20 Transformation Module - Improved schema

What	*nameofinstance*.xml
Description	Configuration for the new version of the transformation module
schema location	https://fewsdocs.deltares.nl/schemas/version1.0/transformationModule.xsd

Contents

Transformation Module Configuration (New Version)

The Transformation module is a general-purpose module that allows for generic transformation and manipulation of time series data. The module may be configured to provide for simple arithmetic manipulation, time interval transformation, shifting the series in time etc, as well as for applying specific hydro-meteorological transformations such as stage discharge relationships etc.

An improvement version of the FEWS Transformation Module is currently under construction. The new version is much more easy to configure than the old version. The new version uses a new schema for configuration, also several new transformations are added.

Configuration

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

TransformHBV_Inputs 1.00 default.xml.

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

The configuration for the transformation module consists of two parts: transformation configuration files in the Config/ModuleConfigFiles directory and coefficient set configuration files in the Config/CoefficientSetsFiles directory.

In a transformation configuration file one or more transformations can be configured. Some transformations require coefficient sets in which given coefficients are defined. For a given transformation that requires a coefficient set there are different ways of defining the coefficient set in the configuration. One way is to specify an embedded coefficient set in the transformation configuration itself. Another way is to put a reference in the transformation configuration. This reference consists of the name of a separate coefficient set configuration file and the id of a coefficient set in that file.

Both the transformations and coefficient sets can be configured to be time dependent. This can be used for instance to define a given coefficient value to be 3 from 1 January 2008 to 1 January 2009, and to be 4 from 1 January 2009 onwards. This can be done by defining multiple periodCoefficientSets, each one with a different period, as in the following xml example.

```
<periodCoefficientSet>
  <period>
    <startDateTime date="2008-01-01" time="00:00:00"/>
    <endDateTime date="2009-01-01" time="00:00:00"/>
  </period>
  <structure>
    <pumpFixedDischarge>
      <discharge>3</discharge>
    </pumpFixedDischarge>
  </structure>
</periodCoefficientSet>
<periodCoefficientSet>
  <period>
    <validAfterDateTime date="2009-01-01"/>
  </period>
  <structure>
    <pumpFixedDischarge>
      <discharge>4</discharge>
    </pumpFixedDischarge>
  </structure>
</periodCoefficientSet>
```

If a date is specified without a time, then the time is assumed to be 00:00:00, so <validAfterDateTime date="2009-01-01"/> is the same as <validAfterDateTime date="2009-01-01" time="00:00:00"/>. To specify dates and times in a particular time zone use the optional time zone element at the beginning of a transformations or a coefficient sets configuration file, e.g. <timeZone>GMT+5:00</timeZone>. Then all dates and times in that configuration file are in the defined time zone. If no time zone is defined, then dates and times are in GMT. Note: 2008-06-20 11:33:00 in time zone GMT+5:00 is physically the same time as 2008-06-20 06:33:00 in GMT.

If for a given transformation there are different coefficientSets configured for different periods in time, then the following rule is used. The start of a period is always inclusive. The end of a period is exclusive if another period follows without a gap in between, otherwise the end of the period is inclusive. If for example there are three periodCoefficientSets defined (A, B and C), each with a different period, as in the following xml example. Then at 2002-01-01 00:00:00 periodCoefficientSet A is valid. At 2003-01-01 00:00:00 periodCoefficientSet B is valid since the start of the period is inclusive. At 2004-01-01 00:00:00 periodCoefficientSet B is still valid, since there is a gap after 2004-01-01 00:00:00. At 2011-01-01 00:00:00 periodCoefficientSet C is valid, since no other periods follow (the period of C is the last period in time that is defined). This same rule applies to time-dependent transformations.

```
<periodCoefficientSet>
  <!-- periodCoefficientSet A -->
  <period>
    <startDateTime date="2002-01-01" time="00:00:00"/>
    <endDateTime date="2003-01-01" time="00:00:00"/>
  </period>
  ...
</periodCoefficientSet>
<periodCoefficientSet>
  <!-- periodCoefficientSet B -->
  <period>
    <startDateTime date="2003-01-01" time="00:00:00"/>
    <endDateTime date="2004-01-01" time="00:00:00"/>
  </period>
  ...
</periodCoefficientSet>
<periodCoefficientSet>
  <!-- periodCoefficientSet C -->
  <period>
    <startDateTime date="2010-01-01" time="00:00:00"/>
    <endDateTime date="2011-01-01" time="00:00:00"/>
  </period>
  ...
</periodCoefficientSet>
```

Validation rules

The concept of the validation rules was introduced as a solution for a common problem in operational situations when using aggregation transformations. When for example an aggregation was done over an entire year a single missing value in the input values would cause that the yearly average was also a missing value.

The validation rules provide a solution for these types of situations. It allows to configure in which cases an output value should be computed although the input contains missing values and/or doubtful values.

The validation rules are optional in the configuration and can be used to define the outputflag and the custom flagsource of the output value based on the number of missing values/unreliables values and/or the number of doubtful values in the used input values **per aggregation timestep**. The available output flags are reliable, doubtful and missing.

With these rules it is possible to define for example that the output of the transformation is reliable if less than 10% of the input is unreliable and/or missing and that if this percentage is above 10% that in that case the output should be a missing value.

It is important to note that input values which are missing and input values which are marked as unreliable are treated the same. Both are seen as missing values by the validation rules.

Below the configuration of the basic example which was described above.

```
<validationRule>
        <inputMissingPercentage>10</inputMissingPercentage>
        <outputValueFlag>reliable</outputValueFlag>
</validationRule>
<validationRule>
        <inputMissingPercentage>100</inputMissingPercentage>
        <outputValueFlag>missing</outputValueFlag>
</validationRule>
```

The configured validation rules are applied in the following way. The first validation rules are applied first. In the example above the first rule is that if 10% or less of the input is missing (or unreliable) that the output flag will be set to reliable. If the input doesn't meet the criteria for the first rule the transformation module will try to apply the second rule. In this case the second rule will always apply because a percentage of 100% is configured.

Configuring a rule with a percentage of 100% is a recommended way of configuring the validation rules. By default if validation rules are configured and none of the configured rules are valid the output will be set to missing. Which means that in this case the second rule of 100% was not necessary because it is also the default hard-coded behaviour of the system.

But for the users of the system it is more understandable if the behaviour of the aggregation is configured instead of a hard-coded fallback mechanism in the software.

To explain the validation rules a bit more a more difficult example will explained. Let's say that we would like configure our aggregation in such a way that the following rules are applied:

1 if the percentage of missing and/or unreliable values is less than 15% the output should be reliable.
2 if the percentage of missing values is less than 40% the output should be doubtful.
3 in all other cases the output should be a missing value.

Below shows a configuration example in which the rules above are implemented.

```
<validationRule>
  <inputMissingPercentage>15</inputMissingPercentage>
  <outputValueFlag>reliable</outputValueFlag>
</validationRule>
<validationRule>
  <inputMissingPercentage>40</inputMissingPercentage>
  <outputValueFlag>doubtful</outputValueFlag>
</validationRule>
<validationRule>
  <inputMissingPercentage>100</inputMissingPercentage>
  <outputValueFlag>missing</outputValueFlag>
</validationRule>
```

The example shows that in total 3 validation rules were needed. The first rule checks if less than 15% of the input is missing/unreliable. If this is not the case than it will be checked if the second rule can be applied. The second rule states that if less than 40% of the input is missing that in that case the output flag should be set to doubtful. The last rule takes care of all the other situations. Note that it has a percentage configured of 100%. Which means that this rule will be applied. However because 2 rules are defined above this rule FEWS will always try to apply these rules first before applying this rule.

In some cases one would like to differ between situations in which the outputflag is the same. In the example above if all of the input values were reliable the output is marked as reliable. But if for example 10% of the input values were unreliable the output is also marked as reliable.

It would be nice if the user of the system would be able to see in the GUI of FEWS why the input was marked reliable. Were there missing values in the input or not? Is the output based on a few missing values?

To make this possible the concept of the custom flag source was added to the validation rules. In addition to configuring an output flag it is also possible to configure a custom flag source. In the table of the Timeseriesdialog the custom flag source can be made visible by pressing ctrl + shift + j. This will make a new column in the table visible in which the custom flag source ids are shown. In the graph itself it also possible to make the custom flag sources visible by pressing ctrl + alt + v. To use the custom flagsources a file CustomFlagSources.xml should be added to the RegionConfig directory. For details see 27 CustomFlagSources In this file the custom flag sources should be defined. By configuring several rules which has the same outputflag but a different custom flagsource it is possible to make a difference between situations in which the outputflag is the same.

Below an example in which the output is reliable when there are no missing values in the input and when the percentage if missing values is less than 15%. However in the first case the output doesn't get a custom flagsource assigned while in the second case the output gets a custom flagsource assigned which is visible in the GUI to indicate that a output value was calculated but that missing values were found in the input.

```
<validationRule>
  <inputMissingPercentage>0</inputMissingPercentage>
  <outputValueFlag>reliable</outputValueFlag>
</validationRule>
<validationRule>
  <inputMissingPercentage>15</inputMissingPercentage>
  <outputValueFlag>reliable</outputValueFlag>
  <outputCustomFlagSourceId>CA</outputCustomFlagSourceId>
</validationRule>
<validationRule>
  <inputMissingPercentage>40</inputMissingPercentage>
  <outputValueFlag>doubtful</outputValueFlag>
</validationRule>
<validationRule>
  <inputMissingPercentage>100</inputMissingPercentage>
  <outputValueFlag>missing</outputValueFlag>
</validationRule>
```

Finally it is also possible to define validation rules based on the number of doubtful values in the input. It is important to note that missing values and unrealiable values found in the input are also counted as doubtful values. It is even possible to define validation rules based on a combination of an allowed percentage of unreliable/missing values and doubtful values. The sequence of applying the rules is also in this case the order in which the rules are configured. The first rule which applies to the
current situation is used.

Let's say for example that we also want rules to be defined for the doubtful input values. For example when only a small number of input values are doubtful we still want the output to be reliable. Otherwise we would like to have the output to be doubtful but with an custom flag source which give us an indication of how many of the input values were doubtful.

Below a configuration example

```
<validationRule>
  <inputDoubtfulPercentage>10</inputDoubtfulPercentage>
  <inputMissingPercentage>0</inputMissingPercentage>
  <outputValueFlag>reliable</outputValueFlag>
</validationRule>
<validationRule>
  <inputDoubtfulPercentage>30</inputDoubtfulPercentage>
  <inputMissingPercentage>0</inputMissingPercentage>
  <outputValueFlag>doubtful</outputValueFlag>
  <outputCustomFlagSourceId>D1</outputCustomFlagSourceId>
</validationRule>
<validationRule>
  <inputDoubtfulPercentage>60</inputDoubtfulPercentage>
  <inputMissingPercentage>0</inputMissingPercentage>
  <outputValueFlag>doubtful</outputValueFlag>
  <outputCustomFlagSourceId>D2</outputCustomFlagSourceId>
</validationRule>
<validationRule>
  <inputDoubtfulPercentage>100</inputDoubtfulPercentage>
  <inputMissingPercentage>0</inputMissingPercentage>
  <outputValueFlag>doubtful</outputValueFlag>
  <outputCustomFlagSourceId>D3</outputCustomFlagSourceId>
</validationRule>
<validationRule>
  <inputMissingPercentage>15</inputMissingPercentage>
  <outputValueFlag>reliable</outputValueFlag>
  <outputCustomFlagSourceId>CA</outputCustomFlagSourceId>
</validationRule>
<validationRule>
  <inputMissingPercentage>40</inputMissingPercentage>
  <outputValueFlag>doubtful</outputValueFlag>
</validationRule>
<validationRule>
  <inputMissingPercentage>100</inputMissingPercentage>
  <outputValueFlag>missing</outputValueFlag>
</validationRule>
```

The explanation above gave an good idea of the possibilities of the use of the validation rules.

In the examples above the inputMissingValuePercentage and the inputDoubtfulPercentage was configured hard-coded in the configuration file. However it is also possible to make a reference to an attribute of a location. To reference to an attribute the referenced attribute should be placed within @.

```
<inputMissingPercentage>@MV@</inputMissingPercentage>
```

For example to reference the attribute MV for the inputMissingValuePercentage the configuration should be like.

To explain the concept of the validation rules more the table below shows the input time series and the output time series of an aggregation accumulative tranformation which uses the validation rules which are shown above in the last eexample.

