **`curl -H 'Accept:application/list;mtype=fmx4' -H 'Accept-Language:fra' http://publications.europa.eu/resource/cellar/b84f49cd-750f-11e3-8e20-01aa75ed71a1 -L`**

Response

The associated content streams in the requested format:

– **zip:** a zip file containing all content stream files of the requested manifestation

– **list:** an html list containing all content stream file names of the requested manifestation

Note: If the given resource is a manifestation and the mtype token does not match its type, the mtype token is ignored and content streams of the given manifestation are returned.

5.2.2 NAL/EUROVOC SERVICES

We describe hereby the available services for retrieving the information related to the NAL/EUROVOC objects.

Some of the services below rely heavily on the notions of:

– concept, which is the class defined by the resource URI **`http://publications.europa.eu/ontology/cdm#concept.`**

It is the superclass of all concepts used in Cellar's ontology and a direct subclass of the SKOS concept (**`http://www.w3.org/2004/02/skos/core#Concept`**), thus it can be seen as the topmost class of Cellar's ontology

– concept scheme, which has the same meaning as the SKOS concept scheme (**`http://www.w3.org/2004/02/skos/core#ConceptScheme`**): an aggregation of one or more concepts.

Semantic relationships (links) between those concepts may also be viewed as part of a concept scheme. This definition is, however, meant to be suggestive rather than restrictive, and there is some flexibility in the formal data model of the Cellar.

5.2.2.1. Retrieve a NAL table

The REST endpoint **`http://publications.europa.eu/webapi/authority-table`** will be removed and the two new endpoints described below will take its place.

This service allows the user to retrieve the complete dump of a NAL or EUROVOC object. CELLAR currently exposes two REST endpoints for the NAL. The first one can be used to retrieve a particular NAL while the second one can be used to retrieve the list of operational NAL in CELLAR:

– **`/nal/list`** which returns a list of all the NAL URIs currently operational in CELLAR

– **`/nal/get`** which retrieves the NAL referenced by the NAL URI argument

5.2.2.1.1. REST endpoint /nal/list

Request

The user must fire a GET request to the following URL:

`http://publications.europa.eu/webapi/nal/list`

Response

The response will be a *comma separated list* of all the NAL URIs currently in use by CELLAR.

```
[http://publications.europa.eu/resource/authority/fd_100,
http://publications.europa.eu/resource/authority/fd_606,
http://publications.europa.eu/resource/authority/fd_557, ...]
```

5.2.2.1.2. REST endpoint /nal/get

This endpoint allows the retrieval of the NAL in RDF format. The XML version of NAL was deprecated in previous versions of CELLAR.

Request

The user must fire a GET request to the following URL:

`http://publications.europa.eu/webapi/nal/get?nalUri={nalUri}`

where **`{nalUri}`** is a NAL URI that can be retrieved from REST endpoint **`/nal/list`** (see previous section).

Response

The response content will be of type **`application/rdf+xml`** returned as an XML-formatted SKOS/RDF.

Here follow some valid examples of request:

– **`http://publications.europa.eu/webapi/nal/get?nalUri=http://publications.europa.eu/resource/authority/file-type`**

– **`http://publications.europa.eu/webapi/nal/get?nalUri=http://eurovoc.europa.eu/100141`**

5.2.2.2. Retrieve the supported languages

Description

This service allows the user to retrieve the supported languages of the system. Also, the user may ask for the supported languages of a particular NAL/EUROVOC concept scheme.

Request

The user must fire a GET request to the following URL:

```
http://publications.europa.eu/webapi/{type}/
getSupportedLanguages?concept_scheme={cs-uri}
```

where:

– `{type}` can be either nal or eurovoc, depending on whether the user wants to retrieve the supported languages for NAL or EUROVOC objects, respectively

– `{cs-uri}` is the resource URI of the NAL/EUROVOC concept scheme.

This parameter is not mandatory: if not specified, all supported languages of the system will be retrieved.

Here follows some examples of valid requests:

– `http://publications.europa.eu/webapi/nal/getSupportedLanguages`

– `http://publications.europa.eu/webapi/nal/getSupportedLanguages?concept_scheme=http%3A%2F%2Fpublications.europa.eu%2Fresource%2Fauthority%2Ffd_330`

– `http://publications.europa.eu/webapi/eurovoc/getSupportedLanguages`

Response

The list of supported languages in JSON format. For more information about JSON format, please see Annexe 3.

Example:

```
[
  {
    "code": "mlt"
  },
  {
    "code": "deu"
  },
  [...other languages]
]
```

5.2.2.3. Retrieve a concept scheme

Description

This service allows the user to retrieve a NAL or EUROVOC concept scheme.

