16 Pcraster Transformation (pcrTransformation)

What	*nameofinstance*.xml
Description	Configuration of the pcraster transformation module
schema location	https://fewsdocs.deltares.nl/schemas/version1.0/pcrTransformationSets.xsd

Introduction

The pcrTransformation model allows a direct link between data in DELFT-FEWS and pcraster using the PCraster API based on in-memory exchange of (XML) data. As such, Delft-Fews can use all available pcraster functions to process data. Pcraster documentation is available elsewhere.

Current status

At this point a working version is available within Delft-Fews that can be used to perform all operations supported by PCraster. This means that all time series data stored in Delft-Fews (grids and scalars) can be used as input to the module; all output is in grid format. If multiple timesteps are fed to the module at one each timestep will be run separately, i.e. it is not possible to refer to data of a previous timestep within the module. A pcraster model usually consists of a initial section (executed only once) and a dynamic section that is executed for each timestep. This version of the pcrTransformation only implements the initial section.

As of release 2008.3 the system includes support for dynamic scripts. Existing script will continue to work without modification (albeit significantly faster) and dynamic scripts are now supported.

Other documentation

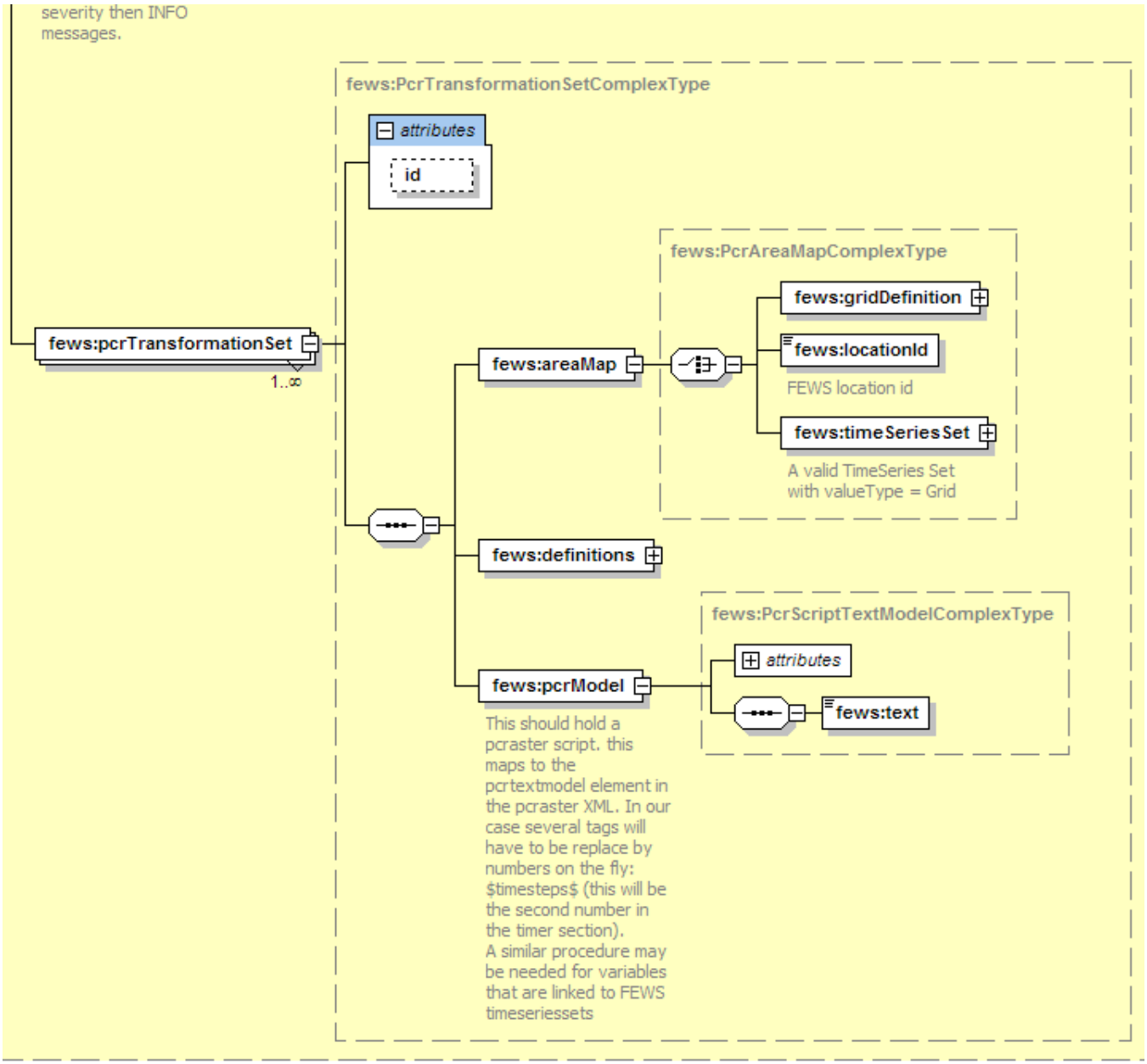
Examples are available in the attached pdf document.

