OpenDataMonitor

SEVENTH FRAMEWORK PROGRAMME

ADDITIONAL DESCRIPTIVE REPORT

D4.1 DEPLOYMENT OF THE TOOL DEMONSTRATOR FINISHED

PROJECT

Acronym:	**OpenDataMonitor**
Title:	Monitoring, Analysis and Visualisation of Open Data Catalogues, Hubs and Repositories
Coordinator:	SYNYO GmbH
Reference:	**611988**
Type:	Collaborative project
Programme:	FP7-ICT
Start:	November 2013
Duration:	24 months
Website:	http://project.opendatamonitor.eu
E-Mail:	office@opendatamonitor.eu

Consortium:

SYNYO GmbH, Research & Development Department, Austria, (SYNYO)

Open Data Institute, Research Department, UK, (ODI)

Athena Research and Innovation Center, IMIS, Greece, (ATHENA)

University of Southampton, Web and Internet Science Group, UK, (SOTON)

Potsdam eGovernment Competence Center, Research Department, Germany, (IFG.CC)

City of Munich, Department of Labor and Economic Development, Germany, (MUNICH)

Entidad Publica Empresarial Red.es, Shared Service Department, Spain, (RED.ES)

SYNYO ODI ΙΝΣΥ IMIS University of Southampton ifG.CC City of Munich red.es

DELIVERABLE

Number:	**D4.1**
Title:	**Deployment of the tool demonstrator finished**
Lead beneficiary:	ATHENA
Work package:	WP4: Demonstration and instructions
Dissemination level:	Public (PU)
Nature:	Demonstrator (D)
Due date:	November 30, 2014
Submission date:	November 30, 2014
Authors:	Vassilis Kaffes, ATHENA Dimitris Skoutas, ATHENA Thodoris Raios, ATHENA
Reviewers:	Bernhard Jaeger, SYNYO Ejona Sauli, SYNYO

Acknowledgement: The OpenDataMonitor project is co-funded by the European Commission under the Seventh Framework Programme (FP7 2007-2013) under grant agreement number 611988.

Disclaimer: The content of this publication is the sole responsibility of the authors, and in no way represents the view of the European Commission or its services.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

The goal of the OpenDataMonitor (ODM) project is to make it possible for interested stakeholders to gain an overview of the evolving open data landscape. More specifically, ODM aims to achieve this goal by designing and developing:

- an extensible and customizable harvesting framework, for facilitating and automating as much as possible the collection of metadata from diverse open data catalogues;
- an integration and harmonisation workflow, for overcoming the high heterogeneity of schemas, values and formats found in the various open data sources;
- scalable analytical and visualisation methods, for allowing end users to explore the results in an intuitive and user-friendly manner in order to obtain a comprehensive overview of the collected information.

This report accompanies the first prototype of the ODM demonstrator, the system being developed to provide the aforementioned capabilities. The prototype includes components for registering open data catalogues for monitoring, harvesting metadata from them, cleaning and harmonising metadata attributes and values, and finally computing and presenting, through a RESTful API, a set of predefined metrics. Detailed information regarding the architecture of the ODM system and its components, as well as the current status of the implementation, can be found in Deliverables D3.1 and D3.3, respectively. The purpose of this report is to provide information about the current deployment of the demonstrator and instructions for setting up and using the software.

The remainder of this report is structured as follows. Section 2 contains deployment information of the prototype. Section 3 describes in detail the installation process of the ODM demonstrator. Section 4 provides a walkthrough of the steps to be followed in order to register a catalogue, execute the harvesting process and finally run the metadata harmonization to produce the metadata collection on which the metrics for monitoring are computed. Section 5 summarizes the features of the first version of the prototype and describes the next steps.

2. DEPLOYMENT OF THE OPENDATAMONITOR DEMONSTRATOR

The developed code for the ODM prototype is hosted in the following repository, which is publicly available:

https://github.com/opendatamonitor

The code will be maintained and updated in this repository throughout the course of the project, especially for incorporating all improvements and enhancements that will be made based on user feedback that will be received during the second year of the project. This will also include updated user manuals and installation and configuration instructions.

The repository contains the following components:

- the CKAN harvester: https://github.com/opendatamonitor/ckanext-harvest
- the back-end of the HTML harvester: https://github.com/opendatamonitor/ckanext-htmlharvest,
- the Web User Interface for configuring the HTML harvester for each registered catalogue: https://github.com/opendatamonitor/ckanext-metaschemaform
- the harmonisation engine https://github.com/opendatamonitor/harmonisation_engine
- the API that provides access to the harmonised metadata: https://github.com/opendatamonitor/odm.restapi

The demo is deployed on a Virtual Machine (VM) hosted on a server with the following specifications:

- CPU: Intel(R) Xeon(R) CPU E5-2420 v2 @ 2.20GH (3 cores)
- Memory: 8 GB
- Disk: 884GB
- Operating system: Debian 7.6 (wheezy)

The VM is accessible under the IP address: 83.212.122.164.

In particular, the forms for registering a catalogue and configuring a harvesting job can be accessed at: http://83.212.122.164/harvest.

