06 Workflow Configuration

What	*A workflow*.xml
Required	no
Description	Definition of sequence of moduleInstances (or workflows) in logical order
schema location	https://fewsdocs.deltares.nl/schemas/version1.0/workflow.xsd

Introduction

Workflows are used in Delft-FEWS to define logical sequences of running forecast modules. The workflow itself simply defines the sequence with which the configured modules are to be run. There is no inherent information nor constraints within the workflow on the role the module has in delivering the forecasting requirement.

Workflows may contain calls to configured module instances, but may also contain calls to other workflows. In the workflowDescriptors configuration described in the Regional Configuration section, the properties of the workflows is defined.

All workflows are defined in the Workflows section of the configuration; when working from a filesystem this is the WorkflowFiles directory. Each workflow will have the same structure and must adhere to the same XML schema definition. Workflows are identified by their name, which are registered to the system through the workflowDescriptors configuration in the Regional Configuration section.

Workflows

Workfows defined may either be available from the Workflows table – when the configuration is loaded into the database – or available in the WorkflowFiles directory when the configuration is available on the file system.

When available on the file system, the name of the XML file for a workflow (e.g ImportExternal.xml) needs to be registered in the WorkflowDescriptors with Id ImportExternal.

Each processing step in the workflow is referred to as an activity. In defining workflows, several levels of complexity in defining these activities are available;

- Simple activities
- Activities with a fallback activity;
- Activities to be run as an ensemble.
- Parallel activities, i.e. a group of activities that can be executed in parallel when multiple cores are available

- Sequence activities, i.e. a group of activities that have to be executed in sequence

Activities

Each activity can either be a single moduleInstance or another workflow.

A single moduleInstance can have a one-to-one relation to a module config file (i.e. file name corresponds to moduleInstanceId) or it use a unique moduleInstanceId while referencing a module config file name which is shared as a template by various module instances. In the latter case, properties can be defined in the workflow to make the timeSeriesSets used within each moduleInstanceId unique. A property has a key and a value. When a module config file holds a property-key (key) the property-value, as defined in e.g. the workflow, will replace the key when the configuration file is read.

As can be seen in the following images, properties can be defined at varies levels in a workflow. In general the property definition at the lowest level overrules a property defined at a higher level. E.g. a property defined at the module instance level overrules a property defined at the (sub)workflow level overrules a property at the global level (i.e. global.properties file).

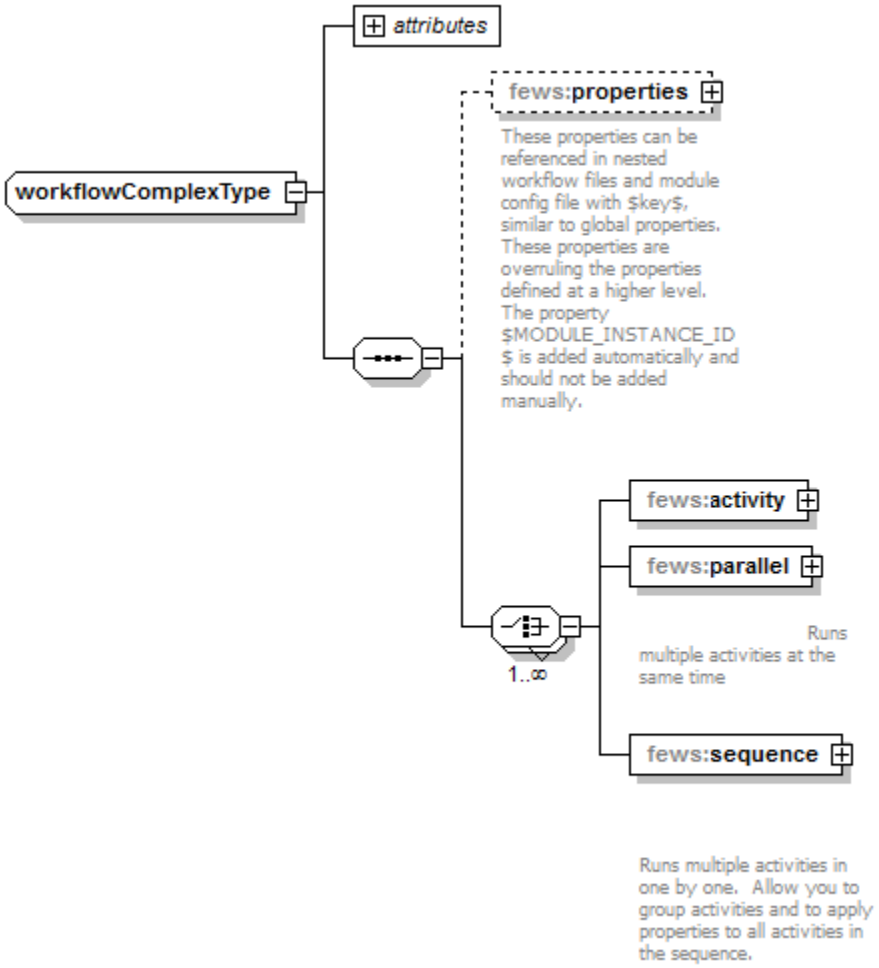


Figure 142 Elements of the Workflow configuration.

