05 General Adapter Module

What	*nameofinstance*.xml
Description	Configuration for the general adapter module
schema location	https://fewsdocs.deltares.nl/schemas/version1.0/generalAdapterRun.xsd

        - timeSerieSets
        - timeSerieSets: timeSerieSet
        - omitMissingValues
        - checkMissingValues
        - omitEmptyTimeSeries
        - omitEmptyFiles
        - forecastSelectionPeriod
        - ignoreNonExistingLocationSets
    - exportMapStacksActivity
        - description
        - exportFile
        - gridFile
        - locationId
        - gridName
        - gridFormat
        - timeSerieSet
    - exportProfilesActivity
    - exportDataSetActvity
        - description
        - moduleInstanceId
        - moduleDataSetName
        - moduleDataSetNameMap
        - coldStateDataSetName
    - exportParameterActivity
        - description
        - moduleInstanceId
        - fileName
        - templateLocationLooping
    - exportLocationAttributesCsvActivity (since 2020.01)
    - exportTableActivity
        - description
        - exportFile
        - tableType
        - operation
        - parameters
        - locationId/locationSetId
    - exportNetcdfActivity
        - description
        - exportFile
        - netcdfFormat
        - metadata
        - geoDatum
        - writeRealizationDimension
        - timeSeriesSets
        - omitMissingValues
        - omitEmptyTimeSeries
        - omitEmptyFiles
        - ignoreNonExistingLocationSets
    - exportRunFileActivity
        - description
        - exportFile
        - properties
    - exportNetcdfRunFileActivity (since stable build 2014.01)
        - description
        - exportFile
        - properties
        - Configuration example
        - Example of an exported netcdf run file (converted to text format):
    - exportCustomFormatRunFileActivity
        - description
        - templateFile
        - exportFile
        - locationId
        - fixedWidth
        - numberOfDecimals
        - configuration example
    - exportAreaSelectionActivity
        - description
        - exportFile
        - gridLocationId
        - gridFormat
    - exportLocationAreaActivity (currently NGMS only)
        - description
        - exportFile
        - gridLocationId
        - gridFormat
        - polygonLocationId
    - exportRatingCurveActivity
        - description

Execute activities requiring openJDK in 2018.02 and later

Since 2018.02, the Forecasting Shell and Operator Client software are shipped with a stripped-down JRE with only those java modules that the OC / FSS / CM client needs. This might cause module components such as adapters, forecast models and simulations that were perfectly running fine on the shipped JRE in 2017.02 or earlier to stop from working since additional modules might be required.

If this is the case, recommended options are to not try and resolve this in the Forecasting Shell launcher service, but to choose one of the following options:

1. Configure the executeActivity -> command -> customJreDir.
2. Configure the executeActivity -> environment variables to provide a PATH to the correct JRE version
3. Supply the component with its own stripped down embedded JRE version and use a wrapper script that is able to find the correct java runtime image.

General Adapter Configuration

A key feature of Delft-FEWS is its ability to run external modules to provide essential forecasting functionality. These modules may be developed by Deltares as well as by other companies or institutions. The Delft-FEWS system does not have any knowledge of the specific implementation of these modules. It is rather the central philosophy to have an open system, that is able to treat external modules as plug-ins that can be used if needed.

The General Adapter is the part of the Delft-FEWS system that implements this feature. It is responsible for the data exchange with the modules and for executing the modules and their adapters. The central philosophy of the General Adapter is that it knows as little as possible of module specific details. Module specific intelligence is strictly separated from the Delft-FEWS system. In this way an open system can be guaranteed. Module specific intelligence required by the module to run is vested in the module adapters.

Communication between the General Adapter and a module is established through the published interface (PI). The PI is an XML based data interchange format. The General Adapter is configured to provide the data required for a module to run in the PI format. A module adapter is then used to translate the data from the PI to the module native format. Vice versa, results will first be exported to the PI format by a module adapter before the General Adapter imports them back into Delft-FEWS.
Grid and scalar timeseries can also be exchanged using netCDF format.
This exchange format is intended for the modules that obtain their input from netCDF files and/or write their output to netCDF files.

The General Adapter module can be configured to carry out a sequence of five types of tasks;

- Startup Activities. These activities are run prior to a module run and any export import of data. The activities defined are generally used to remove files from previous runs that may implicate the current run.
- Export Activities. These activities define all items to be exported through the published interface XML formats to the external module, prior to the module or the module adapters being initialised. Grid and scalar timeseries can be exported also in netCDF format.

- Execute Activities. These execute activities define the external executables or Java classes to be run. Tracking of diagnostics from these external activities is included in this section.
- Import Activities: These activities define all items to be imported following successful completion of the module run.
- Shutdown Activities. These activities are run following completion of all other activities. The activities defined are generally used to remove files no longer required.

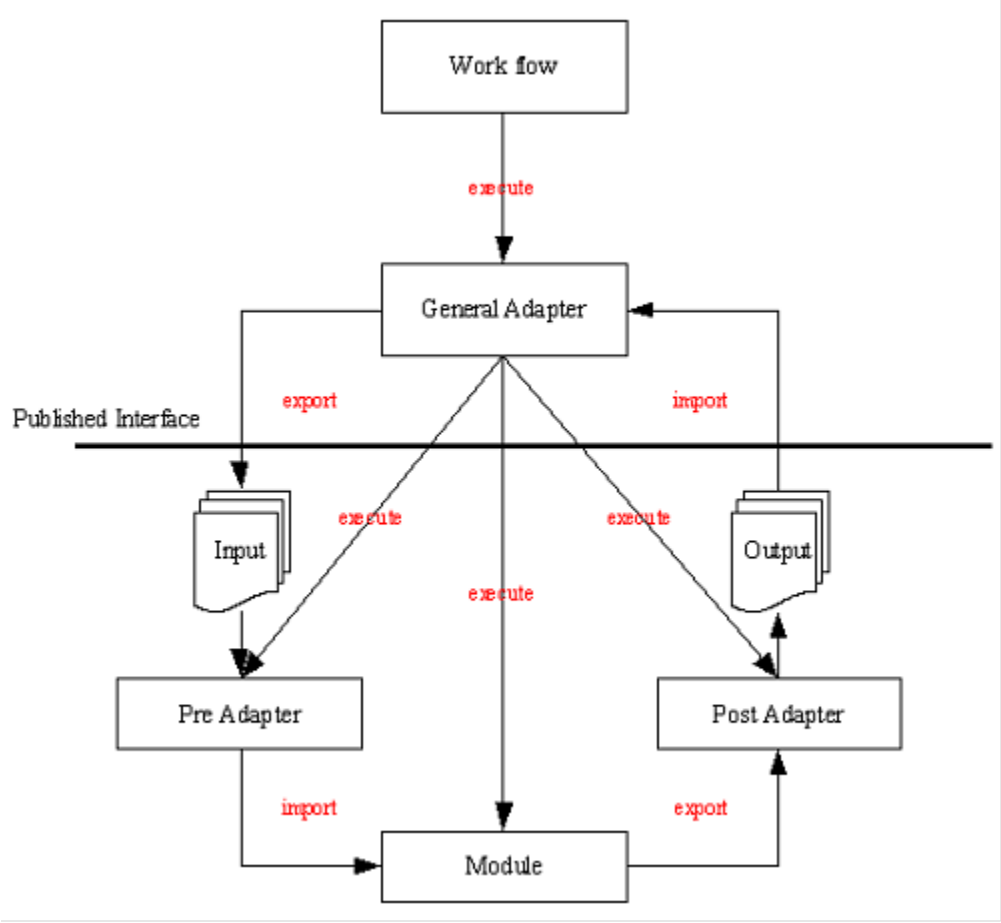


Figure 65 Schematic interaction between the General Adapter and an external module

When available as configuration on the file system, the name of the XML file for configuring an instance of the general adapter module called for example HBV_Maas_Forecast may be:

HBV_Maas_Forecast 1.00 default.xml

HBV_Maas_Forecast	File name for the HBV_Maas_Forecast configuration.
1.00	Version number
default	Flag to indicate the version is the default configuration (otherwise omitted).

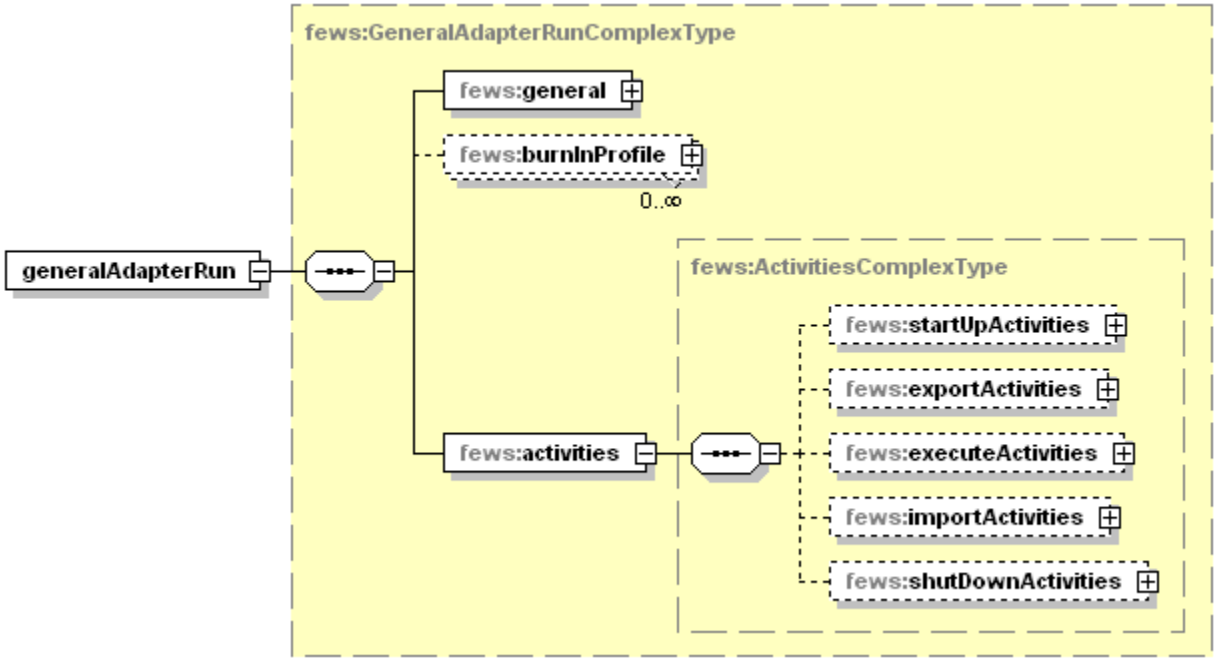


Figure 66 Elements of the General Adapter configuration

general

Root element of general settings.

burnInProfile

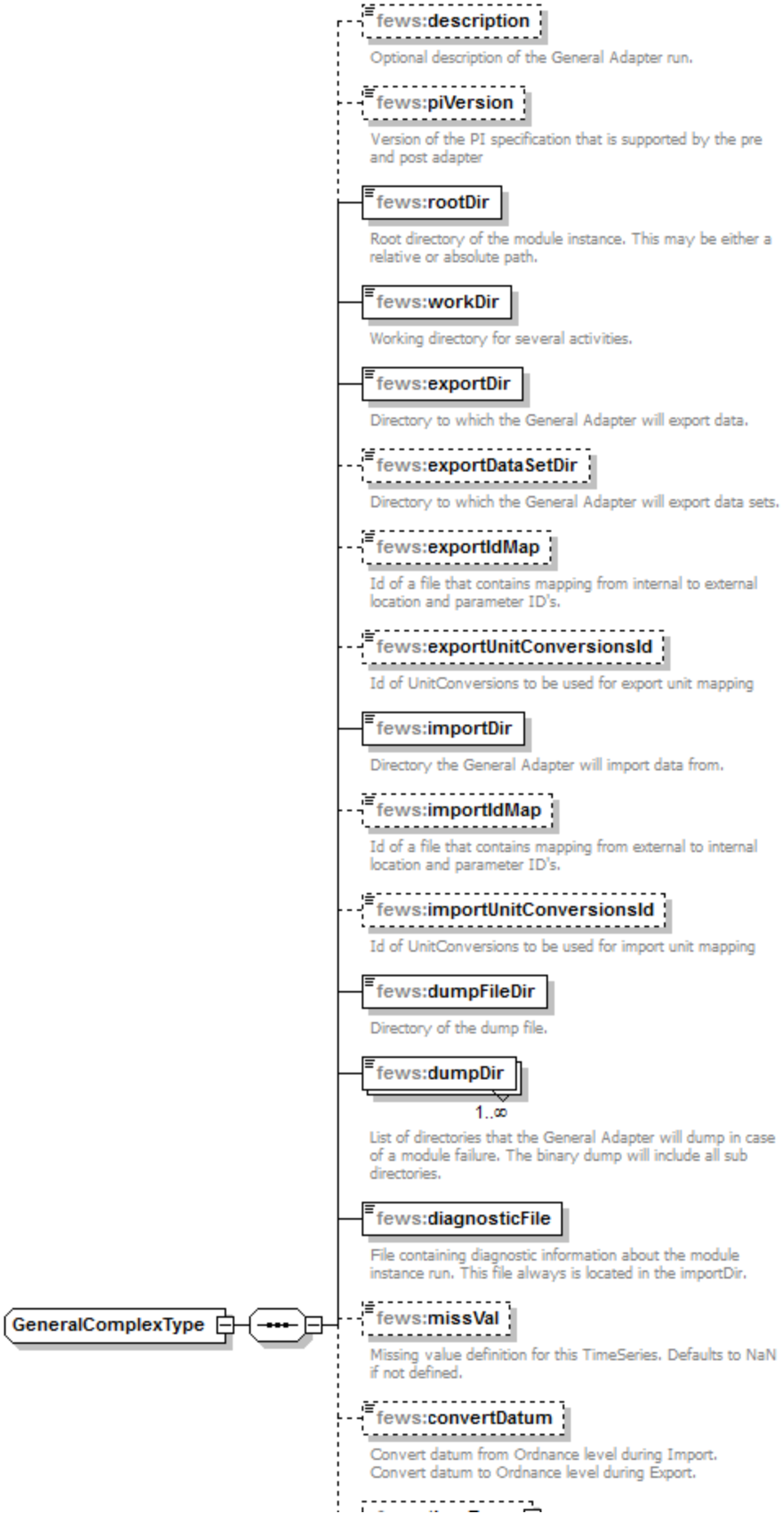
Burn-in period and initial value for cold state starts.

activities

Root element for the activities to be defined. The activities are defined in a fixed order;

- startUpActivities
- exportActivities
- executeActivities
- importActivities
- shutDownActivities

General settings


GeneralComplexType
fews:description
Optional description of the General Adapter run.
fews:piVersion
Version of the PI specification that is supported by the pre and post adapter
fews:rootDir
Root directory of the module instance. This may be either a relative or absolute path.
fews:workDir
Working directory for several activities.
fews:exportDir
Directory to which the General Adapter will export data.
fews:exportDataSetDir
Directory to which the General Adapter will export data sets.
fews:exportIdMap
Id of a file that contains mapping from internal to external location and parameter ID's.
fews:exportUnitConversionsId
Id of UnitConversions to be used for export unit mapping
fews:importDir
Directory the General Adapter will import data from.
fews:importIdMap
Id of a file that contains mapping from external to internal location and parameter ID's.
fews:importUnitConversionsId
Id of UnitConversions to be used for import unit mapping
fews:dumpFileDir
Directory of the dump file.
fews:dumpDir
1..∞
List of directories that the General Adapter will dump in case of a module failure. The binary dump will include all sub directories.
fews:diagnosticFile
File containing diagnostic information about the module instance run. This file always is located in the importDir.
fews:missVal
Missing value definition for this TimeSeries. Defaults to NaN if not defined.
fews:convertDatum
Convert datum from Ordnance level during Import.
Convert datum to Ordnance level during Export.

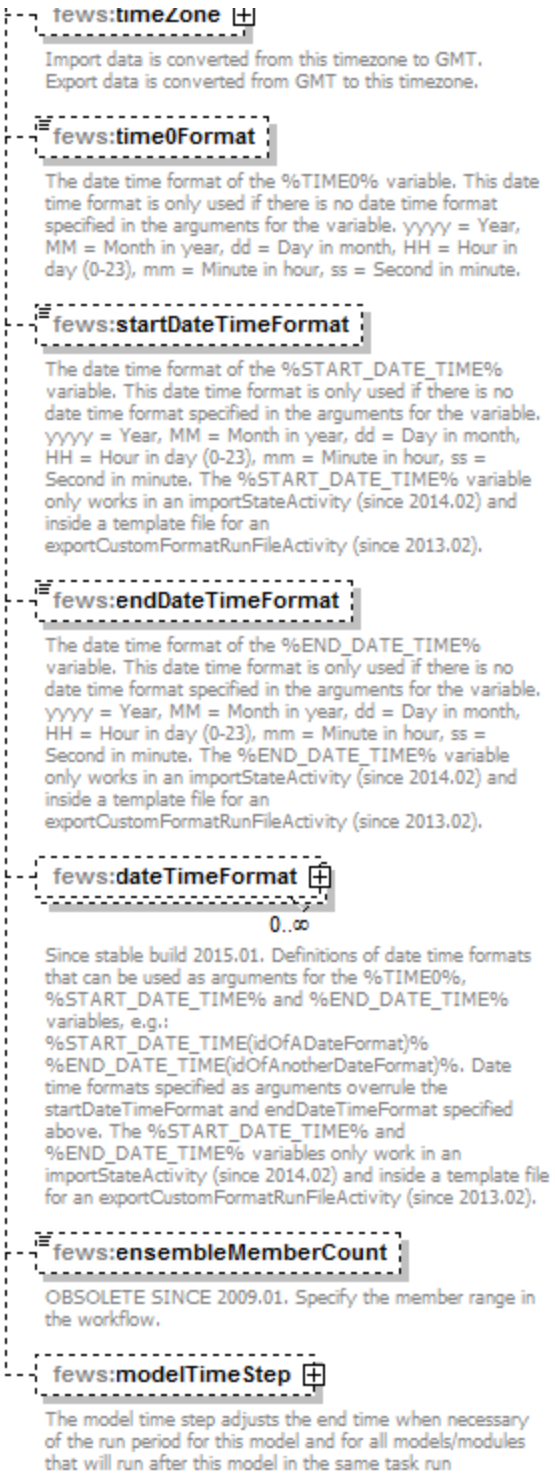


Figure 67 Elements of the general section of the general adapter configuration

description

Optional description of the configuration. Used for reference purposes only.

piVersion

Version of the PI specification that is supported by the pre and post adapter.

