

Guidelines for EPOS-DCAT-AP Metadata provision

February, 2021

TABLE OF CONTENTS

1	INTRODUCTION	3
2	BACKGROUND	3
2.1	EPOS-DCAT-AP METADATA MODEL	3
2.2	EPOS-DCAT-AP RDF/TURTLE SERIALIZATION	4
2.3	EPOS-DCAT-AP METADATA VALIDATION	5
2.4	EPOS-DCAT-AP METADATA COLLECTION	5
3	EDITING AND VALIDATION OF EPOS-DCAT-AP METADATA	6
3.1	MANUAL EDITING AND VALIDATION	6
3.2	USING THE EPOS METADATA EDITOR	7
4	EPOS-DCAT-AP METADATA QUALITY CONTROL	8
4.1	VISUAL PRESENTATION OF METADATA IN GUI	8
4.2	RESOURCE DISCOVERY	9
4.3	RESOURCE DETAILS	10
4.4	RESOURCE (RE)CONFIGURATION	13
5	SUBMITTING EPOS-DCAT-AP METADATA	15
5.1	MANUAL PUSHING METADATA TO GitLAB	15
5.2	PUSHING METADATA TO GitLAB BY USING THE SHAPENESS METADATA EDITOR	15
6	TESTING EPOS-DCAT-AP METADATA	16
7	CONTACTS AND REFERENCES	17
	APPENDIX 1: TEMPORAL METADATA	18
	TEMPORAL INFORMATION FOR DISCOVERY DDSS	18
	TEMPORAL INFORMATION FOR QUERY DDSS	18
	APPENDIX 2: SPATIAL METADATA	20
	SPATIAL INFORMATION FOR DISCOVERY DDSS	20
	SPATIAL METADATA FOR QUERY DDSS	21

1 Introduction

A major challenge in EPOS¹ is the integration of multi-disciplinary, multi-organisational, distributed resources and community assets into a single overarching Research Infrastructure - the EPOS Integrated Core Services (ICS). ICS aggregate and harmonise descriptions of datasets, data products, software and services from different Thematic Core Services (TCS). TCS adopt heterogeneous formats, vocabularies, protocols and standards to represent and make their resources available.

The exchange of metadata between ICS and TCS is crucial to achieve integration and interoperability in EPOS. In order to capture, organise and harmonise information from different sources and to enable semantic interoperability, an extension of DCAT-AP² has been developed and adopted, namely EPOS-DCAT-AP³.

This document has been created to provide a common background with respect to the collection of EPOS-DCAT-AP metadata, and to guide users and metadata curators in entire metadata lifecycle, starting from the creation of metadata descriptions to the submission into the ICS-C platform.

2 Background

2.1 EPOS-DCAT-AP metadata model

EPOS-DCAT-AP is an Application Profile built on DCAT, a W3C Recommendation, developed to address the specific requirements of EPOS.

EPOS-DCAT-AP follows the DCAT-AP recommendations for extensions and includes additional entities and relationships (Fig. 1). In particular, the additional classes enable the description of additional concepts beyond **Dataset** (the main focus of DCAT). For instance, the **WebService** and **Operation** entities, modelled leveraging Schema.org and the Hydra Vocabulary, allow to have flexible and fine-grained representations covering the broad EPOS spectrum that includes both global, well-established and community specific standards for web services e.g. OGC, FDSN.

More detailed information about EPOS-DCAT-AP metadata model are available on GitHub.

¹ <https://www.epos-eu.org/>

² <https://joinup.ec.europa.eu/collection/semantic-interoperability-community-semic/solution/dcat-application-profile-data-portals-europe/release/11>

³ <https://github.com/epos-eu/EPOS-DCAT-AP>

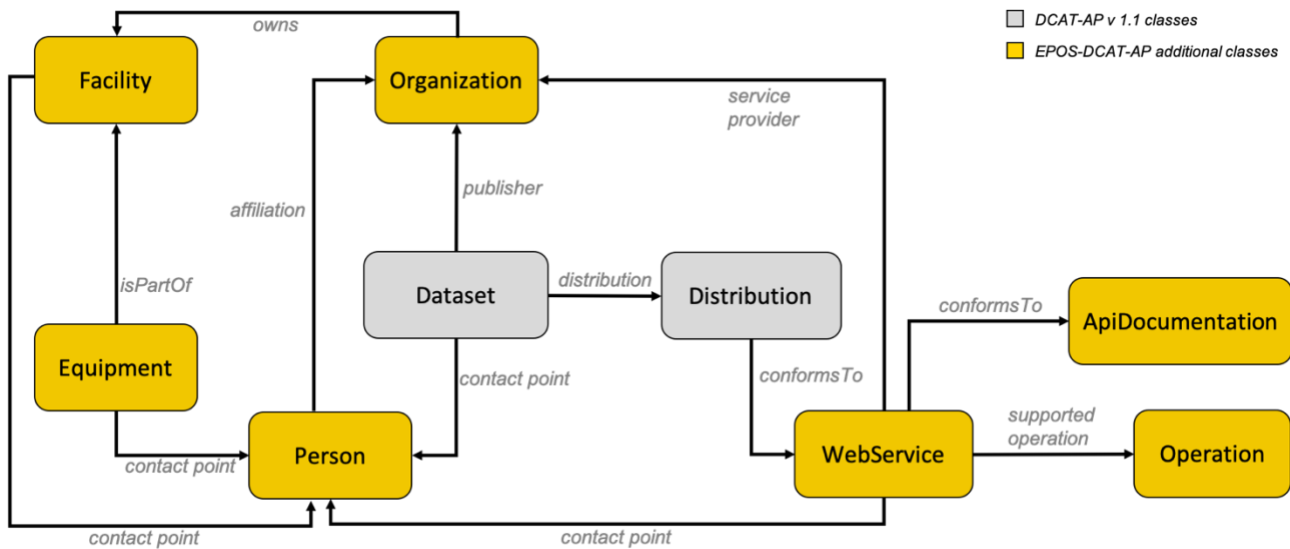


Figure 1 – A simplified EPOS-DCAT-AP UML class diagram.

