CELLAR - End user manual

EU law and publications

Reference	CEM-EUM-8.10.1
Status	SFR
Release Date	08 Dec 2023
Application Name	CELLAR
Target Release	8.10.1

- Abbreviations
  - Acronyms
  - Definitions
- Introduction
  - Purpose of the Document
  - Intended Audience
  - Structure of the Document
- Main concepts
  - Functional Requirements for Bibliographic Records (FRBR)
    - FRBR in CELLAR's context
      - Hierarchy work-expression-manifestation-content stream
      - Hierarchy dossier-event
      - Hierarchy event
      - Hierarchy agent
  - Types of notices
    - Tree notice
    - Branch notice
    - Object notice
    - Identifier notice
    - RDF-Object notice
    - RDF-Tree notice
  - Content streams
  - NALs
  - EUROVOC
  - Resource URI
    - {ps-name}
    - {ps-id}
      - If {ps-name} is 'cellar'
      - If {ps-name} is other than 'cellar'
    - Examples of valid resource URIs
    - CURIE format of a resource URI
- Available services
  - WEMI services
    - Retrieve the tree notice
      - Description
      - Request
      - Response
    - Retrieve the branch notice
      - Description
      - Request
      - Response
    - Retrieve the object-work notice
      - Description
      - Request
      - Response
    - Retrieve the object-expression notice
      - Description
      - Request
      - Response
    - Retrieve the object-manifestation notice

- Description
- Request
- Response

- Retrieve the identifier notice
  - Description
  - Request
  - The response is an XML-formatted notice containing the URI of the cellar ID and its synonym(s).
- Retrieve the RDF/XML formatted metadata for a given resource
  - Description
  - Request
- Retrieve the RDF/XML formatted metadata of the tree whose root is a given resource
  - Description
  - Request
  - Response
- Retrieve content streams
  - 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
  - Response
- Retrieve a compressed content stream's entry
  - Description
  - Request
  - Response
- Retrieve content stream collections
  - Description
  - Request
  - Response

- NAL/EUROVOC services
  - Retrieve a dump
    - Description
    - REST endpoint /nal/list
      - Request
      - Response
    - REST endpoint /nal/get
      - Request
      - Response
    - REST endpoint /resource
      - Request
      - Response
  - Retrieve the supported languages
    - Description
    - Request
    - Response
  - Retrieve a concept scheme
    - Description
    - Request
    - Response
  - Retrieve the concept schemes
    - Description
    - Request
    - Response
  - Retrieve a concept
    - Description
    - Request
    - Response
  - Retrieve the concept relatives
    - Description
    - Request
    - Response
  - Retrieve the top concepts
    - Description
    - Request
    - Response
  - Retrieve the domains
    - Description
    - Request
    - Response

- Notifications: RSS and Atom feeds
  - Request
    - Ingestion channel
    - NAL channel
    - Ontology channel
    - SPARQL-load channel
    - Example requests
  - Response
    - Ingestion channel's items
    - NAL channel's items
    - Ontology channel's items
    - SPARQL-load channel's items

                - Example response
        - SPARQL
        - Ingestion Service
            - Mets upload endpoint
                - Request
                - Response
                - Authentication
                - Authorization
        - Status Endpoints
            - Package-level status endpoint
                - JSON response
            - Object-level status endpoint
                - JSON response
            - Status endpoint lookup
                - JSON response
            - Authentication
            - Authorization
- Annexes
    - Annex 1: List of ISO_639-3 codes of supported European languages
    - Annex 2: cURL
    - Annex 3: JSON
    - Annex 4: OWL
    - Annex 5 : List of available mime types
    -

Abbreviations

Acronyms

Acronym	Meaning
CURIE	Compact URI
cURL	Client URL Request Library
FRBR	Functional Requirements for Bibliographic Records
JSON	JavaScript Object Notation
NAL	Named Authority List
OWL	Web Ontology Language
RDF	Resource Description Framework
SKOS	Simple Knowledge Organization System
SPARQL	SPARQL Protocol and RDF Query Language
URI	Uniform Resource Identifier
UUID	Universally Unique Identifier
WEMI	Work, Expression, Manifestation and Item
XML	Extensible Markup Language

Definitions

Term	Meaning
ISO_639-3	Codes for the representation of names of languages

Introduction

Purpose of the Document

The purpose of this document is to provide the CELLAR end-user with a structured, non-technical, easy-to-read user manual.

Intended Audience

This document is intended for all the CELLAR end-users.

Structure of the Document

The document is organized as follows:

- The present introduction;
- A presentation of the main concepts on which the CELLAR is built upon;
- A full description of CELLAR's available services including some usage scenarios.

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)
- Types of notices
- Content streams
- NALs
- EUROVOC
- Resource URI.

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 writes 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*.

