European Environment Agency

Database Feasibility Study

Reportnet 3.0 project

Version: 1.1
Date: 12/02/2019

Authors: Jonathan Maidens (EEA) & Jan Bliki (EEA)
Contributors: Miguel Villafranca (Tracasa) and Igor Trebol (Tracasa)
under support from the Energy Union Governance project

Document History

Version	Date	Author(s)	Remarks
0.9	20-09-18	Jon Maidens	First draft released to Trasys
0.91	24-09-18	Jon Maidens	Further edits incl feedback from Trasys
1.0	09-10-18	Jon Maidens	First version
1.1	12-02-19	Jon Maidens	Tidy for release as PDF

Kongens Nytorv 6
1050 Copenhagen K
Denmark
Tel.: +45 3336 7100
Fax: +45 3336 7199
eea.europa.eu

Contents

1 Study definition

1.1 Overview

The Reportnet 3.0 project is the response to the driver to streamline environmental reporting, in which Reportnet plays a key role. The goal of the project is to design a new reporting system that will integrate new ideas about reporting, take into account national capabilities and produce a platform that can support the new challenges in reporting for the years 2020 to 2040. Reportnet has been developed since 2000 and has been in operational use since 2002. I.e. the initial design is now ~20 years old. Over time, the reporting needs have changed and Reportnet has been modified for special-case exceptions so many times that the original design is beginning to be compromised.

This document is 1 of 2 feasibility studies which are being run under the Reportnet 3.0 analysis phase. This document tests the feasibility of replacing the current file-based storage with a database storage platform to support the services and workflows supported by the Reportnet platform. The second feasibility study is the INSPIRE integration feasibility study.

1.1.1 Document structure

The document is structured in four sections. The first section (this section) provides the problem statement and the proposed way forward; the second section an evaluation of critical capabilities; the third section a technical comparison of two database types; and, the fourth section tests prototypes using wireframes how a user interface and work flow would look using the database-centric platform.

1.2 Background

1.2.1 The current situation

For a comprehensive overview of the current Reportnet platform architecture and example business process workflows, then please refer to the following two documents:

- Reportnet Business Process Evaluation
- Reportnet AS-IS technical architecture

Reportnet 2 uses files as the basic source of data delivery and data storage. Files are a convenient way to transfer and store data and Reportnet architecture and services are built around this format. Reportnet is agnostic to the format of the information delivered, in the sense it allows delivery in any file format. When the contents of the reports have to be automatically processed, for example to perform quality control or to produce a European dataset, the underlying format needs to be XML. Most reporting obligations consist of structured information and are therefore data flows are based upon XML. The XML has a dual use, for the reporting entity it provides a detailed specification of the data to report, and in Reportnet it allows for assessing that the reports delivered follow the specification.

While exchanging information in XML-format is a widely used practice, the nature of the reporting obligation and the capability of the reporting entities usually leads to supplementary

tools being provided for the practical reporting. Such tools commonly provided are for example web questionnaires, Excel templates, Data Exchange Modules and QA scripts.

1.2.2 Drawbacks of XML-file reporting

The XML-schema is advocated as the right solution because it is an open format and has a built-in validation mechanism. However, there are many ways to implement a data structure into an XML schema, and as most of the schemas are built manually, it's natural each and every schema is different when you produce them manually by different consultancy companies. The reporting obligations are generally complex with data structures of many nested levels which has the implications for single obligations having multiple schemas of high complexity which few people have an overview of (see WFD 2012 reporting definition).

This isolated, manual schema development has a significant impact on everything else built to support this structure in a Reportnet workflow. All components built on top become manually tailored as well. And to help the MS convert their data to XML to manage this complexity, we create external tools for them (DEMs), which in the end costs us quite some money to develop and maintain.

Another issue is reporters generally don't understand an XML-schema (data specification), nor can interpret the XML validation results. This is particularly problematic for voluntary reporting flows where there is no leverage to demand compliant XML when the risk is the data will not be reported at all because the cost is too high. In addition, the data reporters usually have their data in a database or system where we expect them convert this data to the XML format for delivery and storage in Reportnet, before we (EEA) converts the XML file back to a tabular format again, costing both ends quite some money.

Additionally, it is very difficult in a file-based storage system to determine if a file (report in an envelope) substitutes or compliments another, and whether it's for the current reporting cycle or a re-submission of a previous one. Similarly, it requires insider knowledge to know whether a report was accepted at the end, or its just ignored or superseded information lying around in CDR.

To be clear, we are not saying XML-files are an invalid format for data transfer (web services), but addressing significant drawbacks to using XML-files as a storage format.

1.2.3 Value added benefits of moving to a database-centric platform

With the impact of the change from a file-centric to a database-centric platform across all the reporting processes, the goals of this new approach seek to generate added value in the following key aspects:

- **Better user experience and cooperation** - An easier user experience for the reporters and the increased communication among actors could reduce the number of issues the reporters and the developers have to face.
- **Agile** - Related to the above, target the early prototype to ease and lower the cost of updating data models, leading to an early detection and correction of problems.
- **Efficient** – A positive impact on the required time to create final products. Early in the dataflow cycle it would already be possible to provide product owners with products

making use of the data, even if data is not available for all countries. This will allow us to have these products ready for publication once the reporting is finished.

- **More accurate results** - With the early product prototyping and with data being available to thematic experts sooner than what is the norm, will also influence on the cycle time for reviews on data to be generated. Thematic experts will have instant access to the latest data delivered and that has been accepted.
- **Scalable** - The new methodology will decrease the costs for a new dataflow, even more so when where the Member States need to refer or resend previous reported data.
- **Integration** - The record-based database storage would allow us to fully embrace server to server integration such as a record based REST-API for developers. Provide more freedom to the countries without adding complexity to the EEA. It could improve the ways some checks are made, for example, duplicate checks under database rules. The ability to link different fields between related dataflows.
- **Easier managerial overview** - This approach also benefits in the long term for managerial dashboards and identifying gaps and historic trends.

1.2.4 The way forward

The premise for this feasibility study to ascertain whether reporting directly into a database instead of using the file storage format, would achieve the following:

1. Alleviate the currently described problems reporters and requestors face on a daily basis
2. Allow modern system to system integration (Web services like INSPIRE).
3. Improve the reported quality while reduce the rigid workflow enforcements not always fitting every countries setup.
4. Be a platform for realising the identified value-added benefits to take the reporting forward for the next 10 years.

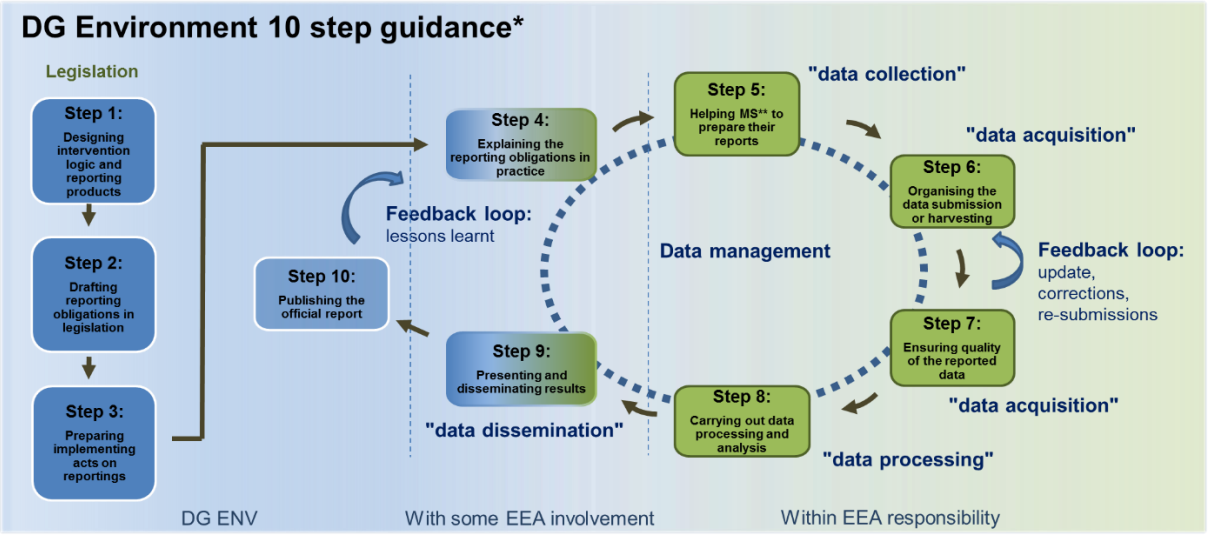
1.2.5 File and database comparison

The following table weighs the strengths and weaknesses of using files as a storage format against a database. The key difference is the management of the data, which in a database is at record level, whereas with a file data can only be managed at the file level. The implications of this is that when you want to change one piece of data, it is necessary to resubmit a whole file, a situation we see in Reportnet today. It is also not possible to establish relations between files so each package of information is independent until brought into a working database.

	File	Database
Strengths	1. Save and download single file is simple 2. Migrating data is easy process 3. Cost effective 4. [XML] Open format 5. [XML] Ability to query data within file (with limitations) 6. [XML] Ability to ensure data within file is managed to ensure its integrity and quality (with limitations)	1. Ability to manage data on the record level 2. Ability to have data integrity checks 3. Flexible and widely understood query language to retrieve data (SQL) 4. Ability to relate stored data (joins etc) 5. Ability to query data efficiently (indexes) 6. Ability to handle multiple users 7. Ability to establish users, roles and permissions 8. Ability to handle many records – large amounts of data
Weaknesses	1. Only manage data on the file level 2. Only one way to organize – directory and filename 3. Only one user at a time can modify a file 4. Unable to establish relationships between files 5. [XML] Integrity checks and querying require external scripts/technologies (XQuery) 6. [XML] Requires a well-defined schema and expertise to develop 7. [XML] Difficult for most people to work with 8. [XML] Schema structure can be implemented in multiple ways, and independently, meaning lack of consistency and re-use	1. Database model design requires expertise 2. Additional license and server costs 3. Need to manage upgrade cycle 4. Performance, if not managed properly 5. Security, if not managed properly

1.2.6 The implications to the reporting process

Changing the storage format from files to a database has implications across the reporting process. The following reference model shows where Reportnet as a platform underpins much of the steps marked in green.



The table below breaks down each step into further sub processes and provides a more detailed understanding of the tasks within each. The process is as it looks, very sequential with only the steps 6 and 7 iterating back and forth. This siloed approach is another aspect which the new platform needs to address:

Step 4 Explaining the reporting obligations in practice	Step 5 Helping MS to prepare their reports	Step 6 Organising the data submission or harvesting	Step 7 Ensuring quality of the reported data	Step 8 Carrying out data processing and analysis	Step 9 Presenting and disseminating results
4.1 Define reporting requirements	5.1 Establish helpdesk	6.1 Member State delivery	7.1 Execute automated Quality Control	8.1 Merge data in European datasets	9.1 Publish online data and map products
4.2 Plan for other obligation e.g. INSPIRE	5.2 Develop reporting tools i.e., DExM, Forms	6.2 Member State resubmission	7.2 Execute manual Quality Control	8.2 Create European dataset products	9.2 Create Implementation Reports
4.3 Design data model	5.3 Implement Quality Control tools	6.3 Monitor status	7.3 Publish Quality Control dashboards and data visualisation		9.3 Create Evaluation Reports
4.4 Define data schema	5.4 Configure Reportnet for reporting				9.4 Undertake Reviews
4.5 Define dataflow	5.5 Configure Reportnet for Member States				9.5 Communicate
4.6 Develop guidance document	5.6 Help Member States in preparation				
4.7 Define Quality Control rules					v3.4

If we take each step individually, then we can illustrate the implication of a database-centric storage platform:

Step 4 - Design starts as a paper exercise to define the reporting requirements before it is used to create a data model which is then translated into flat file format (XML schemas). This step requires a lot of collaboration between the various actors to agree the scope of the reporting and then defining the structure in the reporting format, usually XML. Moving to a database-centric platform will impact the reporting design process as it will change the way the reporting format is designed and tested.

Step 5 – The creation of tools and configuration of the Reportnet 2.0 reporting environment is based around the file handling (XML schema) and the tools to translate and validate it. Moving to a database-centric platform will impact the reporting preparation process the move away from XML fundamentally changes the way validation, workflow, tools and outputs are defined and configured.

Step 6 – The process for MS to submit their data to Reportnet 2.0 is based around the file delivery process and the implications for their own systems to be able to export data to the XML delivery format. Moving to a database-centric platform will impact the reporting delivery process as the delivery format will change.

Step 7 – The process for the data reporters and requestors to validate the reported data in Reportnet 2.0 is based around the XML delivery process for which there are currently many workarounds in place – for example the use of FME and Tableau dashboards. Moving to a database-centric platform will impact the reporting quality control process and technology.

Step 8 – The process for the creation of European datasets and other output products is based on merging all the reported XML into a single database in the first part and secondly there is a data cleanse step before a European product is released. Moving to a database-centric platform will impact the reporting data processing and analysis process, as it will impact the workflows to create these European datasets.

1.3 Study approach

1.3.1 Platform for Evaluating Capabilities

We have identified a platform to develop a working prototype application handling familiar data. The technology employed will serve as a guideline for the actual future implementation. The selected technology to help in exploring the database-centric approach is called Airtable. It was selected because it met the following criteria:

- Web platform
- Collaborative
- Tabular – capable of relating tables

This feasibility study is not endorsing this product as the future platform nor does we find the platform able to support the requirements for a new platform, but we make use of this existing technology to explore the questions we want to test to understand how a database-centric platform would address our issues and serve our needs. The feasibility study is trying to cover a new technical approach and as well a user collaborative approach that should give a more agile and effective process.

We have used Airtable to help test and visualise scenarios. To integrate this with the study analysis, we have created short videos showing the steps followed. Links to these videos are directly referenced in the text within the 'demonstration material' section. The full list of videos and reference to the section where they are referenced can be found in the Appendix.

2 Capabilities evaluated

2.1 Step 4: Explaining the reporting obligations in practice

2.1.1 Evaluation 1.1: Can we design a data model for data delivery using a web based user interface?

Can we avoid the creation of XML schemas and go instantly into the creation of an input tool while the system designs the appropriate data structure.

