Kuali People Management for the Enterprise System Administration Guide - Batch Jobs

Table of Contents

List of Tables

Chapter 1. Batch Jobs

One of KPME's more advanced features is the ability to run calculations in the background at certain times, known as batch jobs. These batch jobs can run at any time of the day, allowing large calculations to run during the middle of the night when the system does not have much load. In KPME's case, theses batch jobs include creating timesheets and leave calendars, automatically submitting and approving these documents, and performing large accrual and carry over calculations.

Basic Setup

The KPME batch system uses Quartz (http://quartz-scheduler.org) to schedule jobs. Quartz allows scheduling of both periodic and on-demand jobs, as well as providing clustering and load balancing features. The purpose of this guide is to just cover the basics of the KPME jobs; it is expected that administrators will consult the Quartz documentation to implement more advanced Quartz features.

There is no additional setup to get batch jobs to run automatically. In the basic setup, a polling program starts up five minutes after application starts to determine whether it needs to schedule any batch jobs. It will keep doing this every five minutes until the application terminates. The batch jobs are, however, highly configurable and can be overridden in **kpme-config.xml**. The default setup in **kpme-config-defaults.xml** is as follows:

Table 1.1. Batch Job Configuration Parameters

Parameter	Description	Default Value
kpme.org.quartz.threadPool.threadCount	The number of threads allocated to run simultaneous jobs.	5
kpme.batch.user.principalName	The KIM principal name of the batch job user.	admin
kpme.batch.startDelay.milliseconds	The start delay (in milliseconds) before batch job startup.	300000
kpme.batch.repeatInterval.milliseconds	The interval (in milliseconds) between polling for new jobs to schedule.	300000
kpme.batch.calendarEntriesPollingWindow.days	The number of days to poll both before and after the current date to detect whether jobs based on calendar entries need to be scheduled.	30
kpme.batch.leavePlanPollingWindow.days	The number of days to poll both before and after the current date to detect whether jobs based on leave plans need to be scheduled.	30
kpme.batch.accrual.cronExpression	The cron expression on when to run the Accrual Service.	0 0 1 1 * ? 2099
kpme.batch.leaveCalendarDelinquency.cronExpression	The cron expression on when to send out notifications that a Leave Calendar is delinquent.	0 0 1 1 * ? 2099

Parameter	Description	Default Value
kpme.batch.serializer.cronExpression	The cron expression on when to serialize Time Blocks to CSV and XML.	0 0 1 1 * ? 2099

Note

The Quartz cron expressions are slightly different from normal cron expressions in that they add a seconds and an optional year field. For more information, see the Quartz Cron Tutorial.

The next section of this guide will cover exactly what types of jobs these parameters control.

Available Batch Jobs

KPME currently has several batch jobs available to run. Some of them are controlled by objects in the system (like Calendar Entry or Leave Plan), while others are controlled by cron expressions. This section will cover those in greater detail.

Calendar Entry Jobs

Many of the jobs in KPME are based on the Calendar Entry object. This object not only controls when user calendars begin and end but also when batch jobs are to be run in relation to that calendar entry in the form of dates and times. These fields can be left blank for any calendar entry, causing the batch job to not run for that calendar entry. The four available dates are as follows:

Table 1.2. Calendar Entry Job Dates and Times

Batch Job Date/Time Entry	Description
Batch Initiate Date/Time	The date and time when to initialize timesheets or leave calendars associated with this calendar entry.
Batch End Pay Period Date/Time	The date and time when to close timesheet clock logs associated with this calendar entry.
Batch Employee Approval Date/Time	The date and time when to approve missed punch documents and submit timesheets or leave calendars associated with this calendar entry.
Batch Supervisor Approval Date/Time	The date and time when to approve timesheets or leave calendars associated with this calendar entry.

There are six jobs associated with Calendar Entries:

- Initiate Job

  Initiates timesheets or leave calendars for the dates associated with this calendar entry. This job is typically run a few days before the start date of this calendar entry so these documents are ready to go when the new period starts.

- End Reporting Period Job

  Sends a notification to all users who use leave calendars that they should submit their leave calendars so they can be reviewed by their supervisor. This is automatically sent at the end date of this calendar entry and it is the only one not configurable.

- End Pay Period Job

  Closes all clock logs belonging to any timesheets associated with this calendar entry. The clock logs are closed at the end of the pay period and then reopened immediately after (in the next calendar entry) so that clock log times are only calculated for this calendar entry. This job is typically run right after the end date of this calendar entry.

- Missed Punch Approval Job

  Approves all of the missed punch documents attached to any timesheets associated with this calendar entry. This job is typically run right after the end date of this calendar entry.

