An Introduction to the Aggregation database

The aggregation database is planned to be an extension of Reportnet's search engine, the socalled Content Registry (CR). It is going to serve two purposes:

1. To discover and keep track of SEIS datasets. It does not matter what format these datasets are. MS-Excel, HTML etc. are all equal.
2. To understand the SEIS datasets that are in XML format. It will do this by transforming the content to a quadruples structure and storing it in a database.

The system is not going to be the only way to get access to the Reportnet datasets, but to be an extra tool in the toolbox.

Discovering the dataset

Datasets tend to be only numbers. Unlike Google, that mainly harvests webpages, a SEIS search engine can't get information from the content of the files. The aggregation database will use two mechanisms to discover relevant datasets.

1. A collaborating node provides a *manifest* of the relevant files existing on its own website. The manifest lists location and any metadata the provider has on each file. This manifest must be downloadable over the Internet and will be periodically harvested by Reportnet's CR. When the file occurs on the manifest, it is then discovered by Reportnet.
2. Users find the file with Google, surfing the web or the owner of the dataset *registers* it at Reportnet's QA Workbench (QAW). This site is for the registration purpose nothing more than a special collaborating SEIS node that provides a manifest to the CR.

Once the content registry has discovered the existence of a dataset, the users can *amend* the available information on the file by adding metadata to it in QAW. There are several relevant pieces of metadata to enter; title, tags, coverage – both spatial and temporal, what purpose the dataset was created for, what methods were used, its quality etc. All these can be entered at the QAW. But a glowing review of a dataset can only be trusted if we know who wrote it. As we'll see, QAW has metadata on the metadata.

What sites will have a manifest file?

As many as possible. Relevant datasets aren't just the ones that the member countries provide. We want to be able to combine data. If we have a list of airport locations and how many flights they have a year – that's relevant. Dams and their locations. Basic statistical data on countries – number of inhabitants, size, GDP – is needed to understand other data. Eionets database of reporting obligations is a database, but every obligation has a factsheet page, and the obligation *looks* like a file. That database should have a manifest because Reportnet deliveries declare in their metadata they are responses to obligations (implying a relationship).

Manifest files don't have to follow just one format. There is room for integration with the INSPIRE discovery service, and there is room for the INSPIRE data formats.

Tracking the dataset

Once a dataset has been discovered by Reportnet's CR, it starts to check periodically whether it is still at the remote location, and whether it has been modified. We are planning to check every 6 weeks. If a dataset is found to be unavailable its record in the search engine is marked as stale, and a notification is sent out via the Unified Notification Service (UNS) to whoever is subscribed.

Examples of tracking

ROD contains references to repositories, legal instrument documents and guidelines. Until now we have used special purpose-built software to check the links and email an administrator when a link went dead. With the new way to track, a simple feature will be added to ROD that provides a manifest of remote references. The CR will automatically check the links, and all the administrator has to do is to subscribe to UNS for dead links in ROD.

Precisely the same mechanism can be used in many other contexts. One is the SERIS database at http://www.eionet.europa.eu/seris.

Using the dataset

All the registration and tracking isn't of much use unless the users can find the datasets again in CR. The plan is of course to use the metadata to search for the datasets. The user types in 'soil' and gets what's known. The usefulness of the system hangs on getting as much metadata as possible. We will therefore encourage people to 'improve' the metainformation as much as possible. E.g. When the user has successfully found some relevant datasets we'll make it easy to enter more metadata by tagging all the hits with one tag.

The user has found some datasets – then what? In Google he can click to the webpage. He'll be able to do the same here. But we're also considering keeping a copy of the dataset when the CR first discovers it, because it will make it possible for us to provide the dataset in other formats. The user might not be able to use a Shapefile. We can make it viewable as an image.

Understanding the dataset

If the dataset is in a known XML format, CR will be able to understand the *content* of the dataset. A file with a known XML format is one that has a schema identifier; or has a persistent location on the Internet, where the XML format has been discovered in some other way. The way to understand the content is to *transform* it and put it into a database.