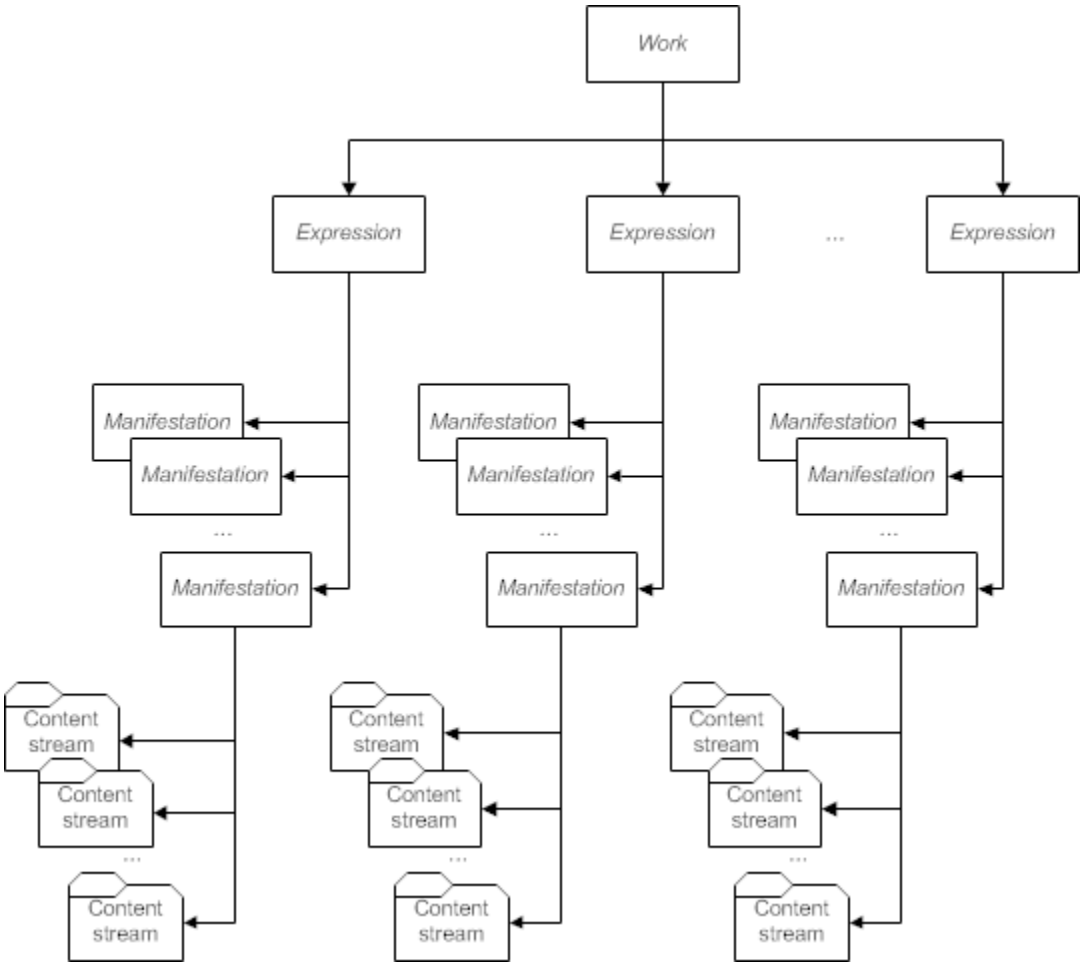
FRBR in CELLAR's context

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.

Hierarchy work-expression-manifestation-content stream

The work-expression-manifestation-content stream hierarchy 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.



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

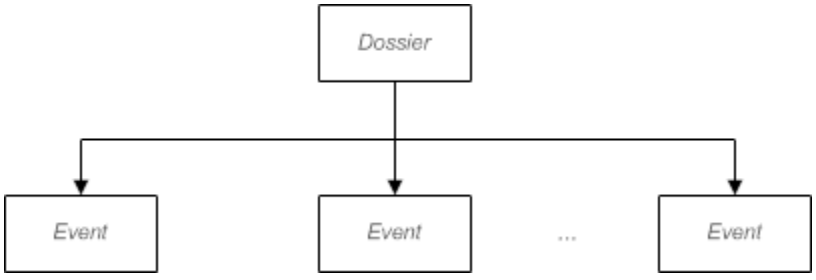
- Legislative data, currently published primarily in the EUR-Lex portal
- General publications, currently published in EU Bookshop
- Tender documents and related works (OJ-S), currently published in TeD portal
- Research documents, currently published in the CORDIS portal

The WEM model is applied consistently throughout for works from all domains. However, the abstract classes such as work are then concretized for the various domains in the Cellar's Common Data Model (CDM) by *subclassing* these abstract classes.

Hierarchy dossier-event

The dossier-event hierarchy 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.



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.

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".
These realizations of the WEMI pattern are the basis of the CELLAR's definition and data layer, that is, its *ontology*.

Hierarchy agent

The agent hierarchy is solely composed by an agent, which covers the W role of the WEMI pattern.
These realizations of the WEMI pattern are the basis of the CELLAR's definition and data layer, that is, its *ontology*.

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.

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

Branch notice

A *Branch notice* is a subset of the Tree notice which represents 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.

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

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 this, this and this paragraph.

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 this paragraph.

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

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 this paragraph.

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 this paragraph.

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 this paragraph).

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 this paragraph.

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.

EUROVOC

Eurovoc is a special type of NAL which represents the multilingual thesaurus maintained by the Publications Office of the European Union.

It is defined at:

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

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

- the European Parliament
- the Office for Official Publications 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*.

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*.

In the following paragraphs you can find the description of each part of the resource URI, with some examples depicted in this paragraph.

Finally, paragraph CURIE format of a resource URI describes the *CURIE* format.

{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, transjai, agent, uriserv, join, swd, comnat,mdr, legissum, ecli, procedure, procedure-event, eli, immc* and *planjo*.

{ps-id}

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

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:

Type	{ps-id}	Example

work dossier event agent	{work-id}	550e8400-e29b-41d4-a716-446655440000
expression	{work-id}.{expr-id}	550e8400-e29b-41d4-a716-446655440000.0001
manifestation	{work-id}.{expr-id}.{man-id}	550e8400-e29b-41d4-a716-446655440000.0001.03
content stream	{work-id}.{expr-id}.{man-id}/{cs-id}	550e8400-e29b-41d4-a716-446655440000.0001.03/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.

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

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

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 this paragraph.
- {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.

Examples of valid resource URIs

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

- 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
```

- 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
```

- 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
```

- 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
```

- 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
```

- 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
```

- 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
```

- 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
```

- 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
```

- 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
```

- 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
```

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 this paragraph).

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
- retrieve the branch notice of a work

- retrieve the object notice of an object (work, expression or manifestation)
- retrieve all the identifiers of a specific document (synonyms)
- retrieve the RDF/XML formatted metadata for a given resource
- retrieve the RDF/XML formatted metadata for the tree whose root is a given resource
- retrieve the content streams of a work given a specific language and format**.**

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

- retrieve a dump
- retrieve the supported languages
- retrieve a concept scheme
- retrieve the concept schemes
- retrieve a concept
- retrieve the concept relatives
- retrieve the top concepts
- retrieve the domains.

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.

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 this paragraph).

The dissemination service uses a global negotiation system that returns always a "303 - See other" response. In order to use the redirect feature, the client must enable the *follow-redirect* option.

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}
```

where:

- {ps-name} is a valid production system name, as described here
- {ps-id} is a valid production system id identifying a work and compatible with its {ps-name}, as described here
- {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.

