

OpenBudgets.eu: Fighting Corruption with Fiscal Transparency

Project Number: 645833

Start Date of Project: 01.05.2015

Duration: 30 months

Deliverable 2.1

Tools for Semantic Lifting of Multiformat Budgetary Data

Dissemination Level	Public
Due Date of Deliverable	Month 10, 29.02.2016
Actual Submission Date	30.03.2016
Work Package	WP 2, Data Collection and Mining
Task	T 2.1
Type	Demonstrator
Approval Status	Draft
Version	1.0
Number of Pages	31
Filename	Deliverable-2.1-H2020 OpenBudgets.eu

Abstract: This deliverable describes data transformation tools for semantic lifting based on the data model designed in Work Package 1. These tools consists of (i) pipelines developed on general purpose ETL (Extract, Transform, Load) platforms, and (ii) a wizard guiding non-expert users in transformation of budget data to OpenSpending's FDP data format. A new user-friendly RDF transformation wizard for the OBEU platform is conceptualized and currently under development.

The information in this document reflects only the author's views and the European Community is not liable for any use that may be made of the information contained therein. The information in this document is provided "as is" without guarantee or warranty of any kind, express or implied, including but not limited to the fitness of the information for a particular purpose. The user thereof uses the information at his/ her sole risk and liability.



History

Version	Date	Reason	Revised by
0.1	14.03.2016	First revision	Fabrizio Orlandi
0.2	16.03.2016	Second revision	Jakub Klímek
1.0	29.03.2016	Final version	Christiane Engels, Fathoni Musyaffa

Author List

Organisation	Name	Contact Information
FhG	Christiane Engels	christiane.engels@iais.fraunhofer.de
UBONN	Fathoni Musyaffa	musyaffa@cs.uni-bonn.de
UBONN	Tiansi Dong	tdong@uni-bonn.de
UEP	Jakub Klímek	klimek@ksi.mff.cuni.cz
UEP	Jindřich Mynarz	jindrich.mynarz@vse.cz
FhG	Fabrizio Orlandi	orlandi@iai.uni-bonn.de
UBONN/FhG	Sören Auer	auer@cs.uni-bonn.de

Executive Summary

This deliverable describes data transformation tools, both having been developed and being under construction, for semantic lifting based on the predefined OBEU data model detailed in Deliverables 1.2, 1.3, 1.4. Semantic Lifting aims at adding 'meaning' or extra meta (semantics) to existing structured/semi-structured data following Linked Data principles and standard Semantic Web technologies. The aim is to be able to load and transform budget data, on the OpenBudgets.eu platform, from different source data formats to the RDF-based target data format developed in WP1. In order to do this, specific data transformation pipelines have been developed using the UnifiedViews and LinkedPipes ETL platforms. Moreover, a data transformation wizard is currently being deployed directly on the OpenSpending platform. A novel RDF ETL wizard is conceptualized and under development.

The general purpose RDF transformation platforms, UnifiedViews¹, and its successor LinkedPipes ETL², are used as the base for the development of tools specific for OBEU data transformation. Data transformation pipelines have been developed, which can transform large datasets into RDF format by running them at one of the two platforms. The creation of such pipelines which are dataset specific demands expert knowledge on the Semantic Web.

In order to allow users to upload datasets into the OBEU platform without expert knowledge in semantic web we aim at using a wizard. Throughout a few simple user interaction steps all information needed for a correct data transformation will be collected.

A data transformation wizard into the OpenSpending's Fiscal Data Package (FDP) format has been developed for data import on the OpenSpending platform. It provides semantic lifting by adding meta-data to CSV datasets yet without incorporating standard Linked Data principles.

Following the same concept of the OpenSpending wizard for FDP data, another RDF data transformation wizard has been conceptualized and is currently under construction. This wizard collects favorite features of pipelines developed at UnifiedViews (as well as LinkedPipes ETL) and the wizard running at the OpenSpending platform.

¹ <http://unifiedviews.eu>

² <http://etl.linkedpipes.com>

Abbreviations and Acronyms

UV	UnifiedViews
LP-ETL	LinkedPipes ETL
OS	OpenSpending
ETL	Extract, Transform, Load
FDP	Fiscal Data Package
DSD	Data Structure Definition
OBEU	OpenBudgets.eu
RDF	Resource Description Framework
ESIF	European Structural and Investment Funds

Table of Contents

[1 Introduction](#)

[2 General Architecture of RDF Data Transformation](#)

[3 Pipelines in UnifiedViews and LinkedPipes ETL platforms](#)

[3.1 Pipelines developed using the UnifiedViews Platform](#)

[3.1.1. ESIF 2014-2020 CSV Pipeline](#)

[3.1.2. EU Budget 2014 XML Pipeline](#)

[3.2 Pipelines developed using the LinkedPipes ETL Platform](#)

[4 The FDP Data Transformation Wizard](#)

[Step 1 Providing a CSV dataset](#)

[Step 2 Describing the dataset](#)

[Step 3 Providing metadata information](#)

[Step 4 Confirming and Downloading](#)

[5 Towards An RDF Data Transformation Wizard](#)

[Step 1 Load a raw dataset.](#)

[Step 2 Map Columns to OBEU Components](#)

[Step 3 Map to existing code lists](#)

[Step 4 Check and Save the Result](#)

[Step 5 Push to RDF Triple Store](#)

[6 Conclusion and Future Work](#)

[7 References](#)

List of Figures

Figure 1. General Architecture of RDF data transformation.....	8
Figure 2. A graphical user interaction layer is added to the general architecture.....	9
Figure 3. ESIF datasets transformation pipeline.....	11
Figure 4. Mapping columns into semantic properties.....	12
Figure 5. Construct query to form IRIs in some properties.....	13
Figure 6. Attaching additional information for the whole dataset.....	14
Figure 7. Attaching DSD into pipeline.....	15
Figure 8. Pipeline to extract functional classification code list from ESIF dataset.....	16
Figure 9. Transformation of EU Budgets 2014 XML dataset into OBEU RDF.....	18
Figure 10. LinkedPipes ETL Pipeline for ESIF 2014-2012 Dataset	19
Figure 11. Visual debugging in LinkedPipes-ETL.....	19
Figure 12. LinkedPipes ETL Pipeline for EU Budget 2014 Dataset.....	20
Figure 13. The start page of the wizard.....	21
Figure 14. The system identified errors in a CSV file, and asked users to view.....	22
Figure 15. Error report appears during the data providing step.....	22
Figure 16. The structure of an uploaded CSV file is shown with 3 rows as sample.....	23
Figure 17. An interface to describe column information.....	24
Figure 18. Candidate data types of columns.....	24
Figure 19. Candidate concepts are inferred from the sample value of the column.....	24
Figure 20. User interface for Metadata.....	25
Figure 21. Confirmation/Download of a generated meta dataset.....	25
Figure 22. Initial user interface for data upload.....	27
Figure 23. Concept of a wizard collection mapping information.....	28
Figure 24. Interface for Code List Extraction.....	29
Figure 25. Mapping Summary for Checking and Confirmation.....	30

List of Tables

Table 1. List of required DPUs to transform CSV to RDF format	10
Table 2. List of required DPUs to transform XML to RDF format.....	17

1 Introduction

This deliverable reports data transformation tools for *Semantic Lifting of Multiformat Financial Data* (Task 2.1). We start with testing two RDF conversion tools developed by the Charles University in Prague and UEP (OBEU partner): UnifiedViews and its successor LinkedPipes ETL. Both are open source tools developed for defining, executing, monitoring, scheduling, and sharing RDF data processing. Both provide graphical user interfaces to perform the ETL tasks, including the process of administration, debugging and monitoring. UnifiedViews has reached version 2.3.0 and is relatively mature in terms of stability.

UnifiedViews is a platform for general purpose of RDF transformation. A ETL data transformation tool is developed by creating *Data Processing Units* (DPU) as plugins of the platform. LinkedPipes ETL is the successor of UnifiedViews, developed based on experience with UnifiedViews, following the same paradigm as UnifiedViews. At the moment of writing this deliverable, it is still in an early developing stage, so DPU availability has not been as extensive as in UnifiedViews. Some DPUs in LinkedPipes are designed differently to the DPUs developed in UnifiedViews due to recently evolved standards such as CSV on the Web³.

Using these two platforms, we already developed over 40 pipelines (data transformation tools) which effectively transformed OBEU financial datasets in heterogeneous formats (e.g., XML, CSV) into the RDF format based on the data model and code lists defined in WP1.

However, there are some concerns regarding the use of UnifiedViews. First relating to performance, and second, relating to the extendability of the user interface. Though LinkedPipes ETL has a potential to solve these concerns, both of them require users to have some expertise in semantic web, for example, both might require users to be able to write SPARQL statements manually.

On other hand, a data upload wizard has been developed and tested. This wizard for OpenSpending promotes CSV dataset into FDP (Financial Data Package) format with user-friendly interfaces, and tested at the platform. This wizard does not require users to have technical expert knowledge.

