

**Oracle® Retail Job Orchestration and Scheduler  
(JOS)**  
Implementation Guide  
Release 16.0.21

May 2017

Oracle® Retail Job Orchestration and Scheduler (JOS) Implementation Guide, Release 16.0.21

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Primary Author:

Contributors:

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

## Value-Added Reseller (VAR) Language

### Oracle Retail VAR Applications

The following restrictions and provisions only apply to the programs referred to in this section and licensed to you. You acknowledge that the programs may contain third party software (VAR applications) licensed to Oracle. Depending upon your product and its version number, the VAR applications may include:

(i) the **MicroStrategy** Components developed and licensed by MicroStrategy Services Corporation (MicroStrategy) of McLean, Virginia to Oracle and imbedded in the MicroStrategy for Oracle Retail Data Warehouse and MicroStrategy for Oracle Retail Planning & Optimization applications.

(ii) the **Wavelink** component developed and licensed by Wavelink Corporation (Wavelink) of Kirkland, Washington, to Oracle and imbedded in Oracle Retail Mobile Store Inventory Management.

(iii) the software component known as **Access Via**<sup>TM</sup> licensed by Access Via of Seattle, Washington, and imbedded in Oracle Retail Signs and Oracle Retail Labels and Tags.

(iv) the software component known as **Adobe Flex**<sup>TM</sup> licensed by Adobe Systems Incorporated of San Jose, California, and imbedded in Oracle Retail Promotion Planning & Optimization application.

You acknowledge and confirm that Oracle grants you use of only the object code of the VAR Applications. Oracle will not deliver source code to the VAR Applications to you. Notwithstanding any other term or condition of the agreement and this ordering document, you shall not cause or permit alteration of any VAR Applications. For purposes of this section, "alteration" refers to all alterations, translations, upgrades, enhancements, customizations or modifications of all or any portion of the VAR Applications including all reconfigurations, reassembly or reverse assembly, re-engineering or reverse engineering and recompilations or reverse compilations of the VAR Applications or any derivatives of the VAR Applications. You acknowledge that it shall be a breach of the agreement to utilize the relationship, and/or confidential information of the VAR Applications for purposes of competitive discovery.

The VAR Applications contain trade secrets of Oracle and Oracle's licensors and Customer shall not attempt, cause, or permit the alteration, decompilation, reverse engineering, disassembly or other reduction of the VAR Applications to a human perceivable form. Oracle reserves the right to replace, with functional equivalent software, any of the VAR Applications in future releases of the applicable program.



---

---

# Contents

<b>Send Us Your Comments</b> .....	<b>xi</b>
<b>Preface</b> .....	<b>xiii</b>
Documentation Accessibility.....	xiii
Related Documents.....	xiii
Customer Support.....	xiii
Improved Process for Oracle Retail Documentation Corrections .....	xiii
Oracle Retail Documentation on the Oracle Technology Network.....	xiv
<b>1 Introduction</b> .....	<b>1</b>
Standards and Specifications.....	1
Java Platform Enterprise Edition (Java EE) .....	1
Java Batch.....	1
Java EE Server .....	2
Java Batch Overview.....	2
<b>2 Job Orchestration and Scheduler</b> .....	<b>3</b>
What is Job Orchestration and Scheduler (JOS)?.....	3
<b>3 JOS Components</b> .....	<b>5</b>
JOS Architecture.....	5
<b>4 Job Admin</b> .....	<b>7</b>
Job Admin Concepts.....	7
Job Admin Components.....	7
RESTful Services .....	7
Job Admin UI.....	10
Best Practices for jobs .....	10
Job Admin Security.....	10
Job Admin Customization .....	10
Job Admin Troubleshooting .....	10
Deployment Error.....	10
Runtime WSMException.....	11
Missing system credentials.....	12
Missing system options.....	12
<b>5 13JOS Process Flows</b> .....	<b>13</b>
Process Flow Concepts.....	13
DSL (Domain Specific Language).....	13
Begin Activity.....	14
Activity .....	14
End Activity .....	14
Process Variables .....	14
Process Flow DSL.....	14
Process Flow DSL characteristics.....	14