2.2 EPOS-DCAT-AP RDF/Turtle serialization

EPOS-DCAT-AP metadata model is represented in Resource Description Framework (RDF⁴). RDF data is commonly represented in the form of triples, comprising three components: subject, predicate, and object. The subject and object of a triple can be regarded as concepts (entities), or literals such as strings or numbers. The predicate can be regarded as the “label” of the edge, capturing the semantics of the expressed relationship.

EPOS-DCAT-AP metadata model is serialized in Turtle⁵ format. Files in Turtle serialization are usually given the file extension “.ttl”.

Figure 2 shows an example of the metadata description, serialized in RDF/Turtle format, of a **Dataset** entity linked to a **Distribution** entity that allow to describe the **WebService** (used for accessing the data) and its **Operation** (i.e., endpoint URL and list of parameters).

⁴ <https://www.w3.org/RDF/>

⁵ <https://www.w3.org/TR/turtle/>

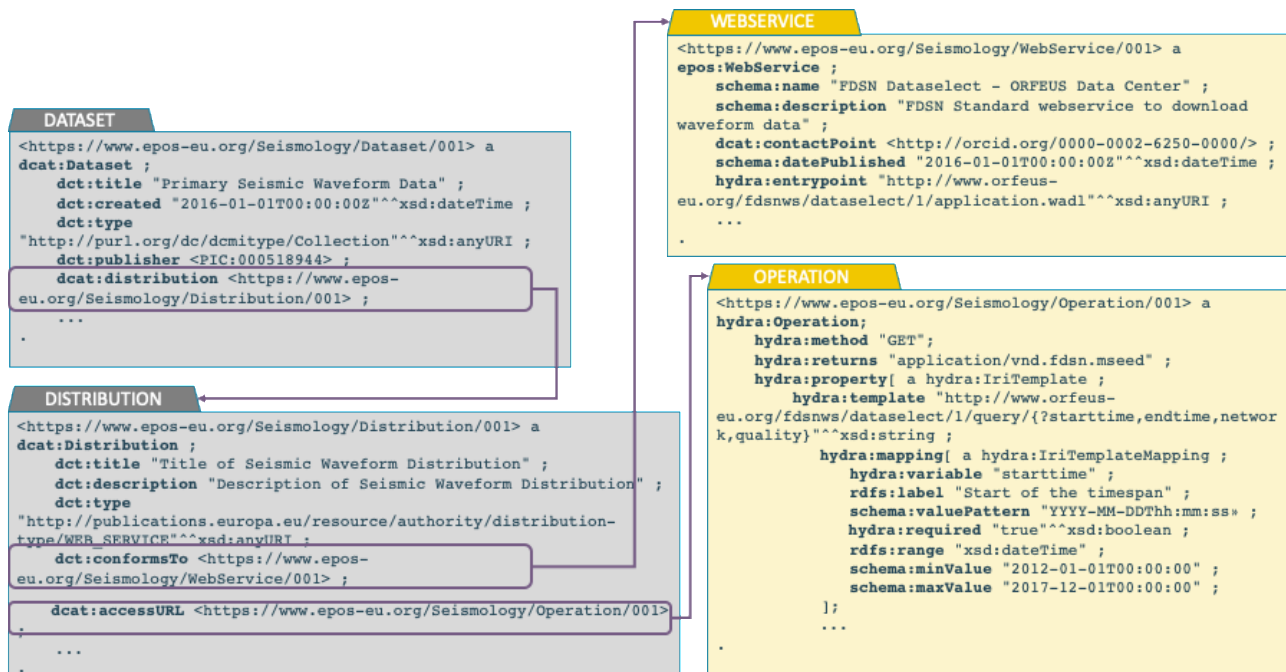


Figure 2 – Example of serialization in RDF/Turtle format of Dataset, Distribution, WebService and Operation entities.

2.3 EPOS-DCAT-AP metadata validation

A W3C recommendation defines Shapes Constraint Language (SHACL⁶) as language for validating RDF content against a set of constraints. EPOS-DCAT-AP defines these constraints, namely defines classes together with constraints on their properties (e.g., cardinality, value type, allowed values, etc.), in order to support the metadata validation and consistency check.

EPOS-DCAT-AP SHACL file is available on GitHub⁷.

2.4 EPOS-DCAT-AP metadata collection

EPOS adopts an incremental metadata collection process which is carried out in stages by prioritizing specific metadata entities.

Currently, the top priority entities are: **Dataset**, **Distribution**, **WebService**, **Operation**, **Person** and **Organization**. Nevertheless, the metadata collection also concerns other EPOS relevant assets, like **Facility**, **Equipment** and **Software**, which has started.

⁶ <https://www.w3.org/TR/shacl/>

⁷ https://github.com/epos-eu/EPOS-DCAT-AP/blob/EPOS-DCAT-AP-shapes/epos-dcat-ap_shapes.ttl

3 Editing and validation of EPOS-DCAT-AP metadata

EPOS-DCAT-AP Turtle files can be edited and validated manually, or through the EPOS Metadata Editor. Either way, it is recommended to follow the metadata template file (https://github.com/epos-eu/EPOS-DCAT-AP/blob/EPOS-DCAT-AP-shapes/examples/EPOS-DCAT-AP_example.ttl) which contains the minimum required metadata elements, thus reducing the workload of metadata authors.

