

20 Transformation Module - Improved schema

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

Contents

- [Contents](#)
- [Transformation Module Configuration \(New Version\)](#)
 - [Configuration](#)
 - [Validation rules](#)
 - [Manual Edits](#)
 - [Description](#)
 - [Copying comments, flags and flag sources from input to output](#)
 - [Preventing previously calculated values to be overwritten with missings](#)
 - [Run transformations for a set of selected locations](#)
 - [Which locations should the user select?](#)
 - [Steps to follow when implementing selection specific calculations](#)
 - [Implement selection specific calculations for IFD](#)
 - [Trim Output](#)
 - [Forecast Loop](#)
 - [List of all available transformations](#)
- [Accumulation Transformations](#)
- [Adjust Transformations](#)
- [Aggregation transformations](#)
- [Altitude](#)
- [Conditional](#)
- [DayMonth Sample](#)
- [Deaccumulation](#)
- [DisaggregationTransformations](#)
- [DischargeStage Transformations](#)
- [Events Transformations](#)
- [Filter Transformations](#)
- [GenerationEnsemble Transformation](#)
- [Generation Transformations](#)
- [Gradient Transformations](#)
- [Interpolation Serial Transformations](#)
- [Interpolation Spatial Transformations](#)
- [Lookup transformations](#)
- [Merge Transformations](#)
- [Multi location attribute transformation](#)
- [PCA and Regression Transformation](#)
- [Polygon related locations](#)
- [Precipitation](#)
- [Profile](#)
- [Review transformations](#)
- [Sample transformations](#)
- [Selection Transformations](#)
- [StageDischarge transformations](#)
- [Statistics Children Locations](#)
- [Statistics Ensemble Transformations](#)
- [Statistics Periodic Transformations](#)
- [Statistics Related Locations](#)
- [Statistics Same Attribute Value](#)
- [Statistics Serial Transformations](#)
- [Statistics Summary Transformations](#)
- [StatisticsValueProperties](#)
- [Statistics Vertical Layer transformations](#)
- [Structure Transformations](#)
- [TimeShift](#)
- [User Transformations](#)