It is desirable to have an RDF data transformation tool which is both powerful and not requiring expert knowledge in the field of semantic web. To this end, an RDF data transformation wizard is conceptualized and under construction. This wizard aims at transforming tabular-structured datasets, i.e. CSV, and tree-structured datasets, i.e., XML, into the OBEU RDF data format. The main idea of this wizard is that DSD (Data Structure Definition) files and code list files of the input datasets will be generated by mapping columns and data cells into predefined terms of the data model. Such a mapping is realized by selecting terms in data model defined in WP1 and choosing corresponding concepts in the dataset. After that, pipelines including SPARQL queries will be generated and executed by the backend of the wizard to transform the input datasets into the RDF format.

The rest of the deliverable is structured as follows: Section 2 presents the high-level architecture of RDF data transformation tools; Section 3 describes the pipelines developed at

³ https://www.w3.org/standards/techs/csv#w3c_all

UnifiedViews and LinkedPipes ETL platforms; Section 4 illustrates the data transformation wizard developed and equipped at the OpenSpending platform; and Section 5 conceptualizes a new RDF data transformation wizard, which is now under construction.

2 General Architecture of RDF Data Transformation

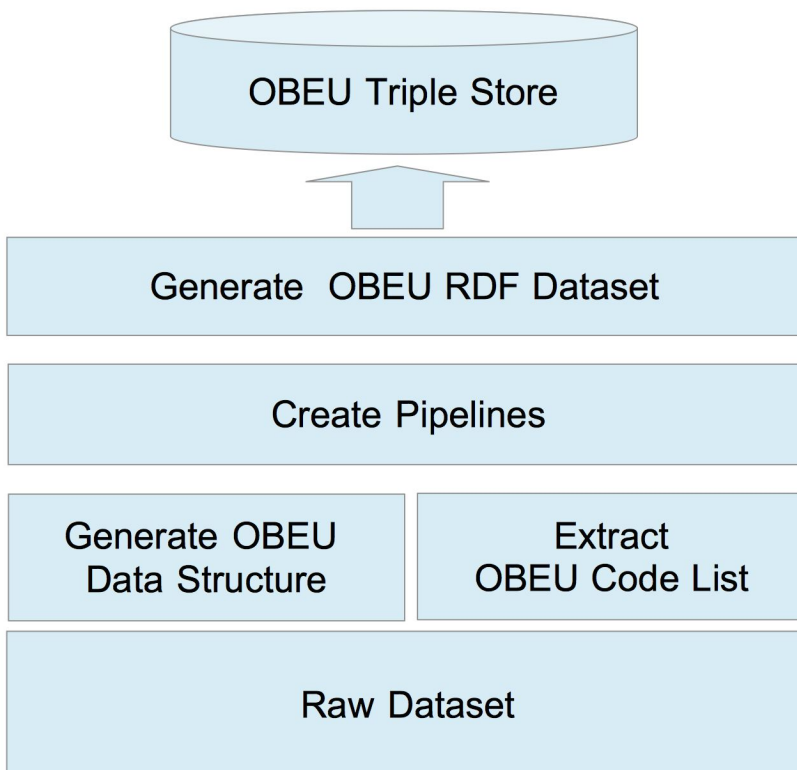


Figure 1. General Architecture of RDF data transformation

The general architecture RDF data transformation process is illustrated in Figure 1. From a raw dataset, we first generate an OBEU data structure definition, and code lists, if included in the dataset. Then the pipeline to transform the raw dataset into an RDF dataset is created. The OBEU data structure file, the code list files, and the transformed RDF dataset are files in RDF format, and will be pushed to the OBEU triple store.

At the UnifiedViews platform, users need to explicitly provide the data structure definition file, and write SPARQL statements to add meta-data information and to modify the transformed RDF dataset if necessary. We will explain this in Section 3 in detail.

For the OBEU platform, a user interaction layer is added as shown in Figure 2 that turns the ETL tool into a wizard, which will be described in detail in Section 5.

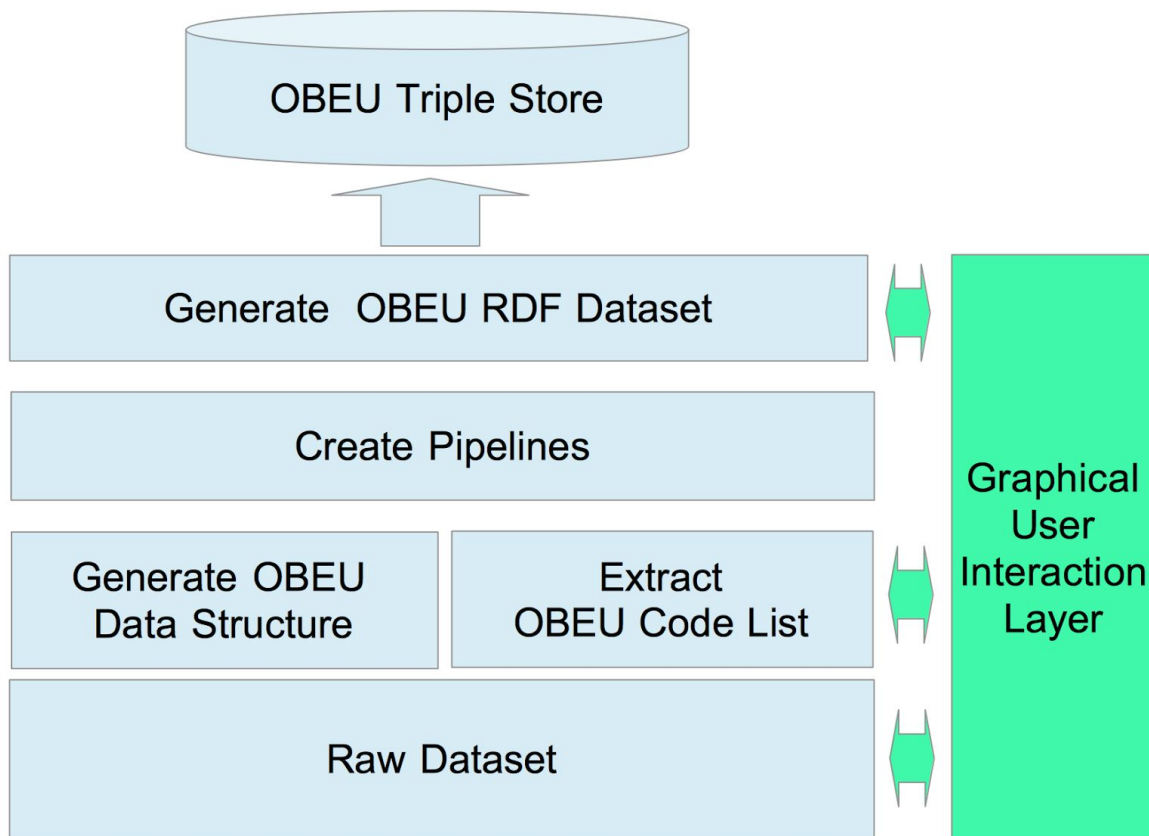


Figure 2. A graphical user interaction layer is added to the general architecture

3 Pipelines in UnifiedViews and LinkedPipes ETL platforms

Both utilized tools, UnifiedViews⁴ and LinkedPipes ETL⁵, have installation instruction on their web pages. UnifiedViews does not include the DPUs upon installation, but its installation guide has extra instructions how to install UnifiedViews DPUs. LinkedPipes ETL includes the basic DPUs after the installation.

Financial datasets have different structures and formats. Therefore, customized transformation pipelines, which consist of DPUs and their interactions, are needed to retrieve the contained financial information from different datasets. In general, ETL pipelines for semantic lifting consist of several steps as follows: (1) downloading the dataset, (2) defining dataset properties (e.g. which fields from raw dataset consist of dimensions and measures) and data structure definition, (3) converting from its native format into RDF, (4) updating the RDF graph, (5) extracting code lists, (6) transforming the RDF graph into files and/or uploading the RDF graph into a triplestore and (7) generating metadata and storing it in the triplestore and a data catalog such as CKAN.

⁴ <https://grips.semantic-web.at/display/UDDOC/Installation+Guide>

⁵ <http://etl.linkedpipes.com/>

Previous deliverables in the OBEU project are related to this document. Deliverable D1.2 (Klímek et al. 2015a) and Deliverable D1.3 (Klímek et al. 2015b) elaborate on how both budget and spending datasets can be defined via *Data Structure Definition* (DSD) files. Deliverable D1.4 (Dudáš et al. 2015) provides an RDF semantic vocabulary for OBEU datasets. Readers are referred to these documents for further explanation of DSDs and vocabularies for OBEU.

3.1 Pipelines developed using the UnifiedViews Platform

We have developed more than 40 transformation pipelines to transform various data formats from several sources using the UnifiedViews platform. Two examples of transformation using UnifiedViews from these datasets are explained in this section. The first example is ESIF 2014 dataset transformation from CSV format into OBEU RDF format. The second example explains transformation of EU Budget 2014 dataset from XML format into OBEU RDF format.

