Deltares

Enabling Delta Life

The Deltares Open Archive

Design

Deltares

Title
The Deltares Open Archive

Client
Delft-FEWS product management

Pages
1

Keywords
Deltares Open Archive; Delft-FEWS archive; THREDDS; GeoNetwork

Summary
This Document describes the design of the Deltares Open Archive (FEWS-build 2014.01)

References
Place references here

Version	Date	Author	Initials	Review	Initials	Approval	Initials
	Jan. 2014	Peter Gijsbers					
	April 2014	Peter Gijsbers		Grijze			

State
draft

Contents

1 Requirements analysis

1.1 Introduction

Many operational water agencies recognize the need for a water data archive. When being asked what needs to be archive, the typical answer is 'everything'. This answer is driven by a lack of deeper consideration what data is actually needed for what purpose, and how long a dataset is actually relevant for that purpose and how they actually think they will find the data when the archive has become terabytes in size. Storing 'everything' will also become very expensive, in hardware, in labour to the data available on appropriate devices and in the ability to find the data which is needed.

To design an appropriate data archive solution, a more detailed use case analysis is needed which identifies:

- what specific uses exist for the archived datasets
- what portion of the operational dataset needs to be preserved for a specific use
- how long the dataset needs to be preserved for this use
- how people intend to search for the dataset they need
- what the data accessibility requirements are (e.g. in terms of standardized or proprietary data formats)
- what the performance requirements are both in terms of discovery and retrieval speed

The end uses determine what datasets and metadata needs to be stored, thus having a major impact on the solution chosen.

1.2 Use Cases

1.2.1 Reviews and inquiries on event handling by the water agency

As indicated, events require heightened attention or even action from the organisation. Given the accountability of organisations for their actions, they may be faced with post-event reviews or even formal inquiries by external parties. Such inquiry will extensively focus on the decisions and actions taken (e.g. warnings issued, dam releases, flood defence actions or evacuation) with emphasize on the timeline of information availability that guided the decision making. To support a post-event inquiry or review, an archive thus should hold information that allows reconstruction of such timeline. This includes both scientific data as well as non-scientific data such as forecast products and records of communications. This is different from a need for exact reproducibility of results or even storage of all data including intermediate calculation results.

The scientific data record should provide insight in the observations and numerical weather predictions that were available at the moment the hydrological forecast was produced. This, together with the model configuration used for the calculation, the initial model states, the manual settings used by the forecasters and the final calculation results should enable forecast reconstruction without the need to store all the intermediate model results. At least as important for an inquiry, if not more important is the non-scientific data record. This should assist in the reconstruction how the forecasters came to their final issued forecasts, what communication was conducted when and to whom and how and when the decisions were made and communicated.

To conclude, an archive which is supposed to support a post-event inquiry or review should hold from start to end all scientific (observations, available NWP forecasts, hydrological forecast calculations and issued forecasts) and non-scientific data (forecast products, records of

communications) to allow reconstruction of a timeline for decision making. Post-event addition of the review findings to the archived dataset would make it more valuable for future use.

Organisations frequently have a legal requirement to keep such dataset for a legally prescribed period of time. A review or inquiry is a process which may take a few days or weeks. Completeness, readability and discoverability are the most important requirement, while it is acceptable when the retrieval process takes some time. Devices may be chosen which fit those needs.

1.2.2 Training material

Operational water agencies need to train their staff in the forecasting and warning process. Preferably this is done with realistic training material. Records of interesting events may provide suitable material, especially when the dataset has the extent as required for the review/inquiry case. Training may be a reason to keep the dataset for a longer period than the legally required term for review and inquiry. Given that training preparation normally does not require urgent data access, similar requirements can be defined for completeness, readability and discoverability and the duration of the retrieval process.

1.2.3 Post-event performance analysis

Hydrological forecasts performance assessment is another activity which operational organisations may want to conduct to assess their current skill and understand where they can improve their forecast capabilities. Such performance assessment only needs data for the variable of interest (e.g. water levels or precipitation). To enable such task, the archive needs observations, the final calculated forecasts and the issued forecasts as these can be used to calculate performance indicators addressing peak accuracy, lead time or timing of threshold crossing. Once the assessment is conducted, and the associated report is stored in the archive, some of the datasets may become less relevant. Post-event analysis does not require urgent access to these datasets.

1.2.4 Real-time diagnostic verification against historic events

Historic events provide diagnostic value to the forecaster to compare the current situation and simulated forecast with observed situations from the past (Demargne, et al., 2010). Quick and easy access in a matter of minutes, preferably by the forecasting system, is needed to allow forecasters to use historic information during a calamity situation.

1.2.5 Model Calibration

Model development and calibration is a common activity conducted for water systems analysis as well as forecasting. It requires a complete, quality controlled record of observations, both in terms of forces (e.g. precipitation and temperature) and water conditions against which to compare the model results (e.g. water levels, flows, wave heights, water quality indicators). Bad quality data in the archive requires additional effort to make the dataset suitable for model calibration. Preferably such quality control effort is conducted before the data is stored in the archive. Alternatively, the archive should allow update of the dataset after quality control. Model calibration does not require urgent access to these datasets. Open access to data, preferably using standard services and/or data format, is important to enable usage by a wide range of calibration tools.

1.2.6 Model Verification

Generally, a calibrated model is verified against a relevant portion of the dataset which has not been used for model calibration. Model verification of a forecasting model is best done by hindcasting and comparison against forecast forcing. Since these datasets can get voluminous, data administrators may choose to store forecast forcing only for interesting events. Again, this activity does not require urgent access to these datasets, while the forecasting system is the most obvious tool using the data.

1.2.7 Other use cases

A variety of other water systems analysis use cases could be imagined, ranging from statistical time series analysis for trends or extreme events to model development for water systems analysis. Typically these use cases require open access to long and complete records of quality controlled observations.

1.2.8 Summary of use case related data requirements

As can be seen in Table 1.1, various use cases require the same data while others only need a portion. In general, a continuous record of observations is needed, while most other data types only are needed for specific periods of interest. Data relevant to real time diagnostics for forecast verification is the only dataset which requires high availability and fast access (i.e. within seconds). All other use cases can cope with slower response times for discovery and retrieval as proper planning of data retrieval from the archive can prevent delay of the work process.

Table 1.1 Relevance of different data types for different use cases

Use case **Data type**	Event review/ inquiry	Training materials	Event analysis (Skill)	Informative event/ Diagnostic	Model calibration	Model verification
Observation	x	x	x	x	x	x
Simulation results (water forecasts)	x	x	x			
External forces (NWP forecasts)	x	x				x
Initial model state	x	x				
Model setup	x	x				
Modelrun settings	x	x				
Rating curves	x	x			x	
Forecast products	x	x				
Communication notes	x	x				
Post event analysis reports	x	x	x			

1.3 Other requirements

1.3.1 Metadata for discovery

When the event is known by the person searching for data, keys used to discover the data are the event name (if any), the start and end date (time) of the event and the area where the event happened. When a general search for events will be conducted, the area of interest will be known, while other search criteria may be needed such as threshold crossings or value crossings. Searching by threshold crossing makes searching easier as less local knowledge is required at the moment of searching. To enable searching by threshold crossings a metadata tagging mechanism is required to highlight occurrence of certain conditions (precipitation, flow, water level or prevailing wind conditions in direction and speed). This tagging could be done during data storage or afterwards by a local expert, or by a tool that automatically can analyse data and add the relevant metadata.

For those use cases that have no relation to events, search criteria are more focussed on obtaining long time series for relevant locations and quantities. In some situations, searches may be desired by related locations (e.g. upstream off) but this requires additional topological knowledge.

Table 2 identifies the search criteria to support a use case, where M=mandatory, D= desirable, O=optional and n.a. = not applicable.

Table 1.2 Search criteria to support a use case

Use case **Search criteria**	Event review/ inquiry	Training	Event analysis (Skill)	Informative event/ Diagnostic	Model calibration	Model verification
By Date (start-end)	M	M	M	O	M	M
By Area	M	M	M	D	D	D
By Event label	M	M	M	M	n.a.	D
By Location	D	D	M	M	M	M
By Location relation	n.a.	n.a.	n.a.	n.a.	O	O
By Variable	D	D	M	M	M	M
By Threshold crossing	D	D	M	D	O	D
By Value crossing	O	O	O	O	O	O

1.3.2 Metadata generation

Preferably metadata is generated during storage. Some metadata can only be generated at a later moment. Tagging an event with a name, start and end date, is generally a manual activity that has to be conducted as a post-event activity. As indicated, data mining tools may be applied to enrich the dataset with more metadata at a later stage, e.g. marking threshold crossings.

1.3.3 Reliable production of datasets

To support inquiries, comprehensive datasets need to be stored. The most reliable solution will continuously store all relevant scientific and non-scientific data and its metadata for the most extensive use case (inquiries/training) and remove unnecessary data as soon as is known that is not needed (i.e. no event has happened). Such continuous storage process should be reliable, having fall-back options that prevent data gaps when temporarily the data store cannot be reached.

1.3.4 Open access

Model calibration and water systems analysis are activities that may be conducted by a variety of software tools. To increase the range of applications, data should be accessible through open industry standards (e.g. OGC based services and/or data formats) as proprietary data formats reduce the ability of researchers and developers. The archive should allow storage of data from multiple sources. If proprietary formats are used, data conversion should be applied, either at storage time to keep a consistent format in the archive, or at retrieval time to provide the data in a standard format. Any time metadata requirements should be met to enable data discovery.

1.3.5 Archive data management

System administrators have different devices with different capabilities available to create an archive infrastructure. Archive data management requires understanding the data volumes involved and making decisions which datasets should be stored for what period of time on which devices. A data storage strategy is crucial to keep the archive maintainable with good performance. Event tagging for different use cases can be used to define such strategy. Preferably, such strategy can be encapsulated in instructions for a data management tool that can assist in data transfer (or removal) to the different devices.

1.3.6 Preserving integrity of the archive data store and catalogue

Data stored in an archive should not be lost. Preserving integrity of the data and the catalogue to find the data are essential. Solutions should be considered that accommodate restoring datasets and re-building of the associated catalogue.

1.3.7 Migration of existing archives

Water agencies often have existing hydrological archives. The data within these archives is valuable and needs to be pre-served when implementing a new archive system. Various strategies could be imagined, ranging from a mechanism to keep supporting the legacy system up till transfer of datasets and metadata to the new archive. Transfer has an advantage that an organisation can leave legacy technology behind.

1.3.8 Availability and performance

Service level requirements for infrastructure availability vary per use case. Archived data used during real time forecasting operations requires nearly continuous accessibility of the dataset with good performance. Most other use cases can cope with an archive which is unavailable for a few hours or where more time is needed to find and retrieve dataset.

1.3.9 Scalability

Over the year, the archive will keep growing with new data sets that keep growing in volumes. Scalability both for storage and the catalogue is crucial while keeping performance at an acceptable level.

1.3.10 Security

Many data archives hold proprietary data which should be protected against misuse. Security mechanism thus should be incorporated in the solution to control access to the data as well as tagging of metadata (events).

1.3.11 Sustainable solution with low maintenance costs

A data archive has to be a sustainable solution which can last for many years. Maintainability is crucial as the archive may need to survive personal or organisational change, both in terms of developers as system administrators. Complex solutions typically involve more components with more interactions and thus more potential points of failure. Simple designs based on industry standard technologies are easier to develop, maintain, administer and repair, thus increasing the sustainability of the archive.

1.3.12 Organisational ICT considerations

Backend systems, such as a data archive, are often managed by specialized ICT departments within a water agency. Typically, these departments have made organisation wide decisions for the technology stack that they are willing to support. This includes preference for hardware suppliers, operating systems and back end software such as RDBMS and application servers. Ability to acquire maintenance services are an important consideration. Open source technology only has an advantage if support can be acquired. OS platform independency is a valuable asset at any time.

2 Overview of the Deltares Open Archive

2.1 What is the archive

Figure 2.1 illustrates that an archive compromises a data store, a catalogue and an application that accesses the archive. The application(s) can produce data for the archive and consume data from the archive. The catalogue can be used to search the archive and find the data that is needed. To improve portability and secure accessibility across hardware, services are placed in-between the user application and the actual data store and catalogue.

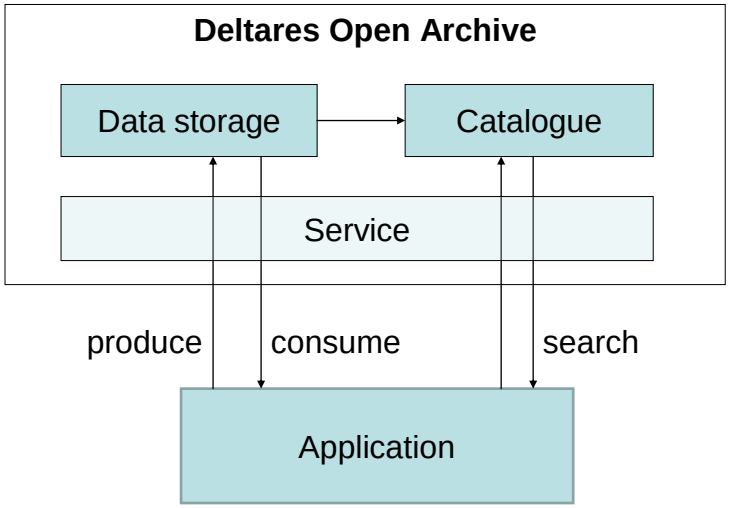


Figure 2.1 Main components of an archive

2.2 The application that uses the archive

It is foreseen that a wide variety of application should be able to use the archive. Among the applications foreseen are Deltares software packages such DeltaShell, Delft-FEWS, tools and scripts, e.g. written in Python or Matlab as well as other customer specific applications.

2.3 The data store: file/directory based

The use cases have illustrated that an archive for operational water management should extend beyond the storage of time series only. The archive should also be to store additional documents (e.g. PDF, doc, txt, html, csv, images, video, model specific data including run settings and model states) in relation to these time series. It is important to note however that

this variation in data sets (file formats, data structures) does not mean that the data or discovery process is unstructured. Generally, the dimensions of time, space, data source and production method/moment provide sufficient structure to store and find the data.

As indicated the requirements ask for a scalable solution with limited complexity, ease of maintenance and limited need for expensive (database) server hardware and software. In addition, there is the desire to create a solution that can works both in a once only project context (e.g. on a desktop) as well as in real time production environments on a server park context.

Given these requirements and the structured nature of the data to store, a file based solution is chosen, allowing easy expansion of storage space. Each data set gets its own directory with an associated metadata file describing the content of the data stored in this directory. Figure 2.2 illustrates that each directory holds a dataset of one or more files, as well as an accompanying metadata file.

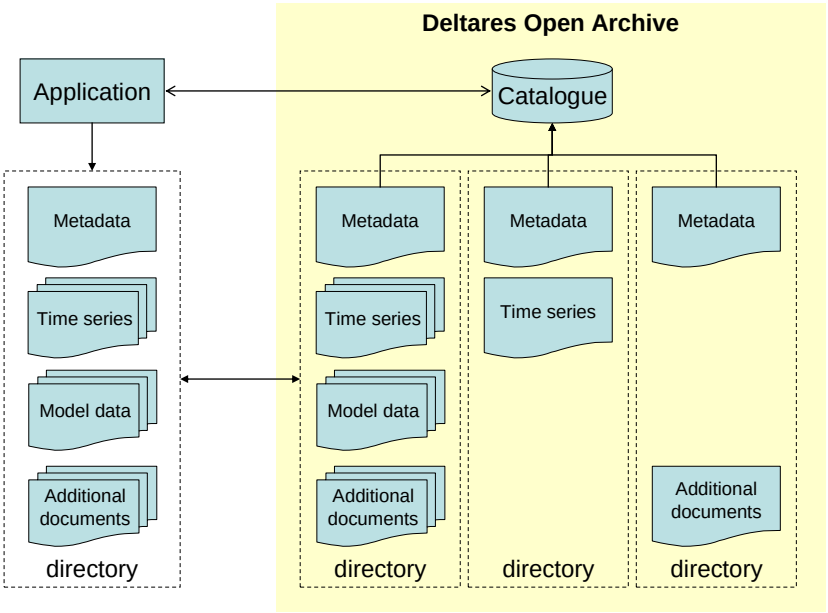


Figure 2.2 Overview of the file based Deltares Open Archive

Applications that use the data from the archive can access the files directly or obtain them through a server.

2.4 The Data sets: a collection of files in different formats

The Deltares Open Archive is meant to support the water management organization in the storage of data relevant for traceability in decision making in operational forecasting and model studies. While traceability is important, reproduction of results may not always be a requirement. The Deltares Open Archive is not intended become a versioning system for model setup and schematization. However, model setups should be storable as they are needed to make results traceability and understand their creation process.

The data to be archived is organized by data sets. A data set comprises one or more files holding interrelated content to be archived as a whole. Each data set is grouped in a directory and can be described by one metadata file.

Table 2.1 provides an overview of the data sets that will be stored in the Deltares Open Archive for operational forecasting purposes. In this context, data will be provided by Delft-FEWS. However, other applications could provide data sets to be archived as well, as long as the metadata requirements are met. More details are discussed in the chapter on data sets for archiving.

Table 2.1 Data sets to be stored in the Deltares Open Archive

Data Set name	**Data types**	**FEWS production process**
Observations	Historic Time series (nc)	Continuous export, interval based
Messages	Message (txt)	Continuous export, interval based
External Forecast	Forecast time series (nc)	Continuous export, interval based
Simulation	Run info (xml) (Forecast) Time Series (nc) FEWS Modifiers (PI-xml) Web-reports (html/zip/pdf) Model states (zip) ThresholdCrossings Skill (csv)	Export with forecast run
Configuration	FEWS Configuration (zip)	Continuous check, Interval based Incidental export when updated
Rating curve	FEWS RatingCurve (PI-xml)	Continuous check, Interval based Incidental export when updated
Event attachments	Reports (PDF/doc) Other attachments (zip)	Manual export

As indicated in the previous sections, the data to be stored are generally time series, model setup and documents. Each of them can come in a variety of file formats and data structures. To achieve the desired 'openness' of the archive, some industry standards are chosen for the file formats in which data is stored.

For time series, the netcdf file format (nc) has been chosen using the CF-convention. This format is an international standard, fit for multi-dimensional data and supported by a wide variety of software packages, tools, programming and scripting languages.

For documents, a variety of standard file formats is chosen, such as PDF, txt, xml, csv and html. For model related data, the zip format has been chosen, as this container allows the packaging of application specific content in a generic way.

2.5 The metadata: key-word based

Archives become valuable when it is easy to discover the data stored. Generally, interesting information is discovered by searching a catalogue which holds metadata, thus preventing the need to access the actual dataset to answer the query. Given that a large portion of the data to be archived is numeric, an important decision is whether the user should be able to query the data for crossing of a specific number (e.g. Give the water levels in Rhine at Lobith where the value is above 12m+MSL).

