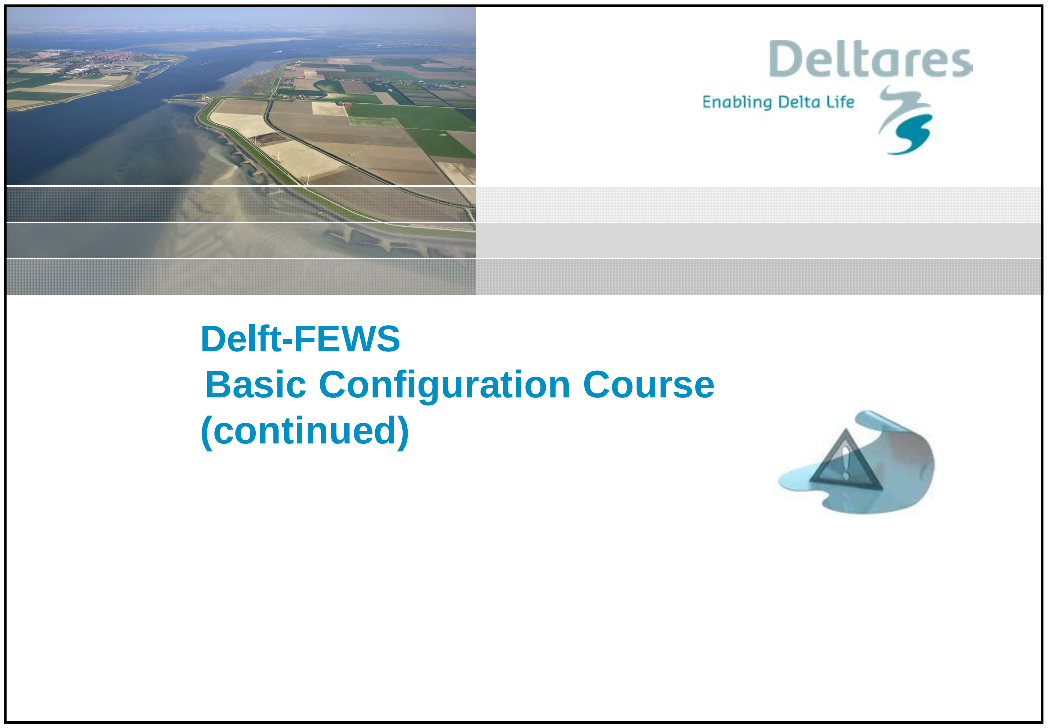
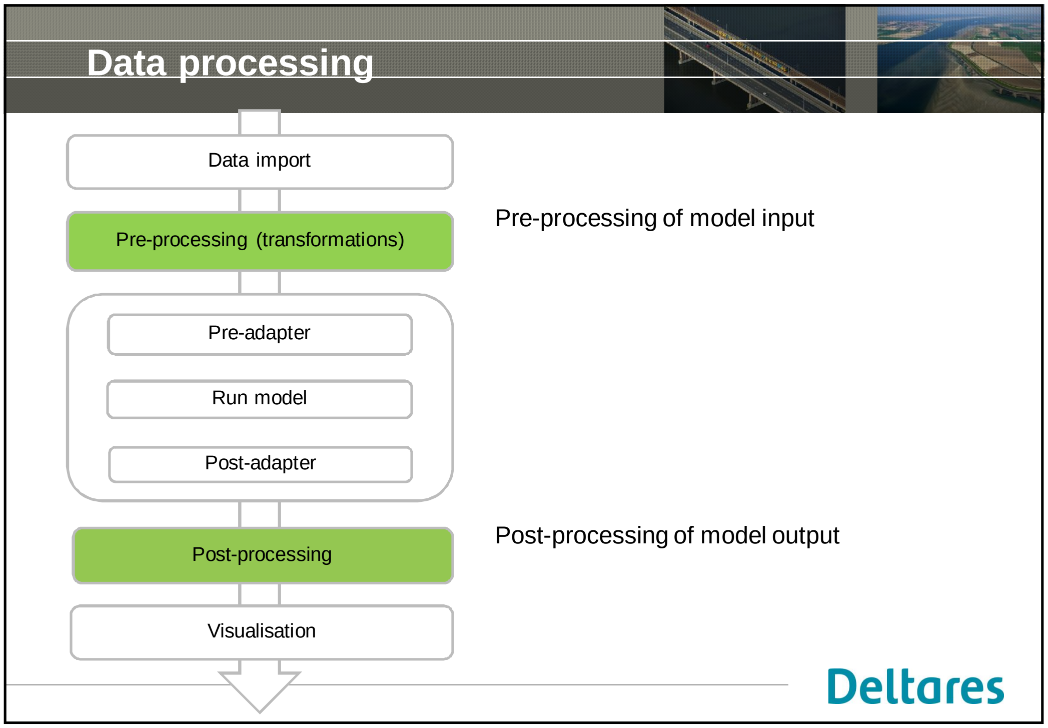


Deltares
Enabling Delta Life

Delft-FEWS
Basic Configuration Course (continued)

Processing data



Data processing

Data import

Pre-processing (transformations)

Pre-adapter

Run model

Post-adapter

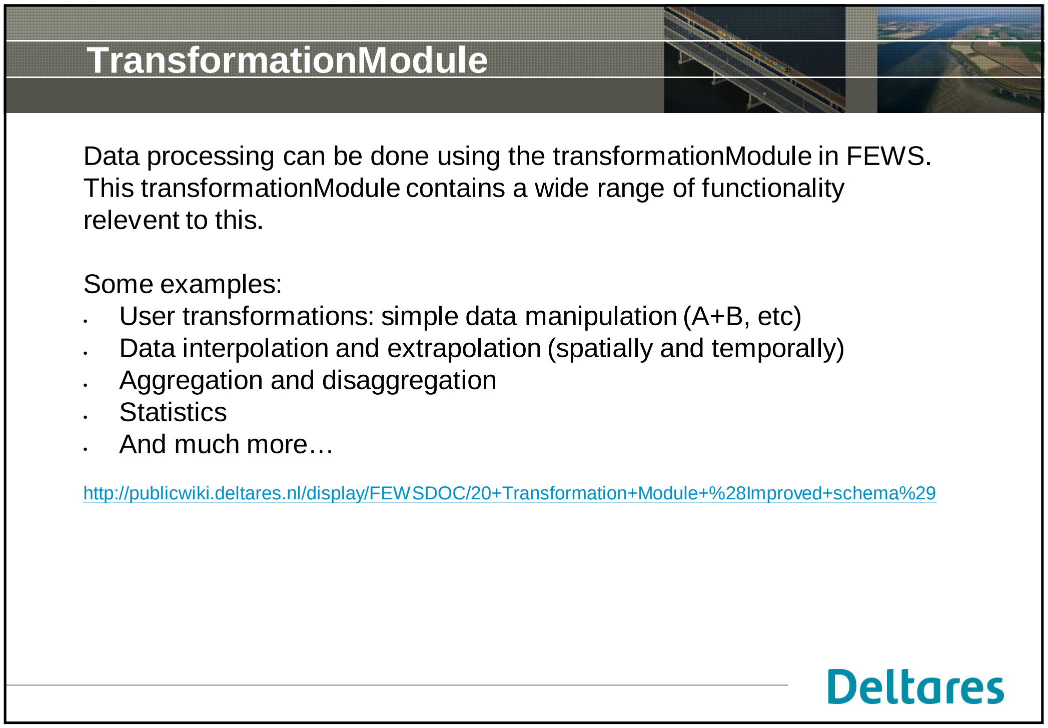
Post-processing

Visualisation

Pre-processing of model input

Post-processing of model output

Deltares



TransformationModule

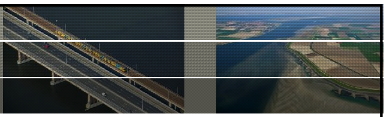
Data processing can be done using the transformationModule in FEWS. This transformationModule contains a wide range of functionality relevent to this.

