UserSimple Transformation

UserSimple

Input

It is possible to define embedded variables in this transformation. In the expression both embedded variables and variables defined at the start of the transformations configuration file can be used. If an embedded variable and a variable defined at the start of the transformations configuration file have the same variableId, then the embedded variable will be used.

In FEWS 2014.01 a new feature was added to the UserSimple transformation making it possible to embed variables containing a LocationSet. This can be done by creating the element 'inputVariableDefinition' where:

- **templateVariableId** refers to the variable defined at the start of the transformation configuration file. This variable should have a time series set definition, which will be used as a template. The time series set will be filtered using the filter location.
- **locationId** filter location. The time series set of the template variable will be filtered using this location and the resulting time series set will be stored in configured variableId.
- **variableId** The variable used in the expression. It will store the time series set from the template variable, after filtering for the filter location. This variable does not need to be explicitly defined at the start of the transformation configuration file.

In the example below Location_A is part of the locationSet Test_Locations:

```
<variable>
        <variableId>X1</variableId>
        <timeSeriesSet>
                <moduleInstanceId>UserSimpleTemplateVariable</moduleInstanceId>
                <valueType>scalar</valueType>
                <parameterId>Q.meting</parameterId>
                <locationSetId>Test_Locations</locationSetId>
                <timeSeriesType>external historical</timeSeriesType>
                <timeStep unit="day"/>
                <relativeViewPeriod unit="day" start="0" end="364"/>
                <readWriteMode>editing visible to all future task runs</readWriteMode>
        </timeSeriesSet>
</variable>
<variable>
        <variableId>Y1</variableId>
        <timeSeriesSet>
                <moduleInstanceId>UserSimpleTemplateVariable</moduleInstanceId>
                <valueType>scalar</valueType>
                <parameterId>Q.afgeleid</parameterId>
                <locationId>Location_A</locationId>
                <timeSeriesType>external historical</timeSeriesType>
                <timeStep unit="day"/>
                <relativeViewPeriod unit="day" start="0" end="364"/>
                <readWriteMode>add originals</readWriteMode>
        </timeSeriesSet>
</variable>
<transformation id="user simple test">
        <user>
                <simple>
                        <inputVariableDefinition>
                                <variableId>meting</variableId>
                                <locationId>Location_A</locationId>
                                <templateVariableId>X1</templateVariableId>
                        </inputVariableDefinition>
                        <expression>meting*1.08</expression>
                        <outputVariable>
                                <variableId>Y1</variableId>
                        </outputVariable>
                </simple>
        </user>
</transformation>
```

Expression

For instance "X1 + X2 * 3" (without the quotes). In the expression input variables or coefficients can be referenced using their id, e.g. "X1 + a" where "X1" is the variableId of a variable defined elsewhere and "a" is the id of a coefficient defined in a coefficientSet. A variableId or coefficientId should not start with a numerical character and should not contain operators.

Operators

The following operators can be used in the expression: +, -, /, *, ^ as well as the logical operators for 'or' || and 'and' && .

Functions

Furthermore the following functions can be used in the expression: sin, cos, tan, asin, acos, atan, sinh, cosh, tanh, asinh, acosh, atanh, log, ln, exp, sqrt, abs, pow, max, min. Note that the trigonometric functions assume that the argument is in radians.

isMissing Function

Furthermore the function isMissing can be used to check if an input value is missing (NaN), e.g. for the expression "isMissing(X)" the result will be 1 when X is missing and the result will be 0 when X is not missing.

Furthermore "pi" in lowercase letters is recognised as the standard mathematical constant . This means that the user cannot use variables or coefficients with id "pi".

Since stable build 2013.02 it is also possible to use the calculated output value of the previous time step, by using "PREVIOUS_OUTPUT_VALUE" in uppercase letters in the expression (can be used multiple times). For each calculation time step "PREVIOUS_OUTPUT_VALUE" is replaced with the previous output value from the output time series. For the first calculation time step in the run period, the previous output value from before the start of the run period is read from the existing output time series in the database (e.g. from a previous run), if present. If for a given calculation time step the previous output value is not present or is unreliable, then "PREVIOUS_OUTPUT_VALUE" is replaced with NaN (missing value).

To write a missing value (NaN) in the expression, you should use "0/0". Also note that an expression of "1/0" results in positive infinity, which is later replaced with NaN by the transformation module framework. However, "1/0" cannot be used in all cases to get an NaN value, since e.g. the expression "1 / (1/0)" gives 0 as a result, as expected.