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/unreliable 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 be explained. Let's say that we would like to 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 then 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 unreliable 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 a 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 applied.

```
<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 as mouse over 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>

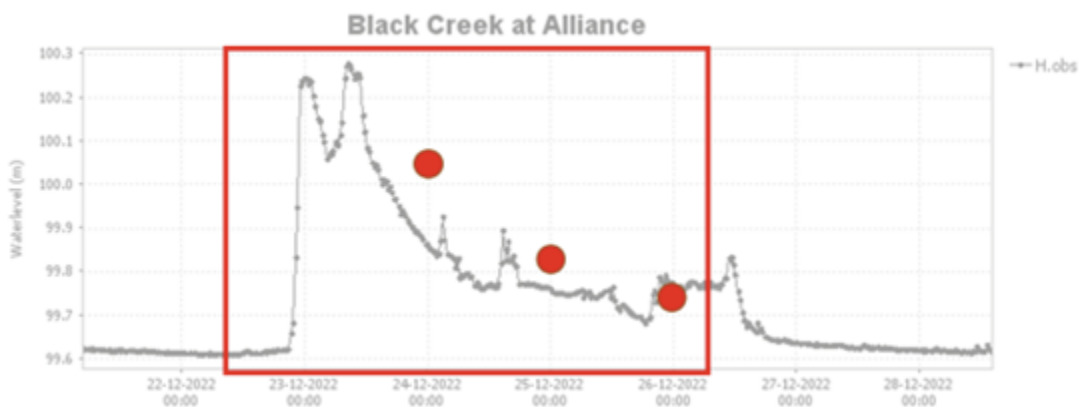
Preventing previously calculated values to be overwritten with missings

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:

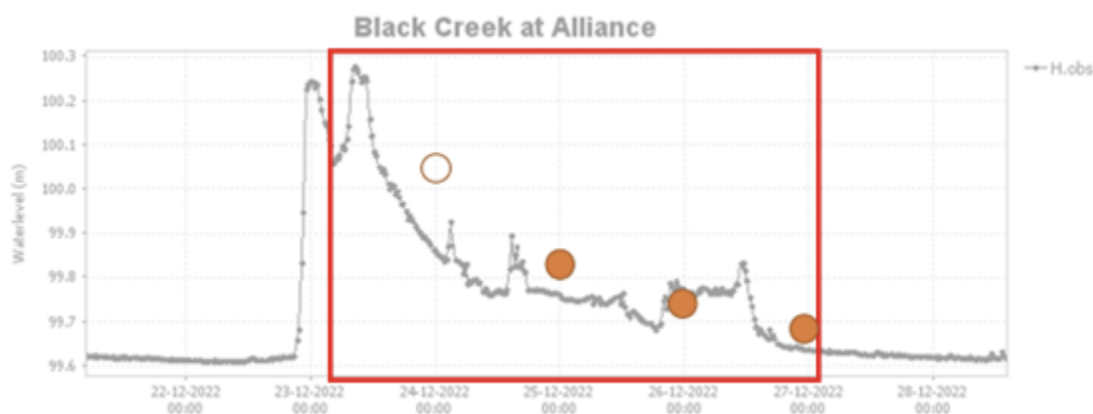
Existing value overwritten by missing

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 *timeserieType* "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 edited or when the configuration is changed.

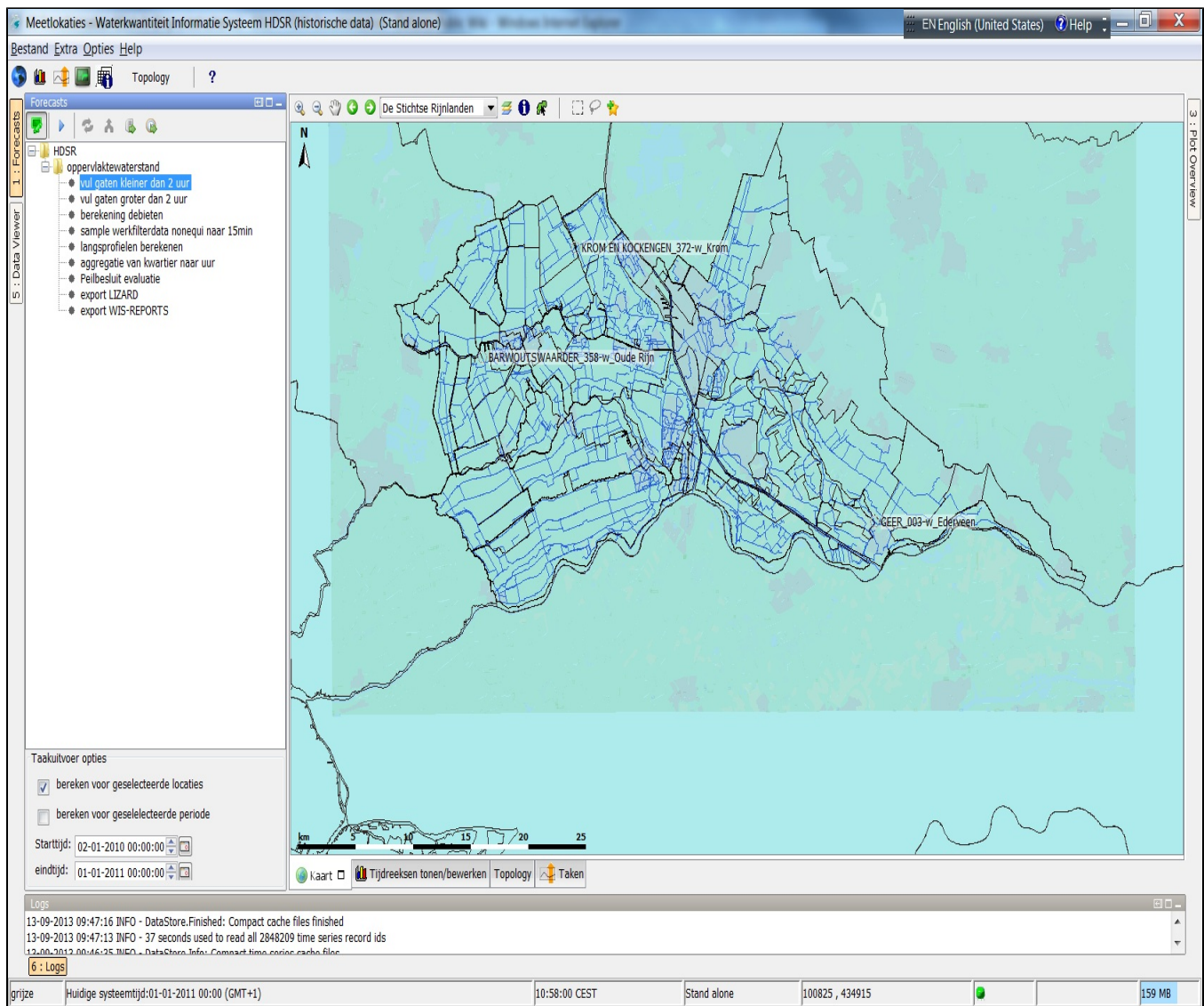
In this case the workflow can skip the calculations for the unchanged locations. The main benefit of this approach is that it saves a lot of processing time.

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>
```