Reportnet 2.0 Situation

XML allows build-in validation on structure and content. There are hundreds of ways to implement a data structure into an XML schema most of these schemas are build manual. It's natural that each and every schema is different when you produce them manually by different consultancy companies. This has a big effect on everything else that is built upon this structure. All components build on top becomes manual tailored as well. This work is also performed isolated (by external consultants) creating a very big risk to go into an academic design that becomes difficult to implement.

Demonstration material

- The following video is an example on how you can produce a data structure (WFD-sample) from a web based interface (See video). This video demonstrate the creation of tables, fields and altering of data types. It also demonstrate that this structure can be instantly used.
- In the following video, we see how a new table on the platform can be created from an existing spreadsheet (See video)
- In the following videos, we see how a collaborative platform allows more than one actor to work together to develop a structure to support a reporting obligation. In this first video (See video) a new structure is created with some test data which is shared. The structure is then updated based on the feedback received. In this second video (See video) the creator shares with a collaborator who directly edits the proposed structure. In the third video, a dataset is shared read-only and the collaborator makes a copy, proposes some updates and then shares the resulting dataset back (See video).
- The following exercise shows how a form can be produced (See video). This video demonstrates how a look and feel and description to the reporter can be improved and how we can quickly test out reporting interfaces whilst designing the data model.

Findings

1. When we can provide a graphical interface that is well integrated with a database engine we can dynamically produce any database structure suitable for delivering data.
2. A platform such as Airtable demonstrate that sufficient functionality can be delivered with an approach like this. Key functionality for a collaborative platform is sharing, rights management and inline communication tools.
3. A data structure automatically created will provide systematic methods that allows to simplify any further implementations build on top no matter what data flow we implement.
4. The data structure produced must be instantly usable for input to amplify the direct benefit to end users. Something we don't have in the data dictionary of Reportnet 2 today.

5. A further consideration is it's important to freeze the data structure at some stage to stabilize the further implementation by countries. It is not advisable to change the core structure too often as this makes it hard to deliver a consistent European dataset. But some aspects such as additional validation could be constantly improved during the reporting process.

Conclusion

A systematic data model allows for a systematic form builder, systematic web services, systematic validation and file import and export for all dataflow. A well-defined web interface with an in-depth integrated database management system gives a far better platform for data reporting then the conventional XML-formats we use today.

It's a must that our infrastructure allows open data access but that doesn't mean that the internal structure of Reportnet 3 must be based on XML. It means that our export functionality and web services must provide open data accessibility.

2.1.2 Evaluation 1.2: Can we manage relationships between tables (1-n,n-n,n-1)?

How could relationships between identities be managed and can this be done in a simplified and easy to understand mechanism.

Reportnet 2 Situation

Reportnet 2 deals with relations of tables using the XML nesting technique. One big document that has nested elements inside the document itself. To verify relationships between external sources; Such as lookup tables and other XML's; XQuery is used and is executed inside Reportnet 2 as a script during the validation process.

Demonstration

- The following video demonstrates how Airtable has simplified the concept. The user can "link to another record" and point to a table and it related item. The user can then decide on table and field level if the content does "allow linking to multiple records" by simple checkbox. This technique allows all possible relationships (1 to 1, 1 to Many, Many to 1 and Many to Many) between tables (See video).
- The following video demonstrate simple lookup that could be a hardcoded list. This demonstrates that not all (simple lists) must be represented as a table relationship reducing the overall data structure complexity (See video).
- Another video demonstrates that relationships can be imported. As this example shows we can keep linked fields during an import of excel/csv (See video).

Findings

1. Handling relationships between entities is crucial and essential to ensure a consistent dataset but are creating most of the complexity in a data flow.
2. To produce a consistent dataset both data requester and data provider must have these relationships implemented inside their databases.
3. XML allows to have the relationships embedded inside the structure. In principle this is redundant because we use XML only as a delivery format between the receiver's data base structure and provider's data structure. We believe that the complexity grows beyond the benefit it provides. It also reduces the data provider's delivery options significantly.

4. Keeping the data delivery format flat and simple and monitor the validation on both receiver and provider allows for easier to implement data delivery and multiple import mechanisms. The data provider can upload multiple files or use a combination of mechanisms to deliver the dataset.
5. A record based database infrastructure would simplify the implement control over relationships between tables.

Conclusion

Keep the delivery format simple and flexible because it reduces the learning curve for the data provider and receiver. Not embedding the validation of the relationships within the format provides additional loading methods to the data provider. Any system holding the data (Provider side and Receiver side) must have a built-in consistency making it redundant on the transfer format.

2.1.3 Evaluation 1.3: Can we alter the structure of the database while keeping existing data?

In this situation we want to understand the impact of changing data structures while data is entered. To what extent should we keep structural change open during the data flow process?

Reportnet 2 Situation

In Reportnet 2 this is performed by altering the XML schema and provide a new schema. Because this is file based, multiple versions become operational for some time and is hard to debug and explain to data providers. This process is very slow and cumbersome and usually needs different technical skills involved.

Demonstration

- We start from a database that has already been populated by one (or more) of the methods above.
- A user (could be a reporter, or data custodian) creates a copy for her experiments.
- She adds two new fields, and changes a relation from single to multiple.
- She adjusts the data in the new fields, but the existing records are otherwise kept.

Findings

1. When input and design is integrated the impact of change on existing data is instantly visible. This helps understanding the impact changes will have between the old and new structure.
2. The coherence of implementation and functional possibility is more assured if the system that is going to receive the deliveries is used from the moment of design.
3. Sometimes the structure is so different that an entire new structure would be required. In such cases there is no benefit to be found.
4. Altering validation rules is something we could easily allow as long they are not configured as blockers (=validation rules that would throw an error).

Conclusion

There are two different types of changes

1) Backwards compatible changes such renaming a field, adding a field (with a default value if this is a required field) or changing the data type from number to string.

2) Backwards incompatible changes such as removing an allowed value from a picklist field or changing the data type from string to number.

The first type of changes can be automatically handled by the system without any issue.

To support a limited set of the second type of changes in the system, the user will have to provide additional input on how existing values shall be transformed in order to comply with the new data structure, otherwise data will be lost in the next reporting period when the user will attempt to reuse the data provided in the previous incompatible format.

Changing the data structure during the initial design is crucial for the iterative definition process, and must allow data to be entered as it provides an easy to understand process for all involved stakeholders.

It is not a good idea to change the data structure while the reporting is in progress, as this will generate a ripple effect to all data providers who need to change most likely there implementations as well. Further work is needed on how this could be managed and what boundaries we have on making changes during the reporting phase, as it does happen.

Data structures are altered between data collection periods, for example in WISE SoE. For small changes this could still work but for significant structure changes, such as WFD and MSFD streamlining, the data structure should be seen as a new data set.

However we do see that some changes such as validation rules should be open for change during the reporting process and should not produce a significant ripple effect.

2.1.4 Evaluation 1.4: Can we have different versions of the data structure and make separate sandboxes?

While developing the dataflow or even when the reporting has started, some bugs or improvements can be found. To be able to be clear on which version of the data structure is being referenced then we need to be able to freeze at a point in time – which we call a **snapshot**. Therefore we can have a live version, but also have other versions being further worked on, or a version to give the Member States a sandbox where they can test their data before the final release.

Reportnet 2 situation

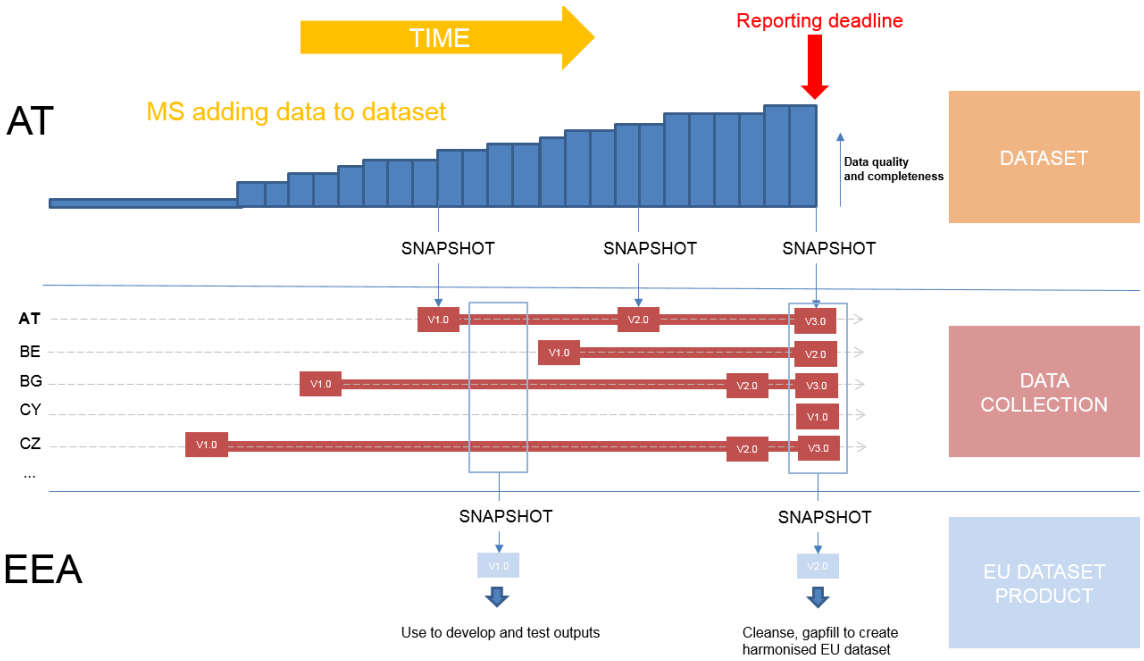
Reportnet has a separate website that is dedicated for testing. This environment is initially used to test scripts or new functionality before it is released in production. But it is more and more used as well for countries to test a new data flow. This system is not entirely disconnected from the production and it does create weaknesses being a test environment for both data flows and software implementation.

Findings

1. Once people can produce snapshots and create copies from an existing dataset you basically allow users to makes their own sandboxes.
2. The test and production environment for a data provider in one and the same platform reduces the complexity to the user drastically.
3. When users have the ability to experiment within the system and have the freedom that doesn't require developers involvement we learn that the learning cycle is fast and the final result is most likely improved.

4. The risk of system saturation caused by an end user because of wrong use or miss use is important to take in account. Our design must ensure that users can't take all server resources during lack of understanding or miss behavior.

The following graphic illustrates the relationships between the reporting of data by the MS and using snapshots to deliver data to the data collection. A country works on the data and when they feel they have a version which meets the requirements of the reporting they make a snapshot and release it to the data collection. They can continue working on the data and make further snapshots updating the collection. They can also (not shown in the graphic) make snapshots which are only available in their personal space – so they can save the data at a point in time and restore it. The requester will makes snapshots of the data collection, adding all the data into a single dataset from which the European dataset is created. EU dataset snapshots are comprised of what has been released by the MS at that point in time:



2.1.5 *Evaluation 1.5: Can we re-use data structures for new data flows?*

It is an important feature to re-use data structures while adding new fields and deleting some of the old ones. For some cases even the data should be re-used and simply updated.

Demonstration

- In the following video, we can see how an existing dataset can be duplicated to use as a template for a new dataflow. The dataset can be duplicated with or without records. (See video)

Findings

1. When we have a database structure that is consistently managed it should be technical easy to allow users to copy-paste tables and datasets from already existing datasets.
2. If we could come with a solution of template datasets or template table definitions then Reportnet could assist in the data structure harmonization. For example a dataset that stores documents could be a template. A table structure that maintains addresses information could be a template. Etc…

Conclusion

The freedom created here allows for a far broader implementation by the data providers and can reduce the implementation cost at provider level and EU level. Multi input mechanisms will reduce the procedural requirements and system development that we currently anticipate countries to implement. It is a must have if we want to reduce the overall cost of reporting.

2.1.6 Conclusion

To assess the impact on model design and implementation, we first evaluate against the Reportnet 2.0 benchmark. Currently XML is the standard for a reporting specification, which has a nested tree/table structure, and through the design process an agreed specification is developed which is frozen/versioned for reporting against.

In our evaluation, we have firstly demonstrated the complex relational structure of XML can be replicated in a dataset comprised of multiple linked tables. The specification can be developed collaboratively by multiple actors to an agreed specification, which through access management can then be frozen, ready for submission by the eventual reporters. In the development process, it is possible to test with data during design, which makes the process iterative and more agile to not only achieve an agreed specification, but a specification which has considered real data. In the development process, it is also possible to have multiple versions of the data structure as candidates, which allows for more agile exploration of alternatives and proposals for the approach.

Through snapshots (essentially backups), we are able to create an integrated system for freezing a structure, or dataset, in time which can then be further worked on in the design process, or submitted in the reporting process and data improvement can continue by the reporter. The system will also make direct linkages between which version of the data corresponds to which version of the structure. Knowing which version of the schema is in use and which schema the data corresponds to, is a significant issue in the current Reportnet.

There are significant positive implications of this approach in attaining the Reportnet 3.0 goals. An online, collaborative platform facilitates actors working closer together. There is increased agility in the process with the ability to test during specification leading to early detection of and correction of problems. This makes the design process more efficient as it is faster and cheaper to agree a data specification. The implications of this is being able to release a reporting ready format reduce downstream costs and complexity on further reporting tools.

In conclusion, a database approach in an online collaborative platform will meet the current process and Reportnet 2.0 capabilities in data model design and implementation, and allow for the capture of further added value.

2.2 Step 5: Helping MS to prepare their reports

2.2.1 Evaluation 2.1: Can we alter the content by different input mechanisms in any order and time?

The main exercise here is to find out if we can allow more freedom to the data provider and how this could improve the data delivery.