After discussion with end users, it became clear that users would like to search for threshold crossings, but not necessarily for numbers. (E.g. Give me the water levels in Rhine at Lobith where the value is above the flood watch level). Searching for data based on topological or topographic knowledge was out of scope (e.g. the water levels for all stations in the Rhine upstream of Lobith where the threshold 'flood watch' was crossed). Searching for any locations in the Rhine where a threshold was crossed was considered used (e.g. the water levels for all stations in the Rhine where the threshold 'flood watch' was crossed).

Given this scope and the associated decision to enable searching for threshold crossings instead of crossings of actual numbers at a particular location, an important solution direction is chosen: the catalogue should support searching by key word (e.g. threshold crossing 'flood watch') instead of numbers without the need to incorporate detailed topological knowledge.

Within the Deltares Open Archive, the content of the catalogue is exactly the same as metadata held in the metadata files that accompany the data set. A harvester process reads the metadata files and populates the catalogue database (see section 4.3.3 for details).

2.6 Events: a special case of metadata

Data discovery strategies for scientific purposes typically start with an area and period of interest, followed by more query details such as locations, variables and quality/status indications. Many of the operational use cases have an additional item in common: they can be related to an 'event', e.g. a storm, flood, drought or spill event. An event can be labelled by a meaningful name, covers an area and a period of interest (e.g. Hurricane Sandy, NJ-NY coast, 28 October – 31 October 2013).

Events can thus be used to discover relevant data in the archive. Events can also be related to use cases, e.g. some events are appropriate for training purposes, others for model calibration. So-called historic events are a specific type for Delft-FEWS as the associated historic time series can be overlaid on top of the current time series.

The Deltares Open Archive acknowledges events as a special type of metadata overarching multiple data sets. The area and period covered by the event determine, in combination with the type of event and associated data types, which data sets belong to an event.

Since use cases differ in data needs, end-of-life and backup strategies, these items can be related to events as well. A data set included as part of an event thus may have different storage and device needs compared to data sets which are not part of an event.

Generally, the events metadata is created manually after the data sets have been stored in the archive and after the (storm) event has passed away.

2.7 Services to enable access

The file based solution of the archive, both in terms of data sets and associated metadata, allows applications to directly access data by disk. To answer queries to the archive, an application could create its own internal representation of the 'catalogue' by scanning all

metadata files on the file system. Once the data set is identified, the data could directly be read from disk.

This solution is the least complex from the view point of processes to manage, but it will in general not be acceptable from the perspective of the access restrictions as well as from the perspective of performance. A more common approach will be using a catalogue server and a data server or a combined server which encapsulated the two functions.

A catalogue server will have its own cache representation of the metadata files to improve query performance. This cache is populated by a crawler process which scans the file system and hands over the metadata to the catalogue.

The Deltares Open Archive supports the OpenGIS Catalogue Service (http://www.opengeospatial.org/standards/specifications/catalog). The Deltares Open Archive catalogue application is based on GeoNetwork open source (http://geonetwork-opensource.org/).

The data server will directly work of the file system, providing an http-service to access the data sets. The Deltares Open Archive data server is based on THREDDS (https://www.unidata.ucar.edu/software/thredds/current/tds/). The THREDDS Data Server (TDS) is a web server that can provide metadata and data access for scientific datasets, using a variety of remote data access protocols. These include the protocols from OPeNDAP (http://www.opendap.org), OGC Web Mapping Service (http://www.opengeospatial.org/standards/wms) and OGC Web Coverage Service (http://www.opengeospatial.org/standards/wcs). Non-scientific data sets such as documents and zip files are supported by the TDS via the standard http-get protocols.

The default Deltares Open Archive setup thus will compromise a catalogue server where the response is a server/file path and a data server to retrieve the files of the data set given the search result.

In future, a combined server is foreseen which hands the application query to the catalogue server, passes the result to the data server and returns the resulting dataset to the application.

2.8 Data provision to archive

The THREDDS Data Server provides an access service to the data. It does not provide additional support process to add or update new data to the archive, or conduct archive management tasks. Other software processes are needed on the backend to conduct those tasks. Most task can be scheduled on an interval. A command line call by a linux cron job or a Windows Task Scheduling job has been chosen as a lightweight solution.

Given the requirement to enable restriction of write access to the data store, in combination with a desired to minimize the need for dedicated monitoring of critical processes, a solution has been chosen that reduces criticality of the backend process.

- Real time operational systems are allowed to directly write data sets (in batches) to the data store. Permissions need to be setup such that the FEWS Forecasting Shell Server (FSS) has direct write access to the data store. No additional process is needed to ingest the data into the archive data store.
- Other applications may (eventually) be able to upload data to the archive via an HTTP service (ArchiveWebServer). In the 2013.02 build, this feature is only implemented for events and event-attachments
- A FileSweeper will clean up any issues related to file locking

- A Harvester will crawl through all directories and scan the metadata files for any updates. Metadata updates and converted into a native format for GeoNetwork and ingested in the catalogue.

Failure of any backend maintenance process (FileSweeper, Harvester, HistoricEventsExtractor) only results in an archive which is slightly out of date when this approach is used.

An exchange directory has been considered to allow any application to provide data to the archive. However, due to file and directory permission issues, an ArchiveWebServer has been introduced.

The decision to allow direct writing of a high volume continuous data stream from the operational system to the archive has been based on the need to minimize the risk of critical system failure due to unavailability of the ArchiveWebServer.

2.9 Archive data management

Archives need to be managed to prevent costly overgrow. Archive data management compromised activities such as backup of data sets to slower disks or tape, restoring of data sets or complete removal of data sets. These data management activities need to be harmonized with the procedures that are implemented to provide the data for archiving.

Within an operational context, it seems logical to continuously the archive with the latest data. When nothing interesting has happened, datasets may quickly become irrelevant. To reduce storage needs and costs, data may even be deleted from the archive. However, such cleanup should be prevented for those situations where something interesting happened. These situations are typically also the reasons why people want to search in the archive for data. The Deltares Open Archive uses *events* for this purpose. Events are a way to cluster datasets and mark them as being relevant. Events can be used to search the archive and discover relevant data. Additionally, events may help in defining a strategy for data management of the archive.

Within the Deltares Open Archive a manual solution is chosen where the System Administrator is in control of data management. The solution uses the creation date of a dataset (stored in the metadata) as well as instructions defined by data type. The instruction defines per data type within a dataset what storage device to use given the age of the data set. If a dataset is part of an event, the settings for this event type may overrule the dataset default.

The System Administrator uses an ArchiveDataManagementTool to scan the archive and create a file which lists the actions to be taken to clean up the archive.

2.10 Archive management console

The archive management console is not yet available. The console will become an administrative web-interface, which provides:

- Insight in the last run/update of the archive maintenance processes (Harvester, FileSweeper and HistoricEventsExporter,)
- Ability to kick off (once only) each archive server process individually
- Ability to run the ArchiveDataManagementTool

2.11 Data flow

To summarize the flow of data and associated data processes as discussed a data flow diagram has been created (Figure 2.3).

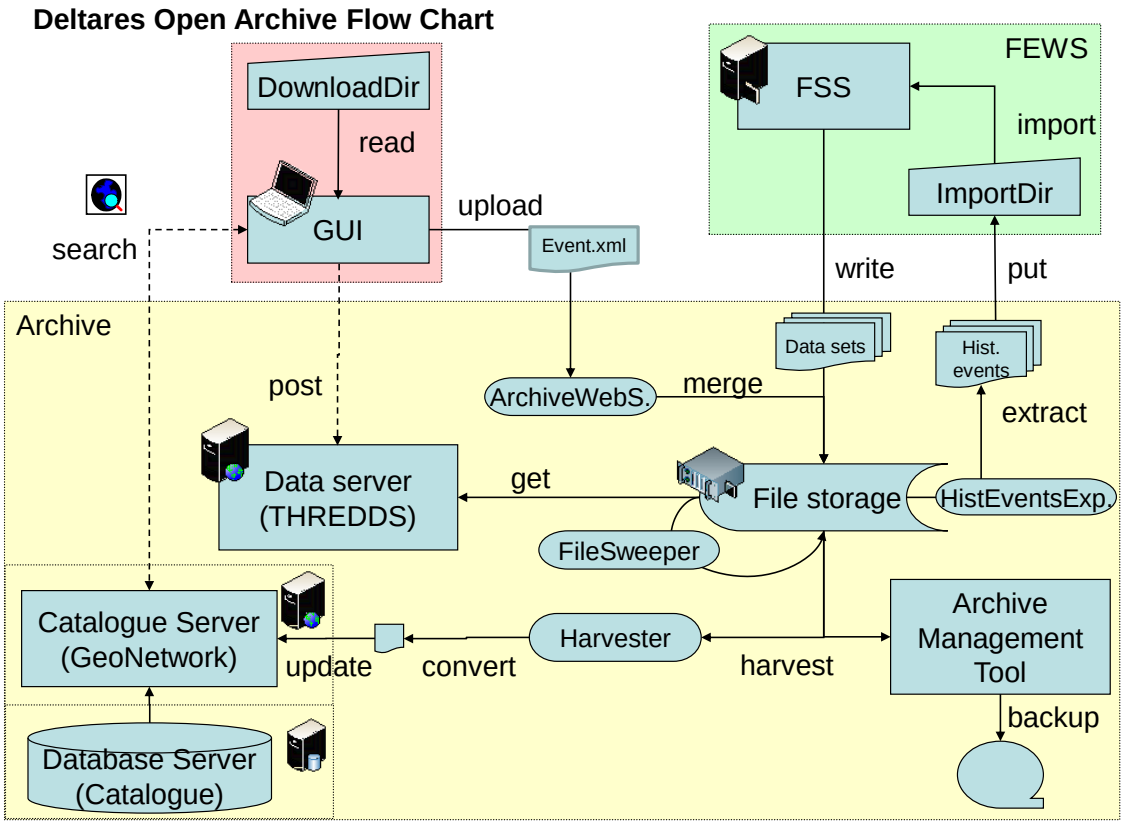


Figure 2.3 Data flow between the various components of the archive

Data sets, including a metadata file can be provided to the Archive, are directly provided by a FEWS FSS. In future, the ArchiveWebServer service can be expanded to accommodate upload of full datasets.

For each dataset, only one version is stored. In case a dataset is updated, file locking may be an issue. In that case, the new dataset receives a temporary file name. A daemon process (the FileSweeper) scans the data store file system frequently for temporary file names and replaces the old file with the new file once the lock removed.

With each new data set of data set update, metadata may change as well. A daemon process (the Harvester) scans the data store file system frequently for metadata updates. Any metadata file change is converted into the native import format for GeoNetwork to update the catalogue.

Delft-FEWS uses ‘Historical Events’ to provide the forecaster diagnostic support using experiences from the past. A daemon process (HistoricEventsExporter) scans the archive for any recent updates (last x months) in the events definition and extracts all events as well as the associated observations for the most recent events (last 10 days). This data is provided to the Delft-FEWS application via the Import Directory of the FSS. Once imported in Delft-FEWS, historical events automatically are available in the Operator Client.

For any other user application, manual interaction is needed with the archive. Generally, the application puts a request to the Catalogue server and receives a response. The response holds the paths to the data sets as well as the metadata in the native Deltares Open Archive format. Using this information, the user application calls the Data Server to obtain the information. The data server provides the data to a Download directory. The user can use the data directly from this Download directory, or move the data to a place where it can be used.

2.12 Logical server components and its permissions

In the end, the archive has five logical service component which require specific permissions (see Table 2.2):

1 The Forecasting Shell Server, writing data to the archive file server.

2 The THREDDS Data service, hosted in Tomcat on the archive file server

3 The Catalogue service, hosted in Tomcat in a separate server

4 A Catalogue database instance, hosted in a database server

5 The Archive web server, typically hosted in Tomcat on the archive file server

   – Events management

6 The archive server processes, hosted and executed on the archive file server

   – Harvester (scheduled)

   – FileSweeper (scheduled)

   – HistoricEventsExporter (scheduled)

   – DataManagementTool (manual executed)

Table 2.2 Permissions on the various folder of the archive

Folder **<root>=/data/archive/<application>/**	**read permission by component**	**write permission**
<root>/data	THREDDS DataManagementTool Harvester FileSweeper HistoricEventsExporter ArchiveWebServer	FSS FileSweeper
<root>/events	ArchiveWebServer	ArchiveWebServer
<root>/eventsbackup	ArchiveWebServer	ArchiveWebServer
<root>/archiveConfig	THREDDS	system administrator
????/tofss/Import/historic_events	FSS	HistoricEventsExporter

3 Datasets for archiving

This chapter describes the data sets which can be handled by the Deltares Open Archive. The data sets are described from the perspective of the creation by the Delft-FEWS operational system.

3.1 Observations

3.1.1 Description of the Observations dataset

An Observations dataset holds one data type: time series, stored in NetCDF files.

Observations are historical time series data, continuous over time, with a regular or irregular time step. Typically these time series are scalar (0-dimensional in space). At each moment in time only one numerical value applies per station and variable. For scalar time series, a quality flag and string comment can be included for individual time stamps

This type of dataset consists of one or more NetCDF files and a 'metaData.xml'. The NetCDF files contain the observation timeseries and the 'metaData.xml' file contains meta-data about the content of the NetCDF files.

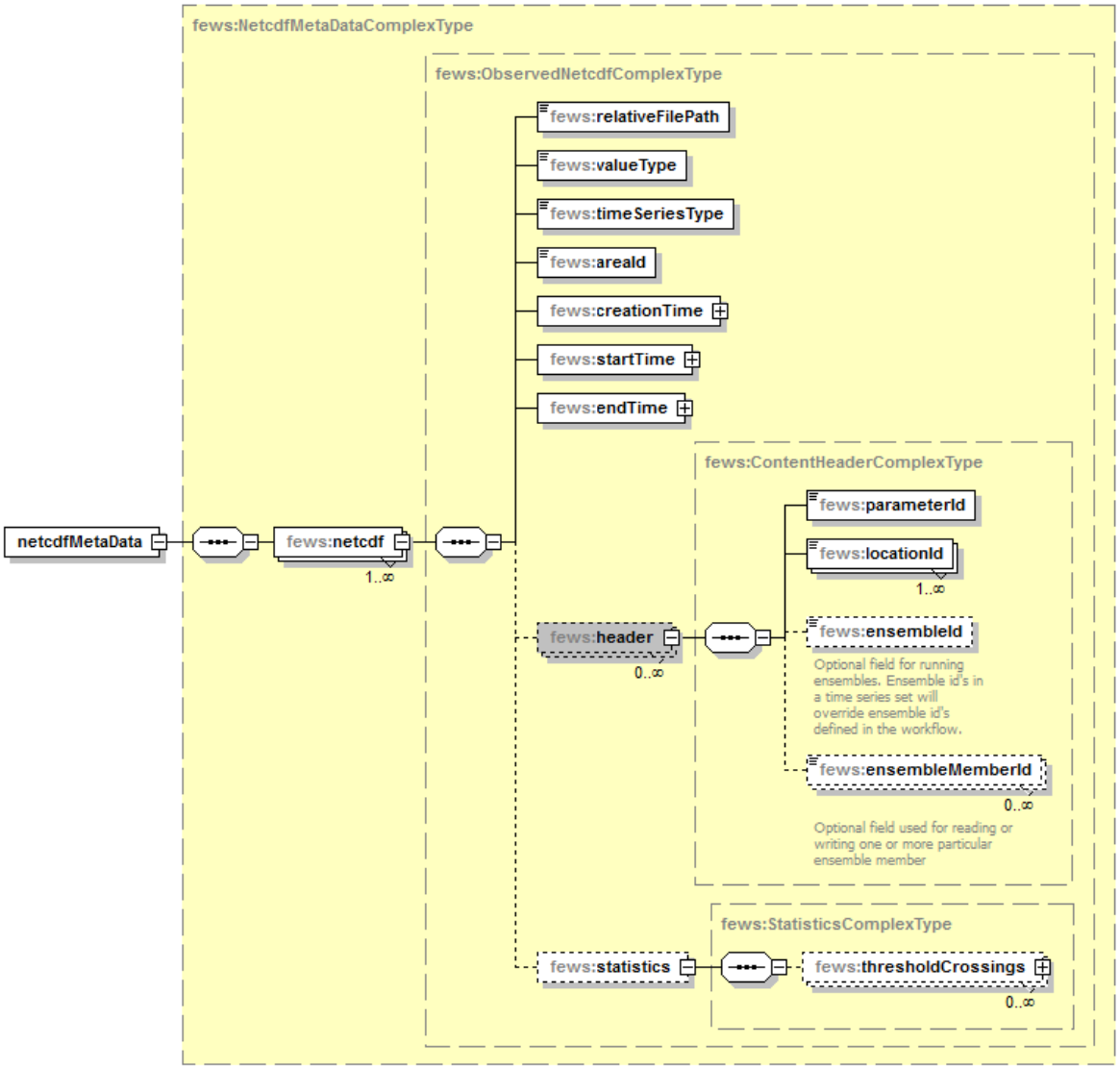
Observation timeseries are stored in NetCDF (1.4) using the data structure and metadata definitions as prescribed in the CF metadata convention (http://cf-pcmdi.llnl.gov/). Multiple stations and variables can be included in one netcdf file.

3.1.2 Time Series Metadata

The metadata file for this data set follows the schema definition as defined in:

http://fews.wldelft.nl/schemas//version1.0/archive-schemas/netcdfMetaData.xsd.

