Export module

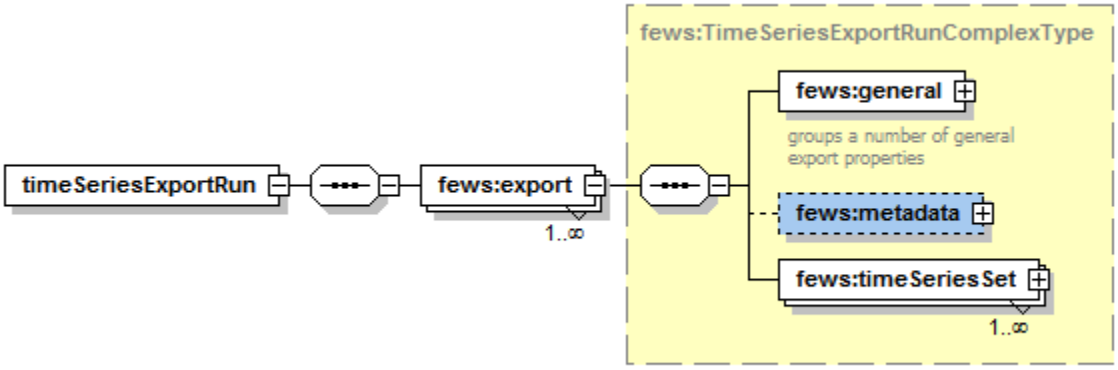
EXTERNAL_FORECAST_TIME

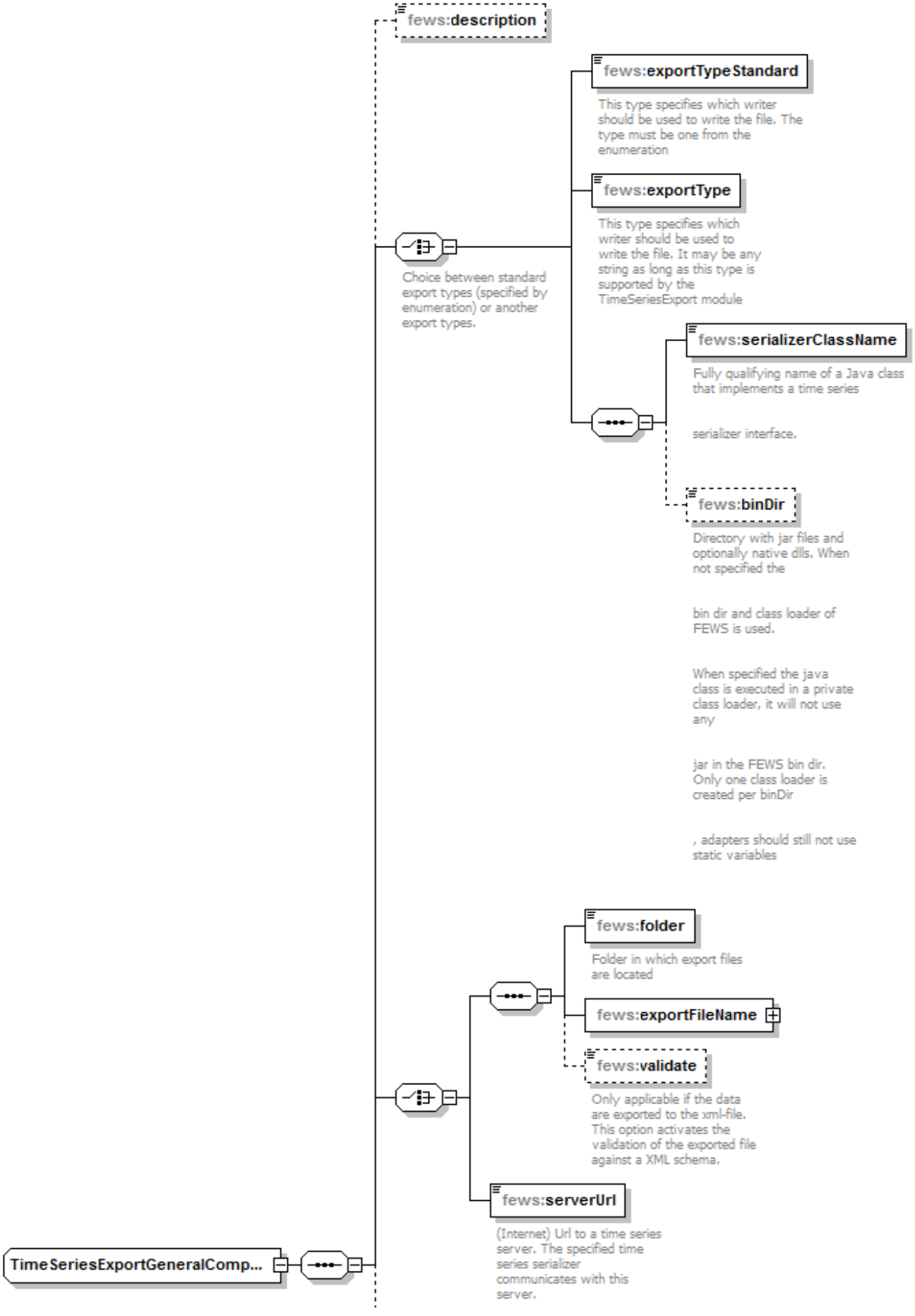
What	nameofinstance.xml
Required	no
Description	Export data (timeseries) from Delft-Fews to several file formats
schema location	https://fewsdocs.deltares.nl/schemas/version1.0/timeSeriesExportRun.xsd

