Prepared by

Søren Roug

Reportek

Administrator manual

February 2018

Version 1.2

Version management

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author(s) | Remarks |
| 1.0 | 2008-05-14 | Søren Roug | Initial version |
| 1.1 | 2010-09-14 | Miruna Bădescu | Changed domain names from \*.eu.int to \*.europa.eu, updated catalog indexes, added chapters on Reportek engine, Dataflow mappings, QA repository and wrote about remote converters. |
| 1.2 | 2018-02-12 | Michael Noren | Basic instructions for setting up dataflow mappings |

Contents

[1. What is Reportek? 4](#_Toc506199526)

[1.1. Dependencies 4](#_Toc506199527)

[1.1.1. Python dependencies 4](#_Toc506199528)

[2. Installation 4](#_Toc506199529)

[2.1. What is the envelopes.rdf script for? 4](#_Toc506199530)

[2.2. Converters 5](#_Toc506199531)

[2.3. Dataflow mappings 5](#_Toc506199532)

[1.1.1 Add or update dataflow mappings 6](#_Toc506199533)

[2.4. Reportek Engine 6](#_Toc506199534)

[2.5. QA Repository 7](#_Toc506199535)

[2.6. Authentication 7](#_Toc506199536)

[2.7. Authorization 7](#_Toc506199537)

[2.8. Permissions 8](#_Toc506199538)

[3. Usage 8](#_Toc506199539)

[4. Integration hints 8](#_Toc506199540)

[4.1. How can I integrate with ROD? 8](#_Toc506199541)

[4.2. How can I use my own obligations? 8](#_Toc506199542)

[5. Configuration 9](#_Toc506199543)

[5.1. Creating a workflow 9](#_Toc506199544)

[5.2. Moving a workflow from test site to production: 9](#_Toc506199545)

# What is Reportek?

Reportek is a Zope Product that implements the second generation of the EIONET national repository.

## Dependencies

Reportek depends on two lists that it must get from somewhere: A list of obligations and a list of localities. You can either get the list from ROD.eionet.europa.eu with XML-RPC or you can use the *SmallObligations* product to roll your own obligations and/or localities. See the file HINTS.txt for information

### Python dependencies

Reportek needs PyXML from http://pyxml.sourceforge.net/

pyshapelib from http://ftp.intevation.de/users/bh/pyshapelib/

shapelib from http://dl.maptools.org/dl/shapelib/

# Installation

If you have not already done it, first create a ZCatalog named Catalog. Keep the default indexes and metadata. Then add the following indexes:

TextIndex: *title*

FieldIndex: *activity\_id, actor, bobobase\_modification\_time, content\_type, country, getCountryName, id, instance\_id, meta\_type, dataflow, partofyear, process\_path, released, status, xml\_schema\_location*

DateIndex: *reportingdate*

KeywordIndex: *dataflow\_uris, years*

PathIndex: *path*

Then add as metadata: *activity\_id, actor, country, dataflow\_uris, id, instance\_id, meta\_type, path, process\_path, reportingdate, status, title*.

Otherwise search will not work. Extract Reportek to the Product folder.

Then there is a ZEXP file called Reportek-scripts. It contains the design template, the DTML code for search etc. Copy it to the import directory. Then import it into the toplevel Zope folder. Depending on your preferences, copy the scripts to where you have created Reportek Collections.

## What is the envelopes.rdf script for?

The envelopes.rdf is used by Reportnet's content registry to harvest deliveries made to the repository. It uses a format called RDF. If you do not want the content registry to harvest, you do not need to install it.

## Converters

Documents have the ability to display their content converted to text or HTML - really anything that is quicker to display than the native format. For this, it uses external converters. Look in Document.py to see what. If you have not installed a certain external program, then Document.py will fall back to the native format.

The folder *Converters* is automatically added in the Root folder after the product is installed. In order to use the converters installed on the server (local converters) you must add an object of type 'Converter' in the Converters folder specifying the path to the actual converter file, the type of the files that can be converted by it and the type of the output files.

E.g. if you have installed the pdftotext converter then type in the Converter path (including the arguments) something like:

'pdftotext -raw %s'

for Linux or something like:

"C:\ProgramFiles\pdftotext" -raw %s

for Windows.

Once a converter is added in the Report Document list, you will be given the possibility to choose among the available conversions (View document as.) for every type of document.

Aside from the local converters, Reportek also can use remote converters from a remote service that uses XML-RPC to expose the necessary functions. To hook up such a service, type the XML-RPC interface of the converter server in the *Remote converters* tab of the *Converters* folder, for instance <http://converters.eionet.europa.eu/RpcRouter>.

## Dataflow mappings

The *DataflowMappings* object is a special type of container, automatically created in the Zope root when the Reportek products is installed.

When XML files are used for the delivery of a dataflow, you usually want to restrict the types of file (based on the XML schema) that reporters should deliver. Do that by adding records in *DataflowMappings*, where you can also specify for each file type if it can be edited online using an XForm.

If the DataflowMappings contains a record for the pair obligation/file type, Reporters will be displayed the link to edit such a file via Webforms.