See Figure 3.1 and Table 3.1 for the detailed explanation.


fews:NetcdfMetaDataComplexType
fews:ObservedNetcdfComplexType
fews:relativeFilePath
fews:valueType
fews:timeSeriesType
fews:areaId
fews:creationTime
fews:startTime
fews:endTime
fews:ContentHeaderComplexType
fews:parameterId
fews:locationId
1..∞
netcdfMetaData
fews:netcdf
1..∞
fews:header
0..∞
fews:ensembleId
Optional field for running ensembles. Ensemble id's in a time series set will override ensemble id's defined in the workflow.
fews:ensembleMemberId
0..∞
Optional field used for reading or writing one or more particular ensemble member
fews:StatisticsComplexType
fews:statistics
fews:thresholdCrossings
0..∞

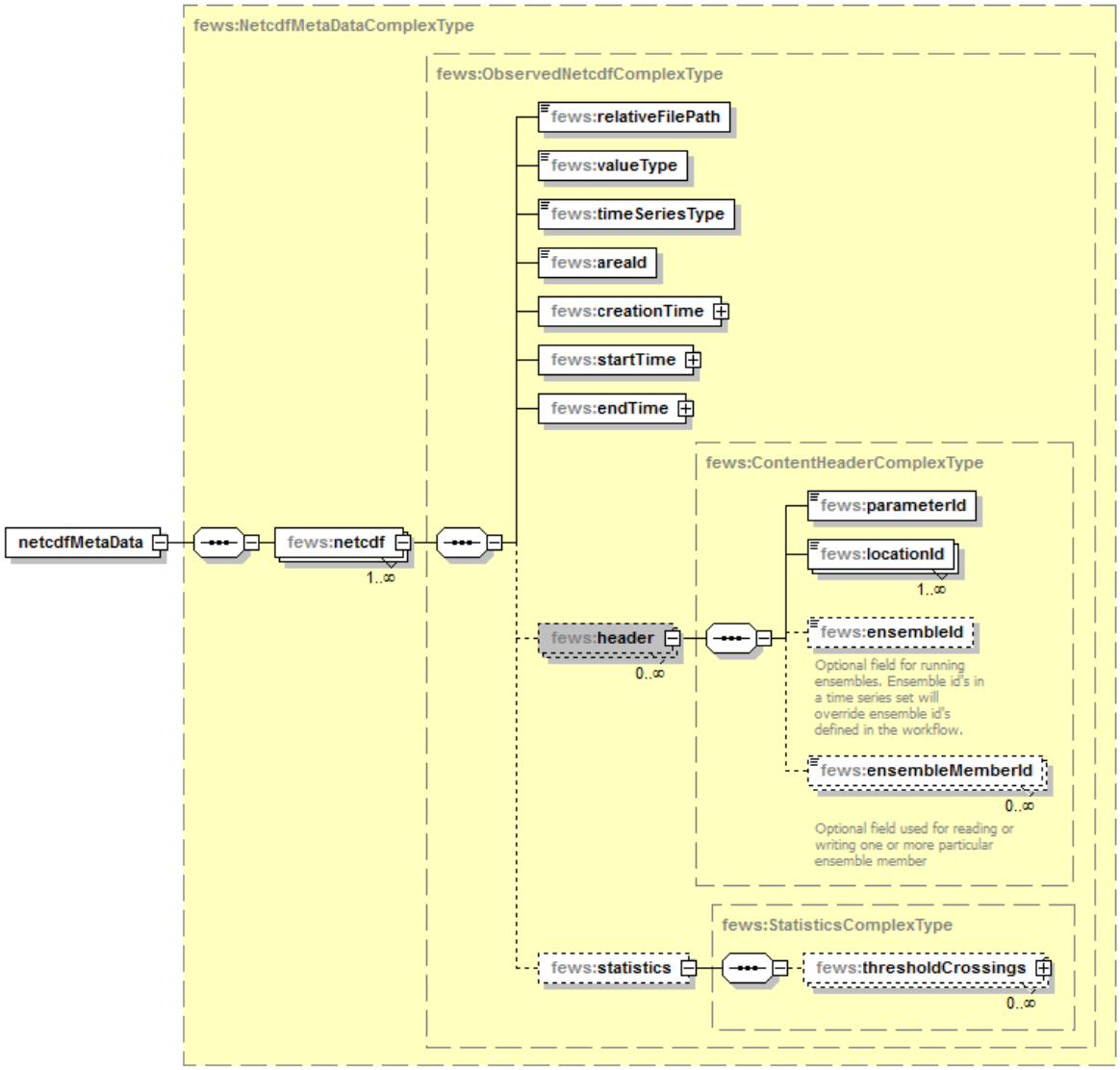


Figure 3.1 Metadata describing the contents of a netcdf file

Table 3.1Metadata content of time series (Netcdf ComplexType)

Element Name	Format	Description
relativeFilePath	string	file path relative from metadata file
valuetype	enum(1): grid scalar	Determines appropriate netcdf parser
timeSeriesType	enum (1): observed simulated externalForecast	Determines directory structure
areaId	string	area reference, Use no space, will become part of directory path

creationTime	attributes date time	Creation time of data set as posted to archive
startTime	attributes date time	Start time of time series held in file
endTime	attributes date time	End time of time series held in file
header	ContentHeader ComplexType	
header-parameterId	string, use underscore (_) no dot (.)	variable name in NetCDF header
header-locationId	string, use underscore (_) no dot (.)	station name in NetCDF header
header-ensembleId	string	ensemble/realization identifier
header-ensembleMemberId	string	ensemble/realization member identifier
statistics	Statistics ComplexType	
statistics-thresholdCrossings	ThresholdsCrossings ComplexType	
thresholdCrossings-header	StatisticsHeader ComplexType	just one location, rest same as normal ContentHeader
thresholdCrossings-thesholdCrossing	ThresholdsCrossing ComplexType	
thesholdCrossing-thresholdId	string	identifier of threshold crossed
value	double	threshold value crossed

3.2 Messages

3.2.1 Description of Messages dataset

The Messages dataset holds one data type: text, stored in text files.

Typically, all text from the same origin (e.g. system logs, model logs or Forecaster Notes) are collected together and stored in one file. Generally, each message in such file has a time stamp associated with it. All messages are collated by day and stored by day. This type of dataset consists of one or more text files and a metaData.xml file.

3.2.2 Messages Metadata

The metadata file for this data set follows the schema definition as defined in:

http://fews.wldelft.nl/schemas//version1.0/archive-schemas/messagesMetaData.xsd.

The dataset may hold multiple message files, where each file is described by its metadata (see Figure 3.2 and Table 3.2).

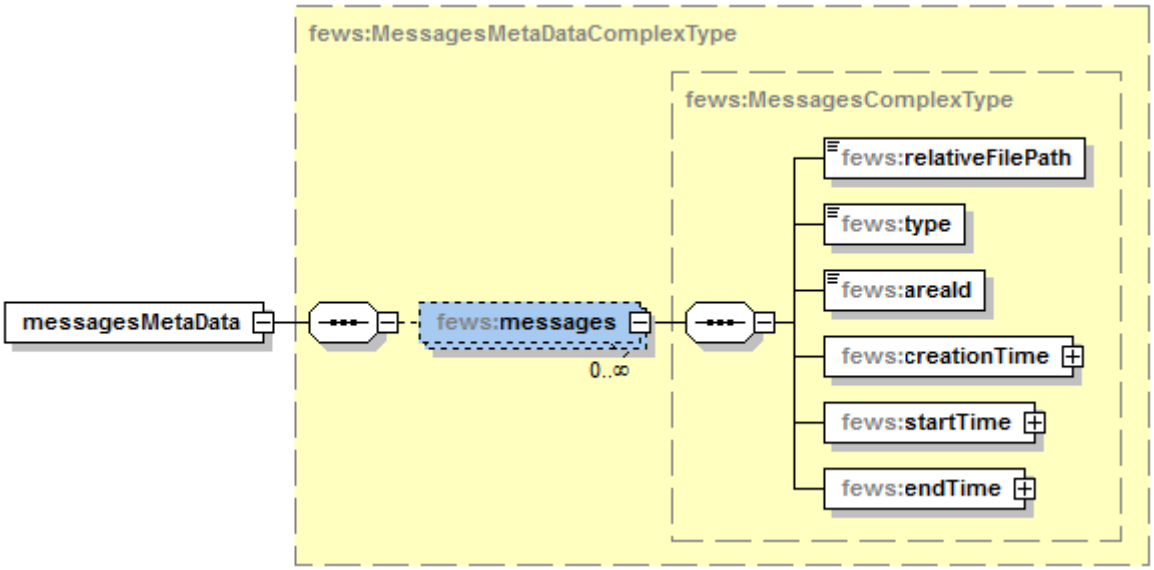


Figure 3.2 Metadata for messages according to messagesMetaData.xsd

Table 3.2 Metadata content of messages

Element Name	Format	Description
relativeFilePath	string	file path relative from metadata file
type	string	as understood by software
areaId	string	area reference for which message applies Use no space, will become part of directory path
creationTime	attributes date time	Creation time of dataset as posted to archive
startTime	attributes date time	Start time of messages held in file
endTime	attributes date time	End time of messages held in file

3.3 External forecasts

3.3.1 Description of external forecasts dataset

The external forecasts dataset holds one data type: time series, stored in NetCDF files.

External forecasts refer to forecast time series from external sources such as UK MetOfice, KNMI, NCEP, ECMWF etc. Characteristic of a forecast time series is the fact that multiple versions may exists at a particular point in time. Each version is identified by its reference or forecast time, e.g. the 11am forecast. The time stamp when the value applies is typically called valid time. Given that hydrological operational forecasting systems may use multiple NWP products, which all share the same reference time, separate datasets need to be archived by NWP source.

External Forecast timeseries are stored in NetCDF (1.4) using the data structure and metadata definitions as prescribed in the CF metadata convention (http://cf-pcmdi.llnl.gov/). For gridded timeseries, each file contains one grid and one or more variables. For scalar time series, multiple stations and variables can be included in one netcdf file.

This type of dataset is accompanied with a metaData.xml file.

3.3.2 Time Series Metadata

The metadata file for this data set follows the schema definition as defined in:

http://fews.wldelft.nl/schemas//version1.0/archive-schemas/netcdfMetaData.xsd.

This is the same metadata definition as discussed for the observations (see Figure 2.1).

The content of the metadata file differs in the choice of timeSeriesType (externalForecast), while the valueType will be 'grid' in most situations. Currently, no statistics can be stored in the metadata for external forecasts.

3.4 Simulations

3.4.1 Data set description

General

The Simulations data set may hold a variety of data types, e.g.

- time series, stored in NetCDF files
- model states, stored in native model/application format
- run info in xml format
- model run settings stored in native application format (e.g. FEWS modifiers in PI-xml)
- reports stored in document formats (e.g. HTML, PDF)

Characteristic to this dataset is its origin: a simulation. This simulation could be a State updating run or a Forecast run in a forecasting application (e.g. Delft-FEWS), or it could be a reference or scenario run by any other simulator (e.g. a Python script or model application like DeltaShell).

The dataset generally holds every data item which is needed for full traceability of the result. This includes the time series as generated by simulation, possibly the time series that are the boundary conditions for the simulation, the model state (if deviating from the default), model run settings, including any deviations from the default, etc. If the dataset holds the 'default' or reference, it may even include the entire model setup.

Timeseries are stored in NetCDF (1.4) using the data structure and metadata definitions as prescribed in the CF metadata convention (http://cf-pcmdi.llnl.gov/). For gridded timeseries, each file contains one grid and one or more variables. For scalar time series, multiple stations and variables can be included in one netcdf file.

Model run settings are stored in native application format.

Reports are stored in HTML, ZIP or PDF format.

Delft-FEWS specific files in the simulations datasets

For Delft-FEWS the run settings are defined in the runinfo.xml file, which meets the schema as defined in http://fews.wldelft.nl/schemas//version1.0/archive-schemas/runinfo.xsd.

As can be seen in Figure 3.3, this runinfo file includes information on the configuration version used to create the results.

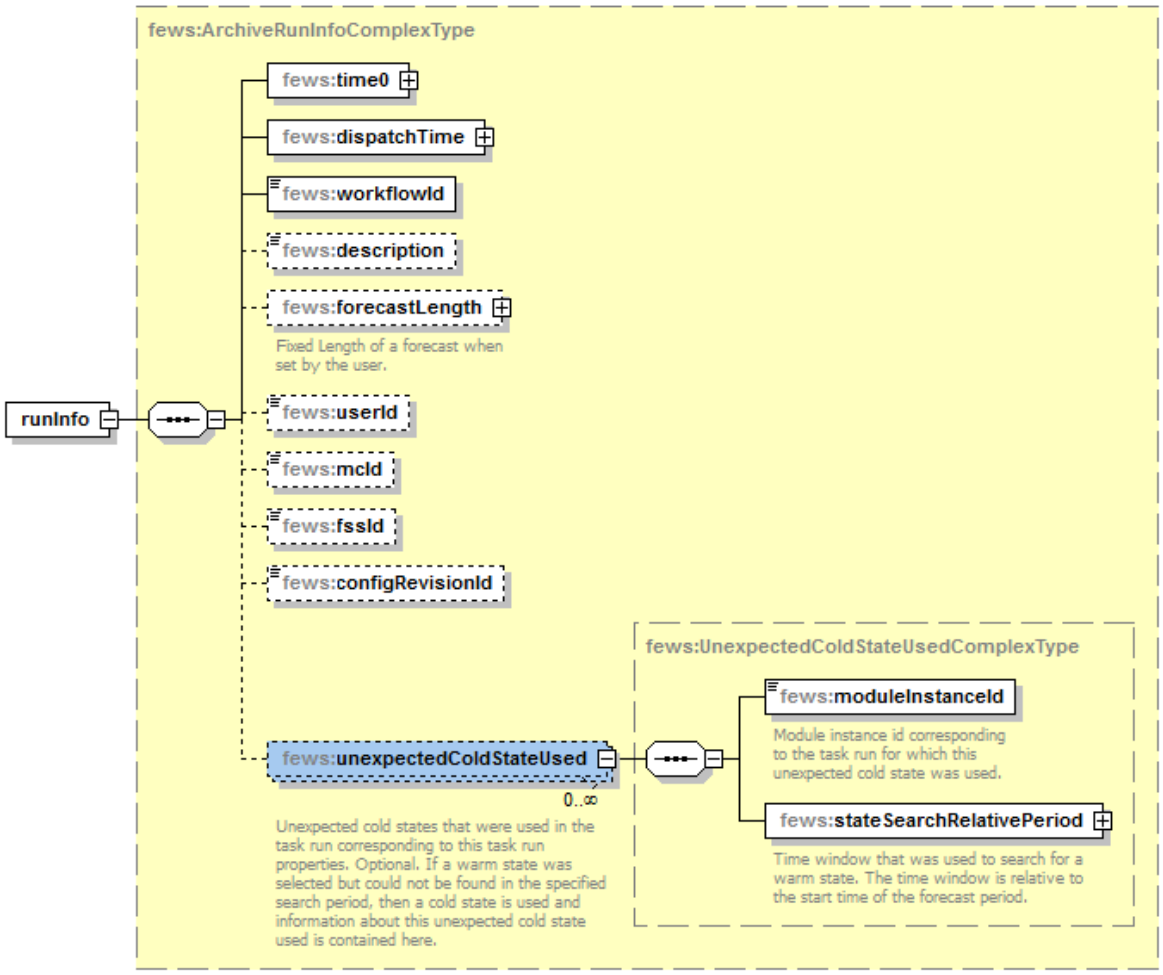


Figure 3.3 Delft-FEWS runinfo.xsd

Modifiers used in this run are stored in the modifiers.xml, which follows a FEW PI-schema: http://fews.wldelft.nl/schemas/version1.0/pi-schemas/pi_modifiers.xsd. At the moment, only location attribute modifiers can be stored (see Figure 3.4).

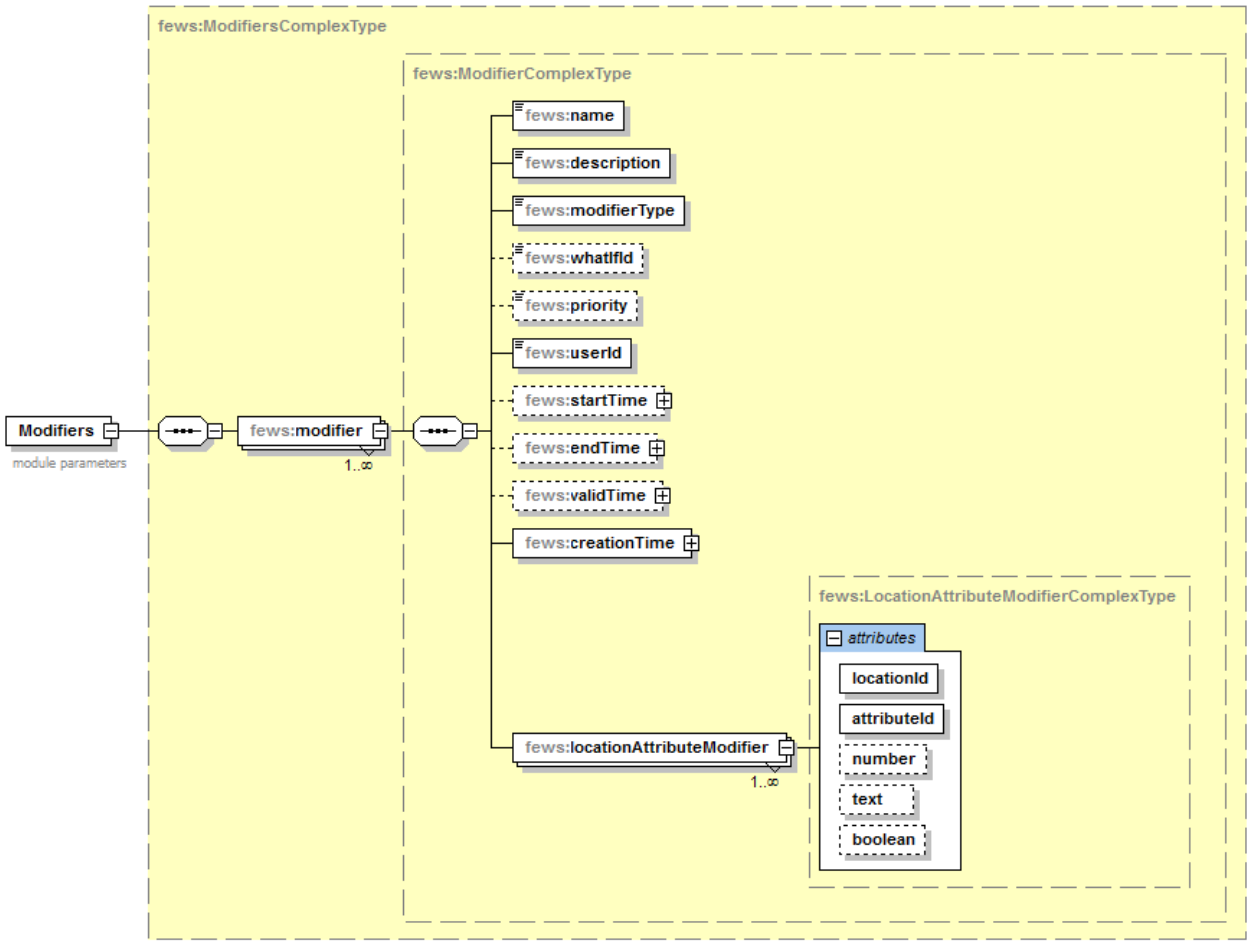


Figure 3.4 Delft-FEWS PI-schema modifiers.xsd

3.4.2 Simulations Metadata

The metadata file for this data set follows the schema definition as defined in:

http://fews.wldelft.nl/schemas//version1.0/archive-schemas/simulationMetaData.xsd.