Moreover, the RESTful API is configured to use the port 27080 under the directory /api/v1.0/. It can be used by appending at the end of the URL the name of the method corresponding to each of the predefined metrics or retrieval methods that were implemented in the context of the ODM project (see Deliverable D3.3). For instance: http://83.212.122.164:27080/api/v1.0/_commands returns a list of supported commands.

3. INSTALLATION AND CONFIGURATION OF THE OPENDATAMONITOR DEMONSTRATOR

The ODM demonstrator is built on top of the CKAN platform[1]. Hence, the installation and configuration process involves essentially setting up a CKAN instance and then installing some additional plugins. There are two ways to install CKAN: from package or from source. The former is quicker and easier, but it is available only for Ubuntu 12.04.64-bit. Information on how to install it can be found here: http://docs.ckan.org/en/943-writing-extensions-tutorial/install-from-package.html. On the other hand, instructions for installing it from source, e.g. for installing on other operating systems or running CKAN, Solr and PostgreSQL on different systems, can be found here: http://docs.ckan.org/en/943-writing-extensions-tutorial/deployment.html.

After having a running instance of CKAN, we need to edit the product.ini file to adjust it to our needs. Specifically, the following keys should be added:

```
[ckan:odm_extensions]
//connection to the MongoDB database
mongoclient=localhost
mongoport=27017
//path to backup file for html harvesting
backup_file_path=
//logging of related plugins
harmonisation_engine_log=
html_harvester_log_file_path=
ckan_harvester_error_log=
//the admin key in deployed platform
admin_api_key=
```

3.1. Install CKAN extensions for harvesting: ckanharvest, htmlharvest, metaschemaform

Firstly, we need to install the MongoDB database (version 2.6.5). As **root** user run the following command:

```
# apt-get install mongodb-org
```

[1] http://ckan.org/

Also, install the RabbitMQ back-end that is needed by the customised ckanext-harvest:

```
# apt-get install rabbitmq-server
```

After this, we can login again as **ckaner** user, enable the virtual environment and continue the installation.

Download and install the ckanharvest extension, an extended version of the ckanext-harvest[2] adapted to ODM project's needs. It provides harvesting for remote CKAN platforms:

```
$ pip install -e 'git+https://github.com/opendatamonitor/ckanext-harvest#egg=ckanext-harvest'
```

Install all dependencies required by the module:

```
$ pip install -r requirements.txt
```

On the CKAN configuration file, development.ini or product.ini, add the harvester main plugin, the harvester for CKAN instances and also define the back-end that we will use:

```
ckan.plugins = harvest ckan_harvester

ckan.harvest.mq.type = rabbitmq
```

Run the following command to create the necessary tables in the database:

```
$ paster --plugin=ckanext-harvest harvester initdb --config=development.ini
```

After installation, the harvester source page should be available under /harvest, i.e.: http://127.0.0.1:5000/harvest (or the specified host:port combination declared in development.ini)

Download and install two plugins that provide the HTML scrapping functionality to CKAN. The metaschemaform, which stores information of a source page's XPATH tree representation in relation to specified attributes for value retrieval, and the htmlharvest, a plugin that implements the above harvesting interface and runs the actual harvesting:

```
$ pip install -e 'git+https://github.com/opendatamonitor/ckanext-metaschemaform#egg=ckanext-metaschemaform'
```

Install all dependencies required by the module:

```
$ pip install -r requirements.txt
```

[2] https://github.com/ckan/ckanext-harvest

To check that the installation was successful, visit the following URL: http://127.0.0.1:5000/metaschemaform .

```
$ pip install -e 'git+https://github.com/opendatamonitor/ckanext-htmlharvest#egg=ckanext-htmlharvest'
```

Install all dependencies required by the module:

```
$ pip install -r requirements.txt
```

On the CKAN configuration file, development.ini or product.ini, add the htmlharvest plugin:

```
ckan.plugins = htmlharvest htmlmetaschemaform
```

In order to initiate a harvesting job (creation of a job is described in detail in section **Fehler! Verweisquelle konnte nicht gefunden werden.**), two different queues need to run in the background. The first one handles the metadata gathering and the other the fetching and importing into the database:

```
$ paster --plugin=ckanext-harvest harvester gather_consumer --config=development.ini
```

On another terminal, run the following command:

```
$ paster --plugin=ckanext-harvest harvester fetch_consumer --config=development.ini
```

Finally, to initiate the process of harvesting, run the following command on a third terminal:

```
$ paster --plugin=ckanext-harvest harvester run --config=development.ini
```

3.2. Manage processes through supervisor(d)

So far, four daemon processes have been started through paster: serve, celery, gather_consumer, fetch_consumer. In development phase, it is convenient to start these processes manually, let them run in the foreground to directly observe their output, and finally kill them with a SIGTERM signal. However, in a production environment, a process manager like supervisor is more helpful for observing the overall state. The following steps explain how this can be done.

As superuser, add a configuration file into /etc/supervisor/conf.d for each demonized program you wish to keep track of. We add four extra files: ckan-serve.conf, ckan-celeryd.conf, ckan-gather_consumer.conf, ckan-fetch_consumer.conf.