The general section requires several directories to be defined. To allow efficient definition of these, tags specific to the General Adapter may be used.

%REGION_HOME%	*Root directory of the DELFT-FEWS configuration*
%ROOT_DIR%	*Directory specified in the rootDir element.*
%WORK_DIR%	*Directory specified in the workDir element.*

The REGION_HOME directory should be used when possible to avoid absolute path names.

inMemoryFileTransfer

Since 2022.02. When true all exported and imported files (XML, NetCdf) are transferred in memory between FEWS and execute activities. An execute activity can read the file content from

var input = (BufferedInputStream) System.getProperties().get(file)

and write to

var output = (BufferedOutputStream) System.getProperties().get(file).

All execute activities that transfer files to and from FEWS should be java activities that don't use a custom jre and check the System.getProperties().get (file) before they fall back to read/write the file on disk. All files can be transferred in memory, except NetCDF4. For NetCDF3 files the System. getProperties().get(file) contains a chunked byte array (byte[][]). An empty byte[][] array is set to tell an execute activity to save the nc file to memory as chunked byte array. The diagnostic file is also transferred in memory. Execute activities should always fall back on the files on disk when the System. getProperties().get(file) == null

Adapters that consistently use the Delf-PI.jar classes for I/O of XML data, configuration and diagnostics do not need to be modified to specifically make use of this feature, as the switching between in-memory or on-disk file transfers will be handled by Delf-PI.jar automatically. The feature has also been implemented for NetCDF files in NetCdfUtils.jar but this is considered experimental and requires further testing before using it in production.

Fortran/C models has to be recompiled to so/dll to allow in memory transfer. From the adapter this so/dll can be invoked with JNA (https://github.com/java-native-access/jna)

The NetCDF3 bytes can be passed to the so/dll use System.getProperties().get/put(file) to get/put the bytes of the netcdf file. The Fortran/C code can use the nc_open_mem/nc_create_mem to open/create the bytes

rootDir

Root directory for the external module. Other directories can be defined relative to this rootDir using predefined tags (see comment box below).

workDir

Working directory to be used by the external module. When started this directory will be the current directory.

exportDir

Directory to export data from Delft-FEWS to the external module. All Published Interface files will be written to this directory (unless overruled in naming the specific export files).

exportDataSetDir

Directory to export module datasets from Delft-FEWS to the external module. A module dataset is a ZIP file, which will be unzipped using this directory as the root directory. If the zip file contains full path information, this will be included as a tree of subdirectories under this directory.

updateExportDataSetDirOnlyOnChange

If set to 'true' datasets are only updated when they have changed. Change is detected by comparing the timestamp in the dataset info file with the modification time of the dataset configuration file. By default this value is 'false' and datasets are always updated. Since the 2018.02 the dataset info file is written to $REGION_HOME$/temp/moduleDataSetsCheckSums when the export dir is sub dir (any level) of the region home.

purgeExportDataSetDirOnUpdate

If a dataset has been detected as updated then this option allows user to purge content of existing moduledataset directory before new moduledataset is exported. By default the value is 'false' and all existing dataset content is overwritten.

exportIdMap

ID of the IdMap used to convert internal parameterId's and locationId's to external parameter and location Id's. See section on configuration for Mapping Id's units and flags.

exportUnitConversionsId

Id of UnitConversions to be used for export unit mapping

importDir

Directory to import result data from the external module to Delft-FEWS. All Published Interface files will be read from this directory (unless overruled in naming the specific export files).

importIdMap

ID of the IdMap used to convert external parameterId's and locationId's to internal parameter and location Id's. This may be defined to be the same as the import directory, but may also contain different mappings. See section on configuration for Mapping Id's units and flags.

importUnitConversionsId

Id of UnitConversions to be used for import unit mapping

dumpFileDir

Directory for writing dump files to. Dump Files are created when one of the execute activities fails. A dump file is a ZIP file which includes all the dumpDir directories defined. The dump file is created immediately on failure, meaning that all data and files are available as they are at the time of failure and can be used for analysis purposes. The ZIP file name is time stamped to indicate when it was created.

dumpDir

Directory to be included in the dump file. All contents of the directory will be zipped. Multiple dumpDir's may be defined.

NOTE: ensure that the dumpDir does not include the dumpFileDir. This creates a circular reference and may result in corrupted ZIP files.

diagnosticFile

File name and path of diagnostic files created in running modules. This file should be formatted using the Published Interface diagnostics file specification.

missVal

Optional specification of missing value identifier to be used in PI-XML exported to modules and imported from modules.

NOTE: it is assumed an external module uses the same missing value identification for both import and export data.

convertDatum

Optional Boolean flag to indicate level data is used and produced by the module at a global rather than a local datum. The convention in Delft-FEWS is that data is stored at a local datum. If set to true data in parameter groups supporting datum conversion will be converted on export to the global datum by adding the z coordinate of the location. (see definition of parameters and locations in Regional Configuration).

timeZone

Time zone with reference to UTC (equivalent to GMT) for all time dependent data communicated with the module. If not defined, UTC+0 (GMT) will be used. This time zone is used when importing pi files and the time zone is not available in the pi file.

timeZoneOffset

The offset of the time zone with reference to UTC (equivalent to GMT). Entries should define the number of hours (or fraction of hours) offset. (e.g. +01:00). This time zone is used when importing pi files and the time zone is not available in the pi file.

timeZoneName

Enumeration of supported time zones. See appendix B for list of supported time zones. This time zone is used when importing pi files and the time zone is not available in the pi file.

time0Format

The date time format of the %TIME0% variable and the %CURRENT_TIME% variable. This date time format is only used if there is no date time format specified in the arguments for the variable. yyyy = Year, MM = Month in year, dd = Day in month, HH = Hour in day (0-23), mm = Minute in hour, ss = Second in minute.

startDateTimeFormat

The date time format of the %START_DATE_TIME% variable. This date time format is only used if there is no date time format specified in the arguments for the variable. yyyy = Year, MM = Month in year, dd = Day in month, HH = Hour in day (0-23), mm = Minute in hour, ss = Second in minute. The %START_DATE_TIME% variable only works in an importStateActivity (since 2014.02) and inside a template file for an exportCustomFormatRunFileActivity (since 2013.02).

endDateTimeFormat

The date time format of the %END_DATE_TIME% variable. This date time format is only used if there is no date time format specified in the arguments for the variable. yyyy = Year, MM = Month in year, dd = Day in month, HH = Hour in day (0-23), mm = Minute in hour, ss = Second in minute. The %END_DATE_TIME% variable only works in an importStateActivity (since 2014.02) and inside a template file for an exportCustomFormatRunFileActivity (since 2013.02).

dateTimeFormat

Since stable build 2015.01. Definitions of date time formats that can be used as arguments for the %TIME0%, %START_DATE_TIME% and %END_DATE_TIME% variables, e.g.: %START_DATE_TIME(idOfADateFormat)% %END_DATE_TIME(idOfAnotherDateFormat)%. Date time formats specified as arguments overrule the startDateTimeFormat and endDateTimeFormat specified above. The %START_DATE_TIME% and %END_DATE_TIME% variables only work in an importStateActivity (since 2014.02) and inside a template file for an exportCustomFormatRunFileActivity (since 2013.02).

- **id**: Identifier for this date time format. Use this identifier to refer to this date time format in the argument of a variable.
- **dateTimePattern**: Pattern for this date time format. This can be e.g. "yyyy-MM-dd HH:mm:ss". yyyy = Year, MM = Month in year, dd = Day in month, HH = Hour in day (0-23), mm = Minute in hour, ss = Second in minute.

ensembleMemberCount

Defines if ensembles are read from or written to a number of sub directories.

modelTimeStep

The model time step adjusts the end time when necessary of the run period for this model and for all models/modules that will run aft05GeneralAdapterModule-importShapeFileActivityer this model in the same task run.

Burn-In Profile

Burn-in profile for cold state starts. Used to replace first part of a timeseries.

For time series with matching parameter-location ids, the first value is replaced by the initialValue. Element length defines the length of timeseries beginning that is to be replaced using linear interpolation.

```
<burnInProfile>
   <length multiplier="4" unit="hour"/>
   <timeSeries>
      <parameterId>H.obs</parameterId>
      <locationSetId>locationInitialValueAttSet</locationSetId>
      <initialValueAttributeId>initialValue</initialValueAttributeId>
   </timeSeries>
</burnInProfile>
```

length

Length of time series beginning that is to be replaced.

timeSeries

parameterId and locationId or locationSetId => to match timeSeries this burn in profile should apply to

initialValue or initialValueAttributeId => to specify the start value hardcoded or via a location attribute

Startup Activities

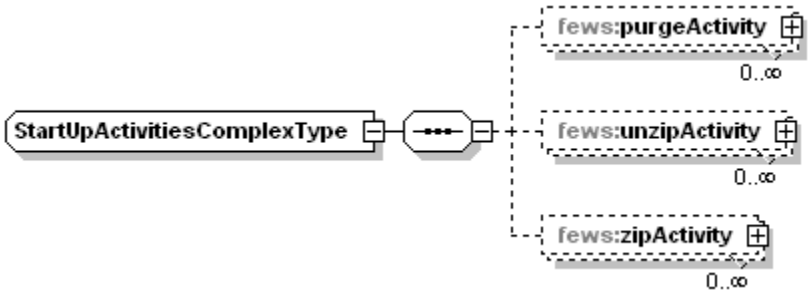


Figure 68 Elements of the startUpActivities section of the General Adapter configuration.

purgeActivity

Root element of a purge activity used to delete files from previous runs. Multiple purge activities may be defined.

filter

Filter specifying files to be removed. Wild cards may be used.

Deleting a whole directory tree can be achieved by defining the directory path in the filter without any file name wild cards. eg: %ROOT_DIR%/exportDir /purgeDirectory,

When using a path with a filename wild card, subdirectories will by default not be removed, an optional element, <includeSubdirectories> can be included to also delete subdirectories, including their content.

Please note that it is possible to delete files outside the general adapter root folder, so it is recommended to always start the filter path with %ROOT_DIR% to prevent this from happening accidentally.

Minimum file age can optionally be used to only purge files over a certain age.

Example (note the use of tags to define the root directory name):

```
        <startUpActivities>
                <purgeActivity>
                        <filter>%ROOT_DIR%/temp</filter>
                </purgeActivity>
                <purgeActivity>
                        <filter>%ROOT_DIR%/logs/*.*</filter>
            <minimumFileAge unit="day" multiplier="2"/>
        </purgeActivity>
        <purgeActivity>
                        <includeSubdirectories>true</includeSubdirectories>
            <filter>%ROOT_DIR%/importDir/*.*</filter>
        </purgeActivity>
    </startUpActivities>
```

unzipActivity

Root element of an unzip activity used to unpack a zip file and put the contained files in the directory of choice. Multiple unzip activities may be defined.

Each activity has the following elements:

- description - optional description of the activity (for documentation only)
- sourceZipFile - the name of the zip file to be unzipped
- destinationDir - the name of the directory where the files will be put

zipActivity

Root element of a zip activity used to pack all files and subdirectories of an indicated directory to a zip file for later use/inspection. One or more source file pattern (including * and ? wildcards) can be specified to include only a subset of the files in the source directory. Multiple zip activities may be defined. The file name may include environment variables, as well as tags defined in the general adapter or on the global.properties. See the environment variables section for a list of internal variables.

Each activity has the following elements:

- description - optional description of the activity (for documentation only)

- sourceDir - the name of the directory containing the files to be zipped
- sourcePattern - (Since 2019.02) a file name pattern to select a subset of the files in source directory. Since 2022.01 tags between %% are recognized
- destinationZipFile - the name of the zip file to be created

Example:

```
<startupActivities>
    <unzipActivity>
        <sourceZipFile>extra_files.zip</sourceZipFile>
        <destinationDir>%ROOT_DIR%/work</destinationDir>
    </unzipActivity>
</startupActivities>
...
<shutdownActivities>
    <zipActivity>
        <sourceDir>%ROOT_DIR%/work</sourceDir>
        <sourcePattern>input/timeseries_input.*</sourcePattern>
        <sourcePattern>diagnostics/diag.xml</sourcePattern>
        <destinationZipFile>%ROOT_DIR%/inspection/%CURRENT_TIME%_%TIME0%_saved.zip</destinationZipFile>
    </zipActivity>
</shutdownActivities>
```

Export Activities

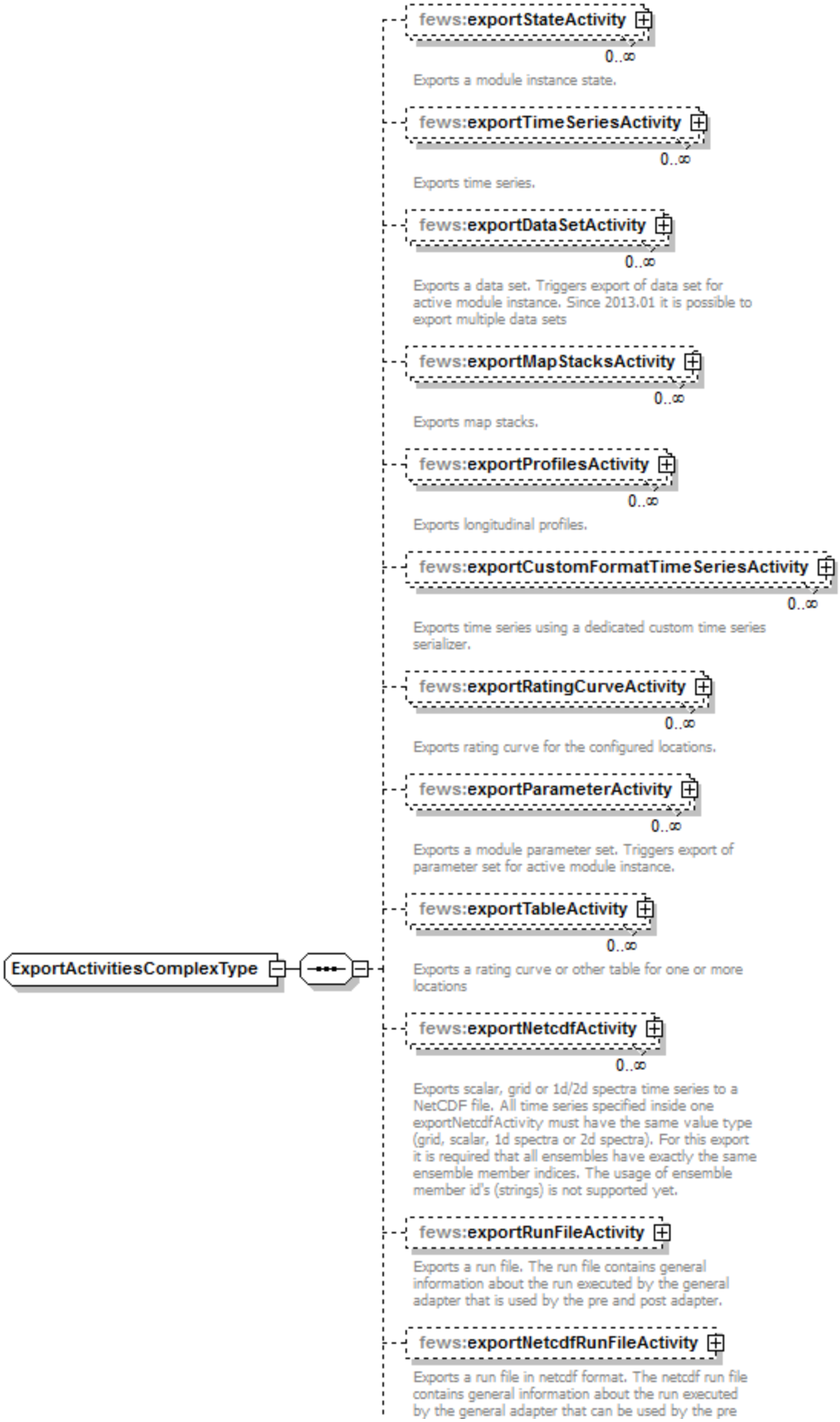

ExportActivitiesComplexType
fews:exportStateActivity
0..∞
Exports a module instance state.
fews:exportTimeSeriesActivity
0..∞
Exports time series.
fews:exportDataSetActivity
0..∞
Exports a data set. Triggers export of data set for active module instance. Since 2013.01 it is possible to export multiple data sets
fews:exportMapStacksActivity
0..∞
Exports map stacks.
fews:exportProfilesActivity
0..∞
Exports longitudinal profiles.
fews:exportCustomFormatTimeSeriesActivity
0..∞
Exports time series using a dedicated custom time series serializer.
fews:exportRatingCurveActivity
0..∞
Exports rating curve for the configured locations.
fews:exportParameterActivity
0..∞
Exports a module parameter set. Triggers export of parameter set for active module instance.
fews:exportTableActivity
0..∞
Exports a rating curve or other table for one or more locations
fews:exportNetcdfActivity
0..∞
Exports scalar, grid or 1d/2d spectra time series to a NetCDF file. All time series specified inside one exportNetcdfActivity must have the same value type (grid, scalar, 1d spectra or 2d spectra). For this export it is required that all ensembles have exactly the same ensemble member indices. The usage of ensemble member id's (strings) is not supported yet.
fews:exportRunFileActivity
Exports a run file. The run file contains general information about the run executed by the general adapter that is used by the pre and post adapter.
fews:exportNetcdfRunFileActivity
Exports a run file in netcdf format. The netcdf run file contains general information about the run executed by the general adapter that can be used by the pre