Time	Input value	input flag	Output value	output flag	custom flagsource
1-1-2012 00:15					
1-1-2012 00:30	1				
1-1-2012 00:45	1				
1-1-2012 01:00	1		3	doubtful	-
1-1-2012 01:15					
1-1-2012 01:30	1				
1-1-2012 01:45					
1-1-2012 02:00	1		NaN	-	-
1-1-2012 02:15	1				

1-1-2012 02:30	1	doubtful			
1-1-2012 02:45	1				
1-1-2012 03:00	1		4	doubtful	D1
1-1-2012 03:15	1				
1-1-2012 03:30	1				
1-1-2012 03:45	1				
1-1-2012 04:00	1		4	reliable	

The first output value is set to doubtful. Because in this case the total percentage of missing values is 25%. Which means that the following rule is applied.

```
<validationRule>
  <inputMissingPercentage>40</inputMissingPercentage>
  <outputValueFlag>doubtful</outputValueFlag>
</validationRule>
```

The second output value is a missing value because in this case the percentage of missing values is equal to 50%. This means that in this case the following rule will be appplied.

```
<validationRule>
  <inputMissingPercentage>100</inputMissingPercentage>
  <outputValueFlag>missing</outputValueFlag>
</validationRule>
```

The third output value is set to doubtful. The input doesn't contain missing values but has a single doubtful input value. The percentage of doubtful values in the input is therefore 25% which means that the following rule will be applied.

```
<validationRule>
  <inputDoubtfulPercentage>30</inputDoubtfulPercentage>
  <inputMissingPercentage>0</inputMissingPercentage>
  <outputValueFlag>doubtful</outputValueFlag>
  <outputCustomFlagSourceId>D1</outputCustomFlagSourceId>
</validationRule>
```

the fourth and last output value has set to reliable with no output flag. In this case all in the input values are reliable. In this case the first rule is applied (in the case when all of the input values are reliable the first rule always applies).

Manual Edits

Since FEWS 2017.02 it is possible to configure if manual edits should be preserved. This setting applies to all transformations that are configured. The default is false. For an example configuration see:

```
<transformationModule xmlns="http://www.wldelft.nl/fews" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:schemaLocation="http://www.wldelft.nl/fews https://fewsdocs.deltares.nl/schemas/version1.0
/transformationModule.xsd" version="1.0">
   <preserveManualEdits>true</preserveManualEdits>
```

Description

Since 2019.02 an optional field description is available. This field can be configured per transformation, and the text will be shown in the workflow tree asmouse ower label (tooltip). This can be used with any type of transformation.

Example:

```
<transformation id="merge">
        <merge>
                <simple>
                        <inputVariable>
                                <variableId>Wiski</variableId>
                        </inputVariable>
                        <inputVariable>
                                <variableId>Server</variableId>
                        </inputVariable>
                        <fillGapConstant>0</fillGapConstant>
                        <outputVariable>
                                <variableId>merge1</variableId>
                        </outputVariable>
                </simple>
        </merge>
        <description>transformation description</description>
</transformation>
```

Copying comments, flags and flag sources from input to output

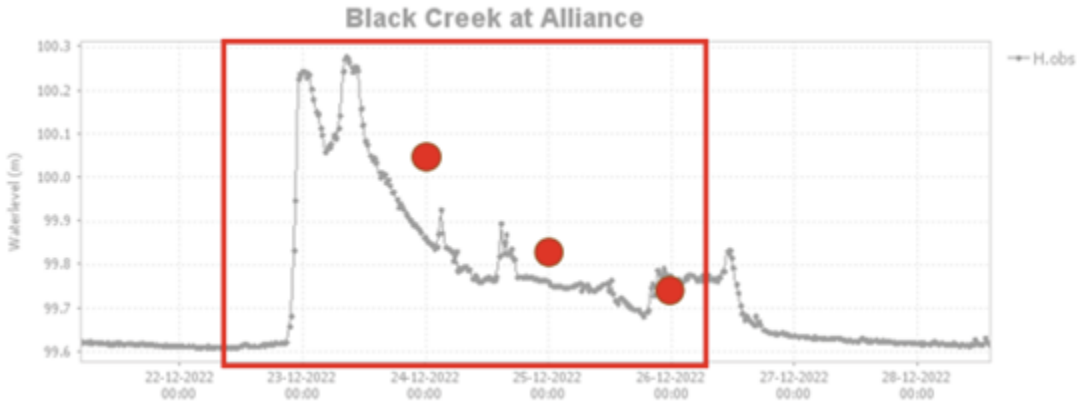
<TODO>

Preventing previously calculated values to be overwritten with missings

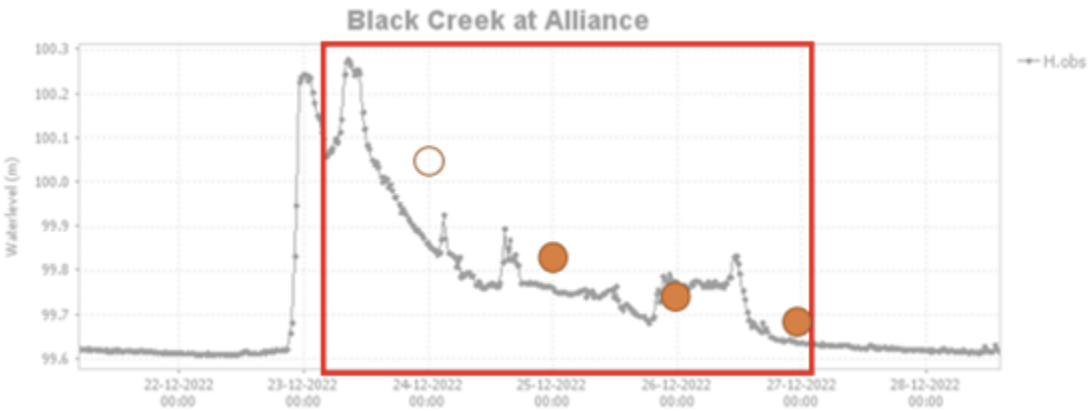
Delft-FEWS processes data in moving windows compared to the timezero of the workflow. These workflows can be run several times per day. In certain conditions this could lead a transformation to calculate a missing value for a datetime for which earlier a correct value was calculated. This can lead to warnings such as:

```
Existing value overwritten by missing
Suspicious write action. Long time series written with only changes at the start and at the end. If this happens often this will explode the database.
```

This mechanism is illustrated with the image below, which shows a water level for which an average per day is calculated. The box shows the moving window (relativeViewPeriod). The image shows the daily values (red dots) calculated in the first run.



In a later run (illustrated below), there are not sufficient values to calculate a daily value for 24 December. So this run will return a missing value for 24 December (while calculating a new value for 27 December.



If these transformations would directly write to the same output timeserie (in this case with *timeseriesType "external historical")*, the later run would cause the average water level for December 24 to be overwritten with a missing value. The log would include warnings about this!

To prevent this, the output timeserie should be of *timeserieType "temporary"*, and next be merged (merge / simple transformation) with the final timeserie. This way, missing values will not overwrite previously calculated values.

Run transformations for a set of selected locations

In some cases it is usefull to run a transformation only for a specific set of locations. For example when the entire workflow has already ran and there is only a change at a specific location. This situation can occur, for example, when a water level has been editted or when the configuration is changed.

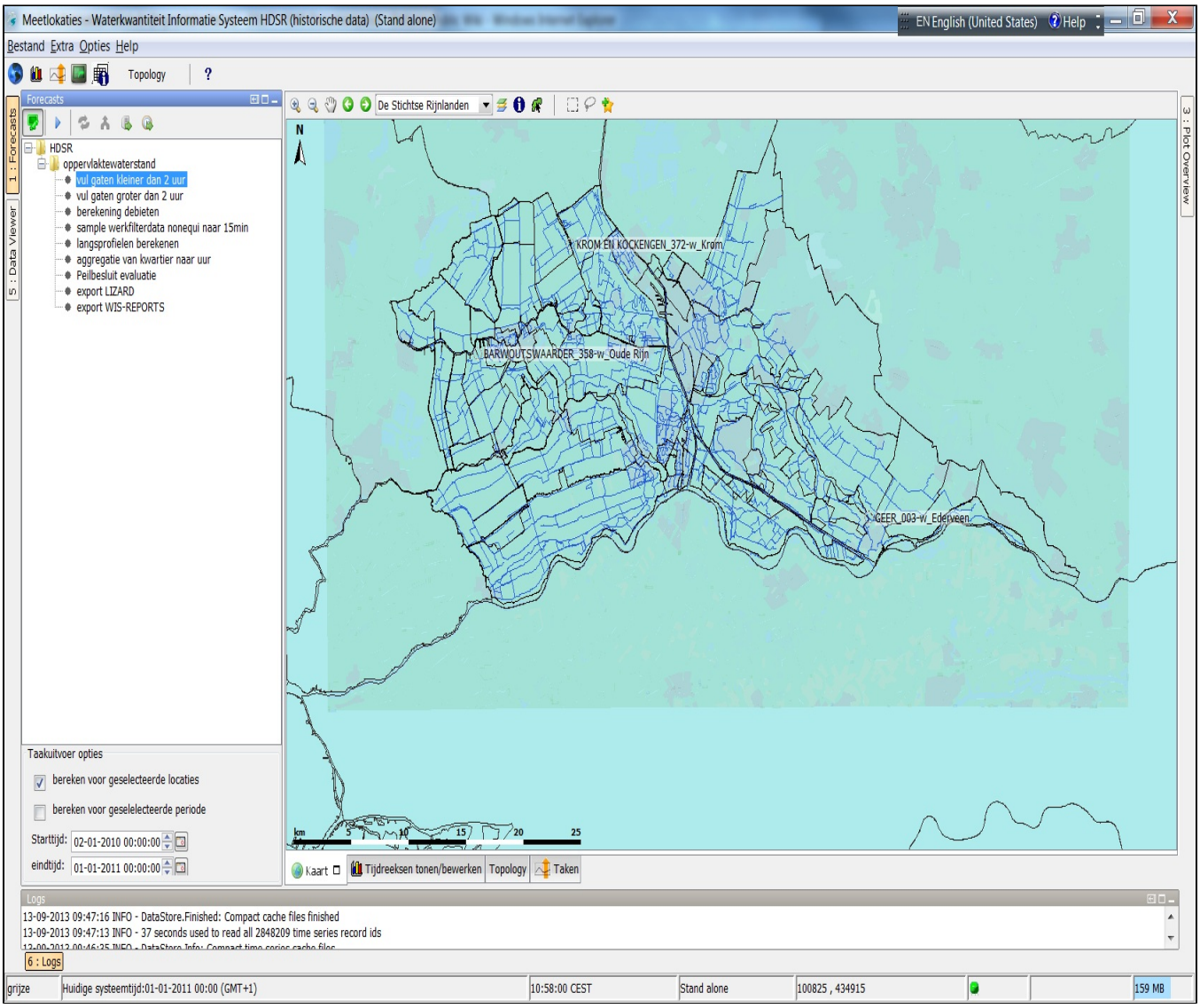
In this case the workflow can skip the calculations for the unchanged locations. The main benefit of this approach is that it saves a lot of processing time.

This functionality is now available in FEWS.However it is important to understand that this functionality cannot be used in all workflows. The functionality can be applied for transformations only. It cannot be used for running models or secondary validations. When a transformation is started for a location selection than the transformation will only start when the location of one of the input time series is selected. When a transformation has created output for a location which was not selected by the user than this location whill be added to the selection.

It is possible to run a workflow for a selected set of locations from the IFD, the task dialog and the manual forecast dialog. By default workflows cannot be run for a selected set of locations. To enable this the option allowSelection should be set to true in the workflowdescriptor of the workflow. Below an example.

```
<workflowDescriptor id="FillRelations" forecast="false" visible="true"autoApprove="false">
  <description>Met deze taak worden de gaten groter dan 2 uur gevuld dmv. relaties.</description>
  <allowSelection>true</allowSelection>
  <schedulingAllowed>true</schedulingAllowed>
</workflowDescriptor>
```