Configuration

The export module can export timeseries for use in other systems. The configuration of the module is split into three sections:

- General: Specify file name, data type etc...
- metadata: Export specific settings
- timeseriesSets: actual data to export


timeSeriesExportRun
fews:export
1..∞
fews:TimeSeriesExportRunComplexType
fews:general
groups a number of general export properties
fews:metadata
fews:timeSeriesSet
1..∞


TimeSeriesExportGeneralComp...
fews:description
fews:exportTypeStandard
This type specifies which writer should be used to write the file. The type must be one from the enumeration
fews:exportType
This type specifies which writer should be used to write the file. It may be any string as long as this type is supported by the TimeSeriesExport module
Choice between standard export types (specified by enumeration) or another export types.
fews:serializerClassName
Fully qualifying name of a Java class that implements a time series
serializer interface.
fews:binDir
Directory with jar files and optionally native dlls. When not specified the
bin dir and class loader of FEWS is used.
When specified the java class is executed in a private class loader, it will not use any
jar in the FEWS bin dir. Only one class loader is created per binDir
, adapters should still not use static variables
fews:folder
Folder in which export files are located
fews:exportFileName
fews:validate
Only applicable if the data are exported to the xml-file. This option activates the validation of the exported file against a XML schema.
fews:serverUrl
(Internet) Url to a time series server. The specified time series serializer communicates with this server.


fews:user
User name
fews:password
Password for user
fews:encryptedPassword
Encrypted password for user
User name and password, required for protected servers
fews:idMapId
Id of IdMap to be used for parameterId mapping
fews:unitConversionsId
Id of UnitConversions to be used for unit mapping
fews:flagConversionsId
Id of flagConversions to be used for flag mapping
fews:exportMissingValue
Missing value definition for this time series. Defaults to NaN if not defined.
fews:exportMissingValueString
fews:omitMissingValues
Are missing values to be written to the export file or should they be left out.
fews:exportTimeZone
Time zone of the export data.
fews:convertDatum
Convert datum to local datum during import. The conversion wil be done for all parameters which use datum (as configured Parameters.xsd)
fews:forecastSelectionPeriod
use this period to select all approved forecasts with forecast start time lying within this period
fews:exportManualChanges
Since 2013.01. FEWS-8798. use this to only export manual changes
fews:coldStartDataExportRelati...
Optional. If this is specified, then the period for which data is exported will be truncated for simulated data that was created in a run that started with a cold state. In other words for this type of data, any data before the specified

start time will not be exported. This start time is relative to the time zero of the run in which the data was created. This can be used to avoid exporting data that was created during the warm-up period of a model run after a cold start. Note: this option only works when the default state selection was configured to be warm state for the run in which the data was created.