The contents are as follows:

```
$ cd  /etc/supervisor/conf.d

$ cat ckan-serve.conf ckan-celeryd.conf ckan-gather_consumer.conf
ckan-fetch_consumer.conf
```

```
[program:ckan_serve]

; Full Path to executable, should be path to virtural environment,

; Full path to config file too.


command=/var/local/ckan/default/pyenv/bin/paster --plugin=ckan serve
/var/local/ckan/default/pyenv/src/ckan/development.ini


; user that owns virtual environment.

user=ckaner


numprocs=1

stdout_logfile=/var/local/ckan/default/log/serve.log

stderr_logfile=/var/local/ckan/default/log/serve.log

autostart=true

autorestart=true

startsecs=10


; Need to wait for currently executing tasks to finish at shutdown.

; Increase this if you have very long running tasks.

stopwaitsecs = 600
```

```
priority=999


[program:ckan_celeryd]

; Full Path to executable, should be path to virtual environment,

; Full path to config file too.


command=/var/local/ckan/default/pyenv/bin/paster --plugin=ckan
celeryd --
config=/var/local/ckan/default/pyenv/src/ckan/development.ini


; user that owns virtual environment.

user=ckaner


numprocs=1

stdout_logfile=/var/local/ckan/default/log/celeryd.log

stderr_logfile=/var/local/ckan/default/log/celeryd.log

autostart=true

autorestart=true

startsecs=10


; Need to wait for currently executing tasks to finish at shutdown.

; Increase this if you have very long running tasks.

stopwaitsecs = 1200

priority=998


[program:ckan-gather_consumer]
```

```
; Full Path to executable, should be path to virtural environment,

; Full path to config file too.


command=/var/local/ckan/default/pyenv/bin/paster --plugin=ckanext-
odm_ckanharvest harvester gather_consumer --
config=/var/local/ckan/default/pyenv/src/ckan/development.ini


; user that owns virtual environment.

user=ckaner


numprocs=1

stdout_logfile=/var/local/ckan/default/log/gather_consumer.log

stderr_logfile=/var/local/ckan/default/log/gather_consumer.log

autostart=true

autorestart=true

startsecs=10


[program:ckan-fetch_consumer]

; Full Path to executable, should be path to virtual environment,

; Full path to config file too.


command=/var/local/ckan/default/pyenv/bin/paster --plugin=ckanext-
odm_ckanharvest harvester fetch_consumer --
config=/var/local/ckan/default/pyenv/src/ckan/development.ini


; user that owns virtual environment.

user=ckaner
```

```
numprocs=1
stdout_logfile=/var/local/ckan/default/log/fetch_consumer.log
stderr_logfile=/var/local/ckan/default/log/fetch_consumer.log
autostart=true
autorestart=true
startsecs=10
```

3.3. Harmonisation engine

The harmonisation engine is a collection of scripts and dictionaries that deal with the heterogeneity of the original collected metadata. In order to install it, we run the following commands:

```
$ git clone git://github.com/opendatamonitor/harmonisation_engine
$ pip install -r requirements.txt
```

Before running it, we need to edit its configuration file, harmonisation.ini:

```
[harmonisation]
//connection ip and port of the mongodb server
mongoclient=localhost
mongoport=27017
//path to store the logs
harmonisation_engine_log=
```

To run the harmonisation engine, run the following command:

```
$ python HarmonisationEngine
```

3.4. ODM RESTful API

Download and install the API to have access to the processed metadata and also to calculate a number of predefined metrics that can be used for monitoring. In an activated virtual environment we run:

```
$ git clone git://github.com/opendatamonitor/odm.restapi
```

Install any required dependencies by executing:

```
$ pip install -r requirements.txt
```

Finally to initiate the API we run:

```
$ python httpd.py
```

We can access the API here:

http://127.0.0.1:27080/api/v1.0/_commands

4. WEB USER INTERFACE GUIDE

In this section, we provide an overview of the Web user interface of the system. It consists of the main harvest dashboard and the forms for registering a new catalogue and configuring the harvesting jobs for both the CKAN and HTML harvesters. Detailed instructions and figures of the supported functionalities are presented below.