Module Configuration

The schema diagram is shown below, three main sections can be distinguished:

1. Definition of the Area Map
2. Definition of internal, input and output variables
3. Definition of the PCRaster Model itself

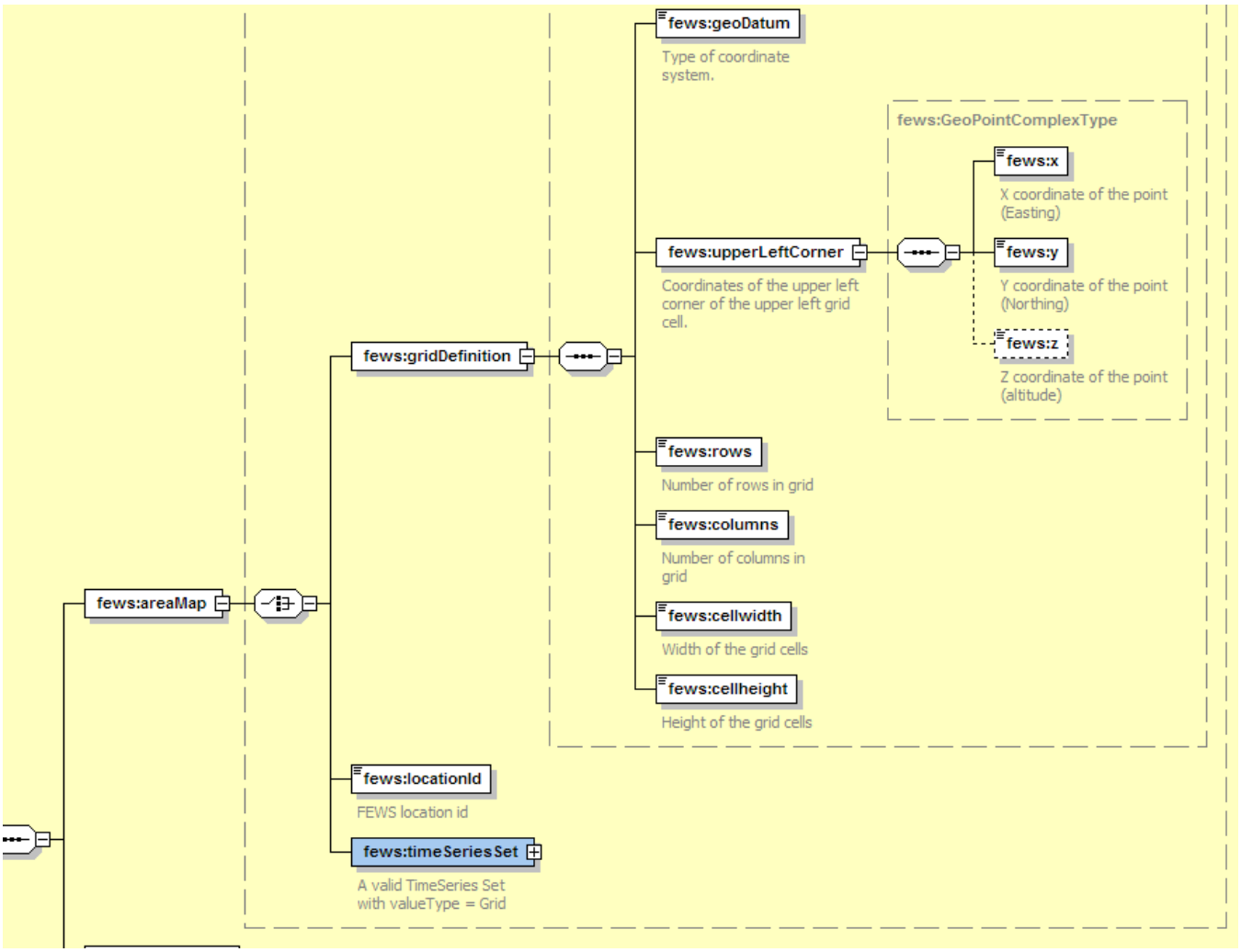
Multiple pcraster models can be configured in a single file.