---

DSL Keywords .....	15
Process Flow API .....	16
Process Flow Variables.....	17
Process Flow Instrumentation.....	18
Sub Processes .....	18
Process Schema.....	18
Process Restart.....	18
Statuses .....	19
Steps for implementing a JOS Flow .....	19
Activity Features .....	19
Skip Activity .....	19
REST endpoint to set the skip activity flag.....	20
Hold/Release Activity .....	20
REST endpoint to set the hold activity flag.....	20
Callback Service .....	20
How to start Process Flow with input parameters.....	20
Call back from Processflow .....	21
How to invoke the Callback Service declaratively.....	21
Process Security.....	27
Troubleshooting .....	28
Process Flow Didn't Start.....	28
Deleted Process Flow Still Listed in the UI .....	28
Best Practices for Process Flow DSL.....	28
<b>6 Scheduler.....</b>	<b>31</b>
Scheduler Overview .....	31
JOS Scheduler Features .....	31
Scheduler Concepts .....	31
Schedule Definition .....	31
Schedule Execution.....	31
Schedule Types.....	31
Interval Schedules.....	32
Calendar Schedules .....	32
Scheduling Mechanisms .....	32
Simple Scheduling .....	32
Advanced Scheduling .....	32
Schedule Frequency.....	33
Schedule Action .....	34
Schedule Action Type .....	35
Schedule Status .....	36
Scheduler Runtime.....	36
Scheduler Startup.....	36
Schedule Runtime Execution .....	37

---

Schedule Execution - Async Action.....	37
Schedule Execution - Sync Action .....	38
Schedule Execution Failover .....	38
Schedule Notification .....	38
Scheduler Infrastructure Schema.....	39
Best Practices for Scheduler .....	39
Scheduler Console.....	39
Schedule Summary .....	40
Schedules and Executions.....	40
Schedule Executions Failed Today .....	40
Schedule Executions Completed / Triggered Today.....	41
Schedule Executions In Progress Today .....	41
Schedules Past Due.....	41
Manage Schedules .....	41
Creating a Schedule .....	42
Schedule Frequency .....	43
Schedule Notification .....	44
Scheduler Security Considerations.....	49
Scheduler Security .....	49
Scheduler Operational Considerations .....	50
Users Roles for Monitoring and Administration.....	50
Monitoring Schedules .....	50
Scheduler Log Files.....	51
Maintaining Historical Schedule Executions .....	51
Scheduler Customization.....	51
Customizing Seed Data Schedules .....	52
Customizing Schedule Actions .....	53
Scheduler Troubleshooting.....	54
Scheduler Known issues .....	54
<b>7 Use Cases.....</b>	<b>55</b>
How do I create a batch job in Job Admin? .....	55
Sample Job XML.....	55
How do I pass job parameters to a shell script invoked by job? .....	55
How do I pass system options to a shell script invoked by job? .....	56
How do I pass system properties to a shell script invoked by job? .....	56
How do I chain multiple jobs in a single flow?.....	57
Sample Process Flow .....	57
How do I create split flows? .....	58
Sample Split Flow .....	58
How do I create split and join flows? .....	60
Sample Split and Join Flow.....	60
Def Process Flow .....	61

---

XYZ Process Flow .....	61
How do I create a join flow with other flows? .....	62
Sample Join Flow .....	62
How do I share data between process flows? .....	63
Sample Flow that shares information with other flows .....	63
How do I create a schedule in Scheduler? .....	64
Sample seed data to create schedule .....	64
<b>8 Security .....</b>	<b>65</b>
RESTful Services .....	65
Sample seed data created during installation .....	65
<b>9 Pre-Implementation Considerations .....</b>	<b>67</b>
Thread Pool Size in WebLogic .....	67
Database Connection Pool Size in WebLogic.....	67
<b>10 High Availability Considerations.....</b>	<b>69</b>
High Availability.....	69
WebLogic Server Cluster Concepts .....	69
Scaling JOS.....	69
JOS on Cluster .....	70
Logging .....	70
Update Log Level.....	70
Create/Update/Delete System Options.....	71
Create/Update/Delete System Credentials.....	71
Scheduler Configuration Changes for Cluster.....	71
<b>11 Deployment Architecture .....</b>	<b>73</b>
JOS and BDI deployment architecture for RMS.....	73
JOS Deployment Architecture.....	73
JOS Scalable Deployment Architecture.....	74
<b>12 Performance Considerations.....</b>	<b>75</b>
CPU and Memory considerations.....	75
<b>A Appendix A: Scheduler REST Endpoints .....</b>	<b>77</b>
<b>B Appendix B: Process Flow REST Endpoints .....</b>	<b>78</b>
<b>C Appendix C: Job Admin REST Endpoints.....</b>	<b>81</b>
<b>D Appendix D: System Setting Service.....</b>	<b>83</b>
Managing System Options using curl.....	83
Create system option.....	84
Update System Option.....	84
Delete System Option.....	84
Reset System Options Cache .....	84
List System Options.....	84
Managing Credentials Using Curl.....	84
Create Credential .....	84



