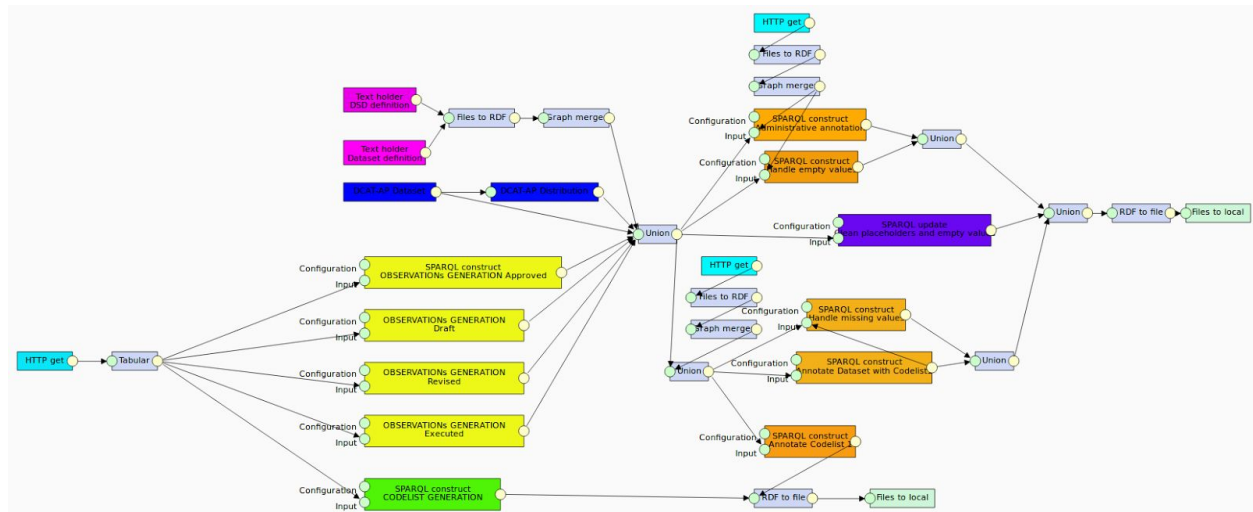


CSV2OBEU RDF Pipeline

Documentation



[Introduction](#)

[Getting familiar with the Environment of LP](#)

[Overview](#)

[Pipelines](#)

[Executions](#)

[Component compatibility](#)

[Component tags](#)

[Resume failed pipeline execution](#)

[Graphical execution overview](#)

[Component debug controls](#)

[Execution detail](#)

[Sharing options](#)

[Pipeline download](#)

[Pipeline import](#)

[Uploading the pipeline into LP-ETL](#)

[Legend](#)

[Editing](#)

[Uploading dataset in raw format](#)

[Configuring Tabular](#)

[Naming conventions](#)

[Changing Column Numbers and prefixes](#)
[Annotating Dataset with External Codelists](#)
[Configuring the Dataset description](#)
[Configuring the Data Structure Definition](#)
[Configuring Metadata](#)

Introduction

Getting familiar with the Environment of LP

Overview

Once you have LinkedPipes ETL [installed and running](#), it is time to explore. Of course, there is not much to see in an empty instance, but let us go through the basics. The image above is what we call a Pipeline. It is a defined data transformation process which consists of interconnected [components](#). You can add a component to a pipeline by clicking in an empty space and selecting the desired component from the list, or by dragging and edge to an empty space. Each component has a name, green input ports and yellow output ports. The ports represent Data Units, which can contain either RDF data or regular files, according to their type. Ports are connected by edges, which indicate the flow of data. Only ports representing Data Units of the same type can be connected. In addition, component configuration can be shared among pipelines using [component templates](#).

Pipelines

Pipelines, defined data transformation processes consisting of interconnected [components](#), are one of the key concepts in LinkedPipes ETL.

[CTIA_1] CTIA Inspections		
[CTIA_1] CTIA Inspections without loaders		
Děčín 2014 - XI		
NKOD extraction		
EU Structural Funds - Recipients Czech Republic		

So far, you can name them, display them, edit them and run them. Once you run a Pipeline, you create its Execution.

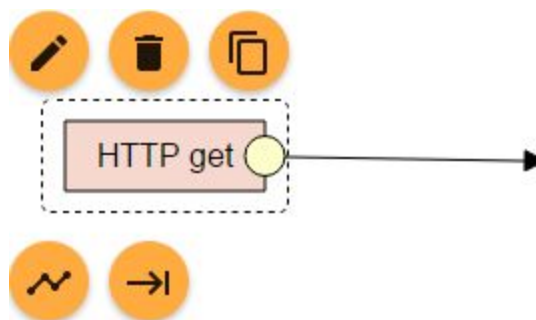
Executions

An Execution is one run of a Pipeline. It can either succeed or fail. On the Executions page, you can see the currently running executions with their progress as well as finished executions.

✓	EU Structural Funds - Recipients Czech Republic 2016-05-16 17:38:08, 00:01:24 Partial execution (debug to: "Tabular") Size: 438.43 mB	▶	🗑	⋮
✓	Municipality of Athens Budget Revenue 2005 working 2016-05-16 17:35:25, 00:00:06 Partial execution (debug to: "uv-t-tabular") Size: 2.83 mB	▶	🗑	⋮
✓	NKOD extraction 2016-05-12 21:22:47, 01:33:56 Size: 471.06 mB	▶	🗑	⋮
!	[CTIA_1] CTIA Inspections 2016-05-11 14:42:35, 00:00:27 Size: 238.68 mB	▶	🗑	⋮

You can see the start and end time of an Execution as and the amount of disk space it takes up. You can free that disk space by using the trash icon. You can also re-run the execution or edit its Pipeline, which is useful for debugging.