Some examples:

- User transformations: simple data manipulation (A+B, etc)
- Data interpolation and extrapolation (spatially and temporally)
- Aggregation and disaggregation
- Statistics
- And much more…

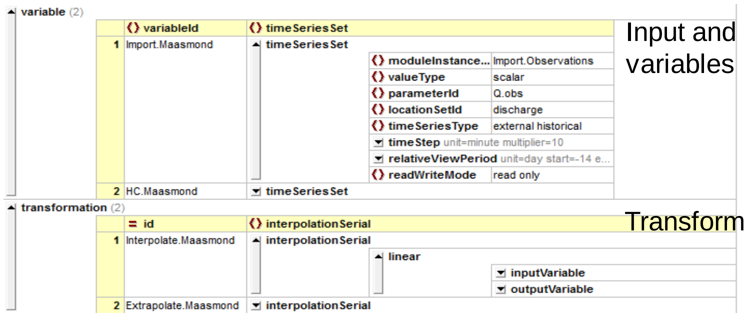
http://publicwiki.deltares.nl/display/FEWSDOC/20+Transformation+Module+%28Improved+schema%29

Deltares



TransformationModule

Basic structure:



variable (2)

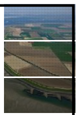
	variableId	timeSeriesSet		
1	Import.Maasmond	timeSeriesSet		
			moduleInstance...	Import.Observations
			valueType	scalar
			parameterId	Q.obs
			locationSetId	discharge
			timeSeriesType	external historical
			timeStep	unit=minute multiplier=10
			relativeViewPeriod	unit=day start=-14 e...
			readWriteMode	read only
2	HC.Maasmond	timeSeriesSet		

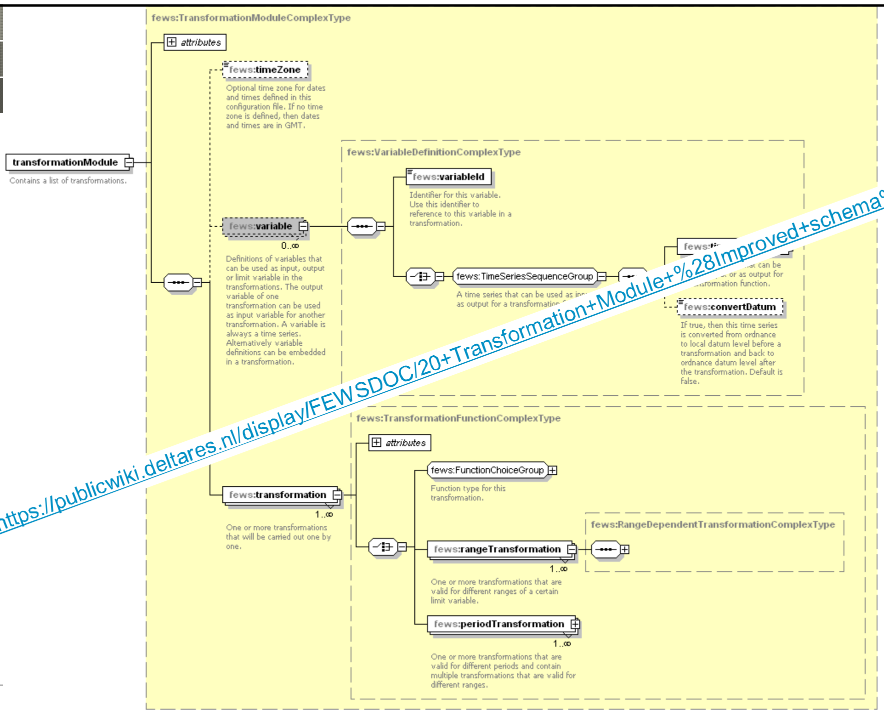
transformation (2)

	= id	interpolationSerial		
1	Interpolate.Maasmond	interpolationSerial		
		linear		
			inputVariable	
			outputVariable	
2	Extrapolate.Maasmond	interpolationSerial		

Input and output variables

Transformations




fews:TransformationModuleComplexType
attributes
fews:timeZone
transformationModule
fews:variable
0..∞
fews:VariableDefinitionComplexType
fews:variableId
fews:TimeSeriesSequenceGroup
fews:ti
fews:convertDatum
fews:TransformationFunctionComplexType
attributes
fews:FunctionChoiceGroup
fews:transformation
1..∞
fews:rangeTransformation
1..∞
fews:RangeDependentTransformationComplexType
fews:periodTransformation
1..∞

Contains a list of transformations.

Optional time zone for dates and times defined in this configuration file. If no time zone is defined, then dates and times are in GMT.

Definitions of variables that can be used as input, output or limit variable in the transformations. The output variable of one transformation can be used as input variable for another transformation. A variable is always a time series. Alternatively variable definitions can be embedded in a transformation.

Identifier for this variable. Use this identifier to reference to this variable in a transformation.

A time series that can be used as inp as output for a transformation f

at can be or as output for nsformation function.

If true, then this time series is converted from ordnance to local datum level before a transformation and back to ordnance datum level after the transformation. Default is false.

Function type for this transformation.

One or more transformations that will be carried out one by one.

One or more transformations that are valid for different ranges of a certain limit variable.

One or more transformations that are valid for different periods and contain multiple transformations that are valid for different ranges.

https://publicwiki.deltares.nl/display/FEWSDOC/20+Transformation+Module+%28Improved+schema%29

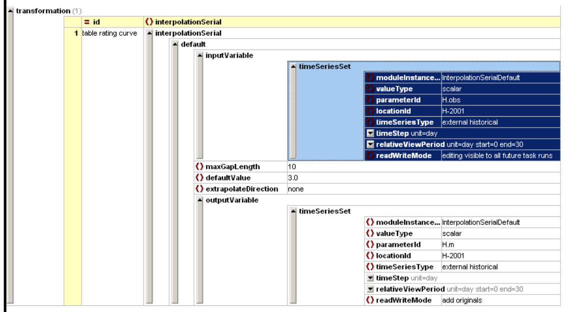
Transformation Module – Interpolation

Serial Interpolation methods

- Default
- Block
- Linear
- Extrapolation

Spatial Interpolation methods

- Average
- Bilinear
- Closest distance
- Inverse distance
- Kriging
- Max / min
- Sum
- Thiessen polygon
- Triangulation
- Weighted



```
transformation (1)
      = id              {} interpolationSerial
    1 table rating curve   interpolationSerial
                              default
                                 inputVariable
                                          timeSeriesSet
                                                {} moduleInstance...  InterpolationSerialDefault
                                                {} valueType          scalar
                                                {} parameterId        H.obs
                                                {} locationId         H-2001
                                                {} timeSeriesType     external historical
                                                   timeStep unit=day
                                                   relativeViewPeriod unit=day start=0 end=30
                                                {} readWriteMode      editing visible to all future task runs
                              {} maxGapLength          10
                              {} defaultValue          3.0
                              {} extrapolateDirection  none
                                 outputVariable
                                          timeSeriesSet
                                                {} moduleInstance...  InterpolationSerialDefault
                                                {} valueType          scalar
                                                {} parameterId        H.m
                                                {} locationId         H-2001
                                                {} timeSeriesType     external historical
                                                   timeStep unit=day
                                                   relativeViewPeriod unit=day start=0 end=30
                                                {} readWriteMode      add originals
```

Exercise

Exercise 9: Fill gaps in imported timeseries

- Analyse module that fills gaps in the water level time series
- Add the new module instance to the module instance descriptors
- Configure a workflow that executes the interpolation module instance
- Check results!