Furthermore it is possible to use "if statements" in the expression. This can e.g. be used to get one output value if X is greater than 3 and get another output value if X is equal to or less than 3. For instance in the expression

```
if(X &gt; 3, 10.5, -2) + 5*a
```

the if statement will be replaced with 10.5 or -2, depending on the value of the variable 'X'. In this case if X is greater than 3, then the if statement is replaced with 10.5. If X is equal to or less than 3, then the if statement is replaced with -2. The following symbols can be used (without the quotes) in an if statement:

`>`	greater than
`>=`	greater than or equals
`<`	less than
`<=`	less than or equals
`'=='`	equals
`'!='`	unequals

If statements can also be nested, e.g.

```
if(m &gt; b, 3, if(m &gt; a, 2, 1))
```

Coefficient set or coefficient set functions

Should contain the coefficients that are used in the free format expression. Define the ids and values of the coefficients here, then refer to the ids of these coefficients in the expression. Make sure that for all the coefficient ids in the free format expression the values are defined here.

When using coefficient set functions (available since build 30246), the value elements can contain tags between "@" signs (e.g. "@NUMBER@") that refer to location attributes that are defined in the locationSets configuration file. The tags are replaced by actual values. These values can be different for different locations and time periods. See 22 Locations and attributes defined in CSV files, Shape-DBF files or external tables for more information.

Output

1. **output**: result of the evaluated expression.
   If comments are provided in the input variables being used in the expression, then the comments are appended for each time step in the output.

Description

Function specified by a custom free format expression and coefficients. Any number of input variables and coefficients can be used in the free format expression. The expression may contain general mathematical operators. A function parser is used to evaluate the expression. For each time step in the output time series the expression is evaluated and the result is written to the output time series.

Configuration examples

In the example below 'X1' is a reference to a variable and 'a' and 'b' are references to a coefficient.

```
<variable>
        <variableId>X1</variableId>
        <timeSeriesSet>
                <moduleInstanceId>UserSimpleTest2</moduleInstanceId>
                <valueType>scalar</valueType>
                <parameterId>Q.m</parameterId>
                <locationId>H-2001</locationId>
                <timeSeriesType>external historical</timeSeriesType>
                <timeStep unit="day"/>
                <relativeViewPeriod unit="day" start="0" end="364"/>
                <readWriteMode>editing visible to all future task runs</readWriteMode>
        </timeSeriesSet>
</variable>
<variable>
        <variableId>Y1</variableId>
        <timeSeriesSet>
                <moduleInstanceId>UserSimpleTest2</moduleInstanceId>
                <valueType>scalar</valueType>
                <parameterId>Q.tra</parameterId>
                <locationId>H-2001</locationId>
                <timeSeriesType>external historical</timeSeriesType>
                <timeStep unit="day"/>
                <relativeViewPeriod unit="day" start="0" end="364"/>
                <readWriteMode>add originals</readWriteMode>
        </timeSeriesSet>
</variable>
<transformation id="user simple test 2">
        <user>
                <simple>
                        <expression>(a + b)*X1 - 3</expression>
                        <coefficientSet>
                                <coefficient id="a" value="1.34"/>
                                <coefficient id="b" value="2.5"/>
                        </coefficientSet>
                        <outputVariable>
                                <variableId>Y1</variableId>
                        </outputVariable>
                </simple>
        </user>
</transformation>
```