3.1 Manual editing and validation

The manual editing of EPOS-DCAT-AP Turtle files can be performed by using a simple text editor (e.g. Notepad, Atom), although this is not recommended unless the user is familiar with Turtle Syntax.

In order to provide metadata of sufficient quality, it is important to detect and deal with syntax errors, as well as validate the content against EPOS-DCAT-AP SHACL constraints.

There are several tools which support both tasks (i.e., syntax and consistency check).

For instance, the online tool available at <http://shacl.org/playground/> can be used following these steps:

1. Copy and paste the content of the EPOS-DCAT-AP SHACL file (https://raw.githubusercontent.com/epos-eu/EPOS-DCAT-AP/EPOS-DCAT-AP-shapes/epos-dcat-ap_shapes.ttl) in the "Shapes Graph" upper left box (Figure 3a);
2. Click on the "Update" button below the left window;
3. Copy and paste the content of the Turtle file in the "Data Graph" upper right box (Figure 3b);
4. Select Turtle format from drop-down list at the bottom;
5. Click on the "Update" button below the right window;
 - a red notification will advise when syntax errors are detected;
6. The "Validation Report" box shows the result of validation (Figure 3c). It reports violations (*sh:Violation*) and warnings (*sh:Warning*). All violations must be fixed in order to have a valid Turtle file.

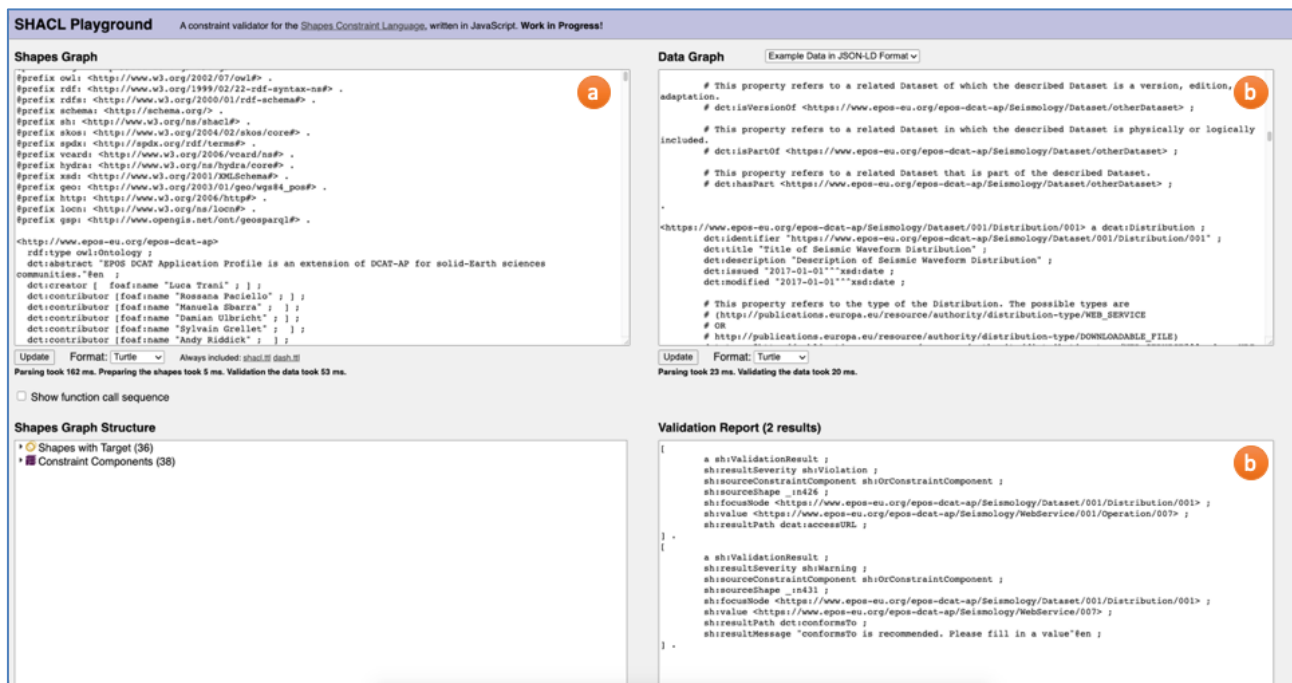


Figure 3 – Example of using of an online tool (<http://shacl.org/playground>) in order to validate EPOS-DCAT-AP Turtle files

3.2 Using the EPOS Metadata Editor

The manual editing of metadata can be a time-consuming, sometimes tiresome process and error-prone. For this reason, a metadata editor, namely SHAPEness, was developed. SHAPEness is a desktop application conceived to help users creating and updating RDF Turtle metadata files. The SHAPEness user interface allows users to easily populate and validate metadata (i.e., syntax and consistency validation), structured as graphs, against a set of SHACL constraints. SHAPEness binaries, with setup instructions and the user manual, are available on GitHub <https://epos-eu.github.io/SHAPEness-Metadata-Editor/index.html>.

4 EPOS-DCAT-AP Metadata Quality Control

4.1 Visual presentation of metadata in GUI

The provision of quality metadata aims at: 1) guaranteeing the appropriate functionality of the EPOS ICS-C — for instance while searching and filtering resources of interest; and 2) providing useful and meaningful information to end-users through the EPOS Data Portal. Clear and correct information helps users to understand what data they can access, download, and also what services are available.

Sections 4.2, 4.3, 4.4, outline some basic rules about how to fill in information in the EPOS-DCAT-AP files, also by providing examples of what metadata values should or shouldn't be provided in order to furnish useful and clear information to the end users.