In the sections below the different elements of the configuration are described

General

description

An optional description

exportTypeStandard

This type specifies which writer should be used to write the file. The type must be one from the enumeration. Presently (2007/02) only bfg and pi are included in this list.

exportType

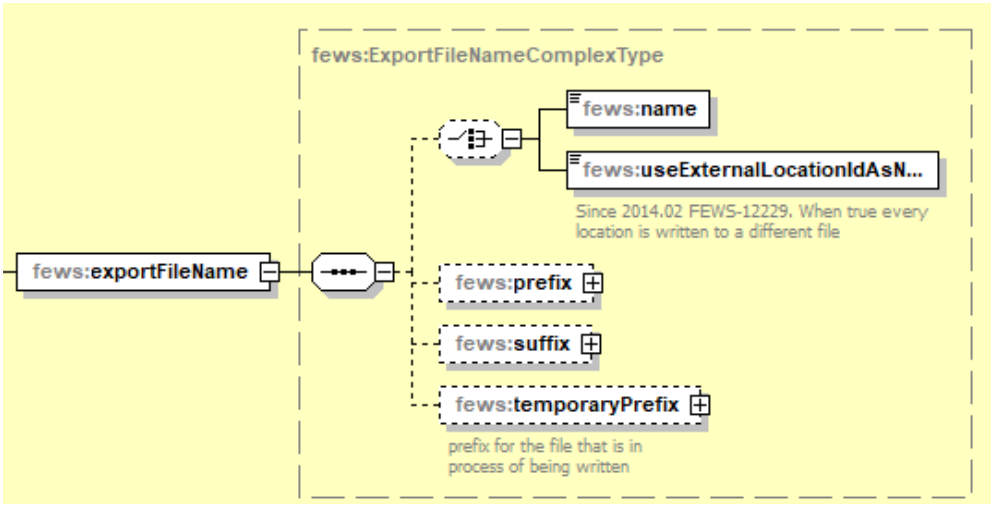
This type specifies which writer should be used to write the file. It may be any string as long as this type is supported by the TimeSeriesExport module. The list of supported types is given here.

folder

Folder (directory) in which to store the exported files.