4.1. Dashboards

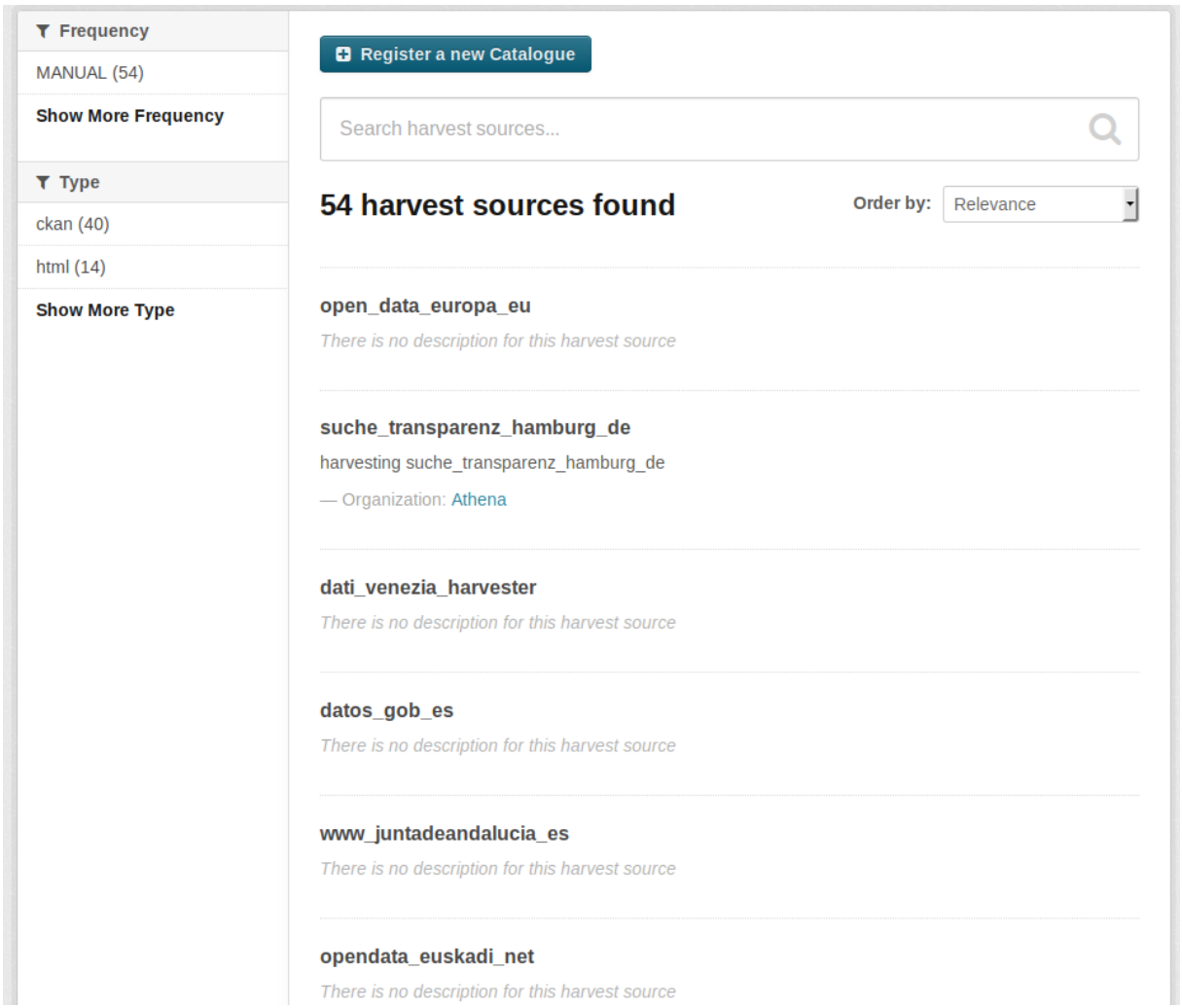

Figure 1: Main harvester dashboard

The central page of harvesting, Figure 1, provides information about the currently registered catalogues. Two types of facets are provided for filtering the results, the harvester type and the periodical harvesting procedure. The first one filters the registered catalogues based on the harvester type (ckan, html) that was used to collect the metadata and the second one over the

declared periodicity of the harvester, upon the registration, for checking new or updated metadata. Additionally basic search functionality is provided for the job title of each of the registered catalogues.

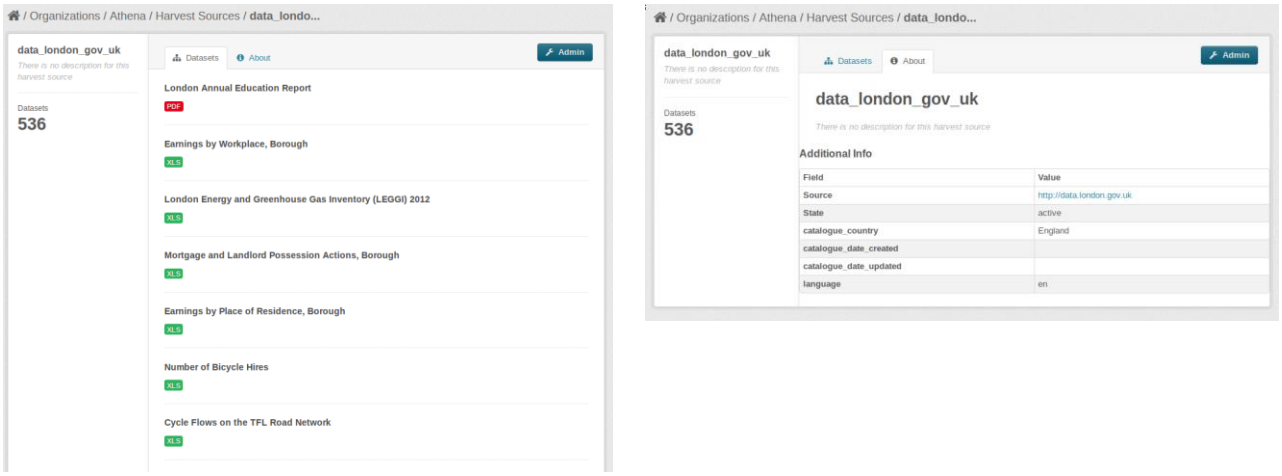

Figure 2: Detailed information related to specific catalogue (left: number of datasets for which metadata where harvested, right: general catalogue info)