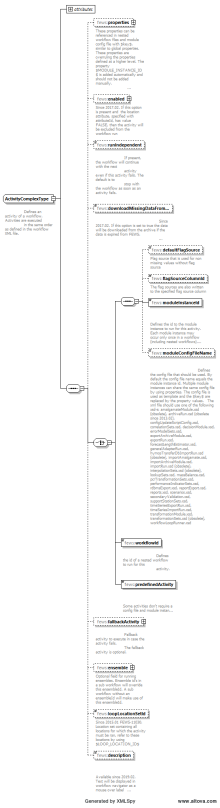


Figure 143 Elements of an Activity in the workflow.

activity

Root element for the definition of a workflow activity. Multiple entries can be defined.

runIndependent

Boolean flag to indicate if the activity is considered to be independent of other activities in the workflows. If the flag is set to "false" (default) then the failure of this activity will cause the complete workflow being considered as having failed. No further activities will be carried out. If the flag is set to "true", then failure of an activity will not cause the workflow to fail. The next activity in the workflow will also be attempted. An indication is given to the user in the Forecast Management display if one or more workflow activities have failed.

enabled

The <enabled> element (since 2017.02) can be used to in- or exclude an activity in a workflow run. This feature is based on a location attribute. If you configure a modifier for this, the forecaster can in- or exclude activities (modules or sub-workflows) from a workflow run from the OC.

The element contains 2 attributes:

- locationId
- attributeId, which is a BOOLEAN attribute for that location (i.e. TRUE or FALSE)
    - when TRUE: the activity will be included,
    - when FALSE: the activity will not be included.

Example below, for a full How To see: How To: configure <enabled> in Workflow based on location attributes

<enabled> element in workflow

```
        <activity>
                <enabled locationId="CAP" attributeId="EXPORT"/>
                <moduleInstanceId>Export_CAP</moduleInstanceId>
                <description>Export CAP messages</description>
        </activity>
</workflow>
```

enable

The <enable> element (since 2024.01) can be used in- or exclude an activity in a workflow, similar to the <enabled> element. However, this element is not coupled to location attributes, instead a boolean value is used, which can be hardcoded as in the example below or defined through a global property in the file global.properties.

```
        <activity>
        <enable>true</enable>
                <moduleInstanceId>Run_Model</moduleInstanceId>
        </activity>
```

Description

Optional field. If configured, the text will be displayed in the workflow navigator tree as mouse over label (tooltip).