As can be seen in Figure 3.5, the associated metadata file holds references to each individual portion of the dataset

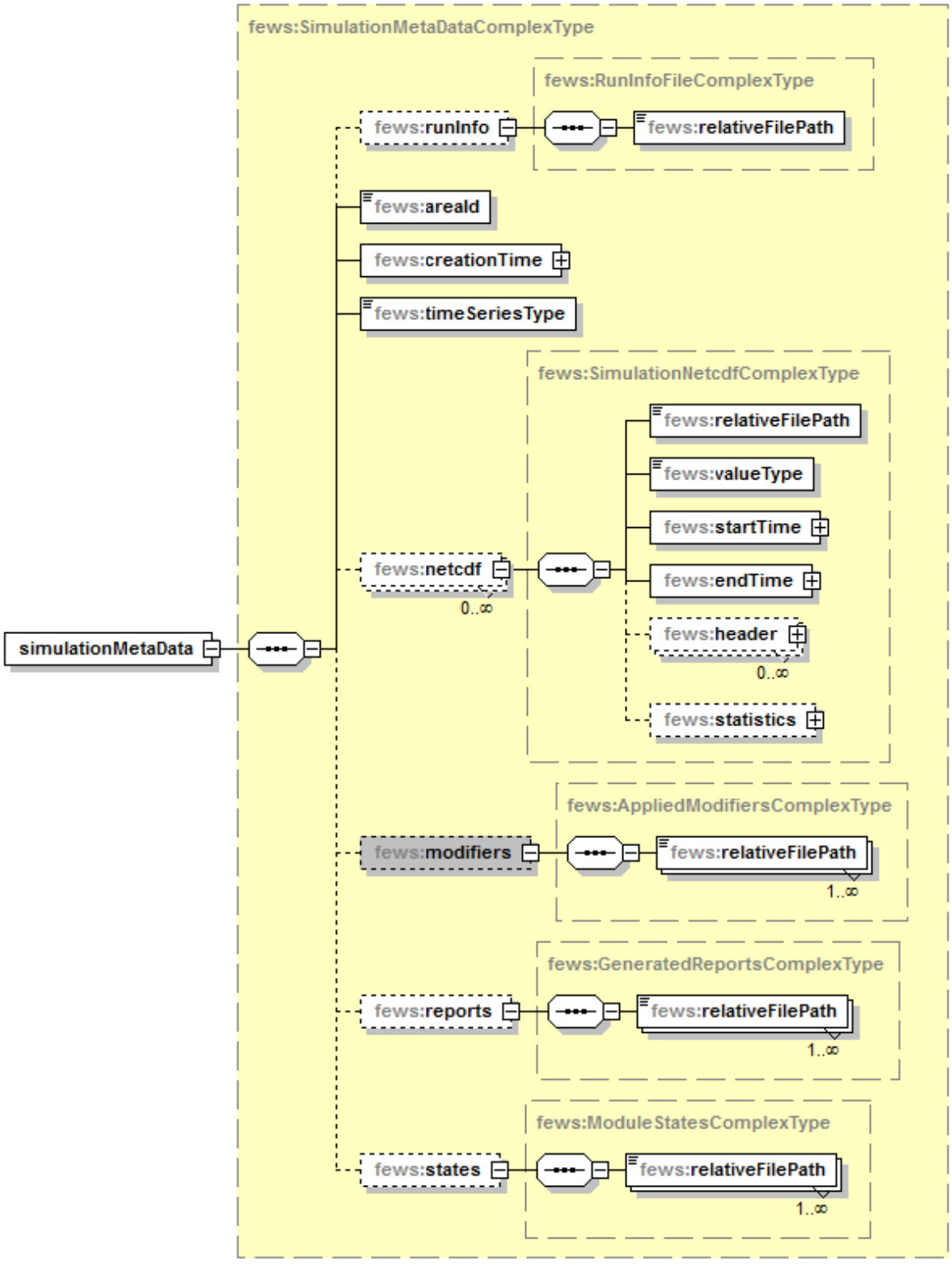


Figure 3.5 Metadata for simulation datasets according to simulationMetaData.xsd

Table 3.3 Metadata content for simulation

Element Name	Format	Description
runInfo	RunInfoFile ComplexType	
runInfo-relativeFilePath	string	path to runInfo file relative from metadata file
areaId	string	area reference, Use no space, will become part of directory path
creationTime	attributes date time	Creation time of data set as posted to archive
timeSeriesType	enum (1): observed simulated externalForecast	Determines directory structure. Typically simulated
netcdf	simulationNetCDF ComplexType	subset of the normal netCDF ComplexType. For details see section 3.1.2
modifiers	AppliedModifiers Complexype	
modifiers-relativeFilePath	string	path(s) to modifier data file(s) relative from metadata file
reports	GeneratedReports ComplexType	
reports-relativeFilePath	string	path(s) to report zip file(s) relative from metadata file
states	ModuleStates ComplexType	
states-relativeFilePath	string	path(s) to module state file(s) relative from metadata file

3.5 Configuration

3.5.1 Description of Configuration dataset

The Configuration dataset is a zip file holding a Delft-FEWS configuration or complete model setup. The Configuration is identified by a revision identifier. For Delft-FEWS, the revision identifier is the Master Controller revision identifier. For non-FEWS configurations, any other unique identifier could be used, e.g. an SVN-revision number.

The dataset is as accompanied by a metadata.xml

3.5.2 Configuration metadata

The metadata file for this data set follows the schema definition as defined in:

http://fews.wldelft.nl/schemas//version1.0/archive-schemas/configMetaData.xsd.

The dataset may hold one (or more) file, described by its metadata (see Figure 3.6 and Table 3.4). The metadata file is not complete yet as the URL is missing and multiple config files should be allowed to accommodate multiple sub-models[1]

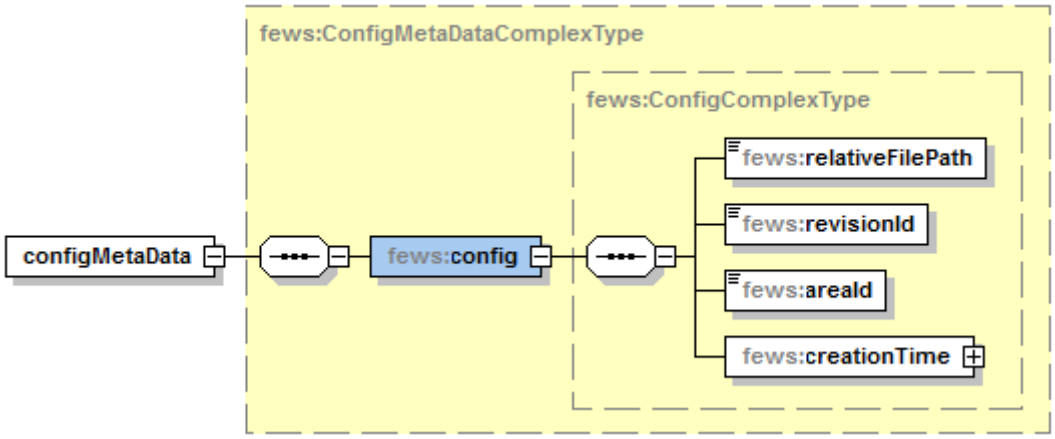


Figure 3.6 Metadata for configurations according to configMetaData.xsd

Table 3.4 Metadata content for configuration

Element Name	Format	Description
relativeFilePath	string	path to configuration zip file relative from metadata file
revisionId	string	Identifier/version number
areaId	string	area reference for which message applies Use no space, will become part of directory path
creationTime	attributes date time	Creation time of dataset as posted to archive

3.6 Rating curves

3.6.1 Description of Rating Curves dataset

The rating curves dataset holds a set of rating curve files accompanied by a metaData.xml file. Any format can be chosen but Deltares software supports the Delft-FEWS PI_ratingcurves.xsd schema definition using a rating table. For details check: http://fews.wldelft.nl/schemas//version1.0/pi-schemas/pi_ratingcurves.xsd.

3.6.2 Rating Curves metadata

The metadata file for this data set follows the schema definition as defined in:

http://fews.wldelft.nl/schemas//version1.0/archive-schemas/ratingCurvesMetaData.xsd.

The dataset may hold multiple rating curve file, described by its metadata (see Figure 3.7 and Table 3.5).

[1] *TODO: See FEWS-9909 and FEWS-9910*

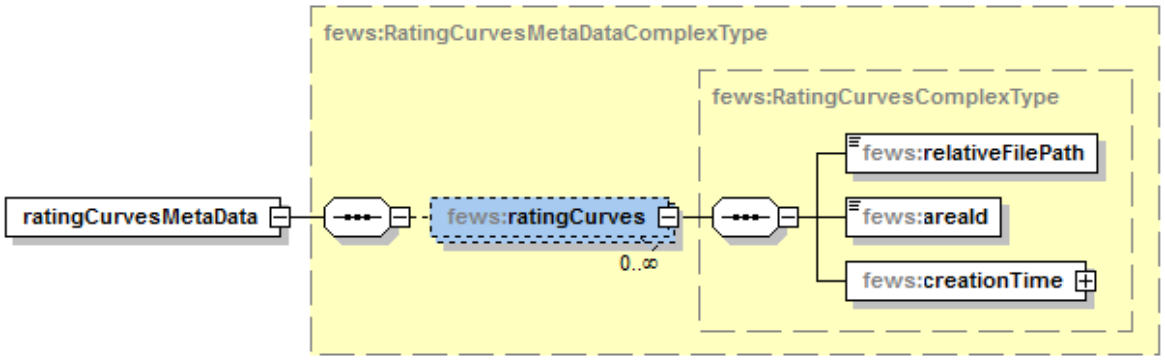


Figure 3.7 Metadata for rating curves according to ratingCurvesMetaData.xsd

Table 3.5 Metadata content for rating curves

Element Name	Format	Description
relativeFilePath	string	file path relative from metadata file
areaId	string	area reference in which rating curves apply Use no space, will become part of directory path
creationTime	attributes date time	Creation time of dataset as posted to archive

3.7 Snap shot

3.7.1 Description of Snap shot dataset

The snap shot dataset holds a complete snap shot of the model database with all its data.

Within Delft-FEWS this is equal to a local datastore as available on a Forecasting shell Server. This dataset ensures that ALL information including its history, as held in the system for a particular run, is archived. Snap shots are particular important when reviews or legal inquiries are of concern to the operator. When snapshots are made at a high frequency, duplication of data is likely to occur.

3.7.2 Snap shot metadata

To be implemented

Figure 3.8 Metadata for snapshots according to snapshotMetaData.xsd

Table 3.6 Metadata content for snapshot

Element Name	Format	Description
relativeFilePath	string	file path relative from metadata file
areaId	string	area reference in which rating curves apply Use no space, will become part of directory path
creationTime	attributes date time	Creation time of dataset as posted to archive

3.8 Events: marking interesting datasets

3.8.1 Description of events

Background

Within an operational context, it seems logical to continuously the archive with the latest data. When nothing interesting has happened, datasets may quickly become irrelevant. To reduce storage needs and costs, data may even be deleted from the archive. However, such cleanup should be prevented for those situations where something interesting happened. These situations are typically also the reasons why people want to search in the archive for data.

The Deltares Open Archive uses events for this purpose. Events are a way to cluster datasets and mark them as being relevant. Events can be used to search the archive and discover relevant data. Additionally, events may help in defining a strategy for data management of the archive.

Events in the Deltares Open Archive context

Extreme situations having a major impact, such as storm events, typically receive a name. Hurricane Sandy is well known to hit New York City, but not everybody remembers that this happened end of 2012. Events thus could well be used to find data which is relevant.

Within the Deltares Open Archive, an event is considered a special kind of metadata which can be characterized by:

- A name
- The area of interest
- The period of interest
- A description

An event may encapsulate multiple datasets of different kinds. E.g. in its full extend, the archive of a storm event would include for the duration of the event: observations, external forecasts, simulations, messages, the model/configuration used and the rating curves used.

To facilitate a training exercise, or an event review based on this event, all datasets are needed. To conduct a forecast performance analysis, all datasets except the messages are needed. For a calibration activity, observations might be sufficient. To verify the calibration through a hindcast, both the observations and the external forecasts would be needed.

In other words, the actual usage determines which datasets are needed. The Deltares Open Archive facilitates this by the introduction of an EventType, where each EventType defines which kinds of datasets remain in the archive.

Within the Delft-FEWS context, a special event type is the Historical event. Historical events refer to historical time series which can be added as an overlay to a graph with the current time series.

Events creation

When the extreme situation is over and everything is back to normal, the moment has arrived to conduct post-event activities such as checking gauges in the field or evaluating how the event was handled. This is also the moment to mark a certain period of time in the archive as an event. Datasets archived for this period will become part of the event.

Marking datasets as being part of an event is generally a manual activity. The Delft-FEWS user interface has a display which supports this activity. However, the event related components of the Deltares Open Archive have been designed such way that other tools may also be used.

Events dataset

Events may hold a specific dataset as well, namely attachments. These attachments can hold the results of post-event analysis, such as reports or new data generated in the analysis.

3.8.2 Event definition

The events follow the schema definition as defined in:

http://fews.wldelft.nl/schemas//version1.0/archive-schemas/event.xsd.

Figure 3.9 and Table 3.7 discuss the event schema. The list of parameters and locations is only relevant to hand over information to Delft-FEWs concerning the locations and parameters that should be made available in a Historic Timeseries Overlay.

Table 3.7 Content of events.xsd

Element Name	Format	Description
id (attribute)	string	Unique events identifier, typically composed of areaId_eventType_creationtime
name (attribute)	string	Event name (optional)
description	string	Description of event
creationDate	attributes date time	Creation date of event as posted to archive
startDate	attributes date time	Start date time of event
endDate	attributes date time	End date time of event
area	attributes id name	Identifier and name used for searching
parameters-parameterId	string	parameters to be included in Delft-FEWS Historic Event overlay
locations-locationId	string	locations to be included in Delft-FEWS Historic Event overlay
eventTypeId	string	identifier of event type
active	boolean	flag indicating status of event, default =TRUE
atachements-attachment	string	filename (recommendation: no spaces)
changed	boolean	indicates if file has changed compared to repository when provided to archive import service

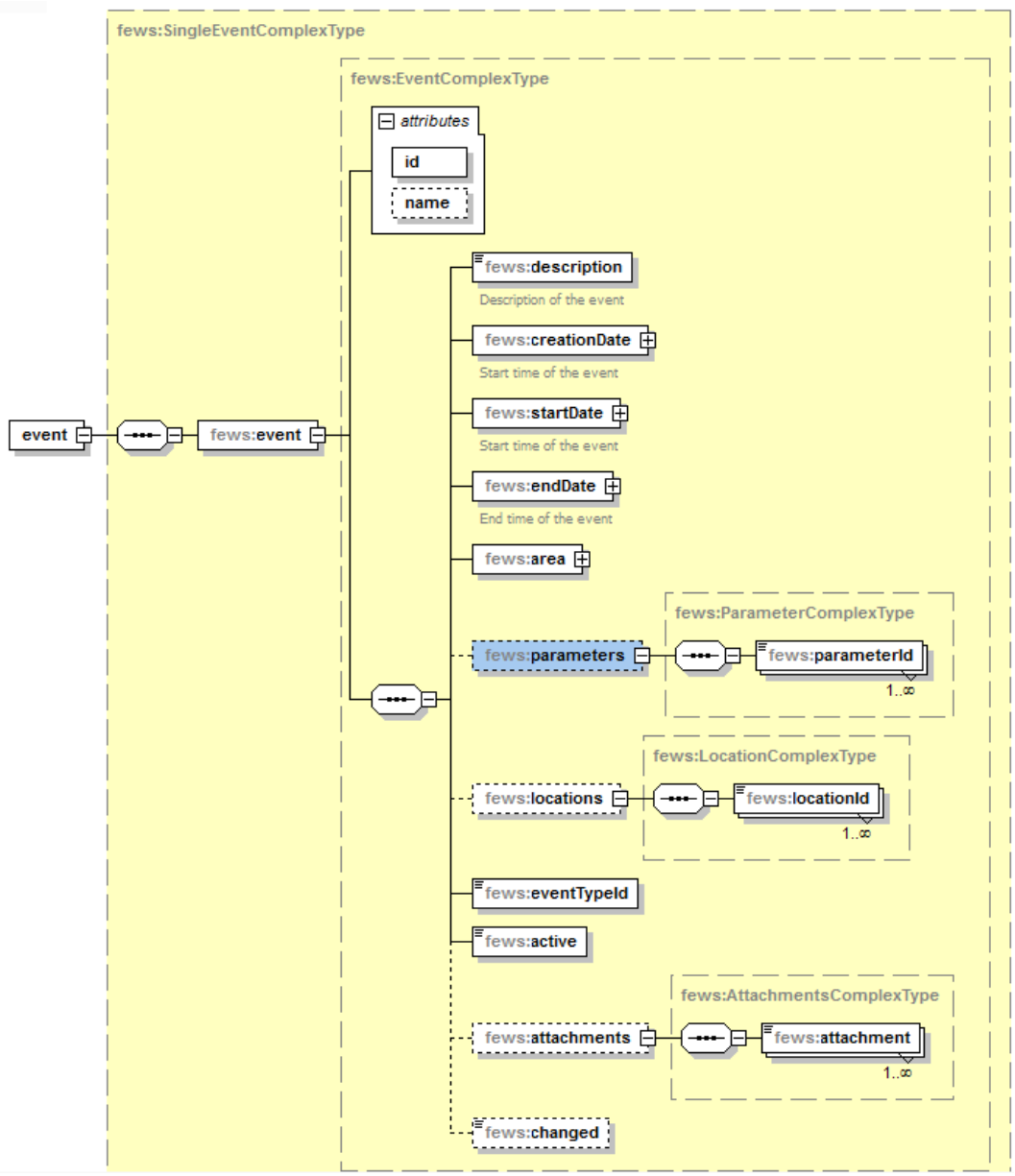


Figure 3.9 Events definition according to events.xsd

The eventTypes follows the schema definition as defined in:

http://fews.wldelft.nl/schemas//version1.0/archive-schemas/eventTypes.xsd.

Figure 3.10 and Table 3.8 Illustrate the content.

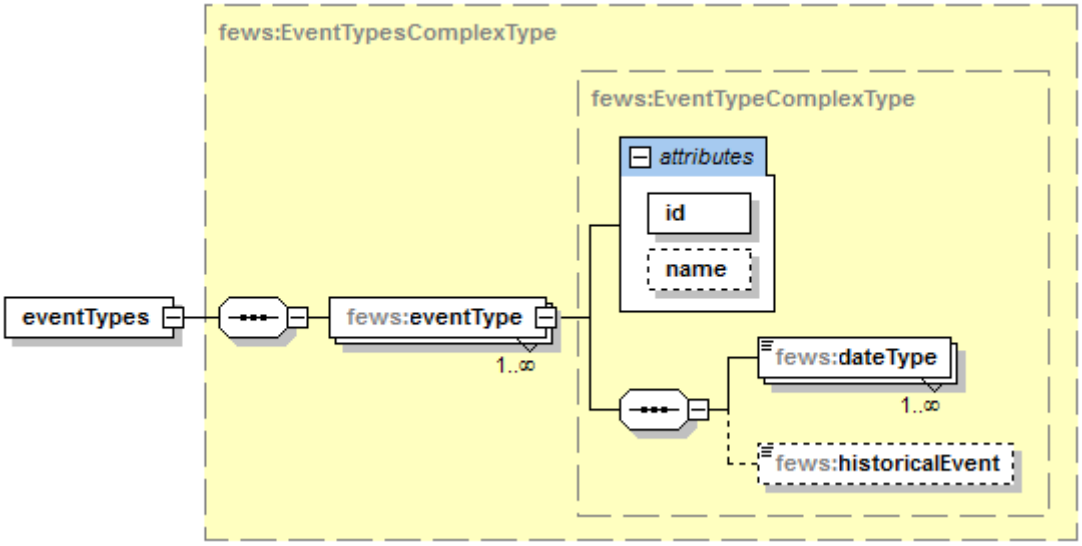


Figure 3.10 EventTypes definition according to eventTypes.xsd

Table 3.8 Content of eventTypes.xsd

Element Name	Format	Description
id	string	Unique events identifier
name	string	Event name
dataType	string (Enumeration): external forecast timeseries forecaster notes model states modifiers observed time series rating curves reports simulated timeseries	data types included in this event
historicalEvent	bool	Flag indicates if event should be handled as Delft-FEWS Historic event

4 Archive Processes

4.1 Data production with Delft-FEWS

4.1.1 The Delft-FEWS ExportArchiveModule

Within Delft-FEWS, the datasets are exported to the archive in a workflow using the ExportArchiveModule. This workflow needs to be scheduled on a regular interval to archive all relevant data. To prevent dependencies on other processes, the ExportArchiveModule is envisioned to write directly into the archive file storage. The FSS thus needs to be able to have write access to those disks.