Reportnet 2.0 Situation

In Reportnet 2.0 data reporters can only report data by providing a set of files inside an envelope. There are different mechanism to deliver these files even a rest API that semi automate the upload of these files is available. But every delivery is seen as a complete data package. That means that you can't partially deliver by web service and partially by file import. An update of a dataset can only be done by re-doing the entire delivery process. This is heavy for the data provider and requester. Today you can find many kind of formats in Reportnet (Access, Excel, CSV,XML,GML,SHP,…) and these can be uploaded in any compress format (Zip,J7,Tar,…).

Demonstration

- In this video a user is going manually into a record and alters a value (See video)
- In this video data is entered through a form based data entry (See video)
- A video that imports data by XML (See video)
- A video that imports data by CSV (See video)
- A video that imports data by Python (See video)
- A video that updates the above file using a web service implementation (See video)
- The following video demonstrates a user sharing a dataset with a collaborator for comments, who identifies an issue and communicates an issue with the data for correction (See video).

Findings

1. We learned when data is managed by individual records and a data structure fully controlled by a system you basically can have multiple input methods mixed during the import of data.
2. We also learned that you can have multiple users into one and the same dataset who together can provide a final dataset.
3. The flexibility allows data providers to decide themselves the level of integration they wish to produce. We are very well aware of the differences our data providers have and this methods would give them the full freedom to decide what they automate (for example INPSIRE way), half automate (upload files) and produce manually.
4. In principle any format could be provided (CSV, XML, JSON, EXCEL, ACCESS) but we believe that the system should always provide an initial related flat structure. This facilities simplicity in the export mechanisms created by the data provider. Any transformation mechanism could be introduced by EEA or the data provider to handle any complex format towards these flat structures allowing to handle any possible format we can imagine.

2.2.2 Evaluation 2.2: Can we pre-populate data

Reportnet 2.0 Situation

With the increased quality control of reported data comes the need to look up other data for comparisons, for example lists of species or the previous year's reported data. In a noticeable

number of dataflows we supply the previously reported data as a starting point for the current reporting instead of empty templates. The trouble is we don't really have any place to store and retrieve such data from that has versioning and allows for storing more complex data than code lists. Therefore the data is currently all over the place – Converters, CDR, CR, DD, … slightly chaotic. We also don't generally know which report is the final accepted one from a MS as CDR does not have this metadata. The ETC/EEA often don't know how to locate and view what reference data is used in a dataflow, and can seldom add or update this data without the help of a developer.

Findings

An approach could be to extend the data dictionary so more complex data than vocabularies has a place to live. Make it much simpler to manage the data (e.g. upload Excel-files, no RDF knowledge required). Add versioning, audit logs for transparency, ability to link versions of data to reporting cycles (instances of ROD obligations).

We want to be able to easily look up what reference data to use for reporting, not forgetting other versions of the reference data for use in re-submissions of past years reports. We also need master/reference data with versioning organised so it can easily be used in different applications. Verification and management of master/reference data is critical for maintaining data integrity. With the proposed approach of using a database as the storage platform, then the management and versioning of these reference data should become much easier as it is all in one place with some pre-agreed logic on top on how to manage them.

2.2.3 Evaluation 2.3: Can we handle spatial data?

When we look at existing dataflow implementations we see two things. Firstly, in many cases spatial data is managed separately from its tabular data. Some data providers even have other institutions involved that deliver the spatial data independently from the tabular data. Secondly, the spatial data is in a separate file format to the tabular data and with Reportnet 2.0 not able to handle spatial files, the quality control of the spatial integrity and referential integrity to the tabular can only occur after harvesting the files and using FME and databases to process them. So for these two reasons there is a huge gap in how we currently manage the spatial data.

How would this look like in the proposed platform, an environment where forms, file import and web services are interlinked. Can we handle spatial data in the same manner for example as field inside a table? How can a form interact with spatial data and how could we make it easy for the user to load such data together with tabular data. In most cases users deliver spatial data as a shape file or (INSPIRE way) GML or WFS. What strategy could we have on extreme large polygons and what about raster datasets?

Shapefiles are a common means of spatial submissions to Reportnet. The following web resource provides a good overview of the shapefile format against other formats: http://switchfromshapefile.org/

In a database storage platform, we would expect a spatially enabled database, which means the geometry can be stored as a specific field type. As the database understands spatial functions then this allows for us to better integrate quality control checks during import, to give more immediate feedback to the reporters. How we handle spatial data will be more easily reproducible, we can enable versioning and multi-user environments. Spatially enabled

database also means SQL can be used to query the data – aligning with our goal of utilizing a commonly understood language for working with the data.

Findings

1. We learned that partial updates of data can be performed if we have primary key-values configured in tables. In a similar way we could allow shapefile, kml or sqllite formats imported where the primary key and spatial data is imported into the dataset. If the primary key exist you would update that field, if the primary key doesn't exist we could insert a new record.
2. In order to reduce the implementation cost we should be careful what projection systems we would cover. But it's obvious that this kind of meta-data information will need to be stored with the dataset itself when we would allow to store spatial data.
3. Databases have the ability to perform basic spatial queries. We believe that we will face several limitations and should be careful in our choices on what we implement. The technical depth of spatial functionality could easily exhaust the entire development budget.
4. If the new system allows integration of third party software we can allow GIS operators to assist in the implementation of the more advanced GIS requirements.

2.2.4 Evaluation 2.4: Can we handle shared documents or binary files?

Today many data flows require the ability to manage binary content, for example PDF, Word etc. These documents could be documentation describing the data flow requirements but in many cases the data reporter is ask to deliver a PDF-report as well.

Reportnet 2 approach

Everything in Reportnet 2 is managed as folders and files. Reportnet doesn't evaluate the binary files and a data reporter could eventually report any format inside an envelope.

Demonstration

- Create a dataset that is specifically designed to manage documents. This dataset could be copied for every data flow and be managed next to the dataset of this dataflow. The use of 'attachment' data type makes this possible (See video).

Findings

1. A simple datatype "Attachments" would allow to maintain and manage any binary file with the rest of the data.
2. Validation of the content would be hard. There is no system today that is clever enough to interpret free text. We could however validate the extension of the document (.pdf,.doc,…) so we can assure that we receive an expected format and even try to open so we can be assured it is not corrupt.
3. Most of the PDF countries currently report can be viewed as questionnaires. We should consider transforming these as forms. This will make the result more accessible and it would enforce the creators of the questionnaire to think more towards an easy to interpret format that could deliver instant output results. Think of modern questionnaire systems today…

Conclusions

The minimum we need to be able to do is store this unstructured data in an efficient way so it is easily associated with the other reported data, and provide key metadata so it is easily identified.

Handling binary data is a must and should be managed as a field type inside a database. Most databases have ways to accommodate this requirement and provides sufficient flexibility. There is a limit on what a system can do with binary data. Artificial intelligent is growing rapidly and we should constantly look out for new approaches that might help us in the future. Ideas such as auto translation or searching for facts and comparison between documents or creating summaries are been mentioned in other evaluation reports. We could accommodate these future requirements if we assure that our new environment has trigger functionality on record or dataset level. Such triggers could initiate external services specialized in particular tasks. Example is an automated translation etc…

2.2.5 How can we provide a secure mechanism for both users and machines to machine?

Authentication and security is a going to be extremely important inside Reportnet 3. There is a big difference between security provided towards end users and providing access to automated scripts running on a regular basis. Automated scripts are usually managed by a team and user credentials might become compromised.

Reportnet 2.0

Reportnet 2.0 has only one mechanism being the LDAP user authentication. All scripts executed need to use an EIONET user in order to provide the necessary rights to a user or automated script. This eventually has a security weakness as these scripts have a user's credentials embedded and can access all the rights this user has. Reportnet 2.0 manages user's authentication and roles separately from Reportnet 2.0

Findings

1. While looking into Microsoft Azure, Airtable, Google and several other SaaS (Software as a Service) implementation we see that authentication for machines is differently implemented then for users.
2. Automated scripts should get a personal API key or access key. This is usually a very long string of text and numbers that needs to be kept secret for that script. These keys linked to one particular resource.
3. It's a good idea to use an external authentication mechanism that can be used for multiple purposes. We believe that the roles and access rights within Reportnet should be deeper integrated and not be managed outside Reportnet.
4. In order to allow multiple authentications and have a system that could work over multiple systems and servers. We recommend to think of implementing SAML2 (Security Assertion Markup Language) or a similar multi-institutional authentication mechanisms, such as OpenID Connect (a superset of OAuth2).

2.2.6 Conclusion

Currently, users can upload XML files directly or fill out the data in a web form or excel file and the XML file is created behind-the-scenes. In the evaluation we have been able to demonstrate an online collaborative platform with the data stored in a tabular format can readily accept

data in a number of different ways. This covers direct entry, forms, file upload (XML, CSV), and through use of web services. The evaluation also showed that with a well-designed platform, it can be easy and cheap to expose the data structure in a number of different ways.

What the evaluation went further to demonstrate is the flexibility which is given to the reporter in how they want to report. They can use direct entry or forms or services in whole or part depending on what is the optimal configuration with how the data is stored and managed locally.

The capability to handled spatial data and web services are extremely sought after capabilities. Reportnet 2.0 is unable to handle either, resulting in a number of downstream systems being built and maintained outside of the core platform. Databases, such as Postgres, SQL Server, Oracle etc – all have spatial data handling capabilities. The question is whether we can handle the delivery of spatial data as easy as the entry of data into a table. There are significant costs in trying to develop a custom system which will be able to handle different types. We will need to impose limitations on the complexity of the system, for example in supported coordinate reference systems. Ideally, Reportnet 3.0 would be to allow for extension of capabilities through plug-ins from third parties. This part of the evaluation would benefit from further analysis. Closely linked to this is the question of INSPIRE service integration, which is analysed in the next section.

In conclusion, a database approach in an online collaborative platform will meet the current process and Reportnet 2.0 capabilities in helping the MS prepare their data delivery, and allow for the capture of further added value. Multiple input formats can be created relatively quickly and cheaply giving the reporter flexibility in how they report. Further investigation is required regards spatial data handling and INSPIRE services.

2.3 Step 6: Organising the data submission or harvesting

2.3.1 Evaluation 3.1: Are web services a valid approach to deliver data for countries?

When we talk about web services we talk about automated scripts that would move data between systems. In the context of data flows we learn that there is no one solutions for all data providers. The level of implementation at provider's side is extremely diverse and requires that we look at this issue from a diverse manner.

Reportnet 2.0

Reportnet 2.0 is not able to integrate with webservices as it is a file delivery system. A workaround on the reporter side is to download the output of a service as a file which can then be submitted as a file in the usual way. Another use case is AirQuality data measurement Collector where near-real-time is handled not by Reportnet but by a separate system. Countries report hourly measurements of air quality indicators as soon as these measurements become available.

Findings

1. EEA experienced that most data flows are partially implemented by INSPIRE (The spatial part only) and these implementations are very diverse. For this reason here is no "one mechanism" implementable. The only way we can make this operational is by allowing a multi input mechanism that must allow partially field updates. In other words we need an environment that allows data to be merged into our system from different sources and mechanisms.
2. Automated scripts only provide a value if we can ensure a stable and consistent environment. It would become too costly if it breaks often and by default users would move towards a manual approach instead. The data structure, security and access point all need to stay as stable as possible over an extensive period of time.
3. Web services can also be used to load data from one system to the other in a manual approach. Meaning that the data is not uploaded using a schedule but instead performed by a person using an ETL tool. This approach seems to be a valid way if more control is required during data update/delete between both systems. A file upload only gives some level of control.

INSPIRE Feasibility Study

In parallel to this feasibility study, there is also the INSPIRE feasibility study. The purpose of the study is to assess the applicability of harvesting national INSPIRE services to automate the collection of geospatial data sets falling under reporting obligations. This is done through two use cases for firstly data harvesting and secondly referencing spatial objects, using Natura 2000 sites and the WFD as thematic domain. The output of this study will help understand the complexity in dealing with INSPIRE services from MS and how this will impact the future system design.

Conclusions

Data becomes more and more accessible over the web and makes it easier to automate the collection process. However, as the different flavours of the same service, a transformation process for every data provider is going to be required and most likely these transformations will have slight differences because data providers have different technologies, different depths of implementations, different organisational structures and different workflows. Even for those who implemented INSPIRE. What web services provide is the ability to allow transformation

software (ETL) to link these systems together over the web. The creation of the transformations and the maintenance cost needs to be evaluated against the manual approach for every data provider individually.

2.3.2 Evaluation 3.2: Can we embed a replication mechanism for versioning and testing?

One of the key capabilities referenced in the section 1.4 is to be able to take snapshots in the design and reporting phase. This is for two functionalities, the first is that you can take a copy and test data with it, to change the structure. The second key functionalities is the "official transfer" of data towards a data requester. This implies that data providers cannot further change the data delivered. A good technique would be the creation of time based snapshots of the data (a copy of the data). Our experience from the past also taught us that transferring data requires testing and is a crucial functionality for our data providers.

Reportnet 2 Situation

There is no real "implementation" that covers this today. Reportnet 2 has two work around that come close to these needs.

Reportnet 2 has a clone environment that can be used for testing. This is partly used for testing new software implementations and as well for countries to test a data flow. Not ideal as it doesn't allow to bring an uploaded data set from test to production. It is also a different infrastructure setup making it vulnerable for a different behavior.

The other approach is the use of additional envelopes. The weakness of this setup is the difficulty to know what envelope replaces the previous delivered envelope as we get multiple meanings for envelope delivery.

Findings

1. We could not find an implementation today that demonstrate exactly this approach but be demonstrated as a process of several steps.
2. We need to keep in mind the performance of the system and management of the overall system in how we best create snapshots. So far a snapshot (export internally) of the data looks as the best option. These snapshots could then be imported back into a similar structure as a copy for the data provider or as a collection of data for the requester.

2.3.3 Conclusion

The capability to integrate with services is key, where INSPIRE compliant services are being developed as part of the reporting specification. Also important is the re-use of data in the reporting which has already been exposed through INSPIRE. We anticipate the INSPIRE feasibility study will show in its research the varied landscape of services within one specific offering across all Member States. A successful reporting platform needs to be able to deal with standardisation, and will not be able to support in a cost effective way a complex transformation layer, where an unknown number of differently formatted input services would ned to be handled. It is proposed this part will need to be the responsibility on the reporter's side.