Request

The user must fire a GET request to the following URL:

`http://publications.europa.eu/webapi/{type}/getConceptScheme?concept_Scheme={cs-uri}`

where

- **`{type}`** can be either nal or eurovoc, depending on whether the user wants to retrieve a NAL or an EUROVOC concept scheme, respectively
- **`{cs-uri}`** is the resource URI of the NAL/EUROVOC concept scheme.

This parameter is mandatory only for NALs (that is, when {type} is nal): if not specified for EUROVOCs ({type} is eurovoc), it defaults to http://eurovoc.europa.eu/100141.

Here follows some examples of valid requests:

- **`http://publications.europa.eu/webapi/nal/getConceptScheme?concept_scheme=http%3A%2F%2Fpublications.europa.eu%2Fresource%2Fauthority%2Ffd_330`**
- **`http://publications.europa.eu/webapi/eurovoc/getConceptScheme?concept_scheme=http%3A%2F%2Feurovoc.europa.eu%2F100225`**
- **`http://publications.europa.eu/webapi/eurovoc/getConceptScheme`**

Response

The concept scheme in JSON format.

Example:

```
{
  "date": null,
  "lastModified": null,
  "version": null,
  "uri": {
    "uri": "http://eurovoc.europa.eu/100225"
  },
  "labels": [
    {
      "language": "ron",
      "string": "3611  tiin e umaniste"
    },
    {
      "language": "hun",
      "string": "3611 humán tudományok"
    },
    [...other labels]
  ]
}
```

5.2.2.4. Retrieve the concept schemes

Description

This service allows the user to retrieve all the concept schemes of NALs or EUROVOCs.

Request

The user must fire a GET request to the following URL:

`http://publications.europa.eu/webapi/{type}/getConceptSchemes`

where `{type}` can be either nal or eurovoc, depending on whether the user wants to retrieve the concept schemes of NALs or EUROVOCs, respectively.

Here follows some examples of valid requests:

– `http://publications.europa.eu/webapi/nal/getConceptSchemes`

– `http://publications.europa.eu/webapi/eurovoc/getConceptSchemes`

Response

The list of concept schemes in JSON format.

Example:

```
[
  {
    "date": null,
    "lastModified": null,
    "version": null,
    "uri": {
      "uri": "http://eurovoc.europa.eu/100225"
    },
    "labels": [
      {
        "language": "ron",
        "string": "3611  tiin e umaniste"
      },
      {
        "language": "hun",
        "string": "3611 humán tudományok"
      },
      [...other labels]
    ]
  },
  {
    "date": null,
      "lastModified": null,
      "version": null,
      "uri": {
        "uri": "http://eurovoc.europa.eu/100226"
      },
      "labels": [
        {
          "language": "ron",
          "string": "4006 organizarea afacerilor"
        },
        {
          "language": "hun",
          "string": "4006 gazdasági szervezetek"
        },
        [...other labels]
      ]
    },
    [...other concepts schemes]
]
```

5.2.2.5. Retrieve a concept

Description

This service allows the user to retrieve the translation of a given concept into a specified language.

Request

The user must fire a GET request to the following URL:

`http://publications.europa.eu/webapi/{type}/getConcept?concept_uri={con-uri}&language={lang}`

where:

– **`{type}`** can be either nal or eurovoc, depending on whether the user wants to retrieve the translation of a NAL or EUROVOC concept, respectively

– **`{con-uri}`** is the resource URI of the NAL/EUROVOC concept

– **`{lang}`** is a 3-chars ISO_639-3 language code identifying the language the user wants to translate the concept with.

Here follows some examples of valid requests:

– **`http://publications.europa.eu/webapi/nal/getConcept?concept_uri=http%3A%2F%2Fpublications.europa.eu%2Fresource%2Fauthority%2Ffd_330&language=eng`**

– **`http://publications.europa.eu/webapi/eurovoc/getConcept?concept_uri=http%3A%2F%2Feurovoc.europa.eu%2F100225&language=fra`**

Response

The translated concept in JSON format.

Example:

```
[
  {
    "language": "fra",
    "identifier": "3928",
    "notations": [

    ],
    "uri": {
      "uri": "http://eurovoc.europa.eu/3928"
    },
    "prefLabel": {
      "language": "fra",
      "string": "sciences du comportement"
    },
    "altLabels": [
      "psychologie du comportement",
      "behaviorisme"
    ],
    "hiddenLabels": [
      "comportement, psychologie du",
      "comportement, sciences du"
    ]
  }
]
```

5.2.2.6. Retrieve the concept relatives

Description

This service allows the user to retrieve the list of concepts having a specific semantic relation with the given concept.

