Scheduler JAVA JOB Creation
Oracle FLEXCUBE Investor Servicing
Release 12.4.0.0.0
[May] [2017]

ORACLE®
FINANCIAL SERVICES
# Table of Contents

1. INTRODUCTION ................................................................................................................. 1-3
2. BACKGROUND ....................................................................................................................... 2-1
3. EXAMPLE ............................................................................................................................. 3-1
1. Introduction

This document introduces the reader to the procedure to be followed for Scheduler JAVA JOB Creation for Oracle FLEXCUBE Investor Servicing.
2. Background

This document is based on the requirement of creating JAVA Jobs to be executed by Quartz Scheduler.

This document describes the procedure to be followed to accomplish the above.
This section describes job creation:

1. Launch the job maintenance screen STDJOBMT for creation of a Job. User can maintain Interface, Reports and PL SQL Jobs.

**Interface Job definition**

**Event Based:**

For Interface Job maintenance, user has to maintain com.ofss.scheduler.quartz.fcis.FCISImpExpSchExecutor class.

For Event Based Job Scheduling, EVENTCODE need to be maintained in Parameters. Valid Event Codes are available in Eventstbl.
Time Based:

To maintain time based job, we have to maintain cron expression.

TimeZone Based

For Timezone based job scheduling, valid timezone need to be maintained in parameters. Timezone parameter should have standard Quartz supported timezone value.
**REPORT Job definition**

To schedule BIP report generation, user has to maintain the `com.ofss.scheduler.quartz.fcis.FCISReportSpollExecutor` class.
User has to maintain TASKID parameter to specify the Report Parameters. UTDTASK is the screen to maintain Report Parameters.

A Java job can be stateful or stateless.

**Stateful Job**

If the job is configured to be STATEFUL then missed instances will be queued up so that scheduler would start executing once the long running job ends. It means, there will be no parallel execution of the java class.

To make a class as Stateful, please add two annotations to the job class.

- @PersistJobDataAfterExecution
- @DisallowConcurrentExecution

**Stateless Job**

If the job is configured as STATELESS, threads can be run in parallel and can execute the same java class.

If the job is configured as stateless, the value of field **Max Number Instances** will denote the number of threads that can be executed in parallel. If the job is configured as stateful, then the above value will denote the number of missed instances that will be queued up so that Quartz Scheduler would start executing them once a long running job ends. This field specifies the number of such job instances that needs to be queued up.

The class defined in the class field has to be provided a fully qualified name.

**Job params** can be added to a particular java job in this screen as shown above.