Figure 69 Elements of the ExportActivity section

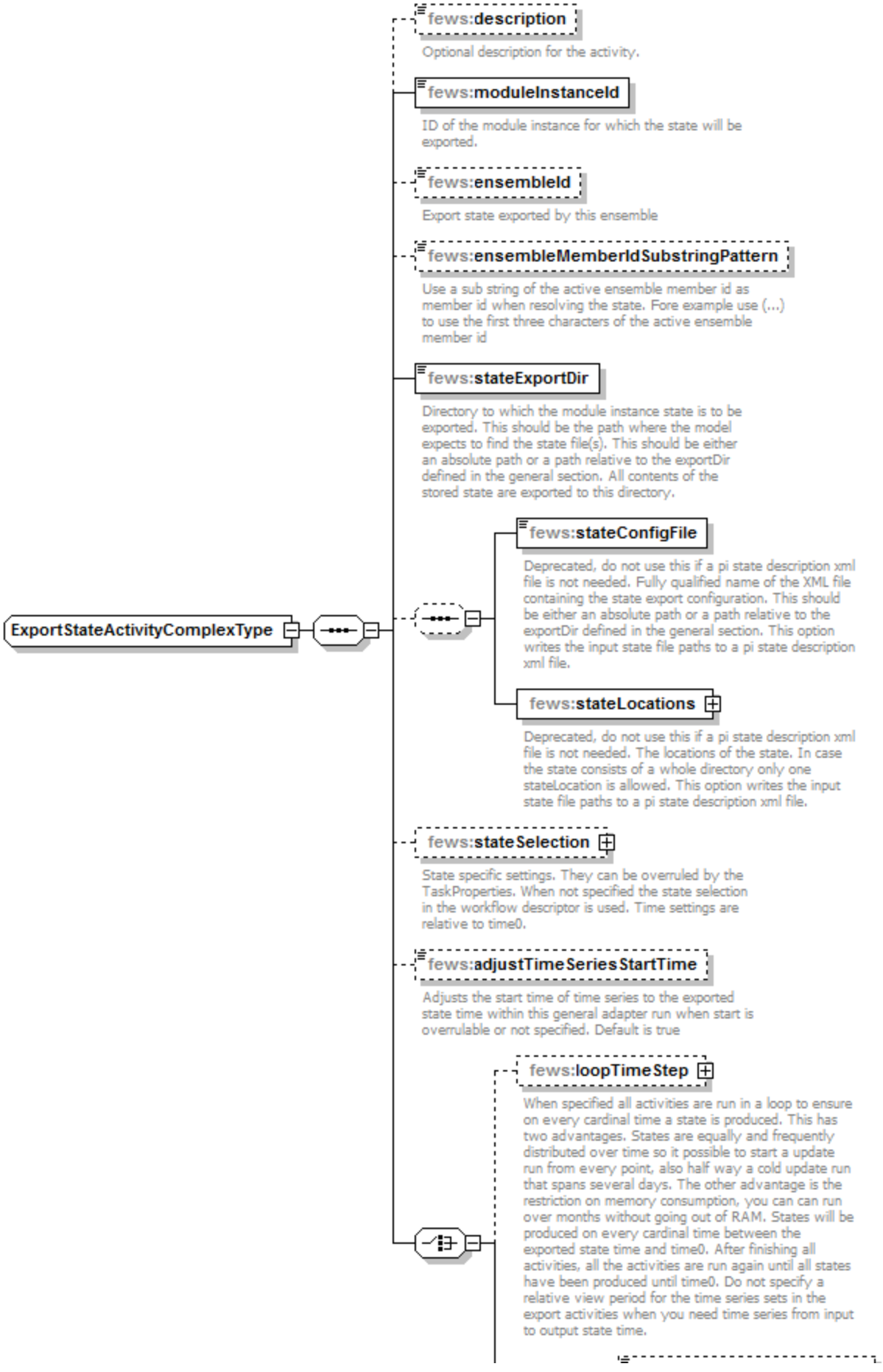
Export activities are defined to allow exporting various data objects from Delft-FEWS to the external modules. The list of objects that can be exported (see figure above) includes;

- *exportStateActivity* to export module states
- *exportTimeSeriesActivity* to export time series for scalar or polygon time series
- *exportDataSetActivity* to export module datasets
- *exportMapStacksActivity* to export time series for grid time series
- *exportProfilesActivity* to export time series for longitudinal profile time series
- *exportCustomFormatTimeSeriesActivity* to export time series in a custom format using a third party serializer
- *exportRatingCurveActivity* to export rating curves
- *exportPiTablesActivity* to export time series with a single shared domainParameter at once to pi_tables.xml
- *exportParameterActivity* to export module parameters
- *exportTableActivity* to export table (e.g. rating table)
- *exportNetcdfActivity* to export scalar or grid time series in Netcdf format
- *exportRunFileActivity* to export a run file (The run file contains general information that is used by the pre and post adapter)
- *exportNetcdfRunFileActivity* to export a netcdf run file (The netcdf run file contains general information that can be used by the pre and post adapter)
- *exportCustomFormatRunFileActivity* to export a custom format run info file that can be used by the model
- *exportAreaSelectionActivity* exports bitmasks for the model adapter with a polygon embedded in the TaskProperties as input (is currently only used within NGMS)
- *exportLocationAreaActivity* exports bitmasks for the model adapter with location ids from the TaskProperties as input (is currently only used within NGMS)

For most types of exportActivity, multiple entries may exist.

NB. See also the exportPlaceholderFile option in the several importActivities, which is another form of preparing input for the module adapter.

exportStateActivity


ExportStateActivityComplexType
fews:description
Optional description for the activity.
fews:moduleInstanceId
ID of the module instance for which the state will be exported.
fews:ensembleId
Export state exported by this ensemble
fews:ensembleMemberIdSubstringPattern
Use a sub string of the active ensemble member id as member id when resolving the state. Fore example use (...) to use the first three characters of the active ensemble member id
fews:stateExportDir
Directory to which the module instance state is to be exported. This should be the path where the model expects to find the state file(s). This should be either an absolute path or a path relative to the exportDir defined in the general section. All contents of the stored state are exported to this directory.
fews:stateConfigFile
Deprecated, do not use this if a pi state description xml file is not needed. Fully qualified name of the XML file containing the state export configuration. This should be either an absolute path or a path relative to the exportDir defined in the general section. This option writes the input state file paths to a pi state description xml file.
fews:stateLocations
Deprecated, do not use this if a pi state description xml file is not needed. The locations of the state. In case the state consists of a whole directory only one stateLocation is allowed. This option writes the input state file paths to a pi state description xml file.
fews:stateSelection
State specific settings. They can be overruled by the TaskProperties. When not specified the state selection in the workflow descriptor is used. Time settings are relative to time0.
fews:adjustTimeSeriesStartTime
Adjusts the start time of time series to the exported state time within this general adapter run when start is overrulable or not specified. Default is true
fews:loopTimeStep
When specified all activities are run in a loop to ensure on every cardinal time a state is produced. This has two advantages. States are equally and frequently distributed over time so it possible to start a update run from every point, also half way a cold update run that spans several days. The other advantage is the restriction on memory consumption, you can can run over months without going out of RAM. States will be produced on every cardinal time between the exported state time and time0. After finishing all activities, all the activities are run again until all states have been produced until time0. Do not specify a relative view period for the time series sets in the export activities when you need time series from input to output state time.

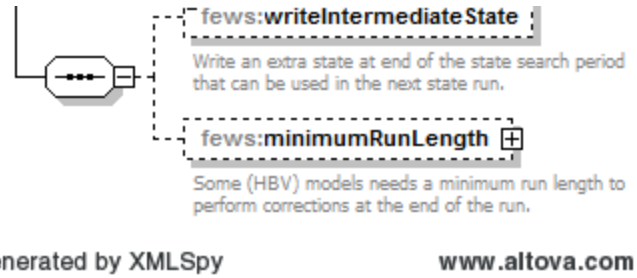


Figure 70 Elements of the ExportStateActivity section.

description

Optional description for the export states configuration. Used for reference purposes only.

moduleInstanceId

Id of the moduleInstance that has written the state to be exported. Generally this will be the same as the Id of the current instance of the General Adapter. This can also be the ID of another instance of the General Adapter. The latter is the case when using a state in a forecast run that has been written in an historical run.

ensembleId

Export state exported by this ensemble.

ensembleMemberIdSubstringPattern

Use a sub string of the active ensemble member id as member id when resolving the state. Fore example use (...) to use the first three characters of the active ensemble member id.

EnsembleMemberId

Instead of ***ensembleMemberIdSubstringPattern*** (see above), EnsembleMemberId could also be used to identify an ensemble member to resolve the state in a forecast run.

stateExportDir

Directory to which the module instance state is to be exported. This should be the path where the model expects to find the state file(s). This should be either an absolute path or a path relative to the exportDir defined in the general section. All contents of the stored state are exported to this directory.

stateConfigFile

Optional. Deprecated, do not use this if a pi state description xml file is not needed. Name (and location) of the PI-XML file describing the states. If the directory location is not explicitly specified the file will be written in the exportDir defined in the general section. This should be either an absolute path or a path relative to the exportDir defined in the general section. This option writes the input state file paths to a pi state description xml file.

stateLocations

Optional. Deprecated, do not use this if a pi state description xml file is not needed. Root element for the description of the state. Both a read location and a write location will need to be defined. This allows the name of the file(s)/directory to be different on read and write. Multiple locations may be defined, but these must all be of the same type. This option writes the input state file paths to a pi state description xml file.

- *Attributes* type: indication of type of state to be imported. This may either be "directory" or "file". Note that multiple locations are supported only if type is "file".
- *stateLocation* - Root element of a state location
- *readLocation* - Location where the external module will read the state. This is the location (and name of file/directory) where the General Adapter writes the state. This should be either an absolute path or a path relative to the stateExportDir defined in this activity.
- *writeLocation* - Location where the external module is expected to write the state. This is the location (and name of file/directory) where the General Adapter expects to read the state. This should be an absolute path.

```
<stateLocations type="file">
        <stateLocation>
                <readLocation>state.inp</readLocation>
                <writeLocation>state.out</writeLocation>
        </stateLocation>
</stateLocations>
```

stateSelection

Root element to specify how a state to be exported to the external module is to be selected. Two main groups are available, cold states and warm states. Only one of these types can be specified. Note that if a warm state selection is specified and an appropriate warm state cannot be found, a cold state will be exported by default.

- *coldState* - Root element for defining the stateSelection method to always export a cold state.
- *groupId* - Id of the group of cold states to be used. This must be a groupId as defined in the ColdModuleInstanceStateGroups configuration (see Regional Configuration).
- *coldState:startDate* - Definition of the start date of the external module run when using the cold state. This startDate is specified relative to the start time of the forecast run. A positive startDate means it is before the start time of the forecast run.
- *coldState:fixedStartTime* - (Since 2012_02) the start date can be configured as an fixed time
- *warmState* - Root element for defining the stateSelection method to search for the most suitable warm state.
- *stateSearchPeriod* - Definition of the search period to be used in selecting a warm state. The database will return the most recent suitable warm state found within this search period.
- *searchForTransientStates* - Default is true, and the prefered option for forecast modules. An historical state generating module however should preferably not start with a transient state, which is a state imported after the state search period. Transient states can be given a much lower expiry time (see option below) to reduce the size of the database. See this memo for more background information.
- *transientStateExpiryTime* - Expiry time of the transient states, which are states with a time after the warm state search period
- *coldStateTime* - Definition of the start time to use for a cold state if a suitable state is not found within the warm state search period.

```
<stateSelection>
        <warmState>
          <stateSearchPeriod unit="hour" start="-48" end="0"/>
                  <searchForTransientStates>false</searchForTransientStates>
                  <transientStateExpiryTime multiplier="1" unit="day"/>
                    <coldStateTime unit="hour" value="-48"/>
        </warmState>
        <overrulingColdStateModuleInstanceId>ExportStateActivityOverrulingColdState<
/overrulingColdStateModuleInstanceId>
</stateSelection>
```

The stateSearch period for a warm state selection must be carefully configured. If the time span between the end of the state search period and the start of forecast is longer than the frequency of forecasts then the span between the coldStateTime and the start of the state search period must be at least as long. The insertColdState option must be set to true. If this is not done, then the warm state of the first state run will not be found by the next state run, resulting in subsequent forecasts using alternating warm states.

overrulingColdStateModuleInstanceId

Different module instance id used for finding cold state (when warm state is configured but can't be found, or always when cold state is configured)

backupWarmStateModuleInstanceId

Different module instance id used for finding warm state, when original warm state can't be found

adjustTimeSeriesStartTime

Optional. Adjusts the start time of time series to the exported state time within this general adapter run when start is overrulable or not specified. Default is true.

loopTimeStep

When specified, all activities are run in a loop to ensure that a state is produced on every cardinal time step between the time of the exported state and T0. This has two advantages:

- states are distributed over time equally and frequently. It is possible to start an update run from every point, also half way of a cold update run that spans several days.
- restriction of memory consumption. You can run an update run over months without going out of RAM.

Do not specify a relative view period for all time series sets in the export activity

writeIntermediateState

When specified, an extra state is written at the end of the state search period. Note that the run is then split in two. E.g my state search period is -10 to -4 days, then there are two update runs, one from the time where a state was found to -4 and one from -4 to T0. A state is written at the end of both runs (T0 and T0 - 4days). You can additionally define a minimum run length. This is necessary for some runs that need a minimum run length for e.g. PT updating. The run is then only split in two if both runs can be run over the minimum run length. If not, there is only one run and the state is written to the end of this run (T0), no intermediate state is written.

minimumRunLength

Optional. Some (HBV) models need a minimum run length to perform corrections at the end of the run.

see example configuration below and this figure

Configuration example

New example that does not use a pi state description xml file. In this case, if the model adapter needs to know the input state file paths, then these can be read from the netcdf version of the run info file (which can be exported using the new ExportNetcdfRunFileActivity described below):

```
<exportStateActivity>
        <moduleInstanceId>HBV_AareBrugg_Hist</moduleInstanceId>
        <stateExportDir>%ROOT_DIR%/FEWS/states</stateExportDir>
        <stateSelection>
                <warmState>
                        <stateSearchPeriod unit="hour" start="-240" end="-96"/>
                </warmState>
        </stateSelection>
        <writeIntermediateState>true</writeIntermediateState>
        <minimumRunLength unit="day" multiplier="4"/>
</exportStateActivity>
```

Old example that writes the input state file paths to a pi state description xml file. Do not use this if a pi state description xml file is not needed:

```
<exportStateActivity>
        <moduleInstanceId>HBV_AareBrugg_Hist</moduleInstanceId>
        <stateExportDir>%ROOT_DIR%/FEWS/states</stateExportDir>
        <stateConfigFile>%ROOT_DIR%/FEWS/states/states.xml</stateConfigFile>
        <stateLocations type="file">
                <stateLocation>
                        <readLocation>HBV_States.zip</readLocation>
                        <writeLocation>HBV_States.zip</writeLocation>
                </stateLocation>
        </stateLocations>
        <stateSelection>
                <warmState>
                        <stateSearchPeriod unit="hour" start="-240" end="-96"/>
                </warmState>
        </stateSelection>
        <writeIntermediateState>true</writeIntermediateState>
        <minimumRunLength unit="day" multiplier="4"/>
</exportStateActivity>
```

exportTimeSeriesActivity

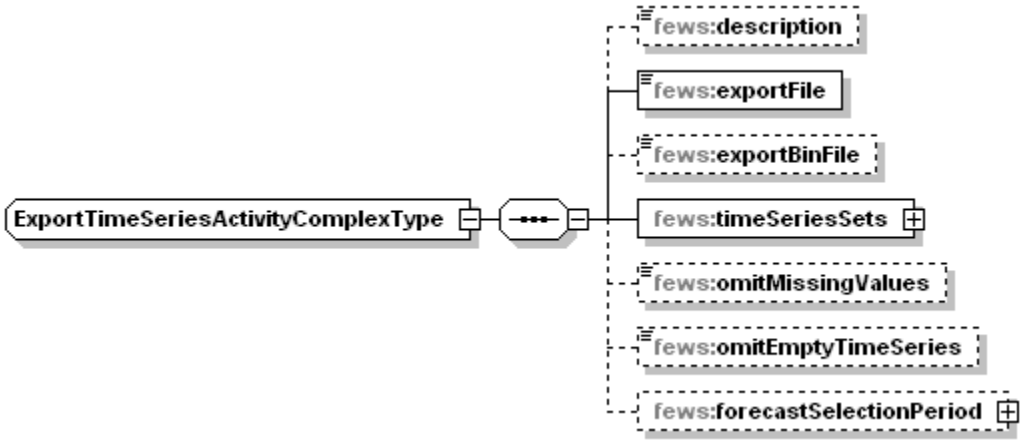


Figure 71 Elements of the exportTimeSeries section

description

Optional description of the timeSeries export configuration.

exportFile

Name (and location) of the PI-XML file with exported time series. If the directory location is not explicitly specified the file will be written in the exportDir defined in the general section.

exportBinFile

When true the events in the PI time series file are written to a binary file instead of the xml file. The written xml file will only contain the time series headers and optionally a time zone. The binary file has the same name as the xml file only the extension is "bin" instead of "xml". During PI time series import the bin file is automatically read when available. The byte order in the bin file is always Intel x86.

ignoreRunPeriod

When true the run period, written in the pi run file, will not be extended.

includeThresholds

When true any thresholds for the exported timeseries will be written in the timeserie headers

timeSerieSets

Root element for defining timeSerieSets to be exported.

timeSerieSets: timeSerieSet

TimeSeriesSets to be exported. These may contain either a (list of) locations or a locationSet. Multiple entries may be defined.

omitMissingValues

Includes missing values in the export file or leave them out. In addition, unreliable data will be exported as missing values because unreliable data cannot be used in calculations.