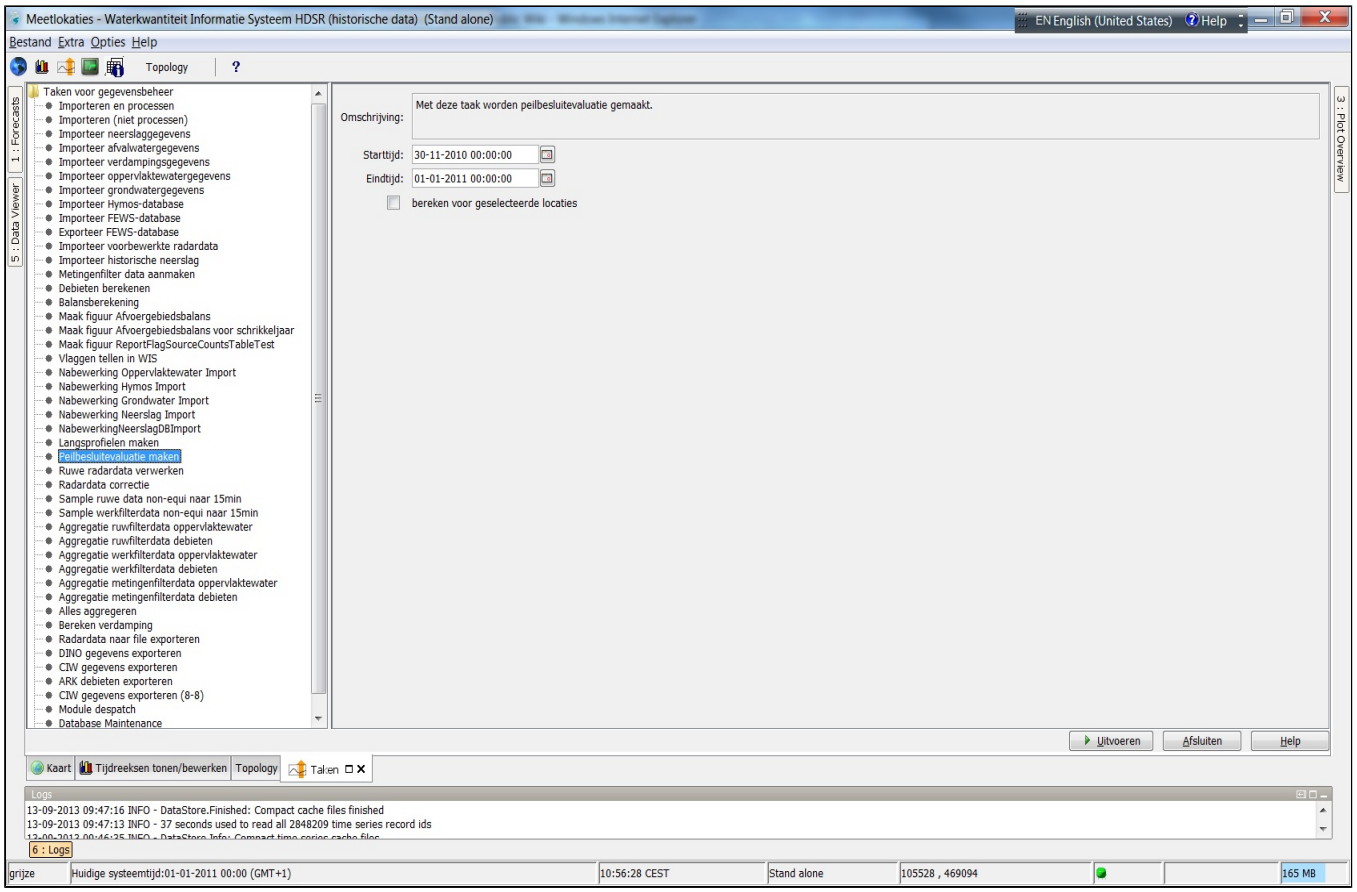
When a node in the IFD is selected with a workflow which has the allowSelection option set true, the GUI will look like this:



In the property dialog below the tree with the nodes two selection boxes will appear.

The first checkbox will enable the option to run a workflow for a specific set of locations. The second checkbox will enable to run the workflow for specified period.

In the taskrun dialog an additinal checkbox will appear.



Which locations should the user select?

The transformation will run for the selected locations. If one the input timeseries is selected in the filters or in the map the transformation will run.

This means that the user should select the locations which are changed. This can be a change in the data or a change in the configuration.

This can be explained with the use of a simple example. Lets say we have a system which has a workflow which consists of a user simple function which estimates the water level at location B by simply copying the water level at location A to location B.

After the copying a set of statistical transformations are run to compute statistics.

The user edits the water level at location A and want to recompute the water level at location B. However the workflow which does this, is configured to do similar estimates at another 500 locations. In this case the user should select location A.

When the run starts the majority of the calculations are skipped except when the water level for location will be recalculated because in this case location A which is one of the input time series is selected. When this calculation is done, FEWS will

remember that location B is now also changed and will add location B to the list of selected locations. When the statistical transformations are started after the water level at location B is recomputed the statistics for location B are also recalculated because in this case the transformation which recomputed the statistics for location B has a input time series with location B and because location B was added to the list of selected locations, the statistical transformations which calculates statistics for location B will also be started.

This functionality cannot be used for spatial transformations. Before enabling this option for a workflow, the configurator should check if the workflow contains spatial transformations.

In addition to the above, this functionality can only be used for non-forecast workflows. Typically this functionality should be used for pre-processing of post-processing.

Therefore it is by default not possible to run a workflow for a specific location selection. This is only possible when in the workflowdescriptors the option allowSelection is to true. This option should only be set to true when the configurator has checked that the workflow is suitable for running for a specific location.

Steps to follow when implementing selection specific calculations

The following steps should be followed when this functionality is implemented.
1) Decide in which situations this functionality is needed
2) Make a list of the workflows which need to run in this type of situations
3) Ensure that the workflow only consists of transformations for which this functionality can be used.
4) Move transformations or other parts of the workflow which are suitable for this type of operations to another workflow
5) Set the option allowSelection to true in the workflow descriptor for the workflow which can be used for selection specific calculations
6) When the workflows will be started from the taskrun dialog of the manual forecast dialog no additional configuration is needed. These displays are available in almost every FEWS system. However when the IFD will be used for this. the following additional steps should be taken.

Implement selecion specific calculations for IFD

First step is to create a topology.xml to configure the content of the tree from which the workflows should be started.

Detailed informations about configuring the topology.xml can be found at 24 Topology

The following steps should be done when using the IFD for selection specific calculations.

1. first create the tree structure by creating nodes in the topology.xml,
2. add workflows to the nodes.
3. add dependencies to the nodes by configuring the previous nodes
4. by default leaf nodes will run locally and not at the server. This is not desired in this case Therefore the option localRun should be set to false for the leafnodes.

Below an example (part of the topology.xml)

```
<nodes id="HDSR"><nodes id="oppervlaktewaterstand"><relativePeriod unit="week" start="-52" end="0" />
  <node id="vul gaten kleiner dan 2 uur">
   <previousNodeId>secondary validatie</previousNodeId>
   <workflowId>FillGap2H_WerkOpvlWater</workflowId>
   <filterId>Fillgap</filterId>
  <localRun>false</localRun>
 </node>
 <node id="vul gaten groter dan 2 uur">
  <previousNodeId>vul gaten kleiner dan 2 uur</previousNodeId>
  <workflowId>FillRelations</workflowId>
  <localRun>false</localRun>
 </node>
 <node id="berekening debieten">
  <previousNodeId>vul gaten groter dan 2 uur</previousNodeId>
  <workflowId>DebietBerekening</workflowId>
  <localRun>false</localRun>
 </node>
 <node id="sample werkfilterdata nonequi naar 15min">
  <previousNodeId>berekening debieten</previousNodeId>
  <workflowId>SampleRuwNaar15M</workflowId>
  <localRun>false</localRun>
 </node>
 <node id="langsprofielen berekenen">
  <previousNodeId>sample werkfilterdata nonequi naar 15min</previousNodeId>
  <workflowId>Langsprofiel</workflowId>
  <localRun>false</localRun>
 </node>
 <node id="aggregatie van kwartier naar uur">
  <previousNodeId>langsprofielen berekenen</previousNodeId>
  <workflowId>AggregeerWerkOpvlWater</workflowId>
  <localRun>false</localRun>
 </node>
 <node id="Peilbesluit evaluatie">
  <previousNodeId>aggregatie van kwartier naar uur</previousNodeId>
  <workflowId>PeilbesluitEvaluatie</workflowId>
  <localRun>false</localRun>
 </node>
 <node id="export LIZARD">
  <previousNodeId>Peilbesluit evaluatie</previousNodeId>
  <workflowId>ExportCIW</workflowId>
  <localRun>false</localRun>
 </node>
 <node id="export WIS-REPORTS">
  <previousNodeId>export LIZARD</previousNodeId>
  <workflowId>ExportCIW</workflowId>
  <localRun>false</localRun>
 </node>
</nodes>
```

second step is to add the following line the explorer.xml to add the IFD tool window to the system.

```
<explorerTask name="Forecasts">
  <predefinedDisplay>topology tree</predefinedDisplay>
  <toolbarTask>false</toolbarTask>
  <menubarTask>false</menubarTask>
  <toolWindow>true</toolWindow>
  <loadAtStartup>true</loadAtStartup>
</explorerTask>
```

Trim Output

A boolean option <trimOutput> is available within transformations. When true, missing values at the start and end of the output will be removed before writing the data to the database. This can prevent existing values to be overwritten with missings.

Forecast Loop

For some transformations it is possible to define a forecast loop by configuring a <forecastLoopSearchPeriod>. This means that the transformation will be run for each forecast found within that period.

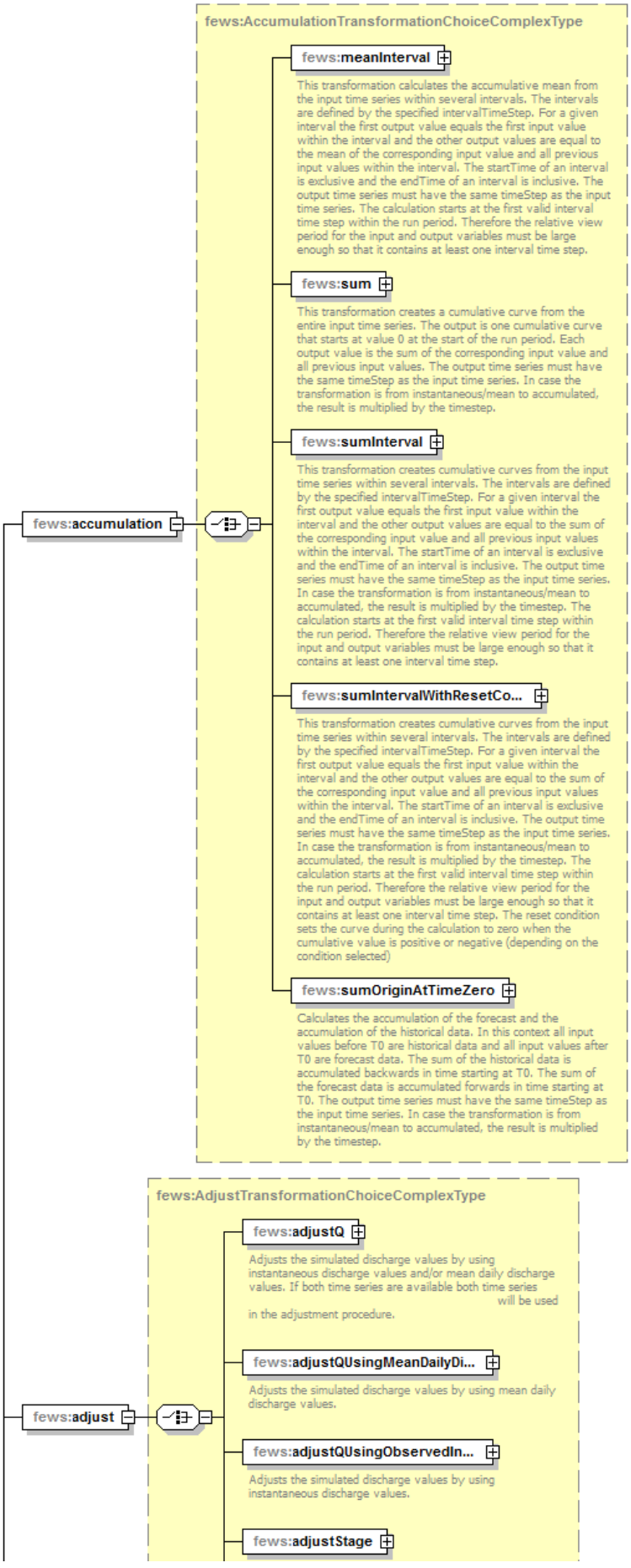
This will only work when the <inputVariable> and <outputVariable> are external forecasts. The output variable will get the same external forecast time as the input time series.