The dataflow mappings are used mainly in the Draft applications to keep track of which types of files should be delivered for certain obligation. For instance, you can loop the mappings for a given obligation and list the links to empty templates for each schema, or display the links to create/change XML files using the corresponding Web forms.

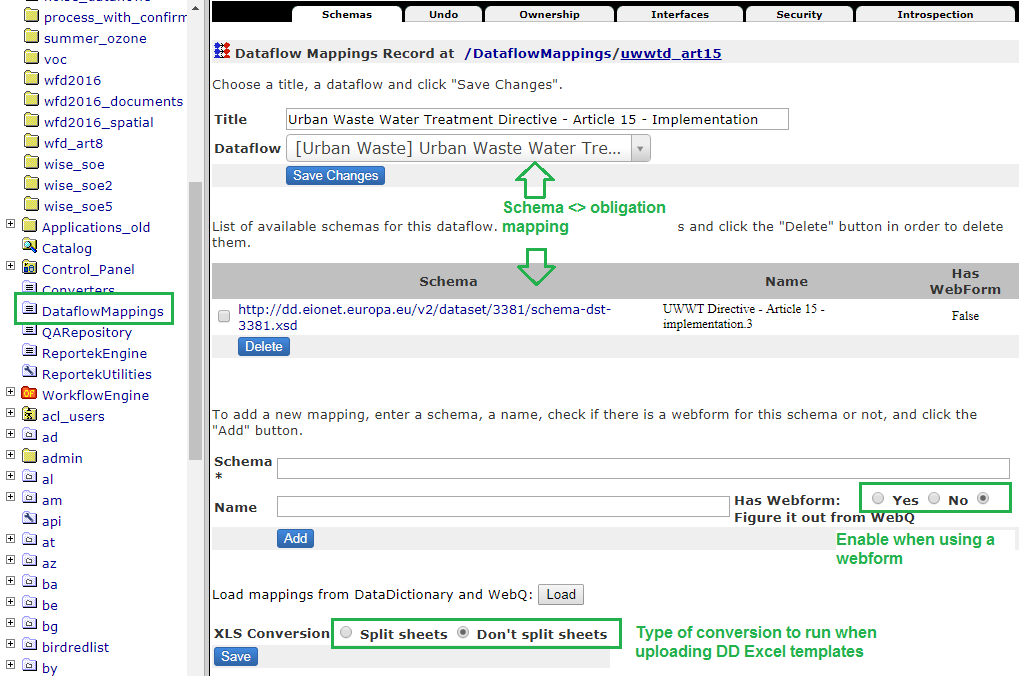
Moreover, QA buttons are shown *only* next to XML files for which there is a record in the DataflowMappings associated with the envelope's obligation. This prevents reporters to obtain positive QA results from files generated from old templates. However, it also means that website administrators must remember to delete the old dataflow mappings and create new ones when designing a new version of the workflow.

### Add or update dataflow mappings

Navigate to <http://cdrtest.eionet.europa.eu/manage> for CDR test, or <http://cdr.eionet.europa.eu/manage> for CDR to open the admin interface. In the left-hand menu, select “Dataflow mappings” to see the list of dataflow mappings.

Select an existing mapping to edit it, or select the top right button “Add dataflow mappings record” to add a new mapping. The information to provide is:

* ID: Unique identifier for the mappings record.
* Title: Name to explain the record, usually the dataflow name.
* Dataflow: Select the reporting obligation for the dataflow.
* Schema: The URL to the schema(s) to allow for the reporting obligation.
* Name: The name of the schema (will be the name of the link for a possible webform link in the envelope).
* Has webform: Select ‘Yes’ if there is a webform for this dataflow, if not selected, no link to the webform will show in the envelope.
* XLS conversion: If the dataflow uses a Data Dictionary Excel template for reporting, this setting determines if the Excel files uploaded to CDR should be converted to one XML-file per sheet (i.e. DD table) or one file for all sheets (i.e. the DD dataset).



## Reportek Engine

The object *ReportekEngine* is automatically created in the Zope root when the Reportek products is installed. It holds various settings that allow all Reportek processes to function:

* links to the Web Questionnaire service
* path of the Quality Assessment application
* settings for the Unified Notification System (UNS)

It also implements functionality that mostly allows end users to overview their activities:

* *Global work list* - the TODO list for all users of the website
* *Country reporters* - lists the users that have the local role of 'Reporter' in each country
* *Autocomplete envelopes* - searches the envelopes that have been in a given status for more than a given number of days and which are tagged with a certain obligation and allows Managers to complete the selected ones.
* *sitemap* - the collapsible site structure

and exposes an API for the site level.

## QA Repository

Like converters, QA scripts can be local programs or remote QA scripts from a remote service. At installation time, Reportek automatically adds the *QARepository* special folder, which is a container for local QA scripts and points to the QA application thatinterfaces with the remote QA service.

## Authentication

You can use Zope's builtin User folder, or you can use *LDAPUserFolder* from [www.dataflake.org](http://www.dataflake.org/). This will hook you into the CIRCA site directory, your own organisation's or EEA's. That is up to you.