Note: this means that the original values that have been set as unrealiable will be converted to missing upon export. If the original values are needed (in for example a model that does the data validation instead of the FEWS ValidationRuleSets), use the regular exportModule.

checkMissingValues

Option to check the timeseries for missing values. This option works in combination with the Parameter element 'allowMissing'. If that allowMissing is set to false, you can check for missing values. So if checkMissingValues = true and allowMissing=false, an error will occur in case there are missing values to be exported. As such, you can prevent to run a model with missing values and let the General Adapter stop with an error.

omitEmptyTimeSeries

When true, a series is not exported when the time series is empty (or when omitMissingValues = true, when the time series is empty after removing the missing values.)

omitEmptyFiles

When true, the file is not exported when when file does not contain any time series (after omitMissingValues and omitEmptyTimeSeries)

forecastSelectionPeriod

Can be used to select all approved forecasts with a forecast start time lying within this period

ignoreNonExistingLocationSets

Option to skip time series sets if the location set does not exist. Useful when module or workflow is run in loop with tags being translated. Default is 'false'.

exportMapStacksActivity

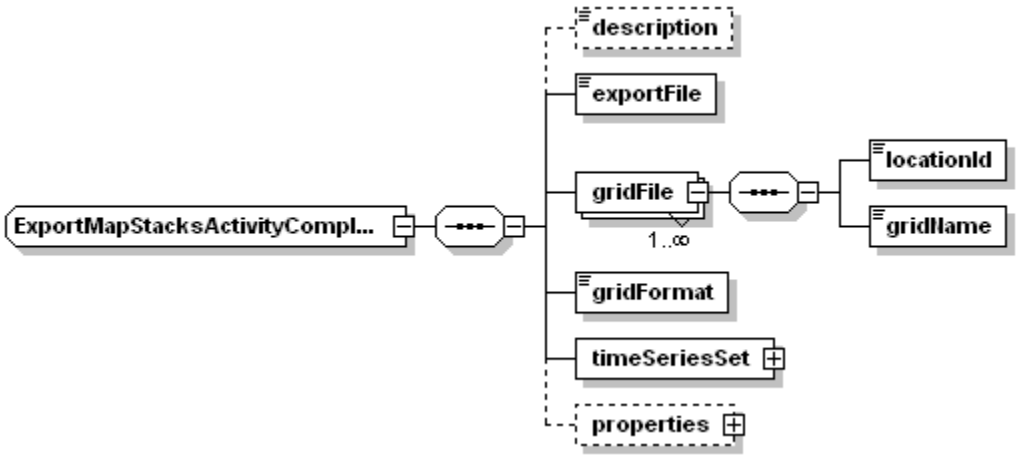


Figure 72 Elements of the ExportMapStacksActivity.

description

Optional description of the timeSeries export configuration.

exportFile

Name (and location) of the PI-XML file describing the map stack of the exported grid time series. If the directory location is not explicitly specified the file will be written in the exportDir defined in the general section.

gridFile

Root element for defining file name of grid to be exported

locationId

LocationId for grid to be exported.

gridName

Name of the files for the grid to be exported. For grid files where each time slice is stored in a different file, this name is the prefix for the full file name. The final file name is created using an index of files exported (e.g the file name for the 4^th^ time step is grid00000.004).

gridFormat

Format of the exported grid. Enumeration of options include;

- *asc* : for exporting to ARC-INFO ASCII grid format
- *pcrgrid* : for exporting to PCRaster native grid file format
- *usgs* : for exporting to USGS DEM format (BIL)

timeSerieSet

TimeSeriesSets to be exported. These should contain only one locationId. For exporting multiple grids, multiple exportMapStack activities should be defined.

Note: when exporting an equidistant timeseriesset, the assumption is that the model adapter uses the start and the timestep to derive the timestamp of the event. Only for non-equidistant timeseriesset, the mapstacks.xml will hold events.

exportProfilesActivity

Configuration of the exportProfiles activity is identical to the exportTimeSeries Activity.

When using a state search period in the General Adapter configuration, the timeSeriesSets to be exported should not contain a start time in the Relative View Period. The start time of the run is determined by the state found. A startOverrullable item should also not be included

exportDataSetActvity

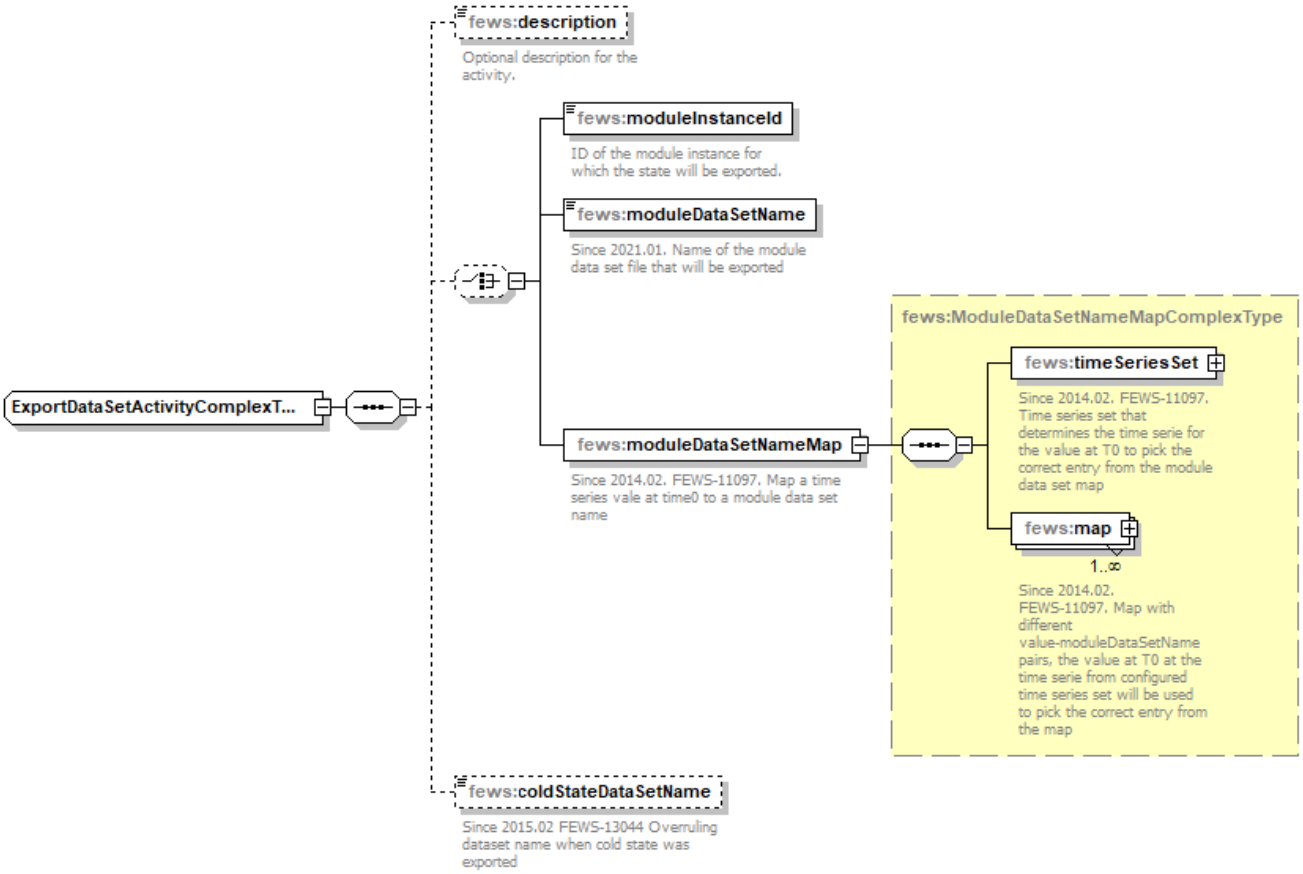


Figure 73 Elements of the exportDataSets section

description

Optional description of the module dataset export configuration.

moduleInstanceId

Optional reference to the moduleInstanceId of the moduleDataSet to be exported. If not defined the moduleInstanceId of the current module instance is taken as a default (see section on Module Datasets and Parameters).

moduleDataSetName

Optional reference to the file name (without zip extension) of a moduleDataSet to be exported. When using this feature, the moduleDataSet does not not need to be registered with a moduleInstanceId. Aligns with the whatif-functionality where configfile-properties can be set based on patterns in the moduledataset name.

moduleDataSetNameMap

Optional reference to the choose the moduleDataSet to be exported based on a mapping between a timeseries the value at T0 and the associated moduleDataSet file name. Since the timeseries value may be derived from a location attribute, the selection can be determinted via a location attribute modifier.

coldStateDataSetName

Optional reference to overrule a 'normal' moduleDateSet export by a different export in case of a cold state start

See also: Configuration guide - 01 Module Datasets

exportParameterActivity

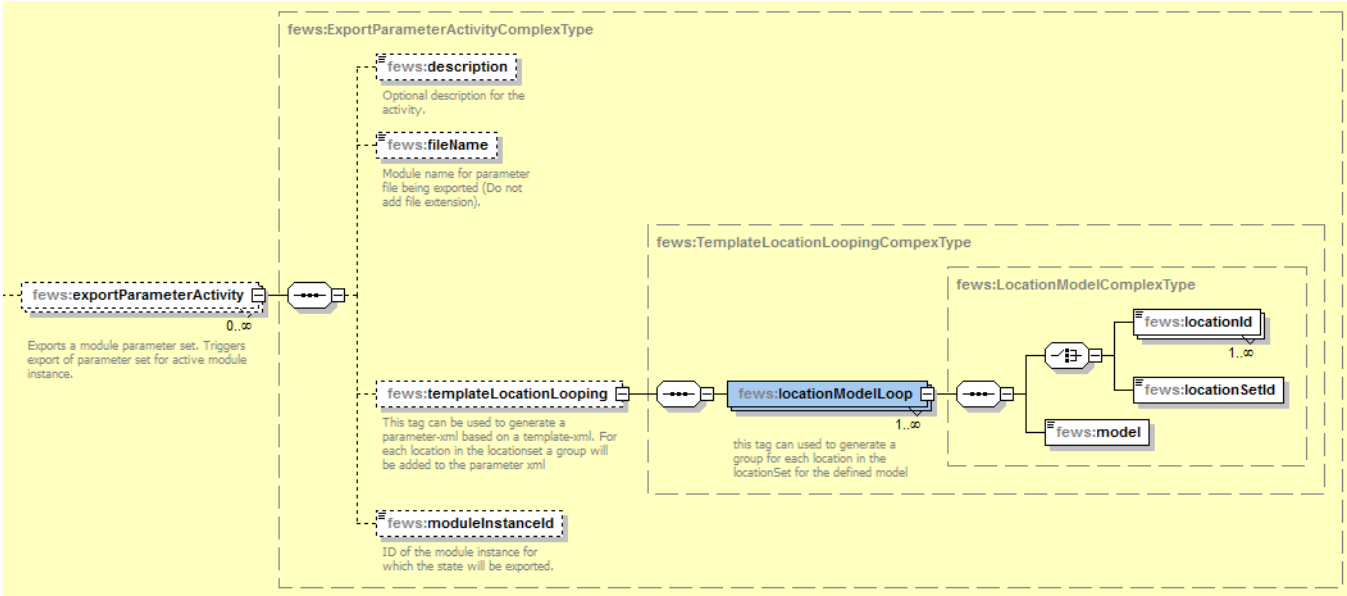


Figure 74 Elements of the exportParameter section

description

Optional description of the module parameter configuration.

moduleInstanceId

Optional reference to the moduleInstanceId of the moduleParameter to be exported. If not defined the moduleInstanceId of the current module instance is taken as a default (see section on Module Datasets and Parameters) . The parameterfile should be stored in the folder config\ModuleParFiles.

fileName

Name (and location) of the PI-XML file with exported parameters. If the directory location is not explicitly specified the file will be written in the exportDir defined in the general section.

templateLocationLooping

It is possible to have the parameterfile filled with location attributes. When exporting the parameter file, FEWS will loop over all the locations in the given locationSet, or through the given locationIds, and add the configured attributes to the parameterfile. For an example of a template parameterfile, see 02 Module Parameters. Notice that the element "model" references to the model in the template parameter file.

exportLocationAttributesCsvActivity (since 2020.01)

Exports attributes of a location set or a single location to csv format, including multi-value attributes. Skips by default when encountering empty or non-existing location sets.

Can export empty files with header (i.e. column names) only when using optional element

```
<exportEmptyHeaderFile>true<exportEmptyHeaderFile>
```

The second optional element is

```
<locationIdColumn>
```

Example config

```
<exportLocationAttributesCsvActivity>
   <exportFile>ExportLocationAttributes.csv</exportFile>
   <locationSetId>ExportLocationAttributesCsv</locationSetId>
   <exportEmptyHeaderFile>true</exportEmptyHeaderFile>
   <locationIdColumn columnName="Id"/>
   <attributeColumn columnName="A" attributeId="A"/>
   <attributeColumn columnName="B" attributeId="B"/>
   <attributeColumn columnName="C" attributeId="C"/>
</exportLocationAttributesCsvActivity>
<exportLocationAttributesCsvActivity>
   <exportFile>ExportLocationAttributesNoId.csv</exportFile>
   <locationSetId>ExportLocationAttributesCsv</locationSetId>
   <attributeColumn columnName="A" attributeId="A"/>
   <attributeColumn columnName="C" attributeId="C"/>
</exportLocationAttributesCsvActivity>
```

Example exported ExportLocationAttributes.csv

```
Id,A,B,C
loc1,A1,B1,C1
loc2,A2,B2,
loc3,,B3,C3
loc4,A4,B4,C4
loc5,A5,,C5
```

exportTableActivity

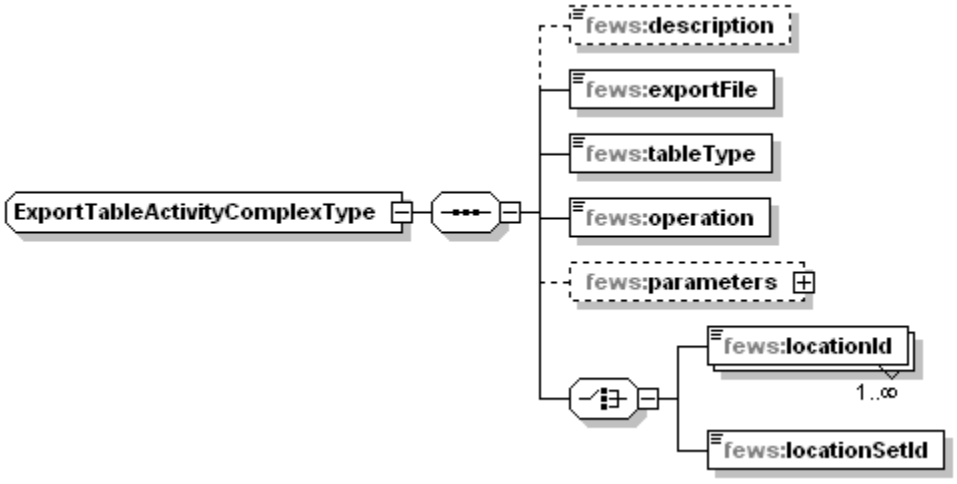


Figure 1 Elements of the ExportTableActivity configuration

description

Optional description of the ExportTableActivity configuration.

exportFile

File to which the table will be exported. This file is always placed in exportDir.

tableType

Type of table to be exported. Currently enumeration must be "ratingCurve"

operation

ID of the table to be exported.

parameters

Parameters for the convertEquation operation. Must include minimumLevel, maximumLevel and stepSize

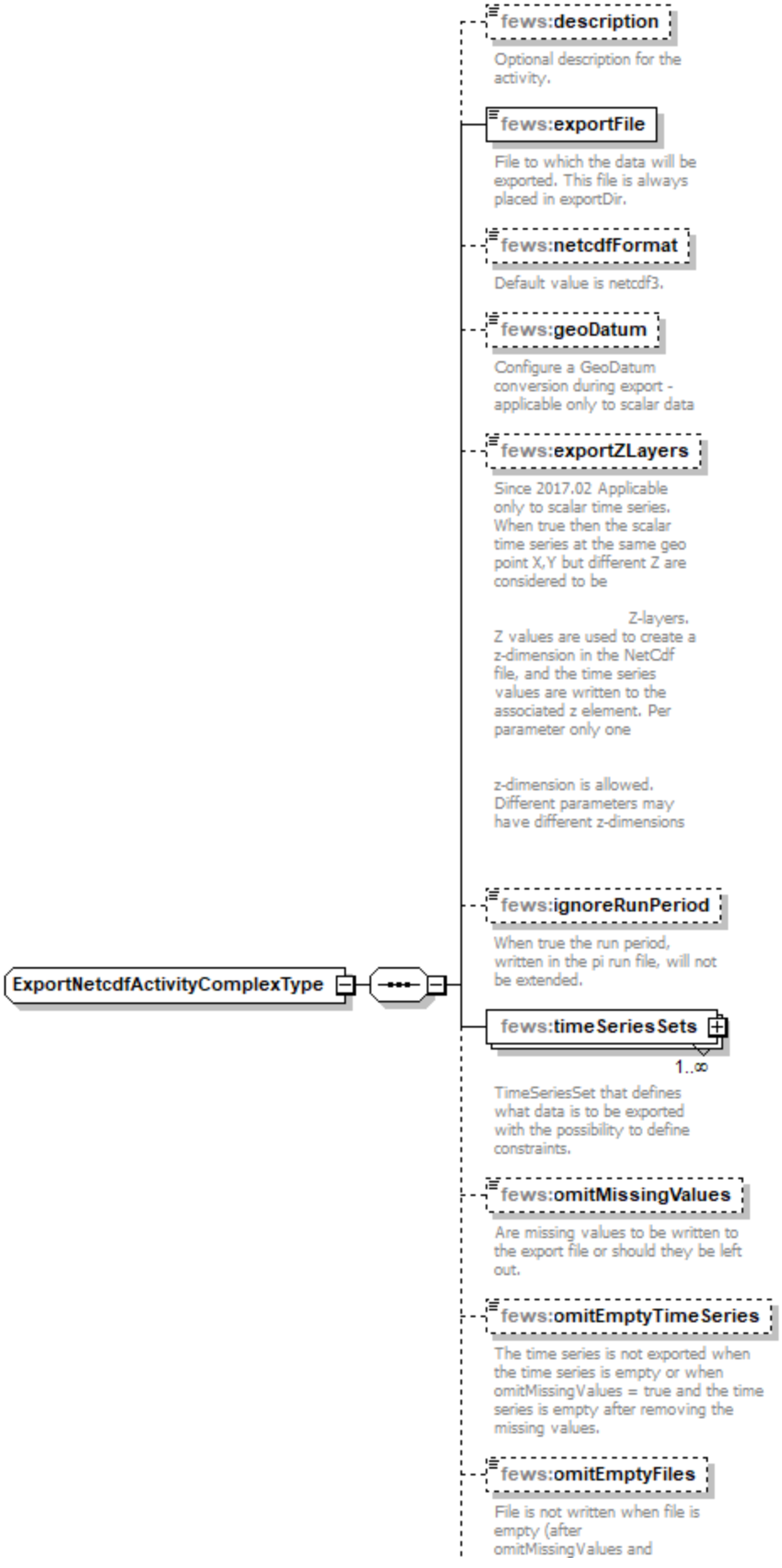
locationId/locationSetId

location id to select the rating curve

exportNetcdfActivity

Exports scalar, grid or 1d/2d spectra time series to a NetCDF file. All time series specified inside one exportNetcdfActivity must have the same value type (grid, scalar, 1d spectra or 2d spectra). For this export it is required that all ensembles have exactly the same ensemble member indices. The usage of ensemble member id's (strings) is not supported yet.