Apart from a generic description of every registered catalogue, each title in the main dashboard links to more detailed and customised information panels for each catalogue. In Figure 2 we view the **Datasets** and **About** tabs and also the **Admin** button. The first tab provides references to the current number of datasets whose metadata are harvested and also links to the actual harvested metadata attributes and their values for each one of the datasets. The second contains general registered information about the catalogue itself. The Admin button, on the other hand, links to a panel (Figure 3) with three buttons and three main tabs.

The first group consists of the **Reharvest** button that initiates manually a harvesting job, the **Clear** button that deletes all datasets related to the catalogue and the **View harvest source** button that goes back to the previously described panel which is presented in Figure 2.

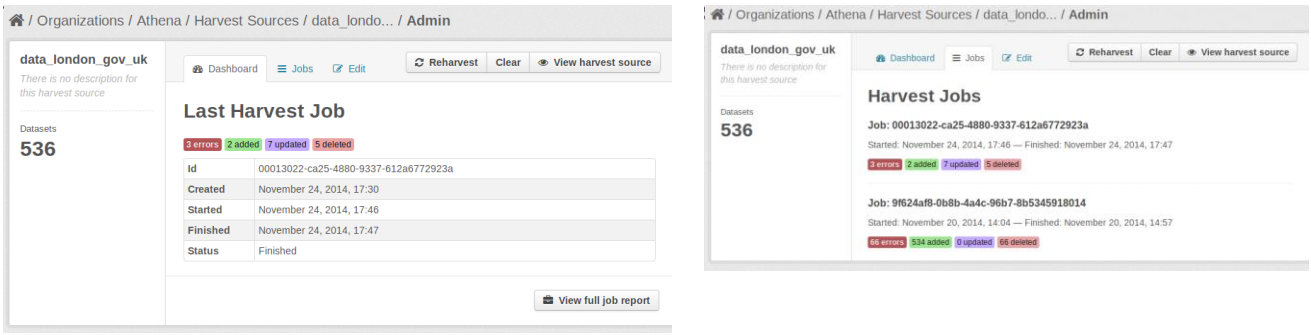

Figure 3: Harvest jobs logging

The other group contains the first two tabs (**Dashboard** and **Jobs**) that provide information about the actual status of the harvesting process and history of previously executed ones. A full job report can be reclaimed by clicking the button in the right bottom corner (**View full job report** in the **Dashboard** tab). The third tab is the **Edit** tab, Figure 4, which is actually a form where we can either update the information provided during the catalogue registration or delete the whole registered job that is related to it.