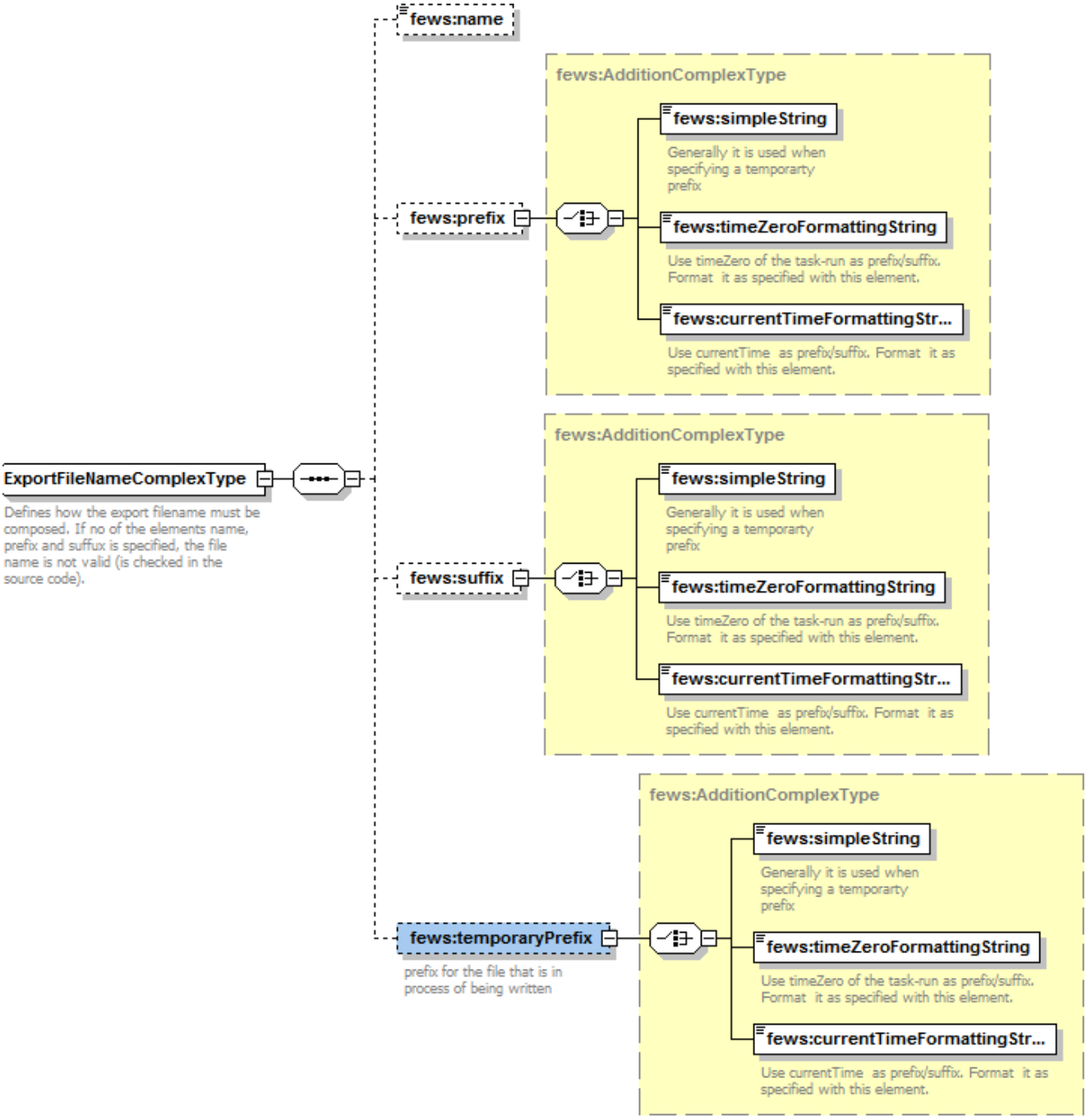
exportFileName

This elements describes how to construct the filename(s) of the exported file(s).



If only the name element is given a fixed name is used for each export. The prefix and suffix elements describe how to create a filename prefix and/or suffix. The temporaryPrefix is used to generate a prefix for the temporary file as it is being written. After that the file is renamed.

The option useExternalLocationIdAsName (instead of the regular file name) forces the use of the location ID or Name as filename. Note that this option does not work in combination with an idMap that filters the locations to be exported.



validate

Optional element. Only applicable if the data are exported to the xml-file. This option activates the validation of the exported file against a XML schema.

idmapId

Id of IdMap to be used for parameterId and locationId mapping

externalLocationNameFunction (since Delft-FEWS 2023.2)

For export functions that can write location names along side location ID's, one can choose the location attribute that should be parsed to the location name field. It requires mentioning the attribute between @ signs; e.g.: <externalLocationNameFunction>@externalLocationName@< /externalLocationNameFunction>

unitConversionsId

Id of UnitConversions to be used for unit mapping

flagConversionsId

Id of flagConversions to be used for flag mapping

exportMissingValue/exportMissingValueString

Missing value definition for this time series. Either a string or a number. Defaults to NaN if not defined.

exportAttribute

Since 2015.02. Location, Parameter or Qualifier attribute that can be used for export. Specification is needed to include it in the data available for export. For parameter attributes it is supported for the following export types:

NetcdfScalarTimeSeriesSerializer
NetcdfMatroosScalarTimeSeriesSerializer
NetcdfScalarNonEquidistantTimeSeriesSerializer
NetcdfVerticalProfileSerializer
NetcdfGridTimeSeriesSerializer
NetcdfZLayersTimeSeriesSerializer
NetcdfDomainTimeSeriesSerializer

exportLocationAttributeAsNetCDFVariable

Since 2021.01. Adds a variable to the NetCdf file. Name of the variable is the value of ncVariable, the value is the configured location attribute of attributeId.

config example:

```
<exportLocationAttributeAsNetCDFVariable>
   <ncVariable>LocationAlias</ncVariable>
   <attributeId>niceName</attributeId>
</exportLocationAttributeAsNetCDFVariable>
```