- Employee Approval Job

  Submits any timesheets or leave calendars associated with this calendar entry to the supervisors. This job is typically run a couple days after the end date of this calendar entry.

- Supervisor Approval Job

  Approves any timesheets or leave calendars associated with this calendar entry if they have been sent to the supervisors for approval. If there are any non-approved missed punch documents belonging to these timesheets or if any of these timesheets or leave calendars are not in a state to be approved, then the job is rescheduled until these are done. This job is typically run a couple days after the Employee Approval Job.

Leave Plan Jobs

The Carry Over job is based on the Leave Plan object. This object not only controls when user leave plans begin but also when the Carry Over job is to be run in relation to that leave plan in the form of dates and times. These fields can be left blank for any leave plan, causing the batch job to not run for that leave plan. The fields that control when the Carry Over job is run are called Batch Prior Year Carry Over Start Date/Time.

The Carry Over job adds leave blocks for any previous leave plan years that hold the accrued leave amounts from year to year so that the Accrual Service does not have to go back to a user's beginning service date to calculate accrued leave.

Cron Expression Jobs

Some jobs are scheduled to run via cron expressions. These cron expressions are powerful in that they can be set to run and repeat to run at any interval and at any time. The most common settings are to run nightly or monthly but many combinations are possible. These cron jobs can also be "turned off" by setting them to "0 0 1 1 * ? 2099", which is the first day of the first month in the year 2099. By default, all of these jobs are turned off since there is no way of knowing reasonable values for when an institution may want to run these.

There are three jobs controlled by cron expressions:

- Time Block Serializer Job

  Serializes all available timeblocks to both CSV and XML.

- Accrual Job

  Runs the Accrual Service, which calculates leave amounts and adds needed leave blocks on to leave calendars.

- Leave Calendar Delinquency Job

  Sends out a notification to all users who have leave calendars that have not been submitted for some time and are now considered delinquent.

Quartz Implementation

Job scheduling information is stored in the KPME database under the **QRTZ_*** tables that were provided by the Quartz install. By default, every single job ever run or scheduled to be run is stored in the database. In Quartz, there are two concepts necessary to understand in order to understand how the KPME Quartz system is set up:

- Jobs

  Jobs store the necessary information to run a process. Each Job is assigned to a Job Group, which provides additional information on how the job is grouped in the system.

- Triggers

  Triggers are what actually schedule Jobs. They store information that allows Quartz to calculate whether a particular job has been run already or not. Each Trigger is assigned to a Trigger Group, very similar as to how Jobs are assigned to Job Groups.

The most important table in the KPME database to view when trying to understand Quartz is **QRTZ_TRIGGERS**. Among others, it contains four fields for the Trigger Name, the Trigger Group, the Job Name, and the Job Group, that administrators can view to see what jobs have already run and what jobs are scheduled to run.

KPME Quartz Job and Trigger Example

To understand how these four fields are populated, consider the example where the system is about to schedule an Initiate Job. The poller for the Initiate Job determines that there is a person with id "user" who needs to have a timesheet initiated for the calendar entry with id "10000" starting on January 1, 2010 at midnight GMT. The four fields are populated as follows:

- Job Name: *InitiateJob-Job-principalId=user*

  The Job Name only includes the **principalId** since that is who the Initiate Job is being run for.

- Job Group: *InitiateJob-JobGroup-hrCalendarEntriesId=10000*

  The Job Group name only includes the **hrCalendarEntriesId** since that is the calendar entry that is currently being processed. Several Jobs for multiple principals can be associated with this calendar entry Job Group.

- Trigger Name: *InitiateJob-Trigger-date=2010-01-01T00:00:00.000-0000*

  The Trigger Name only includes the **date** since that is what determines when it is run.

- Trigger Group: *InitiateJob-TriggerGroup-principalId=user&hrCalendarEntriesId=10000*

  The Trigger Group includes both the **hrCalendarEntriesId** and the **principalId** to make sure that when the system polls the entries again, it does not schedule this particular Job again.

This means that the timesheet is initiated for user only once for calendar entry 10000 on January 1, 2010. Other users associated with calendar entry 10000 will be scheduled for a different Job but in the same Job

Group. Since only one Trigger can be associated per one Job and the combination of Trigger Name and Trigger Group must be unique in the system, then each of these other users have the same Trigger Name but a different Trigger Group with both the **hrCalendarEntriesId** and the **principalId** to meet this Quartz requirement. All four of these fields can hold up to 200 characters, so there is little chance that even long **principalId**s will cause inserts to fail.

Spring Configuration

Quartz's main setup is in **SpringBeans.xml** under "kpmeScheduler". This is where KPME sets all of its Quartz defaults and plugins. Implementers may wish to override this to provide different defaults, especially in the section "quartzProperties". More information can be found in the Quartz documentation.