Request

The user must fire a GET request to the following URL:

`http://publications.europa.eu/webapi/{type}/getConceptRelatives?concept_uri={con-uri}&relation_uri={rel-uri}&language={lang}`

where:

- **`{type}`** can be either nal or eurovoc, depending on whether the user wants to retrieve the concept relatives of a NAL or EUROVOC concept, respectively
- **`{con-uri}`** is the resource URI of the NAL/EUROVOC concept of which retrieving the concept relatives

– **`{rel-uri}`** is the resource URI of the SKOS relation scheme to use, namely:
  - **`http://www.w3.org/2004/02/skos/core#broader`**: to use in order to retrieve the concepts that are more general in meaning than the given concept. Broader concepts are typically rendered as parents in a concept hierarchy
  - **`http://www.w3.org/2004/02/skos/core#narrower`**: to use in order to retrieve the concepts that are more specific in meaning than the given concept. Narrower concepts are typically rendered as children in a concept hierarchy
  - **`http://www.w3.org/2004/02/skos/core#related`**: to use in order to retrieve the concepts that have an associative semantic relationship with the given concept

– **`{lang}`** is a 3-chars ISO_639-3 language code identifying the language the user wants to retrieve the concept relatives with.

Here follows some examples of valid requests:

– **`http://publications.europa.eu/webapi/nal/getConceptRelatives?concept_uri=http%3A%2F%2Fpublications.europa.eu%2Fresource%2Fauthority%2Ffd_330&relation_uri=http%3A%2F%2Fwww.w3.org%2F2004%2F02%2Fskos%2Fcore%23broader&language=eng`**

– **`http://publications.europa.eu/webapi/eurovoc/getConceptRelatives?concept_uri=http%3A%2F%2Feurovoc.europa.eu%2F100225&relation_uri=http%3A%2F%2Fwww.w3.org%2F2004%2F02%2Fskos%2Fcore%23narrower&language=fra`**

Response

The list of concepts that have a semantic relation with the given concept, in JSON format.

Example:

```
[
  {
    "language": "fra",
    "identifier": "3928",
    "notations": [

    ],
    "uri": {
      "uri": "http://eurovoc.europa.eu/3928"
    },
    "prefLabel": {
      "language": "fra",
      "string": "sciences du comportement"
    },
    "altLabels": [
      "psychologie du comportement",
      "behaviorisme"
    ],
    "hiddenLabels": [
      "comportement, psychologie du",
      "comportement, sciences du"
    ]
  },
  {
    "language": "fra",
    "identifier": "3956",
    "notations": [

    ],
    "uri": {
      "uri": "http://eurovoc.europa.eu/3956"
    },
    "prefLabel": {
      "language": "fra",
      "string": "sciences sociales"
    },
    "altLabels": [
      "sciences humaines"
    ],
    "hiddenLabels": [
      "sociales, sciences",
      "humaines, sciences"
    ]
  },
  [...other concepts]
]
```

5.2.2.7. Retrieve the top concepts

Description

This service allows the user to retrieve the top concepts of a given concept scheme in a specified language.

A top concept is a concept that is topmost in the broader/narrower concept hierarchies for a given concept scheme, providing an entry point to these hierarchies.

Request

The user must fire a GET request to the following URL:

```
http://publications.europa.eu/webapi/{type}/
getTopConcepts?concept_scheme={cs-uri}&language={lang}
```

where:

– **{type}** can be either nal or eurovoc, depending on whether the user wants to retrieve the top concepts of a NAL or EUROVOC concept, respectively

– **{cs-uri}** is the resource URI of the NAL/EUROVOC concept scheme of which retrieving the top concepts

– **{lang}** is a 3-chars ISO_639-3 language code identifying the language the user wants to retrieve the top concepts with.

Here follows some examples of valid requests:

– **http://publications.europa.eu/webapi/nal/getTopConcepts?concept_scheme=http%3A%2F%2Fpublications.europa.eu%2Fresource%2Fauthority%2Ffd_330&language=eng**

– **http://publications.europa.eu/webapi/eurovoc/getTopConcepts?concept_scheme=http%3A%2F%2Feurovoc.europa.eu%2F100225&language=fra**

Response

The list of top concepts in JSON format.