Example result (in txt format) of using exportLocationAttributeAsNetCDFVariable:

```
netcdf {
  dimensions:
    time = 3;
    stations = 6;
    char_leng_id = 64;
    char_leng_name = 255;
    char_leng_attribute = 255;
  variables:
    double time(time);
      time:standard_name = "time";
      time:long_name = "time";
      time:units = "minutes since 1970-01-01 00:00:00.0 +0000";
      time:axis = "T";

    double lat(stations);
      lat:standard_name = "latitude";
      lat:long_name = "Station coordinates, latitude";
      lat:units = "degrees_north";
      lat:axis = "Y";
      lat:_FillValue = 9.96921E36;

    double lon(stations);
      lon:standard_name = "longitude";
      lon:long_name = "Station coordinates, longitude";
      lon:units = "degrees_east";
      lon:axis = "X";
```

```
      lon:_FillValue = 9.96921E36;

    double y(stations);
      y:standard_name = "latitude";
      y:long_name = "y coordinate according to WGS 1984";
      y:units = "degrees_north";
      y:axis = "Y";
      y:_FillValue = 9.96921E36;

    double x(stations);
      x:standard_name = "longitude";
      x:long_name = "x coordinate according to WGS 1984";
      x:units = "degrees_east";
      x:axis = "X";
      x:_FillValue = 9.96921E36;

    double z(stations);
      z:long_name = "height above mean sea level";
      z:units = "meters";
      z:_FillValue = 9.96921E36;

    char station_id(stations, char_leng_id);
      station_id:long_name = "station identification code";
      station_id:cf_role = "timeseries_id";

    char station_names(stations, char_leng_name);
      station_names:long_name = "station name";

    char LocationAlias(stations, char_leng_attribute);
      LocationAlias:niceName = "location attribute value";

    float waterlevel(time, stations);
      waterlevel:standard_name = "water_surface_height_above_reference_datum detection_minimum";
      waterlevel:long_name = "waterlevel";
      waterlevel:units = "m";
      waterlevel:_FillValue = -9999.0f;
      waterlevel:coordinates = "lat lon";
      waterlevel:cell_methods = "time: maximum";

  // global attributes:
  :Conventions = "CF-1.6";
  :title = "Data";
  :institution = "Deltares";
  :source = "Export NETCDF-CF_TIMESERIES from Delft-FEWS";
  :history = "actual history attribute text replaced by dummy text in unit test";
  :references = "http://www.delft-fews.com";
  :Metadata_Conventions = "Unidata Dataset Discovery v1.0";
  :summary = "Data exported from Delft-FEWS";
  :date_created = "actual date_created attribute text replaced by dummy text in unit test";
  :fews_implementation_version = "0.0";
  :fews_build_number = "development";
  :coordinate_system = "WGS 1984";
  :featureType = "timeSeries";
  :time_coverage_start = "2003-03-01T00:00:00+0000";
  :time_coverage_end = "2003-03-01T00:30:00+0000";
  :geospatial_lon_min = "-3.5458";
  :geospatial_lon_max = "-2.15939";
  :geospatial_lat_min = "51.74003";
  :geospatial_lat_max = "52.76901";
 data:
time =
  {1.744128E7, 1.7441295E7, 1.744131E7}
lat =
  {51.93591, 51.74003, 52.76901, 52.55523, 52.19899, 52.43011}
lon =
  {-2.15939, -2.2469, -3.1088, -2.37608, -2.38931, -3.5458}
y =
  {51.93591, 51.74003, 52.76901, 52.55523, 52.19899, 52.43011}
x =
  {-2.15939, -2.2469, -3.1088, -2.37608, -2.38931, -3.5458}
z =
```

```
  {1.1, 2.2, 3.3, 9.96921E36, 5.5, 6.6}
station_id ="26", "27", "28", "24", "29", "25"
station_names ="Slate Mill", "Ebley Mill", "Llanymynech", "Burcote", "Knightsford Bridge", "Rhos-Y-Pentref"
LocationAlias ="A-attribute-2026", "A-attribute-2027", "A-attribute-2028", "A-attribute-2024", "A-attribute-
2029", "A-attribute-2025"
waterlevel =
  {
    {-9999.0, 1.21, 1.31, 1.41, 1.51, 1.61},
    {-9999.0, 2.21, 2.31, 2.41, 2.51, 2.6100001},
    {-9999.0, 3.21, 3.31, 3.41, 3.51, 3.6100001}
  }
}
```

omitMissingValues

If set to true records with missing values are not exported

precision

Available since 2018.02. Optional element to set the number of decimals all values should be displayed with. If set, additional zeros will be appended and /or values will be rounded when necessary.

It is possible to configure a valueResolution for parameters via the parameters.xml. The configured precision for time series with parameters which have a value resolution should never exceed the maximum number of decimals needed to display values with this resolution. If the precision does exceed this, a warning will be given and the configured precision will be ignored.

The precision can never exceed 8 decimals due to limitations on the resolution with which values can be stored in the FEWS database (floating point errors).

exportTimeZone

TimeZone in which to export the data. Can either be a string (timeZoneName) or an offset (timeZoneOffset).

convertDatum

Convert (vertical) datum to local datum during export. The conversion will be done for all parameters which use datum (as configured in Parameters.xml) The local datum is defined in the z element in the locations.xml file.

geoDatum

Convert the geographical coordinate system (horizontal datum and projection) to specified geoDatum during export. Not all serializers support this parameter so please check the documentation for a particular serializer to see if it is supported.

ensembleMemberFormat

Available since 2019.02. Can either have value 'name' or 'index'. If 'name' is configured, the ensemble member Id is written. Otherwise the ensemble member index is written.

forecastSelectionPeriod

If configured all forecasts with a forecast time within the configured period will be exported. Since 2020.01 all forecasts will be exported to the same file.

