24 ImportAmalgamate

What	ImportAmalgamate
Required	no
Description	Amalgamates external historical data
schema location	https://fewsdocs.deltares.nl/schemas/version1.0/importAmalgamate.xsd

Description

Workflows that import or process external historical data may produce small time series objects (stored as binary objects,blobs, in the time series database table). These small blobs may only cover a few time steps. If the time series table contains many of these small blobs, the number of records in the database can grow considerable and become very large (>1 mln records). In those cases it is advised to run a database maintenance workflow that amalgamates the small database records in the time series table into larger database records that cover larger time periods and store the time series data in larger blobs. In Delft-FEWS this can be done with the ImportAmalgamate module; this module amalgamates all time series records that are created in specific workflows. After the amalgamate module completed the amalgamate task, the original amalgamated workflows with all its imported external data is scheduled for deletion. The amalgamated time series will be stored in the database as time series that are created by the database maintenance workflow.

Note: Large grids, external forecasts and samples are not handled through this module.

Configuration of workflow duration

When the database maintenance workflow (with the amalgamate module) is executed for the first time the workflow can take some time to complete. In those cases it is advised to change the allowable run time of a workflow run. The fews.master.mc.conf file contains a chaser grace time property, that defines the maximum time that this workflow is allowed to run. Please consider whether this value needs to be temporarily increased.

Configuration

It is advised to use a separate database maintenance workflow that includes the ImportAmalgamate module instance. The configuration of the ImportAmalgamate module instance contains the following elements, see config example below.

workflowId

One or more workflow ids that import or create external historical time series over a short time span (scheduled frequently). Note that this should be the id of the main workflow, i.e. the one that is actually scheduled and leads to a task run. Referring to workflows that are part of the main workflow will not work.

importRunMinimalAge

Workflow runs younger than the specified age are skipped; only workflow runs (TaskRuns) that exist in the TaskRuns database table and are older than the importRunMinimalAge will be included in the ImportAmalgamate module.

Note: After the amalgamate has run it is no longer possible to create an archive with the exact original data available during the run. In those cases make sure the importAmalgamate workflow schedule and minimalAge are streamlined with the archive runs.

afterAmalgamateImportRunMetaDataExpiryTime

Since 2023.01. By default the import/update run is deleted from the database after amalgamation. It is possible to keep the metadata for a while after amalgamation. This metadata can be still seen in the time series lister as an entry without time series. **NB.** Only use for workflows that you need to keep, e.g. not for a frequently running SystemMetrics workflow since otherwise the SystemActivities table may grow unexpectedly.

amalgamateOrphans

For systems that did not have an amalgamate module running it maybe required to amalgamate the complete database. This should only be done once or when new workflows are added to the ImportAmalgamate module. It is possible to do this with the amalgamateOrphans=true. As this task may take too long (regular FSS tasks have a time out of 3 hours) this should be handled with care. In that case it may be useful to have the task run at an FSS that has no localDataStore, but has direct access to the central database. Only in that particular case the amalgamate runs (hard coded in the software) only for 1 hour. The task will be aborted after one hour and the remaining imports are not handled in that taskrun anymore. This is done on purpose, as it is now possible to schedule this task with an interval of e.g. 3 hours (making it possible to the MC to do all the required cleaning stuff like rollingBarrel and markedRecordManager). In a few hours (to even days or in some rare situations evens weeks) all the taskruns are handled and the complete database has been amalgamated. Do not forget to stop this scheduled task after the complete database is amalgamated.

Example

ModuleInstanceDescriptors.xml

```
<moduleInstanceDescriptor id="Amalgamate">
  <moduleId>ImportAmalgamate</moduleId>
</moduleInstanceDescriptor>
```

In the module instance configuration the workflowId's can be specified or a workflow pattern can be used when many workflows need to be amalgamated.