Another capability considered in the evaluation of the platform is to be able to freeze or snapshot a data delivery at a particular point in time. This is to develop a mechanism on which workflow can be built for the official delivery of data by the reporter. Making snapshots allows

for the data to be further worked upon even after the data has gone into the next part of the flow.

In conclusion, a database approach in an online collaborative platform will meet the current process and Reportnet 2.0 capabilities in helping the MS organise their submission for delivery, and allow for the capture of further added value. Further investigation is required regards spatial data handling and INSPIRE services.

2.4 Step 7: Ensuring quality of the reported data

One of the key parts of a reporting flow is the quality checks which are used to ensure the data meets the agreed specification. Data entry without any means of imposing quality expectations will not be a workable system.

We have identified three depths of validation:

1. Record level validation. Example of these validations are for example data types (number, text or date), lookup to a list of values or other dataset, threshold values, consistent polygon…
2. Data level validation for each provider (Lets call this a dataset). Examples here are for example unique value within a dataset, sum of a group must not exceed value…
3. Data validation for the entire data flow (Lets call this a data collection). Example here is a unique identifier over the entire data flow.

2.4.1 Evaluation 4.1: Can we implement record level validation checks?

Reportnet 2 Situation

Reportnet 2 uses XML schemas as a first level of quality control by ensuring that the content within the XML file follows a defined structure. This usually covers the XML consistency. These XML schemas are manually designed by an XML expert. Some validations are performed by XQuery scripts developed by programmers specifically for a particular data flow. Request for change requires developers to be involved and is usually a slow and complex process.

Demonstration

- Using Airtable, trying to enter data which is not of the correct type is blocked by the interface with a user-friendly message.
- Using Airtable, trying to enter data using an import which is not of the correct type is blocked by the interface from entering those records.
- If a field type is set to e-mail or URL Airtable highlights values which fail the integrated validation check (i.e., email needs a '@')
- In the following video, we show how a record can have an error associated with it, based on an external quality control process, here using FME. The error messages are stored as a code list and so the error number gets meaningful message to the reporter (See video)

Findings

1. The Record level validation would require an implementation on two levels. Script level for instant reporting to the end user. Users would get a visual representation of these errors.
2. The record level feedback should show partial results as each test is completed for each record entry
3. Spatial data validations can be performed in modern databases engines today and would as well eliminate a weakness Reportnet 2 has today as well.

2.4.2 Evaluation 4.2: Can we implement dataset level validation checks?

Reportnet 2 Situation

Dataset (schema) validations, for example ensuring waterbody IDs are unique, are performed by XQuery scripts developed by programmers specifically for a particular data flow. Request for change requires developers to be involved and is usually a slow and complex process.

Demonstration

- In the following video, we show how a record can have an error associated with it, based on an external quality control process, here using FME. The error messages are stored as a code list and so the error number gets meaningful message to the reporter. FME allows us to implement simple as well as sophisticated quality checks (See video)

Findings

1. Dataset level validation would require aggregate functionality on a database level. This could provide flags to set actions to or provide overviews in a dashboard alike environment. There is an overlap between the record level validation and dataset level validation what would require implementation of the logic in more than one technology.
2. Simple checks such as unique values can be defined at design time and checked automatically by the system. More complex checks will have to be developed per dataflow and of course need to be versioned (when structure changes, existing complex validations will break). The system shall offer the necessary APIs to allow external systems communicate the outcome of validation checks.

2.4.3 Evaluation 4.3: Can we implement data collection validation checks?

Reportnet 2 Situation

Reportnet 2.0 stores the reported data in the schemas as flat files and as it is not possible to make relations between files. Therefore no collection level checks are performed in the Reportnet 2.0 platform. Using the Common Workspace, what typically happens is FME is used to harvest the XML, which then runs additional QA checks (sometimes the same as executed with XQuery) and inserts the data into a SQL Server database. With the reported data now being collated into a single database then collection level checks can be executed. Any issues found are typically written back to the Reportnet envelopes as HTML for the reporter to address and the envelope status is marked accordingly to the type of issues which have been found.

Demonstration

- Nothing defined

Findings

1. The collection level checks currently undertaken are performed when the data is in a database. With the database-centric approach, we are populating this database directly, and therefore this will remove the XML harvesting step and inefficiencies associated with this (for example in not knowing which version of the XML is the latest). Within the scope of this study, though we have not determined whether it is optimal for the MS to report into a separate database, or whether they will report into a single database for all MS. This is important for knowing when it will be possible to initiate collection level QC and what the trigger is.

2.4.4 Evaluation 4.4: Can we generate QC outputs such as maps and dashboards?

Delivering good qualitative data can only be achieved efficiently if the data provider and data requester can receive a good overview. At an instant we should get information where we have errors in our data or get statistics on our dataset. We are aware that we will never provider a tool that handles all possible cases required because of the complexity of data processing sometimes needed. But a subset of functionality directly integrated into the system would improve the usability drastically.

Reportnet 2 Situation

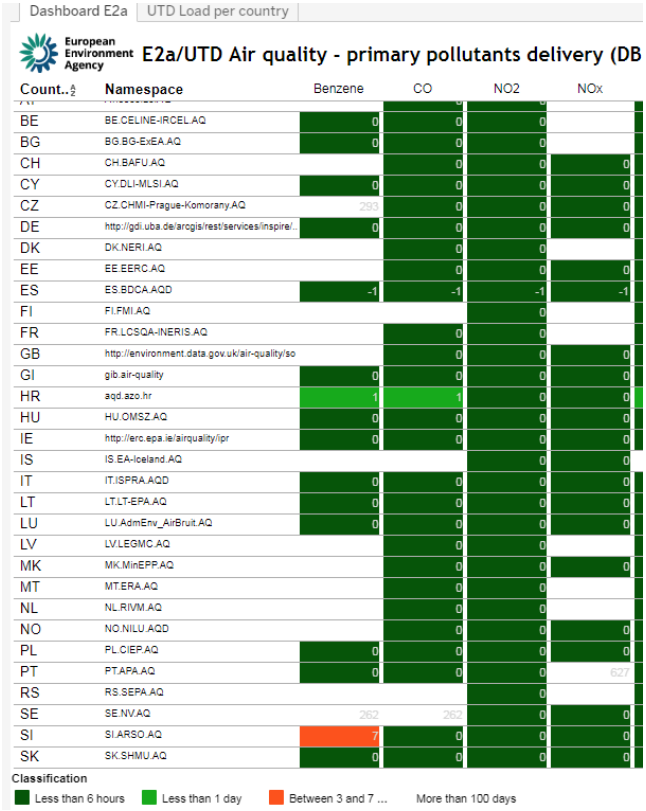
Reportnet 2 has only the HTML validation reports a data provider can see. These reports are generated after the entire file has been loaded and generated by XQuery or XSLT. When the amount of records is large it becomes a heavy and tedious process. In some cases EEA has built a workaround using Tableau dashboards but this requires a full imported of these files inside a database and only is available once the country has officially delivered the content.

Reportnet 2 produced some workarounds that scan the errors and warnings and provide a statistical overview of all errors, blockers and warnings

Demonstration

- Make/show a chart. (See video)
- Make/show a filter (See video)
- Make/show a map (See video)
- A filter or graph could show those records that don't have a relationship. These visual tools could assist the data provider in understanding the errors a dataset contains and assist in the quality assurance process.
- EEA Has examples of these requirements in the form of Tableau dashboards that demonstrate the usefulness to the data providers and data collectors. The following example of Air quality provides an instant overview of how far each data providers has

delivered. Who has critical issues, etc…



Findings

1. Get overview maps or graphs is proven to be crucial in order to rapidly understand quality issues of a dataset or to understand progress.
2. It's a must that some basic features are available for data reporters to use and/or generate graphs on the datasets before they deliver the final dataset.
3. Record-based databases management would allow us to produce aggregates and filters on top of the data instantly on the data set delivered.
4. Dynamic views controlled by filters could provide instant feedback to the user improving the reporting cycle that consist of deliver, evaluate, improve source, deliver, evaluate,...
5. I some cases simple datasets could instantly represent useful outputs to the data requesters and reduce additional development costs in sophisticated analytical software.

2.4.5 Conclusion

Quality control checks are an essential part of the workflow and with Reportnet 2.0 have been shown to be an area which was repeatedly highlighted for improvement. The biggest cost spent on dataflow implementation is the specification and implementation of quality control tests, which all dataflows now require extensively. Since XML validation results are too hard to understand for most reporters, we manually implement even the most basic things such as datatype tests. For the ETC it is as cumbersome to write the specification for the tests as it is for the developers to understand it. The Xquery language used for most QC tests seems both difficult to learn and to be productive in, even when experienced. For each group of tests we manually implement a custom HTML presentation of the outcome (additional cost), and none

of the tests results are really machine readable for re-use. Often the full set of rules needs to be implemented before they can be tested. Finally, the MS cannot easily test their data before submitting it.

A different approach needs to be adopted to the one currently implemented in order to provide more immediate feedback to the user issues, in a language which is clear, and a link to the specific records which are in error. The turnaround time is currently too long and staggered, with reporters getting feedback first on the envelope closing, then after a second round of checks from external tools (e.g. FME) and then from manual checks by the ETC and finally from the Member States identifying issues when they see how the EEA has used or analysed their submissions. In Reportnet 2.0, the data flow has been extended outside of the platform for data custodians to make use of tools such as FME and databases to implement quality checks which are not possible from schema validation – for example conditional checks, spatial data checks. These additional checks do not fit the Reportnet 2.0 applications model, meaning they are not integrated and are an additional cost. Some of these routines write errors back to HTML pages in Reportnet envelopes, others store errors with the data in external databases of collated reportings.

A database centric approach immediately has the advantage over the file-based XML, in that it firstly has the data already stored in a format which allows for the range of quality checks to be run without the need for processing of the data and moving into other systems. The outputs of the quality control tests can be standardised machine readable data, so it can re-used by any downstream use of the data. We also manage the data on a record level rather than a file level – so we can be more precise on where the error is, which introduces a more efficient submission and feedback loop to the reporter. We can also have prepared views on the data for the user to make visual checks on the data to verify it looks how they expect it to. Having errors held with the data in a standardised system also means it is easier for a helpdesk to deal with, particularly to proactively address issues and analyse patters across schemas and time.

With Airtable we can demonstrate how the user gets immediate feedback when trying to enter data where it violates the type or a restricted value, for example trying to enter text in a number field, or if trying to import data from an external file. We can also see how in creating a field we can specific additional restrictions for example on a date, email, url etc. This record level of validation covers a large part of the XML validation. Organising the data into related structures is the second part of XML validation and we are fulfilling this through having the data organised into tables.

We can demonstrate how an external tool can be used to implement those checks and then write back to the table error messages for each row. This is a critical factor for the success of a new platform – being able to write the error with the record in question. It is much easier for the reporter to be able to filter on this and see the error in the context of the data rather than reading it in HTML and having to go back to their source data to find it. In the current dataflows, this is already in place for the data custodians when the reported data is extracted and brought into an external database, error checks are put with the data – possible with a database-centric approach, but not with an XML file-based approach. However, this is not available to the data reporters who rely solely on feedback on the XML file in an envelope.

The final part of QC is data visualisation – being able to see the data presented back to the reporter, or for the requester team – EEA, Commission etc. The AirQuality flow has this part of

the workflow – letting the reporter see their data in a view of how it will be used at European level as a final check. Currently all the process for this happens outside of Reportnet, but needs to be integral in the new platform to give the immediate feedback.

Dashboard overviews are also important – getting an understanding on a number of errors, types, how they should be treated. In a database centric system, this is easier to implement as these overview are views on data already held in the system and can be much more dynamic as issues are addressed and the reporting moves through a workflow.

2.5 Step 8: Carrying out data processing and analysis

Quality control in the previous step, is crucial in order to come to a coherent European dataset. Our data providers come from very different backgrounds having different methods, organizational structures and methods. Validation of the received data is crucial to ensure the integrity of the data.

Data processing and analysis comprises the tasks where quality controlled individual Member States deliveries are merged to create a single European dataset product. There is an important distinction between the collation of all the Member States deliveries into a single database and the steps to create a European dataset product. It is the products on which the maps, tables and reports are published.

Within the database feasibility, we have through the delivery of each Member State a snapshot of their data. The collection then can be snapshotted – following the same logic – allowing for further European level products to be worked on from a version of the collated data. We have traceability and also the capability to update the data with a new snapshot, if the decision is this, or we might set it to refresh automatically. This is all configuration.

2.6 Transition: Legacy data integration

2.6.1 Evaluation 6.1: Can we import data from xml files, to make data flows backwards compatible?

In some cases it could be necessary to maintain a backwards compatibility with the structure that is already in place, using XML as the reporting technology.

Reportnet 2.0 situation

In Reportnet 2.0 the final delivery is an envelope containing one or more XML files. Every dataflow today has its own XML Schema. When moving towards a record based database environment a transformation is going to be required.

Demonstration

- Show the XML importing tool (see video)
- Demonstrate an FME process importing from XML to a database structure (see video)

Findings

1. For simple data flows the concept shown in our demo could easily be implemented even by the local database manager.
2. When using ETL tools any XML structure (including the most complex one) can be mapped against this generated data modal demonstrated here.
3. We learned that the majority of our data providers are more table oriented. By definition it would be simpler if the data provider would be allowed to import data in pieces (one per table). A record based database approach would allow for the system to monitor the consistency between the individual tables. The learning curve and cost for implementation would be drastically lower then to maintain or implement the existing XML structure.

2.6.2 Conclusion

This new approach would not stop us re-using existing XML schemas by implementing import modules. The cost for keeping the old structure would come down to an ETL transformation process generated by EEA. We don't think it would be a good idea to hold part of a dataflow in Reportnet 2 and another part in Reportnet 3. We strongly suggest to keep Reportnet 2 infrastructure existing and have a new approach implemented next to the existing one. The user interface of Reportnet 3 should make both methods seamless to the end user.