3.1.1. ESIF 2014-2020 CSV Pipeline

In this section, an example of pipeline construction for transforming CSV datasets into RDF is described. The ESIF dataset contains programs funded by five European Structural and Investment Funds (ESIF)⁶. This dataset is available in several formats on the EU data portal⁷.

As the UnifiedViews' default installation has no *Data Processing Unit* (DPU), we need to install the required DPUs first in order to perform a transformation task in UnifiedViews. There are several DPUs involved in constructing the pipeline for transforming the ESIF CSV dataset into the OBEU RDF format. The list of required DPUs for ESIF tabular data transformation are provided in Table 1.

DPU	Functionality
uv-e-filesDownload	Downloads files from external sources into the UnifiedViews platform
uv-t-tabular	Maps table into RDF
e-TextHolder	Stores text files, can be used to hold Data Structure Definition
uv-t-filesToRdf	Converts RDF file to RDF in-memory model
uv-t-sparqlConstruct	Provides a way to execute a SPARQL Construct query
uv-t-sparqlUpdate	Provides a way to execute a SPARQL Update query
uv-t-graphMerger	Merges RDF graphs
uv-t-rdfToFiles	Serializes an RDF from graph into a file

⁶ http://ec.europa.eu/contracts_grants/funds_en.htm

⁷ <https://cohesiondata.ec.europa.eu/dataset/ESIF-FINANCE-DETAILS/e4v6-qrrq>

Table 1. List of required DPUs to transform CSV to RDF format

The pipeline flow in Figure 3 describes the conversion of the ESIF 2014-2020 dataset for the multi annual framework 2014 - 2020⁸. The DPU `uv-e-filesDownload` is utilized to download the datasets. In this DPU, a direct link to the dataset in CSV format is provided. The downloaded dataset is then converted from tabular CSV format into RDF using the DPU `uv-t-tabular`. In this step the tabular transformer maps the necessary columns in the dataset with semantic properties. The user provides the mapping using the interface of the DPU `uv-t-tabular`, as shown in Figure 4.

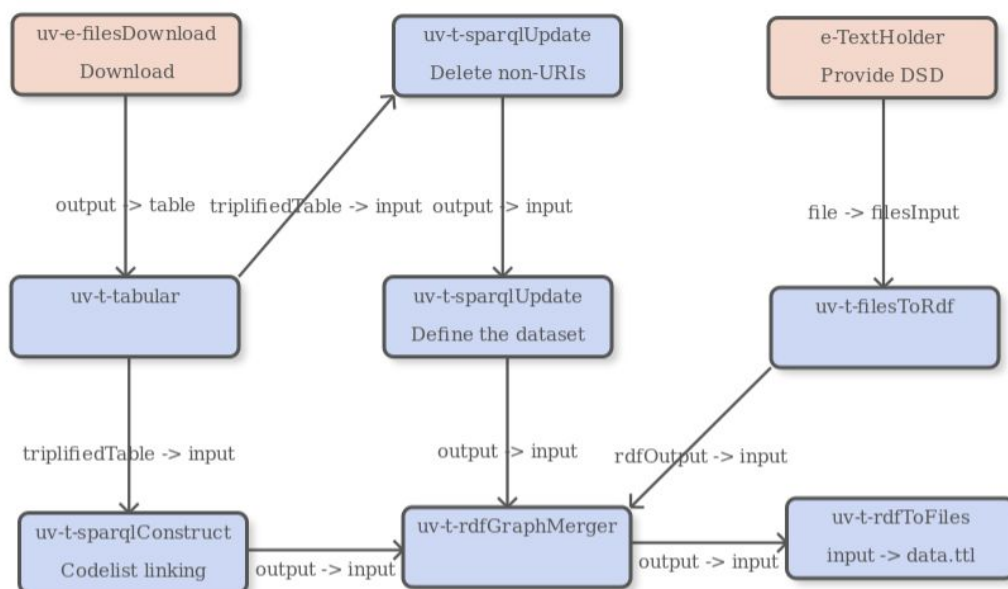


Figure 3. ESIF datasets transformation pipeline.

⁸ http://ec.europa.eu/budget/mff/index_en.cfm

uv-t-tabular detail ✖

Name

Parent

Description

Use custom description Use template configuration

DPU configuration

Fault tolerance

About

Mapping Import/Export

Ignore missing columns

Mapping

Simple

Advanced - experimental functionality!

Xls mapping

Column name	Output type	Language	Use Dbf types	Property URI
<input type="text" value="MS"/>	<input type="text" value="Auto"/>	<input type="text"/>	<input type="checkbox"/>	<input type="text" value="http://data.openbudgets.eu/ontology/dsd/dimension/administrative/"/>
<input type="text" value="TO"/>	<input type="text" value="String"/>	<input type="text"/>	<input type="checkbox"/>	<input type="text" value="http://data.openbudgets.eu/ontology/dsd/dimension/functionalClas"/>
<input type="text" value="Fund"/>	<input type="text" value="String"/>	<input type="text"/>	<input type="checkbox"/>	<input type="text" value="http://data.openbudgets.eu/ontology/dsd/dimension/fund"/>
<input type="text" value="National Amount"/>	<input type="text" value="Auto"/>	<input type="text"/>	<input type="checkbox"/>	<input type="text" value="http://data.openbudgets.eu/ontology/dsd/ESIF-2014-2020/measure"/>
<input type="text" value="Total Amount"/>	<input type="text" value="Auto"/>	<input type="text"/>	<input type="checkbox"/>	<input type="text" value="http://data.openbudgets.eu/ontology/dsd/ESIF-2014-2020/measure"/>
<input type="text" value="EU Amount"/>	<input type="text" value="Auto"/>	<input type="text"/>	<input type="checkbox"/>	<input type="text" value="http://data.openbudgets.eu/ontology/dsd/ESIF-2014-2020/measure"/>

Figure 4. Mapping columns into semantic properties.

According to the provided mapping, the dataset is transformed to RDF. However, the result has to be improved. For example, it is required to set the value of the funds property to “<http://data.openbudgets.eu/resource/codelist/eu-funds/esf>” instead of “ESF”⁹ since ideally in RDF format, the value of dimension properties should be resources, i.e. non-literals, which is not the case when we transform the dataset from its native formats. Therefore we need to transform the values into referents by using SPARQL queries on the graph that has created using the `uv-t-tabular` DPU. To do this, we utilize the transformer DPUs `uv-t-sparqlConstruct` and `uv-t-sparqlUpdate`. The DPU `uv-t-sparqlConstruct` is used for constructing new triples from generated RDF, such as constructing new IRIs for *funds*, *administrative classification* and *functional classification* from the string literals provided in the dataset. The screenshot in Figure 5 provides a SPARQL query to construct IRIs for those dimensions from the corresponding literals in the dataset by adding the respective prefix.