Example:

```
[
  {
    "language": "fra",
    "identifier": "3928",
    "notations": [

    ],
    "uri": {
      "uri": "http://eurovoc.europa.eu/3928"
    },
    "prefLabel": {
      "language": "fra",
      "string": "sciences du comportement"
    },
    "altLabels": [
      "psychologie du comportement",
      "behaviorisme"
    ],
    "hiddenLabels": [
      "comportement, psychologie du",
      "comportement, sciences du"
    ]
  },
  {
    "language": "fra",
    "identifier": "3956",
    "notations": [

    ],
    "uri": {
      "uri": "http://eurovoc.europa.eu/3956"
    },
    "prefLabel": {
      "language": "fra",
      "string": "sciences sociales"
    },
    "altLabels": [
      "sciences humaines"
    ],
    "hiddenLabels": [
      "sociales, sciences",
      "humaines, sciences"
    ]
  },
  [...other concepts]
]
```

5.2.2.8. Retrieve the domains

Description

This service allows the user to retrieve the domains facets of the EUROVOC thesaurus.

Request

The user must fire a GET request to the following URL:

`http://publications.europa.eu/webapi/eurovoc/getDomains`

Response

The list of domains in JSON format.

Example:

```
[
  {
    "identifier": "28",
    "uri": {
      "uri": "http://eurovoc.europa.eu/100149"
    },
    "conceptSchemes": [
      {
        "uri": "http://eurovoc.europa.eu/100212"
      },
      {
        "uri": "http://eurovoc.europa.eu/100213"
      },
        [...other concept scheme URIs]
      ],
      "labels": [
        {
          "language": "ron",
          "string": "28 PROBLEME SOCIALE"
        },
        {
          "language": "hun",
          "string": "28 TÁRSADALMI KÉRDÉSEK"
        },
        [...other labels]
      ]
    },
    {
      "identifier": "24",
      "uri": {
        "uri": "http://eurovoc.europa.eu/100148"
      },
      "conceptSchemes": [
        {
          "uri": "http://eurovoc.europa.eu/100200"
        },
        {
          "uri": "http://eurovoc.europa.eu/100203"
        },
        [...other concept scheme URIs]
      ],
      "labels": [
        {
          "language": "hr",
          "string": "24 FINANCIJE"
        },
        {
          "language": "ron",
          "string": "24 FINANTE"
        },
        [...other labels]
      ]
    },
    [...other domains]
]
```

5.2.3 NOTIFICATIONS: RSS AND ATOM FEEDS

This notification service provides information about the ingesting of documents, the loading of NALs and the loading of ontologies in the form of an RSS or Atom feed. By accessing these feeds, it is possible to get a complete history of the performed actions.

Please note: as this feature is introduced in Cellar 6.2.0, it is yet not possible to retrieve information of actions executed before the installation of this Cellar release.

5.2.3.1. Request

To specify if the response is given in RSS or in Atom format, the HTTP header Accept:{accept-type} must be specified, where {accept-type} is a string which may assume the following values:

– application/rss+xml, in which case the Cellar will provide the results as an RSS feed

– application/atom+xml, in which case the Cellar will provide the results as an ATOM feed

If this header is not set, it defaults to application/rss+xml.

To request a feed, the following URL must be used:

```
http://[CELLAR_IP]:[CELLAR_PORT]/[CELLAR_CONTEXT]/webapi/
notification/{channel} ?{parameters}
```

The channels supported by the public feed are: ingestion, nal, or ontology. The next chapters describe the different notification types and their possible parameters.

5.2.3.1.1. Ingestion feed

Provides an overview of all ingestion actions, filtered by the given parameters.

– **startDate:** Defines the date (inclusive) since which the ingestion notifications shall be retrieved. Its format must match one of the following:
  - yyyy-MM-dd (2013-12-02)
  - yyyy-MM-dd'T'HH:mm:ss (2013-12-02T09:24:22)
  - yyyy-MM-dd'T'HH:mm:ssZZ (2013-12-02T09:24:22-01:00)
  - yyyy-MM-dd'T'HH:mm:ss.SSSZZ (2013-12-02T09:24:22.123-01:00).

– **endDate:** Defines the date (inclusive) until which the ingestion notifications shall be retrieved. It has the same format of startDate

– **type:** A string value, either CREATE, UPDATE or DELETE. It defines the type of ingestion to be retrieved

– **wemiClasses:** A comma-separated list of WEMI classes: work, expression, manifestation, item, dossier, event or agent

– **page:** The number of the page on the feed to display, should the total number of entries returned be higher than 1000 (defined in property cellar.service.notification.itemsPerPage). This parameter may be used to page large results by firing subsequent requests and setting incremental values on this parameter. If not set, page 1 is returned.

Note: only the startDate parameter is mandatory; if an optional parameter is not set, no filter is applied.