For each kind of dataset, the ExportArchiveModule checks the database for changes over a (configured) relative period. It exports any data which meets the export instructions and has changed within this period. Datasets are archived in a pre-defined directory structure, which is based on areaId, date and dataset.

The schema of the associated configuration file (Figure 4.1) is defined at:
http://fews.wldelft.nl/schemas/version1.0/exportArchiveModule.xsd.

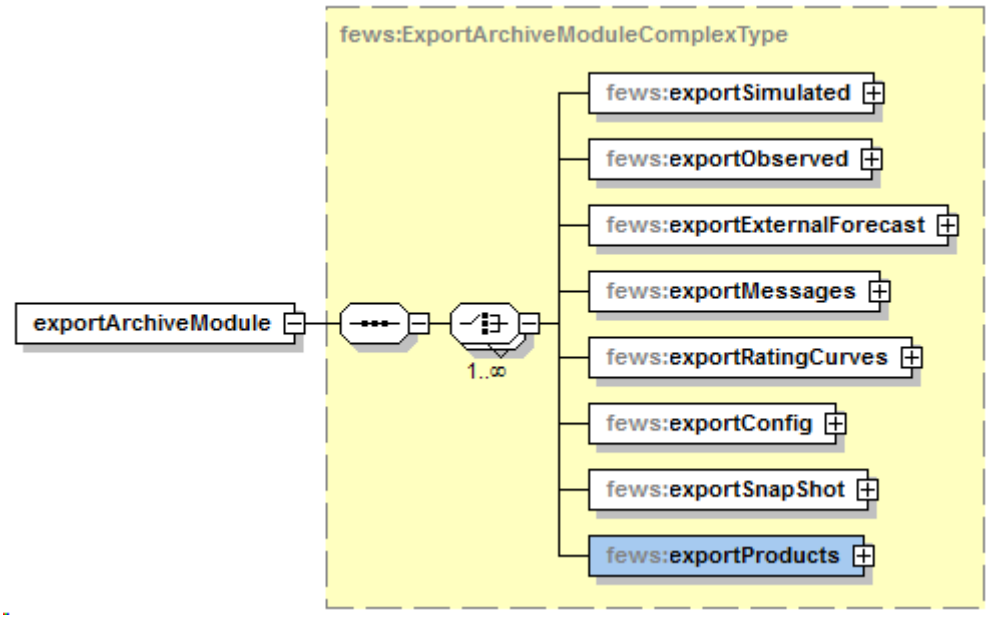


Figure 4.1 Top level of Delft-FEWS exportArchiveModule.xsd

4.1.2 Observations archiving by Delft-FEWS

For observations a dataset is generated for every area on a daily basis. The associated directory structure of the Delft-FEWS export for this type of dataset is as follows:

```
<archive root>/<yyyy>/<MM>/<areaId>/<dd>/observed/
```

When Delft-FEWS generates the netcdf file, data is written to the same data block when the entire matrix is filled, i.e. all time steps are regular and none of the values is missing. For those locations with irregular time stamps, or missing values, a separate data block is used.

Within the netCDF file, each data block is accompanied by a header. Within each header a metadata item called 'timeseries_sets_xml' is included, holding the exact definition of the timeSeriesSet as the data was stored in the FEWS database. This feature allows full reproducibility of the time series via an Import workflow in a Delft-FEWS stand alone application, assuming that the associated Delft-FEWS configuration is in place.

The exportArchiveModule.xsd has a dedicated exportObserved section to configure the observed timeseries that need to be archived (see Figure 4.2). Table 4.1 documents the associated elements:

Table 4.1 Deft-FEWS export configuration for archiving observations

Element	**Format**	**Description**
general ComplexType		
archiveFolder	string	Export destination folder, assumes that the account running the FEWS (FSS) application has write access
relativePeriod		Exports entire dataset by day, for any day where a database change (blob creation time) is detected within the relativePeriod (relative to T0). Existing timeseries files are overwritten*
idMap	string	idMap applied to translate internal FEWS identifiers to identifiers that meet NetCDF-CF criteria.E.g. netcdf does not allow a full stop ('.') in the variable name
netcdfObservedExport Activities ComplexType		
fileName	string	without nc extension, preferably no spaces
areaId	string	area to which the dataset belongs
ncMetaData	string elem.	optional metadata tags within NetCDF file following CF convention. Supported by the internal catalogue of the THREDDS Data Server
includeFlags	bool	default=TRUE; if TRUE, a list of flags is stored, each value pointing to the associated flag
includeComments	bool	default=TRUE; if TRUE, a list of comments is stored, each value pointing to the associated comment
thresholdGroupId	string	identifies FEWS ThresholdGroup which is used to detect threshold crossings to be highlighted in the metaData.xml
timeSeriesSets		FEWS timeseries sets

* When an existing file is locked while it needs to be overwritten, the export function writes a new temporary file. The FileSweeper, a scheduled process, renames this file when the lock is removed from the original file.

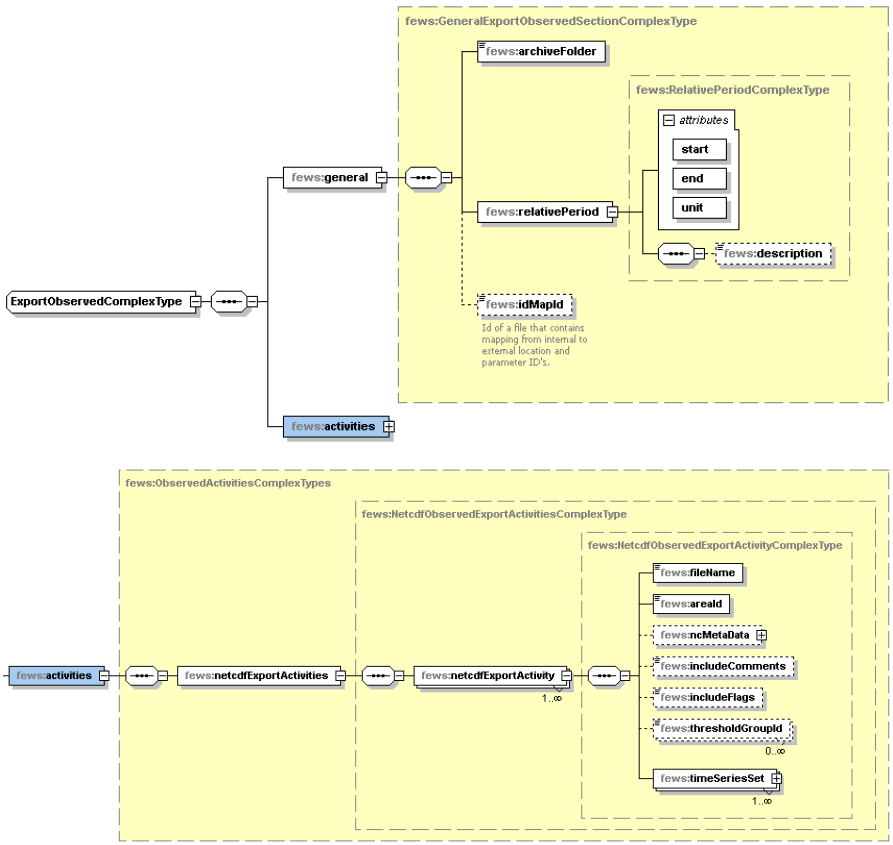


Figure 4.2 Deft-FEWS export configuration for archiving observations

4.1.3 Messages archiving by Delft-FEWS

Delft-FEWS can export messages to the archive via the ArchiveExportModule (exportMessages activity).

For messages a dataset is generated for every area on a daily basis. The associated directory structure of the Delft-FEWS export for this type of dataset is as follows[2]:

`<archiveRoot>/<yyyy>/<MM>/<areaId>/<dd>/messages/`

The exportArchiveModule.xsd has a dedicated exportMessages section to configure the messages that need to be archived (see Figure 4.3). Currently only ForecasterNotes can be archived. Table 4.2 documents the associated elements:

Table 4.2 Delft-FEWS export configuration for archiving messages

Element	Format	Description
general ComplexType		

[2] TO DO: FEWS-10198: update directory structure to above setup

archiveFolder	string	Export destination folder, assumes that the account running the FEWS (FSS) application has write access
relativePeriod		Exports entire dataset by day, for any day where a database change (blob creation time) is detected within the relativePeriod (relative to T0). Existing files are overwritten*
MessagesActivities ComplexType		
forecasterNotesExport Activity ComplexType*		
areaId	string	area for which messages need to be archived

* When an existing file is locked while it needs to be overwritten, the export function writes a new temporary file. The FileSweeper, a scheduled process, renames this file when the lock is removed from the original file.

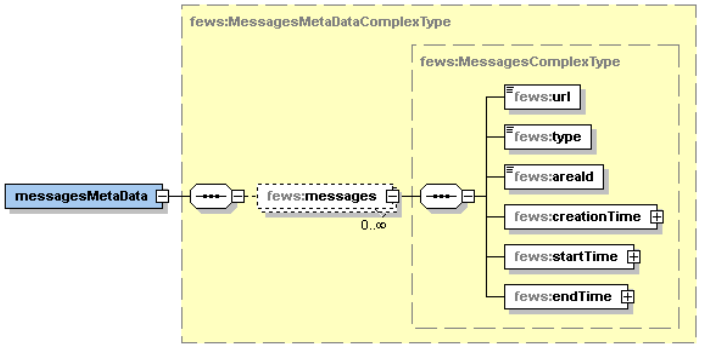


Figure 4.3 Delft-FEWS export configuration for archiving messages

4.1.4 External forecast archiving by Delft-FEWS

Delft-FEWS can export external forecast time series to the archive via the ArchiveExportModule (exportExternalForecast activity).

For external forecasts a dataset is generated for every area for every source for every forecast. The associated directory structure of the Delft-FEWS export for this type of dataset is as follows:

```
<archiveRoot>/<yyyy>/<MM>/<areaId>/<dd>
/external_forecasts/<sourceId>_<ExtForecastTime>
```

The exportArchiveModule.xsd has a dedicated exportExternalForecast section to configure the time series that need to be archived (see Figure 4.4). Table 4.3 documents the associated elements:

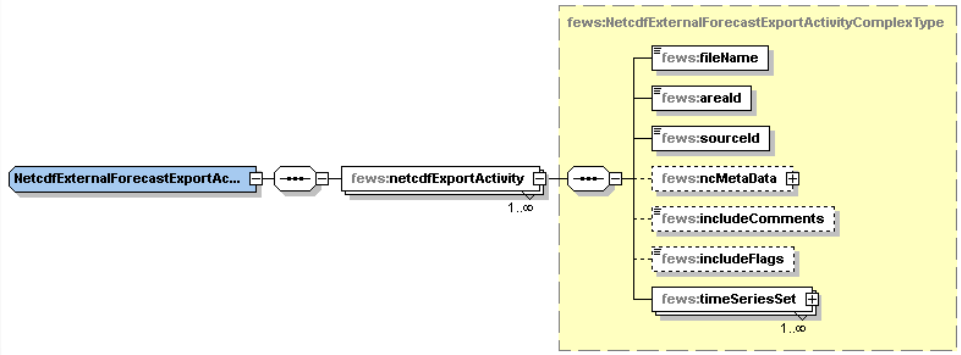


Figure 4.4 Delft-FEWS export configuration for archiving external forecasts

Table 4.3 Delft-FEWS export configuration for archiving external forecasts

Element	Format	Description
GeneralExportForecast Section ComplexType		
archiveFolder	string	Export destination folder, assumes that the account running the FEWS (FSS) application has write access
relativePeriod		Exports entire external forecast dataset by source and forecast time, for any forecast where a database change (blob creation time) is detected within the relativePeriod (relative to T0). Existing timeseries files are overwritten* If no relativePeriod is specified the external forecast with the latest forecast time is exported
idMap	string	idMap applied to translate internal FEWS identifiers to identifiers that meet NetCDF-CF criteria.E.g. netcdf does not allow a full stop ('.') in the variable name
ExternalForecastActivities ComplexType		
NetcdfExternalForecast ExportActivities ComplexType		
NetcdfExternalForecast ExportActivity ComplexType*		
fileName	string	without nc extension, preferably no spaces
areaId	string	area to which the dataset belongs
sourceId		identifies underlying source e.g. NWP product
ncMetaData	string elem.	optional metadata tags within NetCDF file following CF convention. Supported by the internal catalogue of the THREDDS Data

		Server
includeFlags	bool	only applied for scalar values. default=TRUE; if TRUE, a list of flags is stored, with each value pointing to the associated flag
includeComments	bool	only applied for scalar values. default=TRUE; if TRUE, a list of comments is stored, with each value pointing to the associated comment
timeSeriesSets		FEWS timeseries sets

* When an existing file is locked while it needs to be overwritten, the export function writes a new temporary file. The FileSweeper, a scheduled process, renames this file when the lock is removed from the original file.

4.1.5 Simulations archiving by Delft-FEWS

Delft-FEWS can export simulations to the archive via the ArchiveExportModule (exportSimulations activity). Typically this archiving activity is included in the workflow that computes the final approved forecast.

The associated root directory structure of the Delft-FEWS export for this type of dataset is as follows:

```
<archiveRoot>/<yyyy>/<MM>/<areaId>/<dd>/simulated/<workflowId>_<TimeZero>_<DispatchTime>
```

Where `<dd>` refers to the date of the forecast time T0.

This directory holds the metaData.xml as well as runInfo.xml file with the FEWS taskrun properties (see Figure 3.3). Within this directory the following sub-folders may exist:

```
/timeseries
/reports
/modifiers
/states
```

The exportArchiveModule.xsd has a dedicated exportSimulated section to configure the messages that need to be archived (see Figure 4.5). The associated specification is given in Table 4.4.

Table 4.4 Delft-FEWS export configuration for archiving simulations

Element	**Format**	**Description**
GeneralExportForecast Section ComplexType		
archiveFolder	string	Export destination folder, assumes that the account running the FEWS (FSS) application has write access
relativePeriod		Exports entire simulated timeseries by workflow, for any simulated forecast where database change (blob creation time) are detected within the relativePeriod (relative to T0). Existing

		timeseries files are overwritten* If no relativePeriod is specified the Current simulated forecast is exported
idMap	string	idMap applied to translate internal FEWS identifiers to identifiers that meet NetCDF-CF criteria.E.g. netcdf does not allow a full stop ('.') in the variable name
ForecastActivities ComplexType		
NetcdfForecastExport Activities ComplexType		
NetcdfForecastExport Activity ComplexType*		
fileName	string	without nc extension, preferably no spaces
areaId	string	area to which the dataset belongs
ncMetaData	string elem.	optional metadata tags within NetCDF file following CF convention. Supported by the internal catalogue of the THREDDS Data Server
includeFlags	bool	only applied for scalar values. default=TRUE; if TRUE, a list of flags is stored, with each value pointing to the associated flag
includeComments	bool	only applied for scalar values. default=TRUE; if TRUE, a list of comments is stored, with each value pointing to the associated comment
timeSeriesSets		FEWS timeseries sets
ReportsExportActivity ComplexType*		
subfolder	string	Subdirectory within the 'reports' directory
moduleInstanceId	string	moduleInstanceId which created the report
ModuleStatesExport Activity ComplexType*		
moduleInstanceId	string	moduleInstanceId which created the state

* When an existing file is locked while it needs to be overwritten, the export function writes a new temporary file. The FileSweeper, a scheduled process, renames this file when the lock is removed from the original file.

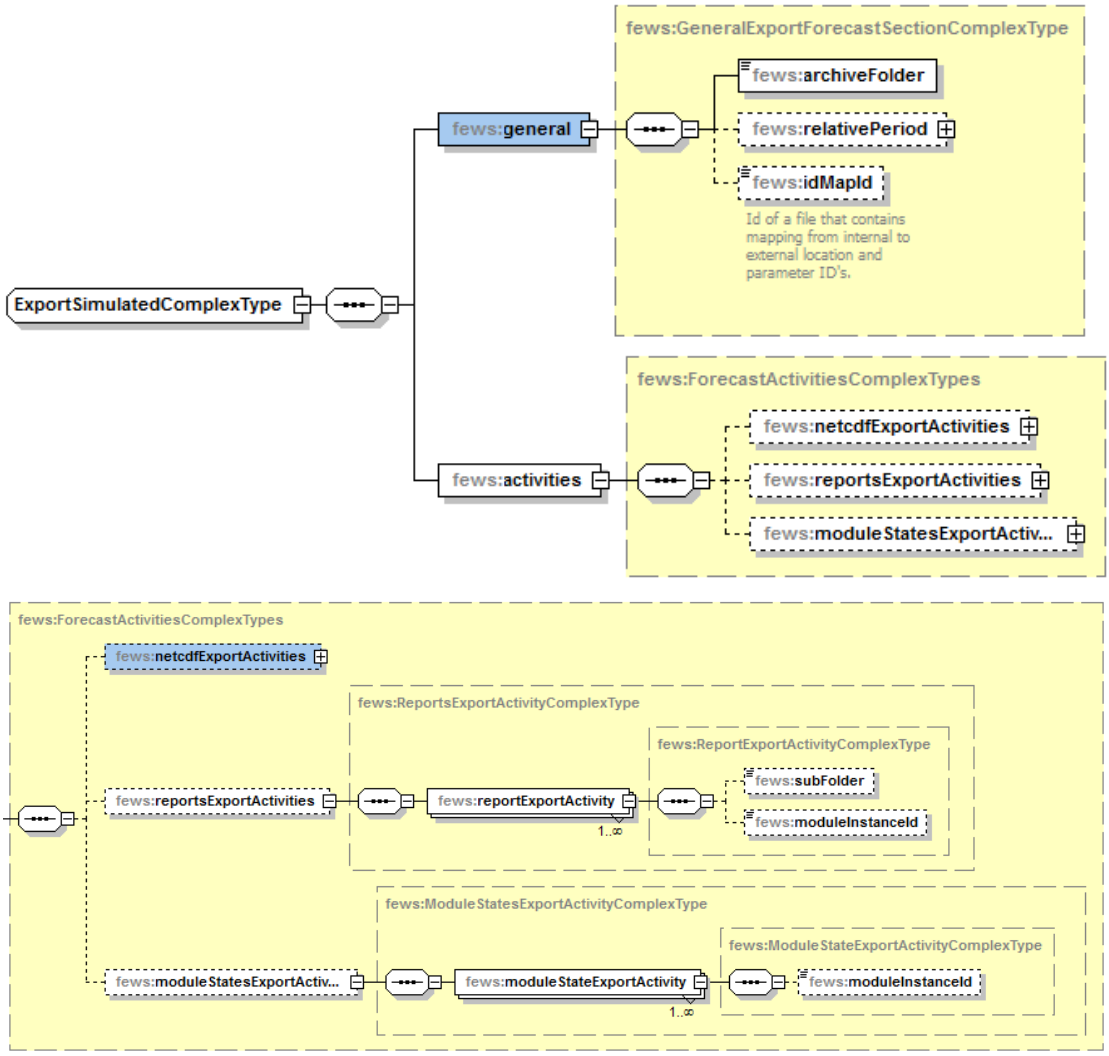


Figure 4.5 Delft-FEWS export configuration for archiving simulations

4.1.6 Delft-FEWS export of configuration

Delft-FEWS can export the current configuration to the archive via the ArchiveExportModule (exportConfig activity). The configuration thus is exported as part of the workflow. The associated root directory structure of the Delft-FEWS export for this type of dataset is as follows:

`<archiveRoot>/<config>//<areaId>/<yyyymmdd>/`

The date refers to the revision date of the configuration. The file name typically holds the revisionId.