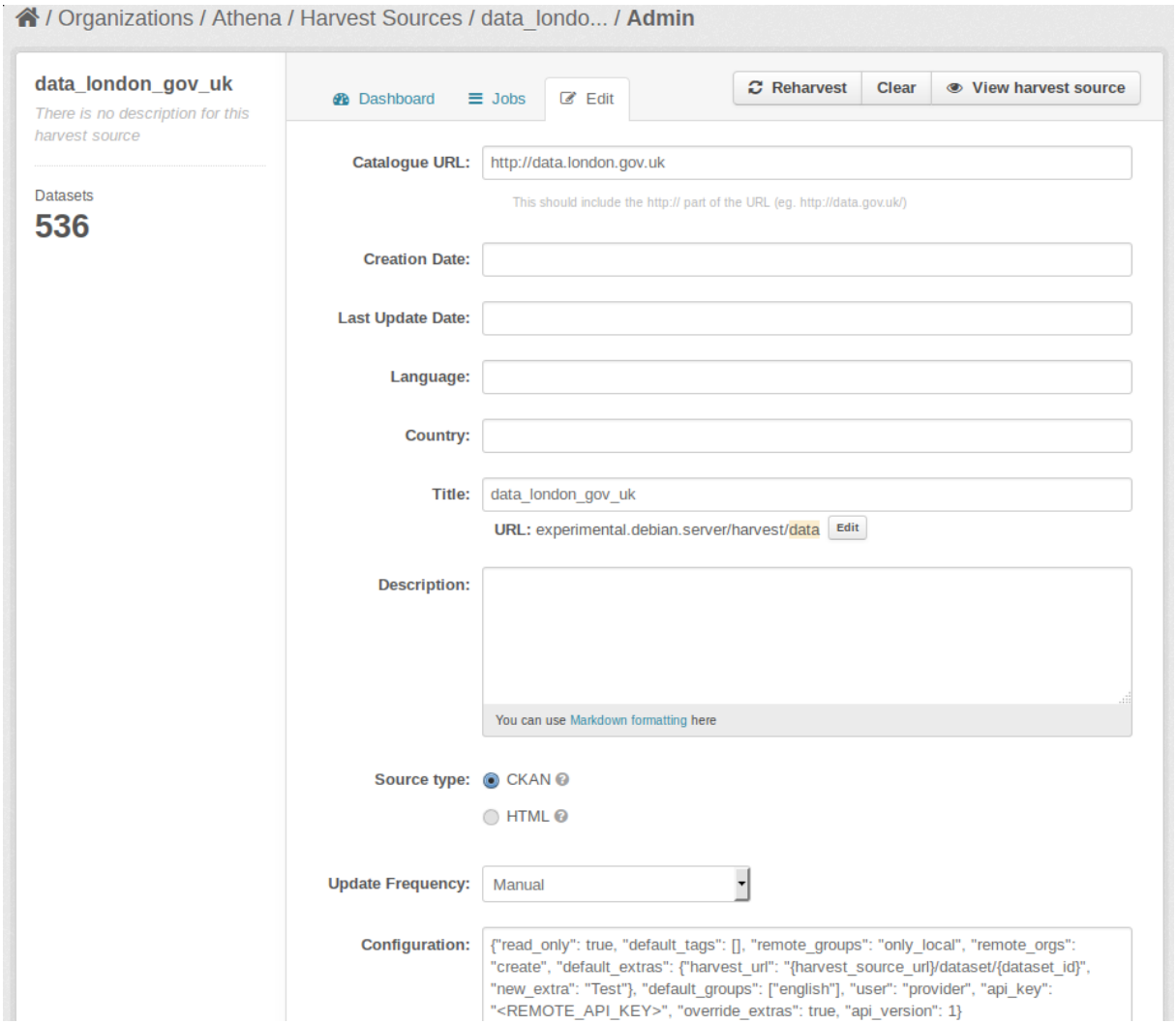

Figure 4: Edit info and job configuration of a registered catalogue

4.2. Catalogue registration and job configuration

The catalogue registration form contains information about the harvesting source e.g. catalogue URL, title, creation date etc. During the registration, information to guide the metadata extraction process is also provided. The form is filled in by the administrator of the CKAN platform or the data-publisher user (e.g. catalogue owner).