Compacting netCDF variables with scale factor and offset into a smaller integer typed variable is not available in contrast to the archive export (applied whenever possible) and time series export (applied when possible if property tryCompactingNetCDFData is set to true).


ExportNetcdfActivityComplexType
fews:description
Optional description for the activity.
fews:exportFile
File to which the data will be exported. This file is always placed in exportDir.
fews:netcdfFormat
Default value is netcdf3.
fews:geoDatum
Configure a GeoDatum conversion during export - applicable only to scalar data
fews:exportZLayers
Since 2017.02 Applicable only to scalar time series. When true then the scalar time series at the same geo point X,Y but different Z are considered to be
Z-layers. Z values are used to create a z-dimension in the NetCdf file, and the time series values are written to the associated z element. Per parameter only one
z-dimension is allowed. Different parameters may have different z-dimensions
fews:ignoreRunPeriod
When true the run period, written in the pi run file, will not be extended.
fews:timeSeriesSets
1..∞
TimeSeriesSet that defines what data is to be exported with the possibility to define constraints.
fews:omitMissingValues
Are missing values to be written to the export file or should they be left out.
fews:omitEmptyTimeSeries
The time series is not exported when the time series is empty or when omitMissingValues = true and the time series is empty after removing the missing values.
fews:omitEmptyFiles
File is not written when file is empty (after omitMissingValues and

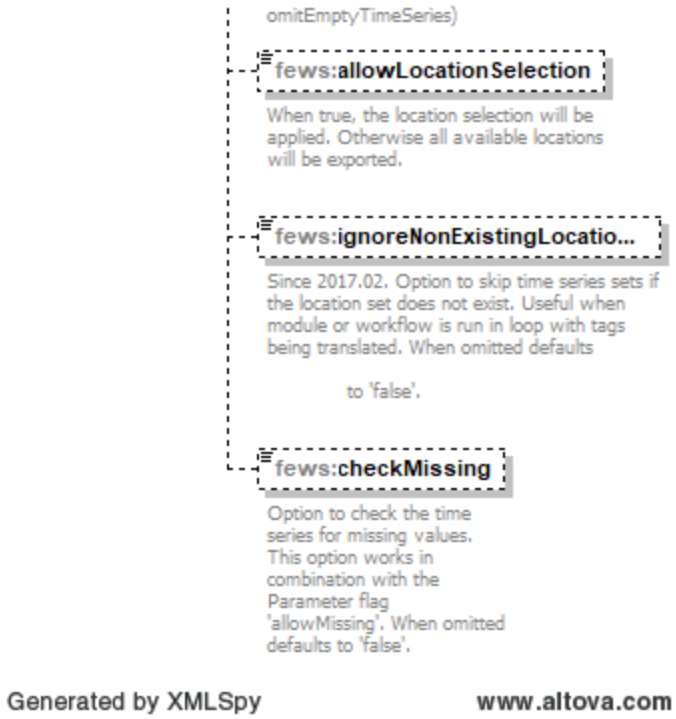


Figure 2 Elements of the ExportNetcdfActivity configuration

description

Optional description of the ExportNetcdfActivity configuration.

exportFile

File to which the data will be exported. This file is always placed in exportDir.

netcdfFormat

Optional. Default value is netcdf3. Currently supported other option is netcdf4.

metadata

Since 2023.01 metadata can be added to the netcdf global attributes, just like the implementation for normal netcdf exports.

geoDatum

Optional. To configure reprojection of location coordinates you can specify the required output geodatum here. This option is available only for scalar datasets.

writeRealizationDimension

Applicable only to scalar and gridded time series with exactly one ensemble member. When false then this ensemble member is not written to the file and parameter in the file does not have realization dimension. A config error will be given when set to false and multiple ensemble members are found in the timeseries.

timeSeriesSets

TimeSeriesSet that defines what data is to be exported.

omitMissingValues

Include missing values in the export file or leave them out.

omitEmptyTimeSeries

The time series is not exported when the time series is empty or when; omitMissingValues = true and the time series is empty after removing the missing values.

omitEmptyFiles

When true, the file is not exported when when file does not contain any time series (after omitMissingValues and omitEmptyTimeSeries)

ignoreNonExistingLocationSets

Option to skip time series sets if the location set does not exist. Useful when module or workflow is run in loop with tags being translated. Default is 'false'.

Example:

```
<exportNetcdfActivity>
    <exportFile>timeseries.nc</exportFile>
    <netcdfFormat>netcdf4</netcdfFormat>
    <timeSeriesSets>
        <timeSeriesSet>
            <moduleInstanceId>GeneralAdapterRun</moduleInstanceId>
            <valueType>scalar</valueType>
            <parameterId>WaterLevel</parameterId>
            <locationId>H-2001</locationId>
            <timeSeriesType>external historical</timeSeriesType>
            <timeStep unit="minute" divider="1" multiplier="15"/>
            <relativeViewPeriod unit="hour" start="0" end="12"/>
            <readWriteMode>read only</readWriteMode>
            <ensembleId>prognose</ensembleId>
        </timeSeriesSet>
    </timeSeriesSets>
</exportNetcdfActivity>
```

exportRunFileActivity

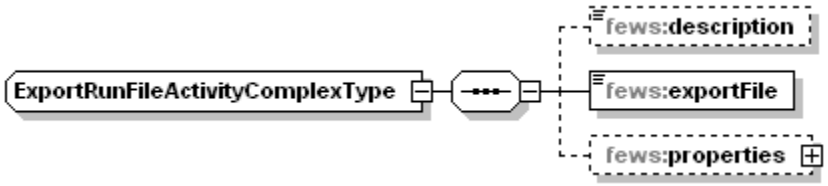


Figure 3 Elements of the ExportRunFileActivity configuration

description

Optional description of the ExportRunFileActivity configuration.

exportFile

File to which the data will be exported. This file is always placed in exportDir.

properties

Kind of environment variables for the pre and post adapters. These properties are copied to the run file. This is also a convenient way to pass global properties to a pre or post adapter. An adapter is not allowed to access the FEWS global.properties directly. Global properties (between $) are replace by there literal values before copied to the run file. These extra options makes an additional pre or post adapter configuration file unnecessary.

Options:

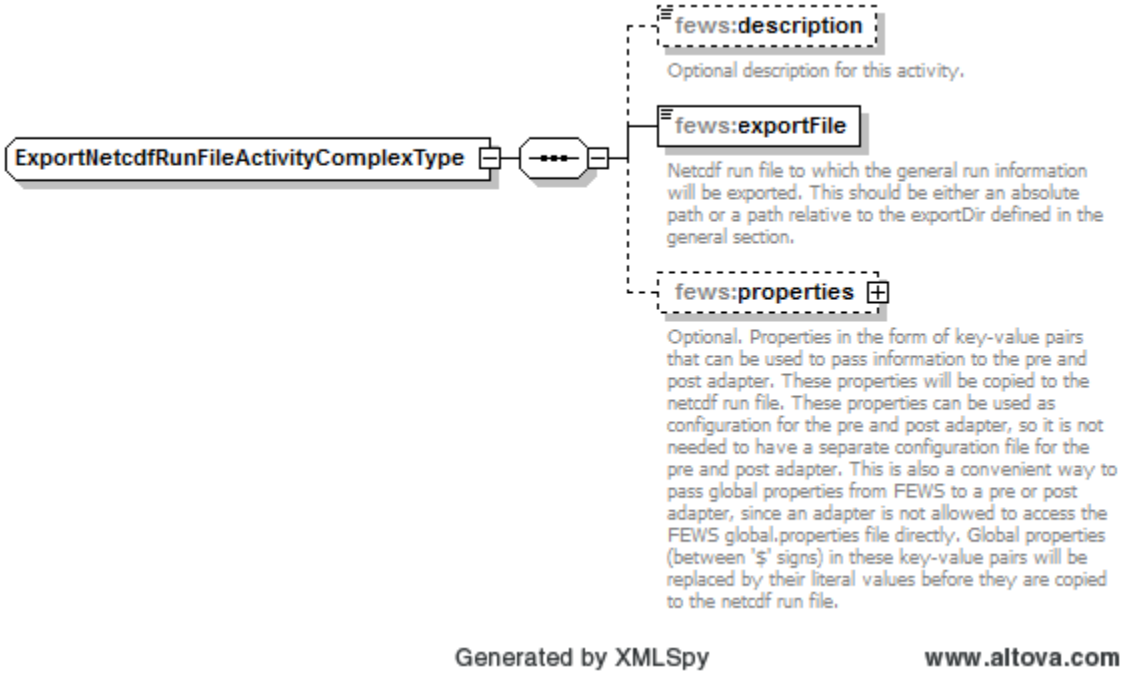
- string
- int
- float
- double (since stable build 2014.01)
- bool

example of exported run file

```
<Run xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.wldelft.nl/fews/PI" xsi:
schemaLocation="http://www.wldelft.nl/fews/PI http://fews.wldelft.nl/schemas/version1.0/pi-schemas/pi_run.xsd"
version="1.5">
        <logLevel>info</logLevel>
    <timeZone>0.0</timeZone>
    <startDateTime date="1900-01-01" time="00:00:00"/>
    <endDateTime date="2100-01-01" time="00:00:00"/>
    <time0 date="2000-01-01" time="00:00:00"/>
    <workDir>workdir</workDir>
    <outputDiagnosticFile>diagnostic</outputDiagnosticFile>
</Run>
```

exportNetcdfRunFileActivity (since stable build 2014.01)

Exports a run file in netcdf format. The netcdf run file contains general information about the run executed by the general adapter that can be used by the pre and post adapter.



Elements of the ExportNetcdfRunFileActivity configuration

description

Optional description for this activity.

exportFile

Netcdf run file to which the general run information will be exported. This should be either an absolute path or a path relative to the exportDir defined in the general section.

properties

Optional. Properties in the form of key-value pairs that can be used to pass information to the pre and post adapter. These properties will be copied to the netcdf run file. These properties can be used as configuration for the pre and post adapter, so it is not needed to have a separate configuration file for the pre and post adapter. This is also a convenient way to pass global properties from FEWS to a pre or post adapter, since an adapter is not allowed to access the FEWS global.properties file directly. Global properties (between '$' signs) in these key-value pairs will be replaced by their literal values before they are copied to the netcdf run file.

Options:

- string
- int
- float
- double (since stable build 2014.01)
- bool

Additionally it is possible to pass a locationAttribute as a property. This allows passing of location attributes to the netcdfRunFile.

Configuration example

```
<exportNetcdfRunFileActivity>
        <description>This run file is passed as argument to XBeachPreAdapter</description>
        <exportFile>run_info.nc</exportFile>
        <properties>
                <bool key="use_friction" value="true"/>
                <locationAttribute key="localModel" locationId="$MODEL$" attributeId="forRunFileExport"/>
        </properties>
</exportNetcdfRunFileActivity>
```

Example of an exported netcdf run file (converted to text format):

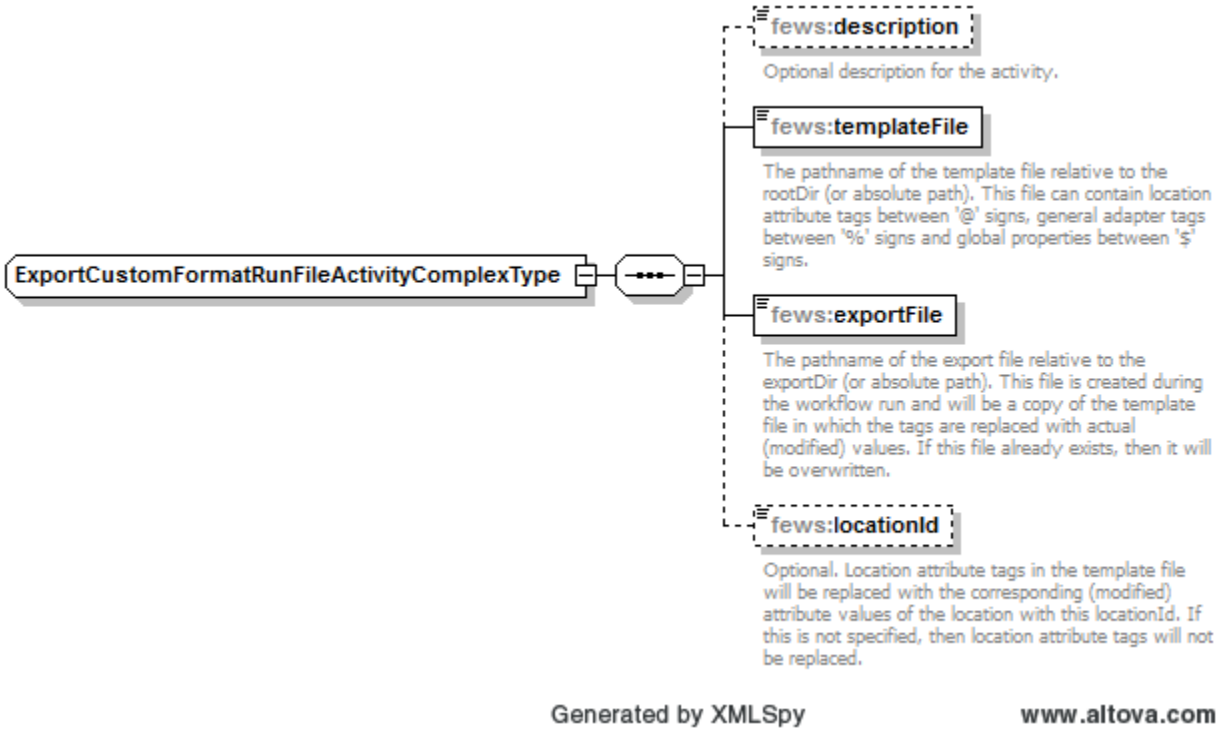
A netcdf run file will be generated by FEWS and contains information that is needed by the pre/post adapter.
The variables input_netcdf_files and input_state_files are optional and list the files that are exported from FEWS. The variables output_netcdf_files and output_state_files are optional and list the files that need to be imported by FEWS.
The attributes of the properties variable form the configuration for the pre/post adapter and can be configured in FEWS (both the attribute names and attribute values are configurable).
All paths in the netcdf run file are relative to the parent directory of the netcdf run file.

An example of a netcdf run file can be downloaded here: run_info_example.nc. Below is the same example netcdf run file, converted to text format:

```
netcdf run_info_example {
dimensions:
        path_length = 255 ;
        input_netcdf_file_count = 2 ;
        input_state_file_count = 1 ;
        output_netcdf_file_count = 2 ;
variables:
        double start_time ;
                start_time:long_name = "start_time" ;
                start_time:standard_name = "time" ;
                start_time:units = "minutes since 2002-11-26 00:00:00.0 +0000" ;
        double end_time ;
                end_time:long_name = "end_time" ;
                end_time:standard_name = "time" ;
                end_time:units = "minutes since 2002-11-26 00:00:00.0 +0000" ;
        char work_dir(path_length) ;
        char input_netcdf_files(input_netcdf_file_count, path_length) ;
        char input_state_files(input_state_file_count, path_length) ;
        char output_netcdf_files(output_netcdf_file_count, path_length) ;
        char properties ;
                properties:string_property = "zs_0 with spaces" ;
                properties:INT_property = 3 ;
                properties:float_PROPERTY = 3.5f ;
                properties:DOUBLE_PROPERTY_321 = 0.123456789 ;
                properties:logical_property_1 = "true" ;
                properties:logical_property_2 = "false" ;

// global attributes:
                :title = "Run file" ;
                :institution = "Deltares" ;
                :source = "Run file from Delft-FEWS" ;
                :history = "2014-03-18 12:22:55 GMT: exported from Delft-FEWS to D:
\\fews\\workspace\\fews\\junit_test_output\\nl\\wldelft\\fews\\system\\plugin\\generaladapter\\exportDir\\run_in
fo.nc" ;
                :references = "http://www.delft-fews.com" ;
data:

 start_time = 0 ;

 end_time = 5880 ;

 work_dir = "..\\work" ;

 input_netcdf_files =
  "..\\work\\boundary_data.nc",
  "..\\work\\more_boundary_data.nc" ;

 input_state_files =
  "..\\input_state\\state.inp" ;

 output_netcdf_files =
  "..\\work\\model_output_data1.nc",
  "..\\work\\model_output_data2.nc" ;

 properties = " " ;
}
```

exportCustomFormatRunFileActivity

Activity to create a custom run info file. This activity replaces tags in a template file with actual (modified) values. The template file can contain location attribute tags between '@' signs, general adapter tags between '%' signs and global properties between '$' signs. The specified template file will be copied to the specified export file before the tags are replaced. So the template file itself is not changed. If the export file already exists, then it will be overwritten.