Defining Area Map

The diagram below shows the possible options when defining an area map. The area map can be define using three methods:

1. Grid Definition (number of rows, columns etc.) ***(do not use if any of the other methods can be used)***
2. Grid location Id (this will use a grid definition the the Grids.xml file in the RegionConfigFiles section).
3. TimeSeriesSet which defines a Grid TimeSeries. (the grid definition is taken from the timeseries itself)



Examples

Here are few examples, which show the different methods available when defining an Area Map.

1. The area map is defined as a (FEWS) grid location id which refers to the grid definition at the same location within Grids.xml configuration file

```
<areaMap>
        <locationId>H-2002</locationId>
</areaMap>
```

2. The area map is defined as a (FEWS Grid) TimeSeries Set. For details on how to define TimeSeriesSet please refer FEWS Configuration Guide.

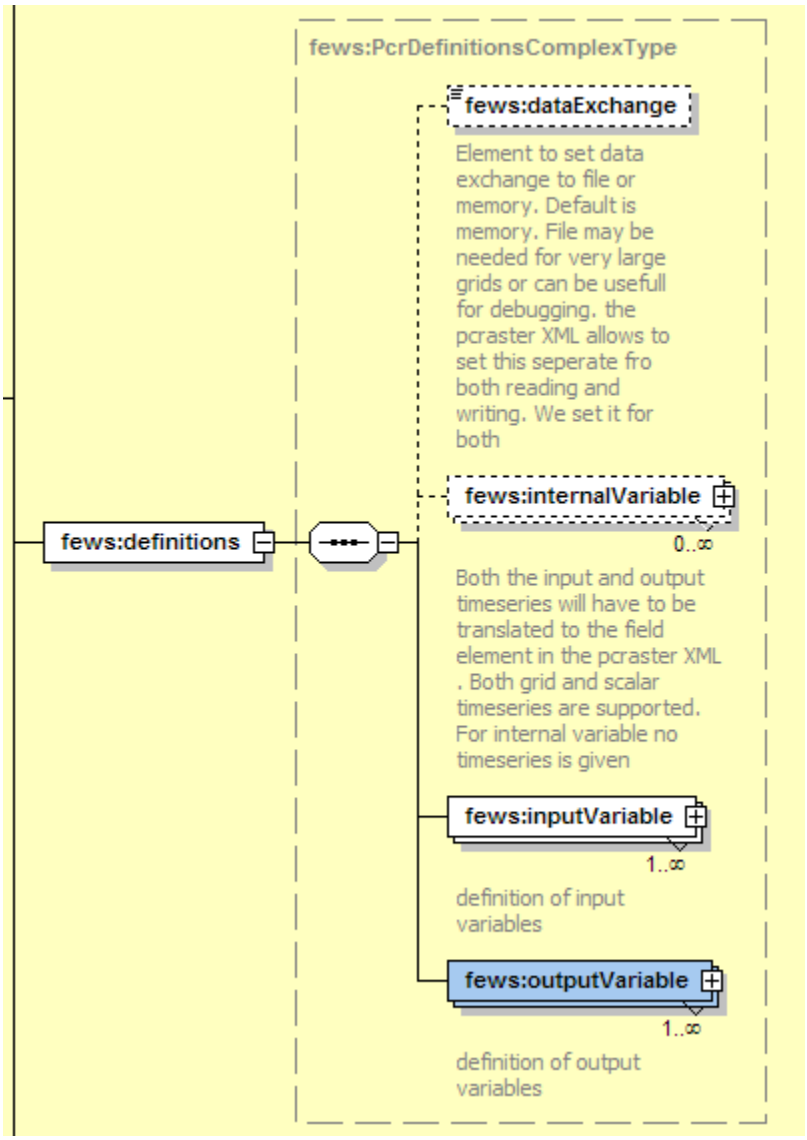
```
<areaMap>
        <moduleInstanceId>ImportGrid</moduleInstanceId>
        <valueType>grid</valueType>
        <parameterId>P.m</parameterId>
        <locationId>MeteoGrid</locationId>
        <timeSeriesType>external historical</timeSeriesType>
        <timeStep unit="hour"/>
        <relativeViewPeriod unit="day" start="0" end="1"/>
        <readWriteMode>read only</readWriteMode>
<areaMap>
```