In Figure 4, a screenshot of the EPOS Data Portal is presented. It illustrates where the metadata values appear in order to provide a better idea about what metadata values are visualized in what elements of the GUI. Box 1 is covered in section 4.2, where metadata elements for discovery of resources, search and filters are discussed; Box 2, discussed in section 4.3, is related to metadata elements describing resources details; metadata elements influencing Box 3, i.e. (re)configuration of resources, are discussed in section 4.4.

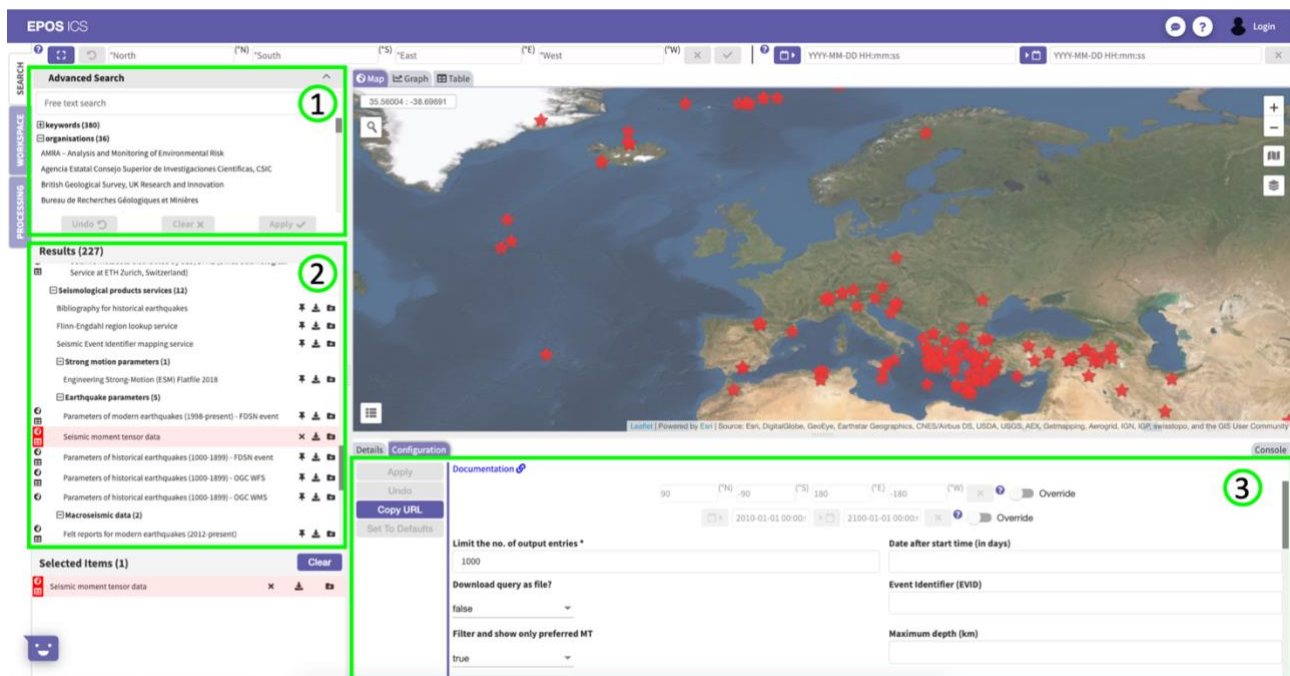


Figure 4 - The EPOS data portal. Green numbered boxes mark three main tasks that can be achieved on the GUI's "search" panel: (1) Discovery, search, and filtering; (2) listing and display of resource details; (3) (re-)configuration of resources.

4.2 Resource discovery

The EPOS data portal will support a number of mechanisms to search for and to discover resources such as **Dataset** and **Web service**. In particular, it will enable users to:

- look for specific terms or keywords associated with a resource;
- navigate through facets that combine groups of resources;
- filter resources provided by a specific Institution and/or Organisation;
- apply spatio-temporal filters.

As such functionalities are enabled by the underpinning information maintained in the metadata catalogue, it is important to verify that this information is as correct and complete as possible.

Facets

The facets organise related resources into groups of elements structured by scientific domains (TCSs). Currently, the facets are static and defined in a JSON file according to the descriptions available at https://docs.google.com/spreadsheets/d/1_heTbkSFrgPZWPC3WNboJMqfCaxdm4VMA_EmBG9kK8k/edit?usp=sharing. These have been provided by TCS representatives who are responsible for their definition and maintenance. Each facet is linked to **Dataset** entity by DDSS ID(s).

DDSS-IDs

DDSS-IDs, associated to the resources, need to be listed in the **Dataset** entities by using *adms:identifier* property. The values of such DDSS-IDs together with the facets allow the GUI to structure the results in a properly way.

Example:

```
<http://services.seismofaults.eu/geoserver/EDSF> a dcat:Dataset;
  dct:title "Seismology/European Database of Seismogenic Faults";
  dct:identifier "http://services.seismofaults.eu/geoserver/EDSF";
  adms:identifier [ a adms:Identifier;
    adms:schemeAgency "DDSS-ID";
    skos:notation "WP08-DDSS-038";
  ];
  ...
.
```

Keywords

Keywords need to be associated with **Dataset** (by using *dcat:keyword*) and **WebService** (by using *schema:keywords*). The values of such keywords should come from a controlled list which is available at <https://docs.google.com/spreadsheets/d/1etf7WVTJP3NJ9Hni1YBvLF33k66FAyNf1qWYAxtHZEw/edit?usp=sharing>. It is recommended to use the values in the column “*proposed*”. This activity is fundamental for the proper and correct functioning of the search features.