5.2.3.1.2. NAL feed

Provides an overview of all NAL loading actions, filtered by the given parameters.

- **startDate, endDate, page:** same as explained above.

5.2.3.1.3. Ontology feed

Provides an overview of all NAL loading actions, filtered by the given parameters.

- **startDate, endDate, page:** same as explained above.

5.2.3.1.4. Example requests

Retrieve: the 3rd page of the RSS feed containing the updates of works and events occurred from the 1st July to the 31st July 2016:

-
  ```
  curl -H 'Accept:application/rss+xml' "http://
  publications.europa.eu/webapi/notification/
  ingestion?startDate=2016-07-01&endDate=2016-07-31&type=UP
  DATE&wemiClasses=work,event&page=3"
  ```

Retrieve the 1st page of the ATOM feed containing the creations, updates and deletions of all types of entities occurred from the 1st July 2016 until now:

-
  ```
  curl -H 'Accept:application/atom+xml' "http://
  publications.europa.eu/webapi/notification/
  ingestion?startDate=2016-07-01"
  ```

Retrieve the 3rd page of the RSS feed containing the successful NAL updates occurred from the 1st July to the 31st July 2016:

-
  ```
  curl -H 'Accept:application/rss+xml' http://
  publications.europa.eu/webapi/notification/
  nal?startDate=2016-07-01&endDate=2016-07-31&page=3
  ```

Retrieve the 1st page of the ATOM feed containing the successful ontology updates occurred from the 1st July 2016 until now:

-
  ```
  curl -H 'Accept:application/atom+xml' http://
  publications.europa.eu/webapi/notification/
  ontology?startDate=2016-07-01
  ```

5.2.3.2. Response

The response contains the following common information (no matter what format or feed).

- **title:** The descriptive title of the feed.
- **startDate:** It contains the value of the as startDate parameter.
- **endDate:** It contains the value of the endDate parameter; it defaults to current date in case it is not provided, or set in the future.

- **page:** Cardinal number of the current page of the results.
- **moreEntries:** If true, the result has been paged and more entries that satisfy the request have been found. Subsequent requests should be fired with increasing page numbers.

The items (RSS) / entries (Atom) of the feed answer differ for each feed type. They are described below.

5.2.3.2.1. *Ingestion items*

- **guid (only for rss entries):** The unique id identifying the ingestion event.
- **notifEntry:id:** Same as above.
- **notifEntry:cellarId:** The cellar ID of the ingested element.
- **notifEntry:rootCellarId:** The cellar ID of the root element of the WEMI hierarchy containing the ingested element.
- **notifEntry:type:** The type of the ingestion action (create, update or delete).
- **notifEntry:priority:** The priority of the ingestion event (AUTHENTICOJ, DAILY or BULK).
- **notifEntry:classes:** The class hierarchy of the ingested element: the top class is the most specific, the bottom one the most general.
- **notifEntry:identifiers:** The sameases of the ingested element.
- **notifEntry:date:** The ingestion date and time.

5.2.3.2.2. *NAL items*

- **guid (only for rss entries):** The URI of the loaded NAL.
- **version:** The version (creation date) of the NAL.
- **date:** The date and time of the NAL loading.

5.2.3.2.3. *Ontology items*

- **guid (only for rss entries):** The URI of the loaded ontology.
- **version:** The version of the loaded ontology.
- **date:** The date and time of the ontology loading.

5.2.3.2.4. *Example response (RSS ingestion)*

```
<rss version="2.0"
  xmlns:notifReq="http://publications.europa.eu/rss/
notificationRequest">
  xmlns:notifEntry="http://publications.europa.eu/rss/
notificationEntry">
  <channel>
   <title>Ingestion Notification Messages Response</title>
    <notifReq:startDate>2012-01-01T00:00:00+01:00</
notifReq:startDate>
```

```
    <notifReq:endDate>2012-12-31T00:00:00+01:00</
notifReq:endDate>
    <notifReq:page>1</notifReq:page>
    <notifReq:moreEntries>false</notifReq:moreEntries>
    <item>
      <guid isPermaLink="false">7081775</guid>
      <notifEntry:id>7081775</notifEntry:id>
      <notifEntry:cellarId>cellar:ca753ae9-cf80-11e2-859e-
01aa75ed71a1</notifEntry:cellarId>
      <notifEntry:rootCellarId>cellar:ca753ae9-cf80-11e2-
859e-01aa75ed71a1</notifEntry:rootCellarId>
      <notifEntry:type>update</notifEntry:type>
      <notifEntry:priority>DAILY</notifEntry:priority>
      <notifEntry:classes>
       <notifEntry:class>http://publications.europa.eu/
ontology/cdm#case-law_national</notifEntry:class>
       <notifEntry:class>http://publications.europa.eu/
ontology/cdm#case-law</notifEntry:class>
       <notifEntry:class>http://publications.europa.eu/
ontology/cdm#resource_legal</notifEntry:class>
<notifEntry:class>http://publications.europa.eu/ontology/
cdm#work</notifEntry:class>
      </notifEntry:classes>
      <notifEntry:identifiers>
        <notifEntry:identifier>oj:JOL_2012_154_R_0012_01</
notifEntry:identifier>
       <notifEntry:identifier>celex:32006D0241</
notifEntry:identifier>
      </notifEntry:identifiers>
      <notifEntry:date>2012-06-11T09:13:58+01:00</
notifEntry:date>
    </item>
  </channel>
</rss>
```