Amalgamate.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<importAmalgamate 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/importAmalgamate.xsd"
>
  <workflowId>Import_Data</workflowId>
  <workflowId>Procesoverzicht_Update</workflowId>
  <workflowIdPattern>*_Process_Observations</workflowIdPattern>
  <importRunMinimalAge unit="hour"/>
  <amalgamateOrphans>false</amalgamateOrphans>
</importAmalgamate>
```

Amalgamate.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml version="1.0" encoding="UTF-8"?>
        <importAmalgamate 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/importAmalgamate.
xsd">
        <workflowId>Import</workflowId>
        <workflowIdPattern>Import*</workflowIdPattern>
    <importRunMinimalAge unit="hour"/>
        <afterAmalgamateImportRunMetaDataExpiryTime multiplier="365" unit="day"/>
</importAmalgamate>
```

Best Practices

The Database Maintenance (ImportAmalgamate) workflow should run frequently (every 12 hours or 1 day) in order to amalgamate the small time series blobs generated in workflows that run with high frequency (5-15 minutes) to larger blobs of 12 hours or 1 day. A second ImportAmalgamate workflow should be scheduled that will run for example every week or every month that amalgamates the already amalgamated workflows in even larger blobs. To do so, schedule a new Database Maintenance workflow with an interval of 30 days that runs an ImportAmalgamate module instance that amalgamates all time series blobs generated in the Database Maintenance workflows. This second ImportAmalgamate step can reduce the number of records in the time series table considerably. The expiry time of the individual time series is preserved while amalgamated. Therefore it is important to configure an expiry time for the time series sets in the workflows that are amalgamated.

Amalgamate_Month.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<importAmalgamate 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/importAmalgamate.xsd"
>
  <workflowId>Database_Maintenance</workflowId>
  <importRunMinimalAge unit="day" multiplier="30"/>
  <amalgamateOrphans>false</amalgamateOrphans>
</importAmalgamate>
```

Steps

1. Identify which external historicals need to be amalgamated.
2. Pick a suitable expiry time T for the taskrun metadata. The taskrun expiry time for amalgamation timeseries should never be set to expire before the timeseries do. By default a TaskRun expires after 10 days and the timeseries become orphaned, but for amalgamation timeseries a much longer expiry time and taskrun expiry time is recommended. It is essential that the taskrun meta data for the "to be amalgamated imported external historical timeseries" does not expire before the amalgamation takes place. When using amalgamate, the taskrun meta data should only be deleted by the amalgamate module.
3. When import runs for external historical timeseries need to run regularly, set up a task and workflow for importing external historical timeseries. The timeseries expiry time is preserved by the amalgamate module and not influenced by the taskrun metadata expiry time. By default the timeseries expiry time is the same as that of the import run, but it is strongly recommended to overrule this by setting an explicit timeseries expiry time in the import module. Use the taskrun metadata expiry time T for the task determined under step @1. Set for instance the expiry time of the task in the Admin Interface to 50 years.
4. Optionally set up an Archive export task and workflow. Obviously this must be before the taskrun metadata expires and before a daily amalgamate has acted on the data.
5. Set up a task and workflow for running a daily amalgamate (after the archive export before taskrun of the import expires). Use the taskrun metadata expiry time T for the task. Set for instance the expiry time of the task in the Admin Interface.
6. When preserving external historical timeseries much longer than a week, setup a separate task and workflow for a weekly amalgamate. This workflow should be setup to process the output of the daily amalgamate. Obviously this workflow must therefore be run before the daily amalgamate taskrun metadata expires. Use the taskrun metadata expiry time T for the task. Set for instance the expiry time of the task in the Admin Interface..
7. When preserving external historical timeseries much longer than a month, setup a separate task and workflow for a monthly amalgamate. This workflow should be setup to process the output of the weekly amalgamate. Obviously this workflow must therefore be run before the weekly amalgamate taskrun metadata expires.

Troubleshooting

1. Verify that the workflow id's in the ImportAmalgamate module refer to the main workflows (i.e. the ones that are actually scheduled) and not to any workflows that are run as part of a main workflow.
2. Verify that the import workflows are inserting expected amounts of external historical TimeSeries with the correct period - e.g. the begin time must start after the end time of the previous run. Check in DatabaseLister!
3. Check that the processing steps that operate on the external historical data do not contain transformations that produce warnings about "Overwriting with missing" (evil!).
4. Verify the history popup (Ctrl-H) in the TimeSeriesDialog does not contain *multiple entries*.
5. Verify that the external historical TimeSeries records are created with a healthy expiryTime.
6. Verify that the TaskRuns for external historical TimeSeries records created with a healthy expiryTime.
7. Verify that the Tasks for external historical TimeSeries records created with a healthy expiryTime.
8. Does the ImportAmalgamate configuration include all the workflowIds of interest.
9. Does the ImportAmalgamate run without errors.

Amalgamate with multiple MC's

Special attention must be given to Delft-FEWS systems that have a duty-standby set-up. The amalgamate module will only amalgamate workflows that are run on the MC where the amalgamate module is scheduled on. In standard duty-standby systems the import and data processing tasks are run on both MC's. Also the amalgamate modules (database maintenance workflows) must be scheduled to run on both MC's. This is especially relevant when some of the imported data is synchronised between the MC's. Import, processing and database maintenance workflows may never be run in failover mode.

There has been a change in the Delft-FEWS code since April 2016, when the check on MC's is introduced.

Tuning the amalgamate using SQL

While tuning the ImportAmalgamate the following sql can be used. Please enable the appropriate *7 day condition on creationTime* in below query for your database flavour to see what Tasks have been produced over the last week with external historical timeseries. Also the AdminInterface database trend information page can be very useful for this purpose.