Institutions/organizations

Organizations and institutions should be described by using the **Organization** entity; *schema:legalName* property will be used by the GUI in the facets search. Please ensure that you only use organization names that agree with the controlled list of organizations available at the following link (if you discover errors or missing institutions in the controlled list, please correct them or leave a comment):

https://docs.google.com/spreadsheets/d/1xcYHyh5_cS3sCevSmieZD5oTvPMi7s8xQZDP6RRtA_8/edit#gid=0

After defining the **Organization** entities, it is possible to link each of these to one or more **Datasets** (by using *dct:publisher*) and **WebService** (by using *schema:provider*). It is possible to define multiple Data Providers institutions for the same Service Provider.

Spatio-temporal information

The spatio-temporal coverage needs to be specified by using *dct:spatial* and *dct:temporal* properties in **Dataset** and **WebService** entities.

Appendix 1 and 2 provide more details about spatio-temporal metadata and show some examples on how to provide them.

4.3 Resource details

The values of the properties shown in the following table are used to populate the details of the resource. In particular, *dct:title* property of **Distribution** entity is used as the name of the resource into the results list (Figure 4 - box 2), while the other properties are used to populate the Details pane (Figure 4 - box 3).

Service Providers should take special care in reviewing the resource names (*dct:title* of **Distribution**, *schema:name* of **WebService**) and their descriptions (*dct:description* of **Distribution**, *schema:description* of **WebService**). Names and descriptions should briefly but clearly describe what a service is about.

Domain-experts who know the service should be able to identify it, but also geoscientists who have never used the service should be able to understand its purpose.

Distribution		
Metadata element	Rules	Example
dct:title	<ol style="list-style-type: none"> 1. Should succinctly describe the Distribution using common terminology, i.e., understandable by geoscientists. 2. Should not include institution/organization information (exception: if multiple resources have the same name, the institution name can be appended). 3. Should not use any acronyms. 	Seismic waveforms from the National Observatory of Athens
dct:license	Should list the licence under which the Distribution is made available	dct:license "https://creativecommons.org/licenses/by-nc/4.0/"^^xsd:anyURI ;
dct:description	<ol style="list-style-type: none"> 1. Should be written in complete sentences. 2. Should provide information on <ul style="list-style-type: none"> • what the distribution provides; • any standard format that it may adhere to (e.g., CSV, NetCDF, XML); • what important coverage limits apply to the underlying data (e.g., specific region). 3. Should not include institution/organization information. 	<p>Suggested standard formulation: <i>"This distribution provides access to dataset XYZ, including x, y, and z. It is encoded in the xy standard format, version X. Its related dataset covers the time period since YYYY and is limited to countryX."</i></p> <p><i>E.g.:</i> This distribution provides access to time series of temperatures and radon gas concentrations at sampling sites.</p>

WebService		
Metadata element	Rules	Example
schema:description	<ol style="list-style-type: none"> 1. Should be written in complete sentences. Start with a capital letter. 2. Should provide information on (a) what the service provides, (b) any implementation standards that it may adhere to (e.g., FDSN station web service 1.0), (c) whether any limits apply to the response (e.g., maximum 1000 events), and (d) what important coverage limits apply to the underlying data (e.g., specific region or a catalog's start time). 3. Should not include institution/organization information. 	<p>Access to seismic station, instrumentation, and response data-Service implements FDSN station web service 1.0 standard</p> <p>Suggested standard formulation: <i>"This web service provides access to XYZ, including x, y, and z. It is implemented according to the xy web service standard, version X. It returns no more than NNN measurements per request. The catalog/dataset covers the time period since YYYY and is limited to countryX."</i></p> <p><i>E.g.:</i> This web service provides access to seismic station metadata, including locations, operation periods, instrumentation, and response data. It is implemented according to the standard 1.0 for seismic station web services set by the Federation of Digital Seismograph Networks (FDSN).</p>
schema:name	<ol style="list-style-type: none"> 1. Should succinctly describe the Resource using common terminology (i.e., understandable by geoscientists). 2. Should not include institution/organization information (exception: if multiple resources have the same basic name, then the institution can be added behind the basic name). 3. Should not use any acronyms. 	<p>Waveform Catalog of the European Integrated Data Archive</p>

4.4 Resource (re)configuration

A resource, which is a web service, needs to be described properly by specifying and defining the parameters that allow to query it.

The configuration pane (Figure 4 - box 3), is created by using the **Operation** entity and its properties.

In particular: *rdfs:label* is used to describe the meaning of the parameter; *hydra:required*, indicates whether the parameter is mandatory or not; *hydra:property*, contains the semantic description of the parameter (e.g., *epos:westernmostLongitude*). It is also used by the GUI to organize parameters in the configuration pane in three groups: "spatio-temporal", "mandatory" and "optional".

The output format of the web service payload needs to be specified by defining *hydra:returns* property. In case the web service supports multiple output formats, then they need to be specified by defining: a) multiple occurrences of *hydra:returns* property (one for each output format); b) a *hydra:property* property with a "schema:encodingFormat" value, as shown below.

```
hydra:mapping [ a hydra:IriTemplateMapping;
  hydra:variable "format"^^xsd:string;
  rdfs:label "Output format";
  hydra:property "schema:encodingFormat";
  rdfs:range "xsd:string";
  http:paramValue "JSON"^^xsd:string;
  http:paramValue "CSV"^^xsd:string;
  schema:defaultValue "JSON"^^xsd:string;
  hydra:required "false"^^xsd:boolean;
];
```

In order to help users to understand how to use the web service parameters, it is possible to link **API Documentation** entity to the **WebService** entity. The documentation URL, specified by using *hydra:entrypoint* property, will appear in the GUI configuration pane.