⁹ European Social Fund, <http://ec.europa.eu/esf/>

uv-t-sparqlConstruct detail ✖

Name

Parent

Description

Use custom description Use template configuration

DPU configuration

Fault tolerance

About

Per-graph execution

SPARQL construct query *

```

PREFIX obeu-dimension: <http://data.openbudgets.eu/ontology/dsd/dimension/>
PREFIX qb: <http://purl.org/linked-data/cube#>

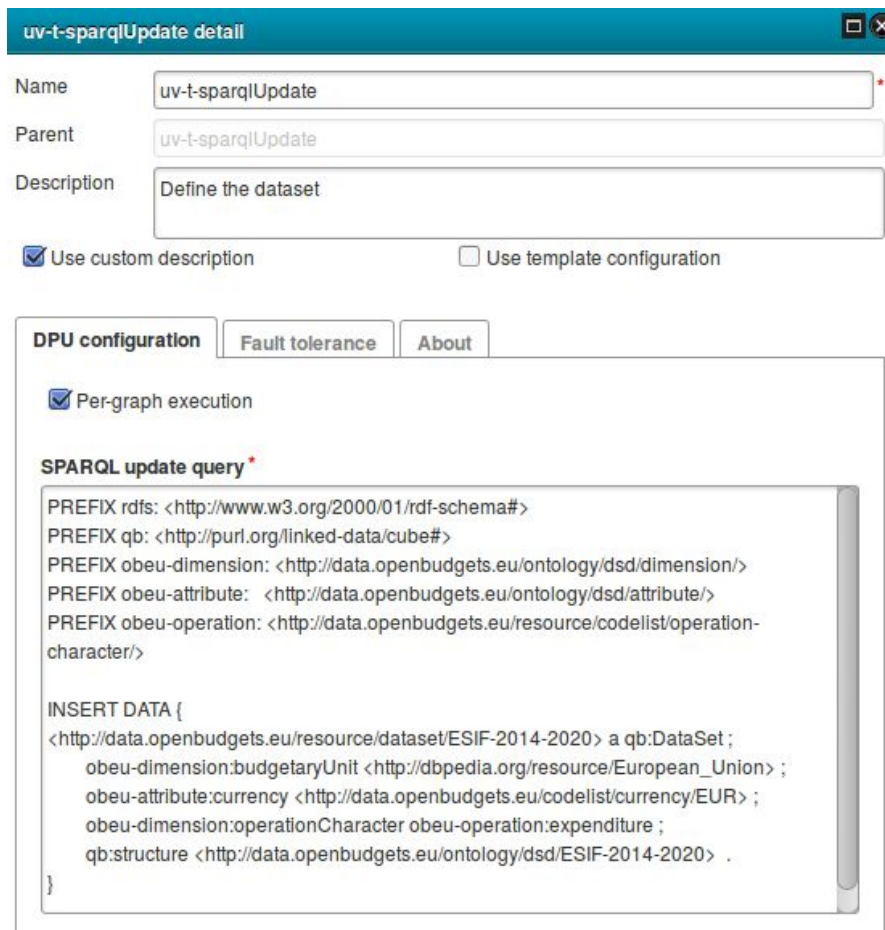
CONSTRUCT { ?s1 obeu-dimension:fund ?uri1.
             ?s1 obeu-dimension:functionalClassification ?uri2.
             ?s1 obeu-dimension:administrativeClassification ?uri3.
             ?s1 qb:dataSet <http://data.openbudgets.eu/resource/dataset/ESIF-2014-2020> .}

WHERE {
  ?s1 obeu-dimension:fund ?o1 .
  ?s1 obeu-dimension:functionalClassification ?o2 .
  ?s1 obeu-dimension:administrativeClassification ?o3 .
  BIND(URI(CONCAT("http://data.openbudgets.eu/resource/ESIF-2014-2020/codelist/eu-funds/",LCASE(?o1))) as ?uri1).
  BIND(URI(CONCAT("http://data.openbudgets.eu/resource/ESIF-2014-2020/codelist/function/",LCASE(?o2))) as ?uri2).
  BIND(URI(CONCAT("http://data.openbudgets.eu/resource/ESIF-2014-2020/codelist/MS",?o3)) as ?uri3).
}

```

Figure 5. Construct query to form IRIs in some properties.

The DPU `uv-t-sparqlUpdate` is required to insert information about the dataset. We need to provide a link to the DSD and specify the property values which are valid for the whole dataset, i.e. that the dataset amount measures are provided in *Euro* currency, the *European Union* is the budgetary unit and the operation character is *expenditure* in this case. The information is *attached* to the dataset instead of a single observation (or row in tabular terms) via a SPARQL query. A screenshot is provided in Figure 6.



uv-t-sparqlUpdate detail

Name

Parent

Description

Use custom description Use template configuration

DPU configuration Fault tolerance About

Per-graph execution

SPARQL update query

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX qb: <http://purl.org/linked-data/cube#>
PREFIX obeu-dimension: <http://data.openbudgets.eu/ontology/dsd/dimension/>
PREFIX obeu-attribute: <http://data.openbudgets.eu/ontology/dsd/attribute/>
PREFIX obeu-operation: <http://data.openbudgets.eu/resource/codelist/operation-character/>

INSERT DATA {
<http://data.openbudgets.eu/resource/dataset/ESIF-2014-2020> a qb:DataSet ;
  obeu-dimension:budgetaryUnit <http://dbpedia.org/resource/European_Union> ;
  obeu-attribute:currency <http://data.openbudgets.eu/codelist/currency/EUR> ;
  obeu-dimension:operationCharacter obeu-operation:expenditure ;
  qb:structure <http://data.openbudgets.eu/ontology/dsd/ESIF-2014-2020> .
}
```

Figure 6. Attaching additional information for the whole dataset.

Meanwhile, the data structure definition (DSD) which has been created separately is imported into UnifiedViews using the `e-TextHolder` DPU, as shown in Figure 7. The DSD is necessary to provide structural information regarding the dataset. The DPU `uv-t-filesToRdf` converts the DSD into RDF, which will be merged with the output of `uv-t-tabular` by the `uv-t-rdfMerger` DPU. Finally, the DPU `uv-t-rdfToFiles` stores the merged RDF graph containing both DSD and dataset into an RDF serialization format.

e-TextHolder detail
✖

Name

Parent

Description

Use custom description Use template configuration

DPU configuration

Fault tolerance

About

Output file name:

File's content:

```
<http://example.openbudgets.eu/ontology/dsd/ESIF-2014-2020> a qb:DataStructureDefinition ;
rdfs:label "Data structure definition for the European Structural and Investment Funds of the years 2014-2020"@en ;
qb:component [ qb:dimension obeu-dimension:budgetaryUnit ;
qb:componentAttachment qb:DataSet ],
[ qb:dimension obeu-dimension:budgetPhase ;
qb:componentAttachment qb:DataSet ],
[ qb:dimension obeu-dimension:operationCharacter ;
qb:componentAttachment qb:DataSet ],
[ qb:dimension obeu-dimension:fiscalPeriod ;
qb:componentAttachment qb:DataSet ],
```

Figure 7. Inserting DSD into pipeline.

Financial datasets usually contain code lists. These code lists may already be available due to previous extraction and transformation steps¹⁰ (Ioannidis et al. 2015), but otherwise should be extracted at the beginning of the ETL process. The ESIF dataset contains code lists that are not yet available from other datasets, therefore, we need to extract these code lists from the raw ESIF dataset. The code lists in the ESIF dataset include EU funded sub-programs along with their labels, EU subprogram objectives for functional classification, and member states for administrative classification. The extracted code lists are then interlinked with other code lists containing similar concepts. This interlinking, which later provides better analytics features among the datasets, is also part of the OBEU project and available as Deliverable D1.9 (Ioannidis et al. 2016). The code lists are available both from Github¹¹ or readily available externally, such as one provided by the EU publication office¹².

To extract the ESIF code lists, separate pipelines are developed. An example pipeline for extracting the functional classification is given in the Figure 8.

¹⁰ <https://github.com/openbudgets/Code-lists>

¹¹ <https://github.com/openbudgets/linksets/>

¹² <http://publications.europa.eu/mdr/authority/currency/index.html>

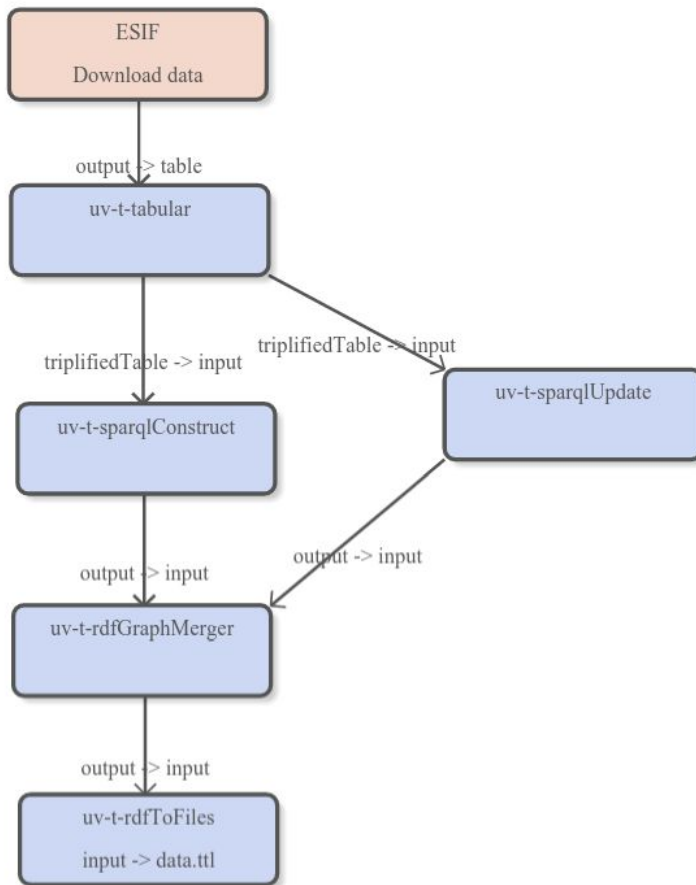


Figure 8. Pipeline to extract functional classification code list from ESIF dataset.

We use similar DPUs as in Figure 3 here, but a different pipeline is created. In the `uv-t-tabular` DPU, now a new mapping between the columns and RDFproperties is described. Both SPARQL queries in `uv-t-sparqlConstruct` and `uv-t-sparqlUpdate` are updated to adapt with code list extraction requirements. The resulting transformed RDF data and the UnifiedViews pipeline for the ESIF 2014 dataset can be found on Github¹³.

Another example of semantic data lifting is the transformation of the Aragon Municipality Budget dataset from CSV data format to RDF. The transformation pipeline is similar to Figure 3. However, the details on the mapping and SPARQL queries are customized, so is the code list transformation.

3.1.2. EU Budget 2014 XML Pipeline

Another dataset that has been transformed into RDF is European Budget 2014¹⁴. The dataset is available in XML format and hence provides another use case for transformation

¹³ <https://github.com/openbudgets/datasets/tree/master/ESIF/2014>

¹⁴ <https://open-data.europa.eu/en/data/dataset/budget-of-the-european-union-2014>

pipelines. Table 2 provides required DPUs for transforming EU Budgets data from XML format into RDF.

DPU	Functionality
uv-e-filesDownload	Downloads files from external source into the UnifiedViews platform
uv-t-unzipper	Uncompress files
uv-t-filesFilter	Filters files based on their names
uv-t-xslt	Performs XSL transformation
uv-t-filesToRdf	Converts RDF files to an in-memory RDF model
uv-t-rdfGraphMerger	Merges RDF graphs
uv-l-rdfToVirtuoso	Loads RDF to Virtuoso server
uv-t-rdfToFiles	Serializes an RDF from graph into a file
uv-l-filesUpload	Uploads file specified URI, can also be used to upload into local location
E-DatasetMetadata	Provides metadata about the dataset
E-DistributionMetadata	Provides metadata about dataset distribution, such as URL of SPARQL Endpoint
L-CKANOdcz	Load the dataset into CKAN

Table 2. List of required DPUs to transform XML to RDF format.

The detailed pipeline for EU Budget data transformation is available in Figure 9. In this figure, the DPU `uv-t-unzipper` decompresses the file downloaded from the EU Open Data website. The decompressed files are then filtered using regular expressions specified in the DPU `uv-t-filesFilter`. In the DPU `uv-t-xslt`, the user needs to specify the XSL template for further processing. Later, the DPU `uv-t-filesToRDF` provides an '*RDF data unit*' from the output of the previous DPU `uv-t-xslt`. Meanwhile, the DSD for this transformation is provided via URL in the DPU `uv-e-filesDownload`. This DSD is then transformed into an *RDF data unit* using DPU `uv-t-filesToRDF` (as in the ESIF CSV pipeline). The resulting RDF data and the DSD are then merged into one graph via the `uv-t-rdfGraphMerger` DPU, and uploaded into Virtuoso server via the DPU `uv-l-rdfToVirtuoso`. The merged graph is serialized using the `uv-t-rdfToFiles` DPU. Here the user provides the RDF serialization format and a filename for the result of the transformation. This file is then uploaded to a specific URL by using the `uv-l-filesUpload` DPU. After this process is done, the `uv-t-rdfGraphMerger` DPU is

run. Metadata about the dataset and the dataset distribution are provided. This is done via the DPUs E-DatasetMetadata and E-DistributionMetadata, respectively. The result is then loaded into a triple store via the DPU uv-l-rdfToVirtuoso, to CKAN API via the DPU L-CKANOdcz and into a file via the DPU uv-l-filesUpload.

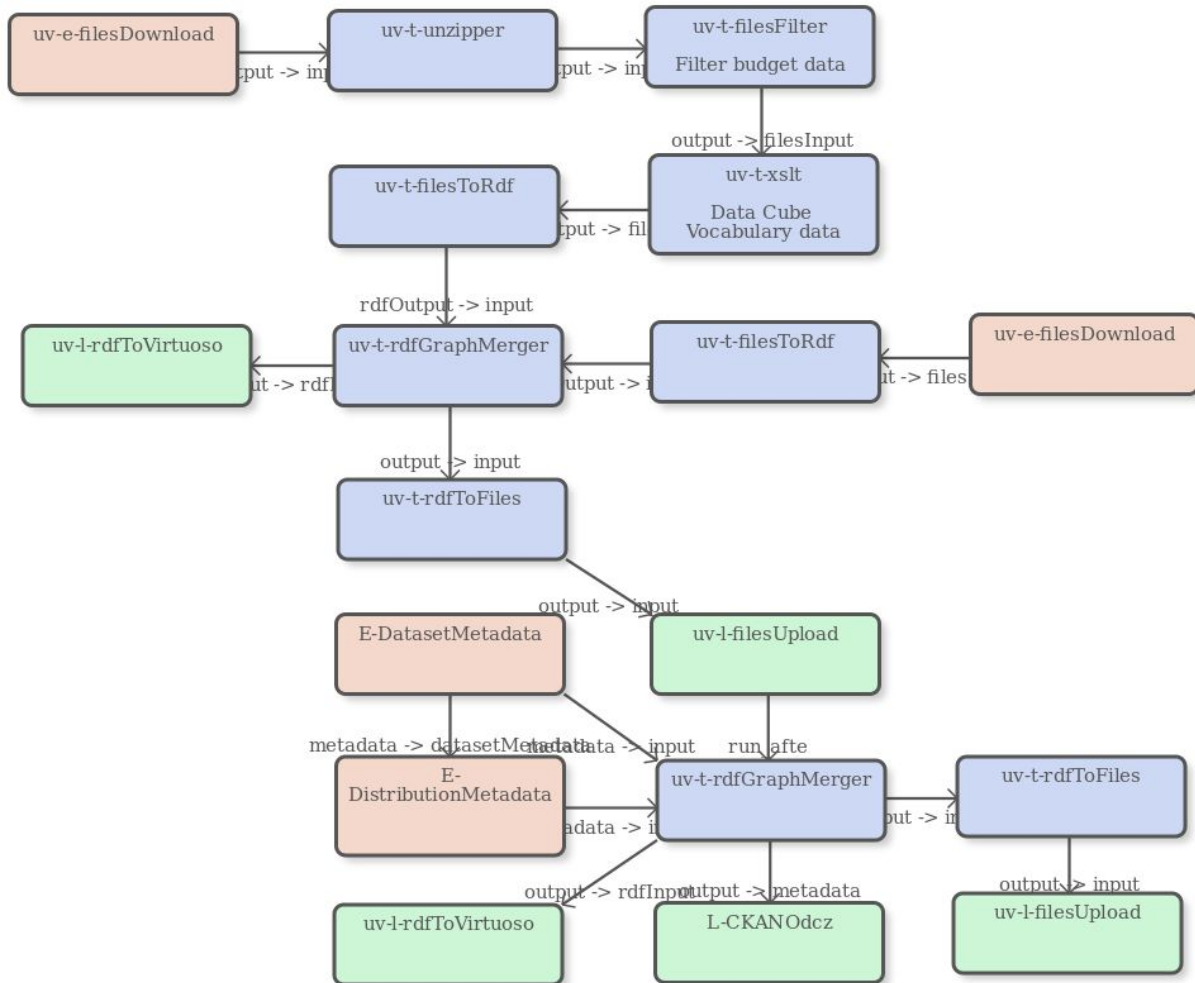


Figure 9. Transformation of EU Budgets 2014 XML dataset into OBEU RDF.

3.2 Pipelines developed using the LinkedPipes ETL Platform

The UnifiedViews platform is quite limited when it comes to integration with other software and UI customization due to lack of APIs and the tight integration of its backend and frontend parts. Based on the experience gathered from usage and support of UnifiedViews, [LinkedPipes ETL](#) (LP-ETL) was implemented. It focuses on better integration using well defined open APIs, advanced debugging support for complex and long running pipelines and last but not least a nicer, more user friendly interface. It is also more lightweight as it only requires Java 8 for backend and Node.js for frontend, MySQL and Apache Tomcat are not used.

The UnifiedViews pipeline for the ESIF dataset can be re-implemented at LinkedPipes ETL platform, as shown in Figure 10.

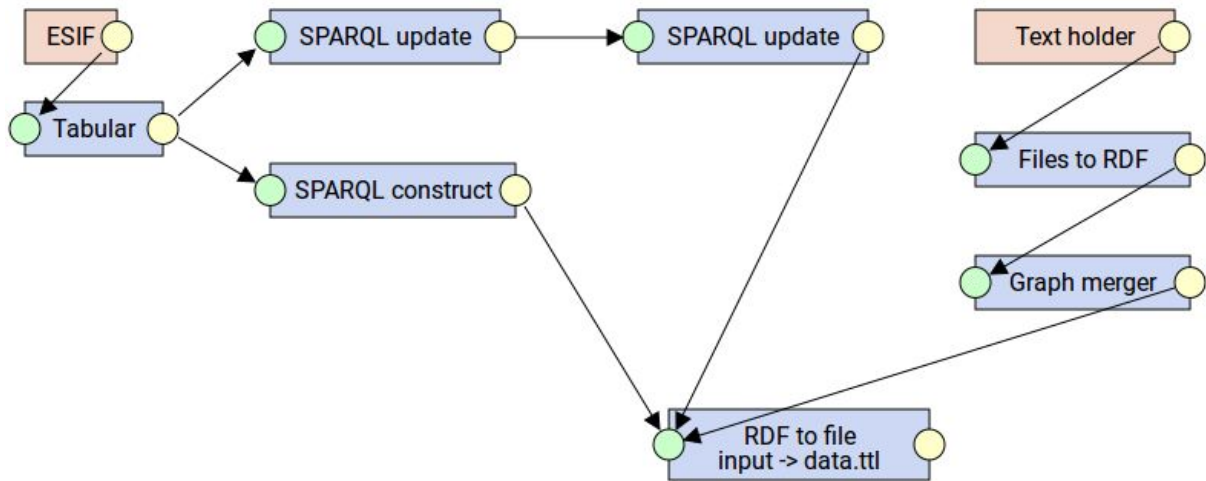


Figure 10. LinkedPipes ETL Pipeline for ESIF 2014-2012 Dataset.