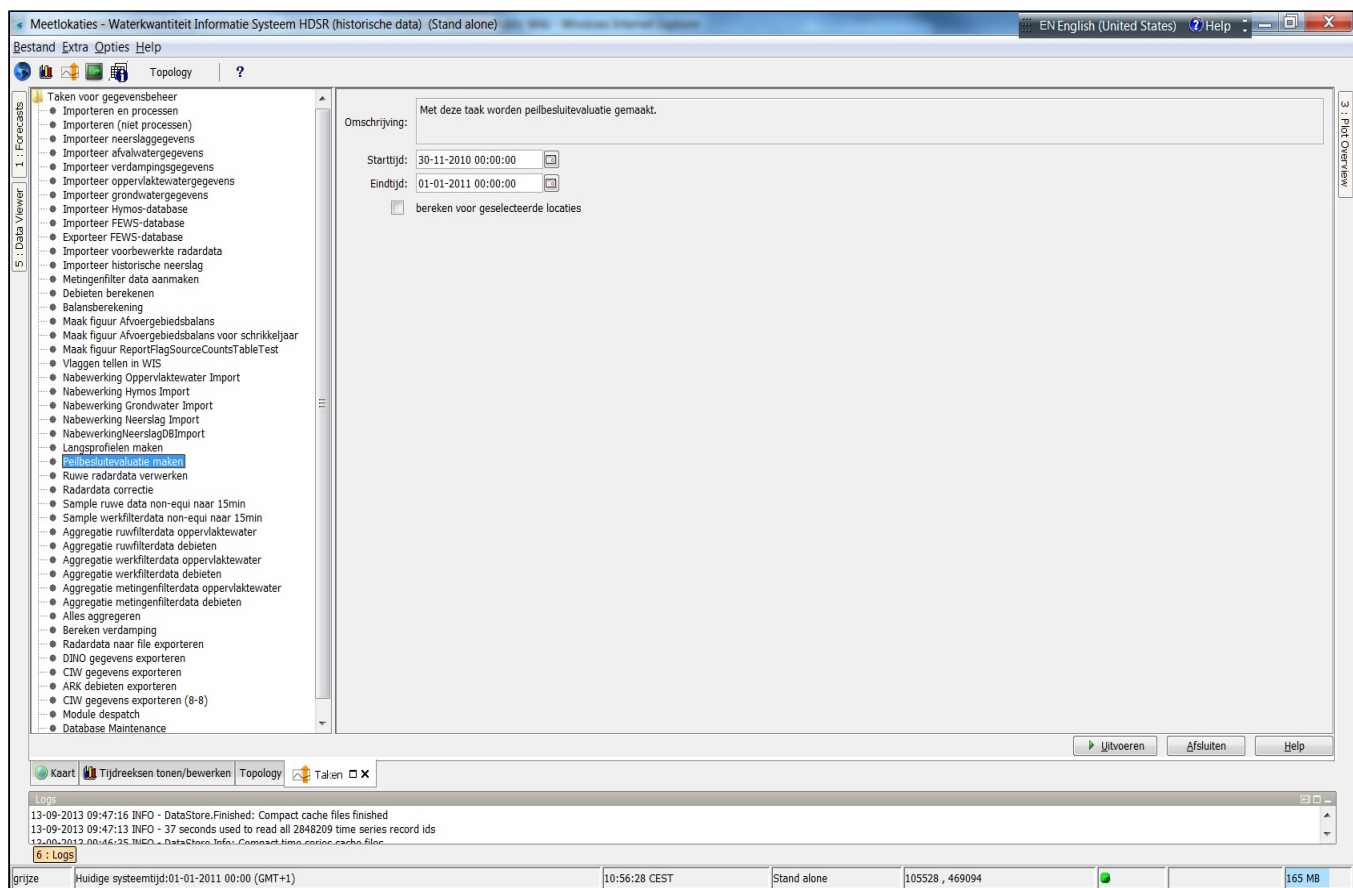
When a node in the IFD is selected with a workflow which has the `allowSelection` option set true, the GUI will look like this:



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 additional 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 selection 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.

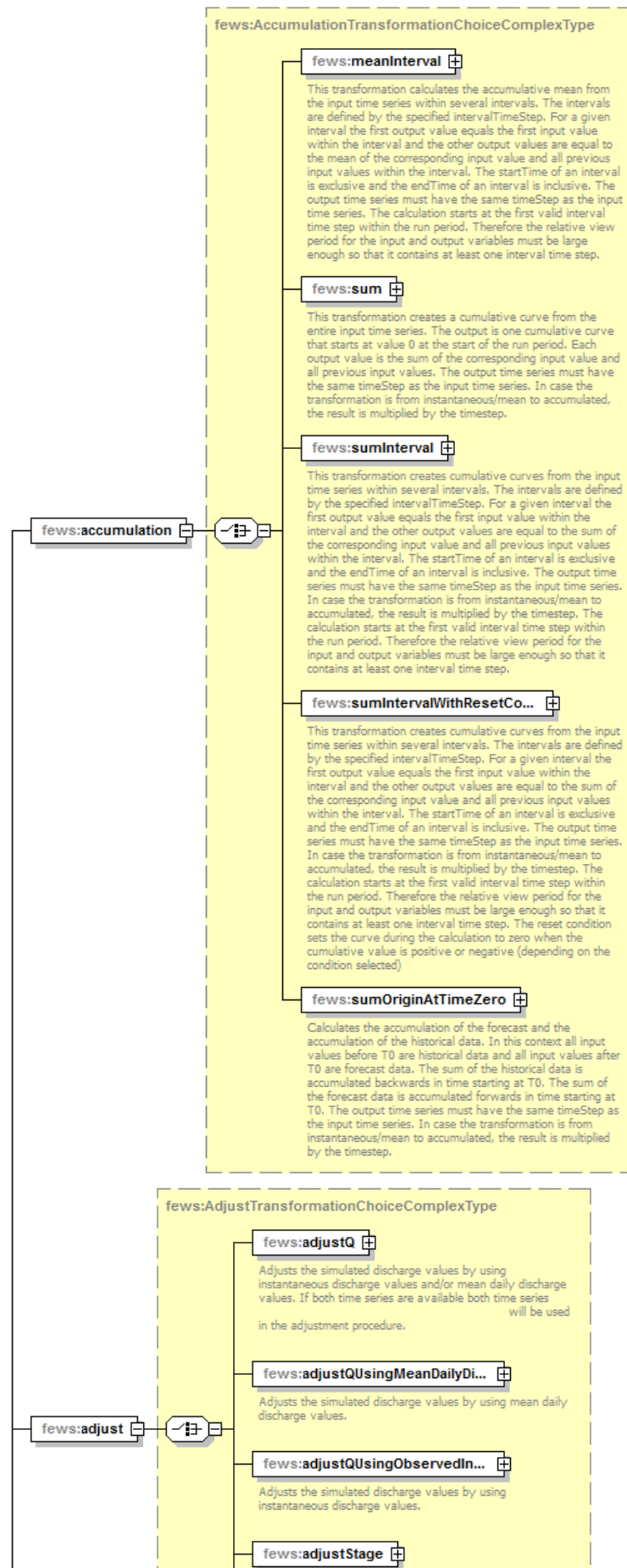
This will only work when the `<inputVariable>` and `<outputVariable>` are external forecasts. The output variable will get the same external forecast time as the input time series.