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<workflow 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/workflow.xsd"
version="1.1">
   <activity>
      <runIndependent>true</runIndependent>
      <moduleInstanceId>CanadaMeteo</moduleInstanceId>
      <description>Description</description>
   </activity>
</workflow>
```

downloadMissingDataFromArchive

Boolean flag to indicate whether data needs to be retrieved from the archive when not available/expired from the operational Delft-FEWS database. This download can only be conducted when no idmapping is needed to obtain a match between the timeseries header used to store the data in the archive and the timeseries header requested by the processing module.

Functionality is available since 2017.02 for external historical data. Only works in combination with an archive connection as configured in SystemConfigFiles/Archives.xml

defaultFlagSource

Optional element; Flag source that is used to write to the flagSourceColumn in an activity that is able to write a flag. Used for non missing values when no other flagSource is applied. The defaultFlagSource will be written in case no flag is changed and data is non missing. Typically this defaultFlagSource is set to be 'OK', and configured in CustomFlagSources configuration file.

flagSourceColumnId

Optional element; This will dictate which flagSourceColumn that will be used to write the flagSources to. A validation step where a flagSourceColumnId has been defined will write the appropriate flagSource in case of a flag change, both to the flagSource column and the regular flagSource element. In case the flag is not changed, the defaultflagSource will be written to the flagSource column. In this case, the flagSource will only be written to the regular flagSource element when this is empty. The flagSourceColumnId needs to be defined in flagSourceColumn.xml configuration file .

moduleInstanceId

The ID of the moduleInstance to run as the workflow activity. This module instance must be defined in the moduleInstanceDescriptors (see Regional Configuration) and a suitable module configuration file must be available (see Module configurations). This moduleInsatnceId is used to administer the data generated.

moduleConfigFileName

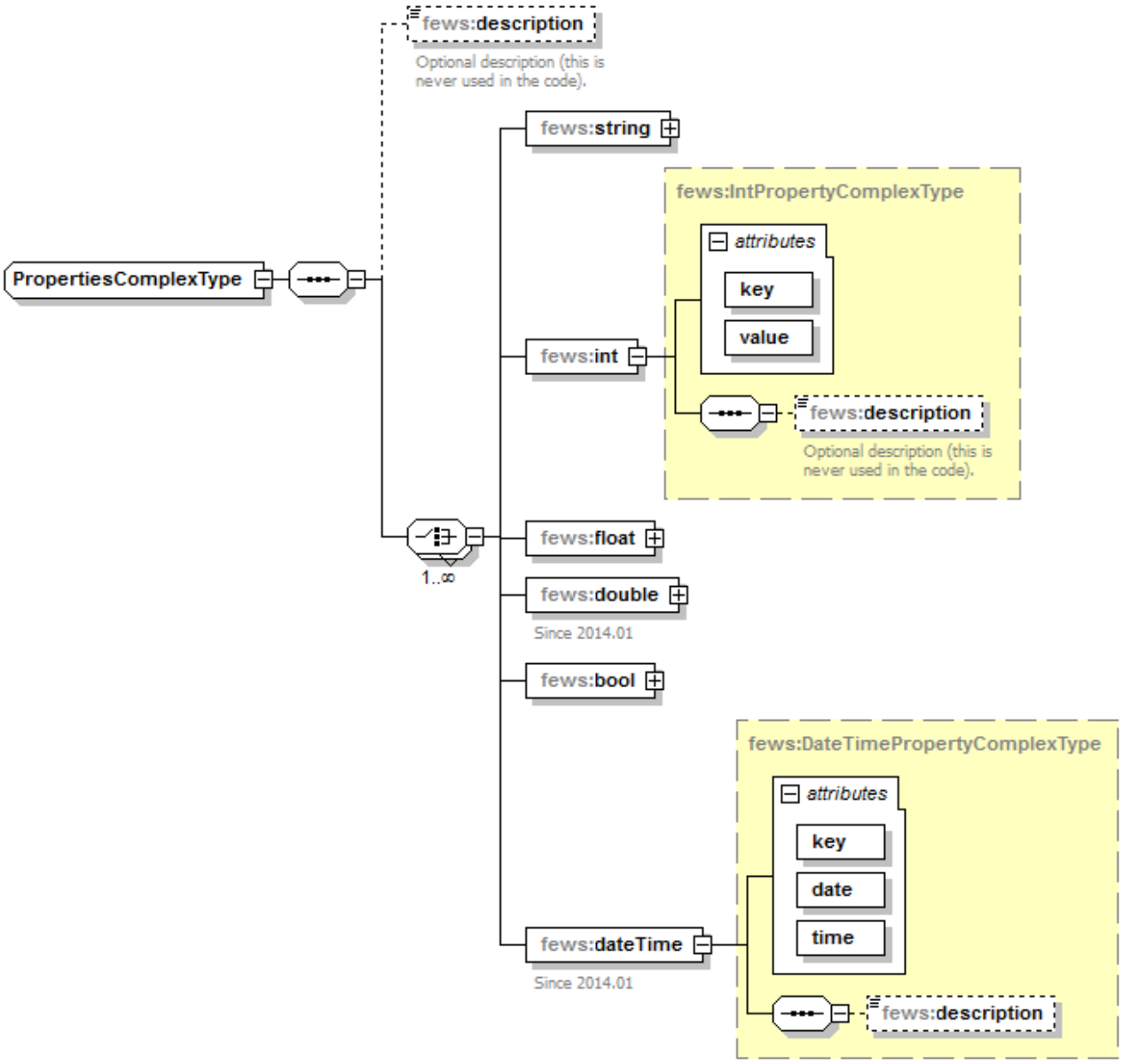
Optional reference to a module config file that holds the processing logic for a module instance. Typically this module config file holds property-keys such that it can acts as template for multiple module instances, each module instacne having a unique set of property-values.

workflowId

This is the name of the subworkflow xml file and NOT the workflow descriptor id as the name suggests. The workflow descriptors (including the workflow descriptor properties) are ignored when resolving the subworkflow. There is no need to add a subworkflow xml file to the workflow descriptors when it is only used as a subworkflow. In case a workflow is defined in de the workflowDescriptors by using a template workflow config file (in combination with properties), you cannot refer to such a workflowId, but should use the template file as workflowId and use properties.

properties

Root element for the definition of one or more properties. Multiple entries can be defined. Properties can be externalized portions of a timeSeriesSet definitions (e.g. locationSetId) or parameter values from the module configuration file to the level of the workflow definition.



properties:string

property definition for a string data type. Holds a key-attribute and a value-attribute, where the attribute value is replacing the key as defined in a module config file.

properties:int

property definition for a integer data type. Holds a key-attribute and a value-attribute, where the attribute value is replacing the key as defined in a module config file.

properties: float

property definition for a float data type. Holds a key-attribute and a value-attribute, where the attribute value is replacing the key as defined in a module config file.

properties:double

property definition for a double data type. Holds a key-attribute and a value-attribute, where the attribute value is replacing the key as defined in a module config file.

properties:bool

property definition for a boolean data type. Holds a key-attribute and a value-attribute, where the attribute value is replacing the key as defined in a module config file.

properties:dateTime

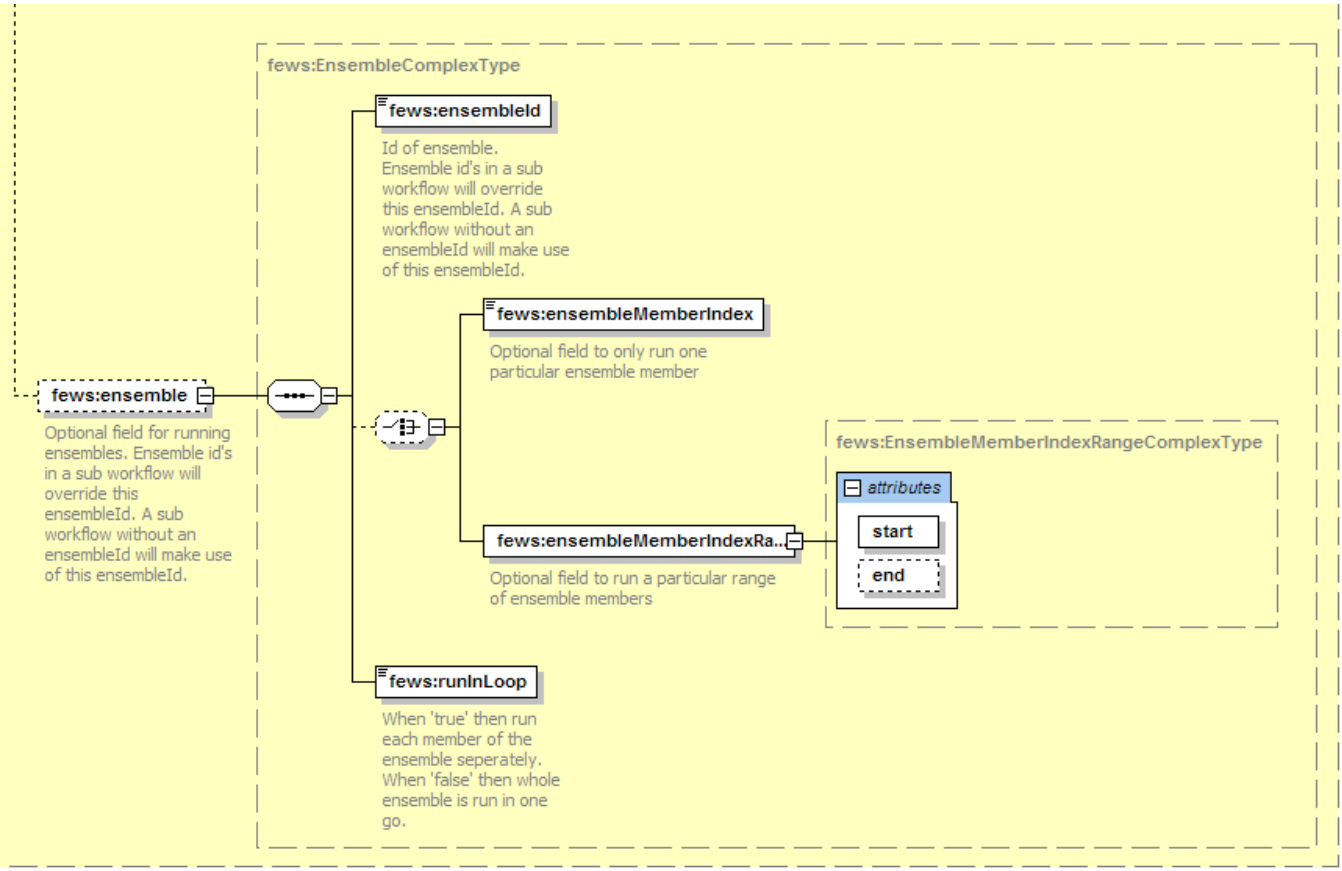
property definition for a dateTime data type. Holds a key-attribute while the value is based on the combination of the date-attribute and the time-attribute.

fallbackActivity

A fallback activity may be defined to run if the activity under which it is defined fails. This can be used to run a simple module if the more complex method used in preference fails, and ensures continuity of the forecasting process. The definition of the fallbackActivity is the same as the definition of an activity. A fallback activity cannot be used to turn a partly completed (completed with errors) TaskRun into a fully successful TaskRun (complete without errors).

ensemble

This element is defined to indicate the activity is to run as an ensemble.



ensembleId

Id of the ensemble to apply in retrieving data from the database. For all time series sets used as input for the activities running as an ensemble, a request for time series with this Id defined will be made. Ensemble id's in a sub workflow will override this ensembleId. A sub workflow without an ensembleId will make use of this ensembleId

runInLoop