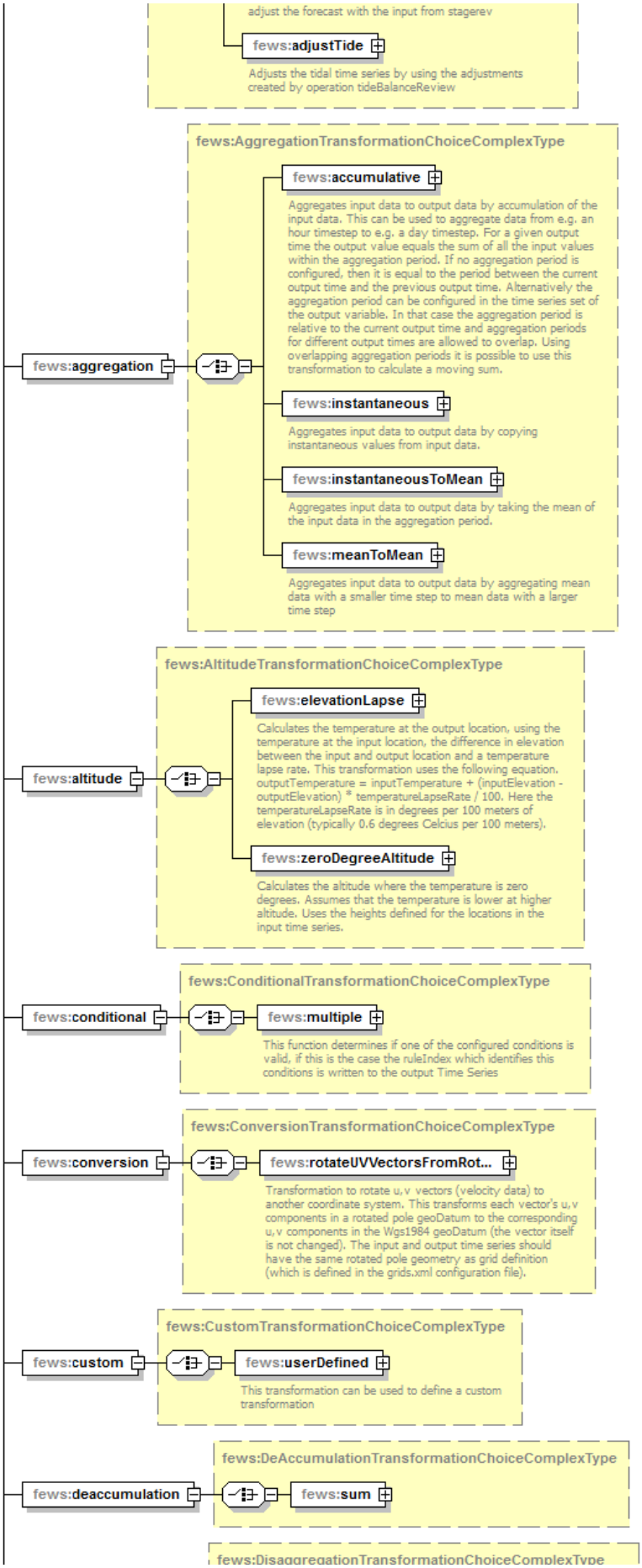
When this is configured in combination with a locationSet it will try to run the transformation for the maximum number of forecasts available for the locations. If for a location some forecasts are unavailable a warning will be logged and those transformation runs will be skipped for that forecast and location combination.

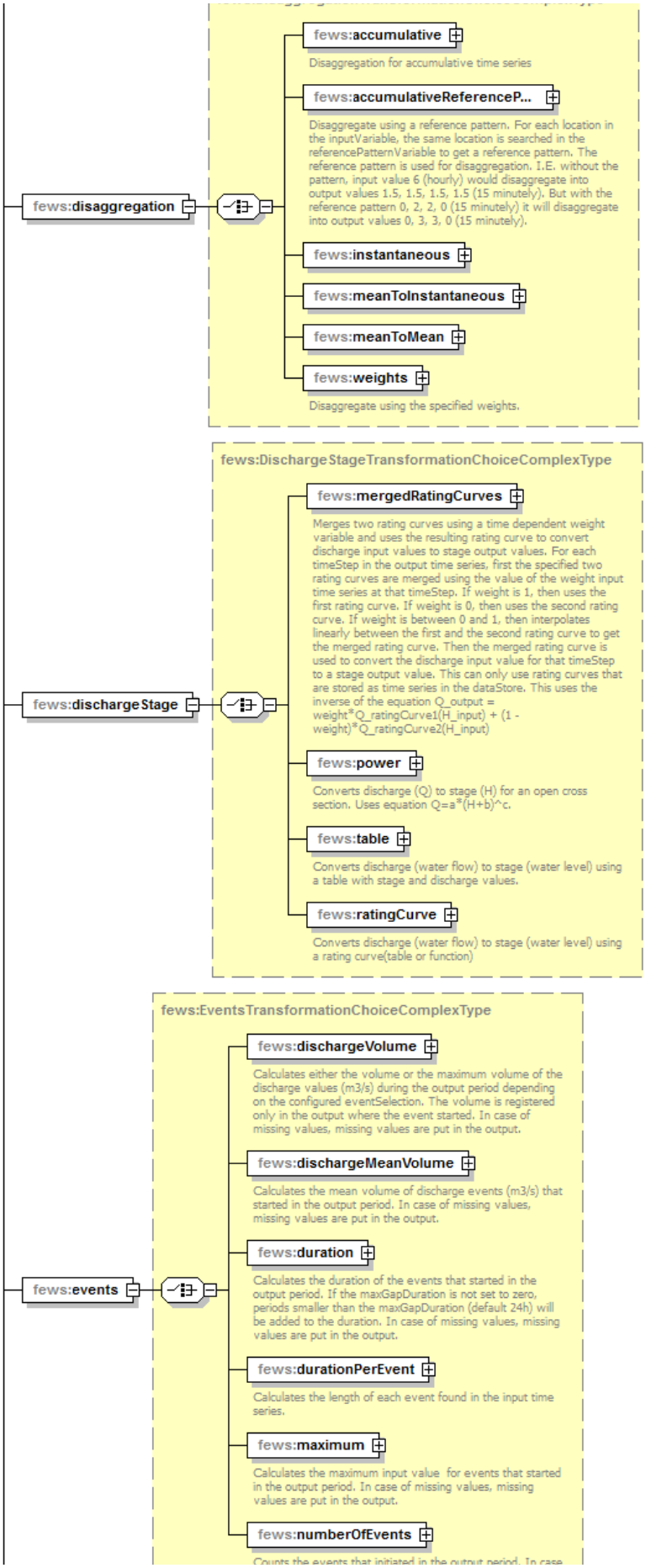
List of all available transformations

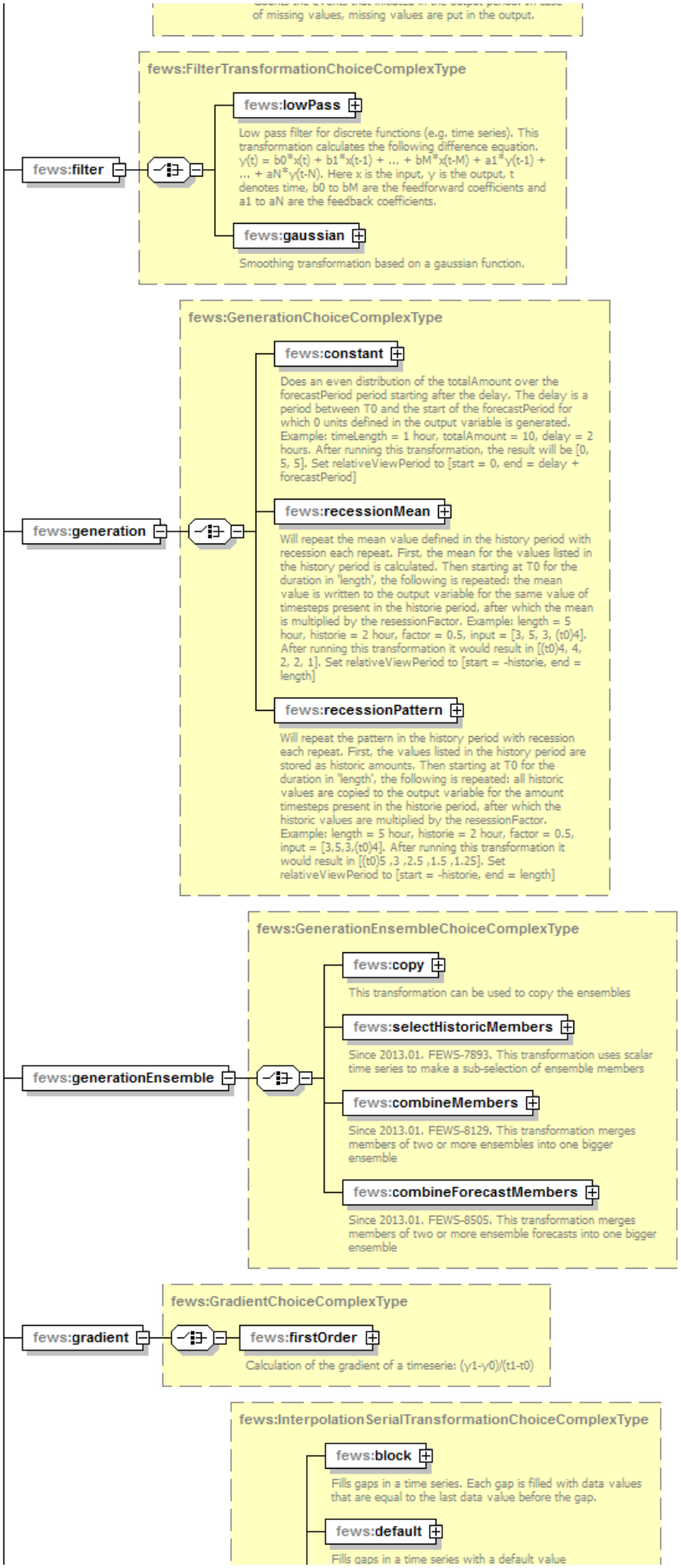
For the most recent development version see the xsd schema at https://fewsdocs.deltares.nl/schemas/version1.0/transformationTypes.xsd

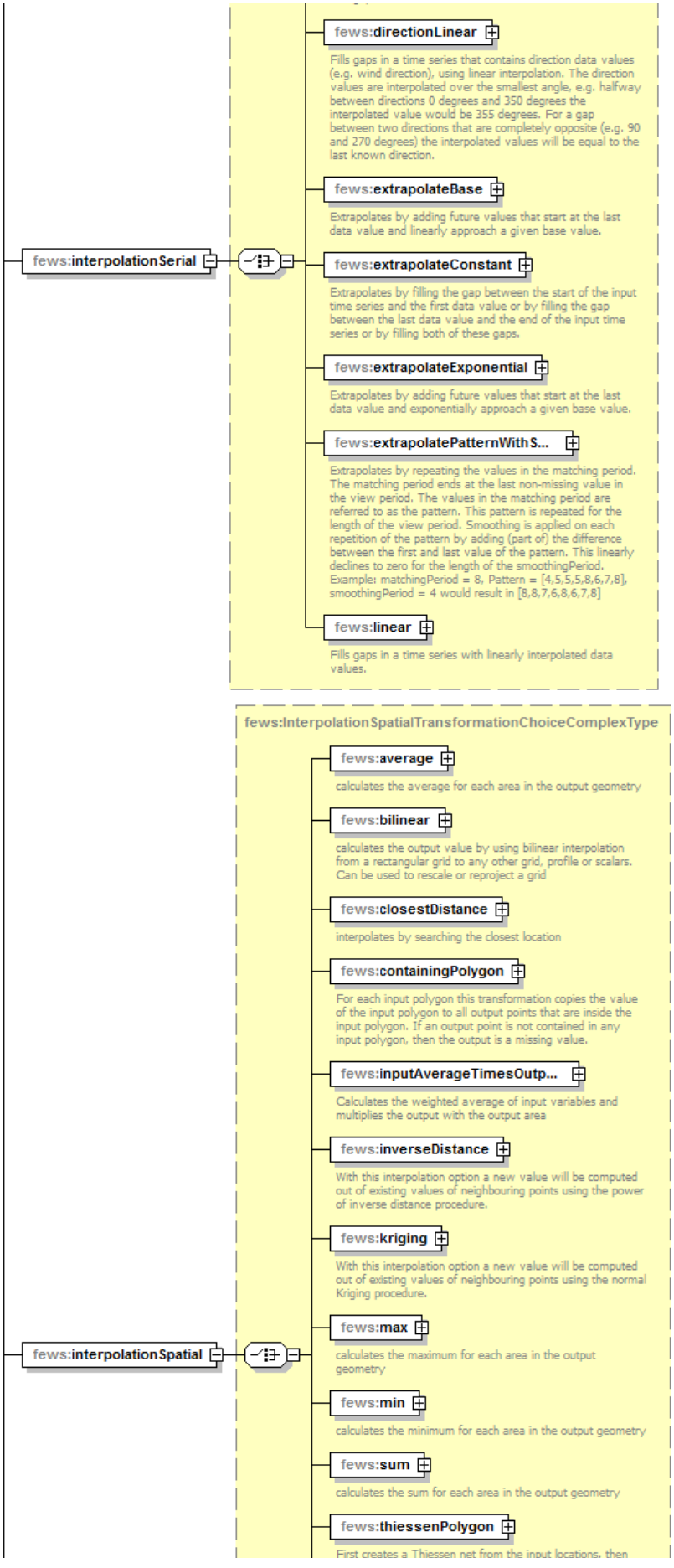
Available since stable build 2014.01:

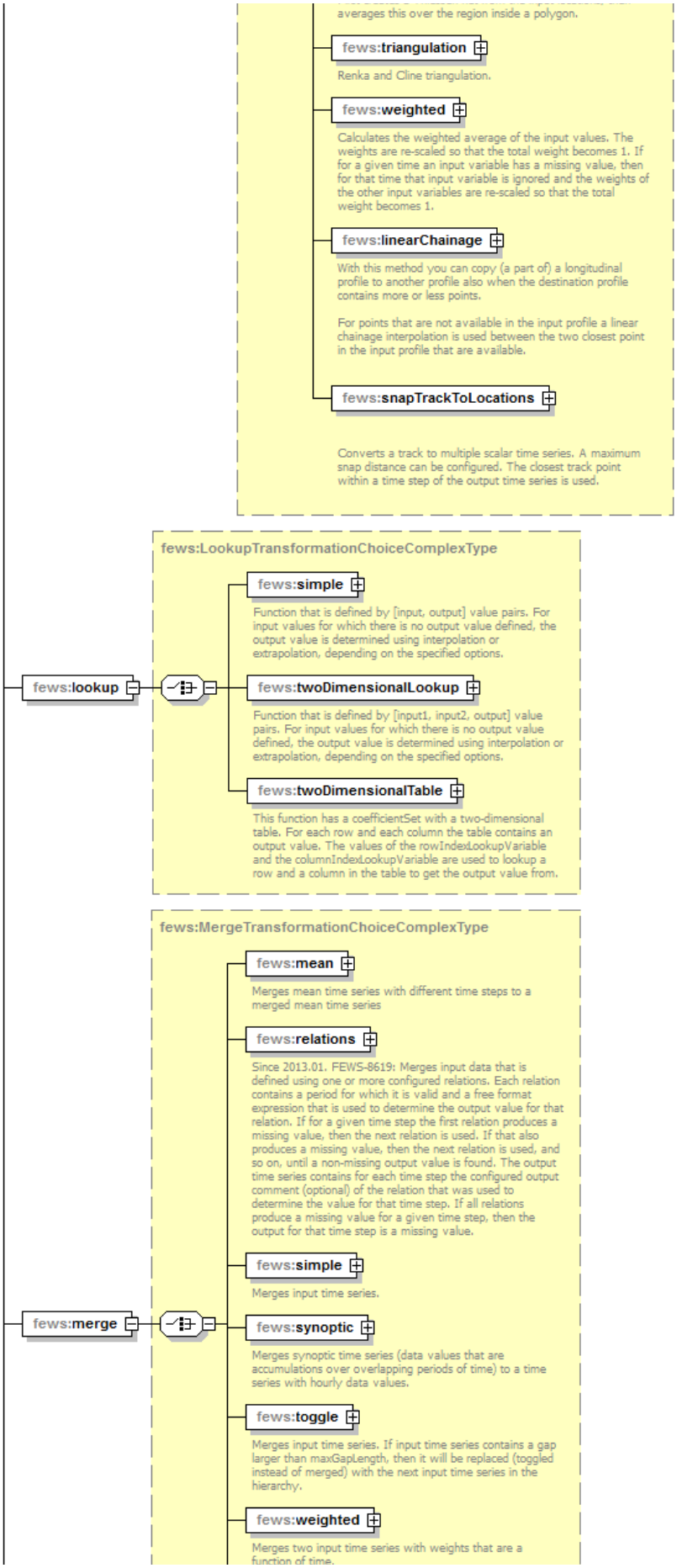

fews:accumulation
fews:AccumulationTransformationChoiceComplexType
fews:meanInterval
This transformation calculates the accumulative mean from the input time series within several intervals. The intervals are defined by the specified intervalTimeStep. For a given interval the first output value equals the first input value within the interval and the other output values are equal to the mean of the corresponding input value and all previous input values within the interval. The startTime of an interval is exclusive and the endTime of an interval is inclusive. The output time series must have the same timeStep as the input time series. The calculation starts at the first valid interval time step within the run period. Therefore the relative view period for the input and output variables must be large enough so that it contains at least one interval time step.
fews:sum
This transformation creates a cumulative curve from the entire input time series. The output is one cumulative curve that starts at value 0 at the start of the run period. Each output value is the sum of the corresponding input value and all previous input values. The output time series must have the same timeStep as the input time series. In case the transformation is from instantaneous/mean to accumulated, the result is multiplied by the timestep.
fews:sumInterval
This transformation creates cumulative curves from the input time series within several intervals. The intervals are defined by the specified intervalTimeStep. For a given interval the first output value equals the first input value within the interval and the other output values are equal to the sum of the corresponding input value and all previous input values within the interval. The startTime of an interval is exclusive and the endTime of an interval is inclusive. The output time series must have the same timeStep as the input time series. In case the transformation is from instantaneous/mean to accumulated, the result is multiplied by the timestep. The calculation starts at the first valid interval time step within the run period. Therefore the relative view period for the input and output variables must be large enough so that it contains at least one interval time step.
fews:sumIntervalWithResetCo...
This transformation creates cumulative curves from the input time series within several intervals. The intervals are defined by the specified intervalTimeStep. For a given interval the first output value equals the first input value within the interval and the other output values are equal to the sum of the corresponding input value and all previous input values within the interval. The startTime of an interval is exclusive and the endTime of an interval is inclusive. The output time series must have the same timeStep as the input time series. In case the transformation is from instantaneous/mean to accumulated, the result is multiplied by the timestep. The calculation starts at the first valid interval time step within the run period. Therefore the relative view period for the input and output variables must be large enough so that it contains at least one interval time step. The reset condition sets the curve during the calculation to zero when the cumulative value is positive or negative (depending on the condition selected)
fews:sumOriginAtTimeZero
Calculates the accumulation of the forecast and the accumulation of the historical data. In this context all input values before T0 are historical data and all input values after T0 are forecast data. The sum of the historical data is accumulated backwards in time starting at T0. The sum of the forecast data is accumulated forwards in time starting at T0. The output time series must have the same timeStep as the input time series. In case the transformation is from instantaneous/mean to accumulated, the result is multiplied by the timestep.
fews:adjust
fews:AdjustTransformationChoiceComplexType
fews:adjustQ
Adjusts the simulated discharge values by using instantaneous discharge values and/or mean daily discharge values. If both time series are available both time series will be used in the adjustment procedure.
fews:adjustQUsingMeanDailyDi...
Adjusts the simulated discharge values by using mean daily discharge values.
fews:adjustQUsingObservedIn...
Adjusts the simulated discharge values by using instantaneous discharge values.
fews:adjustStage

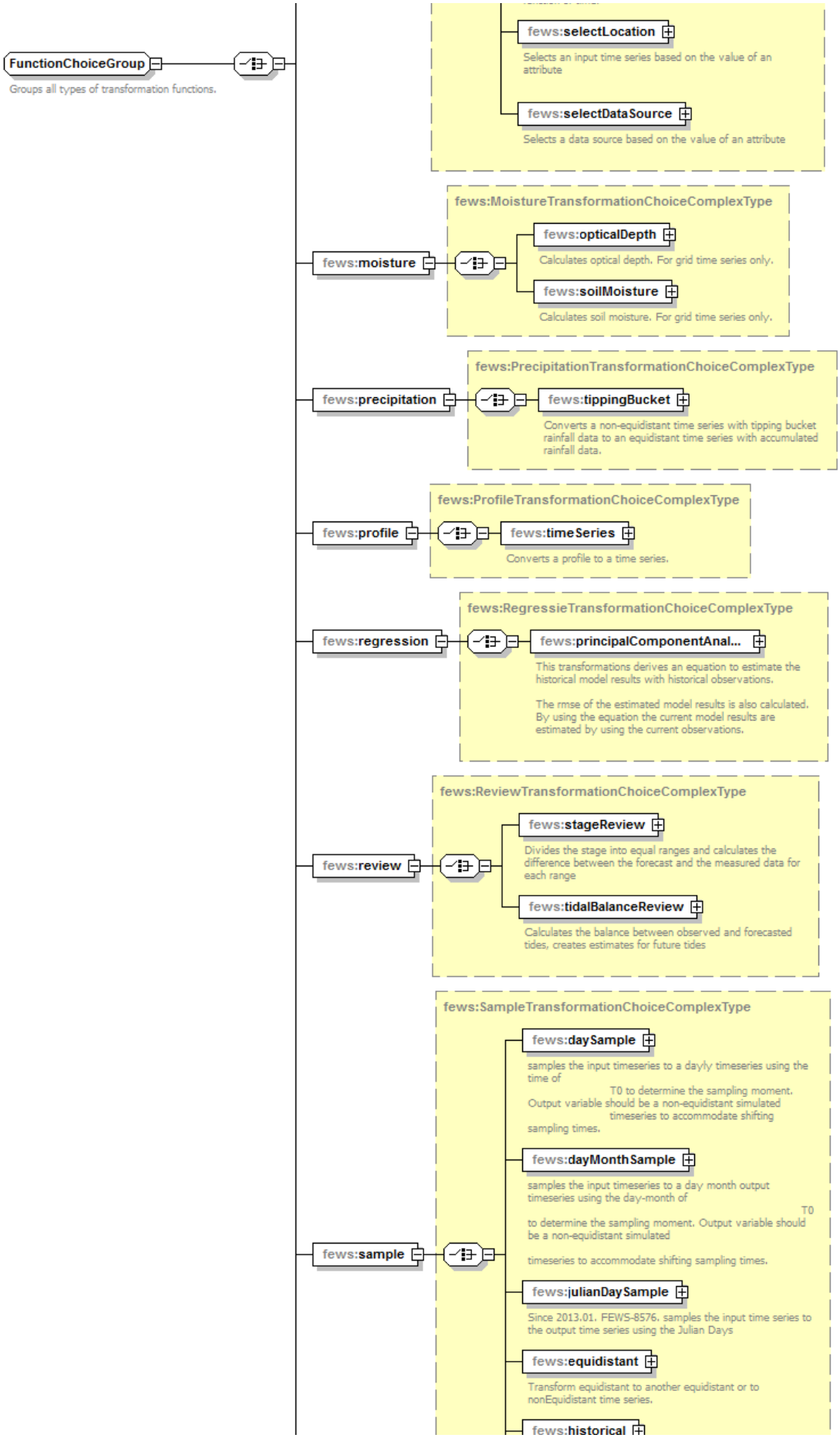

adjust the forecast with the input from stagerev
fews:adjustTide
Adjusts the tidal time series by using the adjustments created by operation tideBalanceReview
fews:AggregationTransformationChoiceComplexType
fews:aggregation
fews:accumulative
Aggregates input data to output data by accumulation of the input data. This can be used to aggregate data from e.g. an hour timestep to e.g. a day timestep. For a given output time the output value equals the sum of all the input values within the aggregation period. If no aggregation period is configured, then it is equal to the period between the current output time and the previous output time. Alternatively the aggregation period can be configured in the time series set of the output variable. In that case the aggregation period is relative to the current output time and aggregation periods for different output times are allowed to overlap. Using overlapping aggregation periods it is possible to use this transformation to calculate a moving sum.
fews:instantaneous
Aggregates input data to output data by copying instantaneous values from input data.
fews:instantaneousToMean
Aggregates input data to output data by taking the mean of the input data in the aggregation period.
fews:meanToMean
Aggregates input data to output data by aggregating mean data with a smaller time step to mean data with a larger time step
fews:AltitudeTransformationChoiceComplexType
fews:altitude
fews:elevationLapse
Calculates the temperature at the output location, using the temperature at the input location, the difference in elevation between the input and output location and a temperature lapse rate. This transformation uses the following equation. outputTemperature = inputTemperature + (inputElevation - outputElevation) * temperatureLapseRate / 100. Here the temperatureLapseRate is in degrees per 100 meters of elevation (typically 0.6 degrees Celcius per 100 meters).
fews:zeroDegreeAltitude
Calculates the altitude where the temperature is zero degrees. Assumes that the temperature is lower at higher altitude. Uses the heights defined for the locations in the input time series.
fews:ConditionalTransformationChoiceComplexType
fews:conditional
fews:multiple
This function determines if one of the configured conditions is valid, if this is the case the ruleIndex which identifies this conditions is written to the output Time Series
fews:ConversionTransformationChoiceComplexType
fews:conversion
fews:rotateUVVectorsFromRot...
Transformation to rotate u,v vectors (velocity data) to another coordinate system. This transforms each vector's u,v components in a rotated pole geoDatum to the corresponding u,v components in the Wgs1984 geoDatum (the vector itself is not changed). The input and output time series should have the same rotated pole geometry as grid definition (which is defined in the grids.xml configuration file).
fews:CustomTransformationChoiceComplexType
fews:custom
fews:userDefined
This transformation can be used to define a custom transformation
fews:DeAccumulationTransformationChoiceComplexType
fews:deaccumulation
fews:sum
fews:DisaggregationTransformationChoiceComplexType