Transformation Module - Interpolation

Possible problems / Issues to note

Time Series Set to Interpolate

- ModuleInstanceId of series!!! (ImportTelemetry)
- Relative View Period

When interpolating to fill gaps

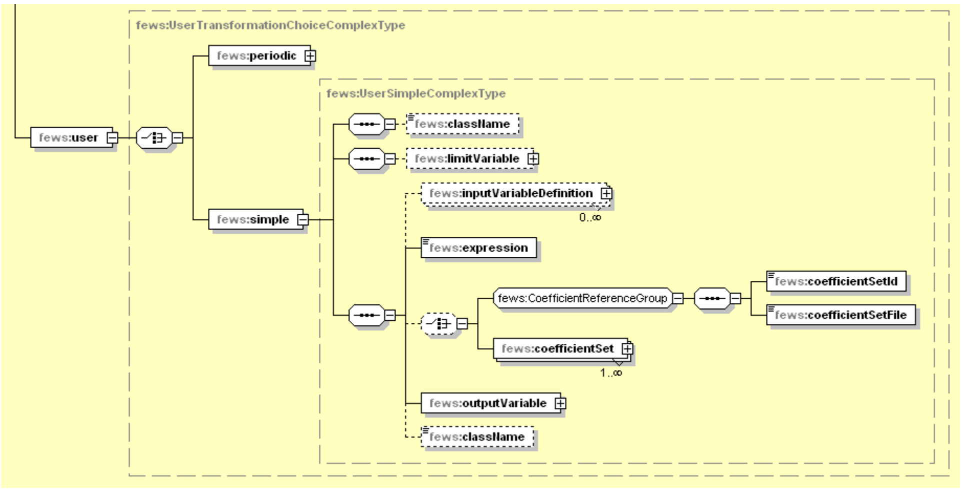
- Input time series same as output time series

Using LocationSets and/or Locations

- Linear interpolation – can be done using a locationSet
- Default – take care as default value may vary per location

Deltares

Transformation Module - Examples
User defined function

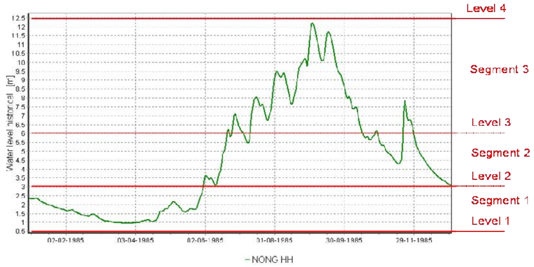


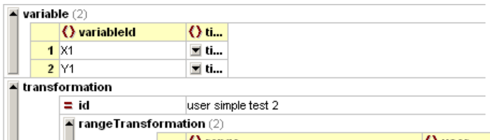
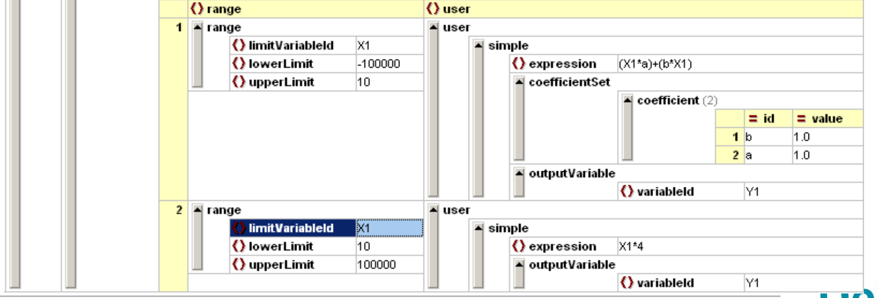
Deltares

Transformation Module - Examples
User defined function

Different function / parameters
per segment / range

Define ranges in rating curves file
/ coefficientsets file



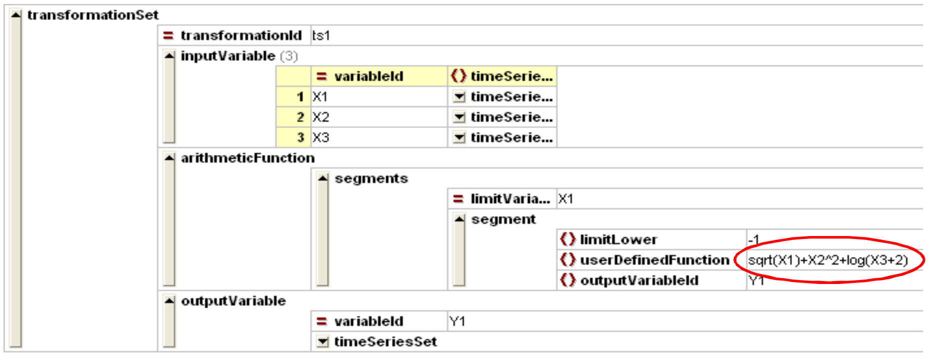



```
variable (2)
          variableId   ti...
      1   X1           ti...
      2   Y1           ti...
transformation
    = id                      user simple test 2
    rangeTransformation (2)
              range                                   user
          1   range                                   user
                  limitVariableId   X1                    simple
                  lowerLimit        -100000                   expression       (X1*a)+(b*X1)
                  upperLimit        10                        coefficientSet
                                                                  coefficient (2)
                                                                          id   value
                                                                      1   b    1.0
                                                                      2   a    1.0
                                                              outputVariable
                                                                  variableId   Y1
          2   range                                   user
                  limitVariableId   X1                    simple
                  lowerLimit        10                        expression       X1*4
                  upperLimit        100000                    outputVariable
                                                                  variableId   Y1
```

Transformation Module - Examples
User defined function

Generic function parser used

“Any” function can be defined

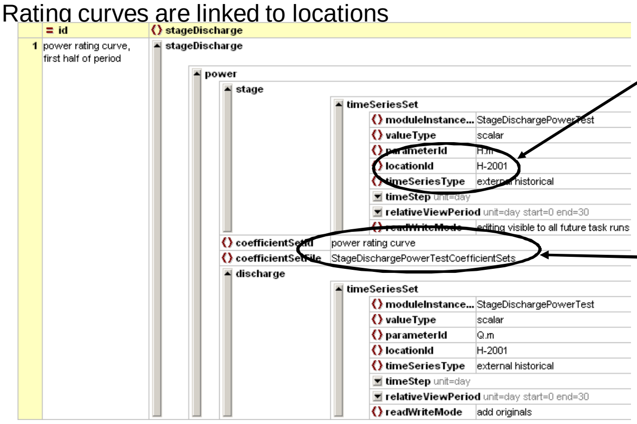


```
transformationSet
    = transformationId   ts1
    inputVariable (3)
                = variableId   timeSerie...
            1   X1             timeSerie...
            2   X2             timeSerie...
            3   X3             timeSerie...
    arithmeticFunction
            segments
                = limitVaria...   X1
                segment
                    limitLower              -1
                    userDefinedFunction     sqrt(X1)+X2^2+log(X3+2)
                    outputVariableId        Y1
    outputVariable
            = variableId   Y1
            timeSeriesSet
```

Transformation Module - Examples
Rating curve

- Same approach as other transformations using stageDischarge class
- Convention on inputId / ouputId & parameter names
- Rating curves are contained in the CoefficientSetFiles (RatingCurves.xml)

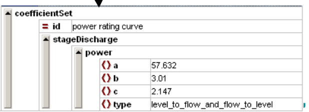
Rating curves are linked to locations



```
   = id                        {} stageDischarge
1  power rating curve,         stageDischarge
   first half of period          power
                                   stage
                                     timeSeriesSet
                                       {} moduleInstance...   StageDischargePowerTest
                                       {} valueType           scalar
                                       {} parameterId         H.m
                                       {} locationId          H-2001
                                       {} timeSeriesType      external historical
                                          timeStep unit=day
                                          relativeViewPeriod unit=day start=0 end=30
                                       {} readWriteMode       editing visible to all future task runs
                                   {} coefficientSetId    power rating curve
                                   {} coefficientSetFile  StageDischargePowerTestCoefficientSets
                                   discharge
                                     timeSeriesSet
                                       {} moduleInstance...   StageDischargePowerTest
                                       {} valueType           scalar
                                       {} parameterId         Q.m
                                       {} locationId          H-2001
                                       {} timeSeriesType      external historical
                                          timeStep unit=day
                                          relativeViewPeriod unit=day start=0 end=30
                                       {} readWriteMode       add originals
```