6. Master data

6.1. Ontology: CDM

The Cellar's ontology is called common data model or CDM. It is based on the model of functional requirements for bibliographical records (FRBR) published by the International Federation of Library Associations [1]. Basic definitions of FRBR are:

Work: a distinct intellectual or artistic creation.

Expression: the realisation of a work in the form of alphanumeric, musical or choreographic notation, sound, image, object, movement, etc.

Manifestation: the physical embodiment of an expression of a work.

Item: a single exemplar of a manifestation.

An Official Journal has at least four related works:

- The full version, with an identifier like **`oj:JOA_1952_001_R`**
- Since 1 July 2013, the electronic signature of the OJ, with an identifier like **`oj:JOL_2014_001_R_SIG`**
- The table of contents or TOC, with an identifier like **`oj:JOA_195_001_R_TOC`**
- The version act by act, with at least one identifier like **`oj:JOA_1952_001_R_0003_01`**

Please note that the identifiers are given as examples; there is no mandatory naming convention to be followed, and therefore any identifier may have any random name. In order to find the signature, TOC and acts belonging to an OJ, you should either consult them in EUR-Lex, or use the CDM properties as shown in use case 4.2.

Expressions are linguistic versions in the Cellar. Please note that some documents, like some maps or posters from the EU Bookshop, may have more than one language. All languages used come from the corresponding authority table (see section 6.2).

Manifestations in the Cellar are the file types: for example, PDF or Formex. All formats come from the corresponding authority table (see section 6.2). There are even 'print' manifestations, which have no associated content, just metadata, to express that the OP keeps a printed version of that expression (language) and work.

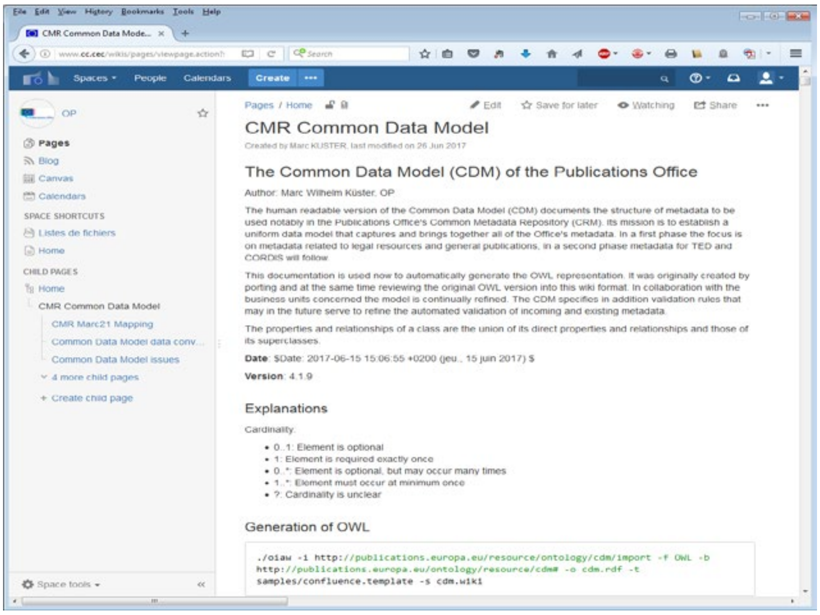
[1] https://www.ifla.org/publications/functional-requirements-for-bibliographic-records

Finally, items are the digital files. There may be more than one, like the OJs with many pages such as the EU budget, which is split into several files. Also metadata is stored in items (files), in RDF format.

The full list of CDM properties is expressed in OWL with RDF format (see section 2.3) and can be consulted on the Metadata Registry (MDR) page (http://publications.europa.eu/mdr/cdm/).