Component compatibility

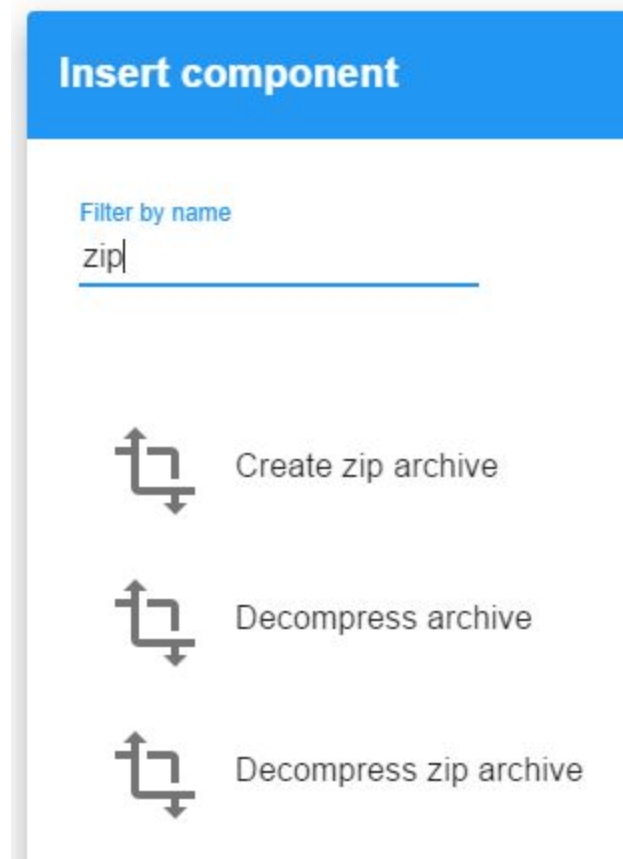


The first component is placed the usual way, just click anywhere on the pipeline and a component list will show up. From the second component on, it gets interesting. Place the next component by dragging and edge from an existing one.

The list of components you see is limited only to those actually compatible with the output of the current component. For [HTTP Get](#), you get only those components that can consume files, for components that produce RDF, only those able to consume RDF are offered.

Component tags

In addition, we tagged the components so that you can find them not only by their name, which sometimes you would have to guess, but also by what they do and by the formats they process. Let's say you want to unzip a file. For that, we have the [Decompress archive](#) component. However, it may be hard to find under that name. Now, you can find it by typing zip.



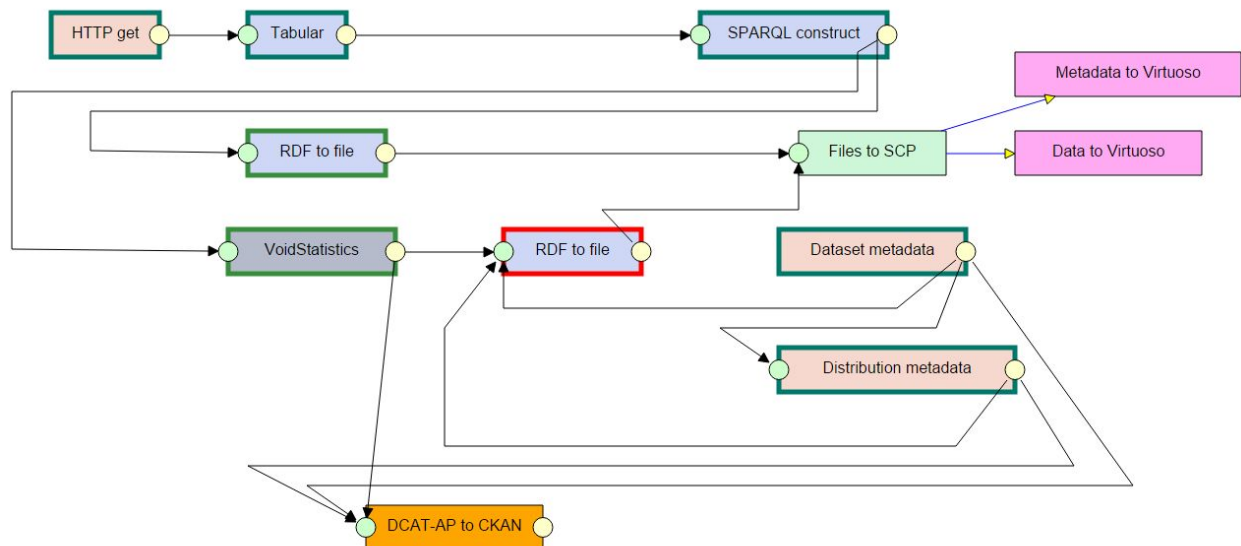
When you execute a pipeline, you can watch it execute in the graphical overview. When a pipeline fails, you can simply fix what was wrong and resume from the point of failure. And when you are developing your query and need to rerun it over and over again until it is perfect, you can do that too.

Resume failed pipeline execution

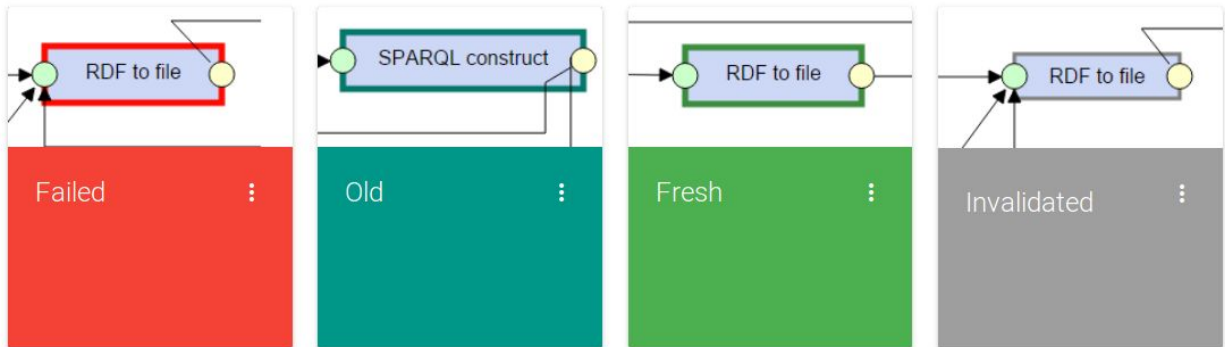
✓	EU Structural Funds - Recipients Czech Republic 2016-05-16 17:38:08, 00:01:24 Partial execution (debug to: "Tabular") Size: 438.43 mB	▶	✕	⋮
✓	Municipality of Athens Budget Revenue 2005 working 2016-05-16 17:35:25, 00:00:06 Partial execution (debug to: "uv-t-tabular") Size: 2.83 mB	▶	✕	⋮
✓	NKOD extraction 2016-05-12 21:22:47, 01:33:56 Size: 471.06 mB	▶	✕	⋮
!	[CTIA_1] CTIA Inspections 2016-05-11 14:42:35, 00:00:27 Size: 238.68 mB	▶	✕	⋮

When your execution fails, you can see that in the execution list. When you click on the execution, the graphical execution overview opens, where you can see what went wrong. By clicking on a data unit (circle on a component) you can browse its contents. By clicking on a component, you can see details about its execution. Now you need to switch to edit mode, fix the component that failed and click on "execute" to resume the execution of the pipeline.