One location or locationSet

Use coefficients from CoefficientSetFile



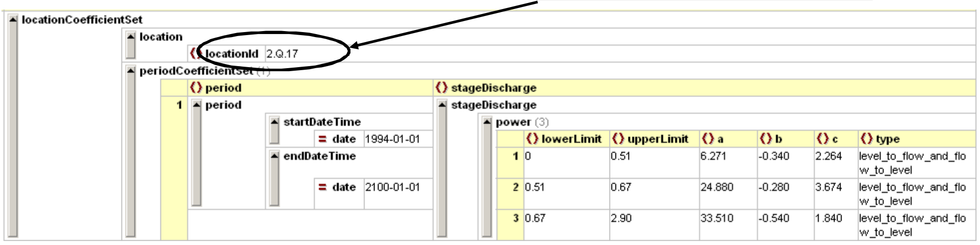
```
coefficientSet
    = id  power rating curve
    stageDischarge
      power
        {} a     57.632
        {} b     3.01
        {} c     2.147
        {} type  level_to_flow_and_flow_to_level
```

Transformation Module - Examples
Rating curve

CoefficientSets

- StageDischargeCoefficientSets.xml (name not fixed)
- In CoefficientSetsFiles folder
- Referenced in Transformation ModuleInstance
- Rating curve defined only once

Referenced by locationId

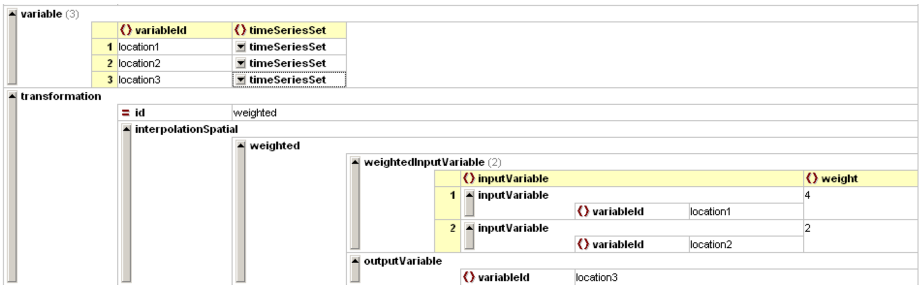


```
locationCoefficientSet
  location
    {} locationId  2.Q.17
  periodCoefficientSet (1)
       {} period                          {} stageDischarge
    1  period                             stageDischarge
         startDateTime                      power (3)
           = date  1994-01-01
         endDateTime
           = date  2100-01-01
```

	{} lowerLimit	{} upperLimit	{} a	{} b	{} c	{} type
1	0	0.51	6.271	-0.340	2.264	level_to_flow_and_flow_to_level
2	0.51	0.67	24.880	-0.280	3.674	level_to_flow_and_flow_to_level
3	0.67	2.90	33.510	-0.540	1.840	level_to_flow_and_flow_to_level

Transformation Module - Examples
Weighted averaging

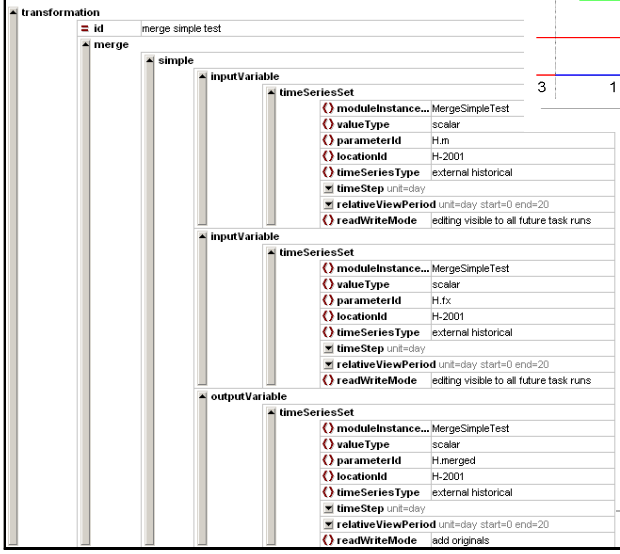
- Convention on inputId / ouputId & parameter names
- formula: (X1 * a + X2 * b + X3 * c) / (a + b + c)
- number of locations is flexible, missing locations will be left out



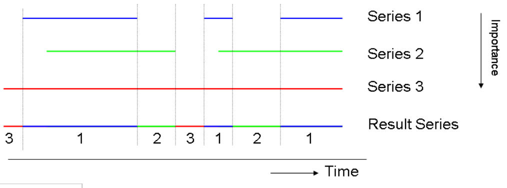
```
variable (3)
        variableId    timeSeriesSet
    1   location1     timeSeriesSet
    2   location2     timeSeriesSet
    3   location3     timeSeriesSet
transformation
        id            weighted
        interpolationSpatial
              weighted
                    weightedInputVariable (2)
                              inputVariable                                weight
                         1    inputVariable                                4
                                          variableId    location1
                         2    inputVariable                                2
                                          variableId    location2
                    outputVariable
                              variableId    location3
```

Transformation Module - Examples
Data Hierarchy / Merge

Fallback options to complete a series



```
transformation
      id        merge simple test
      merge
            simple
                  inputVariable
                        timeSeriesSet
                              moduleInstance...   MergeSimpleTest
                              valueType           scalar
                              parameterId         H.m
                              locationId          H-2001
                              timeSeriesType      external historical
                              timeStep unit=day
                              relativeViewPeriod unit=day start=0 end=20
                              readWriteMode       editing visible to all future task runs
                  inputVariable
                        timeSeriesSet
                              moduleInstance...   MergeSimpleTest
                              valueType           scalar
                              parameterId         H.fx
                              locationId          H-2001
                              timeSeriesType      external historical
                              timeStep unit=day
                              relativeViewPeriod unit=day start=0 end=20
                              readWriteMode       editing visible to all future task runs
                  outputVariable
                        timeSeriesSet
                              moduleInstance...   MergeSimpleTest
                              valueType           scalar
                              parameterId         H.merged
                              locationId          H-2001
                              timeSeriesType      external historical
                              timeStep unit=day
                              relativeViewPeriod unit=day start=0 end=20
                              readWriteMode       add originals
```

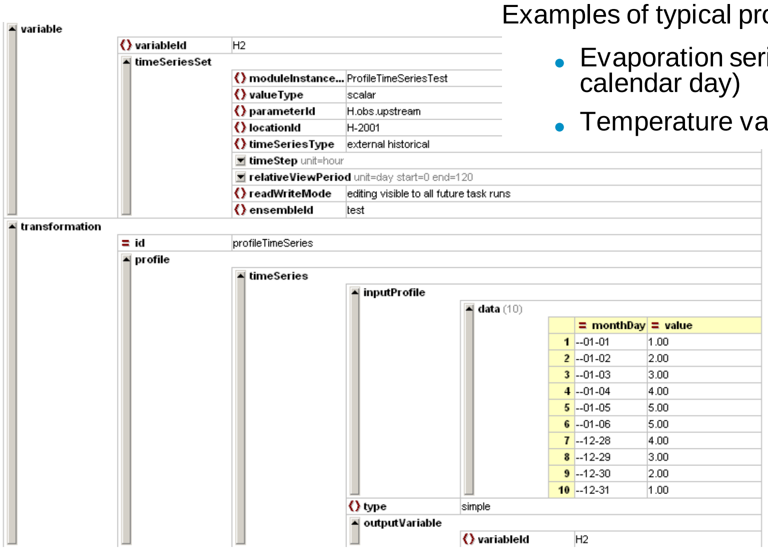


used first

used second

used third

Transformation Module – Example
Typical profile

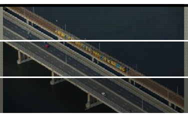


```
variable
        variableId              H2
        timeSeriesSet
                moduleInstance...   ProfileTimeSeriesTest
                valueType           scalar
                parameterId         H.obs.upstream
                locationId          H-2001
                timeSeriesType      external historical
                timeStep unit=hour
                relativeViewPeriod unit=day start=0 end=120
                readWriteMode       editing visible to all future task runs
                ensembleId          test
transformation
        id                      profileTimeSeries
        profile
                timeSeries
                        inputProfile
                                data (10)
                        type            simple
                        outputVariable
                                variableId      H2
```

	monthDay	value
1	--01-01	1.00
2	--01-02	2.00
3	--01-03	3.00
4	--01-04	4.00
5	--01-05	5.00
6	--01-06	5.00
7	--12-28	4.00
8	--12-29	3.00
9	--12-30	2.00
10	--12-31	1.00