The following HTTP header must be set on the request:

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

Here follows some examples of valid requests using *cURL* (for a brief description about what cURL is and how to use it, please refer to this annex):

```
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>
```

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}
```

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.

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>
```

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}
```

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

The following HTTP header 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>
```

Retrieve the object-expression notice

Description

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}
```

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.

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>
```

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}
```

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.

The following HTTP header must be set on the request:

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

The following HTTP header can be set on the request:

- *Negotiate:vlist*
  In this case, 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>
```

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
```

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

```
<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>
```

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}
- {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.

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=normalized*
  In this case, the resulting RDF notice will contain the direct and inferred triples and all the production identifiers will be normalized to their corresponding cellar identifiers (if they exist).

- *Accept:application/rdf+xml;notice=non-inferred*
  In this case, the inferred triples will be excluded from the resulting RDF notice.

- *Accept:application/rdf+xml:notice=non-inferred-normalized*
  In this case, the inferred triples will be excluded from the resulting RDF notice and all the production identifiers will be normalized to their corresponding cellar identifiers (if they exist).

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

- *Negotiate:vlist*
  In this case, 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.

Here follows some examples of valid requests 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
```

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>
```

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}
- {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.

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=tree-normalized*
  In this case, the resulting RDF notice will contain the direct and inferred triples and all the production identifiers will be normalized to their corresponding cellar identifiers (if they exist).
- *Accept:application/rdf+xml;notice=non-inferred-tree*
  In this case, the inferred triples will be excluded from the resulting RDF notice.
- *Accept:application/rdf+xml;notice=non-inferred-tree-normalized*
  In this case, the inferred triples will be excluded from the resulting RDF notice and all the production identifiers will be normalized to their corresponding cellar identifiers (if they exist).

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>
```

Retrieve content streams

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}
- {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.

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 identifies the format of the content stream to return: the list of available mime types can be found in Annex 5**Note:** Specifically for the retrieval of RDF/XML-formatted content streams, the following value must be set: *Accept:application/rdf+xml;notice=content*

- *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 = 2^{63}-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 (an XHTML from the French branch), 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 response is represented by the associated content stream.

Retrieve a compressed content stream's entry

Description

In the case a content stream is compressed (that is equal to say that it has mimetype application/zip), this service allows the user to retrieve a specific entry of the content stream.

Request

This type or retrieval is only possible via a negotiated URL, in the form:

```
http://publications.europa.eu/resource/cellar/{man-id}/DOC_1/{entry-name}
```

where {man-id} is the id of the manifestation hosting the content stream and {entry-name} is the name of the entry to retrieve. [1] Here follows an example of a valid request that retrieves an entry from a content stream, using cURL:

```
curl -v "http://publications.europa.eu/resource/cellar/xyz.0003.01/DOC_1/abc.xml"
```

Response

The associated entry of the content stream returns a 200 OK response, or a 404 response if either the content stream or the entry does not exist.

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}
- {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.

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 identifies 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 an example of valid request that retrieves a ZIP including all FMX4 content streams of the French branch, 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
```

Here is instead an example showing how to retrieve a ZIP including all FMX4 content streams of the English branch, and by using the OJ production identifier:

```
curl -H 'Accept:application/zip;mtype=fmx4' -H 'Accept-Language:eng' "http://publications.europa.eu/resource/oj
/JOL_2005_343_R" -L
```

Finally, here's how to retrieve the list of all FMX4 content streams of the French branch:

```
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 response represents the associated content streams; its format depends on the value set for *Accept* request header:

- *Accept:application/**zip**;mtype=...*
  In this case, a zip file containing all content stream files of the requested manifestation is returned. If the manifestation hosts 1 single content stream with mimetype application/zip, then the stream is not zipped again, but instead it is returned as is

- *Accept:application/**list**;mtype=...*
  In this case, an html list of all content stream file names of the requested manifestation is returned. If one or more content streams have mimetype application/zip, the entries of the compressed streams are also listed.
  Here follows an example of valid response:

```
List of content streams with URIs:
QC30110960AD_0.zip --> http://publications.europa.eu /resource/cellar/xyz.0001.01/DOC_1
QC30110960AD.jpg --> http://publications.europa.eu/resource/cellar/ xyz.0001.01/DOC_1/QC30110960AD.jpg
QC30110960AD_001.pdf --> http://publications.europa.eu /resource/cellar/xyz.0001.01/DOC_1/QC30110960AD_001.pdf
QC30110960AD_002.pdf --> http://publications.europa.eu /resource/cellar/xyz.0001.01/DOC_1/QC30110960AD_002.pdf
QC30110960AD_001.pdf --> http://publications.europa.eu /resource/cellar/xyz.0001.01/DOC_2
```


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.

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 than 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.

Retrieve a dump

Description

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
- /resource which retrieves the NAL from Fedora using the production identifier that matches an item of the WEMI hierarchy in Fedora when the NAL was ingested using the model load behaviour.

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 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]
```

REST endpoint /nal/get

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

Response

The response is the NAL content stream in RDF/XML format.

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

REST endpoint /resource

Request

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

```
http://publications.europa.eu/resource/{item_production_identifier}
```

where {item_production_identifier} can be retrieved:

- either from the table CELLAROWNER.PRODUCTION_IDENTIFIER. The production identifier must be pointing to an ITEM in a WEMI hierarchy. For example:
  - *distribution:access-right/skos_core_access-right-skos.rdf*
  - *distribution:access-right/skos_ap_act_access-right-skos-ap-act.rdf*
- either by looking into the initial package of the NAL where the production identifier is defined. For example:
  - *CONTENTIDS="distribution:corporate-body/skos_ap_act_corporatebodies-skos-ap-act.rdf "/>*

Please note that this ID must be adapted. The ':' must be replaced by '/' to be able to successfully retrieve the NAL using this endpoint.

Here follows an example of valid request, assuming that the production identifier is defined as *distribution:access-right/skos_core_access-right-skos.rdf*:

- http://publications.europa.eu/resource/distribution/access-right/skos_core_access-right-skos.rdf

Response

The content stream in Fedora matching the production identifier given to the /resource endpoint is returned.

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 follow 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.

For example:

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

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 equal to *nal*): if not specified for EUROVOCs ({type} is equal to *eurovoc*), it defaults to http://eurovoc.europa.eu/100141.

Here follow 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

Response

The concept scheme in JSON format.

For example:

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

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 follow 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.

For example:

```
[
  {
    "date": null,
    "lastModified": null,
    "version": null,
    "uri": {
      "uri": "http://eurovoc.europa.eu/100225"
    },
    "labels": [
      {
        "language": "ron",
        "string": "3611 tiine 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]
]
```

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 follow 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.

For 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"
    ]
  }
]
```

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 follow 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.