The exportArchiveModule.xsd has a dedicated exportConfig section to setup the export of the Configuration (see Figure 4.6). This is due to be revised as the same relativePeriod based export mechanism should be adopted for checking what to export (see Table 4.5).

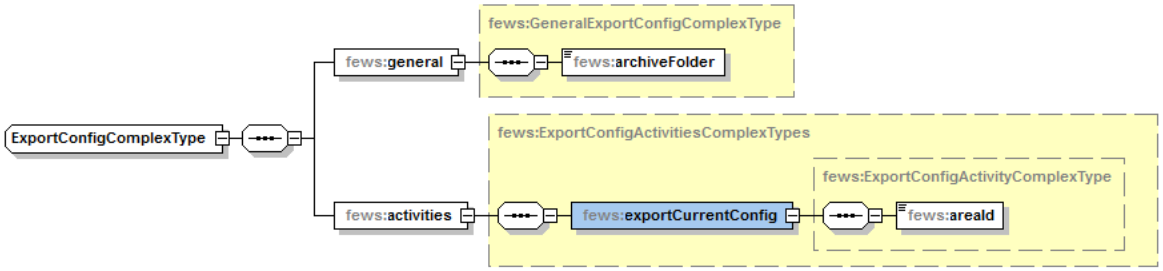


Figure 4.6 Delft-FEWS export configuration for archiving configurations

Table 4.5 Delft-FEWS export configuration for archiving a configuration

Element	Format	Description
GeneralExportConfig ComplexType		
archiveFolder	string	Export destination folder, assumes that the account running the FEWS (FSS) application has write access
relativePeriod (TO DO)		Exports entire configuration when database change (config revision) has been detected within the relativePeriod (relative to T0). Existing files are overwritten* If no relativePeriod is specified the Current configuration is exported
ExportConfigActivities ComplexType		
ExportConfigActivity ComplexType*		
areaId	string	area to which the dataset belongs

4.1.7 Delft-FEWS export of rating curves

Delft-FEWS can export rating curves to the archive via the ArchiveExportModule (exportRatingCurves activity). The entire history of the rating curves is exported. The associated root directory structure of the Delft-FEWS export for this type of dataset is as follows[3]:

`<archiveRoot>/ratingcurves/<areaId>`

The configurator can choose between exporting the full set or just the rating curves that have changed, with or without modifier changes.

The exportArchiveModule.xsd has a dedicated exportRatingCurve section to setup the export of the Configuration. Table 4.6 and Figure 4.7 discuss the general section.

[3] *FEWS-10198: update directory structure to above setup*

Table 4.6 Delft-FEWS export configuration for archiving rating curves (general section)

Element	Format	Description
GeneralExportRating Curves ComplexType		
archiveFolder	string	Export destination folder, assumes that the account running the FEWS (FSS) application has write access
relativePeriod		If defined, only rating curves which have database changes during the relativePeriod (relative to T0) are exported. If no relativePeriod is specified all available rating curves are exported
Prefix-timeZeroFormattingString	string	Adds T0 stamp as prefix to file name E.g.yyyymmdd_hhmm
idMap	string	idMap applied to translate internal FEWS identifiers to external identifiers
unitConversionsId	string	Allows conversion of rating table units
includeModifiers	bool	Default=FALSE; If TRUE, rating curve modifiers should be included in the export

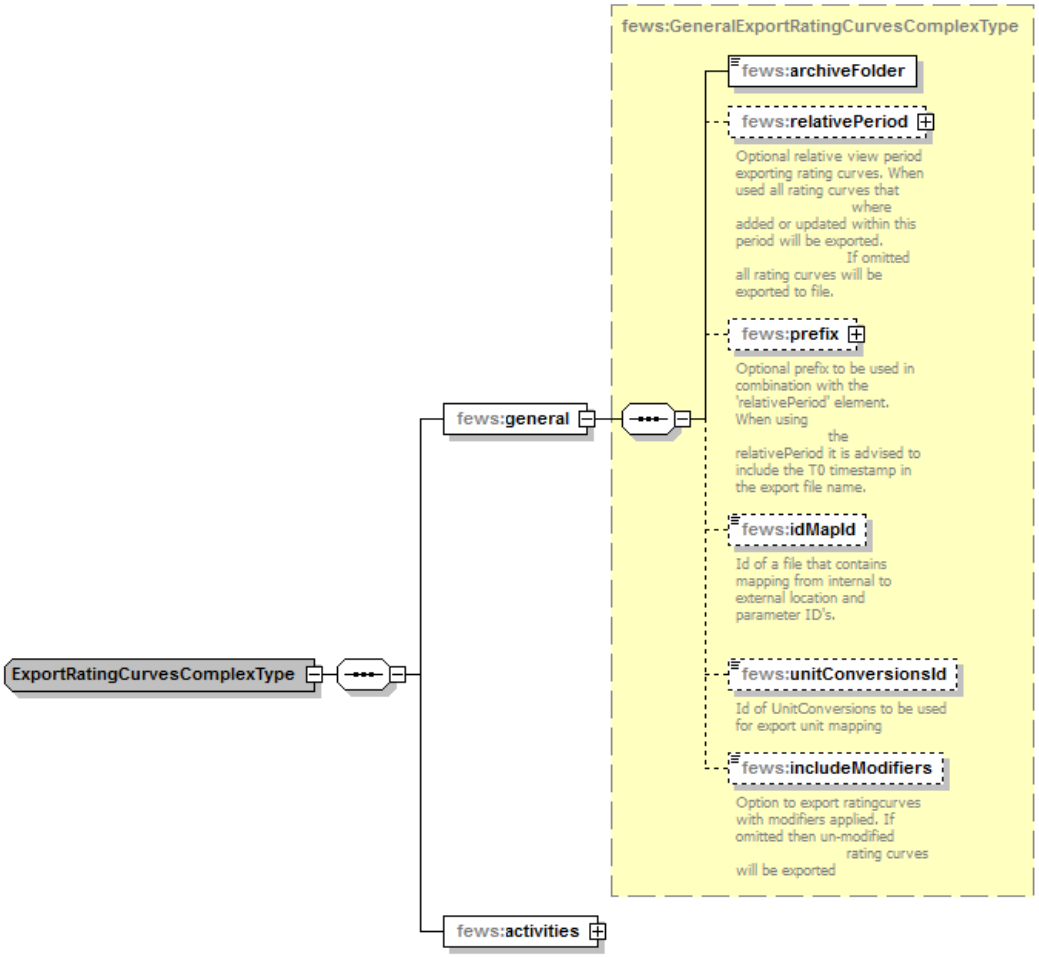


Figure 4.7 Delft-FEWS export configuration for archiving rating curves, general section

Table 4.7and Figure 4.8 discuss the activities section of for exporting rating curves.

Table 4.7 Delft-FEWS export configuration for archiving rating curves (activities section)

Element	Format	Description
RatingCurvesActivities ComplexType		
RatingCurvesExport Activity ComplexType		
fileName	string	without nc extension, preferably no spaces
areaId	string	area to which the dataset belongs
linearTableStageresolution	string	Stage increment between the rows in a rating table
locationId/locationSetId	string	Locations for which rating curves should be exported to this file

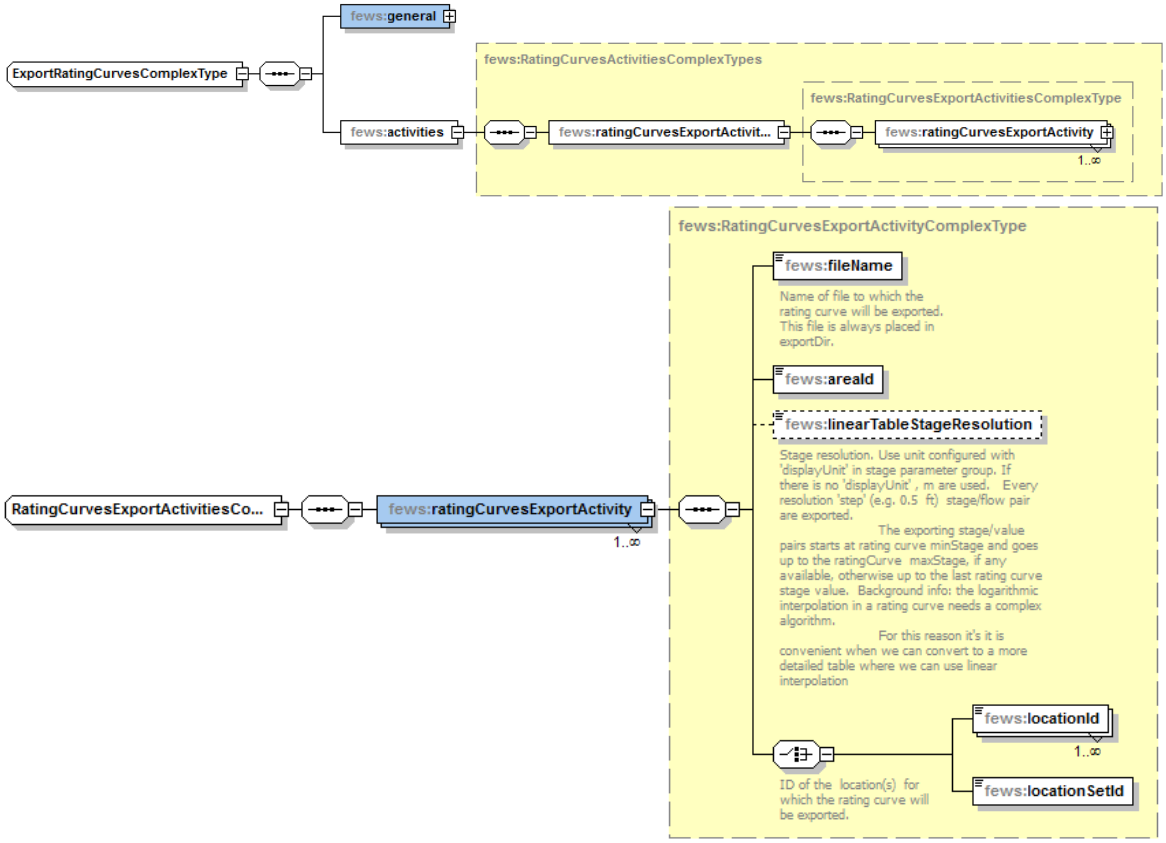


Figure 4.8 Delft-FEWS export configuration for archiving rating curves, activities section

4.1.8 Archiving Delft-FEWS snap shots

Snap shots of the Delft-FEWS database can be archived via the archiveExportModule (exportSnapShot activity).

For snapshots a dataset is generated and stored under the configured area at the memoment of workflow execution. The associated directory structure of the Delft-FEWS export for this type of dataset is as follows:

`<archiveRoot>/<yyyy>/<MM>/<areaId/<snapshot>/`

Figure 4.9 and Table 4.8 documents the associated elements in the exportSnapShot activity:

Table 4.8 Delft-FEWS export configuration for archiving database snap shots

Element	Format	Description
general ComplexType		
archiveFolder	string	Export destination folder, assumes that the account running the FEWS (FSS) application has write access
ExportSnapShotActivity ComplexType		
areaId	string	area (folder) where snapshot is archived

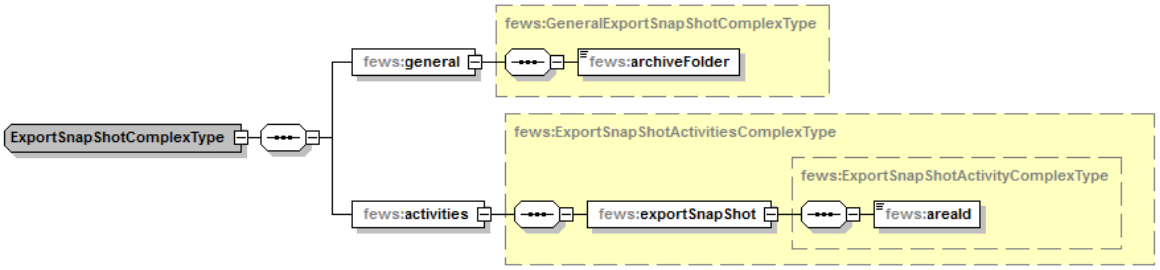


Figure 4.9 Delft-FEWS export configuration for archiving database snap shots

4.2 Archive Web-services

4.2.1 Catalogue server (GeoNetwork)

The Catalogue server will host the metadata of the datasets. This server can be queried using keywords contained in the metadata. Each metadata item holds tags for the areaId and the period. In future, an additional key-value pair constructs will be added to allow additional freedom for tagging data by allowing your own keys (e.g. winddirection).

The catalogue itself will be based on the jave based open source software package GeoNetworks, a package which provides catalogue services using the OGC-Catalogue Service (CSW) standard. The Deltares Open Archive will deploy GeoNetwork on a Tomcat webserver.

Since CSW requires geospatial query capabilities, a database cache will be needed which can accommodate geospatial indices. The database will normally be PostGress database, with PostGIS for spatial indexing. Other RDBMS (e.g. Oracle, Microsoft SQL Server could be utilized if required by the IT-department, but license cost may make testing and applying the spatial component expensive.

The GeoNetwork catalogue requires its meta-data to be in one or more predefined formats. The format chosen for cataloguing datasets is the ISO19139 meta-data template. All required search criteria are included in an XML document that conforms to this schema standard. Besides the required information it would be helpful to also add additional information describing in more detail the contents and source of the datasets.

How are events connected to datasets

The metadata of the datasets do not hold references to events. Instead, the area and period defined in the event are used to identify datasets that meet those criteria.

Events are not included in the GeoNetwork catalogue. Instead the archive server holds its own (file based) catalogue for events. A webserver hosts these files for use by other applications.

4.2.2 Data server (THREDDS)

Datasets will be hosted via the THREDDS Data Server. Client software can use the URLs as held in the metadata to request the actual data from the THREDDS Data Server.

So-called scientific data, i.e. the netCDF files, can be accessed via protocols such as OPeNDAP, OGC-WMS and OGC-WCS. The THREDDS data server is also used to host 'non-scientific data' (e.g. all other file formats) via standard http-protocols.

URL: http://threddstomcatserver:8080/thredds/fileServer/Archive/

4.2.3 Archive web-server

Event files are hosted on the data file server. An archive process manages the event files. It can add, update, merge and delete events with the existing repository of event files.

This archive process is running as a web service: the archive web service, deployed in the same Tomcat server as THREDDS. THREDDS is used to download events, while the archive web service offers HTTP Put methods to upload of event files and event attachments.

The creation and update of events is a manual process conducted outside the archive. The Delft-FEWS application provides a GUI for this purpose (see Figure 4.10). Since the archive web server holds the logic and data to manage events, other tools including web-clients could be implemented to create and update events.

The update process basically comes down to the following steps:

1. Request a download from the server to copy existing event files from the server repository to a local work directory
2. Add and update events in the local work directory. Removal of events is done by setting the active flag to inactive.
3. Upload the updated event files to the archive server. The server merges the changes with its repository.

URL: http://threddstomcatserver:8080/thredds/fileServer/Archive/

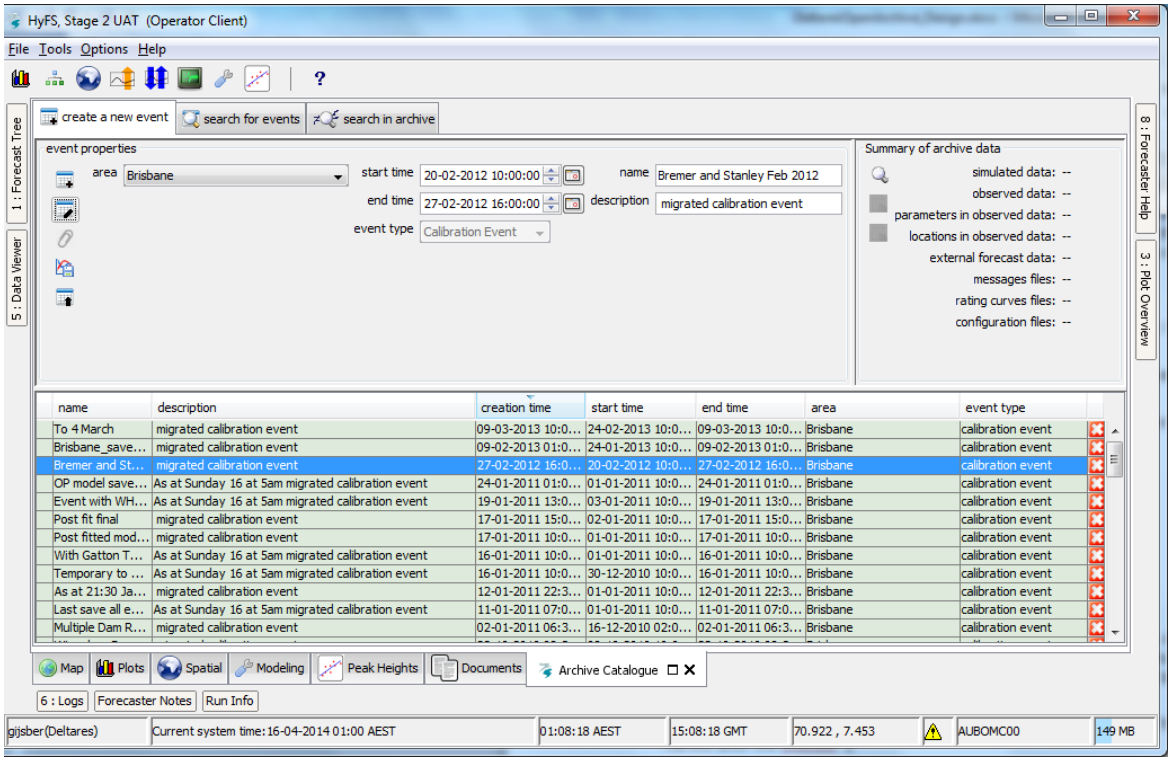


Figure 4.10 Delft-FEWS GUI to create and update events

Adding attachments to events

Events provide a special case as post-event analysis may result in reports or data which needs to be tagged to the event. The Deltares Open Archive facilitates this need by allowing events to hold attachments. Attachments can have any file format but the PDF format is recommended for reports and the zip format for other files.