While the pipeline looks and functions similar to the one developed at UnifiedViews platform, LinkedPipes ETL offers better debugging functionality. First of all, this pipeline produces an RDF file, and does not load it anywhere. In contrast to UnifiedViews, to get the file we had to login to the server via SSH and search the file system for the file due to missing debug functionality for files. In LinkedPipes ETL, this file is directly accessible. Another appealing feature, which was not possible at the UnifiedViews platform, is graphical debugging support. The user can see where his pipeline failed in a graphical manner, fix it, and resume from the point of failure, as shown in Figure 11. The red DPU is the failed one and the green ones are the ones that were executed OK. When the failed one is fixed, LinkedPipes ETL does not run the green components again and continues to execute only the ones actually needed.

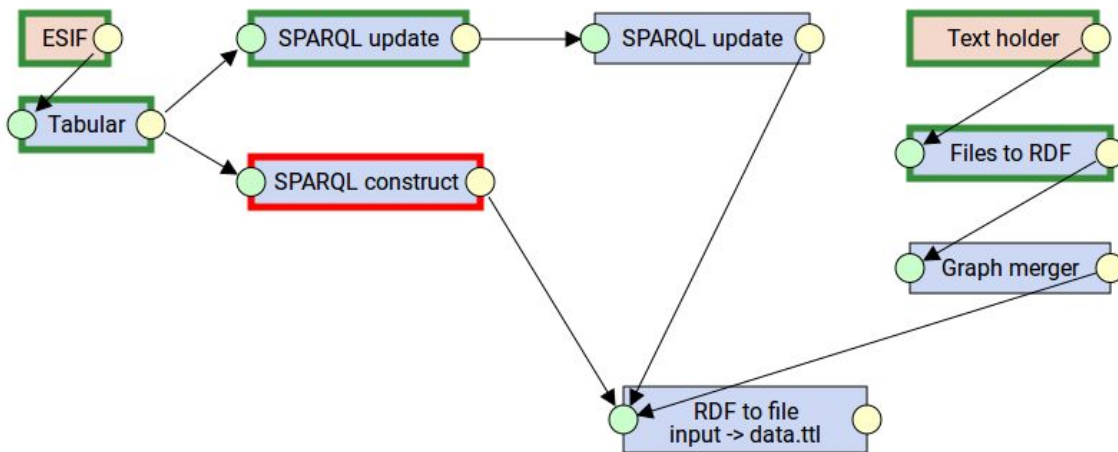


Figure 11. Visual debugging in LinkedPipes-ETL

Pipelines in LinkedPipes are saved as a JSON-LD RDF serialization¹⁵, which is more friendly and easier to both produce and consume than the format used in the relational database at the UnifiedViews platform.

The library of components is documented on the LinkedPipes ETL web¹⁶ and covers all the basic DPUs of UnifiedViews. In addition, UnifiedViews DPUs can be quite easily rewritten to LP-ETL components, the hardest part being the rewrite of the DPU configuration dialog, which instead of Vaadin¹⁷ uses Angular Material¹⁸.

For the EU Budget 2014 datasets, the transformation pipeline for LinkedPipes is shown in Figure 12. This pipeline is conceptually the same as the UnifiedViews pipeline shown in Figure 9.

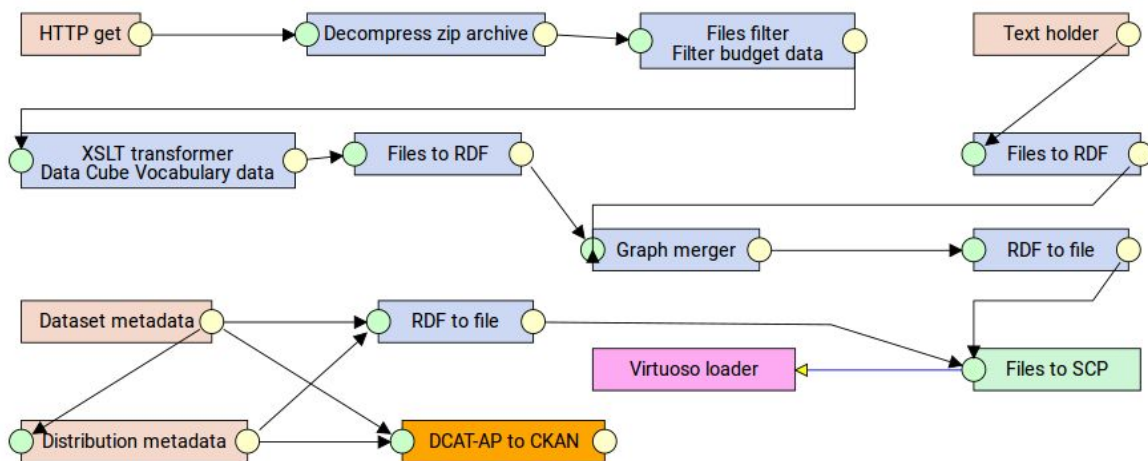


Figure 12. LinkedPipes ETL Pipeline for EU Budget 2014 Dataset.

3.3 List of Transformed Datasets

The following dataset groups have been already transformed into the RDF format.

- [EU Budget 2014](#) (XML→ RDF)
- [ESIF 2014-2020](#) (CSV→ RDF)
- [Aragon Budget](#) (CSV→ RDF)
- [Greek municipalities \(Athens & Thessaloniki\)](#) (CSV → RDF)

The link provided also contains the pipeline files that can be loaded into the UnifiedViews installation.

¹⁵

<https://github.com/openbudgets/datasets/blob/master/ESIF/2014/pipelines/ESIF%202014-2020%20v04.jsonld>

¹⁶ <http://etl.linkedpipes.com/components>

¹⁷ <https://vaadin.com>

¹⁸ <https://material.angularjs.org>

4 The FDP Data Transformation Wizard

In Section 3, we described two data-transformation tools, which require users to manually write out DSD (Data Structure Definition) files, code list files, and SPARQL query statements. For non-technical domain experts, it might be inconvenient. In this section, we will illustrate a data-transformation prototype wizard developed by our Open Knowledge¹⁹ colleagues. This prototype provides user-friendly interfaces to transform CSV files into FDP (Fiscal Data Package) files, by selecting or adding information to columns. The first user interface is illustrated in Figure 13.

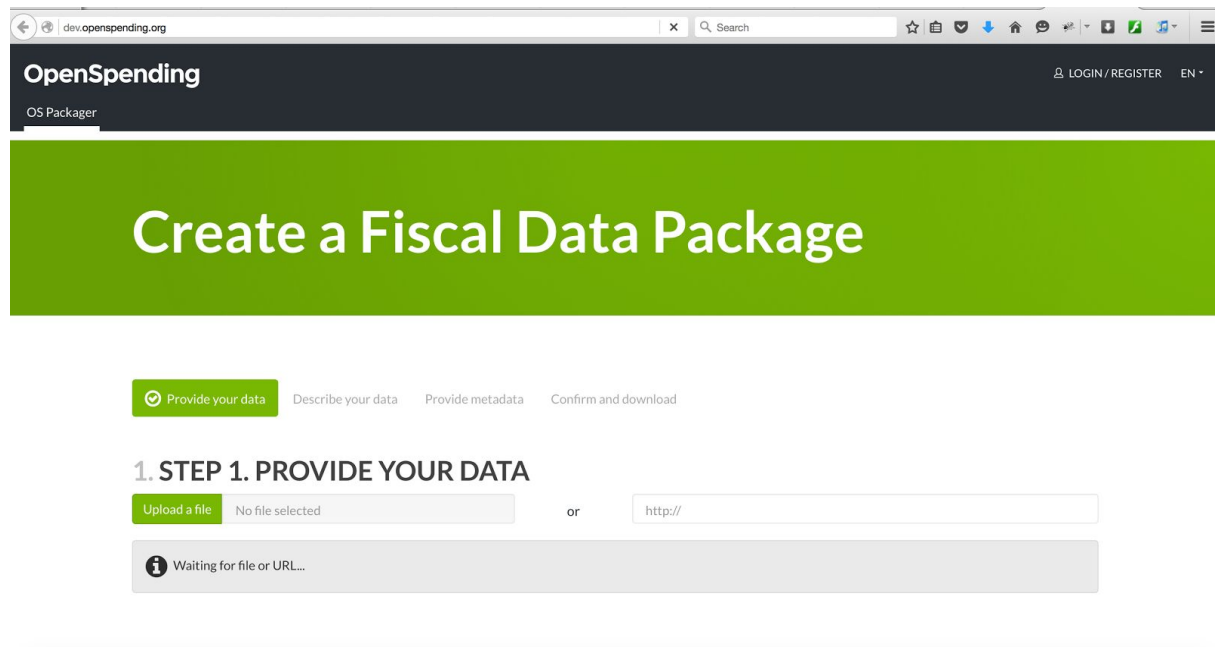
The image shows a web browser window displaying the 'OpenSpending' website. The browser's address bar shows 'dev.openspending.org'. The website header includes the 'OpenSpending' logo and a navigation menu with 'OS Packager', 'LOGIN / REGISTER', and 'EN'. A large green banner with the text 'Create a Fiscal Data Package' is prominent. Below the banner, a progress bar shows four steps: 'Provide your data' (active), 'Describe your data', 'Provide metadata', and 'Confirm and download'. The first step, '1. STEP 1. PROVIDE YOUR DATA', is expanded to show two input options: 'Upload a file' (with a 'No file selected' message) and 'http://'. A grey message box below the inputs says 'Waiting for file or URL...'.

Figure 13. The start page of the wizard

Installation instruction of the up-to-date version can be found in the “*Setting up the development environment*” section at <https://github.com/openspending/openspending>.

Four steps are needed to create a financial data package from a CSV file: (1) providing a CSV dataset; (2) describing the dataset; (3) providing metadata information; and (4) confirming and downloading the created FDP datapackage.

Step 1 Providing a CSV dataset

Users have two ways to select a CSV dataset file: either upload a file from their local machine, or provide the URL of the file, as illustrated in Figure 13.

¹⁹ <https://okfn.org/>

The system will automatically check whether there are any errors in the provided raw dataset, e.g. duplicated lines or column mismatching. Users have to correct these errors before proceeding to the next step as illustrated in Figure 14.

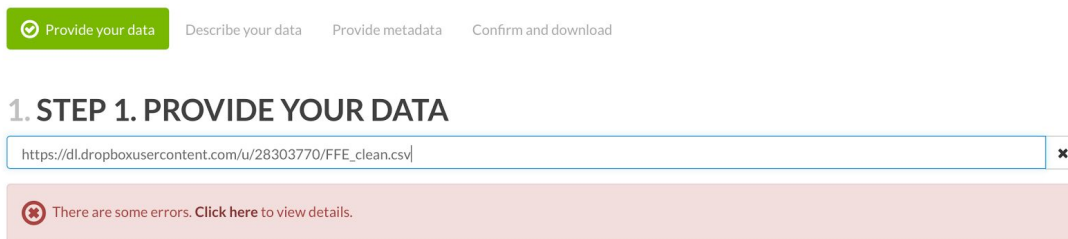


Figure 14. The system identified errors in a CSV file, and asked users to view

For the dataset in Figure 14, there is an error. If we click ‘Click here’, a detailed error message will appear, as shown in Figure 15.

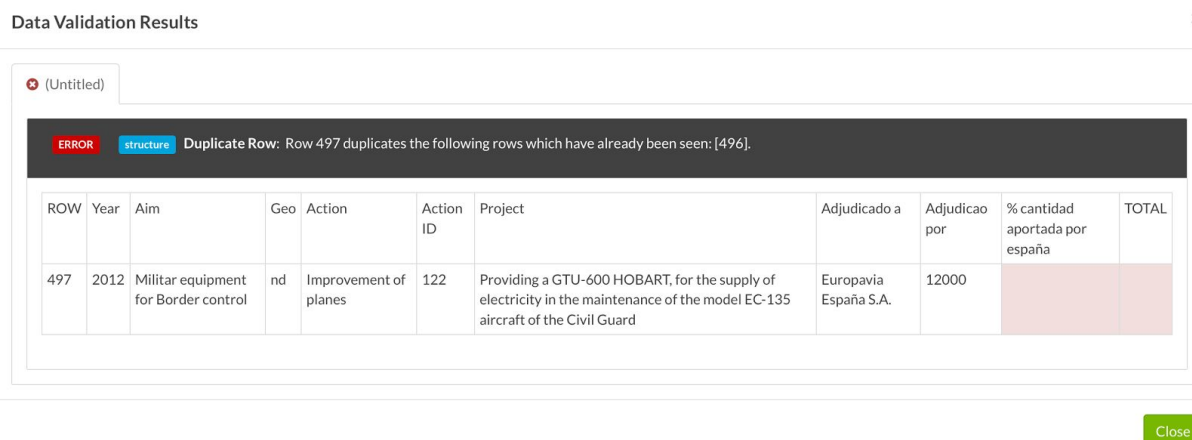


Figure 15. Error report appears during the data providing step

As this dataset is provided through a web-link, users have to download the dataset, correct the errors on their local computer, and upload the error-free dataset to the system.

Step 2 Describing the dataset

After an error-free CSV dataset is uploaded, a user-friendly interface will be provided to describe all columns of the dataset. To help users understand meanings of columns, the system will list several data lines as samples, as shown in Figure 16.

1. STEP 2. DESCRIBE YOUR DATA

Sample of data (3 rows)

COD_LOCALE_PROGETTO	CUP	DPS_TITOLO_PROGETTO	QSN_COD_PRIORITA	QSN_DESCRIZI
10ABF11J11000320007	F11J11000320007	AMMODERNAMENTO E RAFFORZAMENTO DELLE ISTITUZIONI DEL MERCATO DEL LAVORO -PIANO 2009-2010-2011- PERSONALE	7	CompetitivitÃƒf/ produttivi e occu
10ABF11J11000330007	F11J11000330007	ATTUAZIONE DI RIFORME DEI SISTEMI DI ISTRUZIONE E DI FORMAZIONE AL FINE DI SVILUPPARE L'OCCUPABILITÃ - PIANO 2009-2010-2011- PERSONALE	1	Miglioramento e delle risorse um:

Figure 16. The structure of an uploaded CSV file is shown with 3 rows as sample

Column information includes a description of the column, data type, and concept, as illustrated in Figure 17.

Four data types are introduced: *string*, *number*, *integer*, and *any* and six concepts: *Amount*, *Date/Time*, *Entity*, *Classification*, *Location*, and *Activity*.