The question of whether legacy XML files need to be imported would need to be dealt with on a case by case basis. Much of the reported legacy data has been made into products with much cleansing in the process, and this could be in most cases the starting point. Over the years schemas have changed as well which would make it not cost effective to have so many mappings. This is particularly true of time series data, with registry data being a little more manageable as it is resubmitted each year and you would likely want the latest version and not the history. There are many lessons learned within the EEA data custodian community on the integration of legacy data into newer reportings for example.

3 Selection of database engine

The core ability for this new system will be determined by the database engine we select. We see this to be the most difficult aspect because it will have a serious impact on everything we develop in the future.

3.1.1 Options

The idea of using a file based database format such as Microsoft access or SQLite (or Geopackage) came to mind but are not an option if we want multi user and multi service accessibility over the entire platform. This doesn't mean that such format could not be a source for import in the future but it can't be the core data engine solution, as the current file based issues would persist.

We determined that there are two main solutions of database types on the market that could accommodate our needs. The first being the conventional **relational database** that in its modern days is extremely powerful and is well understood by EEA staff today. The other solution is the use of a **document database** (JSON).

3.1.2 Analysis

- Detailed report for **Document database, using MongoDB for analysis:** https://discomap.eea.europa.eu/trial/Reportnet3Study1/docs/Reportnet3_Database_feasibility_study_MongoDB.pdf

- Detailed report for **Relational database, using PostGres for analysis:** https://discomap.eea.europa.eu/trial/Reportnet3Study1/docs/Reportnet3_Database_feasibility_study_PostgreSQL.pdf

3.1.3 Summary

#	Requirement	Document DB (mongo)	Relational DB (postgres)
1	**Querying**	★★★☆☆ The automated querying (by the application) is doable, because developers will take care of it. Somewhat more expensive. Querying by users on a single table is friendly enough. Anything beyond that requires coding, and using the powerful -yet complex and size limited- aggregation pipeline.	★★★★★ Widely-known syntax and full-featured capabilities. Friendly filter syntax
2	**Exploiting data**	★★★☆☆ Working from FME cumbersome and slow but doable. Tableau would need extracts or CSVs through the REST interface. Custom applications should have no problem.	★★★★★ Mature client components for all use cases.
3	**Internal relationships**	★☆☆☆☆	★★★★☆

		mongodb can technically handle it, but there is a price to pay. Either we make the whole system more complex (embed/link case by case), or we lose performance and the advantages of the document approach (always link)	need to implement the logic, but underlying structure is standard.
4	**External relationships**	★★☆☆☆ they will have to be limited to standard cases and be done entirely by the application. Similar implementation to internal.	★★★☆☆ very similar implementation to internal relationships. Visible even without the R3 application. R3 needs to keep track of connections and maintain them.
5	**Inserting data**	★★★★☆ the application wrappers for inserts should be easy to make. The problem lies in concurrency and transactions	★★★★☆ application wrappers will be easy to make. Familiar interface towards third parties
6	**Multi-user access, concurrency and transactions**	★★☆☆☆ transaction isolation is expensive performance wise. We could live without it, but it entails relevant risks: batch writes overlapping and loss of data integrity if individual queries fail.	★★★★☆ transactions are available regardless of R3, and behave “as expected”.
7	**Altering data structures**	★★★★★ no steps need be taken at all regarding the data storage. Optionally, the jsonSchema can be set for additional coherence	★★☆☆☆ Feasible but expensive both in development time and performance impact.
8	**Copying datasets and snapshots**	★★★★☆ creating copies is comparatively simple. Some metadata is needed (reference tables). In a sharded scenario, complexity increases.	★★★★☆ powerful mechanism for creating copies with minor drawbacks. Metadata is needed.
9	**Performing backups**	★★★☆☆ free tools provide full backups, varying performance and growing complexity as the system scales out. Enterprise tools should simplify to a certain extent, but have an unknown cost	★★★★☆ provides out of the box options for full and incremental backups. Minor manual scripting might be necessary
10	**Security**	★★☆☆☆ no LDAP integration with the opensource version prevents direct access to the data by corporate users. All interaction has to go through the API. Authorisation limited to DB level.	★★★★☆ Multiple authentication mechanisms, including LDAP, windows and user+pass. Fine grained control down to row level
11	**Spatial data**	★★☆☆☆ GeoJSON storage. SRS fixed at 4326. Limited query operations. No transformations. No raster support	★★★★★ PostGIS has everything we might ever need.
12	**Binary data**	★★★★☆ Interaction from the R3 logic relatively simple. Files can be batch	★★★★☆ Supported, and relatively easy to interact with.

		extracted with standard command line tools.	
13	**Scalability**	★★★☆☆ horizontal scaling capabilities are built-in. Setup is not trivial but manageable; can be made easier with paid tools. Performance will be sub-optimal due to sharding key election. Affects the complexity of other areas. Simple alternative is easy to setup.	★★★☆☆ Horizontal scaling available with extensions. Sharding key difficult to assign automatically. Simple alternative easy to setup.

3.1.4 Conclusion

In general terms, performance and scaling is of a great concern and our implementation should be based on a multi database instance approach where we isolate dataset implementations in individual databases. Our implementation must take the approach of the super small and super large.

Secondly, we must use “server based” technology if we want multi user access, web services and dashboard implementations to work on one and the same database.

From the analysis, we would recommend the **relational database** as it is much better suited to the structured nature of the data in question. It also optimized for the many analytical post-submission processes that are undertaken and for the handling of spatial data.

4 Example workflows using Wireframes

To answer the user collaborative approach we generated a number of scenarios our users would follow. By producing these realistic approach we hope to learn how this can simplify, improve the process, make it more agile and reduce the cost. For each of these a blog is produced and the links can be found here.

The flows are described below using a narrative and screengrabs. The latest version of the wireframes are available online with the workflows supported with guidance text:
https://app.moqups.com/m5650fuMPK/view/

4.1 Creating a new Dataflow

Actors: (The actors in this scenario)

Data receiver: The EU officer requiring the new data flow.
Data provider: A number of Country data providers who are involved in the design process.
Data steward: The EEA thematic expert responsible for the data flow
Data custodian: The technical expert at EEA who assists in the creation of the data flow.

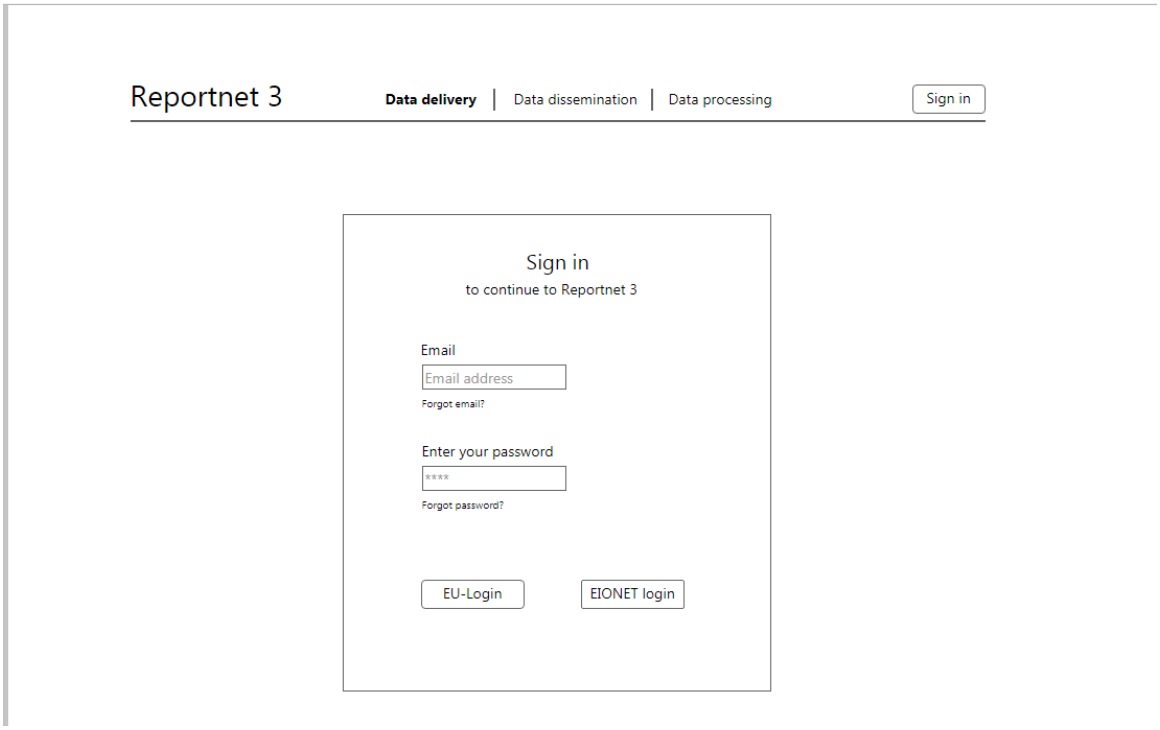
Scenario:

This scenario describes a potential process in how a new dataflow could be created inside Reportnet 3.0.

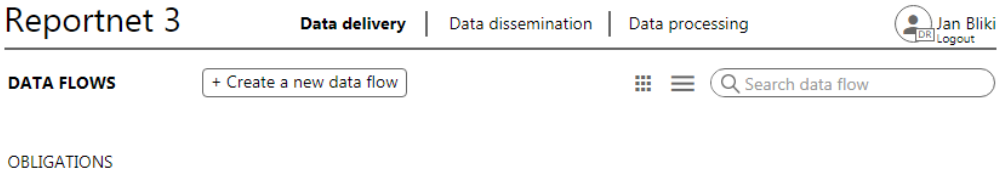
A. Getting ready:

1. The Data requester is planning on a new report obligation. At this moment the Data requester has already designed an intervention logic and ideas around the reporting products (Step 1), Has a draft reporting obligation in legislation (Step 2) and is ready to prepare the implementing acts on reporting (Step 3).
2. The Data requester and EEA agreed that this data flow is going to be managed inside Reportnet 3 and EEA has appointed a data steward and data custodian to assist in this process. This was performed via offical communication.

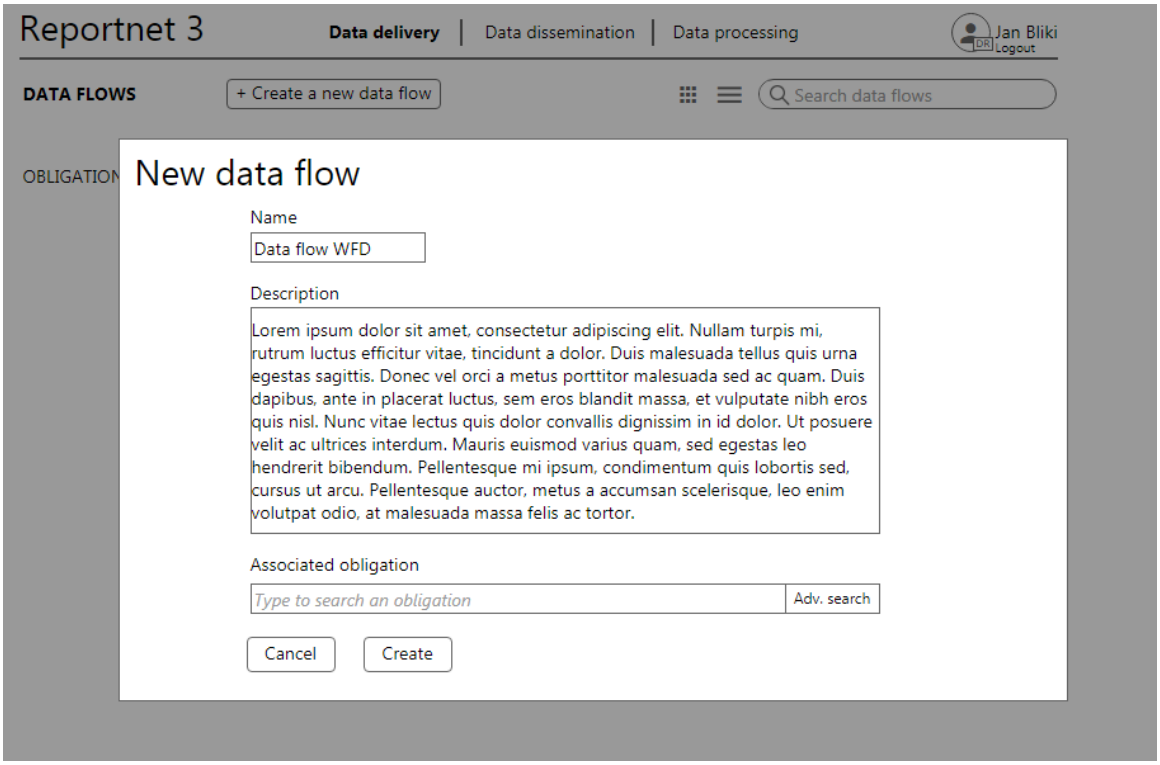
3. The data custodian is assigned (by the data requester) to initiate the process. He navigates to the Reportnet 3 website and login into Reportnet 3 using his preferred login mechanism (EU-login or Eionet login).
   (For practical reasons we have chosen the data custodian to perform the first step but in principle this first step could as well be initiated by the data steward, data consultant or data expert.)



4. The data custodian enters into his/her own personal space inside Reportnet 3. In principle all the data flows he/she is involved in would be listed under his/her personal space. That is true for every participator who would have its own personal space inside Reportnet 3. This data custodian is new and was not involved in any other data flows because the personal space area is empty.