Operation		
Metadata element	Rules	Example
rdfs:label	This string describes the meaning of the parameter. 1. Use short (ca. 10-50 characters), but clear descriptions.	rdfs:label "Network code" ; rdfs:label "Date after start time (in days)" :

	<p>2. Follow standard capitalization rules (first word, proper nouns, etc.) for better readability.</p> <p>2. Denote the physical units</p> <p>3. Use standard wording for the following labels (if applicable, adjust only if needed):</p> <p>Minimum longitude (deg)</p> <p>Maximum longitude (deg)</p> <p>Minimum latitude (deg)</p> <p>Maximum latitude (deg)</p> <p>Start time</p> <p>End time</p>	<p>rdfs:label "Maximum radius (deg)" :</p>
hydra:returns	<p>The possible values are listed here:</p> <p>https://www.iana.org/assignments/media-types/media-types.xhtml</p>	<p>hydra:returns "application/vnd.fdsn.mseed" ;</p>
hydra:property	<p>schema:startDate;</p> <p>schema:endDate;</p> <p>epos:southernmostLatitude;</p> <p>epos:northernmostLatitude;</p> <p>epos:westernmostLongitude;</p> <p>epos:easternmostLongitude;</p> <p>schema:encodingFormat;</p>	<p>hydra:property "schema:endDate";</p>
hydra:required	<p>"true" or "false";</p>	<p>hydra:required "true";</p>

WebService		
Metadata element	Rules	Example
dct:conformsTo	<p>This property refers to the API documentation entity.</p>	<p>dct:conformsTo <https://www.epos-eu.org/epos-dcat-ap/Seismology/WebService/001/APIDocumentation> ;</p>

APIDocumentation		
Metadata element	Rules	Example
hydra:entrypoint	<p>The URL of API documentation.</p>	<p>hydra:entrypoint "https://www.emidius.eu/AHEAD/services/"^xsd:any URI;</p>

5 Submitting EPOS-DCAT-AP metadata

Once the EPOS-DCAT-AP metadata files have been created or updated, they need to be pushed to the EPOS GitLab repository (<https://epos-ci.brgm.fr/ics-tcs/epos-dcat-ap>).

5.1 Manual pushing metadata to GitLab

Users who have created EPOS-DCAT-AP files manually should follow these steps:

1. Log in to GitLab <https://epos-ci.brgm.fr/ics-tcs/epos-dcat-ap>
2. Enter in your TCS repository folder;
3. Create a branch by clicking on "+" button and then on "New branch"
4. The name of the new branch can only contain lower case characters, numbers and hyphens (max length 30);
5. GitLab will redirect you into your branch;
6. Create or upload a new file by clicking on "+" button. If you need to update an existing file, open the desired file and then click on the "Edit" button;
7. Do the commit, inserting a proper label;
8. Click on the blue button on top-right "Create merge request";
9. Insert a title and a description for the request;
10. Assign the merge request by adding in the "Assignee" field: "Rossana Paciello";
11. Finally confirm by clicking on "Submit merge request";
12. The updated file will be merged into the master branch as soon as the maintainer (i.e. Rossana Paciello) will approve your request.

5.2 Pushing metadata to GitLab by using the SHAPEness Metadata Editor

SHAPEness Metadata Editor allows users to push Turtle files, created or updated through it, directly to the EPOS GitLab repository. This action can be performed by inserting the Git repository URL (e.g., <https://epos-ci.brgm.fr/ics-tcs/epos-dcat-ap/tcs-gim.git>), the branch name from which to create the new one (e.g., master), and the authentication credentials.

The metadata editor will create a new branch called "*username-timestamp-shapeness*" (e.g., *rossanapaciello-202102251530-shapeness*).

A detailed explanation about that is provided by the section 4.5 "Exporting Turtle File" of the SHAPEness guideline.

In order to notify the EPOS metadata maintainer about the changes it is necessary to manually perform the steps 8 - 12 described in the previous section 5.1.

6 Testing EPOS-DCAT-AP metadata

The pushing of an EPOS-DCAT-AP metadata file to the EPOS GitLab (manually or via Metadata Editor) triggers an automated ingestion to the ICS. Users can check the content of the created or updated metadata by using a dedicated GUI testing environment. It can be done by following these steps.

- Go to the pipeline of the metadata-cache: <https://epos-ci.brgm.fr/epos/metadata-cache/-/pipelines/new>
- Select **ONLY** your TCS-*** branch: e.g. TCS-GIM
- Click on RUN PIPELINE (no other parameters are needed to be changed)
- The full ingestion can take 20 minutes even if the pipeline is marked as finished
- You can then access your test environment from your web browser:
http://ics-c.epos-ip.org/testing/epos-metadata-cache/tcs-***
 - i.e. for TCS-GIM: <http://ics-c.epos-ip.org/testing/epos-metadata-cache/tcs-gim>
- The testing environment will be deleted every night.

7 Contacts and references

Contacts:

- **Rossana Paciello**, rossana.paciello@ingv.it, for EPOS-DCAT-AP Metadata description issues;
- **Jan Michalek**, jan.michalek@uib.no, for DDSS descriptions and other issues;
- **Jean-Baptiste Roquencourt**, jb.roquencourt@brgm.fr, for the EPOS GitLab repository issues.

References:

- **EPOS-DCAT-AP Metadata Model:** <https://github.com/epos-eu/EPOS-DCAT-AP>
- **SHAPEness Metadata Editor:** <https://epos-eu.github.io/SHAPEness-Metadata-Editor/>
- **EPOS GitLab Repository:** <https://epos-ci.brgm.fr/ics-tcs/epos-dcat-ap>

Appendix 1: Temporal metadata

This appendix shows how and where to enter the temporal information in EPOS-DCAT-AP metadata description.

Temporal information for discovery DDSS

The temporal coverage is specified by using `dct:temporal` property in **Dataset** and **WebService** entities. In particular, this is done by using `schema:startDate` and `schema:endDate`. The date format to be used for these elements is "YYYY-MM-DDThh:mm:ssZ", as shown in the following examples.