Examples of typical profile use:

- Evaporation series (one value per calendar day)
- Temperature variation across day



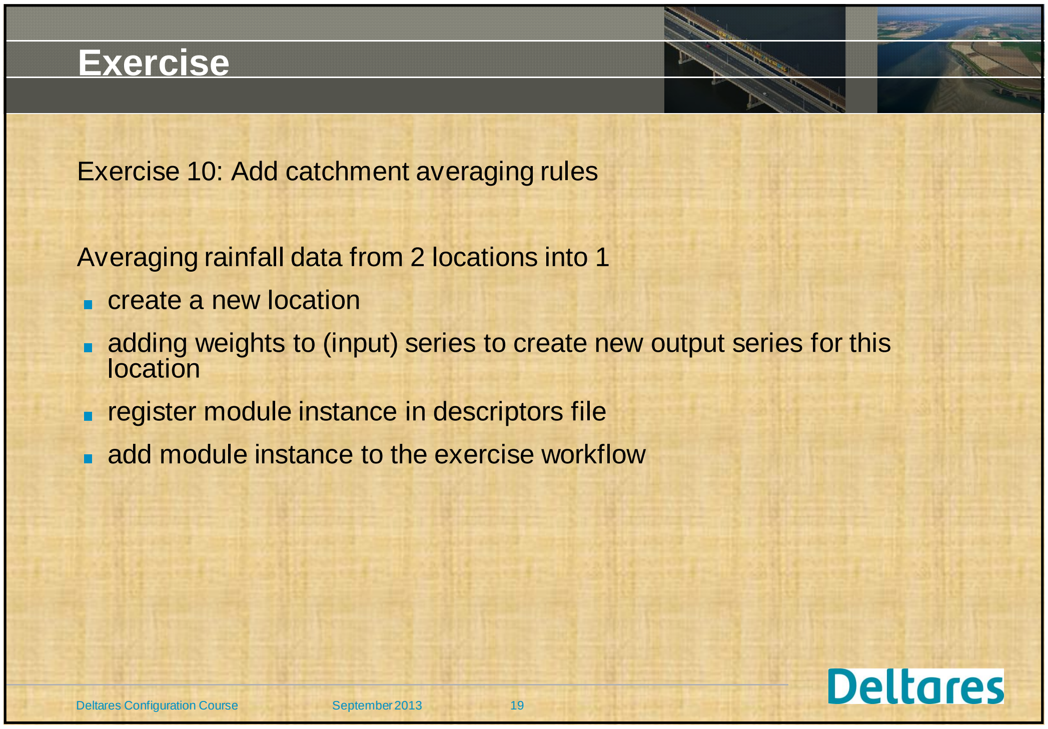
Exercise

Exercise 10: Add rating curves

Add rating curve for location Hydro1

Analyse transformation file

- register module instance in descriptors file
- add module instance to the exercise workflow

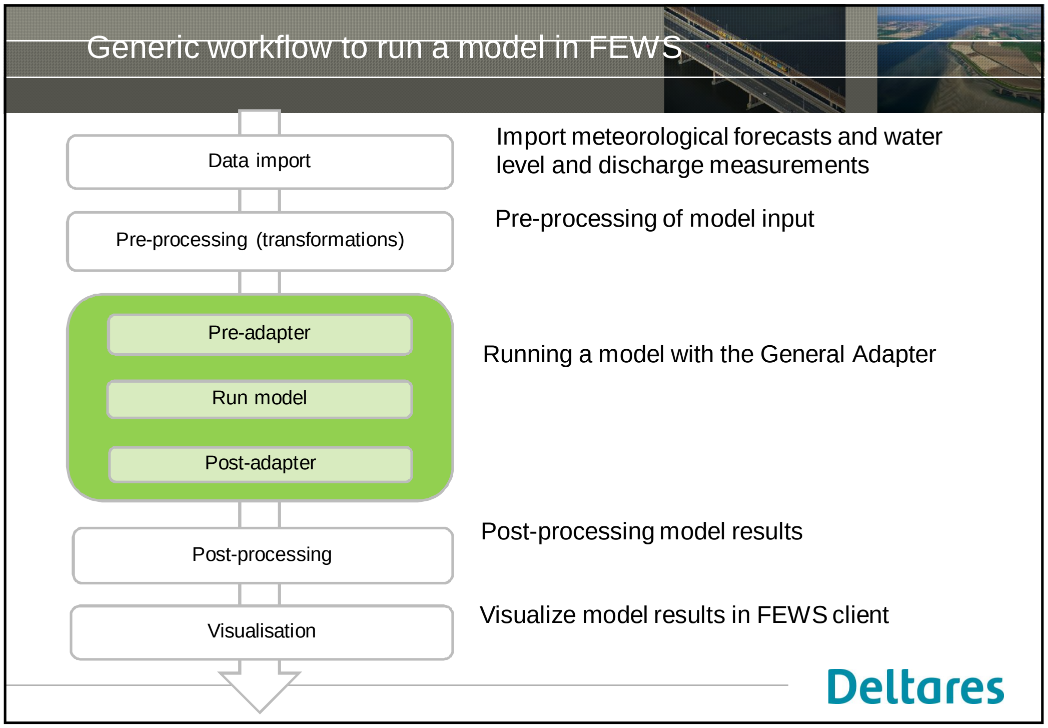


Exercise

Exercise 10: Add catchment averaging rules

Averaging rainfall data from 2 locations into 1

- create a new location
- adding weights to (input) series to create new output series for this location
- register module instance in descriptors file
- add module instance to the exercise workflow



Generic workflow to run a model in FEWS

Data import — Import meteorological forecasts and water level and discharge measurements

Pre-processing (transformations) — Pre-processing of model input

Pre-adapter
Run model
Post-adapter
— Running a model with the General Adapter

Post-processing — Post-processing model results

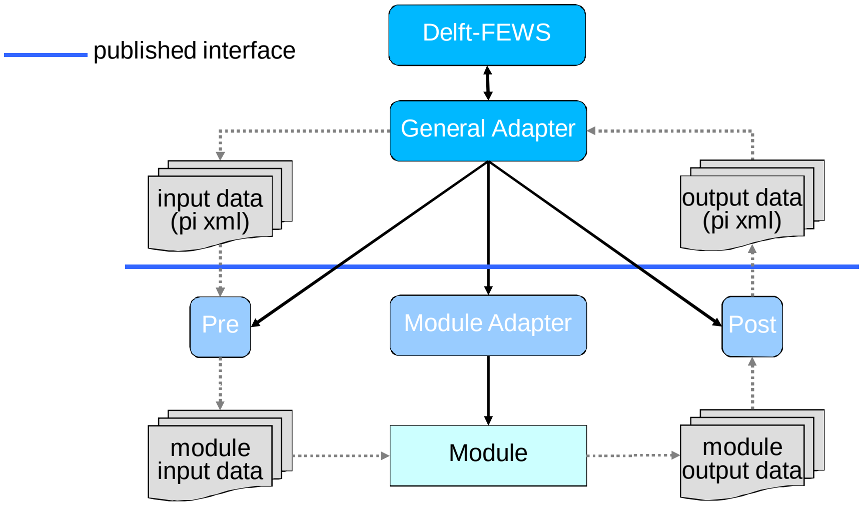
Visualisation — Visualize model results in FEWS client