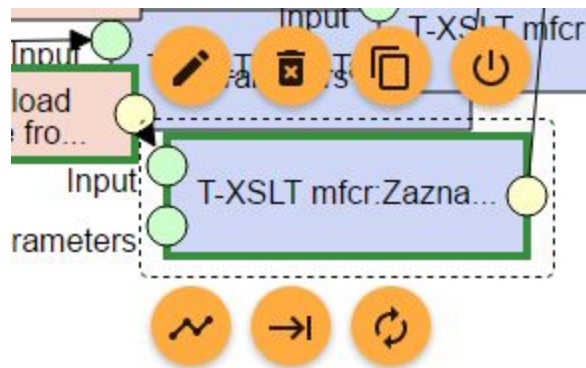
Graphical execution overview






Component state type



Component debug controls




When a component is clicked in the pipeline, its controls show up. The  control invalidates the old and fresh components and all components following it in the execution order.








The  control runs the pipeline up to that component and then stops. The  control disables the component. A disabled component behaves as if it is not in the pipeline when a pipeline is executed. However, it keeps all the configuration and can be enabled again at any time. This is useful for debugging queries and configurations of a single component.

Execution detail

Once an Execution is finished, you may want to inspect it in more detail. One way is browsing the intermediary data passed in the pipeline in the graphical overview. Another way is by

clicking on Execution detail in the  menu. There, you can inspect messages from the execution in the form of a list.

In the Execution detail view, you can see a list of data units attached to executed components. The individual data units are named to reflect their expected content. You can open a data unit and see the files inside via FTP. Soon, we will add the option to directly query RDF data units using SPARQL.

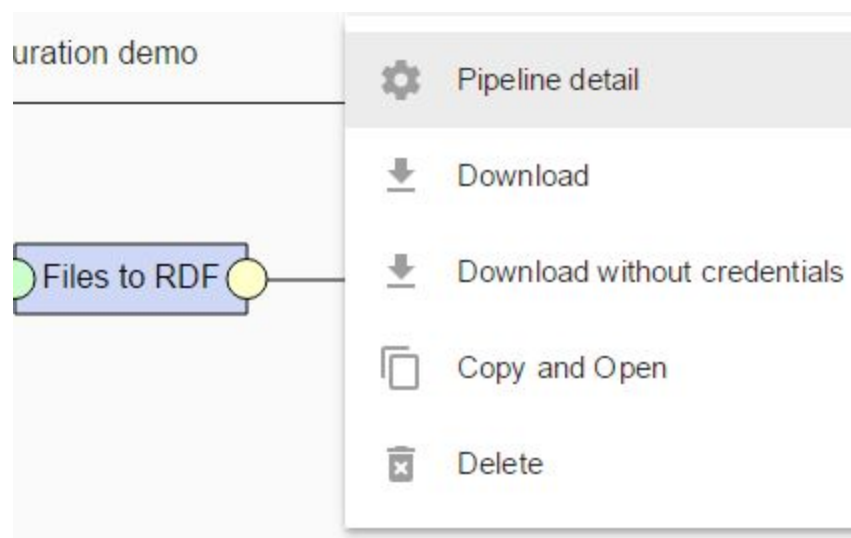
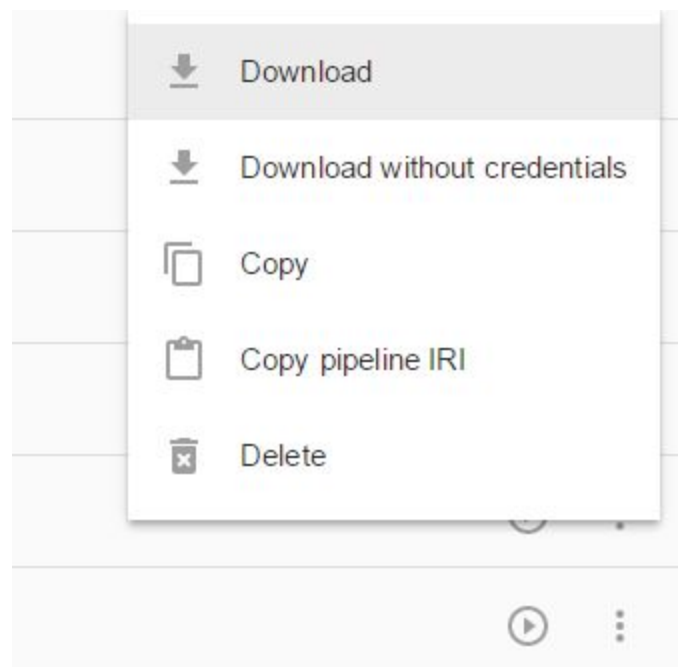
✓	Files to SCP 2016-04-14 09:29:47, 00:00:13	 FILESINPUT
✓	RDF to file 2016-04-14 09:29:39, 00:00:08	 INPUTRDF  OUTPUTFILE
✓	SPARQL construct - Status DCV 2016-04-14 09:27:26, 00:01:17	 INPUTRDF  OUTPUTRDF
✓	SPARQL update 2016-04-14 09:16:08, 00:08:51	 INPUTRDF  OUTPUTRDF

Sharing options

Pipelines can be shared among colleagues and LP-ETL instances easily. They can be downloaded from one instance, shared as files on the web and finally imported to another instance either as a whole pipeline or as a pipeline fragment.

Pipeline download

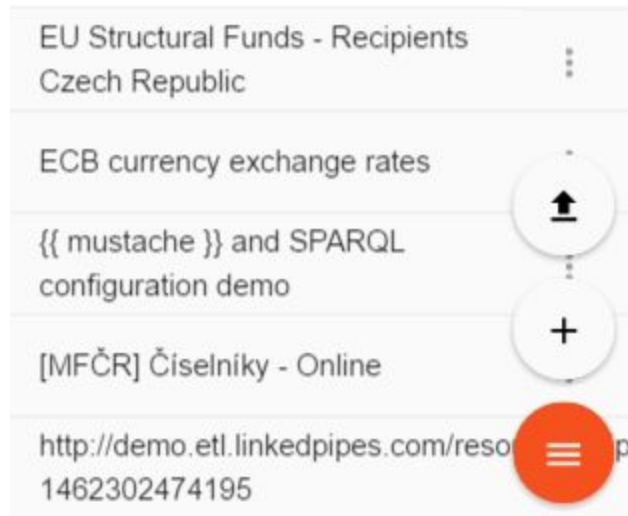
You can download a pipeline as a JSON-LD file from the pipeline list by clicking on the *more_vert* menu. There, you can select either Download or Download without credentials, which removes all potentially sensitive information such as user names, passwords, host names and ports from the downloaded pipeline, so that it can be shared safely. In addition, you can copy the selected pipeline IRI to the clipboard and send the link to someone who can use it to import the pipeline to his LP-ETL instance without the need to download and upload the pipeline files.