3. The area map is given as Grid Definition (***Not recommended***) . The grid definition contains information such as GeoDatum, coordinates of upper left grid point, number of columns and rows and cell width and height.

```
<areaMap>
        <geoDatum>WGS 1984</geoDatum>
        <upperLeftCorner>
                <x>2</x>
                <y>1</y>
                <z>90</z>
        </upperLeftCorner>
        <rows>100</rows>
        <columns>100</columns>
        <cellwidth>0.1</cellwidth>
        <cellheight>0.1</cellheight>
</areaMap>
```

Defining internal, input and output variables

The diagram below gives an overview of schema on how to define PCRaster Variables:



Here one can define

1. Data Exchange Type- *memory* or *file,*
2. Internal PCRaster model variable, with a unique Id and data type. The data type (which are exactly similar to the data types used in PCRaster) can be

- boolean
- nominal
- ordinal
- scalar
- ldd (not yet implemented?)

- directional

3. Input PCRaster model variable. Use to get data from Delft-Fews and pass it on to the pcraster transformation module:

- Variable id (should be matching exactly as defined in the PCRaster text model),
- Data type (similar to that used for internal variables),
- Scalar type data to be passed to PCRaster, if different from the normal data value. At present the following options are available:

  - timeInJulian
  - timeAsDayofYear
  - timeAsDayofMonth
  - timeAsHourofDay
  - timeAsDaysElapsedSince
  - timeAsHoursElapsedSince
  - Reference date. Needed if the scalar type is defined as "timeAsDaysElapsedSince" or "timeAsHoursElapsedSince".
    Spatial type options are: spatial and non-spatial. Generally all grid input timeseries are treated as spatial data, while all scalar timeseries (or constant values) are treated as non-spatial. To treat the scalar timeseries (single data value per time) value or constant value as spatial, one can set this option to "spatial". By doing so, the grid (as defined by area map) will be filled with (single) data value from timeseries for a corresponding timestep. Hence for a given timestep, the input to the PCRaster model will be a grid with a constant value in all the grid cells.

However there is an exception to the above mentioned approach. If the input variable is a scalar timeseries at multiple locations (using LocationSetId in TimeSeriesSet definition) and spatial type is set to spatial, and then the following approach is used:

1. Create a grid (as defined by area map),
2. Get the data value for a current time from a timeseries for a first location,
3. For this location, get the geo-reference coordinates (X, Y),
4. Get the corresponding grid cell within which the above location lies,
5. Put the data value as that grid cell value,
6. Get the data value for the current time from the timeseries for next location.
7. Repeat the steps 3-6 till all the data for the current time at all locations is read and the value is put in the appropriate grid cell.* TimeSeriesSet, if the input data is a timeseries.

  - Value, if the input data is a constant value, and
  - External, if the data has to be read from the external PCRaster formatted file.

4. Output PCRaster model variable.

  - Variable id (should be matching exactly as defined in the PCRaster text model.
  - Data type (similar to that for internal variables), and
  - TimeSeriesSet, (at present) all output data should be grid timeseries.

Please not that the input variable should be regarded as read-only in the actual pcraster script. You should NOT try to modify them within the script. Make a copy in an other variable(e.g. mycopy = theinputvar;) if this is needed.