Delft-FEWS Model Adapters for external models

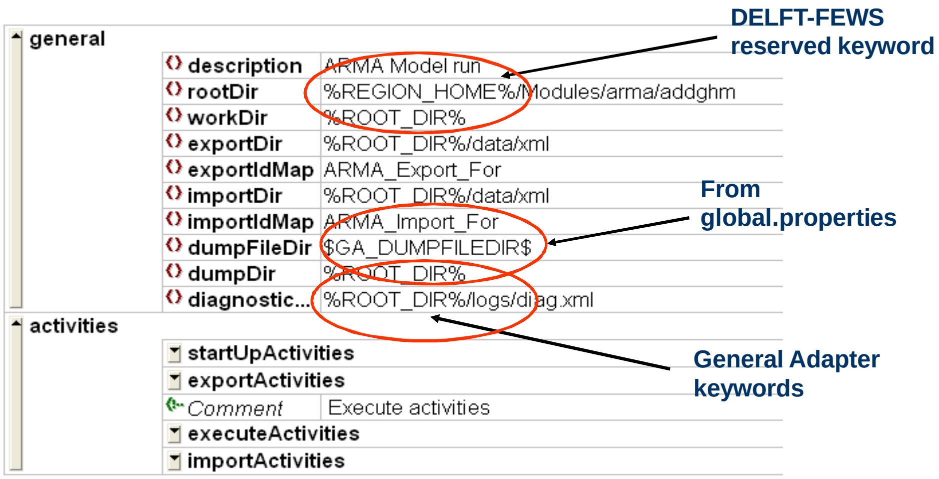
- CEH Adapters (SNOWP, SNOW, PDM, KW, ARMA, TCM, HEC, GRID2GRID)
- HR (ISIS)
- PlanB Adapters (TRITON & PRTF)
- DHI Adapters (Mike11, NAM)
- Midlands Region (DODO, MCRM)
- Southern Region (STF)
- Northwest Region (NW TF – Common Adapter)
- Wales (SW Overtopping module - Common Adapter)
- SouthWest (Bruton/Holbeam Dam module – Common Adapter)
- Deltares (RTC Tools, Delft3D, SOBEK, RIBASIM, HYMOS, Sacrament, SSARR)
- SMHI (HBV)
- University of Karlsruhe (PRMS)
- JRC (Lisflood - PCRaster)
- NWS (SNOW17, SAC-SMA, UNIT-HG, LAG/K, SARRROUTE, SSARRESV, RESSNGL, BASEFLOW, CHANLOSS, APICONT, CONSUSE, GLACIER, LAYCOEF, MUSKROUT, RSNELEV, SACSMA-HT, TATUM)
- USACE (HEC-RAS, HEC-ResSim)

http://publicwiki.deltares.nl/display/FEWSDOC/Models+linked+to+Delft-Fews

General Adapter - Concepts



General Adapter - Configuration



general

description	ARMA Model run
rootDir	%REGION_HOME%/Modules/arma/addghm
workDir	%ROOT_DIR%
exportDir	%ROOT_DIR%/data/xml
exportIdMap	ARMA_Export_For
importDir	%ROOT_DIR%/data/xml
importIdMap	ARMA_Import_For
dumpFileDir	$GA_DUMPFILEDIR$
dumpDir	%ROOT_DIR%
diagnostic...	%ROOT_DIR%/logs/diag.xml

activities

startUpActivities	
exportActivities	
Comment	Execute activities
executeActivities	
importActivities	

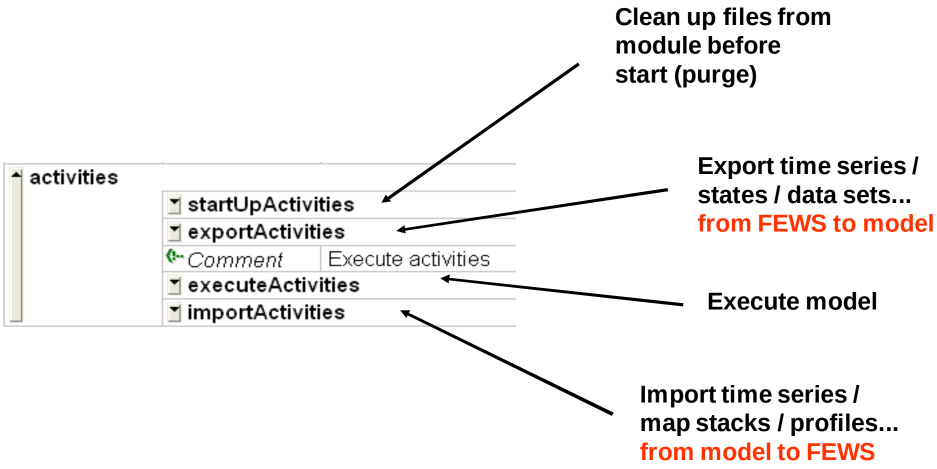
DELFT-FEWS reserved keyword

From global.properties

General Adapter keywords

Deltares

General Adapter - Configuration



activities

startUpActivities	
exportActivities	
Comment	Execute activities
executeActivities	
importActivities	

Clean up files from module before start (purge)

Export time series / states / data sets... from FEWS to model

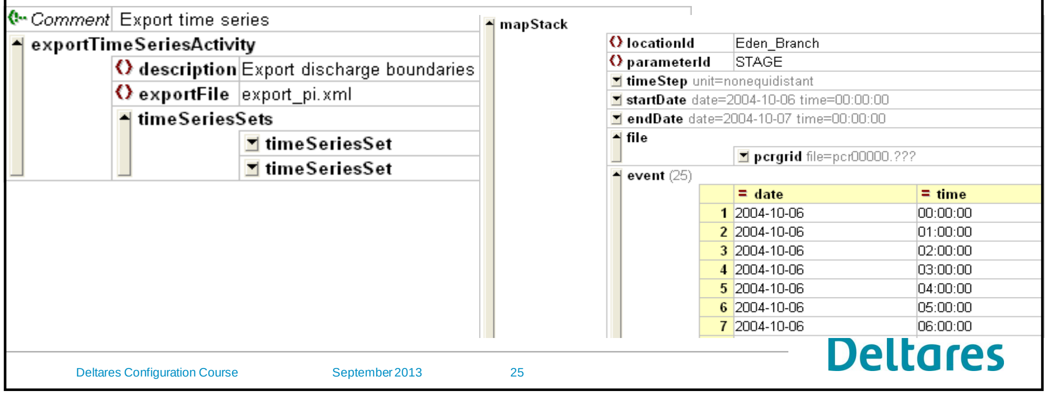
Execute model

Import time series / map stacks / profiles... from model to FEWS

Deltares

Export Activities - time series

- Exports time series from database to a PI-XML file
- Assign File name to export data to
- Assign Time Series Sets to export
- Written to exportDir
- LocationID’s & ParameterID’s will be translated on export (as indicated in IdMapping)



Comment	Export time series	
exportTimeSeriesActivity		
	description	Export discharge boundaries
	exportFile	export_pi.xml
	timeSeriesSets	
		timeSeriesSet
		timeSeriesSet

mapStack

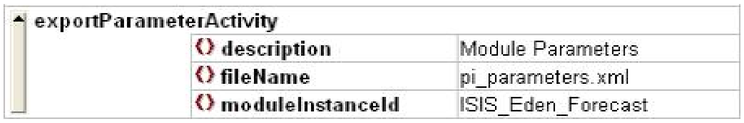
locationId	Eden_Branch
parameterId	STAGE
timeStep	unit=nonequidistant
startDate	date=2004-10-06 time=00:00:00
endDate	date=2004-10-07 time=00:00:00
file	
	pcrgrid file=pcr00000.???

event (25)

	date	time
1	2004-10-06	00:00:00
2	2004-10-06	01:00:00
3	2004-10-06	02:00:00
4	2004-10-06	03:00:00
5	2004-10-06	04:00:00
6	2004-10-06	05:00:00
7	2004-10-06	06:00:00

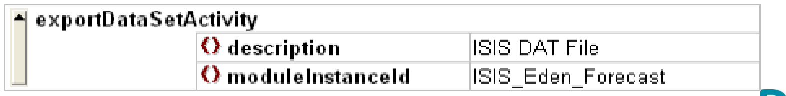
Export Activities - Module parameters / data sets

- DELFT-FEWS administers module parameters
- Exports to external module as PI-XML file
- Held in Config\ModuleParFiles



exportParameterActivity		
	description	Module Parameters
	fileName	pi_parameters.xml
	moduleInstanceId	ISIS_Eden_Forecast

- DELFT-FEWS administers module data set
  - Dataset: native module file
  - Held in Config\ModuleDataSets by DELFT-FEWS as a ZIP file
- Exports to external module by extracting data in ZIP file
- Exported to a “root”directory



exportDataSetActivity		
	description	ISIS DAT File
	moduleInstanceId	ISIS_Eden_Forecast

Export Activities - States

DELFT-FEWS can manage module states

What is a state?