The same options for downloading a pipeline are also available from the pipeline editor, again in the *more_vert* menu. The pipeline IRI is available in the pipeline detail dialog.

Pipeline import

Once you get a link to a pipeline or the pipeline itself in the form of a JSON-LD file, you can import it in an LP-ETL instance either as a whole new pipeline, or as a pipeline fragment into an existing pipeline. To import the pipeline as a new pipeline, click on the *menu* menu and select the *file_upload* item.



TEMPLATES PERSONALIZATION

☒ Upload file

☐ From URL

☐ Update existing templates

Selected file:

SELECT FILE

UPLOAD

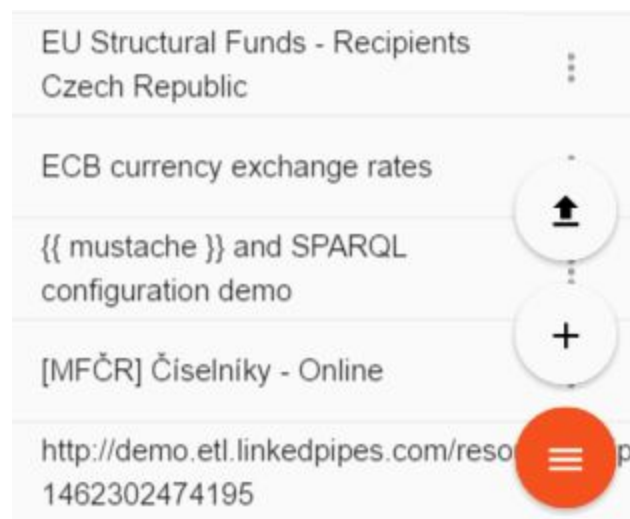
In the pipeline import dialog, you can either upload the pipeline in a JSON-LD file, or provide a URL of an existing pipeline, which can be both uploaded somewhere on the web, or it can be the pipeline IRI from another LP-ETL instance.

Uploading the pipeline into LP-ETL

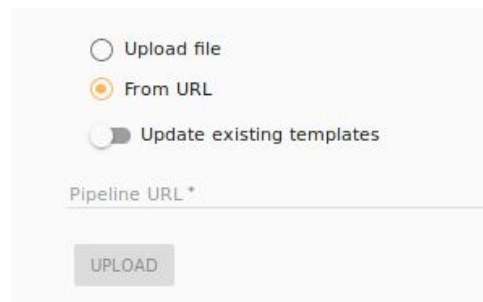
We have created a configurable pipeline in order to transform fiscal data from tabular format (CSV, EXCEL etc) in RDF, using the OpenBudgets.eu Data Model. To start, simply copy the pipeline URL.

<https://raw.githubusercontent.com/skarampatakis/pipeline-fragments/csv2rdf/CSV2RDF/pipeline/CSV2OBEURDF-Template.jsonld>

Then upload the pipeline on your LinkedPipes instance. LinkedPipes can create a new pipeline either by uploading the pipeline itself, or by pointing to a URL where the pipeline is. Point your mouse over the Orange button on the bottom right corner of the screen and click on the Up arrow button.



A menu will be presented where you will select the “From URL” option.



☐ Upload file

☒ From URL

☐ Update existing templates

Pipeline URL *

UPLOAD

On the “Pipeline URL” field, copy the link provided above.

☐ Upload file

☒ From URL

☐ Update existing templates

Pipeline URL *

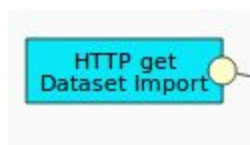
[tps://raw.githubusercontent.com/skarampatakis/pipeline-fragments/csv2rdf/CSV2RDF/pipeline/CSV2OBEURDF-Template.jsonld](https://raw.githubusercontent.com/skarampatakis/pipeline-fragments/csv2rdf/CSV2RDF/pipeline/CSV2OBEURDF-Template.jsonld)

UPLOAD

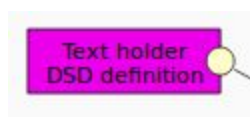
On the next steps we will explain every single aspect of the above procedure.

Legend

In order to have a clear overview of each of the steps required to transform your dataset using this pipeline we have coloured some critical components where you need to edit. Here is the legend of what each color means.



Teal coloured components are used to upload datasets, whether they are the actual datasets or external codelists to be used for annotation purposes. In order to work properly, links should point to downloadable files. You can have them uploaded on a server or on Dropbox like services.



Purple coloured components are used to insert data using simple text files. These components contain DSD configurations and basic dataset descriptions.



Deep blue colored components are used to provide metadata specific information. These are mostly simple forms where you have to edit and provide info. The rest are handled by the components.



Yellow components are used to create the observations triples for each budget phase. These require minor edits, to map columns from the original dataset file to properties of the OBEU Data model. More info of the exact procedure you will find on [this section](#).



Strong green components are used to create codelists that are self included implicitly on your dataset. That means that you should have at least two distinct columns containing a notation and a label for the codelist.

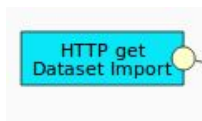


Orange coloured components are used to annotate datasets with codelists. In general you have to change the condition rule (a simple regex), on how the annotator makes the comparison.