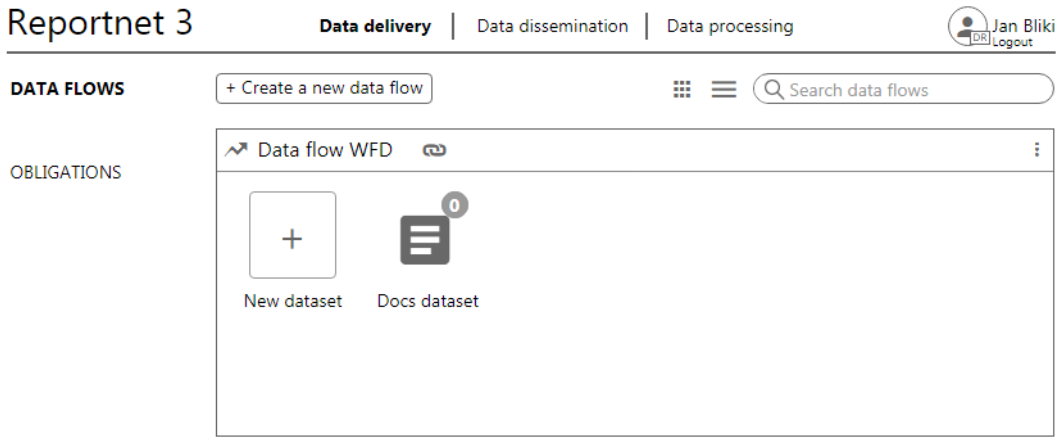


5. By pressing the [+ Create a new data flow] icon the data custodian is able to create a new "Data flow". At this stage the data custodian provide a new name, description and links the new data flow to a data obligation.

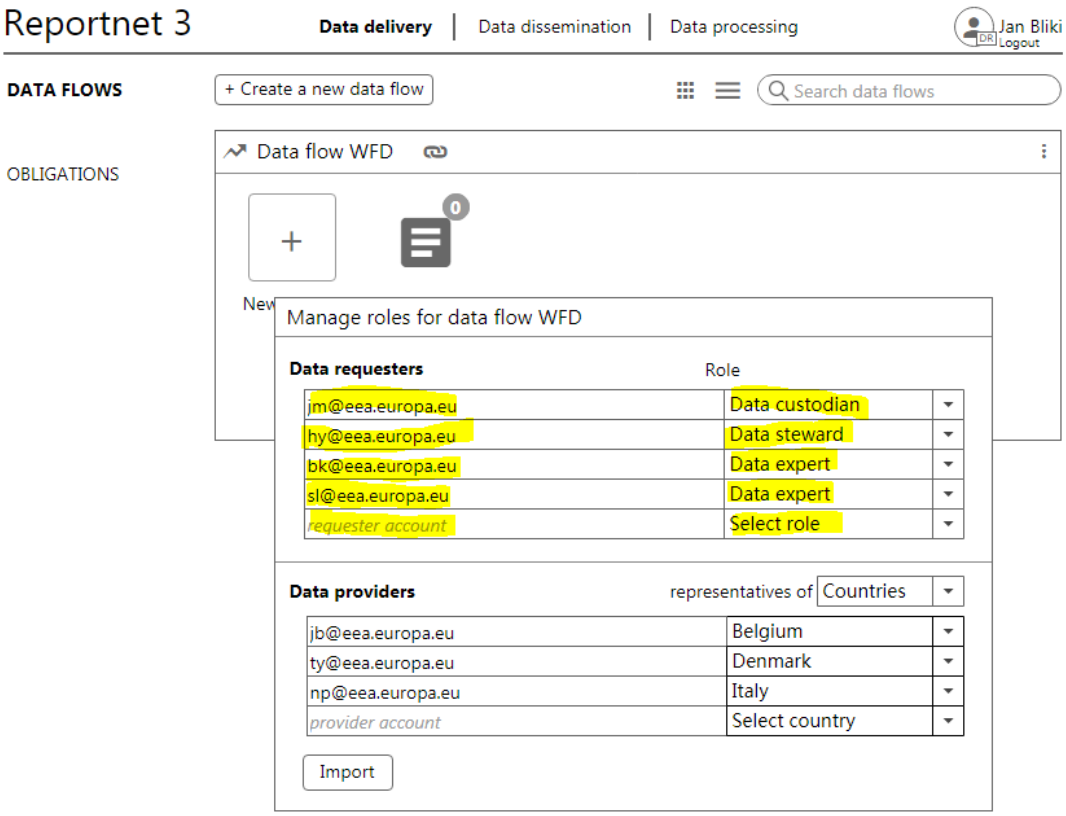


6. When all information is filled, the data custodian presses the create button. A new data flow work area is created and provides instantly two related products under the new data flow.
   a. A dataset icon that will allow to produce a new dataset for this dataflow.
   b. A document icon that can contain all available documents

   Next to the data flow title we have an obligation icon providing a link back to the reporting obligation this dataflow is linked to.



7. The data custodian can share this new dataflow to all contributors by clicking the vertical three dots icon. This will provide him/her with a number of options. By selecting the "Manage roles" another popup shows up.

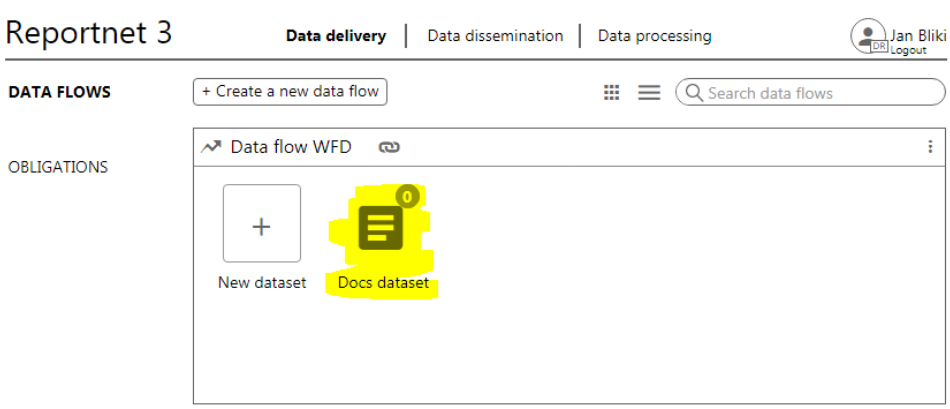


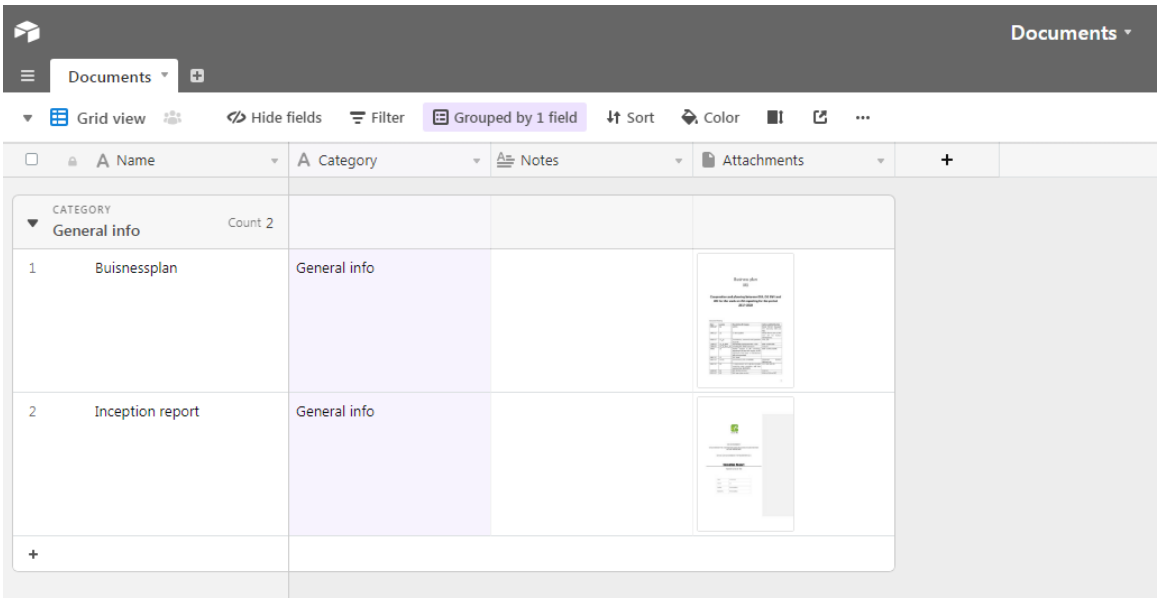
8. The data steward can enter the emails of the "data steward", "Data expert(s)" and "Data consultants". And as well provides a new mailing list name for this dataflow. At this point we do not need a list of data providers but this can be filled at any time during the process.

9. The data custodian uses the mailing list to inform all participators. At this stage all contributors can login and see the same data flow in the system. (Note that every dataflow would have two mailing lists. One that represents the responsible team and another that represents the data providers)

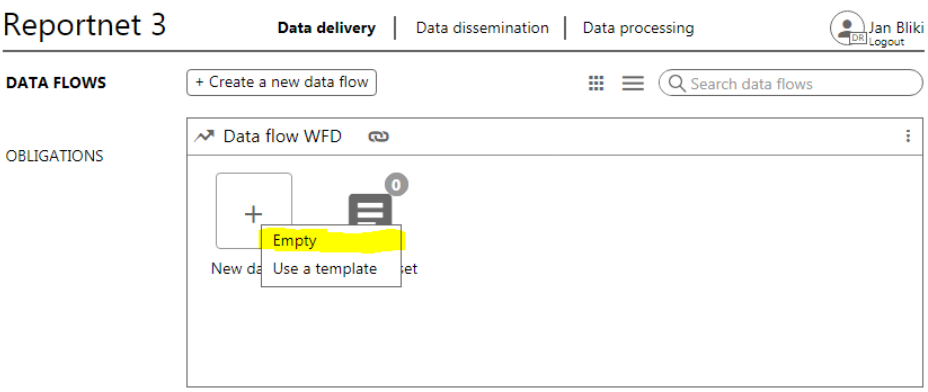
B. Prepare the data flow

1. All participants are asked to upload any already existing documents using the document icon. At this stage we can involve our data consultants and ask them to design the dataset based on the already existing requirements. This action can be performed at any time during the process. New or updated documents can be reported on a constant basis.

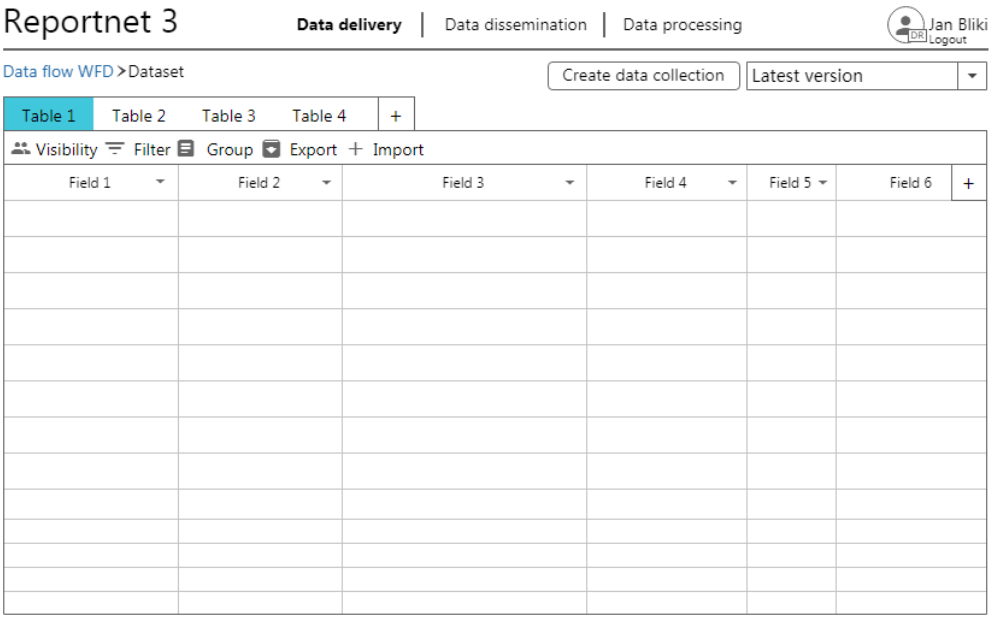




2. The Data consultant is asked to design a dataset based on the already available information given by the involved partners. The data consultant can login to the system and will see in his personal space the new data flow he is supposed to work on.
3. The data consultant clicks on the "New Item" icon and is asked if he wants to create a dataset based on a template or create an empty one. As this is an entire new data flow we decided to select "New empty dataset".