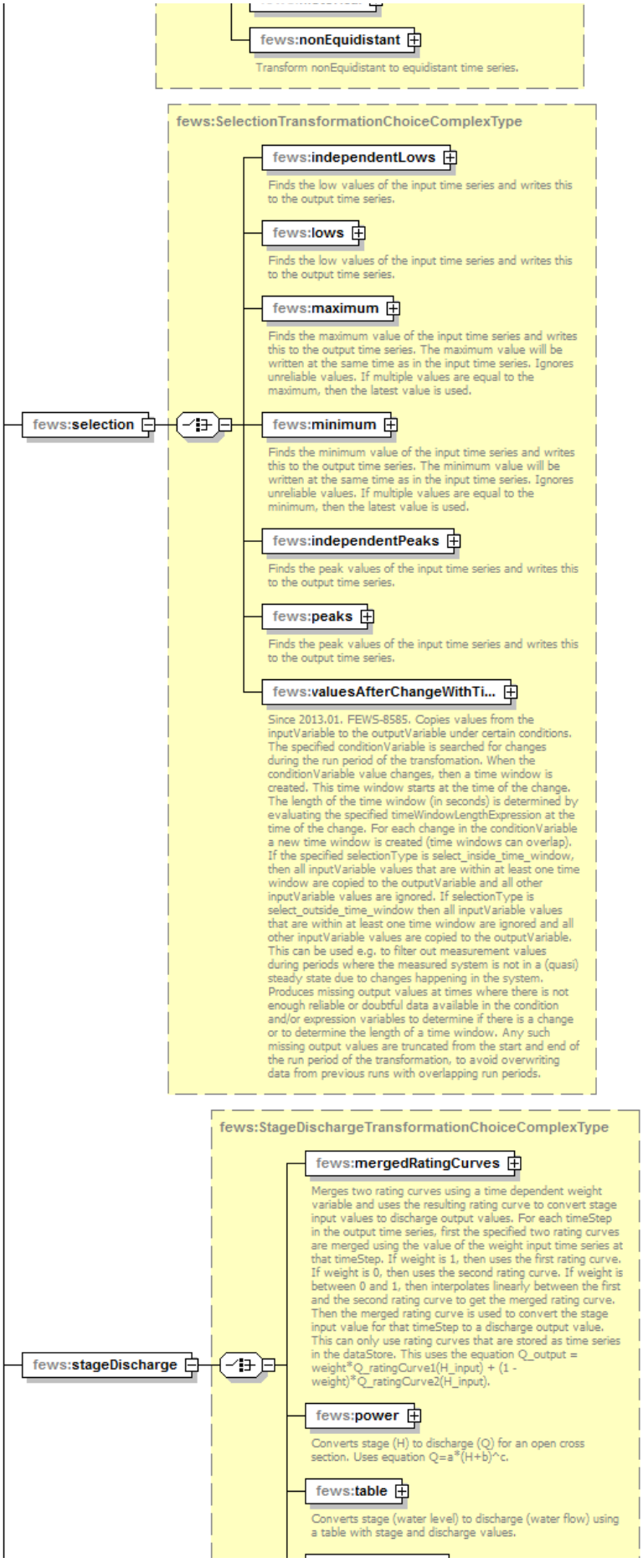

fews:disaggregation
fews:accumulative
Disaggregation for accumulative time series
fews:accumulativeReferenceP...
Disaggregate using a reference pattern. For each location in the inputVariable, the same location is searched in the referencePatternVariable to get a reference pattern. The reference pattern is used for disaggregation. I.E. without the pattern, input value 6 (hourly) would disaggregate into output values 1.5, 1.5, 1.5, 1.5 (15 minutely). But with the reference pattern 0, 2, 2, 0 (15 minutely) it will disaggregate into output values 0, 3, 3, 0 (15 minutely).
fews:instantaneous
fews:meanToInstantaneous
fews:meanToMean
fews:weights
Disaggregate using the specified weights.
fews:dischargeStage
fews:DischargeStageTransformationChoiceComplexType
fews:mergedRatingCurves
Merges two rating curves using a time dependent weight variable and uses the resulting rating curve to convert discharge input values to stage output values. For each timeStep in the output time series, first the specified two rating curves are merged using the value of the weight input time series at that timeStep. If weight is 1, then uses the first rating curve. If weight is 0, then uses the second rating curve. If weight is between 0 and 1, then interpolates linearly between the first and the second rating curve to get the merged rating curve. Then the merged rating curve is used to convert the discharge input value for that timeStep to a stage output value. This can only use rating curves that are stored as time series in the dataStore. This uses the inverse of the equation Q_output = weight*Q_ratingCurve1(H_input) + (1 - weight)*Q_ratingCurve2(H_input)
fews:power
Converts discharge (Q) to stage (H) for an open cross section. Uses equation Q=a*(H+b)^c.
fews:table
Converts discharge (water flow) to stage (water level) using a table with stage and discharge values.
fews:ratingCurve
Converts discharge (water flow) to stage (water level) using a rating curve(table or function)
fews:events
fews:EventsTransformationChoiceComplexType
fews:dischargeVolume
Calculates either the volume or the maximum volume of the discharge values (m3/s) during the output period depending on the configured eventSelection. The volume is registered only in the output where the event started. In case of missing values, missing values are put in the output.
fews:dischargeMeanVolume
Calculates the mean volume of discharge events (m3/s) that started in the output period. In case of missing values, missing values are put in the output.
fews:duration
Calculates the duration of the events that started in the output period. If the maxGapDuration is not set to zero, periods smaller than the maxGapDuration (default 24h) will be added to the duration. In case of missing values, missing values are put in the output.
fews:durationPerEvent
Calculates the length of each event found in the input time series.
fews:maximum
Calculates the maximum input value for events that started in the output period. In case of missing values, missing values are put in the output.
fews:numberOfEvents

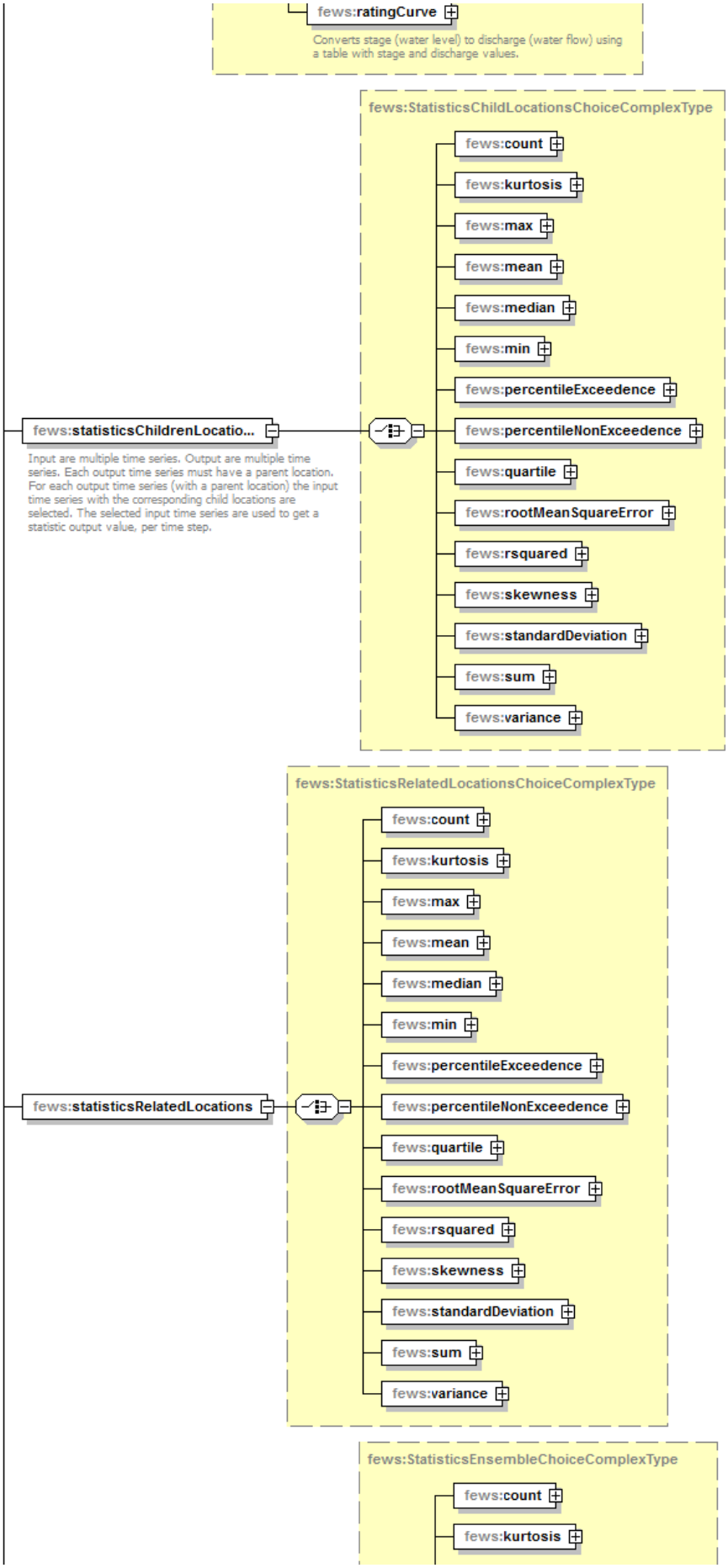

of missing values, missing values are put in the output.
fews:FilterTransformationChoiceComplexType
fews:filter
fews:lowPass
Low pass filter for discrete functions (e.g. time series). This transformation calculates the following difference equation. y(t) = b0*x(t) + b1*x(t-1) + ... + bM*x(t-M) + a1*y(t-1) + ... + aN*y(t-N). Here x is the input, y is the output, t denotes time, b0 to bM are the feedforward coefficients and a1 to aN are the feedback coefficients.
fews:gaussian
Smoothing transformation based on a gaussian function.
fews:GenerationChoiceComplexType
fews:generation
fews:constant
Does an even distribution of the totalAmount over the forecastPeriod period starting after the delay. The delay is a period between T0 and the start of the forecastPeriod for which 0 units defined in the output variable is generated. Example: timeLength = 1 hour, totalAmount = 10, delay = 2 hours. After running this transformation, the result will be [0, 5, 5]. Set relativeViewPeriod to [start = 0, end = delay + forecastPeriod]
fews:recessionMean
Will repeat the mean value defined in the history period with recession each repeat. First, the mean for the values listed in the history period is calculated. Then starting at T0 for the duration in 'length', the following is repeated: the mean value is written to the output variable for the same value of timesteps present in the historie period, after which the mean is multiplied by the resessionFactor. Example: length = 5 hour, historie = 2 hour, factor = 0.5, input = [3, 5, 3, (t0)4]. After running this transformation it would result in [(t0)4, 4, 2, 2, 1]. Set relativeViewPeriod to [start = -historie, end = length]
fews:recessionPattern
Will repeat the pattern in the history period with recession each repeat. First, the values listed in the history period are stored as historic amounts. Then starting at T0 for the duration in 'length', the following is repeated: all historic values are copied to the output variable for the amount timesteps present in the historie period, after which the historic values are multiplied by the resessionFactor. Example: length = 5 hour, historie = 2 hour, factor = 0.5, input = [3,5,3,(t0)4]. After running this transformation it would result in [(t0)5 ,3 ,2.5 ,1.5 ,1.25]. Set relativeViewPeriod to [start = -historie, end = length]
fews:GenerationEnsembleChoiceComplexType
fews:generationEnsemble
fews:copy
This transformation can be used to copy the ensembles
fews:selectHistoricMembers
Since 2013.01. FEWS-7893. This transformation uses scalar time series to make a sub-selection of ensemble members
fews:combineMembers
Since 2013.01. FEWS-8129. This transformation merges members of two or more ensembles into one bigger ensemble
fews:combineForecastMembers
Since 2013.01. FEWS-8505. This transformation merges members of two or more ensemble forecasts into one bigger ensemble
fews:GradientChoiceComplexType
fews:gradient
fews:firstOrder
Calculation of the gradient of a timeserie: (y1-y0)/(t1-t0)
fews:InterpolationSerialTransformationChoiceComplexType
fews:block
Fills gaps in a time series. Each gap is filled with data values that are equal to the last data value before the gap.
fews:default
Fills gaps in a time series with a default value