Editing

Uploading dataset in raw format

Go to the
will have
directly a



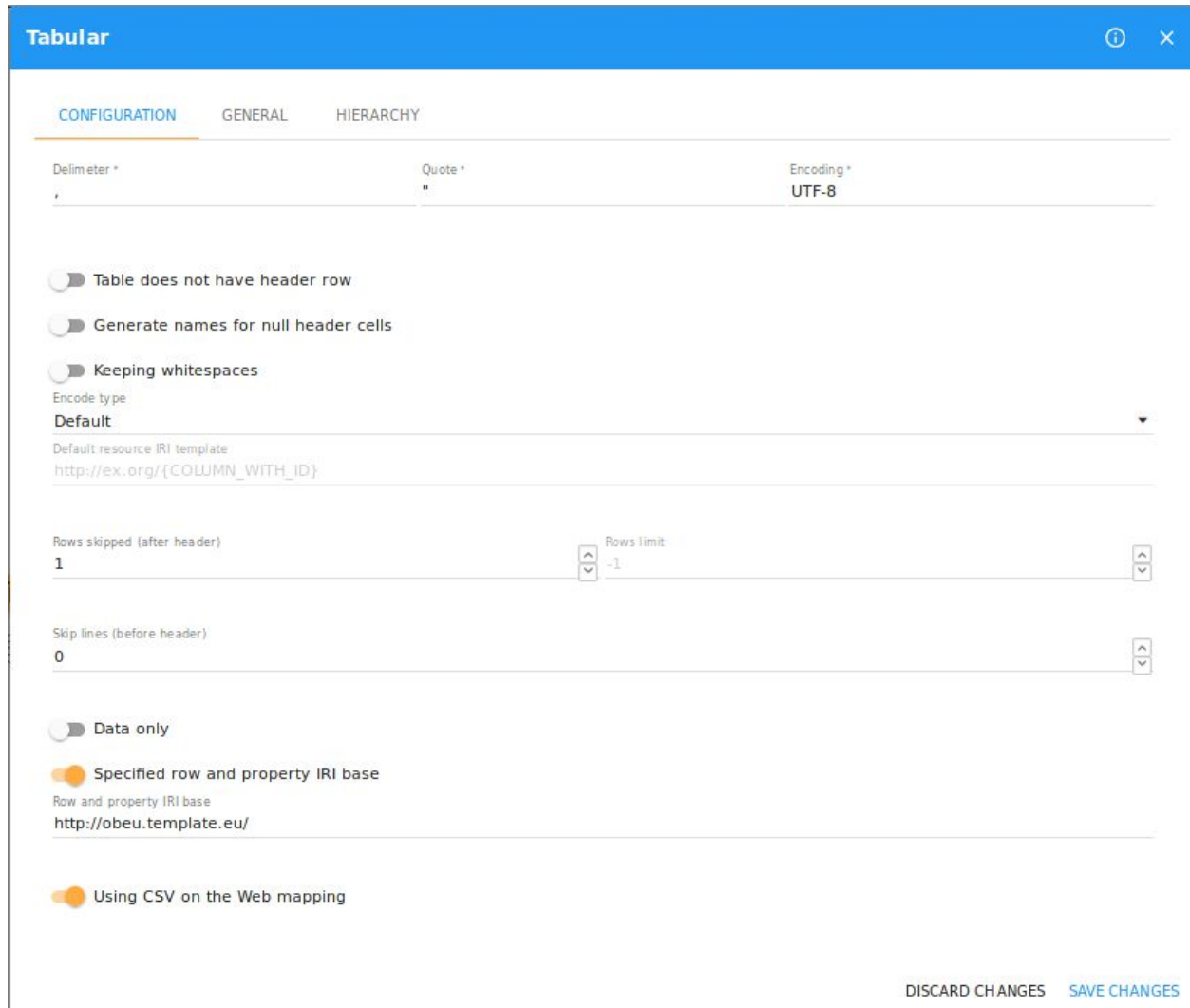
component and click on the edit button once selected. Here you to define the exact URL of your dataset. You can not upload dataset into LP, so your dataset has to be on public webserver or a service like Dropbox. After you copy the URL of your file, paste it at the file URL field

A screenshot of the "HTTP get" configuration form. The form has a blue header bar with the title "HTTP get" and a close button. Below the header, there are four tabs: "CONFIGURATION", "INHERITANCE", "GENERAL", and "HIERARCHY". The "CONFIGURATION" tab is active. The form contains two text input fields: "File URL *" with the value "https://www.dropbox.com/s/w1cegt9f0qsie5v/thessaloniki-2016-revenue.csv?dl=1" and "File name *" with the value "thessaloniki-2016-revenue.csv". There is also a toggle switch labeled "Force to follow redirects" which is currently turned off. At the bottom right, there are two buttons: "DISCARD CHANGES" and "SAVE CHANGES".

In case you use Dropbox for your file uploading(recommended), don't forget to change the "dl" key from "0" to "1", to get the actual file and not a webpage describing the file. You have also to change the filename. Just give the filename of your file and click on "SAVE CHANGES" button.

Configuring Tabular

Configure the "Tabular" component to match your own file configuration.



The screenshot shows the 'Tabular' configuration window with the 'CONFIGURATION' tab selected. The settings are as follows:

- Delimiter ***: ,
- Quote ***: "
- Encoding ***: UTF-8
- Table does not have header row**: ☐
- Generate names for null header cells**: ☐
- Keeping whitespaces**: ☐
- Encode type**: Default
- Default resource IRI template**: `http://ex.org/{COLUMN_WITH_ID}`
- Rows skipped (after header)**: 1
- Rows limit**: -1
- Skip lines (before header)**: 0
- Data only**: ☐
- Specified row and property IRI base**: ☒
Row and property IRI base: `http://obeu.template.eu/`
- Using CSV on the Web mapping**: ☒

At the bottom right, there are buttons for 'DISCARD CHANGES' and 'SAVE CHANGES'.

This should be the delimiter, the quote and the number of rows to skip. Leave the rest as is.

Naming conventions

For OpenBudgets.eu we will use the following RDF prefixes based on a similar approach for the RDF version of SDMX:

- obeu: `<http://data.openbudgets.eu/ontology/>`

- obeu-dsd: <<http://data.openbudgets.eu/resource/dsd/>>
- obeu-dimension: <<http://data.openbudgets.eu/ontology/dsd/dimension/>>
- obeu-measure: <<http://data.openbudgets.eu/ontology/dsd/measure/>>
- obeu-attribute: <<http://data.openbudgets.eu/ontology/dsd/attribute/>>
- obeu-codelist: <<http://data.openbudgets.eu/resource/codelist/>>
- obeu-metadata: <<http://data.openbudgets.eu/ontology/metadata/>>
- obeu-currency: <<http://publications.europa.eu/resource/authority/currency/>>

Before we can continue we have to define the naming conventions to be used. We recommend using the following pattern

http://data.openbudgets.eu/resource/dataset/{continent}/{country}/{division_level}/{authority}/{year}/budget/

On each URL segment use lowercase english alphabet letters. Replace any space character “ ” with hyphen “-”. Use the following table as a guide.