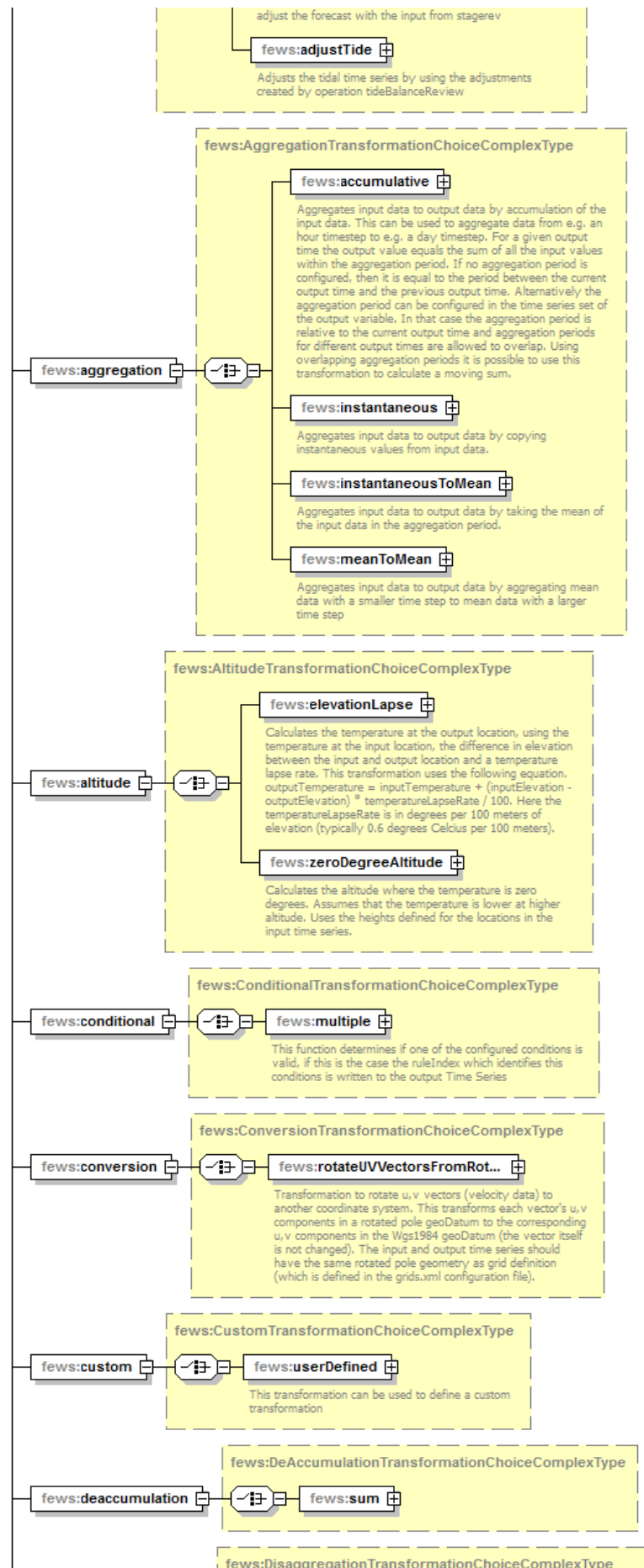
When this is configured in combination with a `locationSet` it will try to run the transformation for the maximum number of forecasts available for the locations. If for a location some forecasts are unavailable a warning will be logged and those transformation runs will be skipped for that forecast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