Boolean flag to indicate if the activity should run as many times as there are members in the ensemble, or if it is to be run only once, but will use all members of the ensemble in that single run. If the value is set to "true", then when running the workflow DELFT-FEWS will first establish how many members there are in the ensemble and then run the activity for each member. If the value is set to "false" then the activity will be run only once. On requesting a time series set within the modules to be run, the database will return all members of that ensemble. When true the $LOOP_ENSEMBLE_MEMBER_ID$ can be used in the module config files. This property is not needed for the time series sets because the loop member is the default ensemble selection for a time series set while runInloop is enabled.

ensembleMemberId

Since 2018.01 (for older versions, configure an ensembleMemberIdRegularExpression with the wanted ensembleMemberId as a "pattern"). Optional field to only run one particular ensemble member. If this field is used only the specified ensemble member will be run.

ensembleMemberIndex

Optional field to only run one particular ensemble member. If this field is used only the specified ensemble member will be run.

ensembleMemberIndexRange

Optional field to run a particular range of ensemble members. Processing starts at member *start* and ends at member *end*. If *end* is not specified the processing will start at *start* and end at the last member of the ensemble.

ensembleMemberIdRegularExpression

Runs all members which id matches the specified expression e.g specify ^Q.* to loop over all members that start with Q.

Activities in a workflow or nested workflows may be called only once. This is to avoid mistakes through multiple calls to the same activity in different locations thus creating ambiguous results, or circular references in defining fallback activities.

When running activities as an ensemble that request time series sets from the database that are not a part of that ensemble, the default ensembleId should be added to the TimeSeriesSets definition. The default ensemble Id is "main".

All time series sets written when running in ensemble mode will have the ensembleId as specified in the workflow ensembleId element, unless overruled by a local ensembleId defined in the timeSeriesSet on writing.

loopLocationSet

Like looping over an ensemble it is possible to loop over and location set. $LOOP_LOCATION_ID$ can be used inside the module configuration to get the active location id. When the configured loopLocationSet does not exist a config error is thrown. With skipNonExistingLoopLocationSet the workflow activity can be skipped is this case, preventing the error.

workflow:parallel execution

Grouping to accommodate parallelization of (a portion) of the workflow when multiple CPU available. As explained on 19 Parallel running of ensemble loops and activities on one forecasting shell instance, one can conduct parallelization on one forecasting shell instance by defining the global.property `runInLoopParallelProcessorCount`. Since FEWS 2017.01 it is also possible to run the parallel activities on multiple forecasting shells. Parallel activities should only be configured when the underlying activities have no data dependency. If data dependency exists, activities should be executed in sequence. This may be forced by embedding them as part of a sequence group.

Two flavours can be accommodated:

- parallel - multipleForecastingShells=true: use this setting to conduct parallel execution of multiple activities. This settings requires specification of element maxEnsembleParts in he WorkflowDescriptors
- parallel - forecastingShellCount: use this setting to conduct parallel execution of any underlying loop (ensemble loop or location loop) within an activity

When using multipleForecastingShells=true, it is recommended not to include more (sequential) activities in the parallel section than the number of forecasting shell instances available. Note that all taskrun partitions receive the same scheduledDispatchedTime. In order to be run successfully, they need to be dispatched within amount of seconds specified by the fews.master.mc.conf launcher maxlatetime, otherwise the taskruns may become terminated because they are overdue. The current implementation of *multipleForecastingShells=true* does not come with intelligence that will take into account the *runIndepent=true* flag. If one of the partitioned taskRuns is completely failed or is terminated, the other partitions that are not completed yet will be terminated.

When using forecastingShellCount = n every underlying loop is split into n parts. Every FS instance is running the same specified activity but skipping the part of the ensemble loop / location loop that is meant for an other FS instance. It is not allowed to start a nested parallel when you are in a ensemble /location loop parallel.

reschedule

Before 2019.02 if an existing scheduled workflow is redefined as a parallel workflow, it needs to be rescheduled to prevent errors. In the worst case situation the MC may become unavailable.

2017.02 Improvement

In 2017.02 fss patch #92224 provides an improvement / bugfix that only applies to *the multipleForecastingShells=true* option in 2017.02. With this improvement, a partly successful forecast partition in 2017.02 should not cause termination of the other partitions.

workflow:sequence

Grouping to enforce sequential execution of multiple activities as they generally have data dependencies. Typically used in workflows where parallel activities are defined to clarify which activities have be to executed in sequence.

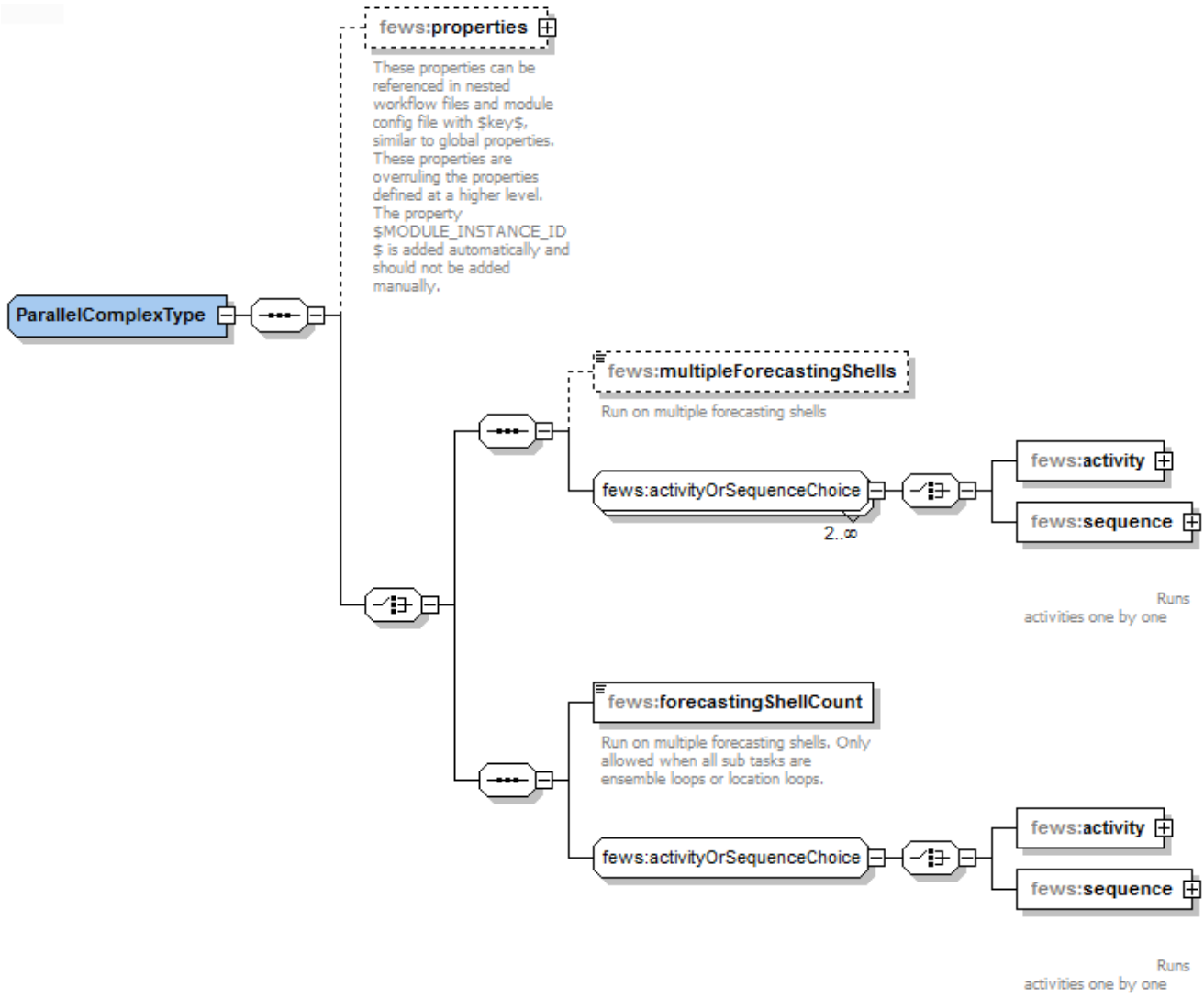


Figure 143 Elements of an Parallel Activity in the workflow.

Examples

The workflow below runs seven moduleInstances. If the first moduleInstance fails in this example all other processing is stopped. If any of the other activities fail the processing will continue.