Segment	Description
{continent}	Here goes the name of the continent of the organization to describe. Possible values are “europe”, “north-america”, “south-america”, “africa”, “asia”, “oceania”, “antarctica”
{country}	Use this list to get the official name in english of the country that your organization belongs. http://www.geonames.org/countries/ Remember to use only lowercase english alphabet and hyphens instead of spaces where needed.
{division_level}	Division level reflect the division level of your organization. Possible values are “general-government”, “regions”, “municipalities”
{year}	Use Gregorian calendar year values in “YYYY” format. For instance if the dataset refers to the fiscal year 2017 use this value.

Now it's time to get to the actual conversion!!

Changing Column Numbers and prefixes

Usually, each dataset should contain at least one column with amounts in some budget phase. These get handled in the yellow components. Inside the components there are some configurations that need to be changed. These are hinted by the “#INFO:” tag. On the info tag there is a short explanation of what and how should be changed to cover your needs. Let's start!

1.

```
#INFO: change mydimension to match your custom dimension
prefix mydimension: <http://data.openbudgets.eu/resource/dataset/europe/greece/municipalities/thessaloniki/2016/budget/revenue
/dsd/dimension/>
```

On this configuration you have to define your dimensions prefixes. Follow the naming convention we have described above. Don't forget to change also the operation character segment. Possible values are “revenue” and “expenditure”. Leave the rest as is.

2.

```
WHERE
{
  #INFO: change column_1 with whatever number your administrative classification placeholder exists, if any.
  OPTIONAL { ?line <http://obeu.template.column_1> ?administrative . }
  bind(if(bound(?administrative), ?administrative, "") as ?administrative_placeholder)

  #INFO: change column_2 with whatever number your codelist id exists
  ?line <http://obeu.template.eu/column_2> ?codelist_id .

  #INFO: change column_4 with whatever number your draft amount exists
  ?line <http://obeu.template.eu/column_4> ?value .

  #INFO: change value in bind to reflect your needs. An appropriate name convention would be "http://data.openbudgets.eu/resource
/dataset/{continent}/{country}/{division_level}/{authority}/{year}/budget/"
```

On the above settings we have to do the actual mapping of column values. Change the number after “column_” to reflect your tabular file structure. The “?codelist_id” variable is usually an id column, a classification identity. Then, ?value is the column where the value for a specific budget phase exists. Here we see the mapping for the “draft” budget phase. Change the number to match yours.

3.

```
#INFO: change value in bind to reflect your needs. An appropriate name convention would be "http://data.openbudgets.eu/resource
/dataset/{continent}/{country}/{division_level}/{authority}/{year}/budget/"

bind("http://data.openbudgets.eu/resource/dataset/europe/greece/municipalities/thessaloniki/2016/budget/" as ?iri_part)
```

Repeat what you have done on step 1. This will be used as the observation's IRI.

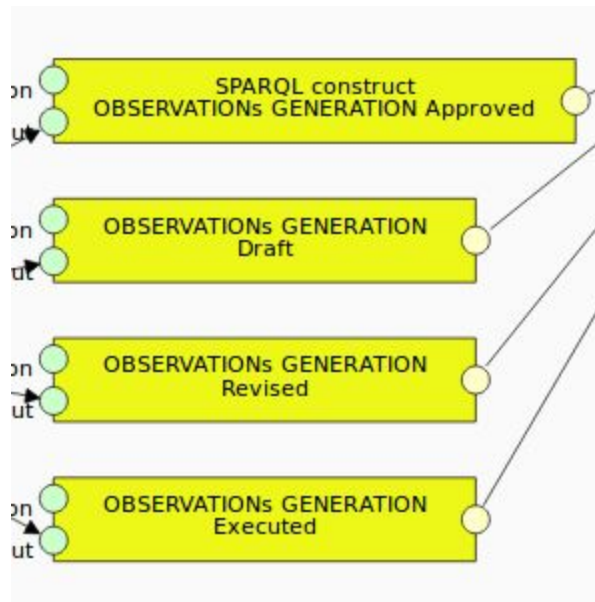
4.

```
# INFO : If "," is used as decimal separator and "." for groups uncomment the following lines
bind(replace(?value, '\\.', ',') as ?temp_value1)
bind(strdt(replace(?temp_value1, '\\,', '\\. '), xsd:decimal) as ?amount)

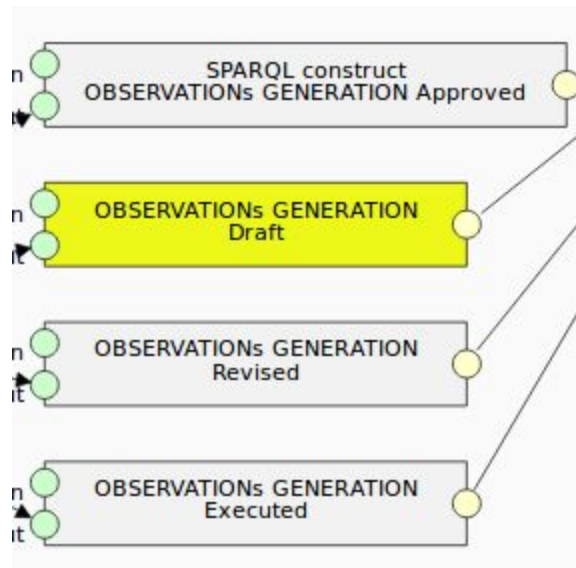
# INFO : Else, uncomment this one
#bind(strdt(?value, xsd:decimal) as ?amount)
```

This is the last configuration you have to do. In some countries the ‘,’ (comma) is used as decimal separator whilst in others is used as grouping character. In the SPARQL specification, only the “.” is used as decimal separator. In order to be in par with the official specification, make the appropriate changes. If your dataset file is using the “.” as decimal separator, uncomment the second bind statement. Else uncomment the first.

Now you have finished modelling your first amount column into RDF DCV observation! Repeat with all different budget phases you may have. In the original pipeline there are 4 different yellow components, one for every standard budget phase in the OBEU Data Model. These are “Draft”, “Approved”, “Revised”, “Executed”.