Elements of the ExportCustomFormatRunFileActivity configuration

Note: the application of this feature is (still) possible, but not recommended. Better use the export(Netcdf)RunFileActivity in combination with a model pre-adapter instead.

description

Optional description for the activity.

templateFile

The pathname of the template file relative to the rootDir (or absolute path). This file can contain location attribute tags between '@' signs, general adapter tags between '%' signs and global properties between '$' signs.

In addition to the tags specified in the documentation on the general section, also tags %START_DATE_TIME% and %END_DATE_TIME% are accommodated

exportFile

The pathname of the export file relative to the exportDir (or absolute path). This file is created during the workflow run and will be a copy of the template file in which the tags are replaced with actual (modified) values. If this file already exists, then it will be overwritten.

locationId

Optional. Location attribute tags in the template file will be replaced with the corresponding (modified) attribute values of the location with this locationId. If this is not specified, then location attribute tags will not be replaced.

fixedWidth

Available since 2017.02. Optional field used to specify the width (length in characters) the location attribute tags and global properties ('@' and '$') should be replaced with. The result will be padded on the left with the required number of spaces. Note that if no numberOfDecimals is specified, all number attributes will be rounded based on the number of significant digits that will fit in the configured fixedWidth. If a numberOfDecimals is specified, the number attributes will be rounded to the specified number of decimals instead. If the text for any attribute or global property contains more than the specified number of characters, a warning is logged and the value is outputted as is.

numberOfDecimals

Available since 2017.02. Optional field used to specify the number of decimals all number attributes ('@') should be rounded to. Note that when a fixedWidth is also specified, the rounded values must always fit in the number of characters specified.

configuration example

```
<generalAdapterRun 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/generalAdapterRun.xsd">
        <general>
                ...
                <startDateTimeFormat>yyyy MM dd HH mm</startDateTimeFormat>
                <endDateTimeFormat>yyyy MM dd HH mm</endDateTimeFormat>
        </general>
        <activities>
                <exportActivities>
                        <exportCustomFormatRunFileActivity>
                                <templateFile>%ROOT_DIR%/templatefiles/event_tox2_template.inp</templateFile>
                                <exportFile>event_tox2.inp</exportFile>
                                <locationId>locationWithAttributes</locationId>
                                <fixedWidth>10</fixedWidth>
                                <numberOfDecimals>5</numberOfDecimals>
                        </exportCustomFormatRunFileActivity>
                </exportActivities>
        </activities>
</generalAdapterRun>
```

Example of template file:

```
%START_DATE_TIME%                     ! MODEL START         : ISYEAR,ISMONTH,ISDATE,ISHR,ISMN [year,month,day,
hour,minute]
%END_DATE_TIME%                       ! MODEL STOP          : ISYEAR,ISMONTH,ISDATE,ISHR,ISMN [year,month,day,
hour,minute]
2                                     ! TIME STEP           :[sec]
@TOXIC_ID@                            ! TOXIC ID            : CAS NO.-TOXIC, 444#-OI
@ACCIDENT_LOC_X@     @ACCIDENT_LOC_Y@ ! ACCIDENT LOC.       : X   Y      TM coordinate (referred in LXLY.INP)
@ACCIDENT_TIME@                       ! ACCIDENT TIME       : IEVYEAR,IEVMONTH,IEVDATE,IEVHR,IEVMN [year,month,
day,hour,minute]
@spill_duration@                      ! spill duration      : [MIN]
@spill_material_mass@                 ! spill material mass : [G]
10.                                   ! CHLA                : CHLA CONC. [UG/L] FOR TOXIC CALCULATION
5.                                    ! TEM                 : WATER TEMPERATURE [DEGREE C] FOR TOXIC CALCULATION
5.                                    ! SSC                 : Suspended Solid Conc.  [MG/L] FOR TOXIC
CALCULATION
0.1                                   ! DOC                 : DOC CONC.  [MG/L] FOR TOXIC CALCULATION
100.                                  ! I                   : IRADIATION [LY/DAY]
5.                                    ! WSPD                : WINDSPEED  [M/S]
$SIMULATION_NUMBER$                   ! simulation number
%TIME0%$Identifier$%TIME0%   $Identifier$ ! test
```

Example of exported file:

```
2002 11 30 09 00                     ! MODEL START         : ISYEAR,ISMONTH,ISDATE,ISHR,ISMN [year,month,day,
hour,minute]
2002 11 30 11 00                      ! MODEL STOP          : ISYEAR,ISMONTH,ISDATE,ISHR,ISMN [year,month,day,
hour,minute]
2                                     ! TIME STEP           :[sec]
131-52-2                             ! TOXIC ID            : CAS NO.-TOXIC, 444#-OI
235875.9     329956.4 ! ACCIDENT LOC.       : X   Y      TM coordinate (referred in LXLY.INP)
2012 01 10 09 00                      ! ACCIDENT TIME       : IEVYEAR,IEVMONTH,IEVDATE,IEVHR,IEVMN [year,month,
day,hour,minute]
10.0                      ! spill duration      : [MIN]
500000.0                 ! spill material mass : [G]
10.                                   ! CHLA                : CHLA CONC. [UG/L] FOR TOXIC CALCULATION
5.                                    ! TEM                 : WATER TEMPERATURE [DEGREE C] FOR TOXIC CALCULATION
5.                                    ! SSC                 : Suspended Solid Conc.  [MG/L] FOR TOXIC
CALCULATION
0.1                                   ! DOC                 : DOC CONC.  [MG/L] FOR TOXIC CALCULATION
100.                                  ! I                   : IRADIATION [LY/DAY]
5.                                    ! WSPD                : WINDSPEED  [M/S]
73                   ! simulation number
200211300900FSS200211300900   FSS ! test
```

exportAreaSelectionActivity

Exports an area selection as shape file (since 2021.02) or mask as a PiMapStack. This activity requires an embedded polygon to be specified within the areaSelectionShapeFileBase64 field of the TaskProperties. This file is then available to be used by the pre and post adapter.

description

Optional description for the activity. Used for reference purposes only.

exportFile

File name of the file to be exported. Shape file (*.shp) or pi map stack file (*.xml). Always placed into the export dir.

gridLocationId

Optional, does not apply to shape file

Locations that contains the reference grid information.

gridFormat

Optional, does not apply to shape file

Format of the PiMapStack file. Can be

- *asc* : for exporting to ARC-INFO ASCII grid format
- *pcrgrid* : for exporting to PCRaster native grid file format
- *usgs* : for exporting to USGS DEM format (BIL)

exportLocationAreaActivity (currently NGMS only)

Exports an area selection mask in the form of a PiMapStack. This activity requires selectedLocationIds to be specified within the TaskProperties. This selection is then available to be used by the pre and post adapter.

description

Optional description for the activity. Used for reference purposes only.

exportFile

File name of the file to be exported. Always placed into the export dir.

gridLocationId

Locations that contains the reference grid information.

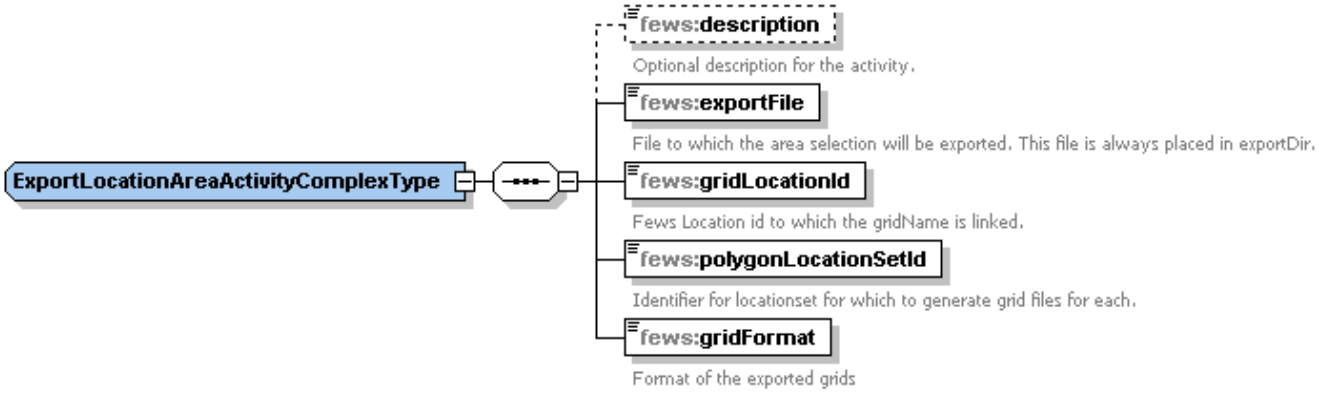
gridFormat

Format of the PiMapStack file. Can be

- *asc* : for exporting to ARC-INFO ASCII grid format
- *pcrgrid* : for exporting to PCRaster native grid file format
- *usgs* : for exporting to USGS DEM format (BIL)

polygonLocationId

Identifier for locationset for which to generate grid files. The locations should have polygons associated with them.



exportRatingCurveActivity

Exports rating curves to pi_ratingcurves.xml file. The rating curves with the start time within the GeneralAdapter run period will be exported.

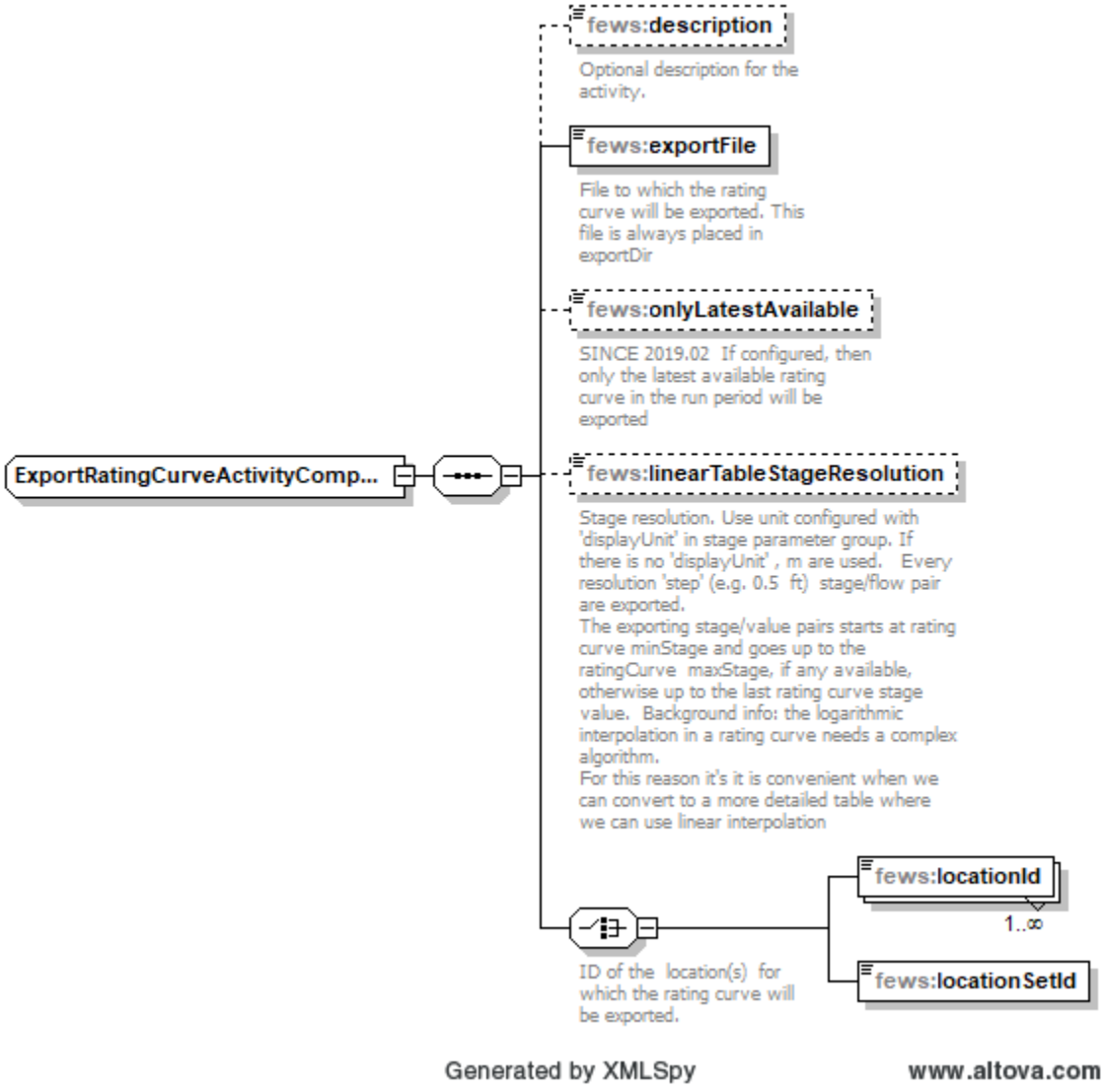


Figure 1 Elements of the exportRatingCurveActivity configuration

description

Optional description for the activity. Used for reference purposes only.

exportFile

File name of the file to be exported. Always placed into the export dir.

onlyLatestAvailable

If 'onlyLatestAvailable' is used, then only the latest available rating curve in the run period will be exported. By default all rating curves within the run period are exported

linearTableStageResolution

Stage resolution. Use unit configured with 'displayUnit' in stage parameter group. If there is no 'displayUnit' , m are used. Every resolution 'step' (e.g. 0.5 ft) stage/flow pair are exported. The exporting stage/value pairs starts at rating curve minStage and goes up to the ratingCurve maxStage, if any available, otherwise up to the last rating curve stage value. The logarithmic interpolation in a rating curve needs a complex algorithm. For this reason it's it is convenient when we can convert to a more detailed table where we can use linear interpolation

locationId

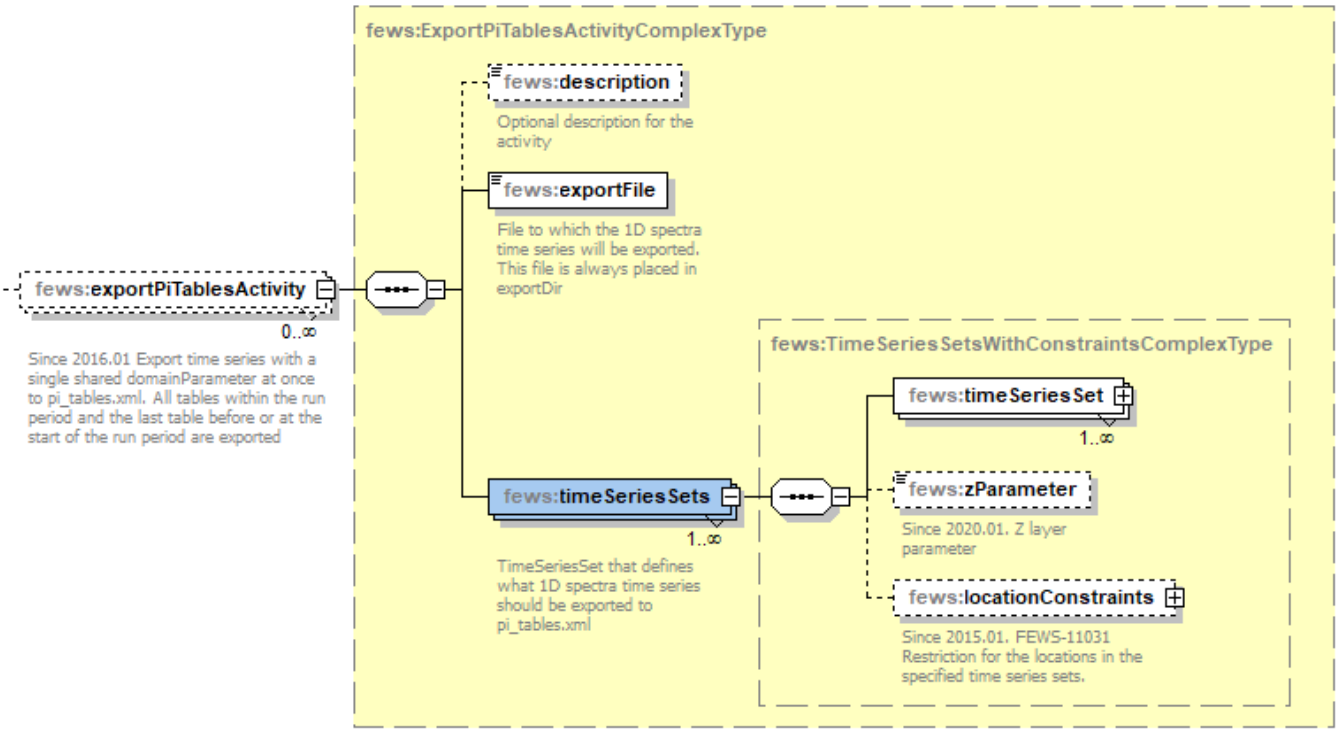
Locations the rating curves will be exported for.

locationSetId

Locations the rating curves will be exported for.

exportPiTablesActivity

Export time series with a single shared domainParameter at once to pi_tables.xml. All tables within the run period and the last table before or at the start of the run period are exported. FEWS-12927.



exportCsvModuleRunTablesActivity (since 2020.02)

Root element for exporting csv files for Module Run Tables that have been imported for display using the Module Run Table Display component.

Elements of the exportCsvModuleRunTablesActivity section.

- charset- Optional charset used for reading the csv file. Defaults to ISO-8859-1.
- description - Optional description only used for reference.