Attachments are archived in sub-directories of the events directory. The sub-directories are named after the eventId.

URL: http://threddstomcatserver:8080/thredds/fileServer/Archive/

Note:

The ArchiveWebServer is foreseen to grow in capabilities as it will also become the host for the Admin webconsole.

4.3 Archive Server processes:

4.3.1 FileSweeper

When a process is overwriting an existing file in the archive, another process might have locked the file for reading. In that case, the file writer saves the file under a temporary name. The task of the FileSweeper is to scan the archive for temporary files and rename them to the original filename, hereby overwriting the original file.

The procedure is as follows (implemented for time series files):

1 The data producer (e.g.Forecasting Shell) writes the files as *.tmp, rename the file to *.new and tries to rename this file to *.nc
2 The FileSweeper scans all directories for files named *.new and tries to rename them to *.nc

The FileSweeper is run on a schedule via the archive admin console.

4.3.2 HistoricEventsExporter

This process identifies which events are FEWS Historic events. The process moves Observation timeseries in the event period from the archive to the HistoricalEvents import directory of the FSS.

To do so, the process scans the Events directory for any files updated in the last X days. If these events reflect a historic event, both the event file as well as the associated (observed) time series file is posted to the FSS import directory.

The historicEventsExporter is run on a schedule via the archive admin console.

4.3.3 Harvester

This process scans the data directory and identifies new or updated metadata files based on file time stamps. Since the native metadata format of the Deltares Open Archive, as presented in chapter 3 does not meet the ISO19139 standard, the harvester transfers the native metadata.xml file into the ISO19139 meta-data files for insertion into catalogue using the GeoNetwork webservices :An example of such xml file is given in Figure 4.11.

Figure 4.11 Metadata record (ISO 19139 standard) as stored in catalogue database

The Deltares metadata files are stored in the Archive file storage in the directory holding the dataset. When the harvester finds a new dataset, it reads the metadata.xml and offers the metadata to the GeoNetwork webservice. GeoNetwork adds a new metadata record to the catalogue and hands back a unique recordId to the harvester. The harvester places this recordId in a file next to the metadata.xml file (see Figure 4.12).

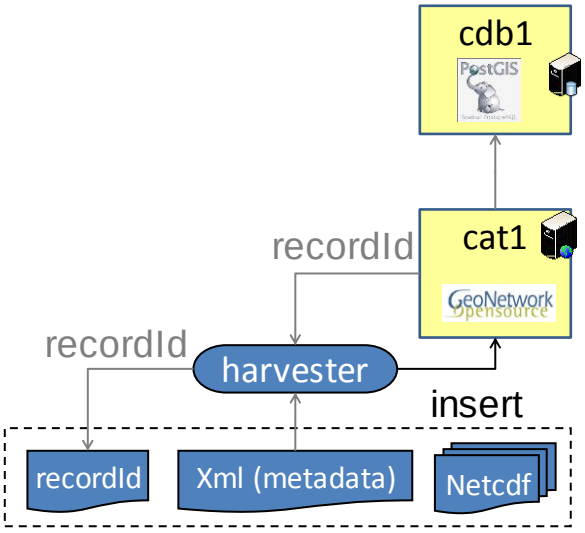


Figure 4.12 Metadata harvesting process

This recordId allows the harvester to update any metadata associated to the dataset. This can include changes of directory locations as long as file dates have changed.

By storing the identifier together with the data it will always be possible to find and alter the corresponding meta-data entry in GeoNetwork or add additional data to the dataset.

The harvester will provide to GeoNetwork:

- URL to the Deltares metadata file
- Relative paths to the files or datatypes in the dataset
- The metadata tags as held in the Deltares metadata file

If the harvester detects a directory with a recordId but without a metadata file, it will remove the record from the catalogue and remove the recordId file from the file system.

The Harvester is run on a schedule via the archive admin console. When the harvester experiences insertion issues due to an overload of the catalogue webservice, it goes into sleep mode for 30 minutes.

4.3.4 Archive Data Management Tool

As part of each dataset, the metadata.xml file holds a creation date/time stamp. In combination for a lifetime that can be associated to each data set and event type, an overview can be generated what data needs to be moved.

The archive data management tool generates this overview in a comma separated report by analyzing the archive file system and using a set of life time rules. The System Administrator can use this report to move the data around.

The tool uses two configuration files.

- The eventTypes.xml file as discussed in section 3.8.2 and Figure 3.10
- The dataManagement.xml configuration file which defines the life times of data types and overruling lifetimes by event type (see Figure 4.13).

The configuration file defines a default lifetime for any piece of data, which can be overruled by data set. EventRules define the lifetime of data associated with a particular event type. If a dataset is part of such event, the lifetime of the event overrules the lifetime of the dataset.

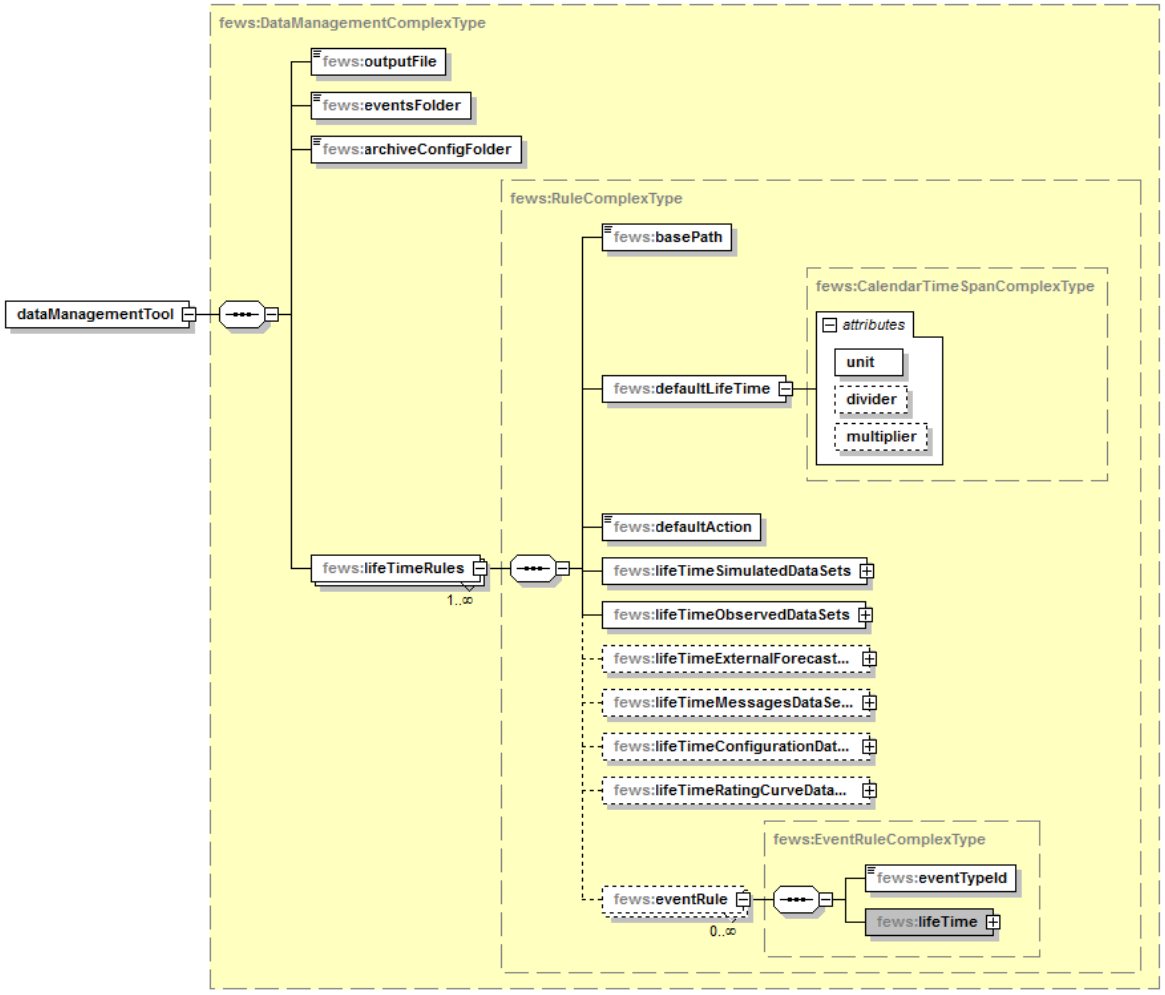


Figure 4.13 Configuration of lifetimes by data type and event type (dataManagementTool.xsd)

Table 4.9 dataManagementTool configuration

Element	Format	Description
outputFile	string	Full path including file name for the comma separated report file generated by the data management tool
eventsFolder	string	Full path referencing the root folder where events are stored
basePath	string	Full path to the root of the data archive
defaultLifeTime	CalendarTimeSpan complexType	default life time for any dataset
CalendarTimeSpan ComplexType	unit (enum), multipler, divider	enumeration: minute, day, week, month, year
defaultAction	string	default string as put in the report when life time is passed
lifeTimeSimulatedDataSets	CalendarTimeSpan complexType	lifetime for datasets of type simulated, overrules default life time
lifeTimeObservedDataSets	CalendarTimeSpan complexType	lifetime for datasets of type observed, overrules default life time
lifeTimeExternalForecastDataSets	CalendarTimeSpan complexType	lifetime for datasets of type external forecasts, overrules default life time
lifeTimeMessagesDataSets	CalendarTimeSpan complexType	lifetime for datasets of type messages, overrules default life time
lifeTimeConfgurationDataSets	CalendarTimeSpan complexType	lifetime for datasets of type configuration, overrules default life time
lifeTimeRatingCurvesDataSets	CalendarTimeSpan complexType	lifetime for datasets of type rating curves, overrules default life time
lifeTimeSnapShotDataSets	CalendarTimeSpan complexType	lifetime for datasets of type snap shot, overrules default life time
eventRule	eventRule ComplexType	defines lifetime for events and associated datasets. Defined be event type. Overrules life time of dataset
eventTypeId	string	identifier of event type
lifeTime	CalendarTimeSpan complexType	lifetime for events and associated datasets of this event type, overrules dataset life time

The datamanagement tool is executed manually from the command line. E.g.

```
/opt/fews/archive/managementtools/datamanagementtool.sh
/opt/fews/archive/managementtools/archiveDataManagement.xml
/opt/fews/archive/managementtools/bin >/dev/null
```

4.4 Data usage processes

4.4.1 Discovery

Data can be discovered by area, period and data type:

- Via the GeoNetwork catalogue
- by browsing the THREDDS data server
- via the FEWS GUI (see Figure 4.14)

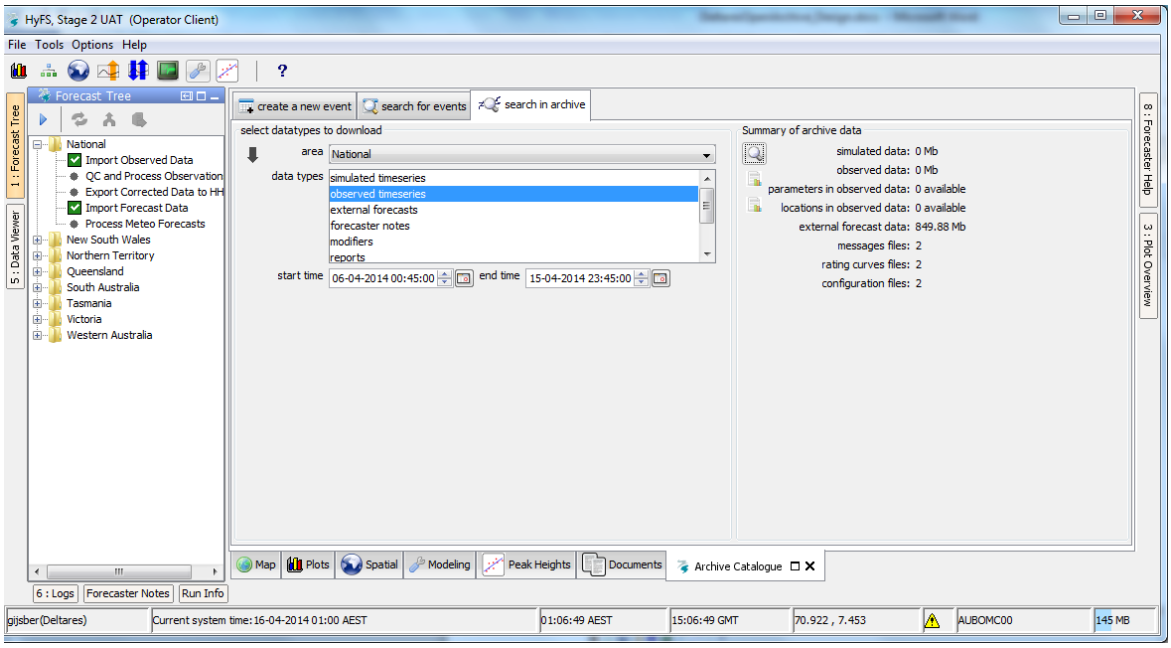


Figure 4.14 Delft FEWS GUI to identify and download available data by data type, area and period.

Currently, the FEWS GUI does not offer the ability to search for datastore snapshots. These need to be retrieved by browsing.

The URL to connect to GeoNetwork is (assuming port 8080 for Tomcat):

http://geonetworktomcatserver:8080/geonetwork10.2/srv/eng/csw/main.home

Data can be discovered by event (area, name and /or period) via the FEWS GUI (see Figure 4.15).

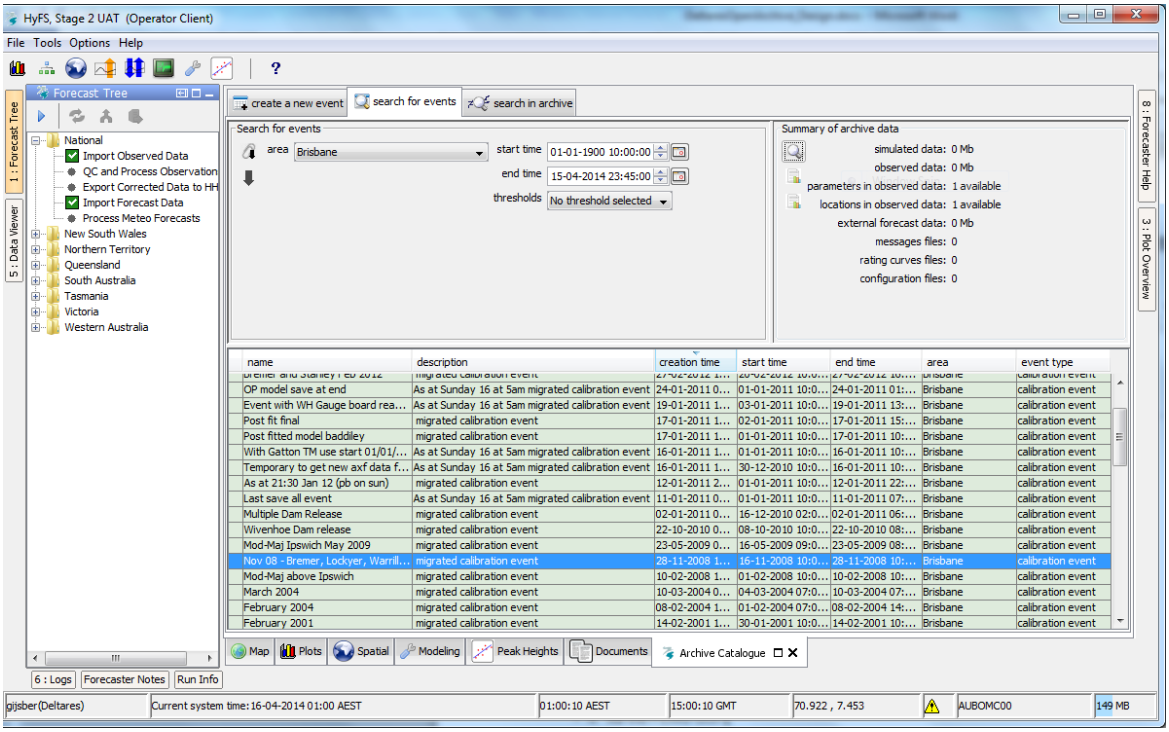


Figure 4.15 Delft FEWS GUI to identify and download available data by event

4.4.2 Retrieval

Datasets can be retrieved from the THREDDS data server. This is a file download only unless the scientific protocols for OPeNDAP or WMS are configured in THREDDS.

The Delft FEWS GUI can directly download the data by dataset or by event. Data is downloaded to a local folder.

4.4.3 Ingest in Delft-FEWS stand alone

Once downloaded, the data can be ingested by a Delft-FEWS stand alone with the aim to bring the local datastore back in the 'original state', i.e. with a similar time series set definition as used when Delft-FEWS ran the workflow to produce the data. This exact timeseries definition is included in the netcdf files during the archive export process. The importArchiveModule has been implemented to uses this definition included in the netcdf files to put the data back in Delft-FEWS local datastore

Figure 4.16 and Table 4.10 describe the configuration details for the import module.