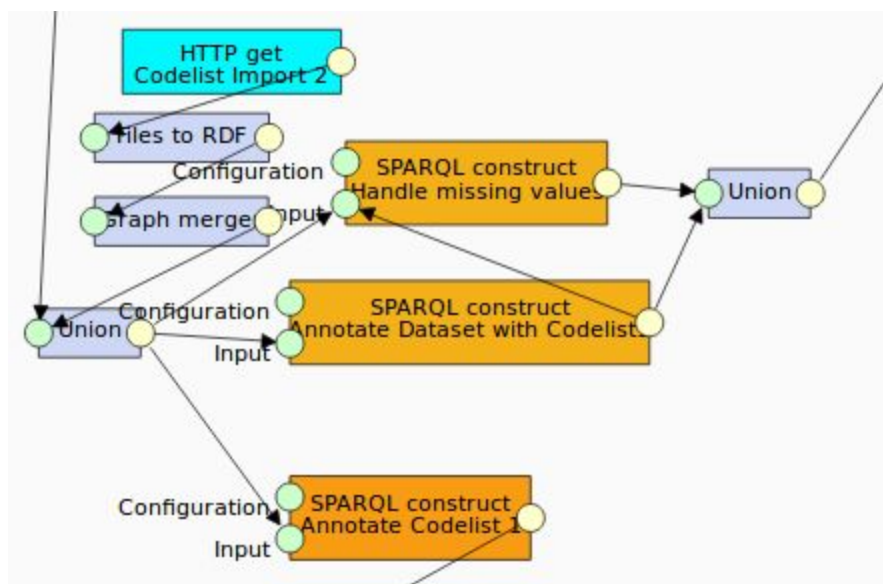
Edit each of these to match your dataset structure. You can disable a budget phase if it is not present on your dataset.



Disabled components will not be evaluated.

Annotating Dataset with External Codelists

There are cases where codelists are not included inline with the actual datasets. In some cases there are hinted in other codelists, sometimes only the notation is mentioned and in other only the label or description. Additionally, some use cases may require annotating the dataset with external resources, based on some criteria. That is the role of the annotating components we have designed within the pipeline.



First we need to define where to get the external codelist from. Change the URL in the “Codelist Import” Component.

Then, change the configuration on the “Annotate Dataset” component. This is a SPARQL construct component.

```
prefix skos: <http://www.w3.org/2004/02/skos/core#>
#INFO: change mydimension to match your custom dimension
prefix mydimension: <http://data.openbudgets.eu/resource/dataset/europe/greece/municipalities/thessaloniki/2016/budget/revenue/dsd/dimension/>
prefix qb: <http://purl.org/linked-data/cube#>

CONSTRUCT {
?observation mydimension:economicClassification ?concept .
}
WHERE
{
?observation a qb:Observation .
?observation mydimension:functionalClassification ?functional .
#INFO: change the two numbers to reflect the start point end length of which to get the part required.
bind (substr(str(?functional), 108,4) as ?code)

?concept a skos:Concept .
?concept skos:notation ?notation .
#INFO: change the filter if required. This will match only equals.
filter (?code = ?notation)
}
```

Next go to the handle missing value component and define the missing value entity.

SPARQL constructⓘ×

CONFIGURATION

INHERITANCE

GENERAL

HIERARCHY

SPARQL CONSTRUCT query

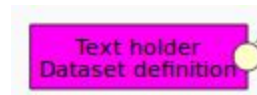
prefix skos: <http://www.w3.org/2004/02/skos/core#>
#INFO: change mydimension to match your custom dimension
prefix mydimension: <http://data.openbudgets.eu/resource/dataset/europe/greece/municipalities/thessaloniki/2016/budget/revenue/dsd/dimension/>
prefix qb: <http://purl.org/linked-data/cube#>

CONSTRUCT {
?observation mydimension:economicClassification <http://data.openbudgets.eu/resource/codelist/kae-ota-esodwn-2014/missing> .
}
WHERE
{
?observation a qb:Observation .
filter not exists {?observation mydimension:economicClassification ?dummy}
}
}

DISCARD CHANGES

SAVE CHANGES

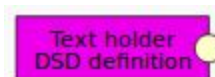
Configuring the Dataset description



Next, we have to create the dataset description. Go to the component and make the appropriate changes.

```
#INFO: change the URL below to match your dataset regarding the naming conventions
<http://data.openbudgets.eu/resource/dataset/europe/greece/municipalities/the
ssaloniki/2016/budget/revenue> a qb:DataSet ;
#INFO: change the URL below to match your dataset regarding the naming conventions
qb:structure
<http://data.openbudgets.eu/resource/dataset/europe/greece/municipalities/the
ssaloniki/2016/budget/revenue/dsd/> ;
#INFO: change the URL below to match your dataset regarding the naming conventions
rdfs:label "Municipality of Thessaloniki, Greece Revenue Budget for the
fiscal year 2016"@en ;
#INFO: change below the currency to match your dataset using ISO 4217 3 letter codes
obeu-attribute:currency obeu-currency:EUR;
#INFO: change below the date to match your dataset
obeu-dimension:fiscalYear <http://reference.data.gov.uk/id/year/2016> ;
#INFO: change below the operation character to match your dataset. Possible values
:"expenditure", "revenue"
obeu-dimension:operationCharacter obeu-operation:revenue;
#INFO: change below the organization to match your own. Use an instance from GeoNames or
DBpedia
obeu-dimension:organization
<http://el.dbpedia.org/resource/Δήμος_Θεσσαλονίκης> ;
dc:publisher <http://openbudgets.eu> .
```

Configuring the Data Structure Definition



Next, we have to create the Data Structure Definition of the Dataset. Go to and make the appropriate changes.

```
# -----DSD-specific namespaces
#INFO: change the URL below to match your dataset regarding the naming conventions
@prefix mydimension:
<http://data.openbudgets.eu/resource/dataset/europe/greece/municipalities/the
ssaloniki/2016/budget/revenue/dsd/dimension/> .
```

```

# -----External Codelists namespaces
@prefix gr-codelist:
<http://data.openbudgets.eu/resource/codelist/europe/greece/> .
@prefix my:
<http://data.openbudgets.eu/resource/dataset/europe/greece/municipalities/the
ssaloniki/2016/budget/revenue/> .
# -----OpenBudgets.eu namespaces -----
@prefix obeu:          <http://data.openbudgets.eu/ontology/> .
@prefix obeu-attribute:
<http://data.openbudgets.eu/ontology/dsd/attribute/> .
@prefix obeu-dimension:
<http://data.openbudgets.eu/ontology/dsd/dimension/>.
@prefix obeu-measure:  <http://data.openbudgets.eu/ontology/dsd/measure/> .
@prefix obeu-budgetphase:
<http://data.openbudgets.eu/resource/codelist/budget-phase/> .
@prefix obeu-codelist: <http://data.openbudgets.eu/resource/codelist/> .
# -----Generic namespaces -----
@prefix qb:            <http://purl.org/linked-data/cube#> .
@prefix rdf:           <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:          <http://www.w3.org/2000/01/rdf-schema#> .
@prefix skos:          <http://www.w3.org/2004/02/skos/core#> .
@prefix xsd:           <http://www.w3.org/2001/XMLSchema#> .