The documentation of the CDM is available on the same page, in wiki format, ready to be imported to a Mediawiki [1] or compatible software installation. In Figure 10, a screen capture of the internal wiki is shown. In case you want a print out of that page in PDF, please contact us (section 1.3).

Figure 10: Screen capture of the CDM internal wiki



CMR Common Data Model

Created by Marc KUSTER, last modified on 26 Jun 2017

The Common Data Model (CDM) of the Publications Office

Author: Marc Wilhelm Küster, OP

The human readable version of the Common Data Model (CDM) documents the structure of metadata to be used notably in the Publications Office's Common Metadata Repository (CRM). Its mission is to establish a uniform data model that captures and brings together all of the Office's metadata. In a first phase the focus is on metadata related to legal resources and general publications, in a second phase metadata for TED and CORDIS will follow.

This documentation is used now to automatically generate the OWL representation. It was originally created by porting and at the same time reviewing the original OWL version into this wiki format. In collaboration with the business units concerned the model is continually refined. The CDM specifies in addition validation rules that may in the future serve to refine the automated validation of incoming and existing metadata.

The properties and relationships of a class are the union of its direct properties and relationships and those of its superclasses.

Date: $Date: 2017-06-15 15:06:55 +0200 (jeu., 15 juin 2017) $

Version: 4.1.9

Explanations

Cardinality:

- 0..1: Element is optional
- 1: Element is required exactly once
- 0..*: Element is optional, but may occur many times
- 1..*: Element must occur at minimum once
- ?: Cardinality is unclear

Generation of OWL

```
./oiaw -i http://publications.europa.eu/resource/ontology/cdm/import -f OWL -b
http://publications.europa.eu/ontology/resource/cdm# -o cdm.rdf -t
samples/confluence.template -s cdm.wiki
```

6.2. Authority tables, Eurovoc

In order to harmonise and standardise the codes and the associated labels used in the Publications Office, a number of named authority lists (NALs) have been defined. These NALs are also known as controlled vocabularies or value lists.

Many metadata in the Cellar are standardised with these NALs, which can be found on the MDR website (http://publications.europa.eu/mdr/authority/) in several formats, including SKOS.

(1) https://www.mediawiki.org/wiki/MediaWiki

Eurovoc is a multilingual, multidisciplinary thesaurus covering the activities of the EU, the European Parliament in particular. Currently, it contains terms in 23 EU languages (Bulgarian, Spanish, Czech, Danish, German, Estonian, Greek, English, French, Croatian, Italian, Latvian, Lithuanian, Hungarian, Maltese, Dutch, Polish, Portuguese, Romanian, Slovak, Slovenian, Finnish and Swedish), plus Serbian.

Eurovoc can also be downloaded to be reused from the MDR website (http://publications.europa.eu/mdr/eurovoc/), and it has its own website (http://eurovoc.europa.eu/).

6.3. Formex

Formex describes a file format created by the Publications Office for the exchange of data with its contractors. In particular, it defines the logical markup for documents which are published in the different series of the *Official Journal of the European Union*.

Since Formex v4 is based on XML, it can be reused. Some collections have manifestations in Formex, some examples are shown in chapter 5: look for URIs ending in .fmx4.

All the information about Formex, its schema and examples can be found on the website (http://formex.publications.europa.eu).

PART III: Annexes

ANNEX I — Acronyms

CDM	common data model
Cellar	digital dissemination repository of the Publications Office
CJ	Court of Justice of the European Union
CURIE	compact URI
cURL	client URL request library
ECLI	European case-law identifier
EU	European Union
FRBR	functional requirements for bibliographical records
FTP	file transfer protocol
HTML	hypertext markup language
HTTP	hypertext transfer protocol
ISO	International Organisation for Standardisation
JSON	JavaScript object notation
MDR	MetaData Registry
NAL	named authority list
OJ	*Official Journal of the European Union*
OP	Publications Office of the European Union
OWL	ontology web language
PDF	portable document format
RDF	resource description framework
RDFS	resource description framework schema
RESTful	representational state transfer set of web services
RSS	rich (or RDF) site summary
SFTP	secure file transfer protocol
SKOS	simple knowlegde organisation system
Sparql	(recursive) Sparql protocol and RDF query language
SQL	structured query language
TB	terabyte, worth 1024 gigabytes or 1024*1024 megabytes
TOC	table of contents
URI	uniform resource identifier
URL	uniform resource locator
UUID	Universally Unique Identifier: identifier standard used in software construction, standardized by the Open Software Foundation (OSF)
W3C	World Wide Web Consortium
WEMI	Work, Expression, Manifestation and Item
XML	extended markup language