---

Update Credential.....	84
Delete Credential .....	85
List Credentials .....	85
<b>E Appendix E: Scheduler UI Screenshots .....</b>	<b>87</b>
Schedule Summary .....	87
Manage Schedules - Schedule Executions .....	87
Manage Schedules - Create Schedule .....	88
Schedule Executions .....	88
System Logs .....	89
<b>F Appendix F: Process Flow UI Screenshots .....</b>	<b>91</b>
Process Flow Live.....	91
Manage Process Flow - Process Flow Executions.....	91
Manage Process Flow - Process Flow Configurations .....	92
Manage Process Flow - Launch Process Flow.....	92
Manage Process Flow - Process Flow Details - Process Details.....	93
Manage Process Flow - Process Flow Details - Process DSL.....	93
Historical Process Flow Executions .....	94
Manage Configurations.....	94
System Logs .....	95
<b>G Appendix G: Job Admin UI Screenshots .....</b>	<b>97</b>
Batch Summary .....	97
Manage Batch Jobs - Job Executions .....	97
Manage Batch Jobs - Job Launch.....	98
Manage Batch Jobs - Job Definition - Job Details.....	98
Manage Batch Jobs - Job Definition - Job XML Content .....	99
Manage Configurations.....	99
System Logs .....	100



---

---

# Send Us Your Comments

Oracle Retail Job Orchestration and Scheduler (JOS), Implementation Guide, Release 16.0.21

Oracle welcomes customers' comments and suggestions on the quality and usefulness of this document.

Your feedback is important, and helps us to best meet your needs as a user of our products. For example:

- Are the implementation steps correct and complete?
- Did you understand the context of the procedures?
- Did you find any errors in the information?
- Does the structure of the information help you with your tasks?
- Do you need different information or graphics? If so, where, and in what format?
- Are the examples correct? Do you need more examples?

If you find any errors or have any other suggestions for improvement, then please tell us your name, the name of the company who has licensed our products, the title and part number of the documentation and the chapter, section, and page number (if available).

---

---

**Note:** Before sending us your comments, you might like to check that you have the latest version of the document and if any concerns are already addressed. To do this, access the Online Documentation available on the Oracle Technology Network Web site. It contains the most current Documentation Library plus all documents revised or released recently.

---

---

Send your comments to us using the electronic mail address: [retail-doc\\_us@oracle.com](mailto:retail-doc_us@oracle.com)

Please give your name, address, electronic mail address, and telephone number (optional).

If you need assistance with Oracle software, then please contact your support representative or Oracle Support Services.

If you require training or instruction in using Oracle software, then please contact your Oracle local office and inquire about our Oracle University offerings. A list of Oracle offices is available on our Web site at [www.oracle.com](http://www.oracle.com).



---

---

# Preface

## and Scheduler (JOS) **Documentation Accessibility**

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at

<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

### **Access to Oracle Support**

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit

<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit

<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

## **Related Documents**

For more information, see the following documents in the Oracle Retail Job Orchestration and Scheduler (JOS) Release 15.0 documentation set:

*Oracle Retail Integration Cloud Service Administration Guide*

*Oracle Retail Integration Cloud Service Administrator Action List*

## **Customer Support**

To contact Oracle Customer Support, access My Oracle Support at the following URL:

<https://support.oracle.com>

When contacting Customer Support, please provide the following:

- Product version and program/module name
- Functional and technical description of the problem (include business impact)
- Detailed step-by-step instructions to re-create
- Exact error message received
- Screen shots of each step you take

## **Improved Process for Oracle Retail Documentation Corrections**

To more quickly address critical corrections to Oracle Retail documentation content, Oracle Retail documentation may be republished whenever a critical correction is needed. For critical corrections, the republication of an Oracle Retail document may at times **not** be attached to a numbered software release; instead, the Oracle Retail document will simply be replaced on the Oracle Technology Network Web site, or, in the case of Data Models, to the applicable My Oracle Support Documentation container where they reside.

An updated version of the applicable Oracle Retail document is indicated by Oracle part number, as well as print date (month and year). An updated version uses the same part number, with a higher-numbered suffix. For example, part number E123456-02 is an updated version of a document with part number E123456-01.

If a more recent version of a document is available, that version supersedes all previous versions.