```
<workflow version="1.1" 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
/workflow.xsd">
        <activity>
                <runIndependent>false</runIndependent>
                <moduleInstanceId>Astronomical</moduleInstanceId>
        </activity>
        <activity>
                <runIndependent>true</runIndependent>
                <moduleInstanceId>BackupPrecipitation_Forecast</moduleInstanceId>
        </activity>
                <runIndependent>true</runIndependent>
                <moduleInstanceId>PrecipitationGaugeToGrid_Forecast</moduleInstanceId>
        </activity>
        <activity>
                <runIndependent>true</runIndependent>
                <moduleInstanceId>Spatial_Interpolation_Precipitation_Forecast</moduleInstanceId>
        </activity>
        <activity>
                <runIndependent>true</runIndependent>
                <moduleInstanceId>MergePrecipitation_Forecast</moduleInstanceId>
        </activity>
        <activity>
                <runIndependent>true</runIndependent>
                <moduleInstanceId>GridToCatchments_Forecast</moduleInstanceId>
        </activity>
        <activity>
                <runIndependent>true</runIndependent>
                <moduleInstanceId>Singapore_Sobek_Forecast</moduleInstanceId>
        </activity>
</workflow>
```

The example below is more complex and includes several modules that are run in ensemble mode.

```
<workflow 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/workflow.xsd"
version="1.1">
        <!--Run Rhein Interpolation -->
        <activity>
                <runIndependent>true</runIndependent>
                <workflowId>Rhein_Interpolate</workflowId>
        </activity>
        <!--Spatial interpolation from grid to HBV-centroids-->
        <activity>
                <runIndependent>true</runIndependent>
                <moduleInstanceId>Rhein_SpatialInterpolationCOSMO-LEPS</moduleInstanceId>
                <ensemble>
                        <ensembleId>COSMO-LEPS</ensembleId>
                        <runInLoop>true</runInLoop>
                </ensemble>
        </activity>
        <!--Aggregate forecast data for display -->
        <activity>
                <runIndependent>true</runIndependent>
                <moduleInstanceId>Rhein_AggregateForecast_COSMO-LEPS</moduleInstanceId>
                <ensemble>
                        <ensembleId>COSMO-LEPS</ensembleId>
                        <runInLoop>true</runInLoop>
                </ensemble>
        </activity>
        <!--Disaggregate timeseries at HBV-centroids -->
        <activity>
                <runIndependent>true</runIndependent>
                <moduleInstanceId>Rhein_DisaggregateSeriesCOSMO-LEPS</moduleInstanceId>
                <ensemble>
                        <ensembleId>COSMO-LEPS</ensembleId>
                        <runInLoop>true</runInLoop>
                </ensemble>
```