- Module initial conditions
- Handled in “native”module format

Type of state;

- Cold State – default initial conditions
- Warm State – initial conditions from a previous run

For each module requiring management of states;

- ZIP file of cold state MUST be available in ColdStates dir.

Export Activities - States

Defining “how” to export state

From the perspective
of the model!!

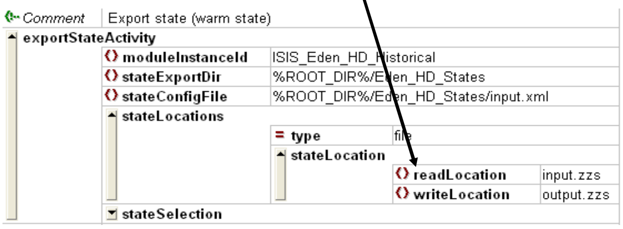
moduleInstanceId

stateExportDir

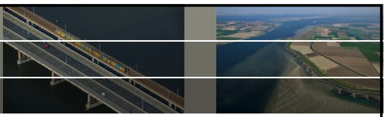
stateConfigFile

stateLocations

- read
- write



```
Comment             Export state (warm state)
exportStateActivity
                    moduleInstanceId   ISIS_Eden_HD_Historical
                    stateExportDir     %ROOT_DIR%/Eden_HD_States
                    stateConfigFile    %ROOT_DIR%/Eden_HD_States/input.xml
                    stateLocations
                                       type            file
                                       stateLocation
                                                       readLocation    input.zzs
                                                       writeLocation   output.zzs
                    stateSelection
```

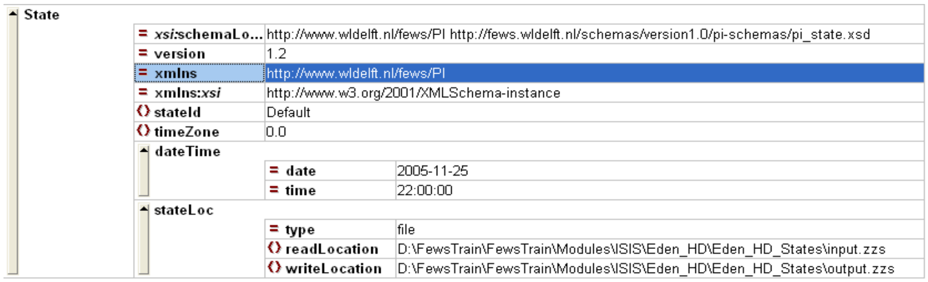


Export Activities - States

stateConfigFile

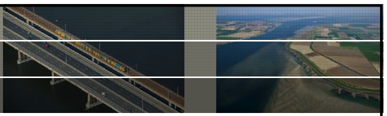
For export: Written by DELFT-FEWS ; read by adapter

For import: Written by adapter ; read by DELFT-FEWS



```
State
        = xsi:schemaLo...   http://www.wldelft.nl/fews/PI http://fews.wldelft.nl/schemas/version1.0/pi-schemas/pi_state.xsd
        = version           1.2
        = xmlns             http://www.wldelft.nl/fews/PI
        = xmlns:xsi         http://www.w3.org/2001/XMLSchema-instance
        () stateId          Default
        () timeZone         0.0
        dateTime
                            = date        2005-11-25
                            = time        22:00:00
        stateLoc
                            = type        file
                            () readLocation   D:\FewsTrain\FewsTrain\Modules\ISIS\Eden_HD\Eden_HD_States\input.zzs
                            () writeLocation  D:\FewsTrain\FewsTrain\Modules\ISIS\Eden_HD\Eden_HD_States\output.zzs
```

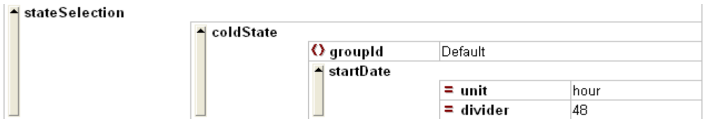
Deltares



Export Activities - States

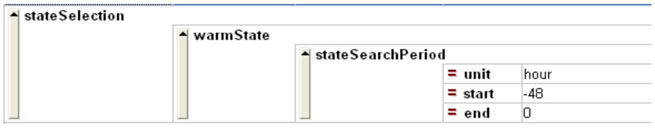
- Defining “what” to state export; stateSelection
- State selection determines length of module run!!!!!

- Cold State: Always use a cold state to initiate module run



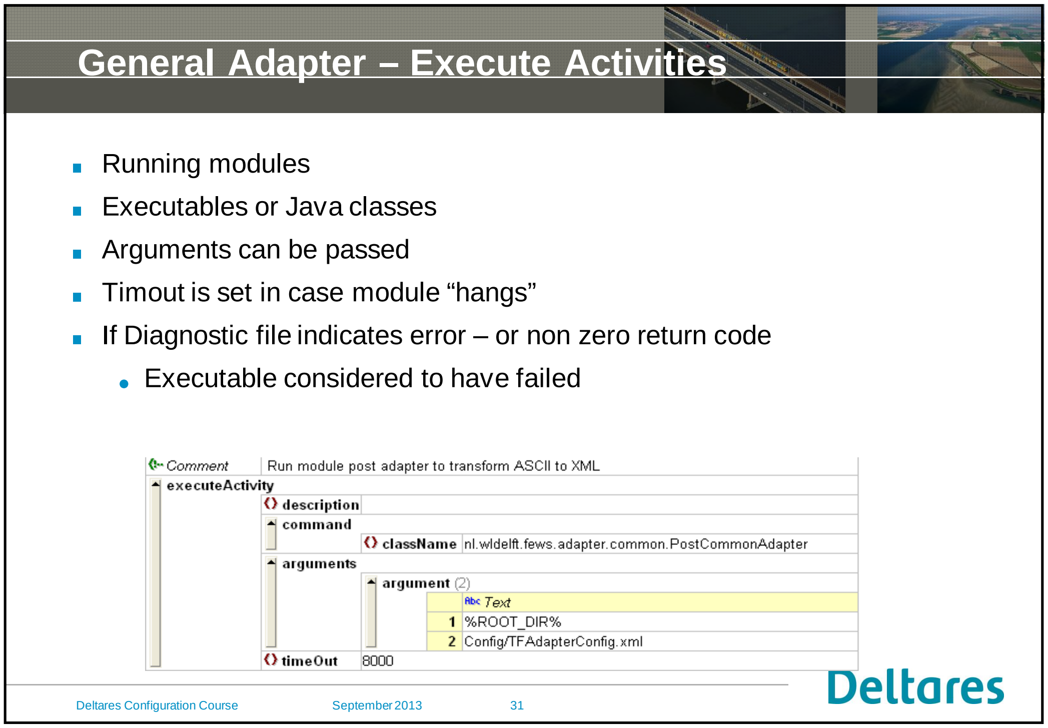
```
stateSelection
        coldState
                () groupId      Default
                startDate
                                = unit      hour
                                = divider   48
```

- Warm State: Use most suitable/recent state in search period



```
stateSelection
        warmState
                stateSearchPeriod
                                = unit      hour
                                = start     -48
                                = end       0
```