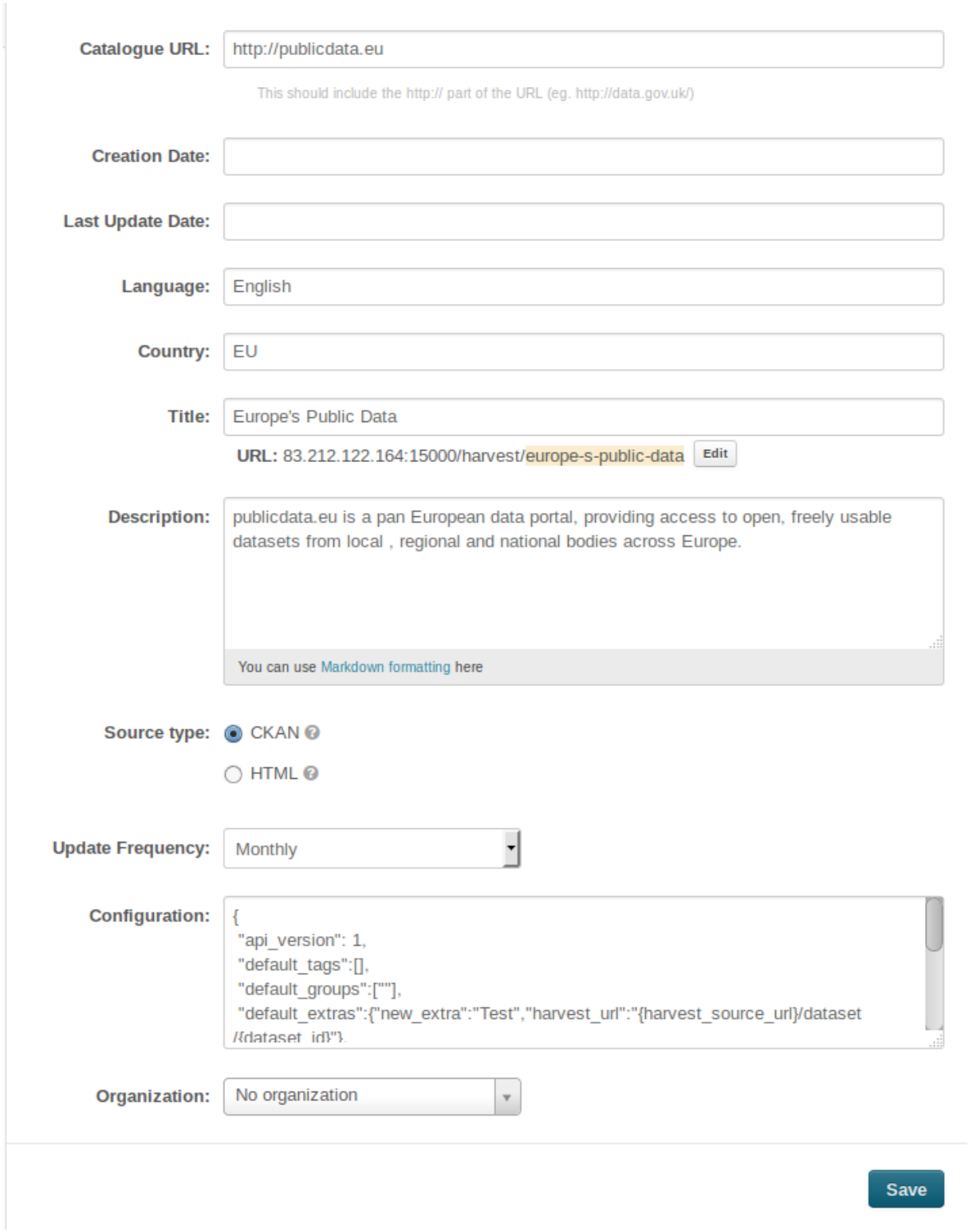

Figure 5: Catalogue registration and job configuration for the CKAN harvester

Figure 5 presents an already filled in form for a CKAN harvester.

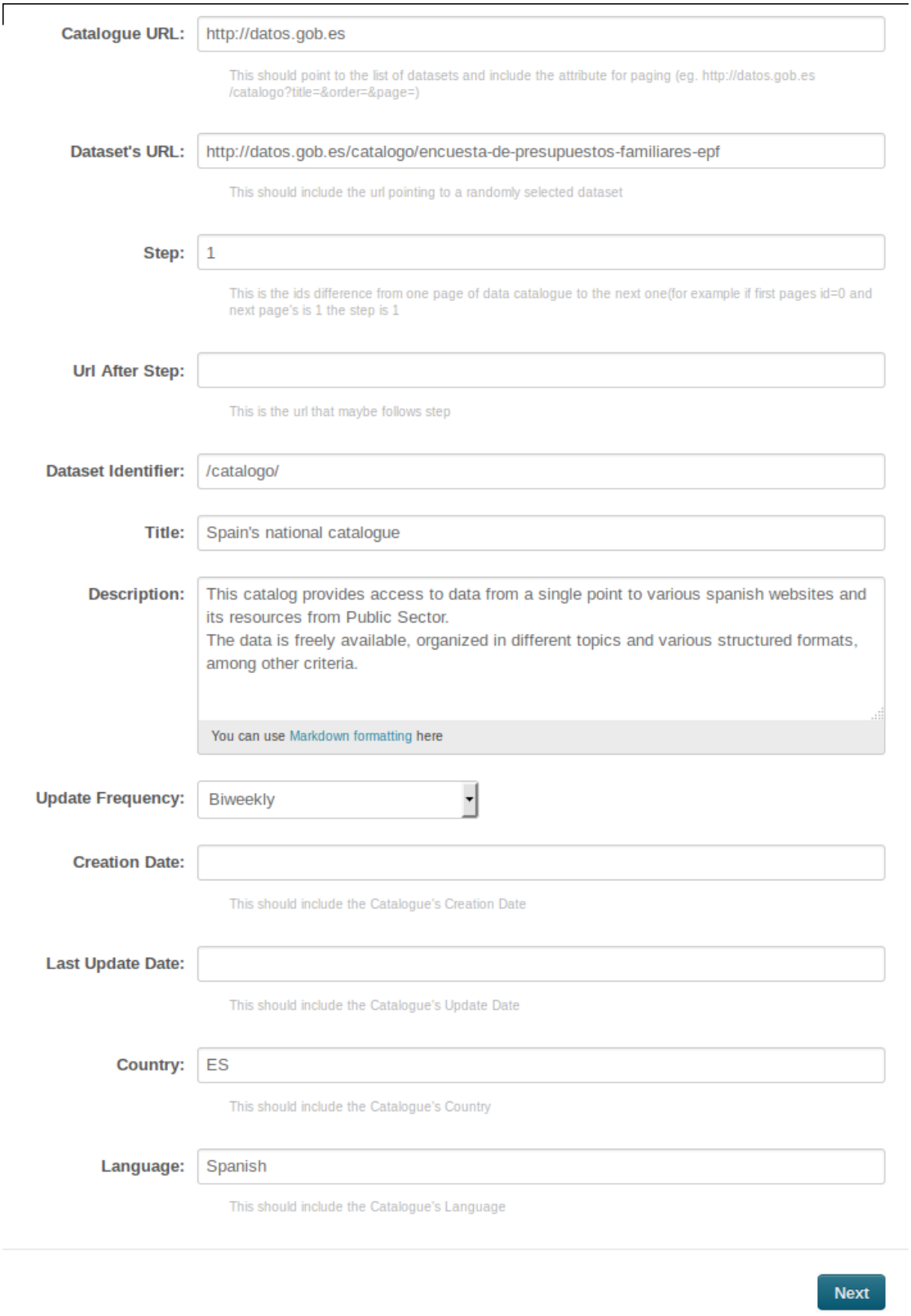

Figure 6: Catalogue registration for the HTML harvester

On the other hand, the job configuration for the HTML harvester is taking place in two phases, as shown in Figure 6 and Figure 7. Since an HTML catalogue does not provide any information about the metadata that it contains, we have to configure rules for the metadata extraction by filling in a second form, as displayed in Figure 7.