Examples

Here are few examples, showing different possibilities to define an interval, input and output variables. Refer to the comments for details:

```
<definitions>
	<!-- dataExchange  options =  Memory -->
	<dataExchange>memory</dataExchange>
	<!-- internalVariable  name used within the PCRaster Test Model -->
	<internalVariable variableId="blnmap" dataType="boolean"/>
	<!-- internalVariable name used within the PCRaster Test Model
, now with the dataType as scalar -->
	<internalVariable variableId="toSpatial" dataType="scalar"/>
	<!-- InputVariable which refers to the external data file -->
	<inputVariablevariableId="externalVar" dataType="scalar">
		<external>d://x.txt</external>
	</inputVariable>
	<!---Input Variable which refers to TimeSeriesGrid Array i.e. Grid as input -->
	<inputVariablevariableId="input" dataType="scalar" convertDatum="false">
		<timeSeriesSet>
			<moduleInstanceId>ModuleInstance</moduleInstanceId>
			<valueType>grid</valueType>
			<parameterId>H.tidal</parameterId>
			<locationId>H-2002</locationId>
			<timeSeriesType>external historical</timeSeriesType>
			<timeStep unit="day"/>
			<relativeViewPeriod unit="day" start="0" end="1"/>
			<readWriteMode>add originals</readWriteMode>
		</timeSeriesSet>
	</inputVariable>
	<!-- InputVariable which refers to the TimeSeries Float Array i.e, scalar value per
		time, non spatial in nature . In other words, a value per time distributed
		constantly over the whole grid for calculation purpose -->
	<inputVariablevariableId="input" dataType="scalar" convertDatum="false">
		<timeSeriesSet>
			<moduleInstanceId> ModuleInstance</moduleInstanceId>
			<valueType>scalar</valueType>
			<parameterId>H.tidal</parameterId>
			<locationId>H-2002</locationId>
			<timeSeriesType>external historical</timeSeriesType>
			<timeStep unit="day"/>
			<relativeViewPeriod unit="day" start="0" end="1"/>
			<readWriteMode>add originals</readWriteMode>
		</timeSeriesSet>
	</inputVariable>
	<!-- InputVariable which refers to the Constant Value i.e, a constant scalar value
		irrespective of time and non spatial in nature. In other words, a constant
		value distributed constantly over the whole grid for calculation purpose -->
	<inputVariablevariableId="constant" dataType="scalar">
		<value>10</value>
	</inputVariable>

	<!-- InputVariable which refers to the TimeSeries Float Array for multiple location
		(given by locationSetID) i.e,   scalar value per time, spatial in nature (as
		given by spatialType),  and  defined only at grid cells where contains the
		location. In other words, the  grid cell which contains the georeference
		position of the location -->
	<inputVariablevariableId="input" dataType="scalar"spatialType="spatial" convertDatum="false">
		<timeSeriesSet>
			<moduleInstanceId>ModuleInstance</moduleInstanceId>
			<valueType>scalar</valueType>
			<parameterId>H.tidal</parameterId>
			<locationId>TestLocLiesWithinGrid_H-2002</locationId>
			<timeSeriesType>external historical</timeSeriesType>
			<timeStep unit="day"/>
			<relativeViewPeriod unit="day" start="0" end="1"/>
			<readWriteMode>add originals</readWriteMode>
		</timeSeriesSet>
	</inputVariable>
	<!-- InputVariable which refers to the TimeSeries Float Array  (NonSpatial) ,
		however  the time is passed to PCRaster  (scalarType =
		timeAsDaysElapsedSince).    For the scalar Type defined as time "Elapsed
		Since" the reference Date  has to be defined  -->

	<inputVariablevariableId="input" dataType="scalar" scalarType="timeAsDaysElapsedSince" referenceDate="1984-10-30" convertDatum
="false">
```

```
                <timeSeriesSet>
                        <moduleInstanceId>ModuleInstance</moduleInstanceId>
                        <valueType>scalar</valueType>
                        <parameterId>H.tidal</parameterId>
                        <locationId>H-2002</locationId>
                        <timeSeriesType>external historical</timeSeriesType>
                        <timeStep unit="day"/>
                        <relativeViewPeriod unit="day" start="0" end="1"/>
                        <readWriteMode>add originals</readWriteMode>
                </timeSeriesSet>

        </inputVariable>


        <outputVariable variableId="transfmap" dataType="scalar" convertDatum="false">
                <timeSeriesSet>
                        <moduleInstanceId>OutputModuleInstance</moduleInstanceId>
                        <valueType>grid</valueType>
                        <parameterId>H.updated</parameterId>
                        <locationId>H-2003</locationId>
                        <timeSeriesType>external historical</timeSeriesType>
                        <timeStep unit="day"/>
                        <relativeViewPeriod unit="day" start="0" end="1"/>
                        <readWriteMode>add originals</readWriteMode>
                </timeSeriesSet>
        </outputVariable>
```