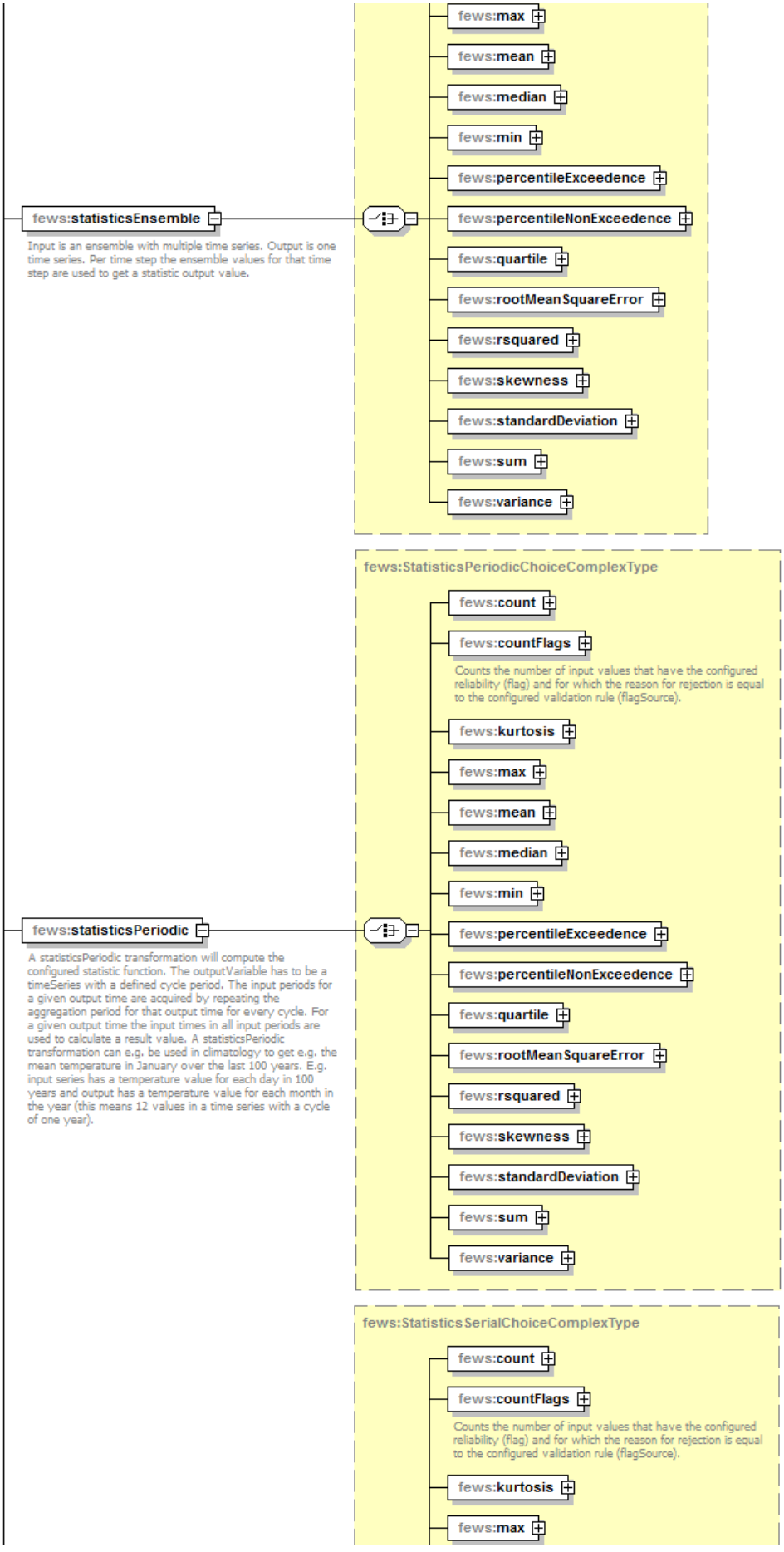

fews:interpolationSerial
fews:directionLinear
Fills gaps in a time series that contains direction data values (e.g. wind direction), using linear interpolation. The direction values are interpolated over the smallest angle, e.g. halfway between directions 0 degrees and 350 degrees the interpolated value would be 355 degrees. For a gap between two directions that are completely opposite (e.g. 90 and 270 degrees) the interpolated values will be equal to the last known direction.
fews:extrapolateBase
Extrapolates by adding future values that start at the last data value and linearly approach a given base value.
fews:extrapolateConstant
Extrapolates by filling the gap between the start of the input time series and the first data value or by filling the gap between the last data value and the end of the input time series or by filling both of these gaps.
fews:extrapolateExponential
Extrapolates by adding future values that start at the last data value and exponentially approach a given base value.
fews:extrapolatePatternWithS...
Extrapolates by repeating the values in the matching period. The matching period ends at the last non-missing value in the view period. The values in the matching period are referred to as the pattern. This pattern is repeated for the length of the view period. Smoothing is applied on each repetition of the pattern by adding (part of) the difference between the first and last value of the pattern. This linearly declines to zero for the length of the smoothingPeriod. Example: matchingPeriod = 8, Pattern = [4,5,5,5,8,6,7,8], smoothingPeriod = 4 would result in [8,8,7,6,8,6,7,8]
fews:linear
Fills gaps in a time series with linearly interpolated data values.
fews:interpolationSpatial
fews:InterpolationSpatialTransformationChoiceComplexType
fews:average
calculates the average for each area in the output geometry
fews:bilinear
calculates the output value by using bilinear interpolation from a rectangular grid to any other grid, profile or scalars. Can be used to rescale or reproject a grid
fews:closestDistance
interpolates by searching the closest location
fews:containingPolygon
For each input polygon this transformation copies the value of the input polygon to all output points that are inside the input polygon. If an output point is not contained in any input polygon, then the output is a missing value.
fews:inputAverageTimesOutp...
Calculates the weighted average of input variables and multiplies the output with the output area
fews:inverseDistance
With this interpolation option a new value will be computed out of existing values of neighbouring points using the power of inverse distance procedure.
fews:kriging
With this interpolation option a new value will be computed out of existing values of neighbouring points using the normal Kriging procedure.
fews:max
calculates the maximum for each area in the output geometry
fews:min
calculates the minimum for each area in the output geometry
fews:sum
calculates the sum for each area in the output geometry
fews:thiessenPolygon
First creates a Thiessen net from the input locations, then

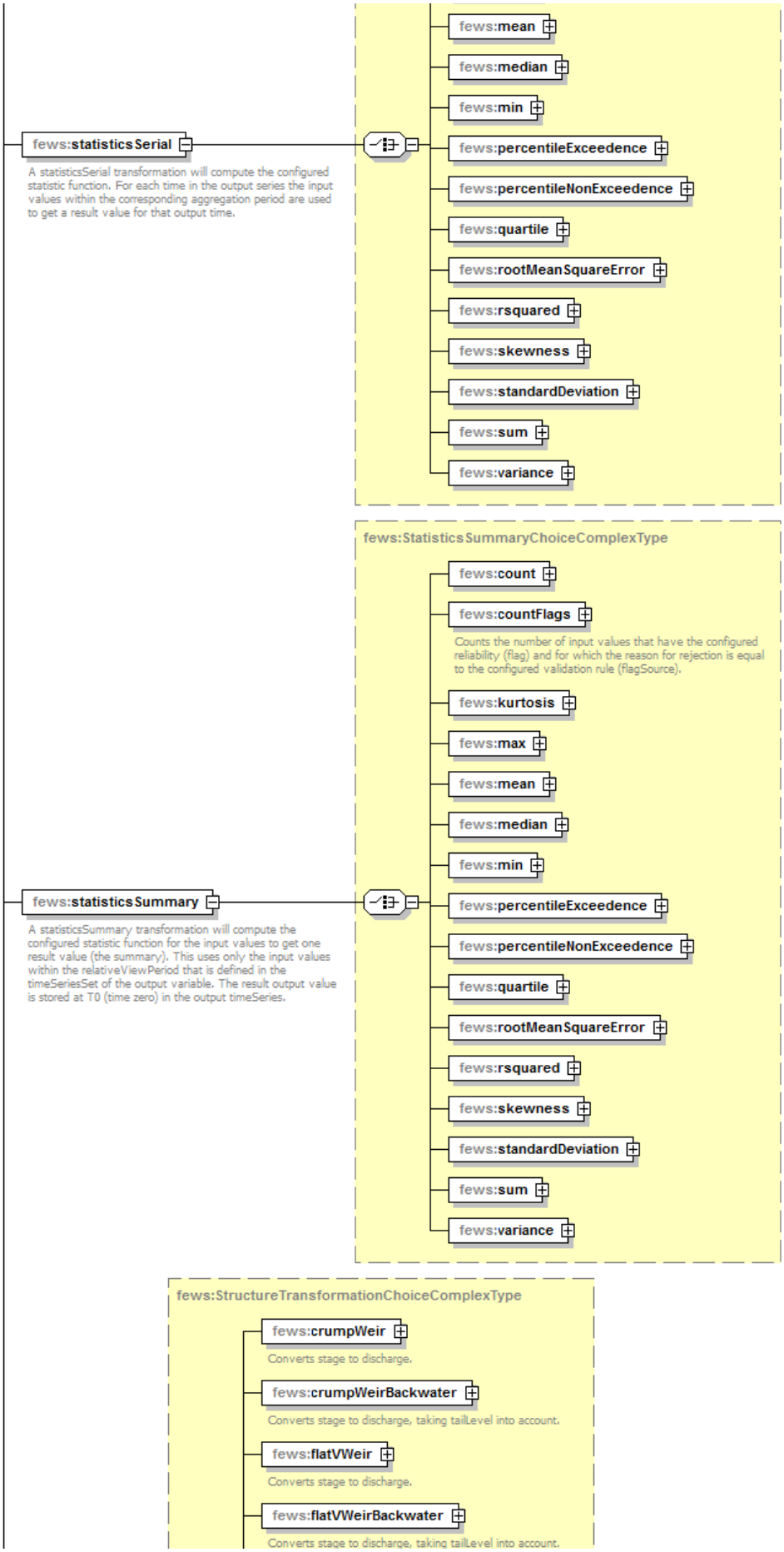

averages this over the region inside a polygon.
fews:triangulation
Renka and Cline triangulation.
fews:weighted
Calculates the weighted average of the input values. The weights are re-scaled so that the total weight becomes 1. If for a given time an input variable has a missing value, then for that time that input variable is ignored and the weights of the other input variables are re-scaled so that the total weight becomes 1.
fews:linearChainage
With this method you can copy (a part of) a longitudinal profile to another profile also when the destination profile contains more or less points.
For points that are not available in the input profile a linear chainage interpolation is used between the two closest point in the input profile that are available.
fews:snapTrackToLocations
Converts a track to multiple scalar time series. A maximum snap distance can be configured. The closest track point within a time step of the output time series is used.
fews:LookupTransformationChoiceComplexType
fews:lookup
fews:simple
Function that is defined by [input, output] value pairs. For input values for which there is no output value defined, the output value is determined using interpolation or extrapolation, depending on the specified options.
fews:twoDimensionalLookup
Function that is defined by [input1, input2, output] value pairs. For input values for which there is no output value defined, the output value is determined using interpolation or extrapolation, depending on the specified options.
fews:twoDimensionalTable
This function has a coefficientSet with a two-dimensional table. For each row and each column the table contains an output value. The values of the rowIndexLookupVariable and the columnIndexLookupVariable are used to lookup a row and a column in the table to get the output value from.
fews:MergeTransformationChoiceComplexType
fews:merge
fews:mean
Merges mean time series with different time steps to a merged mean time series
fews:relations
Since 2013.01. FEWS-8619: Merges input data that is defined using one or more configured relations. Each relation contains a period for which it is valid and a free format expression that is used to determine the output value for that relation. If for a given time step the first relation produces a missing value, then the next relation is used. If that also produces a missing value, then the next relation is used, and so on, until a non-missing output value is found. The output time series contains for each time step the configured output comment (optional) of the relation that was used to determine the value for that time step. If all relations produce a missing value for a given time step, then the output for that time step is a missing value.
fews:simple
Merges input time series.
fews:synoptic
Merges synoptic time series (data values that are accumulations over overlapping periods of time) to a time series with hourly data values.
fews:toggle
Merges input time series. If input time series contains a gap larger than maxGapLength, then it will be replaced (toggled instead of merged) with the next input time series in the hierarchy.
fews:weighted
Merges two input time series with weights that are a function of time.