DATASET example

```
dct:temporal [ a dct:PeriodOfTime ;
               schema:startDate "1992-03-23T02:15:00Z"^^xsd:dateTime ;
               schema:endDate "2010-12-20T23:00:00Z"^^xsd:dateTime ;
             ] ;
```

WEBSERVICE example

```
dct:temporal [ a dct:PeriodOfTime ;
               schema:startDate "1992-03-23T02:15:00Z"^^xsd:dateTime ;
               schema:endDate "2010-12-20T23:00:00Z"^^xsd:dateTime ;
             ] ;
```

Please note that `schema:endDate` may be omitted if **Dataset/Webservice** is continuously accumulating data.

Temporal information for query DDSS

In order to manage, in the appropriate way, the web service parameters that allow to filter DDSS by temporal range, it is necessary to add semantic information in **Operation** entity. This is done by adding **hydra:property** and **schema:valuePattern** elements to the properly parameters.

Metadata element	Description	Allowed values
hydra:property	It contains the vocabulary term which indicates the semantic description of parameter.	- <code>"schema:startDate"</code> , if the parameter represents the start date;

		- "schema:endDate", if the parameter represents the end date;
schema:valuePattern	It contains the regular expression for testing values according to the parameter's specification.	E.g. "YYYY-MM-DD"; "YYYY-MM-DDThh:mm:ss"; "YYYY"; Please refer to the information reported in the spreadsheet: https://docs.google.com/spreadsheets/d/1n-iQVuzW0jc3xwAKl8DE3rePxxhZgfHm7DtzuMByDm8Y/edit?usp=sharing

The following example shows how to use these elements in order to describe the web service parameters used to filter data by temporal range.

OPERATION Example

```
hydra:mapping[ a hydra:IriTemplateMapping;
  hydra:variable "epoch_start";
  hydra:property "schema:startDate";
  schema:valuePattern "YYYY-MM-DD";
  ...
];
```

```
hydra:mapping[ a hydra:IriTemplateMapping;
  hydra:variable "epoch_end";
  hydra:property "schema:endDate";
  schema:valuePattern "YYYY-MM-DD";
  ...
];
```

Appendix 2: Spatial metadata

This appendix shows how and where to enter the spatial information in EPOS-DCAT-AP metadata description.

Spatial information for discovery DDSS

The spatial coverage is specified by using `dct:spatial` property in **Dataset** and **WebService** entities. This property allows to specify the geographic region covered by the **Dataset/WebService** by using one or more geometries denoted by **loc:geometry**. It is important to specify three mandatory items: coordinates, geometry type and coordinate reference system (CRS). The coordinates represent coordinates of the geographic area covered by the **Dataset/WebService**; geometry type is the type of geometry that characterizes the spatial object of the **Dataset/WebService** (e.g., polygon, point); CRS is the spatial reference system in which the data are represented. The syntax encoding used in EPOS-DCAT-AP to describe geometries is Well-known text (WKT). In the WKT representation, CRS84 (<http://www.opengis.net/def/crs/OGC/1.3/CRS84>) is used as a default CRS and it shall be assumed as CRS when it is not explicitly specified. It denotes WGS84 with the order longitude, latitude.

The following examples show how to specify the spatial coverage by using WKT geometry encoding to represent more polygons and default CRS (CRS84).

DATASET example

```
dct:spatial [ a dct:Location ;
```

```
    locn:geometry "POLYGON(3.053 47.975, 7.24 47.975, 7.24 53.504, 3.053 53.504, 3.053 47.975)"^^gsp:wktLiteral ;
```

```
    locn:geometry "POLYGON(-72.2679021 42.9300036, -72.2830504 42.9263287, -72.2806247 42.9365225, -72.2679021 42.9300036)"^^gsp:wktLiteral ;
```

```
];
```

WEBSERVICE example

```
dct:spatial [ a dct:Location ;
```

```
    locn:geometry "POLYGON(3.053 47.975, 7.24 47.975, 7.24 53.504, 3.053 53.504, 3.053 47.975)"^^gsp:wktLiteral ;
```

```
    locn:geometry "POLYGON(-72.2679021 42.9300036, -72.2830504 42.9263287, -72.2806247 42.9365225, -72.2679021 42.9300036)"^^gsp:wktLiteral ;
```

```
];
```

Please note that the CRS must be always specified when it is different from WGS 84 (EPSG:4326).

The following example shows how to specify the spatial coverage by using WKT geometry encoding to represent points and explicit CRS=EPSG:23030.

```
dct:spatial [ a dct:Location ;
```

```
    locn:geometry "<http://www.opengis.net/def/crs/EPsg/0/23030> POINT(439930.8579 4475096.6375)"^^gsp:wktLiteral ;
```

```
];
```

Spatial metadata for query DDSS

In order to manage, in the appropriate way, the web service parameters that allow to filter DDSS by spatial bounding box, it is necessary to add semantic information in **Operation** entity. This is done by adding **hydra:property** element to the proper parameters, as already explained for the temporal information.

Metadata element	Description	Allowed values
hydra:property	It contains the vocabulary term which indicates the semantic description of parameter.	<ul style="list-style-type: none"> - epos:southernmostLatitude: refers to the lower bound (min) latitude; - epos:northernmostLatitude: refers to the upper bound (max) latitude; - epos:westernmostLongitude: refers to the lower bound (min) longitude;

		- epos:easternmostLongitude: refers to the upper bound (max) longitude;
--	--	---

At present, the web services implement the spatial query by adopting two different approach:

- A.** by using four different parameters which represent south latitude, north latitude, west longitude, east longitude (e.g., minlatitude, maxlatitude, minlongitude, maxlongitude);
- B.** by using a single parameter which contains south latitude, north latitude, west longitude, east longitude listed in a customized order (e.g., bbox=minlat,minlon,maxlat,maxlon).

