Scheduler JAVA JOB Creation
Oracle FLEXCUBE Universal Banking
Release 12.4.0.0.0

[May] [2017]

ORACLE
FINANCIAL SERVICES
# Table of Contents

1. INTRODUCTION .................................................................................................................. 1-2
2. BACKGROUND .................................................................................................................... 2-1
3. PROCEDURE ....................................................................................................................... 3-1
4. EXAMPLE ............................................................................................................................. 4-1
1. Introduction

This document introduces the reader to the procedure to be followed for Scheduler JAVA JOB Creation for FLEXCUBE Universal Banking Solutions.
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.
3. Procedure

This section describes job creation:

1. Launch the job maintenance screen STDJOBMT for creation of a Job.
2. Specify the field values. For information regarding individual fields please refer to Installation Docs.

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.
4. Example

Suppose we have created a new FTP_COPY job, to copy ftp files from one location to another. Here it is a java job, where the class to be executed is FtpCopy.java.

So, we have provided the fully qualified class name as com.ofss.scheduler.quartz.ftp. FtpCopy.

The physical location of the java file is
INFRA\FCJNeoWeb\Javasource\com\ofss\scheduler\quartz\ftp\FtpCopy.java.

In this class, we have to override the execute method of Job class to provide the implementation logic, which gets executed when the quartz scheduler executes the class. The execute method format is as below.

```java
public void execute(JobExecutionContext context) throws JobExecutionException {}`n
```

The sample java file attached here.

FtpCopy.java