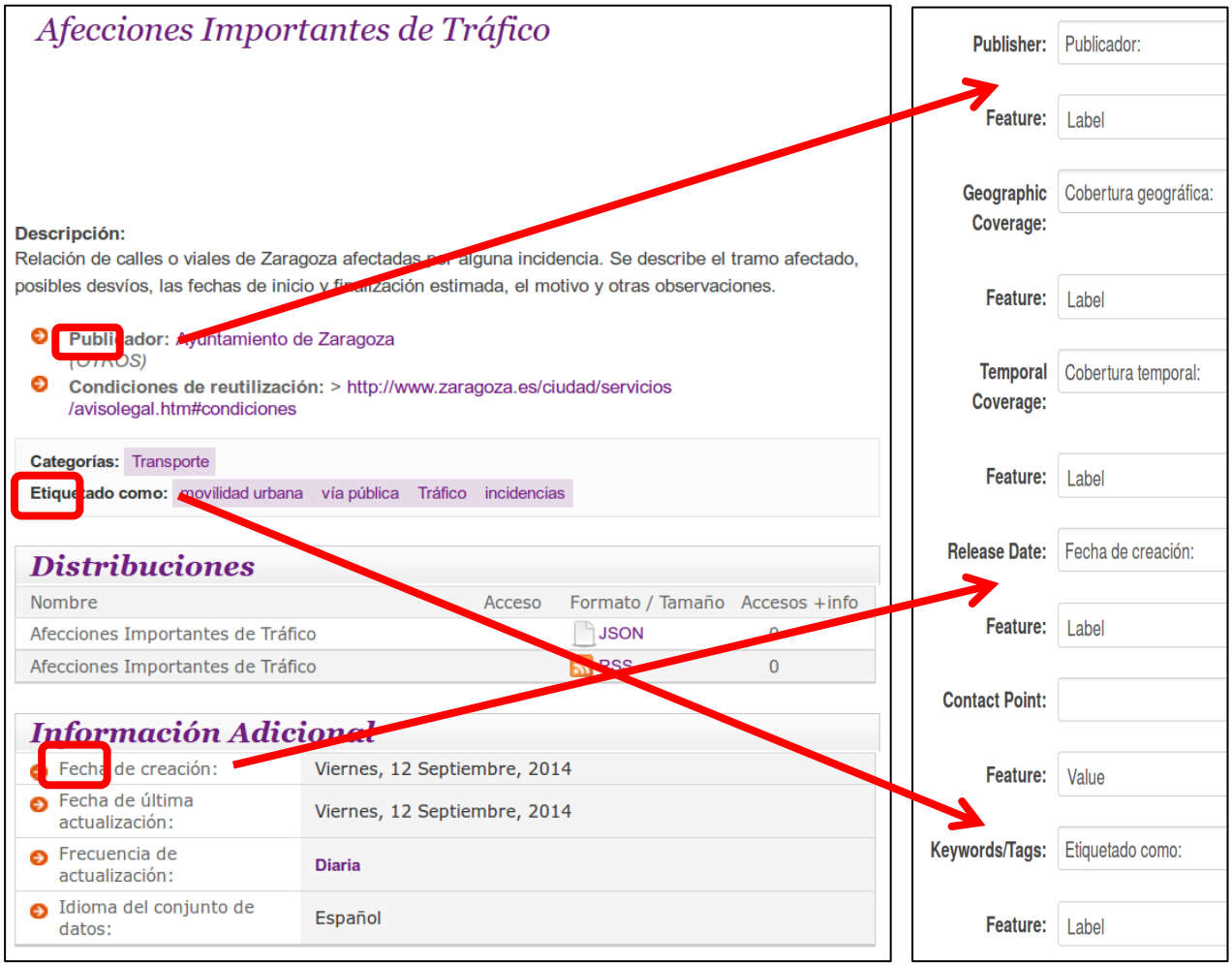

Figure 7: Job configuration for the HTML harvester

5. CONCLUSIONS AND NEXT STEPS

In this report, we have described the deployment of the first version of the ODM demonstrator. This includes components that support and integrate the basic functionalities of the system, in particular for:

- registering and configuring an open data catalogue for monitoring
- running the harvesting jobs to extract metadata from the registered catalogues
- executing a series of harmonisation scripts to reconcile metadata collected from different catalogues
- making the results accessible via a RESTful API.

The code of the ODM demonstrator is available in a public GitHub repository (https://github.com/opendatamonitor), where it will be maintained throughout the project, including also updated instructions for its use.

The main next steps for enhancing the functionality of the demonstrator includes the improvement of the management of the whole processing workflow, so that all steps can be executed automatically, supporting also the periodic execution of harvesting jobs. In addition, an administration panel will be developed to allow for monitoring and configuring the whole system in a more comprehensive way. This will also be extended to allow users to contribute to certain aspects of the process, such as to be able to view, modify and add mappings used for harmonisation of the metadata. Further improvements and enhancements will be performed based on the feedback that will be received by the users.