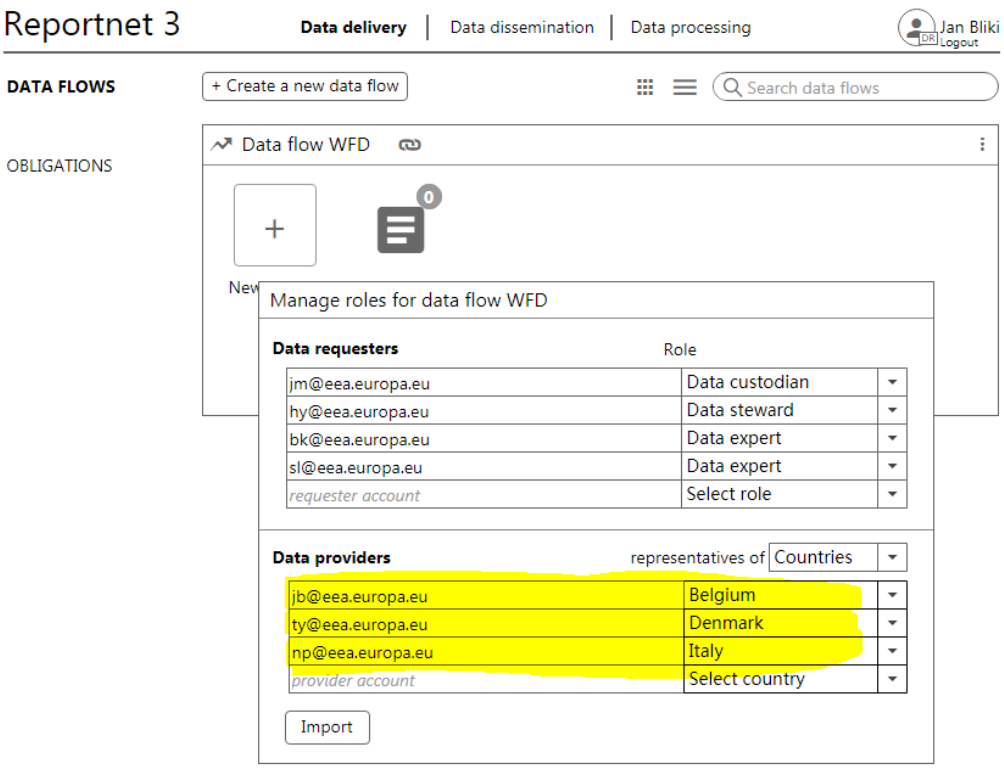
4. The data consultant create tables, items and relationships for a dataset based on the provided documents and discussions with the data custodian, data steward and data expert. At this stage all partners can overlook the progress and interact with the dataset. The data custodian together with the data steward and data expert review the designed proposal. The data custodian is asked to overlook the design from a technical point of view and alters (in discussion with the others) the tables. The data expert can try sample data and once again discuss all findings with the others. (This page is further explained in the dataset creation, forms and imports)



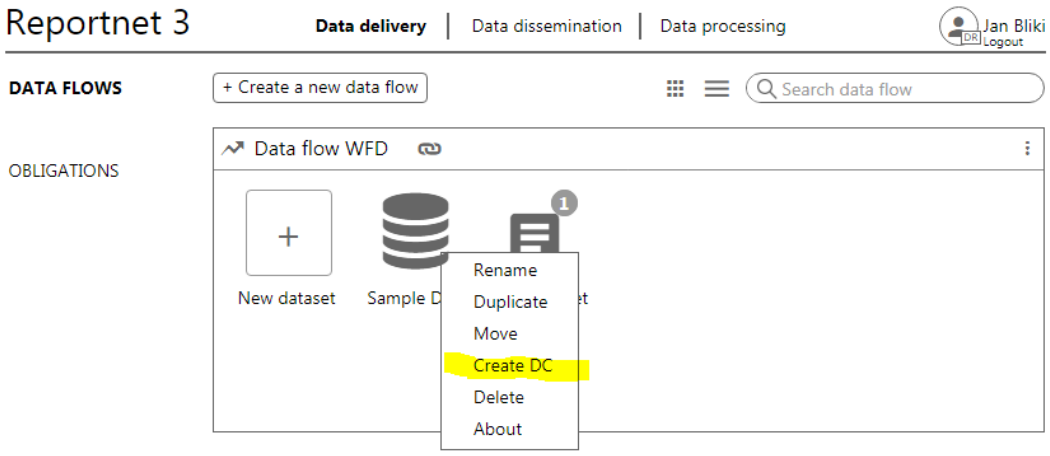
5. Validation rules are created and tested (to the extent possible) by entering test data or fake data entries (Step 7).
6. A number of views and dashboards are created to visually demonstrate key results and once again tested using test or fake data entries. (Step 8)
7. This process iterates a couple of times until an acceptable structure is achieved that could be presented towards the data providers.
8. We have now a consistent agreed data flow by all contributors.

C. Review by data providers

1. The data expert calls for a meeting between the data providers and dataflow responsible.
2. The data expert and data steward compile a list of data providers and use the data flow roles window to enter or imported these from an LDAP role. The reference is filled for each individual data provider (if this was not imported from the LDAP system).

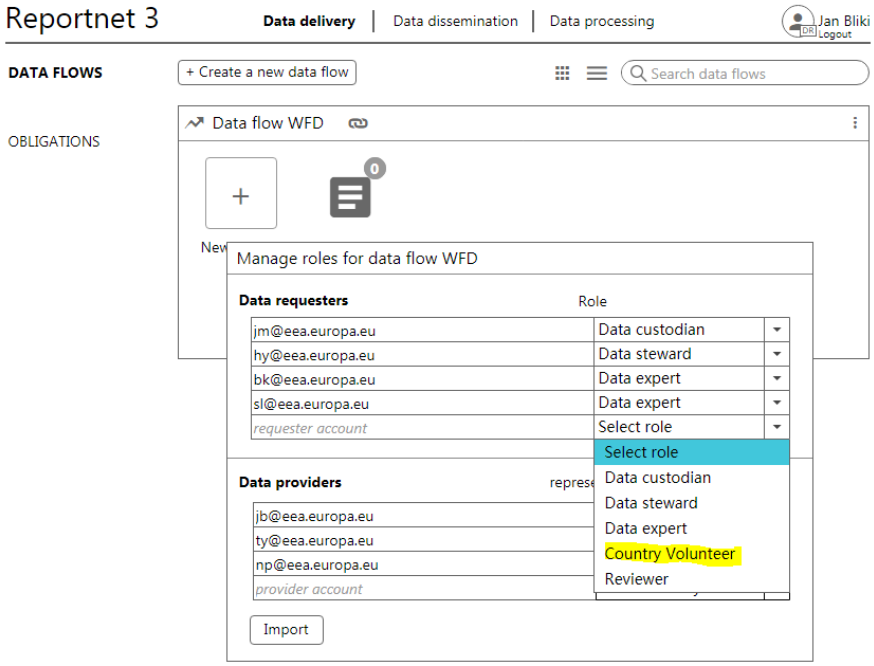


3. The data custodian prepares the helpdesk infrastructure and add the helpdesk mailing list inside the "Responsible"-roles section.
4. The data custodian creates a test "data collection". This will be send to all data providers so they can review and even test the current implementation.

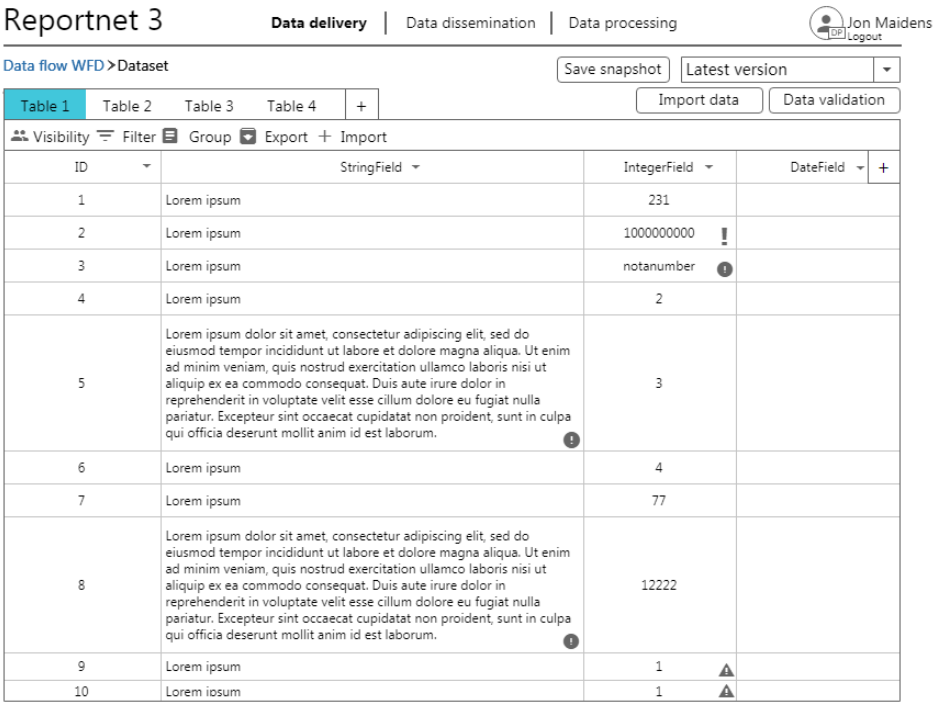


5. An email is prepared for all data providers that includes an invitation for a meeting and instructions on how they can enter into Reportnet 3. The data flow will be available for experimental purposes. Further guidance can be found inside Reportnet 3.
6. The data experts sends the invite out to all data providers using the mailing list for data reporters.
7. During that meeting the new dataset is demonstrated by using the Reportnet 3 interface. All comments are documented and stored inside the documentation section of the data flow.

8. During this meeting volunteers are asked to review and participate in the data flow creation.



9. The volunteered data providers get a couple of months to experiment and review.
10. During that period data providers receive helpdesk support during the entire period.
11. The dataset is further altered based on the outcome of tests and bilateral discussions. (A data structure is depended on how countries have organised themselves. For example if two institutions are involved it might be better to split tables up so the process facilitates the delivery of data. But also possible relationships to other dataflows can be proposed. All these steps can be tested by all partners in the process with example data. This task can be supported by a consultancy team that performs the majority of the work.)
12. Real data received by the volunteered data provider allows us to improve the outputs and validation processes (Step 7,8,9)



13. This is iterated until an acceptable result is achieved.
14. The test data collection is removed from the system as it has performed its purpose.
15. After this stage we reach a final data flow that is ready to launch.

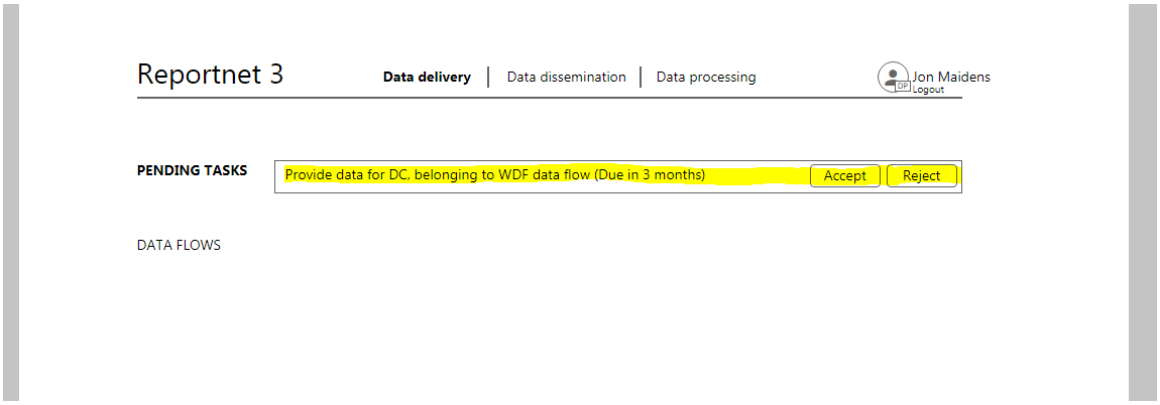
4.2 Data providers delivers data

How would a data provider deliver data and what kind of agile methods can we detect from this approach?

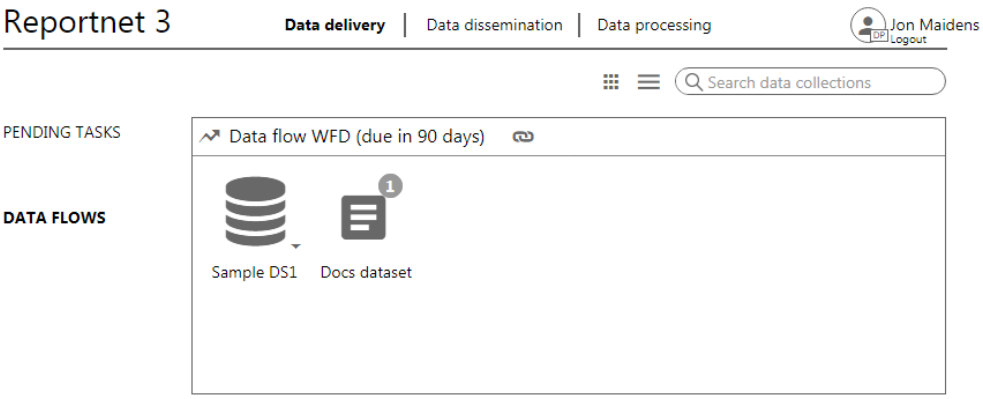
The following scenarios will go through a number of ways a country can deliver. On purpose we tried to make this as flexible as possible in order to reduce the countries implementation to a minimum. Every country can have very different needs and should be able to find his/her way to report in the most convenient way. The following two blocks describe the start of accepting a data flow and when they are finished with a data flow. What is in the middle can be performed in many different ways.

A. Accept the data flow request.

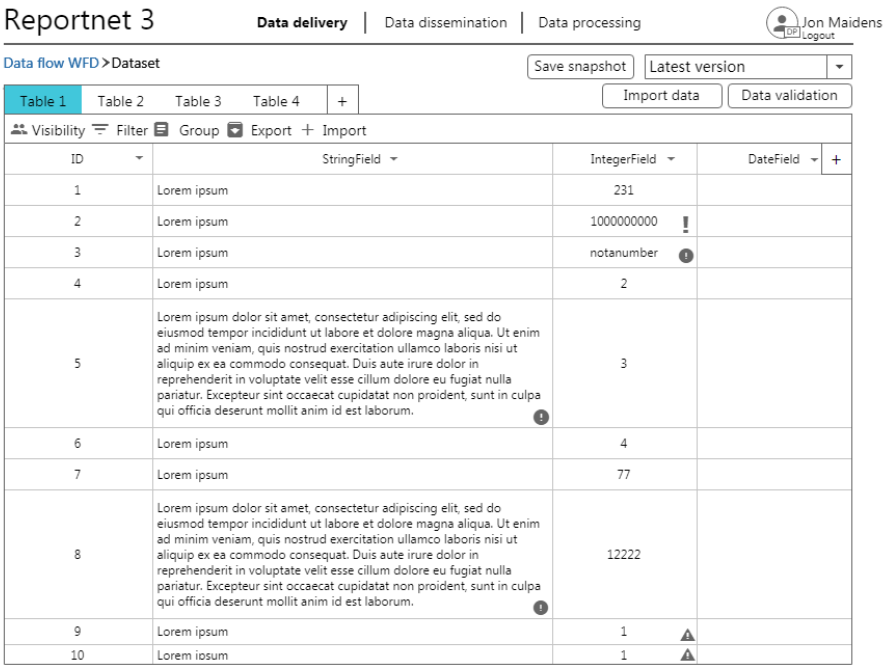
1. The data provider login to Reportnet 3 and finds in his personal box a new action for a data flow request.



2. The data provider clicks on the accept button and a new data flow group is created inside the data providers personal space.



3. The data flow shows a countdown in days when the information should be delivered. It also shows a document section and the obligation this data flow is submitted under. The document section is read only but shows all relevant documents for this data flow request. A copy of the dataset is as well available and ready for input. The structure is read only but the content can be entered.
4. The data provider can click the dataset object and view and interact with all tables and items that are contained in the dataset.