A non-digit string can be of an instance in the concepts of *Entity*, *Classification*, *Location*, or *Activity*. The difference between *number* and *integer* type is that *number* type includes *integer* and *float* numbers. An integer value can be of an instance in the concepts of *Amount*, *Date/Time*, *Entity*, *Classification*, *Location*, *Activity*. A non-integer number will refer to an instance in the concepts of *Amount*, *Entity*, *Classification*, *Location*, *Activity*. Candidate data types and candidate concepts of a column are inferred from the sample values of this columns. The default candidate data types of a column are *string* and *any*, as illustrated in Figure 18. The default candidate concepts of a column are *Entity*, *Classification*, *Location*, *Activity*, as illustrated in Figure 19.

For the Amount column, users are required to choose the currency (for example EURO), direction (either expenditure or revenue), and phase (proposed, approved, adjusted, or executed), and describe the factor (default is 1).

Users must set at least an **Amount** and a **Date / Time** of two columns, to move on to Step 3.

Description					
Title	Cod Locale Progetto	Cup	Dps Titolo Progetto	Qsn Cod Priorita	Qsn Descrizion
Description					
Data type	string	string	string	integer	string
Concept					

i You should map at least an **Amount** and a **Date / Time** to continue.

Figure 17. An interface to describe column information

Qsn Cod Priorita	Qsn Descrizione Priorita	Qsn Cod Obiettivo Generale
integer	string	number
string	string	string
integer	any	number
number		any
any		

Figure 18. Candidate data types of columns

Qsn Cod Priorita	Qsn Descrizione Priorita	Qsn Cod Obiettivo Generale
integer	string	number
Amount	Entity	Amount
Date / Time	Classification	Entity
Entity	Activity	Classification
Classification	Location	Activity
Activity		Location
Location		

Figure 19. Candidate concepts are inferred from the sample value of the column

Step 3 Providing metadata information

At Step 3, users are required to provide meta information of the dataset, e.g., name of the data package, location, period, as illustrated in Figure 20.

- Provide your data
- Describe your data
- Provide metadata
- Confirm and download

1. STEP 3. PROVIDE METADATA

Name your Data Package *

FFE

Short description

Expense data from co-funded EU and Spain solidarity/migration spending

Continent

Europe

Country

Spain

City

Period

Format: YYYY-MM-DD

Format: YYYY-MM-DD

Congratulations! Now you can verify your Data Package and download it.

Continue >

Figure 20. User interface for Metadata

Step 4 Confirming and Downloading

At Step 4, users have the chance to confirm the input and download the metadata package, as illustrated in Figure 21. At the time of writing this deliverable, the function of publishing this dataset is still under construction.

- Provide your data
- Describe your data
- Provide metadata
- Confirm and download

1. STEP 4. CONFIRM AND DOWNLOAD

Basic Information

Package Title

FFE

Package Name

ffe

Location

eu, ES

Files

ffe_clean (FFE_clean.csv)

Column Mapping

total

ffe_clean / TOTAL

datetime

ffe_clean / Year

location

ffe_clean / Geo

classification-functional

ffe_clean / Action

ffe_clean / Action ID

activity

ffe_clean / Project

Download

Publish this Data Package


Figure 21. Confirmation/Download of a generated FDP dataset

5 Towards An RDF Data Transformation Wizard

In this section, we conceptualize an RDF data transformation tool by blending the “nice” parts of the tools described in Section 3 and the wizard presented in Section 4. The “nice” part of Section 3 is creating pipelines for data transformation. The un-appealing part is that users may need to write explicitly DSDs and SPARQL queries for code list extraction and the dataset transformation process. The “nice” part of Section 4 is the idea of using a wizard to collect mapping and metadata information and the rest is carried out by the backend of the wizard. Users do not need to know the technical structure. To cover the un-appealing part of the pipeline method, we will develop a data-transformation wizard, with which sufficient data information can be collected so that DSD creation, code list extraction, and the RDF data transformation can be carried by the backend.

The main workflow can be described as follows: (1) load a raw dataset; (2) select and map columns of the dataset to predefined OBEU dimensions which can be selected from a list; (3) specify code list included or used in the dataset; (4) check and save the result of the transformation which has been done at the backend using a generated DSD, a generated pipeline and possibly extracted code lists based on the previous steps; and (5) go back to (2) for the next tabular data structure in the raw dataset, or push all the results into the RDF triple-store. As it is a wizard, it is possible at each step to go back to the earlier step, e.g. if it is necessary to correct something in the previous step.

Step 1 Load a raw dataset

OpenBudgets.eu - Data Upload 

Choose a dataset to upload:	<input type="text" value="Berlin2014.csv"/>
Is it budget or spending data?	<input type="radio"/> Spending data <input checked="" type="radio"/> Budget data <input data-bbox="1227 680 1262 719" type="button" value="?"/>
Dataset title	<input type="text" value="Berlin budget 2014"/>
Comment for the dataset	<input type="text" value="Budget data for Berlin on state level in fiscal period 2014 ..."/>

Figure 22: Initial user interface for data upload

The user (a domain expert on financial data) selects a dataset to be uploaded and provides a title for the dataset (like eu-budget-2014). In addition, it is possible to give a description on the dataset. Because of the slight difference in the budget/spending data model, the user specifies at the beginning whether it is budget or spending data. This way, the wizard can better support the user in the following mapping steps. A help button will be provided to guide the user in the decision. The initial user interface is illustrated in Figure 22.

The backend of this step is to extract data contents from the dataset. A tree-structured dataset, e.g. XML file, may contain several data-tables. We will extract them out, and structure them into a list. If the raw dataset is CSV dataset, this list will have only one data-table. For each data-table in the list, we can re-use tools of the prototype wizard for structural checking. If passed, a DSD-specific namespace will be automatically generated based on the given title and we can move on the next step, as shown in Figure 23.

OpenBudgets.eu - Data Upload