- exportFile - Path and name of a csv file from the model. This file will be exported.
- moduleInstanceId - Module instance Id of the module run table to export
- displayName - display name (optional) of the module run table to export, this can be used when multiple tables have been imported using the same moduleInstanceId.
- onFailWarnAndContinue - when true GA will log a warning instead of aborting the workflow is a table with the given modulInstanceId cannot be found.

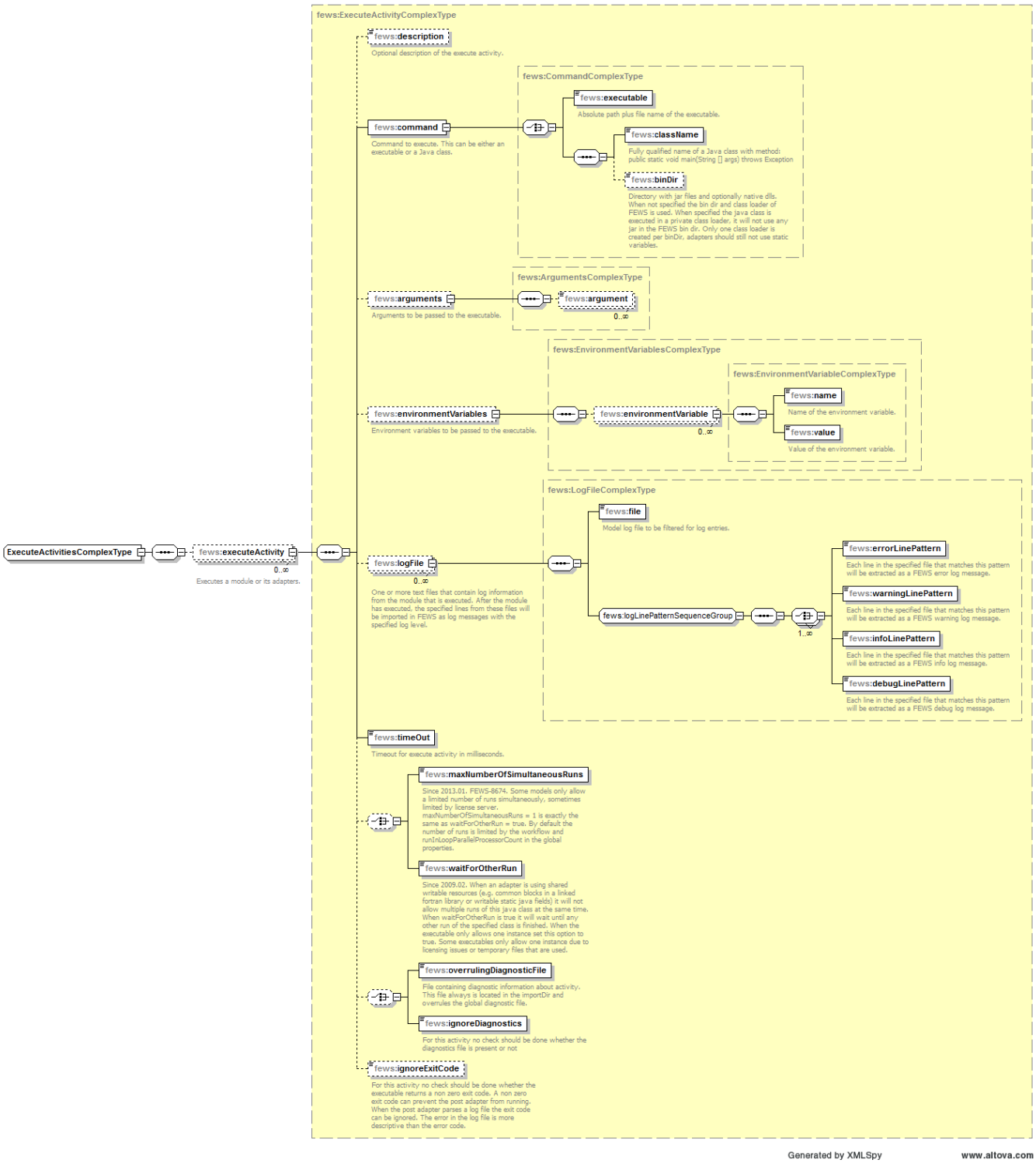
Columns of the Module Run Table will be exported "as-is" to the CSV file.

Configuration example:

In the following example one exportCsvModuleRunTablesActivity is specified for a module run table that was imported earlier using the ModuleInstanceId *Big10_RiverWare_Big10*. As more than one table may have been imported using that module, the Display name of the table is also given as additional selector.

```
<exportCsvModuleRunTablesActivity>
                <moduleInstanceId>Big10_RiverWare_Big10</moduleInstanceId>
                <charset>UTF-8</charset>
                <description>RCT Tool Csv Module Data Activity Test case.</description>
                <exportFile>convergence_ipopt.csv</exportFile>
                <displayName>Violated Constraints RBS</displayName>
                <onFailWarnAndContinue>true</onFailWarnAndContinue>
  </exportCsvModuleRunTablesActivity>
```

Execute Activities



Elements of the ExecuteActivity configuration

executeActivity

Root element for the definition of an execute activity. Executes a module or its adapters. For each external executable or Java class to run, an executeActivity must be defined. Multiple entries may exist.

> In case of a failed run, Delft-FEWS attempts to terminate any child process afterwards. In order to be able to do this reliably, it is important to only create processes with a proper process tree. For instance, an unsupported option is to execute a Windows batch file starting executables in the background where the batch file terminates itself, leaving the child processes orphaned. This is illegal because Delft-FEWS cannot detect these as child processes and therefore not terminate the child processes. This will lead to model executables from dead taskruns to remain while possible locking files, blocking new runs with the same executable.

executeOnPreviousError

Attribute available since 2019.02. Execute this activity even after error occurred in previous execute activity, run will still fail. Can be useful to still process (partial) output.

description

Optional description for the activity. Used for reference purposes only.

command

Root element to define command to execute. This can be either an executable or a Java class.

- executable - File name and location of the executable to run the command is an executable. The file name may include environment variables, as well as tags defined in the general adapter or on the global.properties.
- className - Name of Java Class to run if the command defined as a Java class. This class may be made available to Delft-FEWS in a separate JAR file in the \Bin directory.
  This class can be executed outside FEWS for testing with
  bin/windows/Delft-FEWSc.exe -Xmx512m -Djava.library.path=*adapterbin* -Wclasspath.1=*adapterbin**.jar -Wmain.class=*classname arguments*
- binDir - Optional. Directory with jar files and optionally native dlls. When not specified the bin dir and classloader 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. All dependencies should also be in this configured bin dir.
- moduleDataSetName - Optional. As of 2014.02 it is possible to update the binaries in the binDir using a module data set. The loaded binaries will first be unloaded before the content of the module dataset is unpacked to binDir. Since 2017.01 the binDir will be deleted in order to update it, therefore a restriction has been build in that the binDir should be a sub directory of the region home or named "bin".
- customJreDir - Optional. As of 2016.01 it possible to configure a directory of a specific jre that should be used to run a java class. This could be required for modules that are not compatible with the java version used by FEWS. Or this method could be used to instantiate a separate JVM for the process being executed
- jvmArg - Optional, unbounded and only in combination of customJreDir. Specifies extra jvm arguments for instance -mx512m to give the jvm more memory than the -mx256m that FEWS uses by default for the custom jre.

Example customJre using Fews-bin

```
<command>
        <className>nl.wldelft.fews.adapter.urbsiniadapter.UrbsIniAdapter</className>
        <customJreDir>$BIN_DIR$/windows/jre</customJreDir>
        <jvmArg>-mx512m</jvmArg>
</command>
```

console

Element for connecting the console output of the activity to FEWS

- redirectToLogFile: Since 2018.02. Write the console output written by the executable to the specified file.
- restartWorkflowLinePattern: Since 2018.01. As soon the executable writes a line, matching this pattern, to the console the complete workflow is restarted
- maxRestartWorkflowCount: Since 2018.01. Max number of times the workflow is restarted. Default is 2
- progressPattern: Since 2021.02. Pattern to extract a percentage number from a console log line, each line will be checked for the pattern when it comes through. The places of the question marks should contain the value. Will be read as double but written as integer rounded down. Question marks should be at the start or end of the pattern. For instance "progress: ???" When used in combination with "activityDurationWeight" it will be scaled accordingly. For instance when 2 activities are present with a weight of 1, the first progress will be rescaled from 0 to 50 and the second from 50 to 100. The progress of a running task can be viewed in the Running Forecasts tab of the system monitor.

activityDurationWeight

Weight of the activity compared to others, used for tracking progress percentage of the module.

arguments

Optional. Root element for defining arguments to be passed to the executable/Java class

- argument - Definition of an argument to be passed to the executable/Java Class

environmentVariables

Optional. Root element for defining environment variables prior to running the executable/Java class

- environmentVariable - Definition of an environment variable prior to running the executable/Java class
- environmentVariable.name - Name of environment variable
- environmentVariable.value - Value of environment variable

For example, to append some directory to the Windows PATH environment variable use:

Appending a directory to the Windows PATH environment variable

```
<environmentVariables>
        <environmentVariable>
                <name>path</name>
                <value>%PATH%;%ROOT_DIR%\wflow_bin;</value>
        </environmentVariable>
</environmentVariables>
```

logFile

Optional (since stable build 2014.01) . One or more text files that contain log information from the module that is executed. After the module has executed, the specified lines from these files will be imported in FEWS as log messages with the specified log level.

- file - Path and name of a log file to be filtered for log entries. This should be either an absolute path or a path relative to the rootDir defined in the general section.
- errorLinePattern - Each line in the specified file that matches this pattern will be extracted as a FEWS error log message.
- warningLinePattern - Each line in the specified file that matches this pattern will be extracted as a FEWS warning log message.
- infoLinePattern - Each line in the specified file that matches this pattern will be extracted as a FEWS info log message.
- debugLinePattern - Each line in the specified file that matches this pattern will be extracted as a FEWS debug log message.
- restartWorkflowLinePattern - The first line in the specified file that matches this pattern will restart the complete root workflow (Since 2018.01)

Configuration example:

```
<logFile>
        <file>XBerror.txt</file>
        <!-- Import every line as a separate FEWS error log message. -->
        <errorLinePattern>*</errorLinePattern>
</logFile>
<logFile>
        <file>XBwarning.txt</file>
        <!-- Import every line that contains "warning" as a separate FEWS info log message. -->
        <infoLinePattern>*warning*</infoLinePattern>
</logFile>
<logFile>
        <file>XBlog.txt</file>
        <!-- Import every line that contains "ERROR" as a separate FEWS debug log message. -->
        <debugLinePattern>*ERROR*</debugLinePattern>
</logFile>
```

timeOut

Timeout to be used when running module (in milliseconds). If run time exceeds timeout it will be terminated and the run considered as having failed.

maxNumberOfSimultaneousRuns

Optional. Since 2013.01. FEWS-8674. Some models only allow a limited number of runs simultaneously, sometimes limited by license server. maxNumberOfSimultaneousRuns = 1 is exactly the same as waitForOtherRun = true. By default the number of runs is limited by the workflow and runInLoopParallelProcessorCount in the global properties.

waitForOtherRun

Optional. Since 2009.02. When an adapter is using shared writable resources (e.g. common blocks in a linked fortran library or writable static java fields) it will not allow multiple runs of this java class at the same time. When waitForOtherRun is true it will wait until any other run of the specified class is finished. When the executable only allows one instance set this option to true. Some executables only allow one instance due to licensing issues or temporary files that are used.

overrulingDiagnosticFile

Optional. File containing diagnostic information about activity. This file always is located in the importDir and overrules the global diagnostic file.

ignoreDiagnostics

Optional. For this activity no check should be done whether the diagnostics file is present or not.

ignoreExitCode

Optional. For this activity no check should be done whether the executable returns a non zero exit code. A non zero exit code can prevent the post adapter from running. When the post adapter parses a log file the exit code can be ignored. The error in the log file is more descriptive than the error code.

Internal GA variables

Several variables are available to be used as an argument to an external program or in the exportCustomFormatRunFileActivity. You can use in any filename or directory the properties from the global.properties file or the next internal variables:

- TEMP_DIR. The %TEMP_DIR% variable is an internal variable which points to a unique temporary directory which is created in the $REGION_HOME$/Temp and which will be removed afterwards.
- ROOT_DIR
- WORK_DIR
- ENSEMBLE_MEMBER_ID
- ENSEMBLE_MEMBER_INDEX
- TIME0
- START_DATE_TIME (the START_DATE_TIME variable only works in an importStateActivity (since 2014.02) and inside a template file for an exportCustomFormatRunFileActivity (since 2013.02)).
- END_DATE_TIME (the END_DATE_TIME variable only works in an importStateActivity (since 2014.02) and inside a template file for an exportCustomFormatRunFileActivity (since 2013.02)).
- CURRENT_TIME (since 2015.01)
- TASK_ID
- TASK_RUN_ID: task run ID 'as is' i.e. including a colon
- TASK_RUN_ID_FOR_PATH: similar to TASK_RUN_ID except present tag replaces any special character that may invalidate a file name (for example, a colon or a path separator) by an underscore
- TASK_DESCRIPTION
- TASK_USER_ID
- WHAT_IF_ID (Since 2016.02 build 65440)
- WHAT_IF_ID_FOR_PATH (Since 2022.01 build 110777)
- WHAT_IF_NAME (Since 2016.02 build 69419)
- TIME_ZONE_OFFSET_SECONDS

The colon characters ":" will be replaced by an underscore "_". Use the internal variables with % characters (like %TEMP_DIR%) and the global.properties variables with $ characters (like e.g. $DUMP_DIR$).

Import Activities

Figure 76 Elements of the ImportActivities configuration

exportPlaceholderFile

> ⚠ Option exportPlaceholderFile is not supported for spectra data.

This option can be used for all import activities, except for importStateActivity.
If <exportPlaceholderFile>true</exportPlaceholderFile>, then the General adapter will generate placeholder files. A placeholder file is a file with headers only, without timeseries. Its name is the same as the filename configured for the ImportActivity and this placeholderfile is written to the import directory. The placeholder files are written before any execute activity is started. The models cq model adapters should read this placeholder files to see which data should be provided to import in FEWS.

- <importTimeSeriesActivity> writes headers to pi_timeseries.xml
- <importMapStacksActivity> writes headers to pi_mapstacks.xml

- <importNetcdfActivity> writes headers to NetCDF file (only for scalar and grid data, not for spectra data) (It writes the variables as is and does not compact them into smaller types like short or byte like tryCompactingNetCDFData in export module)
- <importProfilesActivity> writes headers to pi_profiles.xml

The intention of this exportPlaceholderFile functionality is that the model cq modeladapter reads the placeholders to see which timeseries are required by FEWS.
After simulation, the model cq modeladapter overwrites these files with its own data over the placeholder files ready to be imported by the import activity.

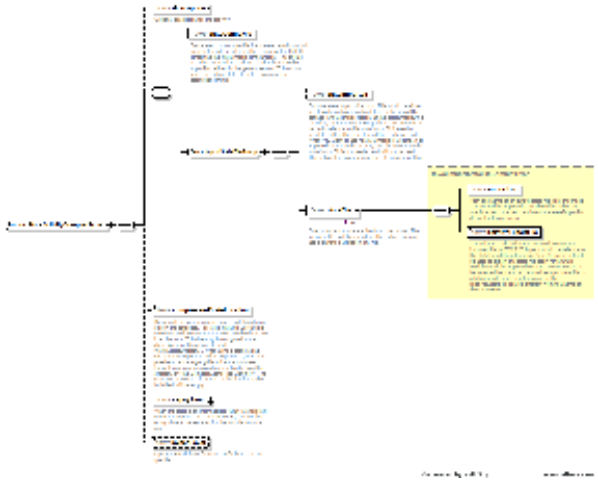
```
<importTimeSeriesActivity>
        <exportPlaceholderFile>true</exportPlaceholderFile>
        <importFile>output.xml</importFile>
        <timeSeriesSets>
        ....
        </timeSeriesSets>
</importTimeSeriesActivity>
```

It is available in the same way for importTimeSeriesActivity, importMapStacksActivity, importNetcdfActivity (only for scalar and grid data, not for spectra data) and importProfilesActivity.

description

Optional description of import activity. Used for reference purposes only

importStateActivity



Elements of the ImportStateActivity section.

Root element for importing modules states resulting from the run of the external modules. Multiple elements may be defined. If no state is to be imported (for example in forecast run as opposed to state run), then the element should not be defined.