```
        </activity>
        <!--Merge timeseries from historical run and forecast run -->
        <activity>
                <runIndependent>true</runIndependent>
                <moduleInstanceId>HBV_Rhein_Merge_COSMO-LEPS</moduleInstanceId>
                <ensemble>
                        <ensembleId>COSMO-LEPS</ensembleId>
                        <runInLoop>true</runInLoop>
                </ensemble>
        </activity>
        <!--Aggregate inputs for display -->
        <activity>
                <runIndependent>true</runIndependent>
                <moduleInstanceId>HBV_Rhein_AggregateInputs_COSMO-LEPS</moduleInstanceId>
                <ensemble>
                        <ensembleId>COSMO-LEPS</ensembleId>
                        <runInLoop>true</runInLoop>
                </ensemble>
        </activity>
        <!--Interpolate timeseries from historical run and forecast run -->
        <activity>
                <runIndependent>true</runIndependent>
                <moduleInstanceId>HBV_Rhein_Interpolate_COSMO-LEPS</moduleInstanceId>
                <ensemble>
                        <ensembleId>COSMO-LEPS</ensembleId>
                        <runInLoop>true</runInLoop>
                </ensemble>
        </activity>
        <!--Run HBV-model for forecast period-->
        <activity>
                <runIndependent>true</runIndependent>
                <moduleInstanceId>HBV_Rhein_COSMO-LEPS</moduleInstanceId>
                <ensemble>
                        <ensembleId>COSMO-LEPS</ensembleId>
                        <runInLoop>true</runInLoop>
                </ensemble>
        </activity>
                <!--Run ErrorModule for forecast period-->
        <activity>
                <runIndependent>true</runIndependent>
                <moduleInstanceId>HBV_Rhein_AR_COSMO-LEPS</moduleInstanceId>
                <ensemble>
                        <ensembleId>COSMO-LEPS</ensembleId>
                        <runInLoop>true</runInLoop>
                </ensemble>
        </activity>
        <!--Calculate Statistics-->
        <activity>
                <runIndependent>true</runIndependent>
                <workflowId>Statistics_COSMO-LEPS</workflowId>
        </activity>
        <!--Export forecast data to wavos format -->
        <activity>
                <runIndependent>true</runIndependent>
                <moduleInstanceId>Rhein_ExportForecast_COSMO-LEPS</moduleInstanceId>
        </activity>
</workflow>
```

The example below is complex in a different way as it uses properties to hand over catchment specific information to a report.

Usage of properties to add catchment specific information to a web-report