Defining the PCRaster Model

In this section, one can provide the PCRaster model as simple ASCII text. The text model given here is in fact the valid PCRaster model and can be run directly using PCRaster model, except that it does not contain any area map or variable definition part. All the variables used within this model should appear in the definition section as described in the section above.

Example

Here is an example, showing how to configure a PCRaster Model.

```
<pcrModel id="String">
    <text>
        <!---PCRaster accepts #  as Comment--->
        # there is no dynamic section!
        # initial
        # result should be the grid within constant value of 1.8 and 0.8
        # generate unique Id's
        Unq = uniqueid(boolean(input));
        transfmap = spreadzone(ordinal(cover(Unq,0)),0,1);
    </text>
</pcrModel>
```

Please remember, the PCRaster model, which is defined here, is full PCRaster model script written in PCRaster Modelling Environment language. Take care that all the variables ids defined in Variable definition section matches the variables used here in the model. In other words, the model defined here should be a PCRaster compatible script.

Sample configuration to perform a typical PCRaster Transformation

A working sample configuration for PcrTransformation is shown as below:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Solar radiation module demonstration configuration -->
<pcrTransformationSets 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
/pcrTransformationSets.xsd" version="1.1">
        <logLevel>WARN</logLevel>
        <pcrTransformationSet id="Potradiation">
                <areaMap>
                        <locationId>Radiation</locationId>
                </areaMap>
                <definitions>
                        <dataExchange>memory</dataExchange>
                        <inputVariable variableId="Altitude" dataType="scalar" convertDatum="false"
spatialType="spatial">
                                <timeSeriesSet>
                                        <moduleInstanceId>Radiation</moduleInstanceId>
                                        <valueType>grid</valueType>
```

```
                                <parameterId>Time.event</parameterId>
                                <locationId>Radiation</locationId>
                                <timeSeriesType>external historical</timeSeriesType>
                                <timeStep unit="minute" multiplier="15"/>
                                <relativeViewPeriod unit="hour" start="0" end="48"/>
                                <readWriteMode>add originals</readWriteMode>
                        </timeSeriesSet>
                </inputVariable>
                <inputVariable variableId="YearDay" dataType="scalar" convertDatum="false" scalarType="
timeAsDayofYear">
                        <timeSeriesSet>
                                <moduleInstanceId>Radiation</moduleInstanceId>
                                <valueType>scalar</valueType>
                                <parameterId>Time.event</parameterId>
                                <locationId>Radiation</locationId>
                                <timeSeriesType>external historical</timeSeriesType>
                                <timeStep unit="minute" multiplier="15"/>
                                <relativeViewPeriod unit="hour" start="0" end="48"/>
                                <readWriteMode>add originals</readWriteMode>
                        </timeSeriesSet>
                </inputVariable>
                <inputVariable variableId="Hour" dataType="scalar" convertDatum="false" scalarType="
timeAsHourofDay">
                        <timeSeriesSet>
                                <moduleInstanceId>Radiation</moduleInstanceId>
                                <valueType>scalar</valueType>
                                <parameterId>Time.event</parameterId>
                                <locationId>Radiation</locationId>
                                <timeSeriesType>external historical</timeSeriesType>
                                <timeStep unit="minute" multiplier="15"/>
                                <relativeViewPeriod unit="hour" start="0" end="48"/>
                                <readWriteMode>add originals</readWriteMode>
                        </timeSeriesSet>
                </inputVariable>
                <!-- Total potential Solar radiation -->
                <outputVariable variableId="SL" dataType="scalar" convertDatum="false">
                        <timeSeriesSet>
                                <moduleInstanceId>Radiation</moduleInstanceId>
                                <valueType>grid</valueType>
                                <parameterId>Sol.pot</parameterId>
                                <locationId>Radiation</locationId>
                                <timeSeriesType>external historical</timeSeriesType>
                                <timeStep unit="minute" multiplier="15"/>
                                <relativeViewPeriod unit="hour" start="0" end="48"/>
                                <readWriteMode>add originals</readWriteMode>
                        </timeSeriesSet>
                </outputVariable>
                <!-- Diffuse radiation -->
                <outputVariable variableId="SLDF" dataType="scalar" convertDatum="false">
                        <timeSeriesSet>
                                <moduleInstanceId>Radiation</moduleInstanceId>
                                <valueType>grid</valueType>
                                <parameterId>Sol.pot.diffuse</parameterId>
                                <locationId>Radiation</locationId>
                                <timeSeriesType>external historical</timeSeriesType>
                                <timeStep unit="minute" multiplier="15"/>
                                <relativeViewPeriod unit="hour" start="0" end="48"/>
                                <readWriteMode>add originals</readWriteMode>
                        </timeSeriesSet>
                </outputVariable>
                <!-- direct radiation -->
                <outputVariable variableId="SLDR" dataType="scalar" convertDatum="false">
                        <timeSeriesSet>
                                <moduleInstanceId>Radiation</moduleInstanceId>
                                <valueType>grid</valueType>
                                <parameterId>Sol.pot.direct</parameterId>
                                <locationId>Radiation</locationId>
                                <timeSeriesType>external historical</timeSeriesType>
                                <timeStep unit="minute" multiplier="15"/>
                                <relativeViewPeriod unit="hour" start="0" end="48"/>
                                <readWriteMode>add originals</readWriteMode>
```