The example below uses an if statement. Here 'X' is a reference to a variable and 'a' is a reference to a coefficient.

```
<variable>
        <variableId>X</variableId>
        <timeSeriesSet>
                <moduleInstanceId>UserSimpleWithIfElseStatementTest</moduleInstanceId>
                <valueType>scalar</valueType>
                <parameterId>Q.m</parameterId>
                <locationId>H-2001</locationId>
                <timeSeriesType>external historical</timeSeriesType>
                <timeStep unit="day"/>
                <relativeViewPeriod unit="day" start="0" end="9"/>
                <readWriteMode>editing visible to all future task runs</readWriteMode>
        </timeSeriesSet>
</variable>
<variable>
        <variableId>Y</variableId>
        <timeSeriesSet>
                <moduleInstanceId>UserSimpleWithIfElseStatementTest</moduleInstanceId>
                <valueType>scalar</valueType>
                <parameterId>Q.tra</parameterId>
                <locationId>H-2001</locationId>
                <timeSeriesType>external historical</timeSeriesType>
                <timeStep unit="day"/>
                <relativeViewPeriod unit="day" start="0" end="9"/>
                <readWriteMode>add originals</readWriteMode>
        </timeSeriesSet>
</variable>
<transformation id="user simple with if else statement test">
        <user>
                <simple>
                        <!-- if X > 3, then the expression part if(X &gt; 3, 10.5, -2) is replaced with 10.5 -->
                        <!-- if X <= 3, then the expression part if(X &gt; 3, 10.5, -2) is replaced with -2 -->
                        <expression>if(X &gt; 3, 10.5, -2) + 5*a</expression>
                        <coefficientSet>
                                <coefficient id="a" value="1.5"/>
                        </coefficientSet>
                        <outputVariable>
                                <variableId>Y</variableId>
                        </outputVariable>
                </simple>
        </user>
</transformation>
```

The example below uses coefficientSetFunctions (available since build 30246, i.e. since stable build 2011.01). Here 'X' is a reference to a variable and 'a', 'b' and 'c' are references to coefficients. Here the coefficients are defined in coefficientSetFunctions, where (coef_a), (coef_b) and (coef_c) refer to location number attributes that are defined in the locationSets configuration file.

```
<variable>
        <variableId>X</variableId>
        <timeSeriesSet>
                <moduleInstanceId>UserSimpleWithCoefficientSetFunctionsTest</moduleInstanceId>
                <valueType>scalar</valueType>
                <parameterId>Q.m</parameterId>
                <locationId>locationWithAttributes4</locationId>
                <timeSeriesType>external historical</timeSeriesType>
                <timeStep unit="day"/>
                <relativeViewPeriod unit="day" start="0" end="9"/>
                <readWriteMode>editing visible to all future task runs</readWriteMode>
        </timeSeriesSet>
</variable>
<variable>
        <variableId>Y</variableId>
        <timeSeriesSet>
                <moduleInstanceId>UserSimpleWithCoefficientSetFunctionsTest</moduleInstanceId>
                <valueType>scalar</valueType>
                <parameterId>Q.tra</parameterId>
                <locationId>locationWithAttributes4</locationId>
                <timeSeriesType>external historical</timeSeriesType>
                <timeStep unit="day"/>
                <relativeViewPeriod unit="day" start="0" end="9"/>
                <readWriteMode>add originals</readWriteMode>
        </timeSeriesSet>
</variable>
<transformation id="user simple with coefficient set functions test">
        <user>
                <simple>
                        <expression>a*(X^2) + b*X + c</expression>
                        <coefficientSetFunctions>
                                <coefficient id="a" value="@coef_a@"/>
                                <coefficient id="b" value="@coef_b@"/>
                                <coefficient id="c" value="@coef_c@"/>
                        </coefficientSetFunctions>
                        <outputVariable>
                                <variableId>Y</variableId>
                        </outputVariable>
                </simple>
        </user>
</transformation>
```

The example below enables the use of string comparison in a coefficient, which may be usefull in combination with modifiable attributes. If you have for example your modifier attribute OPTION_TYPE that can be "hard" or "soft" you make a coefficient that as a boolean checks which type you are using. The numerical value of the boolean coefficient can then be used in the expression. The below example will be resolved as:

- if OPTION_TYPE = "hard" then COEF_OPTION = 1
- if OPTION_TYPE <> "hard" then COEF_OPTION = 0

if OPTION_TYPE = NULL (undefined) then the string is interpreted as an empty string "", so the COEF_OPTION will still be 0 (and not NaN).

This method is possible since FEWS 2014.01

boolean coefficient

```
<variable>
        <variableId>X</variableId>
    ....
</variable>
<variable>
        <variableId>Y</variableId>
    ....
</variable>
<transformation id="user simple with boolean coefficient test">
        <user>
                <simple>
            <expression>IF(COEF_OPTION == 1, X, Y)</expression>
            <coefficientSetFunctions>
               <coefficient id="COEF_OPTION" value="@OPTION_TYPE@==&quot;hard&quot;"/>
            </coefficientSetFunctions>
                <outputVariable>
                        <variableId>Y</variableId>
                    </outputVariable>
                </simple>
        </user>
</transformation>
```

You can use an attribute of a related location by using this format: @<relationID>:<attribute>@.