For 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]
]
```

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.

For 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]
]
```

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.

For 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]
]
```

Notifications: RSS and Atom feeds

This notification service provides information about the ingestion and embargoing 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.

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

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*.

The following URL must be used to call the public feed:

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

The public feed supports the following channels:

- *ingestion* ({channel} is set with value *ingestion*)
- *nal*
- *ontology*
- sparql-load

each of which supports a specific set of {parameters}.

Ingestion channel

This channel provides an overview of all ingestion actions, filtered by the following parameters:

- *startDate*: Defines the date (inclusive) since which the ingestion notifications shall be retrieved. Its format must match one of these 4 standards:
  - 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.
- relatives: Defines if only the direct changed objects or only the related objects to the real changed objects are included in the response. if “relatives=false” only the direct changed objects are included in the response (DB entries with "relatives = false" or relatives = null”). If “relatives=true” only the related objects to the real changed objects are included in the reponse (DB entries with "relatives = true"). If this request parameter is not set in the request, this DB value is ignored and all objects are listed in the response.

Only the *startDate* parameter is mandatory; if an optional parameter is not set, no filter is applied.

NAL channel

This channel provides an overview of all NAL loading actions, filtered by the following parameters:

- *startDate*, *endDate*, *page*

which have the same meaning as for the ingestion channel.

Ontology channel

This channel provides an overview of all NAL loading actions, filtered by the following parameters:

- *startDate*, *endDate*, *page*

which have the same meaning as for the ingestion channel.

SPARQL-load channel

This channel provides an overview of all NAL loading actions, filtered by the following parameters:

- *startDate*, *endDate*, *page*

which have the same meaning as for the ingestion channel.

Example requests

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

retrieves the 3rd page of of the RSS feed containing the updates of works and events occurred from 1st July 2016 until 31st July 2016.

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

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

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

retrieves the 3rd page of the RSS feed containing the successful NAL updates occurred from 1st July 2016 until 31st July 2016.

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

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

Response

The response contains the following common information, regardless the format, feed or channel used:

- *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 of the feed differ depending on the channel used, as described in the following paragraphs.

Ingestion channel's 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.

NAL channel's 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.

Ontology channel's 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.

SPARQL-load channel's items

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

Example response

Here follows an example of response to a request fetching the ingestion channel in RSS format:

```
<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>
```

SPARQL

The Publications Office uses RDF as standard for its metadata, which can be queried with the *SPARQL Protocol and RDF Query Language* (*SPARQL*), via a SPARQL endpoint.

A complete syntax and semantics specification of the SPARQL can be found at:

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

while Publications Office's production SPARQL endpoint is available at:

- http://publications.europa.eu/en/linked-data

Ingestion Service

The *Ingestion Service* provides an HTTP endpoint allowing to upload Mets packages that will be ingested.

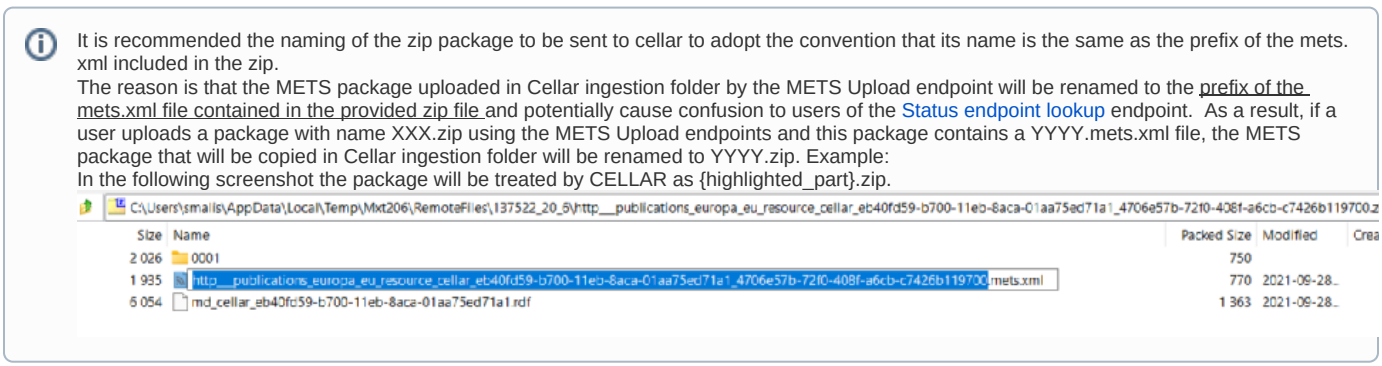
Mets upload endpoint

Request

The METS Upload endpoint is accessible via POST request from the “http://{CELLAR_ENV_URL}/admin/v1/secapi/mets” location and by using the following query parameters:

- **priority**: This optional parameter defines the Cellar reception folder where the packages are uploaded. The default value is “bulk”, in case this parameter is not defined. It could contain one of the following values:
  - authenticoj: reception-folder-authentic
  - daily: reception-folder
  - bulk: reception-folder-bulk
  - bulk_lowpriority: reception-folder-bulk-lowPriority
- **callback**: This optional parameter enables clients to define a callback address as part of the METS upload request. It is optional. If a callback address is specified, Cellar performs an HTTP POST to the callback endpoint as soon as the ingestion of the input package has either failed or succeeded.

The METS package is provided in the request body of the POST request.



It is recommended the naming of the zip package to be sent to cellar to adopt the convention that its name is the same as the prefix of the mets.xml included in the zip.
The reason is that the METS package uploaded in Cellar ingestion folder by the METS Upload endpoint will be renamed to the prefix of the mets.xml file contained in the provided zip file and potentially cause confusion to users of the Status endpoint lookup endpoint. As a result, if a user uploads a package with name XXX.zip using the METS Upload endpoints and this package contains a YYYY.mets.xml file, the METS package that will be copied in Cellar ingestion folder will be renamed to YYYY.zip. Example:
In the following screenshot the package will be treated by CELLAR as {highlighted_part}.zip.

Response

The success JSON response of the METS Upload endpoint contains the package-level Status endpoint together with the received date of the package. A JSON response example follows:

```
{
  "status_endpoint":"http://<HOST:PORT>/admin/v1/secapi/status/mets?id=062ef66c-b771-11ed-afa1-0242ac120002",
  "received" : "2022-04-22T17:24:42.123Z"
}
```

Authentication

The Mets Upload endpoint is a protected REST API using EU Login authentication and authorization. You may find below more details on EU Login authentication per different type of actor (CELLAR user or M2M/job).

CELLAR user: METS Upload functionality is provided in the form of a new panel of the Cellar Admin UI containing a simple upload form, through which a logged in user with proper privileges is able to select the local METS to upload. As this is a browser-based approach, it allows the user to be redirected to the graphical log-in form of the EU-Login server. For more information check CELLAR - Cellar Admin UI description#METSUpload.

As an alternative the specific METS Upload endpoint can also be triggered via using cURL on condition that the user has already performed a login on Cellar UI so that a JSESSIONID cookie is created; This cookie may then be appended to the curl request for as long the user session is valid. In detail, the steps of this workaround are the following:

- User logs in to the CELLAR Admin UI (any page) using a web browser after a successful EU-Login authentication is taken place.
- User opens the browser's cookie panel and copies the value of CELLAR's session cookie having the key JSESSIONID.
- User triggers the METS Upload endpoint via cURL while supplying the CELLAR session cookie, in order to use the already authenticated session.

Note: Although this methodology enables the usage of the endpoint via cURL it is considered as an unofficial approach; additionally, in case the session expires, the user would have to repeat all steps in order to acquire a fresh session cookie. Considering the above, this is not recommended for production environment.

Example