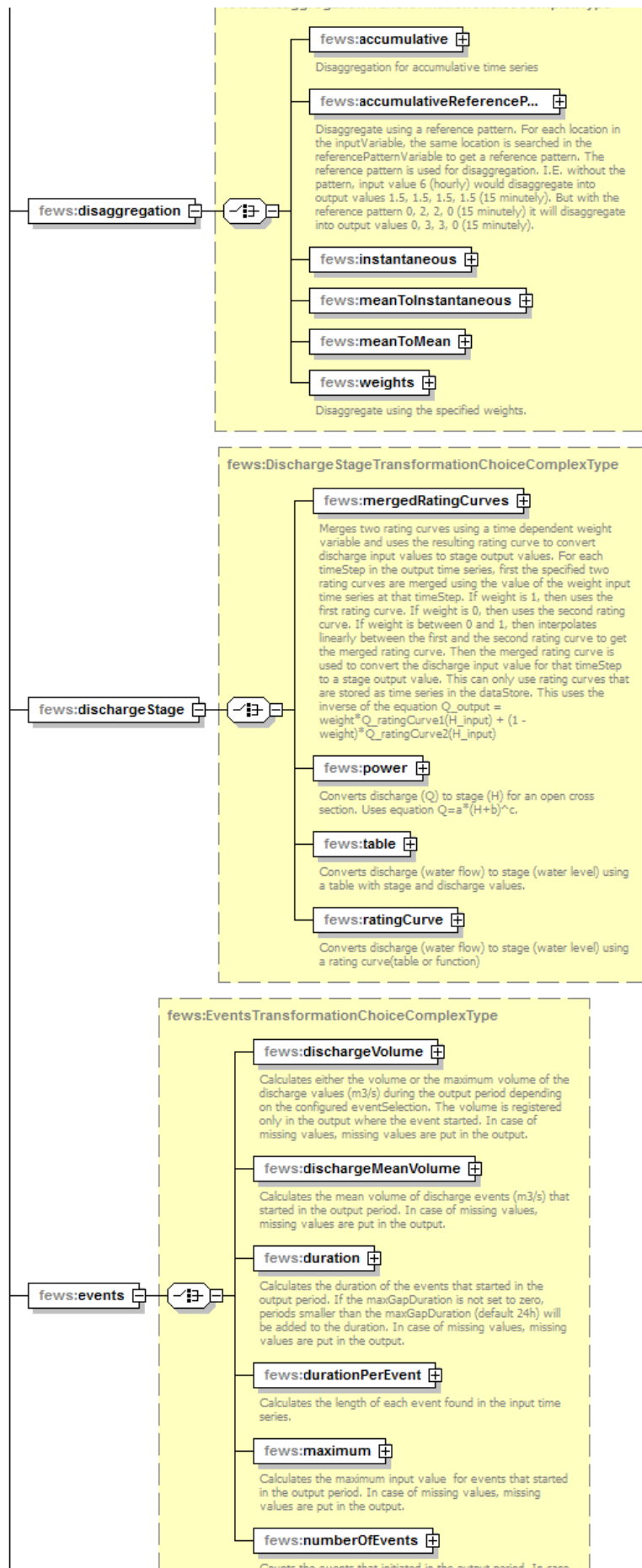
List of all available transformations

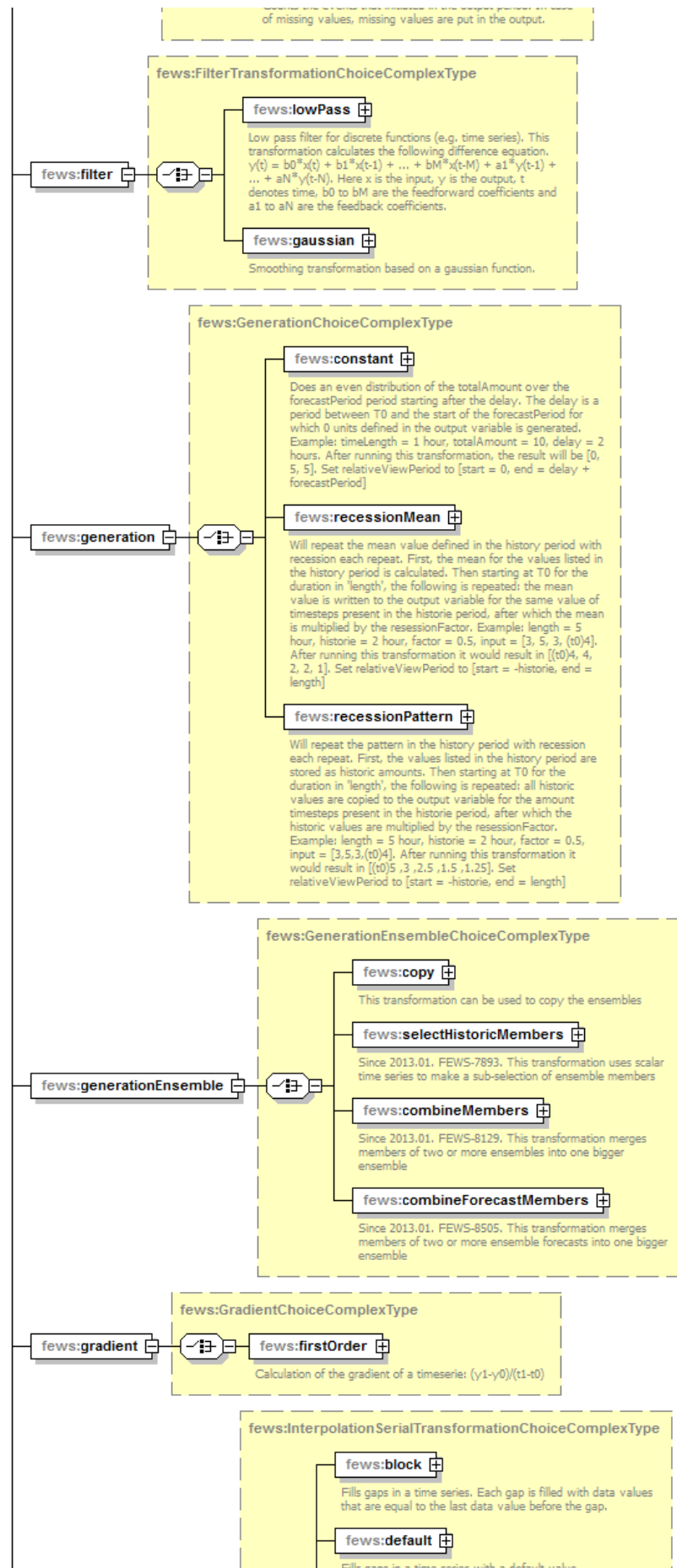
For the most recent development version see the xsd schema at <https://fewsdocs.deltares.nl/schemas/version1.0/transformationTypes.xsd>