Let's imagine that Austria reports two new Airbase stations. Number 32301 and 32302 They could store the information in an XML file called *stations.xml* like this:

```
<stations xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://air-
climate.eionet.europa.eu/o3excess/stations.xsd">
  <station>
    <local_code>32301</local_code>
    <name>St. Pölten - Eybnerstraße</name>
    <longitude_decimal>15.63166</longitude_decimal>
    <latitude_decimal>48.21139</latitude_decimal>
    <altitude>270</altitude>
    <station_type_eoi>Industrial</station_type_eoi>
    <area_type_eoi>urban</area_type_eoi>
  </station>
 ...
</stations>
```

It will be simpler to explain if we show the data as a table like show below. The transformation will take any XML content and store it in a database table with only four columns called Subject, Predicate, Object and Source. Every record gets a *type*. This one is called "Station".

Station structure as a table

Identifier	local_code	name	...
#32301	32301	St. Pölten - Eybnerstraße	...
#32302	32302	Europaplatz	...

Triples

Subject	Predicate	Object	*Source*
#32301	type	Station	*stations.xml*
#32301	local_code	32301	*stations.xml*
#32301	name	St. Pölten - Eybnerstraße	*stations.xml*
#32302	type	Station	*stations.xml*
#32302	local_code	32302	*stations.xml*
#32302	name	Europaplatz	*stations.xml*

Now imagine what will happen when CR loads files of the same format from several locations. The data is automatically aggregated. Hence the term aggregation database. You can even load files with conflicting information on the same stations. They are kept separate because the Source values will be different.

The mechanism makes it possible to have other sources add columns (predicates) to a table if the key (subject) is the same.

Querying the database

The question is how do we make use of it? Here we must implement a query language in CR. If we start by something simple, such as a query on all records of type Station, CR will then find all rows with the predicate=”type” and object=”Station”. It would then for all the subjects it has found look up which predicates exist and show a table.

Identifier	Local code	Name	...
#32301	32301	St. Pölten - Eybnerstraße	...
#32302	32302	Europaplatz	...
...	...	...	...

The data could then be exported as MS-Excel, MS-Access or whatever else is convenient for the user.

Trusting the dataset

How would you know that a dataset you’ve found is trustworthy? As always, you look at the website that hosts it. Is the site to be trusted? But for datasets, there is more. Is it produced for a purpose compatible with your own needs? Is it of sufficient quality? The QAW website will store all the known metadata, and when you know the URL of a dataset you can query the QAW for the metadata. But QAW will also store where the metadata came from. For example if someone has registered *stations.xml* with QAW and entered information on the methodology, QAW will store which user wrote it.

But how do you trust aggregated data? The same way. By inspecting the source. But since it is aggregated, there must be a source on *every* field. You will be able to look at the data separated, or *overlapped*, in a way that new data overwrites old data for the same identifier. A list of overlapped records could look like this, using colour codes to show the source of the fields:

Identifier	Eol	Updated	Local code	Name	Long	Lat
#32301	AT44	2008-05-28	32301	St. Pölten - Mühlweg	15.62880	48.21194
#32302	AT220	2007-05-31	32302	St. Pölten - Europaplatz	15.61111	48.20999
...		...	...	...	...	

	Source: http://www.uba.at/stations.xml	[Turn off]
	Source: http://air-climate.eionet.europa.eu/airbase-stations.xml	[Turn off]
	Source: http://www.uba.at/station-updates.xml	[Turn off]

You will be able to inspect the metainformation for each source, turn individual sources on and off and when satisfied, export the data as MS-Excel, MS-Access etc.

Gapfilling

You can weed out bad data by excluding the source, but what if you want to *correct* some data? This can be achieved by creating your own source. There are two ways; copying an existing source or creating an empty. In both cases, the source will be stored on QAW.

Identifier	Local code	Name	Date	Longitude
#32301	32301	St. Pölten - Eybnerstraße	2005-10-8	15.63166
#32301		St. Pölten - Mühlweg	2008-6-18	15.62880
#30299	30299	Gent	2004-11-12	3.733333
#42882	42882	Köln	2001-4-14	6.966667
#32301			2008-11-27	15.63100


Your gapfilling layer

What you normally will see is that you will be able to change any value in the table, and the change will be stored in your gapfilling layer. Others will see your layer as yet another source to consider for the merge.