```
curl -X POST -H "Content-type: multipart/form-data" "http://<HOST>:<PORT>/admin/v1/secapi/mets" -F "fileData=@
/path/to/package.zip" -F "priority=BULK" -F " callback=testURI" -b "JSESSIONID=B12AA1BBEE27C9314B192826EE894938"
```

The filesystem path of the Mets package must be adapted.

M2M/job: External systems can also access the Mets Upload endpoint via machine to machine communication. Due to the fact that Cellar is secured via EU Login CAS protocol this m2m communication should be performed via using Cas Proxy Tickets. Technical details on supporting this approach are provided in CELLAR - Operation Manual#Machinetomachinecommunication.

Authorization

The Mets Upload endpoint is also authorized using the existing Cellar authorization mechanism. More specifically, any actor attempting to access the METS Upload endpoint should be assigned with the ingestion role. Find below more information on the authorization for each type of actor:

- **CELLAR users** assigned with the ingestion role, via the Security panel of the Cellar Admin UI, are authorized to use the METS Upload endpoint.
- **M2M/job**, are authorized to use the METS Upload endpoint similarly to the users' scenario previously mentioned, with the additional prerequisite that a CELLAR user with the same name as the EU-Login Job/certificate account needs to be created beforehand.

Status Endpoints

Package-level status endpoint

The package-level endpoint provides information on the status of a given METS package. The endpoint's resource path is "admin/v1/secapi/status/mets" and the query parameters are:

1. **id**: a mandatory string query parameter uniquely identifying the processing of a specific METS package. This value is stored in the PACKAGE_HISTORY.PACKAGE_UUID column per ingested METS package.
2. **ingestion**: an optional boolean query parameter specifying whether to include ingestion-specific status information in the response or not. In case it is not specified, the default value is false.
3. **indexation**: an optional boolean query parameter specifying whether to include indexation-specific status information in the response or not. In case it is not specified, the default value is false.
4. **verbosity**: an optional numeric parameter specifying the level of verbosity of a response. Value "1" is corresponding to a coarse level of granularity and "2" to a detailed level of information. In case it is not specified, the default value is 1.

For a package-level request, the response should include the information relevant to the requested package. As a result, if the object is part of a multi-structmap METS package, the "package-level" response should contain information for all contained structmaps.

JSON response

Please find below a table containing a description and other information for the JSON response attributes.

JSON response attribute	Description	Mandatory	Verbosity 1	Verbosity 2
timestamp	The system timestamp when the response is created.	YES	YES	YES
id	A unique ingestion identifier per package.	YES	YES	YES
input	The name of the METS package.	YES	YES	YES
received	A timestamp presenting when the package was moved to the reception folder. The ACCESS time of the UNIX system dates assigned to a file, will be used. The same file date is used also during the ingestion prioritization phase.	YES	YES	YES
request_params.ingestion	The value of this attribute contains a copy of the request's parameter "ingestion".	YES	YES	YES
request_params.indexation	The value of this attribute contains a copy of the request's parameter "indexation".	YES	YES	YES
request_params.verbosity	The value of this attribute contains a copy of the request's parameter "verbosity".	YES	YES	YES
ingestion.status	The value of this attribute contains a calculated value, considering the attributes "ingestion.pre_processing. status", "ingestion.structmaps.structmap[1].status", …, "ingestion.structmaps.structmap[n].status". Valid values are "NOT AVAILABLE", "WAITING", "RUNNING", "SUCCESS", "FAILED". Check Annex 3 of the Technical Analysis document D.CSRA Cellar Status Rest APIs - Technical Analysis. docx for more information on the calculation of the general ingestion status.		YES	YES
ingestion.pre_processing.status	The value of this attribute contains the package-related pre-processing status, before structmaps processing is launched. Valid values are "NOT AVAILABLE", "WAITING", "RUNNING", "SUCCESS", "FAILED". Check Annex 2 of the Technical Analysis document D.CSRA Cellar Status Rest APIs - Technical Analysis. docx for more information on the pre_process package status definition.	YES*	YES	YES
ingestion.pre_processing.logs	The value of this attribute contains a list of items. Each item is referring to an entry of the AUDIT_TRAIL_EVENT table. These entries are filtered using the AUDIT_TRAIL_EVENT.PACKAGE_ID column. The information included in each item is specified in the PROCESS, ACTION, TYPE, EVENT_DATE, MESSAGE columns of the database entry.	NO	NO	YES
ingestion.structmaps	The value of this attribute contains a list of items. Each item is presenting information related to the ingestion of a structmap. All structmaps are part of the same package.	YES*	YES	YES
ingestion.structmaps.structmap[n]. cellarId	The value of this attribute contains the cellar identifier of the top level entity of this structmap. It is presented only if the resource exists.	NO	YES	YES
ingestion.structmaps.structmap[n].psis	The value of this attribute contains a list of items. Each item is referring to a production identifier of the structmap.	YES*	YES	YES
ingestion.structmaps.structmap[n]. structMapID	The value of this attribute contains the name of the structmap as specified in the "structMap ID" field of the mets.xml file contained in the METS package.	YES*	YES	YES
ingestion.structmaps.structmap[n]. status	The value of this attribute contains the status of the structmap ingestion process. Valid values are "NOT AVAILABLE", "WAITING", "RUNNING", "SUCCESS", "FAILED"	YES*	YES	YES

	Check Annex 2 of the Technical Analysis document D.CSRA Cellar Status Rest APIs - Technical Analysis.docx for more information on the structmap ingestion status definition.			
ingestion.structmaps.structmap[n].logs	The value of this attribute contains a list of items. Each item is referring to an entry of the AUDIT_TRAIL_EVENT table. These entries are filtered using the AUDIT_TRAIL_EVENT.STRUCTMAP_ID column. The information included in each item is specified in the PROCESS, ACTION, TYPE, EVENT_DATE, MESSAGE columns of the database entry.	NO	NO	YES
ingestion.structmaps.structmap[n].shacl	The value of this attribute contains the SHACL validation report of the structmap. The SHACL report's content is defined by a configuration parameter specifying if the report should contain: 1. Errors and Warnings (WARNINGS) 2. Only Errors 3. No report	NO	NO	YES
indexation.status	The value of this attribute contains a calculated value, considering the attributes "indexation.structmaps.structmap[1].status", …, "indexation.structmaps.structmap[n].status". Valid values are "NOT AVAILABLE", "WAITING", "RUNNING", "SUCCESS", "FAILED" Check Annex 4 of the Technical Analysis document D.CSRA Cellar Status Rest APIs - Technical Analysis.docx for more information on the general indexation status definition.	YES*	YES	YES
indexation.structmaps	The value of this attribute contains a list of items. Each item is presenting information related to the indexation process of a structmap. All structmaps are part of the same package.	YES*	YES	YES
indexation.structmaps.structmap[n].cellarId	The value of this attribute contains the cellar identifier of the top level entity of this structmap. It is presented only if the resource exists.	NO	YES	YES
indexation.structmaps.structmap[n].psis	The value of this attribute contains a list of items. Each item is referring to a production identifier of the structmap.	YES*	YES	YES
indexation.structmaps.structmap[n].structMapID	The value of this attribute contains the name of the structmap as specified in the "structMap ID" field of the mets.xml file contained in the METS package.	YES*	YES	YES
indexation.structmaps.structmap[n].status	The value of this attribute contains a calculated value considering the value indexation.structmaps.structmap[n].requests.request[1].indexation_request_status. Valid values are: "NOT AVAILABLE", "WAITING", "RUNNING", "SUCCESS", "FAILED". Check Annex 4 of the Technical Analysis document D.CSRA Cellar Status Rest APIs - Technical Analysis.docx for more information on the structmap indexation status definition.	YES*	YES	YES
indexation.structmaps.structmap[n].requests	The value of this attribute contains information on the direct index request created for a specific structmap. NOTE: Currently only one index request of this type is created for each ingested structmap.	NO	NO	YES
indexation.structmaps.structmap[n].requests.request[1].indexation_request_status	The value of this attribute contains the status of the index request. Valid values are: "N" (New) "P" (Pending) "X" (Execution) "D" (Done) "E" (Error) "R" (Redundant)	NO	NO	YES
indexation.structmaps.structmap[n].requests.request[1].created_on	The value of this attribute contains the index request creation date.	NO	NO	YES
	The value of this attribute contains the index request execution start date.	NO	NO	YES

indexation.structmaps.structmap[n]. requests.request[1]. execution_start_date				
indexation.structmaps.structmap[n]. requests.request[1].execution_date	The value of this attribute contains the index request execution date.	NO	NO	YES
indexation.structmaps.structmap[n]. requests.request[1].linguistic_versions	The value of this attribute contains a list of languages. For each of these languages an index notice has been created, triggered by the same index request.	NO	NO	YES

Mandatory - YES*: These JSON properties are mandatory only in case the request parameter “ingestion” or “indexation” is set to true. Otherwise, all parameters under “ingestion” or “indexation” properties will not be part of the response.

Several JSON response examples could be found in Annex 1 - JSON response examples of "package-level" endpoints of the Technical Analysis document D.CSRA Cellar Status Rest APIs - Technical Analysis.doc.

Object-level status endpoint

The object-level endpoint provides information on the status of a specific Cellar resource (top-level entity such us work, dossier, agent and top-level event). The endpoint’s resource path is “admin/v1/secapi/status/psi” and the query parameters are:

1. **id**: a mandatory string query parameter uniquely identifying a (possible non-existing) Cellar top-level resource through one of its production system identifiers (PSIs). This value is stored in the STRUCTMAP_PID.PRODUCTION_IDENTIFIER column per ingested structmap.
2. **ingestion**: an optional boolean query parameter specifying whether to include ingestion-specific status information in the response or not. In case it is not specified, the default value is false.
3. **indexation**: an optional boolean query parameter specifying whether to include indexation-specific status information in the response or not. In case it is not specified, the default value is false.

1. **datetime** - an optional datetime query parameter acting as a filter on the received date of the corresponding package. The endpoint returns the latest status information on the object ingested by the package with PACKAGE_HISTORY_RECEIVED_DATE equal or before of the “datetime” parameter. In case it is not specified, the latest status information for the requested object is returned. This field should follow the format "YYYY-MM-DDTHH:mm:ss".

1. **verbosity**: an optional numeric parameter specifying the level of verbosity of a response. Value “1” is corresponding to a coarse level of granularity and “2” to a detailed level of information. In case it is not specified, the default value is 1.

For an object-level request, the response should include the information relevant to the requested object. If the object is part of a multi-structmap METS package, the "object-level" response should contain information only on the structmap referring to the requested object.

JSON response

Please find below a table containing a description and other information for the JSON response attributes.

JSON response attribute	**Description**	**Mandatory**	**Verbosity 1**	**Verbosity 2**
timestamp	The system timestamp when the response is created.	YES	YES	YES
id	A unique ingestion identifier per package.	YES	YES	YES
input	The name of the METS package.	YES	YES	YES
received	A timestamp presenting when the package was moved to the reception folder. The ACCESS time of the UNIX system dates assigned to a file, will be used. The same file date is used also during the ingestion prioritization phase.	YES	YES	YES
request_params.ingestion	The value of this attribute contains a copy of the request’s parameter “ingestion”.	YES	YES	YES
request_params.indexation	The value of this attribute contains a copy of the request’s parameter “indexation”.	YES	YES	YES
request_params.verbosity	The value of this attribute contains a copy of the request’s parameter “verbosity”.	YES	YES	YES
ingestion.status	The value of this attribute contains a calculated value, considering the attributes “ingestion.pre_processing.status”, “ingestion.structmaps.structmap[1].status”. Valid values are “NOT AVAILABLE”, “WAITING”, “RUNNING”, “SUCCESS”, “FAILED” Check Annex 3 of the Technical Analysis document D.CSRA Cellar Status Rest APIs - Technical Analysis.docx for more information on the calculation of the general ingestion status.	YES*	YES	YES
ingestion.pre_processing.status	The value of this attribute contains the package-related pre-processing status, before structmaps processing is launched.	YES*	YES	YES

	Valid values are “NOT AVAILABLE”, “WAITING”, “RUNNING”, “SUCCESS”, “FAILED” Check Annex 2 of the Technical Analysis document D.CSRA Cellar Status Rest APIs - Technical Analysis.docx for more information on the pre_processing status definition.			
ingestion.pre_processing.logs	The value of this attribute contains a list of items. Each item is referring to an entry of the AUDIT_TRAIL_EVENT table. These entries are filtered using the AUDIT_TRAIL_EVENT.PACKAGE_ID column. The information included in each item is specified in the PROCESS, ACTION, TYPE, EVENT_DATE, MESSAGE columns of the database entry.	NO	NO	YES
ingestion.structmaps	The value of this attribute contains a list of items. Each item is presenting information related to the ingestion of a structmap. All structmaps are part of the same package. For the “object-level request”, the JSON response contains only one structmap.	YES*	YES	YES
ingestion.structmaps.structmap[1].cellarId	The value of this attribute contains the cellar identifier of the top level entity of this structmap. It is presented only if the resource exists.	NO	YES	YES
ingestion.structmaps.structmap[1].psis	The value of this attribute contains a list of items. Each item is referring to a production identifier of the structmap.	YES*	YES	YES
ingestion.structmaps.structmap[1].structMapID	The value of this attribute contains the name of the structmap as specified in the “structMap ID” field of the mets.xml file contained in the METS package.	YES*	YES	YES
ingestion.structmaps.structmap[1].status	The value of this attribute contains the status of the structmap ingestion process. Valid values are “NOT AVAILABLE”, “WAITING”, “RUNNING”, “SUCCESS”, “FAILED” Check Annex 2 of the Technical Analysis document D.CSRA Cellar Status Rest APIs - Technical Analysis.docx for more information on the structmap status definition.	YES*	YES	YES
ingestion.structmaps.structmap[1].logs	The value of this attribute contains a list of items. Each item is referring to an entry of the AUDIT_TRAIL_EVENT table. These entries are filtered using the AUDIT_TRAIL_EVENT.STRUCTMAP_ID column. The information included in each item is specified in the PROCESS, ACTION, TYPE, EVENT_DATE, MESSAGE columns of the database entry.	NO	NO	YES
ingestion.structmaps.structmap[1].shacl	The value of this attribute contains the SHACL validation report of the structmap. The SHACL report’s content is defined by a configuration parameter specifying if the report should contain: 1. Errors and Warnings 2. Only Errors 3. No report	NO	NO	YES
indexation.status	The value of this attribute contains a calculated value, considering the attribute “indexation.structmaps.structmap[1].status”. Valid values are “NOT AVAILABLE”, “WAITING”, “RUNNING”, “SUCCESS”, “FAILED” Check Annex 4 of the Technical Analysis document D.CSRA Cellar Status Rest APIs - Technical Analysis.docx for more information on the general indexation status definition.	YES*	YES	YES
indexation.structmaps	The value of this attribute contains a list of items. Each item is presenting information related to the indexation of a structmap. All structmaps are part of the same package. For the “object-level request”, the JSON response contains only one structmap.	YES*	YES	YES
		NO	YES	YES

indexation.structmaps.structmap[1].cellarId	The value of this attribute contains the cellar identifier of the top level entity of this structmap. It is presented only if the resource exists.			
indexation.structmaps.structmap[1].psis	The value of this attribute contains a list of items. Each item is referring to a production identifier of the structmap.	YES*	YES	YES
indexation.structmaps.structmap[1].structMapID	The value of this attribute contains the name of the structmap as specified in the “structMap ID” field of the mets.xml file contained in the METS package.	YES*	YES	YES
indexation.structmaps.structmap[1].status	The value of this attribute contains a calculated value considering the value indexation.structmaps.structmap[1].requests.request[1].indexation_request_status. Valid values are: “NOT AVAILABLE”, “WAITING”, “RUNNING”, “SUCCESS”, “FAILED”. Check Annex 4 of the Technical Analysis document D.CSRA Cellar Status Rest APIs - Technical Analysis.docx for more information on the structmap indexation status definition.	YES*	YES	YES
indexation.structmaps.structmap[1].requests	The value of this attribute contains information on the direct index request created for a specific structmap. NOTE: Currently only one index request of this type is created for each ingested structmap.	NO	NO	YES
indexation.structmaps.structmap[1].requests.request[1].indexation_request_status	The value of this attribute contains the status of the index request. Valid values are: “N” (New) “P” (Pending) “E” (Executing) “D” (Done) “E” (Error) “R” (Redundant)	NO	NO	YES
indexation.structmaps.structmap[1].requests.request[1].created_on	The value of this attribute contains the index request creation date.	NO	NO	YES
indexation.structmaps.structmap[1].requests.request[1].execution_start_date	The value of this attribute contains the index request execution start date.	NO	NO	YES
indexation.structmaps.structmap[1].requests.request[1].execution_date	The value of this attribute contains the index request execution date.	NO	NO	YES
indexation.structmaps.structmap[1].requests.request[1].linguistic_versions	The value of this attribute contains a list of languages. For each of these languages an index notice has been created, triggered by the same index request.	NO	NO	YES

Mandatory - YES*: These JSON properties are mandatory only in case the request parameter “ingestion” or “indexation” is set to true. Otherwise, all parameters under “ingestion” or “indexation” properties will not be part of the response.

Several JSON response examples could be found in Annex 1 - JSON response examples of the "object-level" endpoint of the Technical Analysis document D.CSRA Cellar Status Rest APIs - Technical Analysis.doc.

Status endpoint lookup

The Status endpoint lookup REST API provides information on the status endpoint(s) based on the name of a METS package. The endpoint’s resource path is “admin/v1/secapi/status/package” and the query parameter is:

id: a mandatory string query parameter containing the name of a METS package. This value is stored in the PACKAGE_HISTORY.METS_PACKAGE_NAME column per ingested package.

JSON response

The JSON response of the Status endpoint lookup RETS API contains the status package-level endpoint together with the received date of the package. In case there are multiple ingestions for the same package name, a list of items is returned in the response. As a result, the package received date could be used to distinguish the different ingestion processes initiated by a package with the same name.

Please find below a table containing a description and other information for the JSON response attributes.

JSON response attribute	Description

status_endpoints	The value of this attribute contains a list of items. Each item is referring to a status package-level endpoint.
status_endpoints [n].status_endpoint	The value of this attribute contains the “package-level” status endpoint.
status_endpoints [n].received	A timestamp presenting when the package was moved to the reception folder. The ACCESS time of the UNIX system dates assigned to a file, will be used. The same file date is used also during the ingestion prioritization phase.

A JSON response example follows:

```
{
  "status_endpoints": [
    {
      "status_endpoint": "http://<SERVICE>/secapi/status/mets?id=083bf55c-b771-11ed-afa1-0242ac120002",
      "received": "2023-02-08T14:12:40.112Z"
    },
    {
      "status_endpoint": "http://<SERVICE>/secapi/status/mets?id=e03057a8-bce4-11ed-afa1-0242ac120005",
      "received": "2022-01-15T12:11:10.336Z"
    }
  ]
}
```

Authentication

The Status endpoints are authenticated REST APIs using EU Login authentication. Find below more information on EU Login authentication per different type of actor (CELLAR user or M2M/job).

CELLAR user: Upon a successful login via EU Login to cellar admin interface, the status endpoints functionality can be accessible by authorized users by visiting the specific endpoint urls after adjustment of course the required url params as detailed in the respective subsections (e.g package level status endpoint) of Status service section, provided above. For example user can directly access the package level status endpoint by visiting http://<HOST: PORT>/admin/v1/secapi/status/mets?id=062ef66c-b771-11ed-afa1-0242ac120002.

Alternatively, there is a hybrid workaround enabling end users to successfully trigger Status endpoints via cURL. In this case the following steps are required:

- User logs in to the CELLAR Admin UI (any page) using a web browser after a successful EU-Login authentication is taken place.
- User opens the browser’s cookie panel and copies the value of CELLAR’s session cookie named JSESSIONID.
- User triggers the Status endpoints via cURL while supplying the CELLAR session cookie, in order to use the already authenticated session.

Note: Although this methodology enables the usage of the endpoints via cURL it is considered a unofficial approach; additionally, in case the session expires, the user would have to repeat all steps in order to acquire a fresh session cookie. Considering the above, this is not recommended for production environment.

Example - Package-level status endpoint