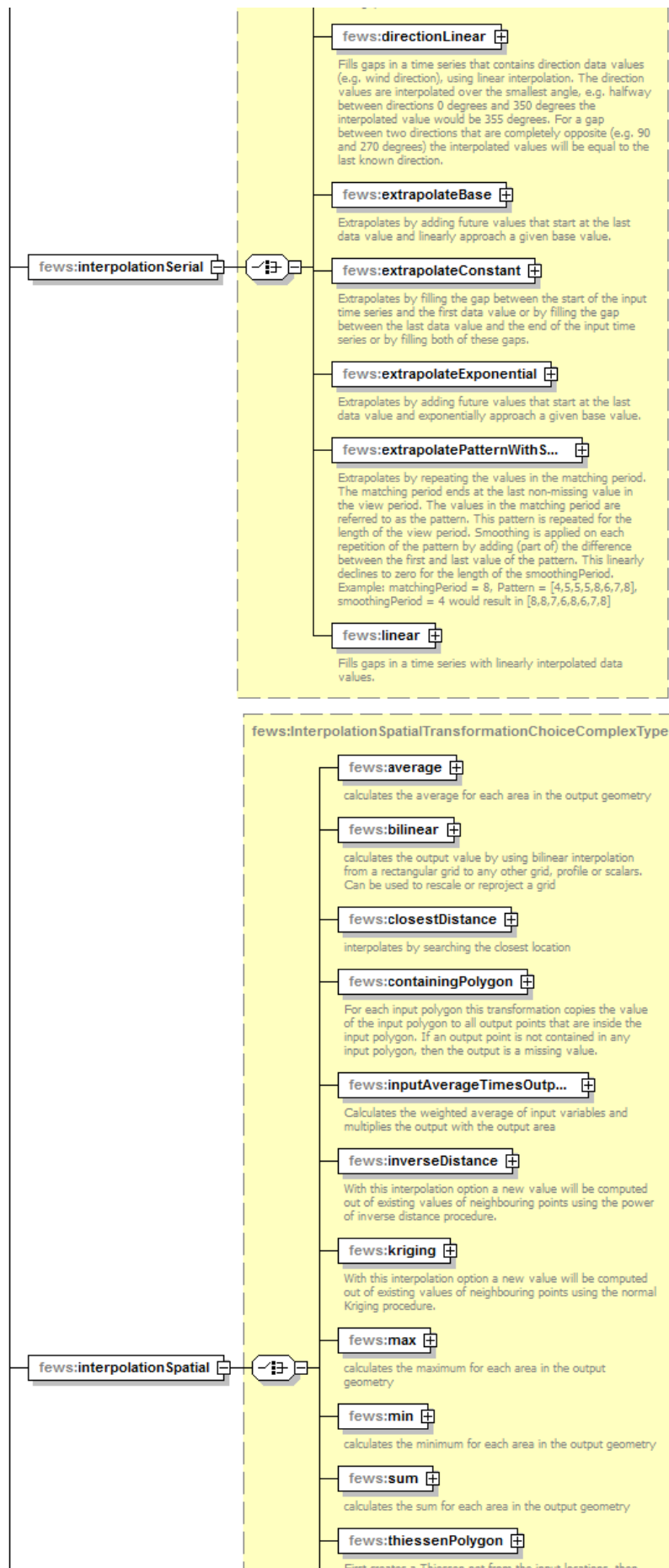
Available since stable build 2014.01:

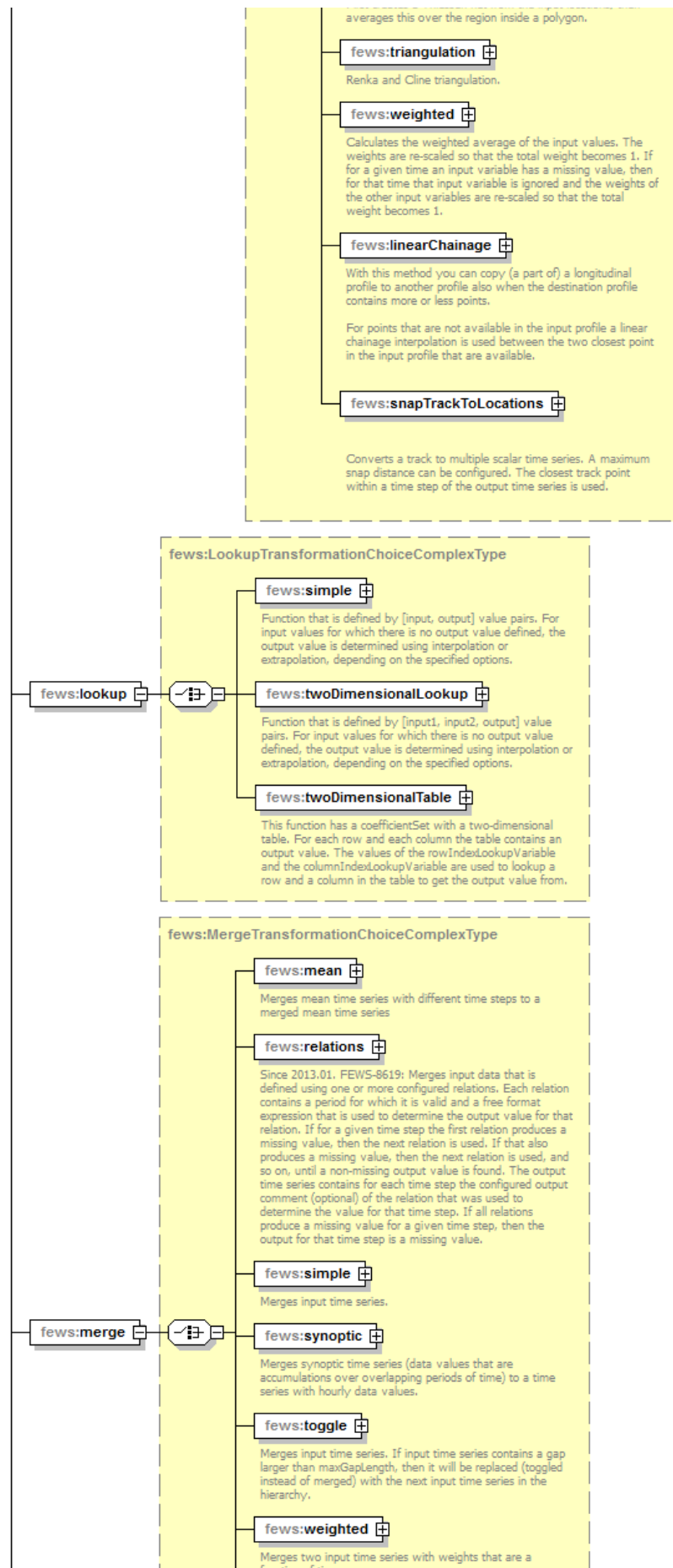




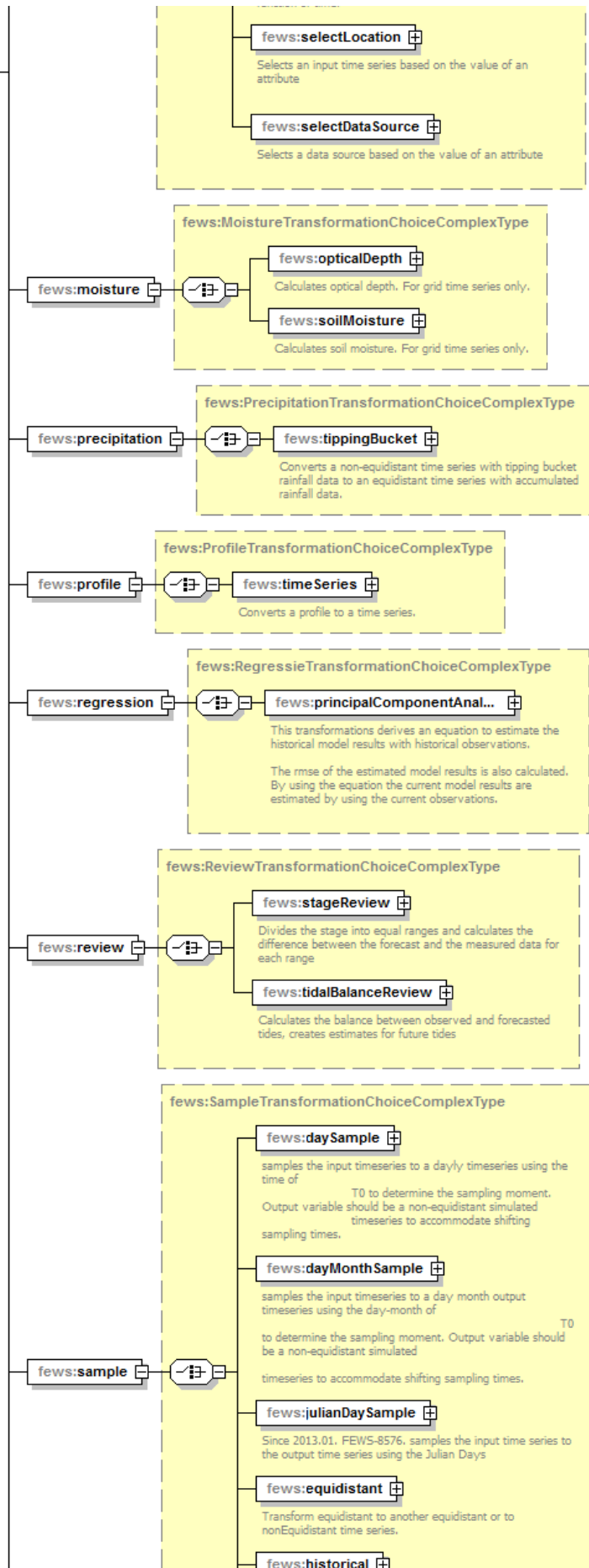


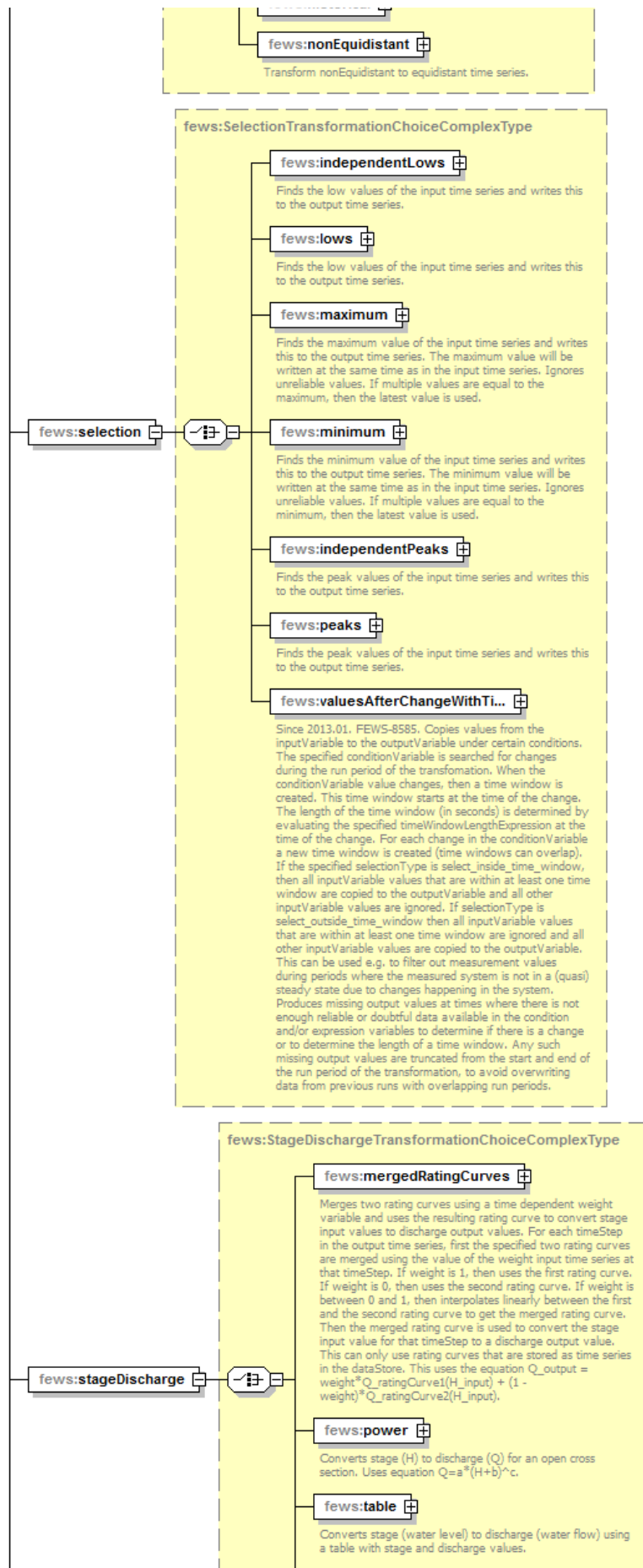






FunctionChoiceGroup
Groups all types of transformation functions.





fews:ratingCurve

Converts stage (water level) to discharge (water flow) using a table with stage and discharge values.

fews:statisticsChildrenLocations

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: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:statisticsRelatedLocations

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: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: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.