- description - Optional description for the activity.
- stateConfigFile - Optional, do not use this if a pi state description xml file is not needed. Name (and location) of the PI-XML file describing the states to be imported. This file contains all necessary information to define state type and location. The moduleInstanceId of the state imported is per definition the current module instance. This should be either an absolute path or a path relative to the importDir defined in the general section. This option reads the output state file paths from a pi state description xml file.
- stateImportDir - Optional state import directory. This should be either an absolute path or a path relative to the importDir defined in the general section. If this stateImportDir is specified, then the importFile paths in this activity can be paths relative to this stateImportDir instead of absolute paths. This stateImportDir is only needed when you want to use relative importFile paths. If all importFile paths in this activity are absolute, then this stateImportDir is not needed and will not be used. This option does not use a pi state description xml file.
- stateFile - One or more output state files from the model. The specified files will be imported. This option does not use a pi state description xml file.
- importFile - Path and name of an output state file from the model. This file will be imported. This should be either an absolute path or a path relative to the stateImportDir defined in this activity. The %END_DATE_TIME% tag can be used here (since 2014.02), in case the state file name contains a timestamp of the end time of the run. **Important: the state time will be saved at the time of the last timesteps exported from the GA. For example, if you want the state to be saved at T0 but you export a timeseries from the GA which is exceeds T0, the time at which the state is saved will be the last available timestamp in that particular timeseries. Use the <ignoreRunPeriod> option the export timeseries activity to ignore timeseries to be included in the check at which time the state is saved!**
- relativeExportFile - This relative path and name are used to store the imported file in FEWS. This path should be relative to the stateExportDir in the exportStateActivity that will be used to export this state file again for a future model run. If the imported output state file needs to be renamed before it can be used as input state file for a future model run, then the name of the relativeExportFile can be different from the name of the importFile. The %END_DATE_TIME% tag can be used here (since 2014.02), in case the state file name contains a timestamp of the end time of the run.
- compressedStateLocation - Optional. By default the warm state is zipped and stored as a blob in the database. For large states (>50MB) it is recommended to store the data outside the database in a directory. This directory is configured in the clientConfig.xml (optional element warmStatesDirectory). When using a stand alone system or using oracle and a single MC system it is possible to store larger grids inside the

database. Expired states are removed automatically from this directory by the CompactCacheFiles workflow. The state description is still written to the database, the blob field will be empty.
- expiryTime - Optional. When the state is an intermediate result in a forecast run you can make the state expire. By default the expiry time is the same as for the module instance run.
- synchLevel - Optional synch level for state. Defaults to 0 is not specified (i.e. same as data generated by the forecast run)

Configuration example:

New example that does not use a pi state description xml file. In this case, the output state file paths are configured directly in the importStateActivity as absolute paths:

```
<importStateActivity>
        <stateFile>
                <importFile>%WORK_DIR%/state.out</importFile>
                <!-- Rename imported state file so that it can be used as input state for a future model run. --
>
                <relativeExportFile>state.inp</relativeExportFile>
        </stateFile>
        <stateFile>
                <importFile>%WORK_DIR%/state2.out</importFile>
                <!-- Rename imported state file so that it can be used as input state for a future model run. --
>
                <relativeExportFile>state2.inp</relativeExportFile>
        </stateFile>
</importStateActivity>
```

New example that does not use a pi state description xml file. In this case, the output state file paths are configured directly in the importStateActivity as paths relative to a stateImportDir:

```
<importStateActivity>
        <stateImportDir>%WORK_DIR%</stateImportDir>
        <stateFile>
                <importFile>state.out</importFile>
                <!-- Rename imported state file so that it can be used as input state for a future model run. --
>
                <relativeExportFile>state.inp</relativeExportFile>
        </stateFile>
        <stateFile>
                <importFile>state2.out</importFile>
                <!-- Rename imported state file so that it can be used as input state for a future model run. --
>
                <relativeExportFile>state2.inp</relativeExportFile>
        </stateFile>
</importStateActivity>
```

Since 2017.02. Import a dynamic number of states for different state times at once. Pi state description xml file is not used. When exported the file is named "start.bin":

```
<importStateActivity>
   <stateImportDir>states</stateImportDir>
   <stateFileDateTimePattern>'state'yyyyMMddHHmm'.bin'</stateFileDateTimePattern>
   <relativeExportFile>start.bin</relativeExportFile>
</importStateActivity>
```

Old example that reads the input state file paths from a pi state description xml file. Do not use this if a pi state description xml file is not needed:

```
<importStateActivity>
        <stateConfigFile>pi_output_state_description.xml</stateConfigFile>
</importStateActivity>
```

Example of a client config for stand alone when importing to the warm states directory. The meta data is still stored in the data base. Files in this directory that don't have meta data will be automatically deleted by the rolling barrel.

```
<clientConfiguration 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/clientConfig.xsd">
   <localDataStoreFormat>Firebird</localDataStoreFormat>
   <warmStatesDirectory>%REGION_HOME%/warmStates</warmStatesDirectory>
</clientConfiguration>
```

importTimeSeriesActivity

Root element for importing scalar and polygon time series resulting from the run of the external modules. Multiple elements may be defined. importFile and timeSeriesSet should be defined.

- importFile - PI-XML file describing the time series to be imported. The file contains all information on type of data to be imported (scalar, longitudinal, grid, polygon). For all data types except the grid the file also contains the time series data If the directory location is not explicitly specified the file will be expected to be read from the importDir defined in the general section.

importMapStacksActivity

Root element for importing grid time series resulting from the run of the external modules. Multiple elements may be defined. importFile and timeSeriesSet should be defined.

importNetcdfActivity

Imports scalar, grid or 1d/2d spectra time series from a NetCDF file resulting from the run of the external modules. All time series specified inside one importNetcdfActivity must have the same value type (grid, scalar, 1d spectra or 2d spectra). For this activity the exportPlaceholderFile option only works for scalar and grid data, not for spectra data. Note: this activity is exactly the same as importPiNetcdfActivity, only the name is better, since the NetCDF format has nothing to do with the PI format.

- **maximumSnapDistance** Since 2014.01. FEWS-10771. Optional maximum horizontal snap distance in meters. When the parser provides horizontal location coordinates (x,y) and no locationIds, then the location mapping will be done by matching the horizontal coordinates. The horizontal snap distance is the tolerance used to detect which internal and external horizontal coordinates are the same. This only works when the input format provides the coordinate system for the coordinates of the locations. When the parser does not provide the coordinates for a time series an error is logged. Note: this option has no effect for grid data. Note 2: it is not possible to import data using horizontal coordinates and using locationIds in the same importNetcdfActivity, need to define separate import activities for that (one with maximumSnapDistance and one without maximumSnapDistance).
- **maximumVerticalSnapDistance** Since 2014.02. Optional maximum vertical snap distance in meters. When the parser provides vertical location coordinates (z) and no locationIds, then the location mapping will be done by matching the vertical coordinates. The vertical snap distance is the tolerance used to detect which internal and external vertical coordinates are the same. This only works when the input format provides the coordinates of the locations. When the parser does not provide the vertical coordinates for a time series an error is logged. Note: this option currently only works for importing horizontal layers from netcdf 3D grid data. Note 2: it is not possible to import data with z-coordinates (layers from 3D grids) and data without z-coordinates (2D grids) in the same importNetcdfActivity, need to define separate import activities for that (one with maximumVerticalSnapDistance and one without maximumVerticalSnapDistance).
- **startWhileRunningExecuteActivities** Default is false. If this is true, then this importActivity will run continuously during the configured execute activities. Additionally this importActivity will also run as part of the configured import activities as normal. This way it is possible to import data that is produced by an execute activity, while it is being produced. For instance if a model run writes new output data to an existing file after each timeStep, then the continuously running importActivity will immediately import the file, including the new data. This way the new data can be viewed in FEWS as soon as it becomes available, i.e. already during the model run. Currently the data that is imported during the execute activities can only be viewed after selecting "open most recent running forecast and adjust system time" from the debug menu in the FEWS Explorer window. If the running forecast is opened and selected in the data viewer, then the displays are updated each time when new data becomes available during the run. This feature only has effect for stand alone FEWS systems and for FEWS systems that use direct database access.

import multiple files at once

- **folder** Import all file in the specified folder.
- **fileNamePatternFilter** Only import files that match the pattern. (e.g. *.nc)
- **fileNameLocationIdPattern**. Since 2023.02. Regular Expression. When a match of the pattern in the filename is found, this will overrule the external grid location Id for the time series being imported A simple pattern is (.*) which matches the whole filename. (.*)\.nc extracts the file name without the .nc extension. An other simple pattern is .{2}(.*).{4} that removes the first 2 and last 4 character of the filename to get the id More complicated expressions can be found at http://en.wikipedia.org/wiki/Regular_expression

```
<importNetcdfActivity>
       <folder>%ROOT_DIR%/importDir</folder>
       <fileNamePatternFilter>*.nc</fileNamePatternFilter>
       <fileNameLocationIdPattern>(.*)\.nc</fileNameLocationIdPattern>
       <timeSeriesSets>
               <timeSeriesSet>
                       <moduleInstanceId>GeneralAdapterRun</moduleInstanceId>
                       <valueType>scalar</valueType>
                       <parameterId>WaterLevel</parameterId>
```

```
                        <locationId>H-2001</locationId>
                        <timeSeriesType>external historical</timeSeriesType>
                        <timeStep unit="minute" divider="1" multiplier="15"/>
                        <readWriteMode>add originals</readWriteMode>
                        <ensembleId>prognose</ensembleId>
                </timeSeriesSet>
        </timeSeriesSets>
        <ignoreNonExistingLocationSets>true</ignoreNonExistingLocationSets>
</importNetcdfActivity>
```

Combined with workflow ensemble loop

When an importNetcdfActivity is run within an ensemble loop defined in the workflow configuration and the %ENSEMBLE_MEMBER_ID% tag is used in the <importDir> or the <importFile> it will loop over the ensemble members and use the ensemble member index for the imported scalar time series (since 2017.02).

```
        <activity>
                <moduleInstanceId>ImportNetcdfScalarEnsembles</moduleInstanceId>
                <ensemble>
                        <ensembleId>Ens</ensembleId>
                        <ensembleMemberIndexRange start="0" end="4"/>
                        <runInLoop>false</runInLoop>
                </ensemble>
        </activity>
```

When it is not desired that all other activities are also run for each ensemble member, make sure <runInLoop> is set to false.

This is useful when 1 general adapter run generates many similar ensemble files that do not contain data from which their ensemble id can be determined besides their file name or their parent directories. Like a run from OpenDA.

importPiNetcdfActivity

Deprecated, do not use. Please use importNetcdfActivity instead. Imports scalar, grid or 1d/2d spectra time series from a NetCDF file resulting from the run of the external modules. All time series specified inside one importPiNetcdfActivity must have the same value type (grid, scalar, 1d spectra or 2d spectra). For this activity the exportPlaceholderFile option only works for scalar and grid data, not for spectra data. For more documentation see importNetcdfActivity.

importProfilesActivity

Root element for importing longitudinal profile time series resulting from the run of the external modules. Multiple elements may be defined. importFile and timeSeriesSet should be defined.

Optional description of import activity. Used for reference purposes only

importShapeFileActivity

Root element for importing shape files resulting from the run of the external modules.

A <shapeFileImportDir> can be defined which is either relative (to the general <importDir>) or absolute. When it is not defined the general <importDir> will be used. If the defined directory does not exist, an exception will be thrown. The import directory can be empty.

A <fileDateTimePattern> should be defined to filter out the shape files that need to be imported and to extract the time for the shape in the time series. If the directory is not empty, but none of the files match the defined pattern, an exception will be thrown. Since 2023.01 the fileDateTimePattern is optional. When there is no fileDateTimePattern the *.shp file in the import directory will be imported. The time zero is used as time stamp. Only one shp-file should exist in the import directory. Files without the shp/dbf extension are ignored

A <geoDatum> can be defined if the shape file is not in WGS84. When this is defined, a conversion to WGS84 will take place before storing the time series into the database.

The <timeSeriesSet> will automatically be used for the imported shape, no idmapping will take place.

GA importShapeFileActivity

```
<importActivities>
        <importShapeFileActivity>
                <shapeFileImportDir>importshapefileactivityimportdir</shapeFileImportDir>
                <fileDateTimePattern>'ImportShapeFileActivity_'ddMMMyyyyHHmmss'.shp'</fileDateTimePattern>
                <geoDatum>$GEODATUM$</geoDatum>
                <timeSeriesSet>
                        <moduleInstanceId>ImportActivityNoneEquidistantProfile</moduleInstanceId>
                        <valueType>polygon</valueType>
```

```
                        <parameterId>H.obs</parameterId>
                        <locationId>SX.E7842</locationId>
                        <timeSeriesType>external historical</timeSeriesType>
                        <timeStep unit="nonequidistant"/>
                        <relativeViewPeriod unit="hour" start="0" end="2"/>
                        <readWriteMode>read only</readWriteMode>
                </timeSeriesSet>
        </importShapeFileActivity>
</importActivities>
```

The attributes (dbf file columns) can be imported by configuring the attributes to import. The attributes are store as time series properties. Only the first row of the dbf file will be imported.

GA importShapeFileActivity

```
<importActivities>
        <importShapeFileActivity>
                <shapeFileImportDir>importshapefileactivityimportdir</shapeFileImportDir>
                <fileDateTimePattern>'ImportShapeFileActivity_'ddMMMyyyyHHmmss'.shp'</fileDateTimePattern>
                <geoDatum>$GEODATUM$</geoDatum>
                <charset>ISO-8859-1</charset>
                <shapeFileAttribute attributeId="Range Min" propertyKey="Min"/>
                <shapeFileAttribute attributeId="Range Max" propertyKey="Max"/>
                <timeSeriesSet>
                        <moduleInstanceId>ImportActivityNoneEquidistantProfile</moduleInstanceId>
                        <valueType>polygon</valueType>
                        <parameterId>H.obs</parameterId>
                        <locationId>SX.E7842</locationId>
                        <timeSeriesType>external historical</timeSeriesType>
                        <timeStep unit="nonequidistant"/>
                        <relativeViewPeriod unit="hour" start="0" end="2"/>
                        <readWriteMode>read only</readWriteMode>
                </timeSeriesSet>
        </importShapeFileActivity>
</importActivities>
```

The (placeholder) location used for this polygon is defined in Locations.xls in the same way you define a placeholder location for a grid.

polygon location definition Location.xls

```
                <location id="RASMapper_Nickajack">
                        <description>Location to hold Nickajack polygon and raster results</description>
                        <shortName>RiverSystem</shortName>
                        <x>0</x>
                        <y>0</y>
                        <z>0</z>
                </location>
```

Please note that you can not make use of the dataLayer element when displaying polygon data in the spatial display, see below for a config example. Also, it is not possible to view this data from the database viewer.

gridDisplay config for polygons

```
                        <gridPlot id="Extents" name="Inundation Extent">
                                <mapExtentId>Chickamauga Reservoir</mapExtentId>
                                <timeSeriesSet>
                                        <moduleInstanceId>RASMapper_Chickamauga</moduleInstanceId>
                                        <valueType>polygon</valueType>
                                        <parameterId>inundation_boundary</parameterId>
                                        <locationId>RASMapper_Chickamauga</locationId>
                                        <timeSeriesType>simulated forecasting</timeSeriesType>
                                        <timeStep unit="hour" multiplier="1"/>
                                        <relativeViewPeriod unit="hour" start="-240" end="240"/>
                                        <readWriteMode>read only</readWriteMode>
```

```
                        </timeSeriesSet>
                </gridPlot>
```

importCsvModuleRunTablesActivity (since 2015.01)

Root element for importing csv files that can be displayed using the Module Run Table Display component.

Elements of the importCsvModuleRunTablesActivity section.

- charset- Optional charset used for reading the csv file. Defaults to ISO-8859-1.
- description - Optional description only used for reference.
- importFile - Path and name of a csv file from the model. This file will be imported.
- displayName - name used to display the import file name in the ModuleRunTableDisplay component.
- table - mapping of column names to displayNames and types that will be used to store into FEWS. If a column isn't mapped the csv column name is used as displayName and String is used as column type. The following types can be mapped using the name attribute to map a csv column name to a displayName:
    - stringColumn - set the column type to string.
    - booleanColumn - set the column type to boolean. Only the values 'true' and 'false' are allowed.
    - integerColumn - set the column type to integer.
    - doubleColumn - set the column type to double.
    - dateTimeColumn - set the column type to dateTime. In case a date pattern is specified, the string will be parsed according this format. If no format was specified, a timestamp is assumed in milliseconds.

Only columns that are mapped will be imported. The imported values will be stored in the order of the configured column mappings.

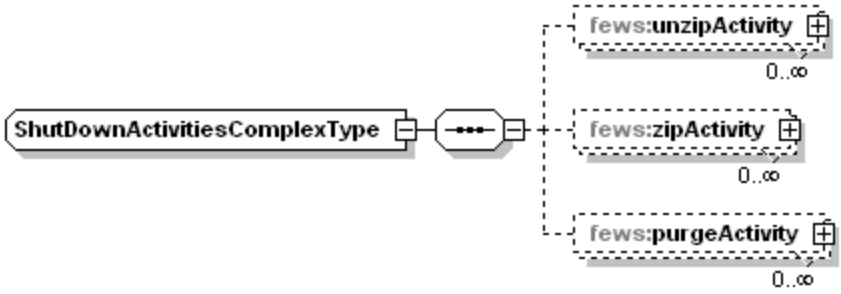
Configuration example:

In the following example one importCsvModuleRunTablesActivity is specified for the import file import_csv_module_test1.csv. In case more than one csv file has to be imported, for each csv file an activity has to be configured. Note that in this example the csv column 'Float' hasn't been mapped. This will result in not importing the 'Float' column values.

```
        <importActivities>
                <importCsvModuleRunTablesActivity>
                        <charset>UTF-8</charset>
                        <description>Csv Module Data Activity Test case.</description>
                        <importFile>import_csv_module_test1.csv</importFile>
                        <displayName>Import CSV Module Test1</displayName>
                        <table>
                                <!-- Map de following CSV definition:
                                                DateTime;Status;Boolean;Integer;Float;Double;
String
                                                2003-02-22 00:00;Ok;yes;14;1.4;1.45;Hello world
1
                                                2003-02-23 00:00;Not Ok;;true;24;2.4;2.45;Hello
world 2
                                -->
                                <dateTimeColumn name="DateTime" displayName="datum" pattern="yyyy-MM-dd
HH:ss"/>
                                <stringColumn name="Status" displayName="status field"/>
                                <booleanColumn name="Boolean" displayName="Yes or No"/>
                                <integerColumn name="Integer" displayName="Int field"/>
                                <doubleColumn name="Double" displayName="Double field"/>
                                <stringColumn name="String" displayName="String field"/>
                        </table>
                </importCsvModuleRunTablesActivity>
        </importActivities>
```

If errors have been made in the mapping of the csv table, one warning will be logged per column to avoid flooding the log file with warnings. The import will always continue and a empty value will be set. The exceptions are Booleans that will be set to false and dateTimes will be set to 0.

Shutdown Activities



Elements of the Shutdown Activities configuration
This activity is the identical to the startUpActivities. The only difference is that these are carried out after the module run and import of data. See definition of StartUp activities for configuration.

ignoreNonExistingLocationSets

<command> <className>nl.wldelft.fews.adapter.urbsiniadapter.UrbsIniAdapter</className> <customJreDir>%ROOT_DIR%/jre</customJreDir> <jvmArg>-mx512m</jvmArg> </command>