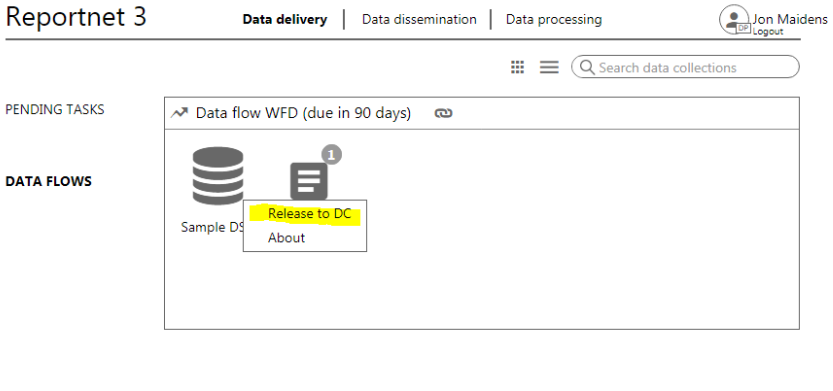
5. The data provider can right click the dataset and is allowed to generate multiple copies for testing purposes.
6. The data provider can now interact with the dataset and deliver data.

B. Countries deliver data

- See using forms (See video)
- See using files (See video)
- See using REST services (See video)
- See using files and INSPIRE services.

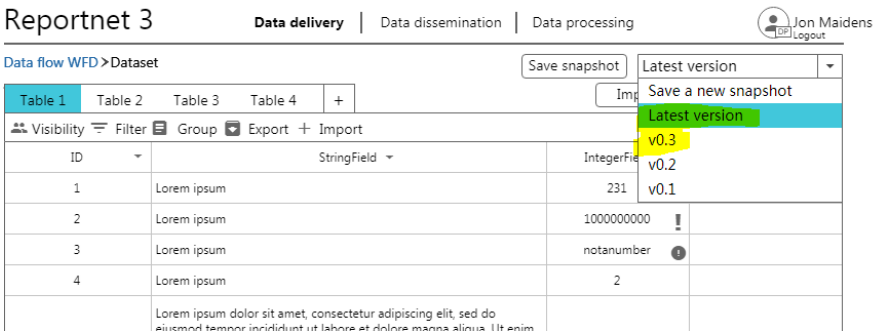
C. Deliver a data flow to the data requester

1. When the data provider find its data delivery ready for delivery. He/she performs the following steps.
2. The data provider right clicks on the dataset and selects "Release to data collector".



3. If there is more than one data collection that matches this dataset. A popup window will asks the data provider to select the right one.

4. Reportnet 3 automatically produces a snapshot of the current dataset (backup) and links this snapshot to the data collection.



5. The data collection is now ready for further processing – either quality control or EU dataset creation.

Conclusions and Recommendations

Conclusions

This new approach can provide a far greater flexibility on all stages of a data flow. During the creation of a data flow all stakeholders would get constant feedback because data and structure can be tested at the same time. During delivery of data towards a data flow countries could use different import mechanisms simultaneously and in many variations. When datasets can be copied the ability to test and production can be merged together providing a far friendlier user experience. When stakeholders can configure filters, validation rules from an interface also flexibility can be given to the implementation of quality control rules.

If we can produce a platform that allows multiple users to see the same data flow, dataset or data collection we can provide a working environment that stimulates collaboration and discussion. A “What you see is what you get” approach allows interaction of people with different backgrounds and reduces the technical barrier we currently have.

When data flows can be produced using a web interface without the need of software developers during that process we can assume that we can handle more data flows implementations within the same resources. It’s crucial that we reserve a budget for software development and maintenance of the infrastructure but always separate these from specific data flow implementations. Data flow implementations should have their own resource budget using the infrastructure without the need of development requirements.

A modern system like this requires a full operational REST API that allows integration towards external systems. Web services is providing flexibility and opens a door for innovation and integration. From a data provider perspective it allows to integrate their local infrastructures to ours. From a data collector point of view it allows to extend the functionality on output, validation or extraction.

Systems like Airtable and ArcGIS online demonstrate that data flows can be developed and implemented more rapidly when users create, test and implement a data flow in a collaborated environment. Because of it agile nature and strongly reduced requirement for an interventions by developer within the data flow implementation it became obvious that the time needed to implement a new dataflow can easily be reduced by a factor ten or more.

If we revisit our goals we are using to target how to realise the benefits of the new platform:

- **Better user experience and cooperation** – the wireframes showing how a user would interact with a database-centric platform, demonstrate collaboration can be tighter between actors at both the design and delivery phase, than they currently are in Reportnet 2.0.
- **Agile** – Taking the dual capabilities of collaboration and a platform which is capable of making snapshots at any point it time, we have the means for greater agility in both design phase and in the data delivery stage. Freeze an object, carry on working, undergo testing and quickly try out alternatives.
- **Efficient** – We want to achieve a more efficient deign and reporting workflow which will reduce the time in both phases for the requestors and reporters. There are

significant benefits to achieve this with the enablement of better collaboration and a platform which is closer in format to the reporter systems (databases) and does not require harvesting of files to complete quality control and creation of European level collections.

- **More accurate results** – We would expect to capture benefits of more accurate results by providing quality control feedback mechanisms which are all integrated in the reporting platform, faster, directly associated with the data in question and cover all aspects regardless of type. We would also expect to achieve this benefit from the reduced cycle time – this releases more time for the reporter and the requester to work on the quality of the data. Downstream will also benefit from having access to consistently formatted data earlier.
- **Scalable** – Having a centralised, web enabled system – configurable for each dataflow – means a simpler release and management architecture, which translates into being able to support more dataflows at any one time. The underlying storage platform which is inherently scalable in its architecture.
- **Integration** - The record based database storage would allow us to fully embrace server to server integration such as a record based REST-API for developers. Provide more freedom to the countries without adding complexity to the EEA. We would also gain benefits from being able to expose the reported data through services and take advantage of third party tools for further processing
- **Easier managerial overview** – One of the most sought after enhancements to the reporting system are dashboard overviews of the status of the dataflow. Currently there are many being created outside of the Reportnet 2.0 platform using Tableau to achieve this. They are an important part of the quality assurance, efficiency and collaboration aspects of the platform.

Recommendations

The implementation of such system would require a heavy development cycle that creates a strong central platform we would need to further extend over the years. As part of transition, we strongly recommend to develop an entire new interface for the current Reportnet 2 file reporting mechanism, without changing the internal workings of Reportnet 2. In parallel, we develop next to this platform a new infrastructure enabling the capabilities foreseen in this document. Over time this new infrastructure can replace the old implementations gradually when these data flows go into a redesign phase. For the end-user, they would have one website containing both platforms making this seamless for the end user. This new website should as well focus on the end users job by providing a personal view.

The transition period for when EEA is operating both Reportnet 2.0 for existing obligations and Reportnet 3.0 for new obligations will take some years. It is not the intention to force all reporting obligations to transition over to Reportnet 3.0 as soon as it is ready – this will be too costly and lead to many issues. Existing obligations will move over when it fits with their reporting cycle and the business case for doing so. However, the authors recommend that there is an upgrade to the Reportnet 2.0 interface so that it follows the workflow driven proposal for Reportnet 3.0. Giving the same look and feel for requesters and reporters to ease the transition.

Requirements

A list of requirements which the study has identified to enable the database centric approach which are to be added to the Reportnet 3.0 project requirements.

#	Requirement
1	Collaborative (multi-user platform)
2	Centralised
3	Web-enabled interface
4	Record based management of data
5	Relational database storage platform
6	Workflow driven interface
7	Flexible on delivery format of data (XML; CSV; JSON etc.) – decided by reporter
8	Handle full workflow of delivery, quality assurance and creation of European collection - for tabular data, spatial data and documents
9	REST API enabled for input and output of data
10	Configurable dashboards on status tailored for each actor perspective (reporter, steward, custodian, Commission, thematic expert, country responsible)
11	Capability for commenting on data structure in design phase and data in reporting phase
12	Capable of simple visualisation of data (maps, charts) as standard, and capability for more complex visualisation configuration
13	Enable pre-filling of data from previous reporting cycles and linkage to reference data (master data management)
14	Enable snapshots of data as engine for submission mechanism
15	Enable configuration of quality rules on records, datasets and collections and store outputs at same level
16	New interface for Reportnet 2.0 following Reportnet 3.0 look-and-feel
17	Module architecture
18	Centrally store quality feedback with data
19	Enable versioning of codelists

List of abbreviations

Abbreviation	Name	Reference
EEA	European Environment Agency	www.eea.europa.eu

References

- How to create a dataflow in Reportnet 2
- Dataflow checklist
- Airtable.com
- ArcGIS online
- Azure Cosmos 2018
- MongoDB
- Microsoft SQL Server
- PostgreSQL
- SAML2

Annex 1 – list of videos

#	Video	Section in feasibility document	Link
1	Creating a data structure using a web-based interface	3.1.2 Evaluation 1.1: Can we design a data model for data delivery using a web based user interface?	https://discomap.eea.europa.eu/trial/Reportnet3Study1/CreateBase.html
2	Duplicating a data structure (dataset) using a web-based interface	3.1.6 Evaluation 1.5: Can we re-use data structures for new data flows?	https://discomap.eea.europa.eu/trial/Reportnet3Study1/CopyBase.html
3	Creating a table by CSV import	3.1.2 Evaluation 1.1: Can we design a data model for data delivery using a web based user interface?	https://discomap.eea.europa.eu/trial/Reportnet3Study1/ImportSheet.html
4	Creating a form to populate data in a table	3.1.2 Evaluation 1.1: Can we design a data model for data delivery using a web based user interface?	https://discomap.eea.europa.eu/trial/Reportnet3Study1/Form.html
5	Exposing a table through a web service and populating with data	3.2.3 Evaluation 2.2: Can we alter the content by different input mechanisms in any order and time?	https://discomap.eea.europa.eu/trial/Reportnet3Study1/API.html
6	link to another record	3.1.3 Evaluation 1.2: Can we manage relationships between tables (1-n,n-n,n-1)?	https://discomap.eea.europa.eu/trial/Reportnet3Study1/LinkMultipleRecords.html
7	code ists - simple lookup that could be a hardcoded list	3.1.3 Evaluation 1.2: Can we manage relationships between tables (1-n,n-n,n-1)?	https://discomap.eea.europa.eu/trial/Reportnet3Study1/CreateCodelist.html
8	use code ists - simple lookup that could be a hardcoded list		https://discomap.eea.europa.eu/trial/Reportnet3Study1/UseCodelist.html
9	code lists import - relationships can be imported	3.1.3 Evaluation 1.2: Can we manage relationships between tables (1-n,n-n,n-1)?	https://discomap.eea.europa.eu/trial/Reportnet3Study1/ImportCodeList.html
10	imports data by XML	3.2.3 Evaluation 2.2: Can we alter the content by different input mechanisms in any order and time?	https://discomap.eea.europa.eu/trial/Reportnet3Study1/ImportFromXML.html

11	imports data by CSV	3.2.3 Evaluation 2.2: Can we alter the content by different input mechanisms in any order and time?	https://discomap.eea.europa.eu/trial/Reportnet3Study1/UpdateFromCSV.html
12	imports data by Python	3.2.3 Evaluation 2.2: Can we alter the content by different input mechanisms in any order and time?	https://discomap.eea.europa.eu/trial/Reportnet3Study1/InsertFromPython.html
13	user is going manually into a record and alters a value	3.2.3 Evaluation 2.2: Can we alter the content by different input mechanisms in any order and time?	https://discomap.eea.europa.eu/trial/Reportnet3Study1/EditRecordManually.html
14	collaboration: user enters data - shares for comments - issue identified	3.1.2 Evaluation 1.1: Can we design a data model for data delivery using a web based user interface?	https://discomap.eea.europa.eu/trial/Reportnet3Study1/ShareBase.html
15	collaboration: user enters data - shares for edits - issue identified	3.1.2 Evaluation 1.1: Can we design a data model for data delivery using a web based user interface?	https://discomap.eea.europa.eu/trial/Reportnet3Study1/ShareBaseOwner.html
16	a user can share a dataset so more than one person can enter data		https://discomap.eea.europa.eu/trial/Reportnet3Study1/EditorCollaboration.html
17	Create a dataset that is specifically designed to manage documents. This dataset could be copied for every data flow and be managed next to the dataset of this dataflow. The use of 'attachment' data type makes this possible	3.2.5 Evaluation 2.4: Can we handle shared documents or binary files?	https://discomap.eea.europa.eu/trial/Reportnet3Study1/Attachments.html
18	Show a script or FME process that validates the data using the API. We create hidden flag fields that get filled by the FME process	3.3.2 Evaluation 3.1: Can we implement validation checks over the data?	https://discomap.eea.europa.eu/trial/Reportnet3Study1/QCWithFME.html

19	Make/show a chart.	Evaluation 6.1: Can we generate QC outputs such as maps and dashboards?	https://discomap.eea.europa.eu/trial/Reportnet3Study1/CreateChart.html
20	Make/show a filter	Evaluation 6.1: Can we generate QC outputs such as maps and dashboards?	https://discomap.eea.europa.eu/trial/Reportnet3Study1/CreateFilter.html
21	Make/show a map	Evaluation 6.1: Can we generate QC outputs such as maps and dashboards?	https://discomap.eea.europa.eu/trial/Reportnet3Study1/CreateMap.html
22	A filter or graph could show those records that don't have a relationship. These visual tools could assist the data provider in understanding the errors a dataset contains and assist in the quality assurance process.	3.1.3 Evaluation 1.2: Can we manage relationships between tables (1-n,n-n,n-1)?	https://discomap.eea.europa.eu/trial/Reportnet3Study1/QCChart.html
23	collaboration: user enters data - shares as reado only - makes copy and extends	3.1.2 Evaluation 1.1: Can we design a data model for data delivery using a web based user interface?	https://discomap.eea.europa.eu/trial/Reportnet3Study1/ShareBaseReadOnly.html