```
<workflow version="1.1" 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
/workflow.xsd">
    <!--Create Calibration reports-->
    <properties>
        <string key="CATCHMENT" value="armidale"/>
        <string key="CATCHMENT_NAME" value="Armidale"/>
        <string key="BASIN" value="macleay"/>
        <string key="STATE" value="NSW"/>
        <string key="TIMEZONE" value="AET"/>
        <string key="ENSEMBLE" value="OFFICIAL"/>
        <string key="ENSMEMBER" value="URBS Final"/>
    </properties>
    <!--Compute -sub-catchment averaged data as sub-catchment data is not archived-->
    <activity>
        <runIndependent>true</runIndependent>
        <moduleInstanceId>Processing_15m_Site_Calibration</moduleInstanceId>
    </activity>
    <activity>
        <runIndependent>true</runIndependent>
        <moduleInstanceId>Performance_15m_Calibration</moduleInstanceId>
    </activity>
    <activity>
        <runIndependent>true</runIndependent>
        <moduleInstanceId>Report_15m_Site_Calibration</moduleInstanceId>
    </activity>
     <activity>
        <runIndependent>true</runIndependent>
        <moduleInstanceId>Report_Catchment_Calibration</moduleInstanceId>
    </activity>
</workflow>
```

The following workflow example uses properties to make a apply the logic, encapsulated in general module config file, to a specific catchment. The second code block shows how some of the properties are used to make timeseriessets in the Rainfall_15m_Forecast module config file unique for the locations of a certain catchment.

Workflow defining property-values to apply general processing logic to a specific catchment

```
<workflow 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/workflow.xsd"
version="1.1">
    <properties>
        <string key="RADAR" value="noradar"/>
        <string key="CATCHMENT" value="armidale"/>
        <string key="BASIN" value="macleay"/>
        <string key="ENSEMBLE" value="IFD"/>
        <string key="ENSMEMBER" value="URBS Local"/>
    </properties>
    <activity>
        <runIndependent>true</runIndependent>
        <moduleInstanceId>armidale_Rainfall_Forecast</moduleInstanceId>
        <moduleConfigFileName>Rainfall_15m_Forecast</moduleConfigFileName>
    </activity>
    <activity>
        <runIndependent>true</runIndependent>
        <workflowId>armidale_URBS_Catchment_Forecast</workflowId>
        <ensemble>
            <ensembleId>IFD</ensembleId>
            <runInLoop>true</runInLoop>
        </ensemble>
    </activity>
</workflow>
```

Property-keys to allow catchment specific timeseries definitions in the Rainfall_15m_Forecast module config file

```
    <variable>
        <variableId>ACCESS_G_subarea_3h</variableId>
        <timeSeriesSet>
            <moduleInstanceId>$CATCHMENT$_Rainfall_Forecast</moduleInstanceId>
            <valueType>scalar</valueType>
            <parameterId>P.nwp.fcst</parameterId>
            <qualifierId>ACCESS_G</qualifierId>
            <locationSetId>URBS_subareas.$CATCHMENT$</locationSetId>
            <timeSeriesType>temporary</timeSeriesType>
            <timeStep unit="hour" multiplier="3"/>
            <readWriteMode>read complete forecast</readWriteMode>
        </timeSeriesSet>
    </variable>
```

Since 2015.01 it is possible to use these property keys in all elements of the timeSeriesSet including e.g. $TIME_SERIES_TYPE$ and $START_DAYS$

Additional property-keys

```
    <variable>
        <variableId>Variable</variableId>
                        <timeSeriesSet>
                                <moduleInstanceId>import</moduleInstanceId>
                                <valueType>scalar</valueType
                                <parameterId>H.obs</parameterId>
                                <locationId>H-2001</locationId>
                                <timeSeriesType>$TIME_SERIES_TYPE$</timeSeriesType>
                                <timeStep unit="day"/> <relativeViewPeriod unit="day" start="$START_DAYS$" end="
1"/>
                                <readWriteMode>read only</readWriteMode>
                        </timeSeriesSet>
    </variable>
```

Example of a sub-workflow with parallel activities using properties defined at a higher level workflow

Sub-workflow using properties defined at top-level workflow