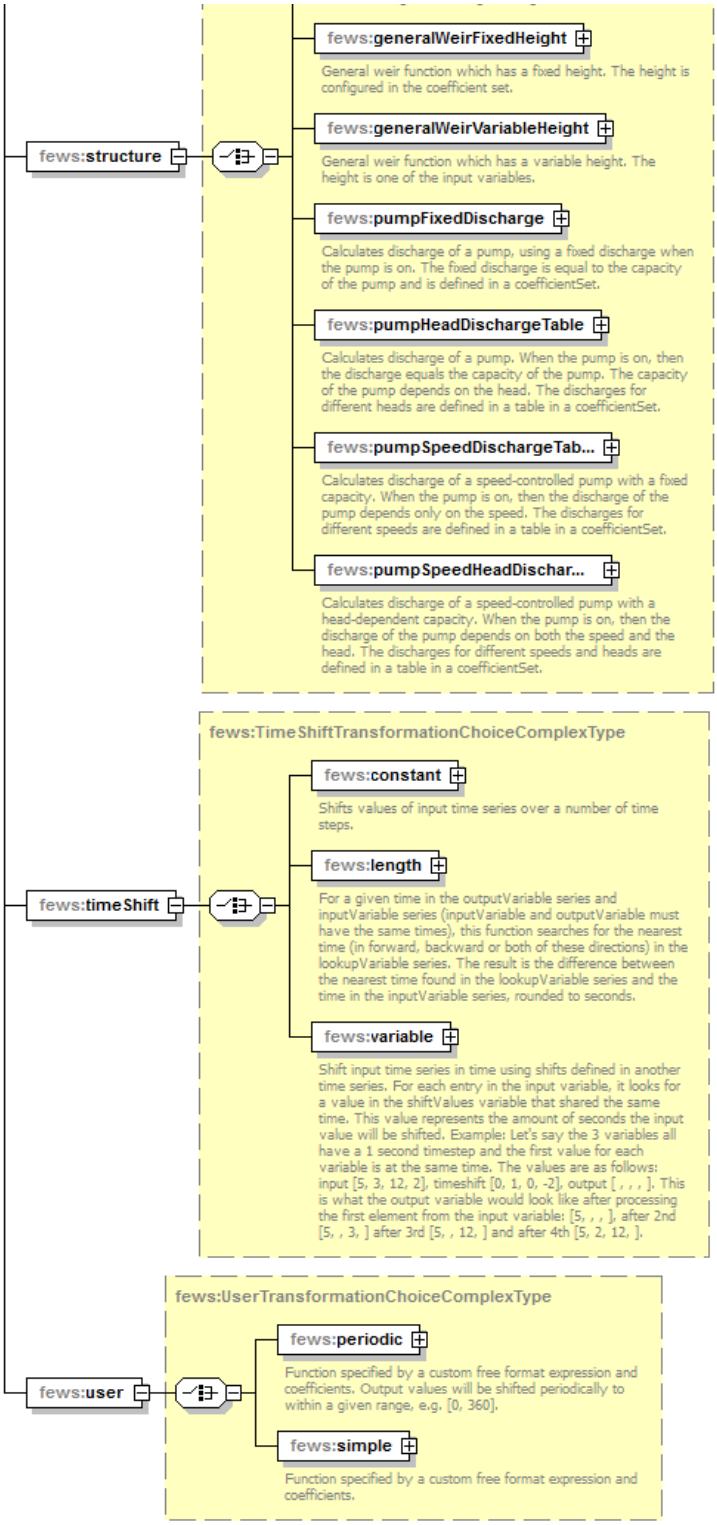

FunctionChoiceGroup
Groups all types of transformation functions.
fews:selectLocation
Selects an input time series based on the value of an attribute
fews:selectDataSource
Selects a data source based on the value of an attribute
fews:moisture
fews:MoistureTransformationChoiceComplexType
fews:opticalDepth
Calculates optical depth. For grid time series only.
fews:soilMoisture
Calculates soil moisture. For grid time series only.
fews:precipitation
fews:PrecipitationTransformationChoiceComplexType
fews:tippingBucket
Converts a non-equidistant time series with tipping bucket rainfall data to an equidistant time series with accumulated rainfall data.
fews:profile
fews:ProfileTransformationChoiceComplexType
fews:timeSeries
Converts a profile to a time series.
fews:regression
fews:RegressieTransformationChoiceComplexType
fews:principalComponentAnal...
This transformations derives an equation to estimate the historical model results with historical observations.
The rmse of the estimated model results is also calculated. By using the equation the current model results are estimated by using the current observations.
fews:review
fews:ReviewTransformationChoiceComplexType
fews:stageReview
Divides the stage into equal ranges and calculates the difference between the forecast and the measured data for each range
fews:tidalBalanceReview
Calculates the balance between observed and forecasted tides, creates estimates for future tides
fews:sample
fews:SampleTransformationChoiceComplexType
fews:daySample
samples the input timeseries to a dayly timeseries using the time of
T0 to determine the sampling moment. Output variable should be a non-equidistant simulated
timeseries to accommodate shifting sampling times.
fews:dayMonthSample
samples the input timeseries to a day month output timeseries using the day-month of
T0
to determine the sampling moment. Output variable should be a non-equidistant simulated
timeseries to accommodate shifting sampling times.
fews:julianDaySample
Since 2013.01. FEWS-8576. samples the input time series to the output time series using the Julian Days
fews:equidistant
Transform equidistant to another equidistant or to nonEquidistant time series.
fews:historical


fews:nonEquidistant
Transform nonEquidistant to equidistant time series.
fews:SelectionTransformationChoiceComplexType
fews:independentLows
Finds the low values of the input time series and writes this to the output time series.
fews:lows
Finds the low values of the input time series and writes this to the output time series.
fews:maximum
Finds the maximum value of the input time series and writes this to the output time series. The maximum value will be written at the same time as in the input time series. Ignores unreliable values. If multiple values are equal to the maximum, then the latest value is used.
fews:selection
fews:minimum
Finds the minimum value of the input time series and writes this to the output time series. The minimum value will be written at the same time as in the input time series. Ignores unreliable values. If multiple values are equal to the minimum, then the latest value is used.
fews:independentPeaks
Finds the peak values of the input time series and writes this to the output time series.
fews:peaks
Finds the peak values of the input time series and writes this to the output time series.
fews:valuesAfterChangeWithTi...
Since 2013.01. FEWS-8585. Copies values from the inputVariable to the outputVariable under certain conditions. The specified conditionVariable is searched for changes during the run period of the transfomation. When the conditionVariable value changes, then a time window is created. This time window starts at the time of the change. The length of the time window (in seconds) is determined by evaluating the specified timeWindowLengthExpression at the time of the change. For each change in the conditionVariable a new time window is created (time windows can overlap). If the specified selectionType is select_inside_time_window, then all inputVariable values that are within at least one time window are copied to the outputVariable and all other inputVariable values are ignored. If selectionType is select_outside_time_window then all inputVariable values that are within at least one time window are ignored and all other inputVariable values are copied to the outputVariable. This can be used e.g. to filter out measurement values during periods where the measured system is not in a (quasi) steady state due to changes happening in the system. Produces missing output values at times where there is not enough reliable or doubtful data available in the condition and/or expression variables to determine if there is a change or to determine the length of a time window. Any such missing output values are truncated from the start and end of the run period of the transformation, to avoid overwriting data from previous runs with overlapping run periods.
fews:StageDischargeTransformationChoiceComplexType
fews:mergedRatingCurves
Merges two rating curves using a time dependent weight variable and uses the resulting rating curve to convert stage input values to discharge output values. For each timeStep in the output time series, first the specified two rating curves are merged using the value of the weight input time series at that timeStep. If weight is 1, then uses the first rating curve. If weight is 0, then uses the second rating curve. If weight is between 0 and 1, then interpolates linearly between the first and the second rating curve to get the merged rating curve. Then the merged rating curve is used to convert the stage input value for that timeStep to a discharge output value. This can only use rating curves that are stored as time series in the dataStore. This uses the equation Q_output = weight*Q_ratingCurve1(H_input) + (1 - weight)*Q_ratingCurve2(H_input).
fews:stageDischarge
fews:power
Converts stage (H) to discharge (Q) for an open cross section. Uses equation Q=a*(H+b)^c.
fews:table
Converts stage (water level) to discharge (water flow) using a table with stage and discharge values.


fews:ratingCurve
Converts stage (water level) to discharge (water flow) using a table with stage and discharge values.
fews:StatisticsChildLocationsChoiceComplexType
fews:count
fews:kurtosis
fews:max
fews:mean
fews:median
fews:min
fews:percentileExceedence
fews:percentileNonExceedence
fews:quartile
fews:rootMeanSquareError
fews:rsquared
fews:skewness
fews:standardDeviation
fews:sum
fews:variance
fews:statisticsChildrenLocatio...
Input are multiple time series. Output are multiple time series. Each output time series must have a parent location. For each output time series (with a parent location) the input time series with the corresponding child locations are selected. The selected input time series are used to get a statistic output value, per time step.
fews:StatisticsRelatedLocationsChoiceComplexType
fews:count
fews:kurtosis
fews:max
fews:mean
fews:median
fews:min
fews:percentileExceedence
fews:percentileNonExceedence
fews:quartile
fews:rootMeanSquareError
fews:rsquared
fews:skewness
fews:standardDeviation
fews:sum
fews:variance
fews:statisticsRelatedLocations
fews:StatisticsEnsembleChoiceComplexType
fews:count
fews:kurtosis


fews:statisticsEnsemble
Input is an ensemble with multiple time series. Output is one time series. Per time step the ensemble values for that time step are used to get a statistic output value.
fews:max
fews:mean
fews:median
fews:min
fews:percentileExceedence
fews:percentileNonExceedence
fews:quartile
fews:rootMeanSquareError
fews:rsquared
fews:skewness
fews:standardDeviation
fews:sum
fews:variance
fews:statisticsPeriodic
A statisticsPeriodic transformation will compute the configured statistic function. The outputVariable has to be a timeSeries with a defined cycle period. The input periods for a given output time are acquired by repeating the aggregation period for that output time for every cycle. For a given output time the input times in all input periods are used to calculate a result value. A statisticsPeriodic transformation can e.g. be used in climatology to get e.g. the mean temperature in January over the last 100 years. E.g. input series has a temperature value for each day in 100 years and output has a temperature value for each month in the year (this means 12 values in a time series with a cycle of one year).
fews:StatisticsPeriodicChoiceComplexType
fews:count
fews:countFlags
Counts the number of input values that have the configured reliability (flag) and for which the reason for rejection is equal to the configured validation rule (flagSource).
fews:kurtosis
fews:max
fews:mean
fews:median
fews:min
fews:percentileExceedence
fews:percentileNonExceedence
fews:quartile
fews:rootMeanSquareError
fews:rsquared
fews:skewness
fews:standardDeviation
fews:sum
fews:variance
fews:StatisticsSerialChoiceComplexType
fews:count
fews:countFlags
Counts the number of input values that have the configured reliability (flag) and for which the reason for rejection is equal to the configured validation rule (flagSource).
fews:kurtosis
fews:max


fews:statisticsSerial
A statisticsSerial transformation will compute the configured statistic function. For each time in the output series the input values within the corresponding aggregation period are used to get a result value for that output time.
fews:mean
fews:median
fews:min
fews:percentileExceedence
fews:percentileNonExceedence
fews:quartile
fews:rootMeanSquareError
fews:rsquared
fews:skewness
fews:standardDeviation
fews:sum
fews:variance
fews:statisticsSummary
A statisticsSummary transformation will compute the configured statistic function for the input values to get one result value (the summary). This uses only the input values within the relativeViewPeriod that is defined in the timeSeriesSet of the output variable. The result output value is stored at T0 (time zero) in the output timeSeries.
fews:StatisticsSummaryChoiceComplexType
fews:count
fews:countFlags
Counts the number of input values that have the configured reliability (flag) and for which the reason for rejection is equal to the configured validation rule (flagSource).
fews:kurtosis
fews:max
fews:mean
fews:median
fews:min
fews:percentileExceedence
fews:percentileNonExceedence
fews:quartile
fews:rootMeanSquareError
fews:rsquared
fews:skewness
fews:standardDeviation
fews:sum
fews:variance
fews:StructureTransformationChoiceComplexType
fews:crumpWeir
Converts stage to discharge.
fews:crumpWeirBackwater
Converts stage to discharge, taking tailLevel into account.
fews:flatVWeir
Converts stage to discharge.
fews:flatVWeirBackwater
Converts stage to discharge, taking tailLevel into account.


fews:structure
fews:generalWeirFixedHeight
General weir function which has a fixed height. The height is configured in the coefficient set.
fews:generalWeirVariableHeight
General weir function which has a variable height. The height is one of the input variables.
fews:pumpFixedDischarge
Calculates discharge of a pump, using a fixed discharge when the pump is on. The fixed discharge is equal to the capacity of the pump and is defined in a coefficientSet.
fews:pumpHeadDischargeTable
Calculates discharge of a pump. When the pump is on, then the discharge equals the capacity of the pump. The capacity of the pump depends on the head. The discharges for different heads are defined in a table in a coefficientSet.
fews:pumpSpeedDischargeTab...
Calculates discharge of a speed-controlled pump with a fixed capacity. When the pump is on, then the discharge of the pump depends only on the speed. The discharges for different speeds are defined in a table in a coefficientSet.
fews:pumpSpeedHeadDischar...
Calculates discharge of a speed-controlled pump with a head-dependent capacity. When the pump is on, then the discharge of the pump depends on both the speed and the head. The discharges for different speeds and heads are defined in a table in a coefficientSet.
fews:timeShift
fews:TimeShiftTransformationChoiceComplexType
fews:constant
Shifts values of input time series over a number of time steps.
fews:length
For a given time in the outputVariable series and inputVariable series (inputVariable and outputVariable must have the same times), this function searches for the nearest time (in forward, backward or both of these directions) in the lookupVariable series. The result is the difference between the nearest time found in the lookupVariable series and the time in the inputVariable series, rounded to seconds.
fews:variable
Shift input time series in time using shifts defined in another time series. For each entry in the input variable, it looks for a value in the shiftValues variable that shared the same time. This value represents the amount of seconds the input value will be shifted. Example: Let's say the 3 variables all have a 1 second timestep and the first value for each variable is at the same time. The values are as follows: input [5, 3, 12, 2], timeshift [0, 1, 0, -2], output [, , ,]. This is what the output variable would look like after processing the first element from the input variable: [5, , ,], after 2nd [5, , 3,] after 3rd [5, , 12,] and after 4th [5, 2, 12,].
fews:user
fews:UserTransformationChoiceComplexType
fews:periodic
Function specified by a custom free format expression and coefficients. Output values will be shifted periodically to within a given range, e.g. [0, 360].
fews:simple
Function specified by a custom free format expression and coefficients.