```
SELECT t1.workflowId, t1.maxTaskExpiry, t1.taskCnt, t1.taskRunCnt, t1.maxTaskRunExpiryTime, t2.timeSeriesCnt,
t2.maxTimeSeriesExpirytime FROM (
 SELECT t3.workflowId, t3.taskId, t3.maxTaskExpiry, t3.taskCnt, t4.taskRunCnt, t4.maxTaskRunExpiryTime FROM (
 SELECT workflowId, taskId, MAX(expiryTime) AS maxTaskExpiry, count(*) AS taskCnt FROM Tasks WHERE

----postgresql ----
-- creationTime > current_date - interval '7 days' AND

---- sqlserver ----
-- creationTime > DATEADD(DAY,-7, GETDATE()) AND

---- oracle ----
-- creationTime > TRUNC(sysdate) - INTERVAL '7' DAY AND

workflowId IN (
SELECT workflowId FROM Tasks WHERE taskId IN (SELECT taskId FROM TaskRuns WHERE taskRunId IN (SELECT
creatorTaskRunId FROM TimeSeries WHERE timeSeriesType=0))
---- or only already amalgamated workflowIds, e.g. : ----
--SELECT workflowid FROM WorkflowFiles WHERE workflowId='Import_Amalgamate'
)
 GROUP BY taskId, workflowId
 ) t3 LEFT JOIN (
 SELECT workflowId, t.taskId, count(tr.taskRunId) AS taskRunCnt, max(tr.expiryTime) AS maxTaskRunExpiryTime
 FROM Tasks t, TaskRuns tr
 WHERE t.taskId=tr.taskId
 GROUP BY t.taskId, workflowId
 ) t4 ON t3.taskId=t4.taskId AND t3.workflowId=t4.workflowId
) t1 LEFT JOIN (
 SELECT t.taskId, workflowId, count(1) AS timeSeriesCnt, MAX(ts.expiryTime) AS maxTimeSeriesExpirytime
 FROM Tasks t, TaskRuns tr, TimeSeries ts
 WHERE t.taskId=tr.taskId AND ts.creatorTaskRunId=tr.taskRunId
 GROUP BY workflowId, t.taskId
) t2 ON t1.taskId=t2.taskId AND t1.workflowId=t2.workflowId
ORDER BY t1.workflowId, t1.taskId
```