When also configuring a <timeZeroFormattingString> in the <prefix> of the <exportFileName>, each forecast will be exported to a separate file to easily differentiate between the different forecasts.

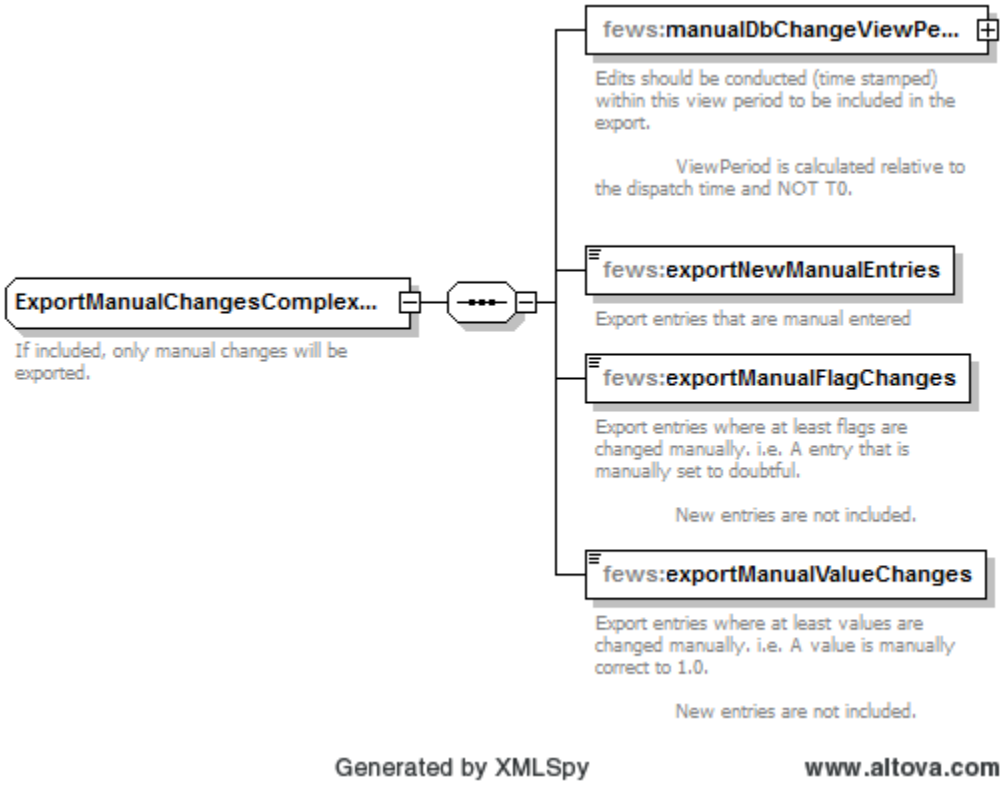
Note: earlier versions export a separate file for each forecast by default. The forecast time is used to create a file extension (format yyyyMMddHH), unless <timeZeroFormattingString> is configured

Example configuration:

```
<export>
        <general>
                <exportType>SomeValidExportType</exportType>
                <folder>MyExportFolder</folder>
                <exportFileName>
                        <name>_MyExportedFile.txt</name>
                        <prefix>
                                <timeZeroFormattingString>yyyyMMddHHmm</timeZeroFormattingString>
                        </prefix>
                </exportFileName>
                <idMapId>MyIdMap</idMapId>
                <forecastSelectionPeriod start="-2" end="0" unit="day"/>
        </general>
        ...
</export>
```

exportManualChanges

If used, only manual changed to the data will be exported. Unless the manualDBChangeViewPeriod is used, the relativeViewPeriod from the associated timeSeriesSet is used. Note: the view period is calculated relative to the dispatch time and not T0



exportChanges

If configured, any changes to the data in the configured period will be exported. Unless the dbChangeViewPeriod is configured, the relativeViewPeriod from the associated timeSeriesSet is used . Note: the view period is calculated relative to the dispatch time and not T0

An example:

```
<general>
    <exportType>PI</exportType>
    <folder>$EXPORT_FOLDER$</folder>
    <exportFileName>
        <name>exportedTimeSeries.xml</name>
    </exportFileName>
    <exportChanges>
         <dbChangeViewPeriod unit="day" multiplier="2"/>
    </exportChanges>
</general>
```

columnSeparator and decimalSeparator

Since 2016.01 (so far only implemented for GeneralCsv export type) it is possible to choose from multiple column separators: comma "," or semi-colon ";" or pipe "|" or tab "	" or space " "

When specifying a column separator it is compulsory to also specify the decimal separator as comma "," or point "."