```
curl -X GET "http://<HOST>:<PORT>/admin/v1/secapi/status/mets" -F "id=083bf55c-b771-11ed-afa1-0242ac120002" -F
"ingestion=true" -F "indexation=true" -F "verbosity=2" -b "JSESSIONID=B12AA1BBEE27C9314B192826EE894938"
```

Example - Object-level status endpoint

```
curl -X GET "<HOST>:<PORT>/admin/v1/secapi/status/psi" -F "id=083bf55c-b771-11ed-afa1-0242ac120002" -F
"ingestion=true" -F "indexation=true" -F "verbosity=2" -b "JSESSIONID=B12AA1BBEE27C9314B192826EE894938"
```

Example - Status endpoint lookup

```
curl -X GET "<HOST>:<PORT>/admin/v1/secapi/status/package" -F "id=package.zip" -b
"JSESSIONID=B12AA1BBEE27C9314B192826EE894938"
```

M2M/job: Check CELLAR - Operation Manual#Machinetomachinecommunication for more information on the different approaches to trigger Status endpoints.

Authorization

The Status endpoints is also authorized using the existing Cellar authorization mechanism. More specifically, any actor attempting to access the Status endpoints should be assigned with the status service role. Find below more information on the authorization for each type of actor:

- **CELLAR users** assigned with the status service role, via the Security panel of the Cellar Admin UI, are authorized to use the Status endpoints.
- **M2M/job**, are authorized to use the Status endpoints similarly to the users' scenario previously mentioned, with the additional prerequisite that a CELLAR user with the same name as the EU-Login Job/certificate account needs to be created beforehand.

Annexes

Annex 1: List of ISO_639-3 codes of supported European languages

The Cellar supports the European languages identified by the following 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 2: 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:

- it is independent from the OS
- the way a browser allows the user to build the HTTP requests may differ from browser to browser
- its syntax does not depend on the version used, while the browser may change during time the way it represents the HTTP request
- 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 3: JSON

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

It has several advantages:

- it is easy for humans to read and write
- it is easy for machines to parse and generate
- it is based on a subset of the JavaScript Programming Language, used worldwide
- 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 4: 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.

The last version of Cellar CDM is accessible via the WIKI:

- http://publications.europa.eu/mdr/resource/cdm

Annex 5 : List of available mime types

The list of the current available mime types is based on the latest NAL file type available on the MDR website, which enlists the following:

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

1. If CELLAR property *fedora.service.ingestion.datastream.renaming.enabled* is set to *false*, the entry is retrievable as http://publications.europa.eu/resource/cellar/{man-id}/{stream-name}/{entry-name}.