ANNEX II — List of tables

ANNEX III — List of figures

ANNEX IV — List of ISO_639-3 codes of supported European languages

The Cellar supports the European languages identified by the following ISO_639-3 codes:

Table 19: Supported European languages with their ISO_639-3 codes

ISO_639-3 code	Language
bul	Bulgarian
ces	Czech
dan	Danish
deu	German
ell	Modern Greek
eng	English
est	Estonian
fin	Finnish
fra	French
gle	Irish
hrv	Croatian
hun	Hungarian
isl	Icelandic
ita	Italian
lav	Latvian
lit	Lithuanian
mlt	Maltese
nld	Dutch
nor	Norwegian
pol	Polish
por	Portuguese
ron	Romanian, Moldavian, Moldovan
slk	Slovak
slv	Slovene
spa	Spanish, Castillian
swe	Swedish

ANNEX V — cURL

cURL (Client URL Request Library) is a computer software providing command-line tool for transferring data using various protocols, the most important of which, for our purposes, is HTTP/HTTPS.

The present document uses cURL for depicting all the examples of HTTP requests: cURL is preferable to in-browser or other graphical tools, as:

1. it is independent from the OS
2. the way a browser allows the user to build the HTTP requests may differ from browser to browser
3. its syntax does not depend on the version used, while the browser may change during time the way it represents the HTTP request
4. its syntax is simple and direct to the goal.

Basic use of cURL involves simply typing **`curl`** at the command line, followed by the URL of the output to retrieve. For example, to retrieve the example.com homepage, type:

```
curl http://www.example.com
```

For specifying an HTTP request header it is enough to type:

```
curl -H 'myHeaderName:myHeaderValue' http://www.example.com
```

where **`myHeaderName`** is the name of the header and **`myHeaderValue`** is its value.

This is enough for our purposes: for more information, please refer to cURL home page at http://curl.haxx.se/.

ANNEX VI — JSON

JSON (JavaScript Object Notation) is a lightweight data-interchange format.

It has several advantages:

1. it is easy for humans to read and write
2. it is easy for machines to parse and generate
3. it is based on a subset of the JavaScript Programming Language, used worldwide
4. it is a text format that is completely language independent, but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others.

These properties make JSON an ideal data-interchange language.

JSON's basic types are:

– **Number** (double precision floating-point format in JavaScript, generally depends on implementation)
– **String** (double-quoted Unicode, with backslash escaping)
– **Boolean** (true or false)
– **Array** (an ordered sequence of values, comma-separated and enclosed in square brackets; the values do not need to be of the same type)
– **Object** (an unordered collection of key:value pairs with the ':' character separating the key and the value, comma-separated and enclosed in curly braces; the keys must be strings and should be distinct from each other)
– **null** (empty)

Non-significant white space may be added freely around the "structural characters" (i.e. the brackets "[{]}", colon ":" and comma ",").

The following example shows the JSON representation of an object that describes a person. The object has string fields for first name and last name, a number field for age, contains an object representing the person's address, and contains a list (an array) of phone number objects.

```
{
    "firstName": "John",
    "lastName": "Smith",
    "age": 25,
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": "10021"
    },
    "phoneNumber": [
        {
            "type": "home",
            "number": "212 555-1234"
        },
        {
            "type": "fax",
            "number": "646 555-4567"
        }
    ]
}
```

ANNEX VII — OWL

The Common Data Model is expressed formally as an ontology – a set of concepts within a domain, and the relationships among those concepts – according to a format called the Web Ontology Language (OWL). The ontology formally defines the various classes and properties and assigns unique URIs to them that reside under the URI:

http://publications.europa.eu/ontology/cdm

The ontology also defines certain inferred behaviours for classes and properties. For example, being a member of a subclass, e.g. a directive, implies being a member also of its superclasses, e.g. secondary legislation and resource legal. Also, if act A repeals another act B it is possible to infer that B is repealed by A. Inferred classes and properties are also exposed by the Cellar alongside explicitly provided ones.

ANNEX VIII — Citation and licence

If you need to cite any document present in the Cellar, we recommend that you follow the rules from the Interinstitutional style guide (http://publications.europa.eu/code/en/en-250900.htm).

Cellar content can be used in accordance with the conditions laid out in <https://publications.europa.eu/en/web/about-us/legal-notices/eu-law-and-publications-website#copyright>. As to the *Who is who* dataset, please check specific rules in <http://europa.eu/whoiswho/public/index.cfm?fuseaction=idea.show_page&pagename=legal_notice>.

OA-01-17-789-EN-N

Publications Office

ISBN 978-92-78-41542-6