```
                                </timeSeriesSet>
                        </outputVariable>
                </definitions>
                <pcrModel id="String">
                        <text><![CDATA[
#! --unittrue --degrees
# Test script to determine radiation over a grid.
#
# Inputs from Delft-Fews into this script
# - YearDay -> scalar with  day since beginning of year
# - Hour of day -> Fractional hour of day (e.g. 12.5 = 12:30)
# Ouputs to FEWS
# - SL -> Total Solar radiation
#
# This version determines Clear Sky radiation assuming a level surface using a uniform
# altitude. This level is configured in the script below.
Altitude=spatial(10);

Latitude = ycoordinate(boolean(Altitude));
Longitude = xcoordinate(boolean(Altitude));

Day =YearDay;
pi = 3.1416;
Sc       = 1367.0;          # Solar constant (Gates, 1980) [W/m2]
Trans    = 0.6;             # Transmissivity tau (Gates, 1980)

AtmPcor = ((288-0.0065*Altitude)/288)**5.256;            # atm pressure corr [-]

# Solar geometry
# ----------------------------
# SolDec  :declination sun per day  between +23 and -23 [deg]
# HourAng :hour angle [-] of sun during day
# SolAlt  :solar altitude [deg], height of sun above horizon
# SolDec  = -23.4*cos(360*(Day+10)/365);
# Now added a new function that should work on all latitudes!
theta    =(Day-1)*360/365;  # day expressed in degrees

# Time change equal to 4 min per degree longtitude
# Assume the time input to be GMT
HourS = Hour + (Longitude * 4/60);

SolDec =180/pi * (0.006918-0.399912 * cos(theta)+0.070257 * sin(theta) -   0.006758 * cos(2*theta)+0.000907 *
sin(2*theta) -  0.002697 * cos(3*theta)+0.001480 * sin(3*theta));

HourAng = 15*(HourS-12.01);

SolAlt  = scalar(asin(scalar(sin(Latitude)*sin(SolDec)+cos(Latitude)*
           cos(SolDec)*cos(HourAng))));

# Solar azimuth
# ----------------------------
# SolAzi  :angle solar beams to N-S axes earth [deg]
 SolAzi = scalar(acos((sin(SolDec)*cos(Latitude)-cos(SolDec)*
          sin(Latitude)*cos(HourAng))/cos(SolAlt)));
 SolAzi = if(HourS le 12 then SolAzi else 360 - SolAzi);

 Slope = spatial(0.0001);
 Aspect = spatial(1);

# Surface azimuth
# ----------------------------
# cosIncident :cosine of angle of incident; angle solar beams to angle surface
 cosIncident = sin(SolAlt)*cos(Slope)+cos(SolAlt)*sin(Slope)
               *cos(SolAzi-Aspect);