## Authorization

Four (Five) types of users are foreseen:

* The anonymous visitor, who can browse released reports and download the public files.
* The trusted client, who can browse released reports and download public and protected files.
* A collection administrator, who maintain the hierarchial structure, but who doesn't upload any reports.
* A release-coordinator, who releases an envelope to the public. The public must be confident that they are not downloading an incomplete report, so a release/-revocation is logged in the activity log. Once a report is released, it is no longer possible to upload files. If a mistake is found then the release-status can be revoked.
* A reporter, who creates the actual reports. A reporter can request a report to be released when he is finished.
* And then there still is the Zope Manager, who can fix everything if/when the security paradigm doesn't cut it.

A user can be one, more or all of these roles at the same time.

Additionally, a user can only delete objects he himself have created. And in the case of files, only if the envelope is not released.

## Permissions

There are seven permissions:

* Add Collections, which is given to the collection administrator
* Change Collections, which is given to "owner". Thereby the collection administrator can only modify the collections he has created.
* Add Envelopes, which is given to the "reporter". This allows people to create envelopes. If you have the right to create an envelope, you also have the right to add files into it. If there are certain parts of the hierarchy that a reporter should be restricted to, then give him the permission as a local role, or create a user folder on that level.
* Change Envelopes. Give this permission to "owner" to let a "reporter" fill his own envelopes or give the permission to "reporter" to let all reporters modify all envelopes.
* Add Feedback given to "Client" to be able to add feedback for release of envelopes
* Delete Objects. Typically give this permission to "owner" and maybe "release-coordinator" or "collection administrator".
* Release Envelopes. Can be given to "owner", "reporter" or some other class of users.

# Usage

The purpose of this product is to make it easy to store the obligatory environmental reports from each country. There are several organisations who receive these reports and for each organisation there are several reports covering subjects such as water and air quality.

To deal with the many reports we let the user organise them in a hierarchical set of collections of his own choice. At the leaves of the structure are the envelopes. They contain all the files and necessary meta-data. The collections also have meta-data, but they only serve as default values for envelope creation.

To prepare a report you first create an envelope. Then you upload the files and finally you release it for the public.

# Integration hints

## How can I integrate with ROD?

The integration with ROD in this version limited to getting the list of obligations - also known as activities or dataflows for historical reasons - and countries. This is done with XML-RPC. You install the *XMLRPCMethod* product from Zope.org and create an XMLRPCMethod called dataflow\_table with the URL <http://rod.eionet.europa.eu/rpcrouter>,

the method name WebRODService.getActivities and set up caching with say 86400 seconds of max age.

## How can I use my own obligations?

You install the *SmallObligations* product.

There are two lists that Collections and referrals need: a list of localities for coverage and a list of obligations.

The localities can be set up to reflect your toplevel collections with this DTML method called localities\_table in the toplevel. This will only work if none of your localities are countries that exist in ROD.

<dtml-call "REQUEST.set('result',[])">

<dtml-in "PARENTS[-1].objectValues(['Report Collection'])" sort=title,id>

<dtml-call "result.append({ 'iso': \_.string.upper(id),

'name': title,'uri': absolute\_url() })">

</dtml-in>

<dtml-return result>

A slightly better approach is to use the python script called localities\_table\_ov in Reportek-scripts. But it is too large to show.

You can also use the SmallLocalities from the SmallObligations product if you need a little more functionality. In this case, you create e.g. a SmallLocalitiesFolder called "regions" in the toplevel folder. Then you add the localities inside as "Small Locality" objects. Then you create a DTML method called localities\_table with one line:

<dtml-return "REQUEST.PARENTS[-1].regions.list\_localities()">

Again, there is a python script called localities\_table in Reportek-scripts. For obligations you must use the SmallObligations product. You then create a SmallObligationsFolder in the toplevel and create obligations under it. You then create a DTML method called dataflow\_table with the one line:

<dtml-return "REQUEST.PARENTS[-1].obligations.list\_obligations()">

Again, there is a python script called dataflow\_table in Reportek-scripts.

# Configuration

## Creating a workflow

This entire process is detailed in the document [How to create a dataflow in Reportnet.](http://www.eionet.europa.eu/reportnet/How%20to%20create%20a%20Reportnet%20dataflow.pdf)

After creating the file templates, converters, QA scripts and Xforms, the Reportek-related steps should be first done on the test site (e.g. [http://cdrtest.eionet.europa.eu](http://cdrtest.eionet.europa.eu/)), the workflow reviewed there by experts using test data and then moved to the production site ([http://cdr.eionet.europa.eu](http://cdr.eionet.europa.eu/)).

## Moving a workflow from test site to production:

1. Move the methods and Python scripts from the Zope root
2. Copy the local converters over
3. If it uses Web forms or the automatic QA service, move over the DataflowMappings objects
4. Call <http://cdrtest.eionet.europa.eu/WorkflowEngine/exportToXml>, edit the file and cut all the other processes than voc\_directive. Then import in to CDR at <http://cdr.eionet.europa.eu/WorkflowEngine/importfromxml_html>
5. Do the process mapping in WorkflowEngine