The web services which adopt the **A approach** only need to add **hydra:property** element for each spatial parameter as shown in the following example.

OPERATION example – A approach

```
<example/Operation/Station> a hydra:Operation;
  hydra:method "GET";
  hydra:returns "application/xml";
  hydra:property [ a hydra:IriTemplate;
    hydra:template
"http://webservices.ingv.it/fdsnws/station/1/query {?starttime, endtime,
network, station, location, channel, minlatitude, maxlatitude,
minlongitude, maxlongitude, level, nodata}"^^xsd:string;

    hydra:mapping[ a hydra:IriTemplateMapping;
      hydra:variable "minlatitude"^^xsd:string;
      hydra:property "epos:southernmostLatitude";
      rdfs:label "Minimum Latitude";
      schema:minValue "-90.0";
      schema:maxValue "90.0";
      hydra:required "true"^^xsd:boolean;
      schema:defaultValue "36.61" ;
    ];
    hydra:mapping[ a hydra:IriTemplateMapping;
      hydra:variable "maxlatitude"^^xsd:string;
```

```

hydra:property "epos:northernmostLatitude";
rdfs:label "Maximum Latitude";
schema:minValue "-90.0";
schema:maxValue "90.0";
hydra:required "true"^^xsd:boolean;
schema:defaultValue "47.11" ;
];
hydra:mapping[ a hydra:IriTemplateMapping;
hydra:variable "minlongitude"^^xsd:string;
hydra:property "epos:westernmostLongitude";
rdfs:label "Minimum Longitude";
schema:minValue "-180.0";
schema:maxValue "180.0";
hydra:required "true"^^xsd:boolean;
schema:defaultValue "6.74" ;
];
hydra:mapping[ a hydra:IriTemplateMapping;
hydra:variable "maxlongitude"^^xsd:string;
hydra:property "epos:easternmostLongitude";
rdfs:label "Maximum Longitude";
schema:minValue "-180.0";
schema:maxValue "180.0";
hydra:required "true"^^xsd:boolean;
schema:defaultValue "18.48" ;
];
];
.

```

The web services which adopt the **B approach** need to:

1. **modify** the URI template by specifying how the single parameter accepts the geographic coordinates (i.e., north, south, west, east).

Looking at the snippet below, it means that the single parameter (e.g., called bbox) shall be shifted out of the curly brackets; then, the geographic coordinates (i.e., north, south, west, east) shall be added to the URI Template using the order with which they are requested by the web service (e.g., {minlatitude, maxlatitude, minlongitude, maxlongitude}). Note that the name of these parameters (hydra:variable) can be chosen arbitrarily because the semantic, added by using hydra:property, allows to manage their in the appropriate way.


```
hydra:template "https://catalog.terradue.com/gep-epos/search{?
format, pt, start, stop, bbox, geom, track}"^^xsd:string;
```

```
hydra:template "https://catalog.terradue.com/gep-epos/search{?
format, pt, start, stop, geom, track}&bbox={minlatitude, maxlatitude,
minlongitude, maxlongitude}"^^xsd:string;
```

2. **remove** the description of the single parameter (e.g., bbox);

```
hydra:mapping[ a
hydra:IriTemplateMapping,
  hydra:variable "bbox"^^xsd:string;
  rdfs:range "xsd:string";
  rdfs:label "Bounding box";
  hydra:required
  "false"^^xsd:boolean;
];
```

3. **add** the description for all spatial parameter defined in the URI template (see point 1).

The following example shows how to modify according to the B approach.

OPERATION example – B approach

```
<example/Operation/Station> a hydra:Operation;
  hydra:method "GET"^^xsd:string;
  hydra:returns "application/atom+xml";
  hydra:property[ a hydra:IriTemplate;
```

```
hydra:template "https://catalog.terradue.com/gep-epos/search{?
format, pt, start, stop, geom, track}&bbox={minlatitude, maxlatitude,
minlongitude, maxlongitude}"^^xsd:string;
```

```
hydra:mapping[ a hydra:IriTemplateMapping;
  hydra:variable "minlatitude"^^xsd:string;
  hydra:property "epos:southernmostLatitude";
  rdfs:label "Minimum Latitude";
  schema:minValue "-90.0";
  schema:maxValue "90.0";
  hydra:required "true"^^xsd:boolean;
  schema:defaultValue "36.61" ;
];
hydra:mapping[ a hydra:IriTemplateMapping;
  hydra:variable "maxlatitude"^^xsd:string;
  hydra:property "epos:northernmostLatitude";
  rdfs:label "Maximum Latitude";
  schema:minValue "-90.0";
  schema:maxValue "90.0";
  hydra:required "true"^^xsd:boolean;
  schema:defaultValue "47.11" ;
];
hydra:mapping[ a hydra:IriTemplateMapping;
  hydra:variable "minlongitude"^^xsd:string;
  hydra:property "epos:westernmostLongitude";
  rdfs:label "Minimum Longitude";
  schema:minValue "-180.0";
  schema:maxValue "180.0";
  hydra:required "true"^^xsd:boolean;
  schema:defaultValue "6.74" ;
];
hydra:mapping[ a hydra:IriTemplateMapping;
  hydra:variable "maxlongitude"^^xsd:string;
  hydra:property "epos:easternmostLongitude";
  rdfs:label "Maximum Longitude";
  schema:minValue "-180.0";
  schema:maxValue "180.0";
  hydra:required "true"^^xsd:boolean;
  schema:defaultValue "18.48" ;
];
];
```

Please note that at this stage it will be considered only the web services that support the spatial query by using a **rectangle**.