1. Attribute selection

Berlin2014.csv

Year	depNo	Department	Objective	Description	Title	Amount
2014						
2014						
2014						

Select and map attributes:

- Budgetary Unit
- Fiscal Period
- Functional Classification
- Administrative Classification
- Amount
- Currency
- ...

Administrative Classification

Given in a column:

Uses a code list

Is attached to the dataset:

Figure 23: Concept of a wizard collection mapping information.

Step 2 Map Columns to OBEU Components

In this step, a graphical interface will be prompted to collect information for each column of the current CSV dataset. Possible component properties, i.e., *dimensions*, *measures*, and *attributes*, will be listed, so that the user can select easily. For a budget dataset, the mandatory components are *budgetary unit*, *fiscal period*, *amount*, and *currency*. For a spending dataset, the mandatory components are *organization*, *operation character*, *date*, *amount*, *currency (attribute)*. Information in this step shall be sufficient to generate component properties of the corresponding DSD.

The user can choose which columns (components) will form the Data Structure Definition. In case that a CSV dataset has more than one DSD, users are able to specify each of them.

The usage of the wizard should be simple, straightforward and self-explanatory. Especially, the user should be able to proceed without any knowledge on the OBEU data format. Naming policy will be used for generation OBEU data format.

In this step, we shall collect all necessary information for automatically generating data structure definitions (`qb:DataStructureDefinition`) and the component properties as an RDF file in Turtle serialization (`xxx-components.ttl`). Naming policy will be created,

so that user friendly terms can be automatically translated into the machine readable codes in the RDF formats.

Step 3 Map to existing code lists

OpenBudgets.eu - Data Upload



2. Codelist specification

Berlin2014.csv

Year	depNo	Department	Objective	Title	Description	Amount
2014						
2014						
2014						

Continue

The dataset contains codelists for the following attributes.
Please specify them here:

Administrative Classification

Functional Classification

Codelist specified in a separate resource

Select resource

Codelists to be extracted from the dataset

- Code:
- Label:
- Description:

Figure 24. Interface for Code List Extraction

This step will continue the information collection with a focus on code lists, so that these can be generated automatically. A user interface will be prompted for those dimensions that contain code lists as specified in the previous step, as illustrated in Figure 24. Under certain conditions, we can use a LinkedPipes ETL pipeline to extract a code list, for example, the functional classification in the ESIF dataset, where the codes are given with labels and comments in different columns. Also users shall be able to mark explicitly if the code lists contain narrower/broader relations among columns.

Some columns (dimensions) have predefined data contents. For example, *operation character dimension* has some predefined data contents: *expenditure*, *financing*, and *revenue*. If a column is *operation character dimension*, users shall be mark its data as one of the *expenditure*, *financing*, and *revenue*.

At the end of the mapping process, the user is presented a summary and can check the identified components. An example is given in Figure 25.

OpenBudgets.eu - Data Upload

4. Checking and Upload

Identified components:

Year	depNo	Department	Objective	Description	Title	Amount
2014						
2014						
2014						

berlin-budget-2014 (Berlin2014.csv, budget data)

Selected and specified components...

... having data set scope:

- Budgetary Unit Berlin
- Fiscal Period Year: 2014
- Currency EUR

... having observation scope:

- Functional Classification See details
- Administrative Classification See details
- Amount See details

[Continue](#)

Figure 25. Mapping Summary for Checking and Confirmation

The information provided so far shall be complete for automatically generate the whole DSD file (xxx-dsd.ttl), which includes metadata properties (xxx-metadata.ttl), code lists (xxx-codelists.ttl), and component properties (xxx-components.ttl). Based on these, the transformation pipeline including SPARQL construct/update statement will be automatically generated and executed to transform the current CSV dataset into RDF format.

Step 4 Check and Save the Result

The result will be presented to the users, to check whether the result is correct. If no errors are found, the result will be saved locally.

If the raw dataset is a tree-structured, it may contain several tabular-structured datasets, the wizard will move to Step 2 for the next tabular-structured dataset. Otherwise, move on to Step 5.

Step 5 Push to RDF Triple Store

All locally saved results are pushed into the RDF Triple store.

6 Conclusion and Future Work

This deliverable mainly summarizes the experimental work on developing data transformation tools, in particular pipelines development using UnifiedViews and LinkedPipes ETL, and testing the data transformation wizard. Based on these efforts, we conceptualized a wizard for data transformation, which aims at providing user-friendly interfaces to transform tabular-structured datasets (e.g., CSV) and tree-structured (e.g., XML) datasets into RDF format. This work is under construction.

7 References

- Klímek J., Kučera J., Mynarz J., Sedmihradská L., Zbranek J. (2015a): OpenBudgets.eu - Deliverable D1.2 - Design of data structure definition for public budget data, <http://openbudgets.eu/assets/deliverables/D1.2.pdf>
- Klímek J., Kučera J., Mynarz J., Sedmihradská L., Zbranek J. (2015b): OpenBudgets.eu - Deliverable D1.3 - Design of data structure definition for public spending data, <http://openbudgets.eu/assets/deliverables/D1.3.pdf>
- Dudáš M., Horáková L., Klímek J., Kučera J., Mynarz J., Sedmihradská L., Zbranek J., Dong T., (2015): OpenBudgets.eu - Deliverable 1.4 - User Documentation, <http://openbudgets.eu/assets/deliverables/D1.4.pdf>
- Ioannidis, L., Philippides, P.-M., Bratsas, C., Koupidis, K. (2015): OpenBudgets.eu – Deliverable D1.6 – Survey of code lists for the data model's coded dimensions, <http://openbudgets.eu/assets/deliverables/D1.6.pdf>
- Ioanidis L., Klímek J., Musyaffa F., Mynarz J., Sedmihradská J., Zbranek J., (2016): OpenBudgets.eu - Deliverable 1.4 - Linking Code Lists to External Datasets, <http://openbudgets.eu/assets/deliverables/D1.9.pdf>