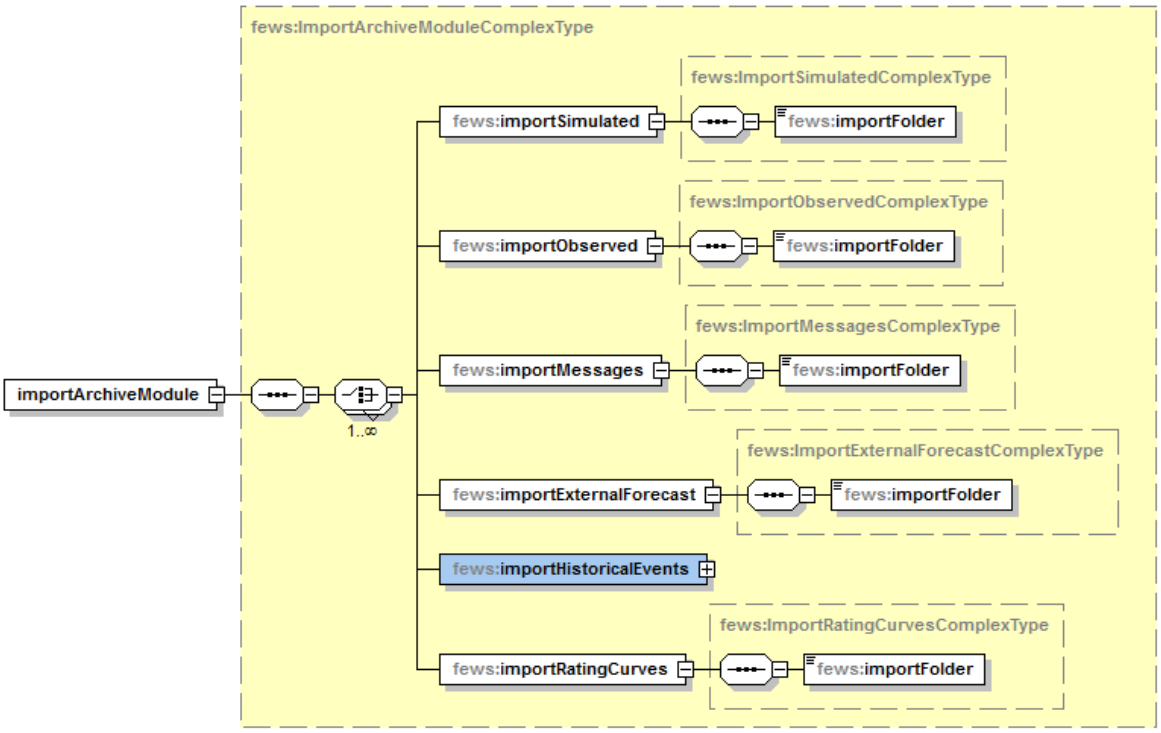


Figure 4.16 Configuration of archive Import (importArchiveModule.xsd)

Table 4.10 importArchiveModule configuration

Element	Format	Description
importSimulated/importFolder	string (path)	Full path where the simulated datasets are made available for import by Delft-FEWS
importObserved/importFolder	string (path)	Full path where the observed datasets are made available for import by Delft-FEWS
importMessages/importFolder	string (path)	Full path where the messages datasets are made available for import by Delft-FEWS
importExternalForecast/importFolder	string (path)	Full path where the external forecasts are made available for import by Delft-FEWS
importRatingCurves/importFolder	string (path)	Full path where the rating curve datasets are made available for import by Delft-FEWS

4.4.4 Ingest of Historic events in Delft-FEWS (FSS)

Historic events are a special datatype to Delft-FEWS as they can be used to overlay on an existing time series graph. Historic events are the only datasets which can be ingested in a

Delft-FEWS client server system. Normally, the archive server has a process running (the HistoricEventsExporter) which extracts historic events data from the archive and pushes them to the Forecasting Shell Server import directories for ingest in the operational database.

This import process has its own data administration process, and does not use the timeseries definition as embedded in the netcdf files. Hence an idMap may be needed to translate netcdf variables into Delft-FEWS parameters and locations. In addition, backup and failure folders can be defined to prevent loss of data in the automated process.

Figure 4.17 and Table 4.11 describe the configuration for importing historic events. This module should be executed in a separate workflow compared to the other archived datasets.

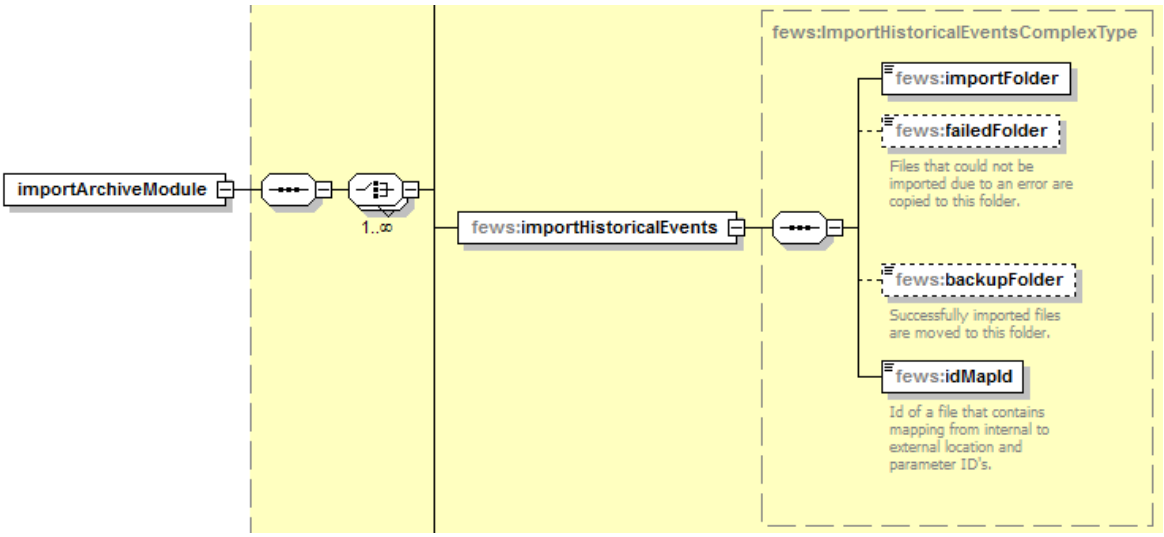


Figure 4.17 Configuration of historical events import (importArchiveModule.xsd)

Table 4.11 Historical events import configuration (importArchiveModule)

Element	Format	Description
importHistoricalEvents	importHistoricalEvents ComplexType	root element for import of Historical events into Delft-FEWS database
importHistoricalEvents ComplexType		
importFolder	string (path)	Full path where the historical events datasets are made available for import by Delft-FEWS
failedFolder	string (path	Full path where Delft-FEWS can put any dataset which failed on import
backupFolder	string (path	Full path where Delft-FEWS can back up any dataset which is being imported
idMapId	string	Idmap to translate NetCDF variables to FEWS parameters/locations

5 Archive server configuration

Archive server configuration is composed of (i) content configuration (ii) data management configuration (iii) server configuration.

5.1 Archive Content Configuration

The archive content configuration defines the event types and areas that can be discovered from the archive the Areas.xml holds the id (= folder name) and pretty name that can be used to discover the data. The archive content configuration is held in directory /data/archive/archiveConfig.

The configuration is composed of:

- eventTypes.xml (see Figure 3.10)
- Areas.xml (see Figure 5.1 and Table 5.1)

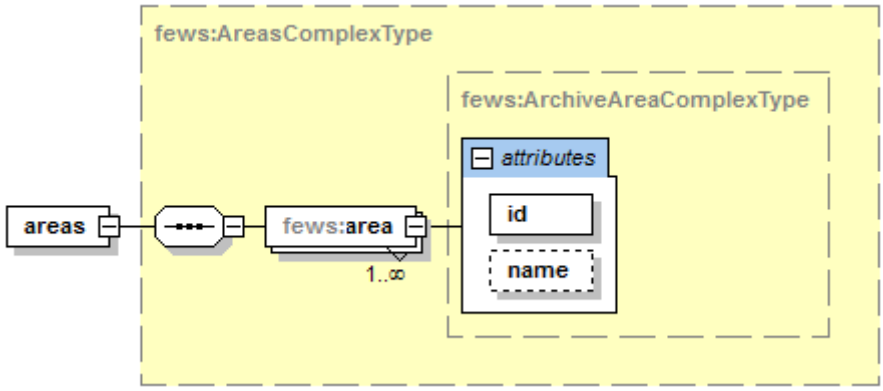


Figure 5.1 Archive content configuration of areas (areas.xsd)

Table 5.1 Delft-FEWS export configuration for archiving rating curves (general section)

Element	Format	Description
Areas ComplexType		
area	id-string name-string	id = folder name (preferably lower case, no spaces recommended) name = Pretty name (Capitals, spaces etc)

5.2 Archive Server Configuration

The archive server configuration provides the folder references, server connection details and harvester instructions for the various archive server processes. The file typically resides in: `/opt/fews/archive/managementtools/archiveServerConfig.xml`.
Details are provided in Figure 5.2 with an example in Figure 5.3.

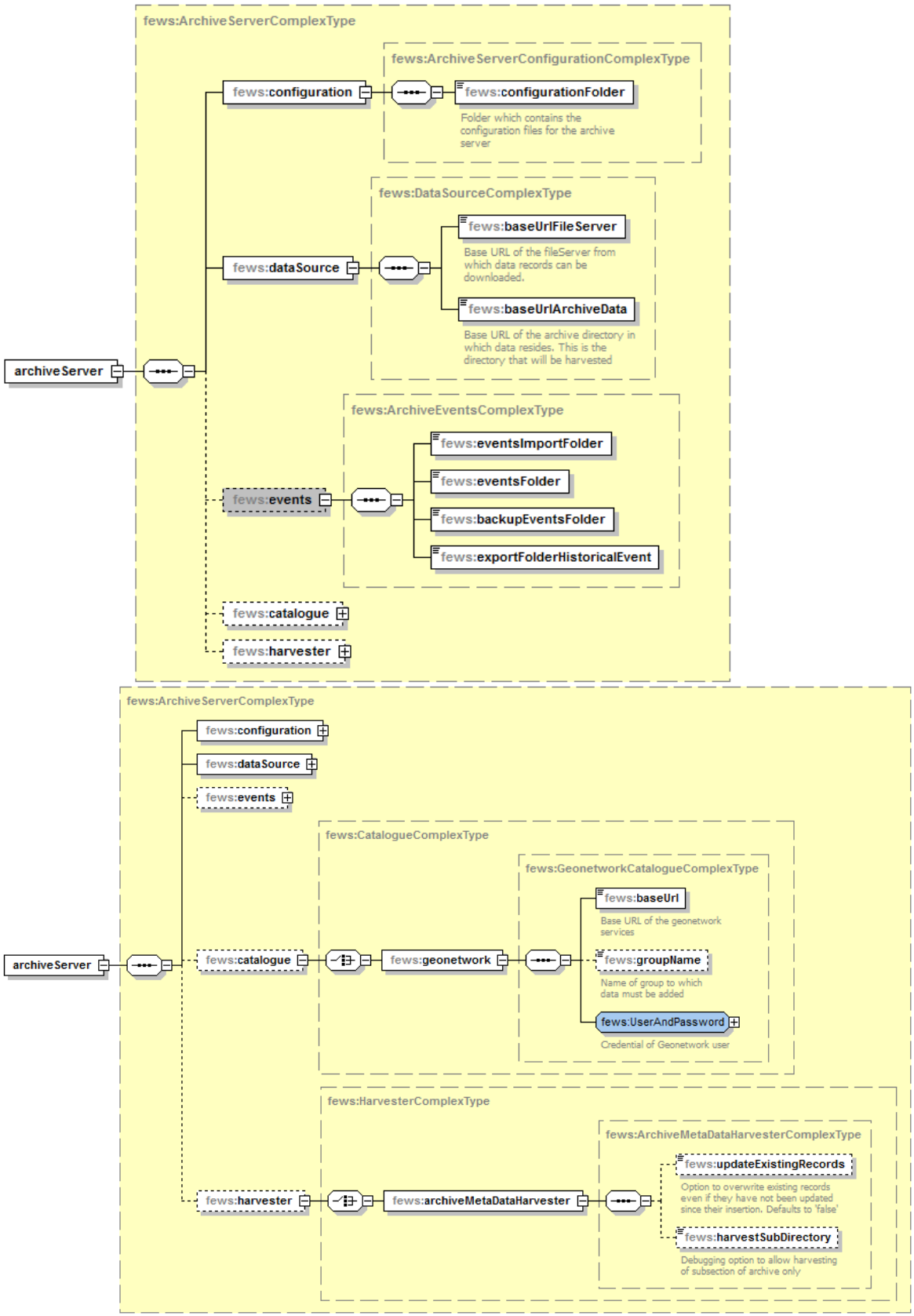


Figure 5.2 Archive server configuration (archiveServer.xsd)

Table 5.2 Contents of archive server configuration (archiveServer.xsd)

Element	Format	Description
ArchiveServer ComplexType		
archiveEnvironmentName	string (optional)	name to be used in archive web console
configuration	ArchiveServerConfiguration ComplexType	reference to archive content configuration
configurationFolder	string (path)	folder where the archive configuration (Areas.xml, eventTypes.xml) resides
dataSource	DataSource ComplexType	reference to place where archive data files reside and (THREDDS) webserver address hosting these files
baseUrlFileServer	string (http)	Url of webserver hosting the data files
baseUrlArchiveData	string (path)	Url/path where archive data resides
events	ArchiveEvents complexType	
eventsFolder	string (path)	Url/path to events repository, i.e. folder where archived events reside
eventsBackupFolder	string (path)	Url/path where backup of events resides to enable merging new events with the evnts repository
exportFolderHistoricEvents	string (path)	Url/path to FEWS directory from which an FSS will pick up historical events for import
catalogue	Catalogue Complextype	reference to catalogue service
geonetwork	GeoNetwork Catalogue ComplexType	reference to geoNetwork catalogue service
geonetwork-baseUrl	string (URL)	Url to GeoNetwork webservice
groupName	string	Group name where catalogue records should be added
user	string	geonetwork user name
password/ encryptedPassword	string	geonetwork password/encrypted password
harvester	Harvester ComplexType	general section for harvester settings
archiveMetaDataHarvester	ArchiveMetadataHarvester ComplexType	settings for harvester dedicated to ArchiveMetaData

updateExistingRecords	boolean	option to force overwrite existing catalogue records even with any change in dataset or metadata
harvestSubDirectory	string (path)	path, relative to baseUrlArchiveData, to harvest only a sub-directory of the archive

```
<arc:configuration>
    <configurationFolder>//data/archive/archiveConfig</configurationFolder>
</arc:configuration>
<arc:dataSource>
    <arc:baseUrlFileServer>http://tl-tc010.xtr.deltares.nl:8080/thredds/fileServer/Archive</arc:baseUrlFileServer>
    <arc:baseUrlArchiveData>file:///data/archive/data</arc:baseUrlArchiveData>
</arc:dataSource>
<!--Optional:-->
<arc:events>
    <arc:eventsImportFolder>/data/archive/importEvents</arc:eventsImportFolder>
    <!-- deprecated-->
    <arc:eventsFolder>/data/archive/events</arc:eventsFolder>
    <backupEventsFolder>/data/archive/events_backup</backupEventsFolder>
    <arc:exportFolderHistoricalEvent>/data/archive/events_backup</arc:exportFolderHistoricalEvent>
</arc:events>
<!--Optional:-->
<arc:catalogue>
    <arc:geonetwork>
        <arc:baseUrl>http://tl-tc011.xtr.deltares.nl:8080/geonetwork10.2/srv/eng/</arc:baseUrl>
        <arc:user>hyfs</arc:user>
        <!--You have a CHOICE of the next 2 items at this level-->
        <!--<arc:encryptedPassword>string</arc:encryptedPassword>-->
        <arc:password>hyfs123</arc:password>
    </arc:geonetwork>
</arc:catalogue>
<!--Optional:-->
<arc:harvester>
    <arc:archiveMetaDataHarvester>
        <!--Optional:-->
        <arc:updateExistingRecords>false</arc:updateExistingRecords>
        <!--     <arc:harvestSubDirectory>2014</arc:harvestSubDirectory> -->
    </arc:archiveMetaDataHarvester>
</arc:harvester>
</arc:archiveServer>
```

Figure 5.3 Example archive server configuration file

6 Hardware

6.1 Archive server components

An archive needs hardware to run the application and store the data. While the archive consists of many smaller components, the most logical grouping is presented in Figure 6.1.

The archive server is the base layer of the archive. It is primarily a large file server, with a few support processes and a two web-services deployed with Tomcat: the THREDDS data server and the Deltares Archive Web Server.

The catalogue is composed of two components: the catalogue service (GeoNetwork) and the associated catalogue database. The catalogue database is typically the open source RDBMS PostGress-PostGIS, although IT departments could opt for the more costly options Oracle Spatial or SQL Server Spatial'

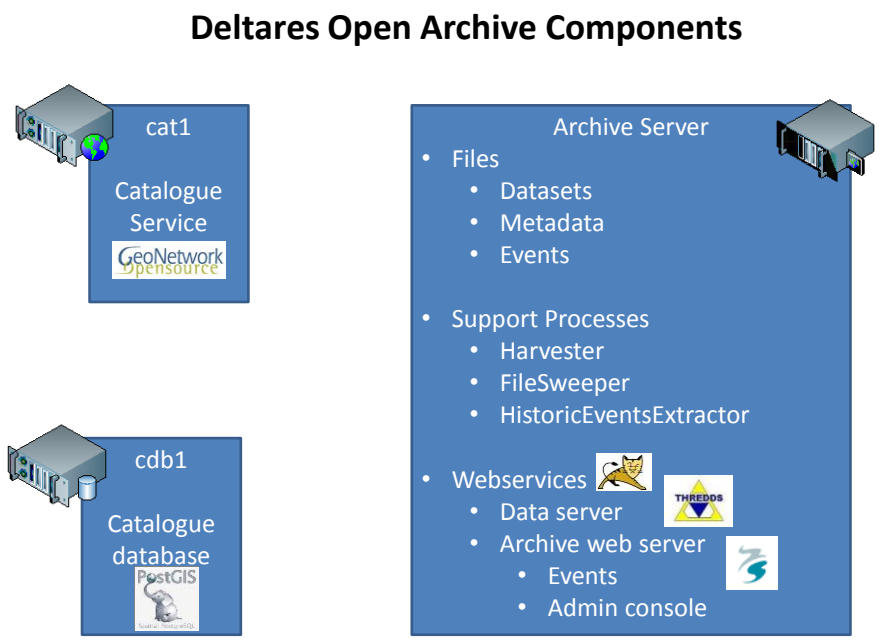


Figure 6.1 Archive server components (logical and physical)

In terms of hardware, various configurations could be imagined, primarily on the catalogue side. Physical or virtual deployment, expected user load and up-time requirements play a major role in deciding what the best configuration is for a particular organisation.

If many users will simultaneously query the catalogue, the load may be become reasonable heavy. This would advocate for separate physical servers for the catalogue and the catalogue web-service. If medium to low usage is expected, an IT-department may choose to host the catalogue database instance on an existing database server and/or they may choose to combine the catalogue database server and the catalogue webservice on one machine.

When deployed within a VM environment, it might be best to put each logical component on a separate VM. Experience needs to be gained to identify where the threshold is to move from virtual to physical deployment.

6.2 Example hardware specs

The exact hardware specification depends very much on the anticipated use by the organisation. Table 6.1 is just a general starter for the discussion what the exact hardware specs should be.

Table 6.1 Example hardware specification for the archive server

Hardware Component	**disk space**	**RAM**	**CPU**
Archive server	1Tb -100 Tb	4GB	2
Catalogue server	5Gb	2GB	1
Database server	5Gb	1GB	1