```
<workflow 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/workflow.xsd"
version="1.1">
    <parallel>
        <activity logStartedAsDebug="true" logFinishedAsDebug="true">
            <runIndependent>false</runIndependent>
            <moduleInstanceId>RTCTools_KOT_Check_$RUNTYPE$</moduleInstanceId>
            <moduleConfigFileName>RTCTools_Check_Qinc_Fx</moduleConfigFileName>
        </activity>
        <activity logStartedAsDebug="true" logFinishedAsDebug="true">
            <runIndependent>false</runIndependent>
            <moduleInstanceId>RTCTools_KOT_Check_$RUNTYPE$</moduleInstanceId>
            <moduleConfigFileName>RTCTools_Check_QO_Seed</moduleConfigFileName>
        </activity>
        <activity logStartedAsDebug="true" logFinishedAsDebug="true">
            <runIndependent>false</runIndependent>
            <moduleInstanceId>RTCTools_KOT_Check_$RUNTYPE$</moduleInstanceId>
            <moduleConfigFileName>RTCTools_Check_FB</moduleConfigFileName>
        </activity>
    </parallel>
</workflow>
```

loopLocationSetId

For a workflow, a location-loop can be defined in which each location (site) in a location set is run separately. The $LOOP_LOCATION_ID$ tag in the configuration can be used to point to the corresponding locationID of the current location. This is useful in combination with (related location) Constraints, because then it is possible to filter a locationSet on the basis of the $LOOP_LOCATION_ID$.

You can loop over a particular locationSet using the following config. In the ModuleInstance (e.g. exportActivity of a GeneralAdapter) in every 'loop' the $LOOP_LOCATION_ID$ tag will be replaced with the 'actual' locationId.

Sub-workflow using properties defined at top-level workflow

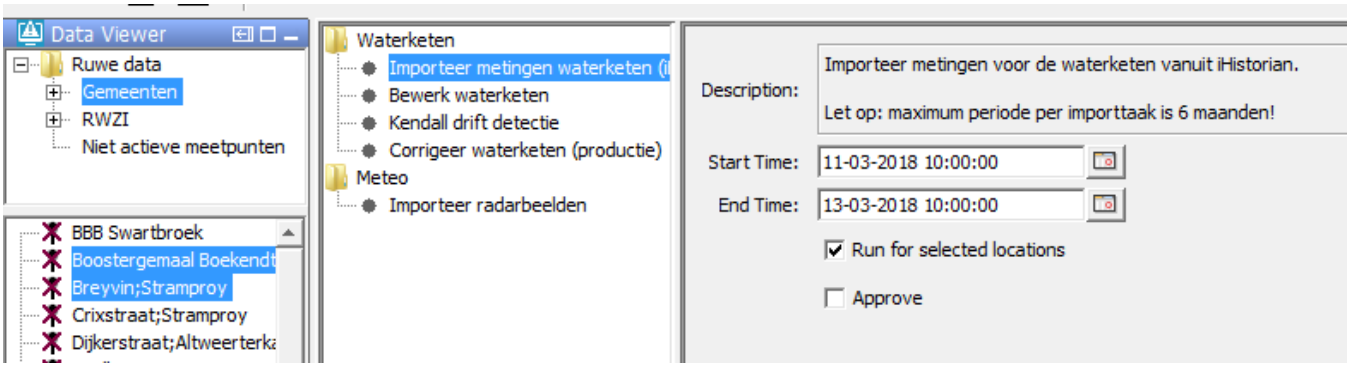
```
<activity>
        <runIndependent>true</runIndependent>
        <moduleInstanceId>ModuleInstanceId</moduleInstanceId>
        <loopLocationSetId>LocationSetId</loopLocationSetId>
</activity>
```

Example usage of $LOOP_LOCATION_ID$ in ModuleConfigFile:

$LOOP_LOCATION_ID$

```
                        <timeSeriesSet>
                                <moduleInstanceId>import</moduleInstanceId>
                                <valueType>scalar</valueType
                                <parameterId>H.obs</parameterId>
                                <locationId>$LOOP_LOCATION_ID$</locationId>
                                <timeSeriesType>external historical</timeSeriesType>
                                <timeStep unit="day"/> <relativeViewPeriod unit="day" start="$START_DAYS$" end="
1"/>
                                <readWriteMode>read only</readWriteMode>
                        </timeSeriesSet>
```

The locationLoop also works in combination with "run for selected locations", this will only run for the locations that are in the locationSet AND selection



completed (since 2021.01)

At the end of a sub-workflow the simulated time series for one or multiple module instances can be made visible for other workflows and users with an Operator Client before the workflow completes successfully. This can be achieved by adding a "completed" element at the end of a sub-workflow.

completed

```
<completed>
        <moduleInstanceId>TRITON_WestSussex</moduleInstanceId>
</completed>

or

<completed>
        <moduleInstanceIdPattern>TRITON_*</moduleInstanceIdPattern>
</completed>
```

This functionality is useful when a workflow consists of sub-workflows with activities that take long to run and that you want to see or use before the main workflow is completed.

It must be noted that the module instances become read-only for the rest of the workflow. When auto approve is configured for the workflow the simulated time series becomes visible for other workflows and users before the complete workflow completes. The workflow can also continue on different forecasting shells if needed.

See deleteTemporary if you want to transfer temporary timeseries to the next partition forecasting shell.

deleteTemporary

Explicit deletes temporary series. At the end workflow partition or the end of the workflow temporary series are automatically deleted. Deleting specified module
instances earlier prevents the temporary series are flushed to disk for no use when they are still in the memory buffer. Time series that are explicitly deleted in another
workflow partition then the partition were they were created are always flushed to the database. This makes it possible to use temporary time series even when the workflows
continues on a different forecasting shell. When the workflow terminates or completes all temporary series for the run are deleted from the database