Deltares

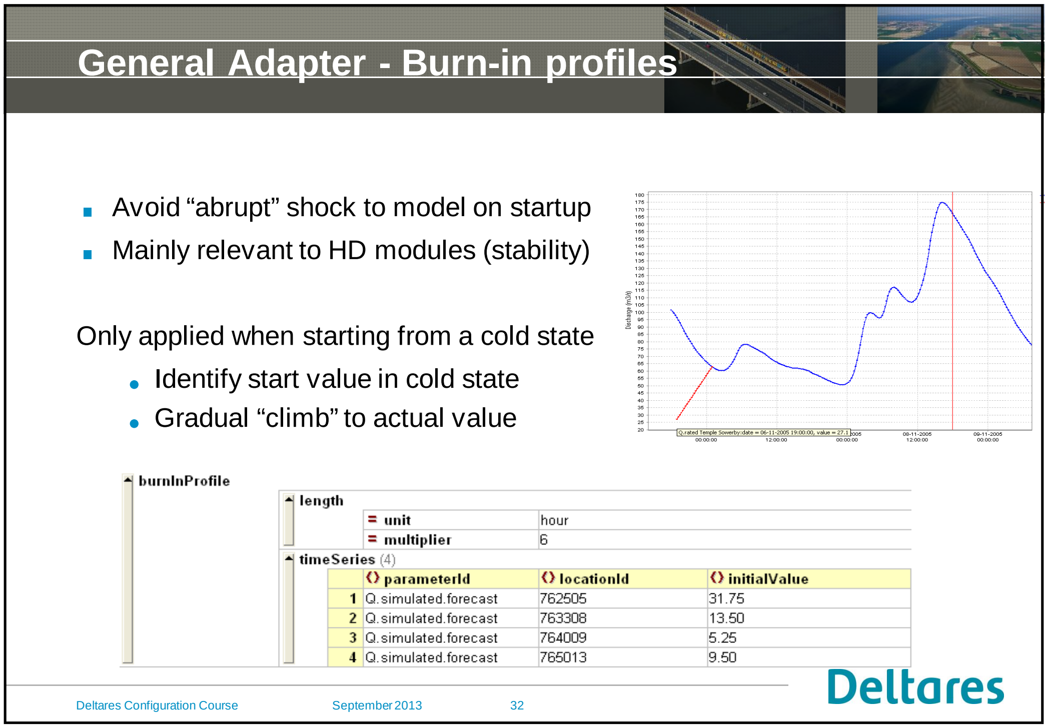


General Adapter – Execute Activities

- Running modules
- Executables or Java classes
- Arguments can be passed
- Timout is set in case module “hangs”
- If Diagnostic file indicates error – or non zero return code
  - Executable considered to have failed

Comment	Run module post adapter to transform ASCII to XML			
executeActivity				
	description			
	command			
		className	nl.wldelft.fews.adapter.common.PostCommonAdapter	
	arguments			
		argument (2)		
				Text
			1	%ROOT_DIR%
			2	Config/TFAdapterConfig.xml
	timeOut	8000		

Deltares



General Adapter - Burn-in profiles

- Avoid “abrupt” shock to model on startup
- Mainly relevant to HD modules (stability)

Only applied when starting from a cold state

- Identify start value in cold state
- Gradual “climb” to actual value

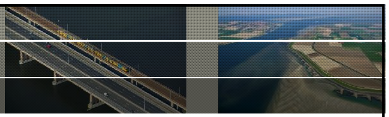
burnInProfile

length		
unit	hour	
multiplier	6	

timeSeries (4)

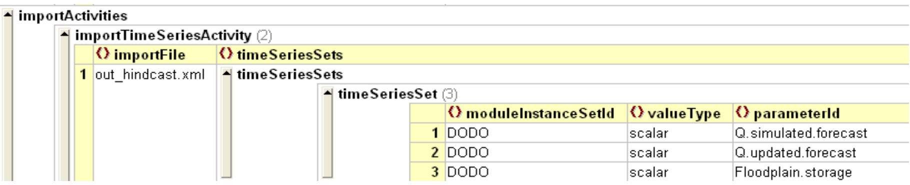
	parameterId	locationId	initialValue
1	Q.simulated.forecast	762505	31.75
2	Q.simulated.forecast	763308	13.50
3	Q.simulated.forecast	764009	5.25
4	Q.simulated.forecast	765013	9.50

Deltares



Import Activities

- Importing time series / states
- Same principle as exporting
- File name & Time Series Set to save data to
- IdMapping used to translate locationId’s and parameterId’s
- Grids imported as mapstack + XML file



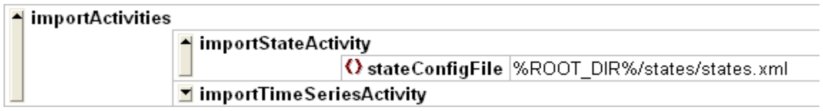
importActivities

importTimeSeriesActivity (2)

	importFile	timeSeriesSets
1	out_hindcast.xml	timeSeriesSets

timeSeriesSet (3)

	moduleInstanceSetId	valueType	parameterId
1	DODO	scalar	Q.simulated.forecast
2	DODO	scalar	Q.updated.forecast
3	DODO	scalar	Floodplain.storage

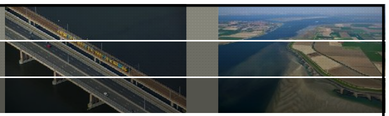


importActivities

importStateActivity

stateConfigFile %ROOT_DIR%/states/states.xml

importTimeSeriesActivity



Module diagnostics/debugging

PI-XML includes module diagnostics exchange

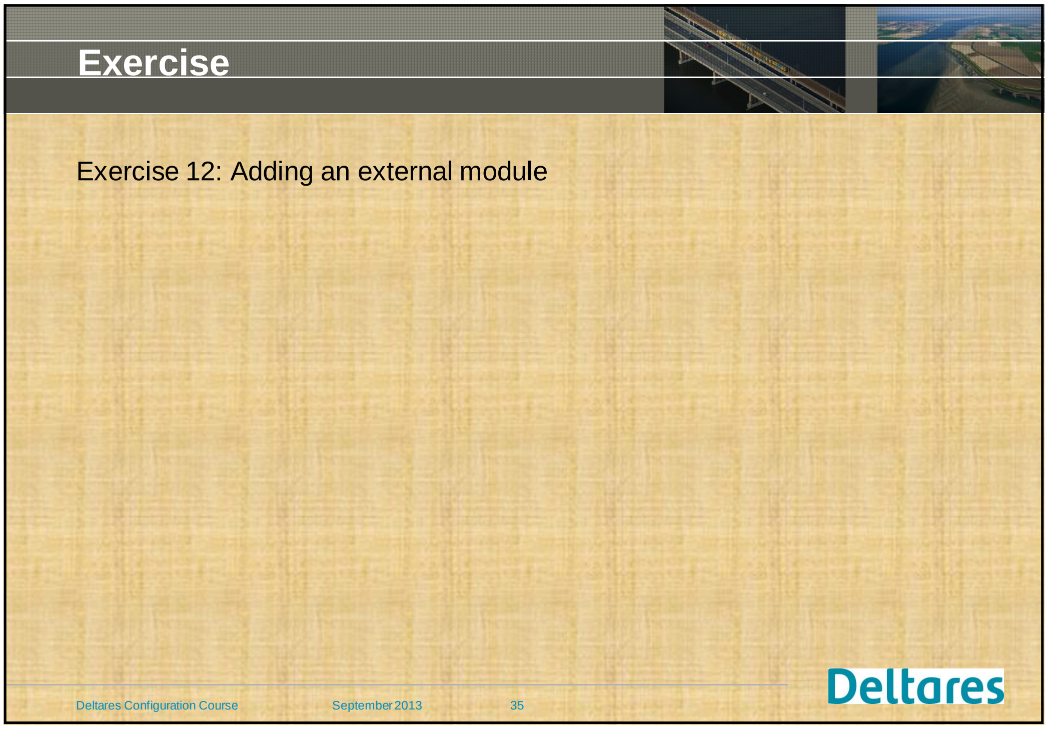
- To be written by module/adapter
- General Adapter Defines where diagnostics file is expected
- Not all logging passed – use native module log files

External module log inserted into log

- Most messages DEBUG level
- Error/Fatal: System messages

Detecting module failure

- Error in diagnostics log
- non-zero return code



Exercise

Exercise 12: Adding an external module