For an example see generalCsv export type.

properties

Here properties for specific serializers can be configured. For example for the NetCDFSerializers the following properties will be taken into account:

Example configuration: properties

```
<properties>
   <bool key="includeComments" value="true"/>
   <bool key="includeFlags" value="true"/>
   <bool key="includeTSProperties" value="true"/>
   <bool key="tryCompactingNetCDFData" value="true"/>
   <string key="netCDFWriteFormat" value="netcdf4"/>
   <int key="netCDF4DeflateLevel" value="6"/>
</properties>
```

includecomments

Export comment for each time step to NetCDF, default false

includeFlags

Export flag for each time step to NetCDF, default false

includeTSProperties

Export time series properties for each time step to NetCDF, default false

tryCompactingNetCDFData

Depending on the difference between the minimum and maximum and the value resolution of a netcdf variable, try to use smaller sized integer variables like short or byte to compact the data. A scale factor and offset will be used to fit the data in the smaller sized variable and will added to the netCDF variable as attributes. This kind of compression will keep the precision of the value resolution. All standard netCDF viewers will take these attributes into account automatically, but other tools and especially scripts might not. This property will be false by default and only works for scalar and grid data.

netCDFWriteFormat

With this property the netcdf format can be set to netcdf4, default it will be netcdf3. Netcdf4 is needed to write compressed netcdf files which can result in 2 to 100 times smaller files.

netCDF4DeflateLevel

This property only works with netcdf4.

With this property the deflate level for writing compressed netcdf files can be set from 0 to 9. 0 meaning no compression and 9 maximum compression. Default will be 5, this level gives best compression without losing too much time when reading or writing.

metadata

(Meta data export has only been implemented for a limited set of export types. Currrently the NetCDF, grid2shp, LILA and HHRR types export meta data)

Optional metadata that is written in the exported file. The options netcdfMapDPhase and alertMapDPhase are deprecated (do not use these). For the other options it is possible to use the following tags:
%TIME_ZERO% the T0 of this time series export run.
%CURRENT_TIME% the current time.
%COLD_STATE_TIME% the cold state time.
%FORECAST_END_TIME% the forecast time set by the forecast length estimator or the forecast length option in the manual forecast dialog
%MODULE_INSTANCE_ID% the id of this module instance.
%MODULE_INSTANCE_NAME% the name of this module instance.
%MODULE_INSTANCE_DESCRIPTION% the configured description of this module instance.

%MODULE_INSTANCE_ATTRIBUTE(attributeId, moduleInstanceId)% moduleInstanceId is optional. When not specified the module instance of the module itself is used, like for the %MODULE_INSTANCE_ID% tag.
%WORKFLOW_ID% the id of the workflow in which this export runs.
%WORKFLOW_NAME% the name of the workflow in which this export runs.
%WORKFLOW_DESCRIPTION% the configured description of the workflow in which this export runs.

%TASK_DESCRIPTION% user description of the forecast in which the export runs.

%WHAT_IF_NAME% name of the what-if used in the forecast in which the export runs.

%USER_ID% the id of the user by which this export run is executed

%EXTERNAL_FORECAST_TIME% - external analysis time. When configuring it, it needs two parameters: the first is the external forecast id, the second should be the time format. Neither the id nor the time format should contain "," (a comma). The two arguments should be separated by a comma.
Example: %EXTERNAL_FORECAST_TIME(thisIsTheId),(yyyy/MM/dd HH:mm:ss z)%

%COLD_STATE_TIME(yyyy/MM/dd HH:mm:ss z)% - the cold state start time. If data is unavailable it will be filled as "Unknown".

Since 2022.02 location/parameter/qualifier/moduleInstance attributes between @ are also recognized.