```

```

# DSD

```

```

#INFO: change the URL below to match your dataset regarding the naming conventions
<http://data.openbudgets.eu/resource/dataset/europe/greece/municipalities/the
ssaloniki/2016/budget/revenue/dsd/> a

```

```

    qb:DataStructureDefinition ;
    rdfs:label "Data structure definition for the revenue budget of
Thessaloniki Municipality, Greece for the fiscal year 2016"@en ;
    qb:component
    [ qb:dimension obeu-dimension:organization ;
      qb:componentAttachment qb:DataSet ],
    [ qb:dimension obeu-dimension:operationCharacter ;
      qb:componentAttachment qb:DataSet ],
    [ qb:dimension obeu-dimension:fiscalYear;
      qb:componentAttachment qb:DataSet ],
    [ qb:dimension mydimension:administrativeClassification ],
    [ qb:dimension mydimension:economicClassification ],
    [ qb:dimension mydimension:functionalClassification ],
    [ qb:dimension mydimension:budgetphase],
    [ qb:attribute obeu-attribute:currency ;

```

```
qb:componentRequired "true"^^xsd:boolean ;
qb:componentAttachment qb:DataSet ],
[ qb:measure obeu-measure:amount ] .
```

#dimension definition

```
mydimension:economicClassification a rdf:Property, qb:CodedProperty,
qb:DimensionProperty ;
  rdfs:label "KA of Expenditure"@en ;
  rdfs:comment "Budget expenditure codes (KAE) for greek municipalities"@en
;
  rdfs:subPropertyOf obeu-dimension:economicClassification ;
  qb:codeList gr-codelist:kae-ota-esodwn-2014 ;
  rdfs:range skos:Concept ;
  rdfs:isDefinedBy
<http://data.openbudgets.eu/resource/dataset/europe/greece/municipalities/the
sssaloniki/2016/revenue/dsd> .
```

```
mydimension:administrativeClassification a rdf:Property, qb:CodedProperty,
qb:DimensionProperty ;
  rdfs:label "Municipality Office"@en ;
  rdfs:comment "The organization unit (office) of the municipalities in
Greece that is responsible for the budget amount."@en ;
  rdfs:subPropertyOf obeu-dimension:administrativeClassification ;
  qb:codeList gr-codelist:administrations ;
  rdfs:range skos:Concept ;
  rdfs:isDefinedBy
<http://data.openbudgets.eu/resource/dataset/europe/greece/municipalities/the
sssaloniki/2016/revenue/dsd> .
```

```
mydimension:functionalClassification a rdf:Property, qb:CodedProperty,
qb:DimensionProperty ;
  rdfs:label "Municipality Office - KAE Programme"@en ;
  rdfs:comment "Specific function of expenditure budget per office for
Municipality of Thessaloniki revenue Budget of year 2016"@en ;
  rdfs:subPropertyOf obeu-dimension:functionalClassification ;
  qb:codeList my:codelist ;
  rdfs:range skos:Concept ;
  rdfs:isDefinedBy
<http://data.openbudgets.eu/resource/dataset/europe/greece/municipalities/the
sssaloniki/2016/revenue/dsd> .
```

```
mydimension:budgetphase a rdf:Property, qb:CodedProperty,
qb:DimensionProperty ;
```

```

    rdfs:label "Budget Phase"@en, "Φάση Προϋπολογισμού"@el ;
    rdfs:comment "Budget Phase"@en, "Φάση Προϋπολογισμού"@el ;
    rdfs:subPropertyOf obeu-dimension:budgetPhase ;
    qb:codeList obeu-codelist:budget-phase ;
    rdfs:range skos:Concept ;
    rdfs:isDefinedBy
<http://data.openbudgets.eu/resource/dataset/europe/greece/municipalities/the
sssaloniki/2016/revenue/dsd> .

```

Configuring Metadata

Within the OpenBudgets.eu project we work with data modelled according to the Data Cube Vocabulary (DCV). To comply with DCAT-AP v1.1 and to structure the data and metadata in a uniform way, we follow the following guidelines:

1. The qb:DataSet instance complies with the IRI pattern
<http://data.openbudgets.eu/resource/dataset/DATASETID>
2. The named graph containing the data complies with the same IRI pattern
<http://data.openbudgets.eu/resource/dataset/DATASETID>
3. The dcat:Dataset instance complies with the IRI pattern
<http://data.openbudgets.eu/resource/dataset/DATASETID/metadata>
4. There is a dcterms:subject property linking the dcat:Dataset to the
qb:DataSet instance that it describes.
5. The named graph containing the metadata complies with the same IRI pattern
<http://data.openbudgets.eu/resource/dataset/DATASETID/metadata>
6. The dcat:Distribution instance complies with the IRI pattern
<http://data.openbudgets.eu/resource/dataset/DATASETID/metadata/DISTRIBUTIONID>

Where DISTRIBUTIONID is different for every distribution of this Dataset (i.e. rdf-trig, csv, sparql, etc.)

7. The dataset publisher is the project partner responsible for that dataset.
8. The author of the dataset is the person responsible for that dataset.
9. Access URL and Download URL for file distributions must both point to the data dump.

10. For the SPARQL endpoint distribution, the Access URL points to the endpoint. The additional metadata items are not to be used for this type of distribution.

Note that DCAT-AP v1.1 is supported in LinkedPipes ETL - the tool used to produce RDF datasets in OpenBudgets.eu - by two specific components: DCAT-AP Dataset and DCAT-AP Distribution. Nevertheless, this is an open specification and compliant metadata can be also produced in other ways.