# Radiation outer atmosphere
# ----------------------------
 OpCorr = Trans**((sqrt(1229+(614*sin(SolAlt))**2)
          -614*sin(SolAlt))*AtmPcor);    # correction for air masses [-]
 Sout   = Sc*(1+0.034*cos(360*Day/365)); # radiation outer atmosphere [W/m2]
 Snor   = Sout*OpCorr;                  # rad on surface normal to the beam [W/m2]
```

```
# Radiation at DEM
# ----------------------------
# Sdir   :direct sunlight on a horizontal surface [W/m2] if no shade
# Sdiff  :diffuse light [W/m2] for shade and no shade
# Stot   :total incomming light Sdir+Sdiff [W/m2] at Hour
# Radiation :avg of Stot(Hour) and Stot(Hour-HourStep)
# NOTE: PradM only valid for HourStep and DayStep = 1
 Sdir   = if(Snor*cosIncident<0,0.0,Snor*cosIncident);
 Sdiff  = if(Sout*(0.271-0.294*OpCorr)*sin(SolAlt)<0, 0.0,
          Sout*(0.271-0.294*OpCorr)*sin(SolAlt));

# Fill in missing values with areaaaverage
SLDR=cover((Sdir*1),(Altitude * 0) + areaaverage(Sdir*1,boolean(Altitude))); # hourly rad [W/m2]
SLDF=cover((Sdiff*1),(Altitude * 0) + areaaverage(Sdiff*1,boolean(Altitude))); # hourly rad [W/m2]

SL   = SLDR + SLDF;       # Total rad  in [W/m2]



                        ]]></text>
                </pcrModel>
        </pcrTransformationSet>
</pcrTransformationSets>
```

Points precipitation to grid example

```
<?xml version="1.0" encoding="UTF-8"?>
<pcrTransformationSets xmlns="http://www.wldelft.nl/fews" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.wldelft.nl/fews
http://fews.wldelft.nl/schemas/version1.0/pcrTransformationSets.xsd" version="1.1">
        <logLevel>WARN</logLevel>
        <pcrTransformationSet id="Thiessen">
                <areaMap>
                        <locationId>FineGrid</locationId>
                </areaMap>
                <definitions>
                        <dataExchange>memory</dataExchange>
                        <inputVariable variableId="P" dataType="scalar" convertDatum="false">
                                <timeSeriesSet>
                                        <moduleInstanceId>ImportPubRts</moduleInstanceId>
                                        <valueType>scalar</valueType>
                                        <parameterId>P.obs</parameterId>
                                        <locationSetId>MetGauges_P.obs</locationSetId>
                                        <timeSeriesType>external historical</timeSeriesType>
                                        <timeStep unit="minute" multiplier="10"/>
                                        <relativeViewPeriod unit="hour" start="-96" startOverrulable="true"
end="0"/>
                                        <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                        </inputVariable>
                        <outputVariable variableId="MeasMap" dataType="scalar" convertDatum="false">
                                <timeSeriesSet>
                                        <moduleInstanceId>PrecipitationGaugeToGrid_Historical</moduleInstanceId>
                                        <valueType>grid</valueType>
                                        <parameterId>P.obs</parameterId>
                                        <locationId>FineGrid</locationId>
                                        <timeSeriesType>simulated forecasting</timeSeriesType>
                                        <timeStep unit="minute" multiplier="10"/>
                                        <relativeViewPeriod unit="hour" start="-96" end="0" startOverrulable="
true"/>
                                        <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                        </outputVariable>
                </definitions>
                <pcrModel id="String">
                        <text><![CDATA[#! --unittrue --degrees
dynamic
# Simple Thiesen polygons to get spatial average precipitation on a grid

# Creat unique Id's for input stations
Unq = uniqueid(boolean(P));
# Now generate polygons and fill those
GaugeArea = spreadzone(ordinal(cover(Unq,0)),0,1);
MeasMap = areaaverage(P,GaugeArea);
]]></text>
                </pcrModel>
        </pcrTransformationSet>
</pcrTransformationSets>
```

PCRaster installation

For 64 bit support PCRaster needs to be installed manually. See: PCRaster