Configuration example of metadata:

```
<metadata>
        <title>title</title>
        <institution>  institution  </institution>
        <source>source</source>
        <history>Exported at time zero = %TIME_ZERO(yyyy/MM/dd HH:mm:ss z)% in module instance %
MODULE_INSTANCE_ID% as part of workflow %WORKFLOW_NAME% by user %USER_ID%.</history>
        <references>references</references>
        <comment>The actual time of writing was %CURRENT_TIME(yyyy-MM-dd HH:mm:ss z)%</comment>
        <summary>A summary of the data for @ATTRIBUTEID@</summary>
        <keyword>keyword1</keyword>
        <keyword>  keyword  with  lots  of  spaces  </keyword>
        <keyword>keyword 3</keyword>
        <customAttributes>
                <string key="emptyAttribute" value="    "/>
                <int key="  custom2  " value="123456"/>
                <string key="custom_3" value="This is a custom attribute with 'quotes' in it."/>
                <string key="    " value="attribute with empty key specified is not written"/>
                <float key="just_another_float" value="3.5"/>
                <bool key="truth" value="true"/>
        </customAttributes>
</metadata>
```

title

A short description of the dataset. Its value will be used by THREDDS opendap servers as the name of the dataset. It therefore should be human readable and reasonable to display in a list of such names.

institution

Specifies where the original data was produced.

source

The method of production of the original data. If it was model-generated, source should name the model and its version, as specifically as could be useful. If it is observational, source should characterize it (e.g. "surface observation" or "radiosonde").

history

Provides an audit trail for modifications to the original data. It should contain a separate line for each modification with each line including a timestamp, user name, modification name, and modification arguments. Its value will be used by THREDDS opendap servers as a history-type documentation. It is recommended that each line begins with a timestamp indicating the date and time of day at which the modification was performed.

references

Published or web-based references that describe the data or methods used to produce it.

comment

Miscellaneous information about the data or methods used to produce it.

summary

The "summary" attribute gives a longer description of the dataset. In many discovery systems, the title and the summary will be displayed in the results list from a search. It should therefore capture the essence of the dataset it describes. For instance, include information on the type of data contained in the dataset, how the data was created (e.g. instrument X or model X, run Y), the creator of the dataset, the project for which the data was created, the geospatial coverage of the data, and the temporal coverage of the data.

keyword

Optional one or more key words or phrases that are relevant to the dataset. The values in this list may be taken from a controlled list of keywords (e.g. the AGU Index list or the GCMD Science Keywords).

customAttributes

If you want to add an attribute that is not predefined in the schema, then you can add it as a custom attribute here.

timeseriesSet

Define the timeseriesset to be exported. Please note that not all exports support all timeseriestypes (e.g. csv only supports scalar type).

filterId

Since 2018.01 it is possible to configure a filter id that refers to a filter from Filters.xml in the RegionConfigFiles.

```
    </general>
   <filterId>AllQualifiersFilter</filterId>
</export>
```

This way time series can be exported based on all options present in a filter like location, parameter and qualifier constraints:

```
<filter id="AllQualifiersFilter">
   <timeSeries>
      <moduleInstanceId>ExportRunMultipleTimeSeries</moduleInstanceId>
   </timeSeries>
   <relativeViewPeriod unit="day" start="-7" end="0"/>
   <locationConstraints>
      <idContains contains="12965"/>
   </locationConstraints>
   <parameterConstraints>
      <idContains contains="H.m"/>
   </parameterConstraints>
</filter>
```

Annotation location set id

Since 2021.01 it is possible to export annotations via the generalCsv type.

It will export all annotations that exist for the configured <annotationLocationSetId>annotationLocationSet</annotationLocationSetId>

The value colum will be used for the text of the annotation itself.

Example annotations export

```
<timeSeriesExportRun xmlns="http://www.wldelft.nl/fews" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.wldelft.nl/fews http://fews.wldelft.nl/schemas/version1.0/timeSeriesExportRun.
xsd">
   <export>
      <general>
         <exportTypeStandard>generalCsv</exportTypeStandard>
         <folder>export</folder>
         <exportFileName>
            <name>ExportGeneralCsvAnnotations.csv</name>
         </exportFileName>
         <table>
            <dateTimeColumn name="DATE" pattern="dd-MM-yy HH:mm"/>
            <startDateTimeColumn name="START" pattern="dd-MM-yy HH:mm"/>
            <endDateTimeColumn name="END" pattern="dd-MM-yy HH:mm"/>
            <locationColumn name="locatie"/>
            <propertyColumn name="firstProperty" key="firstProperty"/>
            <propertyColumn name="secondProperty" key="secondProperty"/>
            <valueColumn name="annotatie"/>
         </table>
      </general>
      <annotationLocationSetId>annotationLocationSet</annotationLocationSetId>
   </export>
</timeSeriesExportRun>
```