---

## Oracle Retail Documentation on the Oracle Technology Network

Oracle Retail product documentation is available on the following web site:

<http://www.oracle.com/technetwork/documentation/oracle-retail-100266.html>

---

---

# Introduction

Job Orchestration and Scheduler (JOS) is a product suite for managing, executing, orchestrating and scheduling batch jobs for an enterprise.

## Standards and Specifications

JOS is designed and built on industry standard nonproprietary Java EE 7 and Java Batch (JSR 352).

### Java Platform Enterprise Edition (Java EE)

Java Platform Enterprise Edition (Java EE) is an umbrella standard for Java's enterprise computing facilities. It bundles together technologies for a complete enterprise-class server-side development and deployment platform in java.

Java EE specification includes several other API specifications, such as JDBC, RMI, Transaction, JMS, Web Services, XML, Persistence, mail, and others and defines how to coordinate among them. Java EE specification also features some specifications unique to enterprise computing. These include Enterprise JavaBeans (EJB), servlets, portlets, Java Server Pages (JSP), Java Server Faces (JSF) and several Web service technologies.

A Java EE application server manages transactions, security, scalability, concurrency, pooling, and management of the EJB/Web components that are deployed to it. This frees the developers to concentrate more on the business logic/problem of the components rather than spending time building scalable, robust infrastructure on which to run on.

### Java Batch

JSR 352 is a Java specification for building, deploying, and running batch applications. *Batch* is an industry metaphor for background bulk processing. Myriad business processes depend on batch processing and demand powerful standards-based facilities for enabling this essential workload type.

JSR 352 addresses three critical concerns: a batch programming model, a job specification language, and a batch runtime. This constitutes a separation of concerns.

1. Application developers have clear, reusable interfaces for constructing batch style applications;
2. Job writers have a powerful expression language for how to execute the steps of a batch execution;
3. Solution integrators have a runtime API for initiating and controlling batch execution.

JSR 352 defines a Job Specification Language (JSL) to define batch jobs, a set of interfaces that describes the artifacts that comprise the batch programming model to implement batch business logic, and a batch runtime for running batch jobs, according to a defined life cycle.

The batch runtime is a part of the Java EE 7 runtime and has full access to all other features of the platform, including transaction management, persistence, messaging, and more

## Java EE Server

WebLogic Server implements the Java EE specification and is the Java EE server vendor for JOS in this release. WebLogic server provides many additional services beyond the standard services required by the Java EE specification.

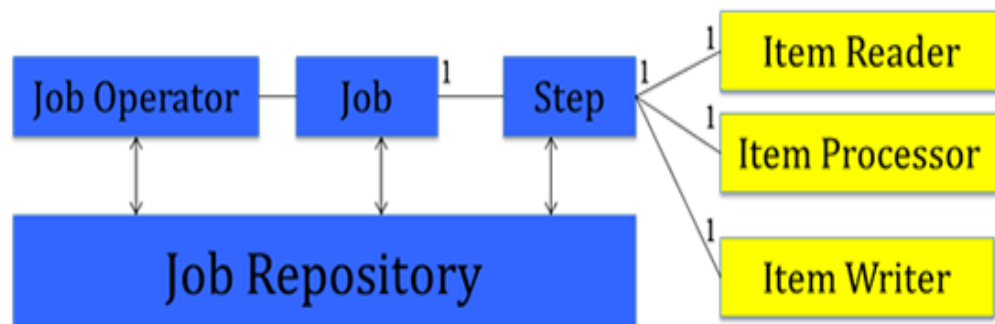
See the WebLogic® Application Server documentation for more information:

<http://docs.oracle.com/middleware/12211/wls/index.html>

<http://www.oracle.com/technetwork/middleware/fusion-middleware/documentation/index.html>

## Java Batch Overview

Batch processing for Java platform was introduced in Java EE 7. It provides programming model for batch applications and a runtime to run and manage batch jobs. Batch processing is typically bulk oriented, non-interactive, and long running.



A job encapsulates the batch process. A job contains one or more steps. A job is put together using Job Specification language (JSL) that specifies the sequence in which steps must be executed.

- A step contains all the necessary logic and data to perform the actual processing. A chunk-style step contains ItemReader, ItemProcessor, and ItemWriter.
- Job Operator provides an interface to manage all aspects of job processing.
- Job Repository holds information about jobs currently running and jobs that ran in the past.



---

---

# Job Orchestration and Scheduler

## What is Job Orchestration and Scheduler (JOS)?

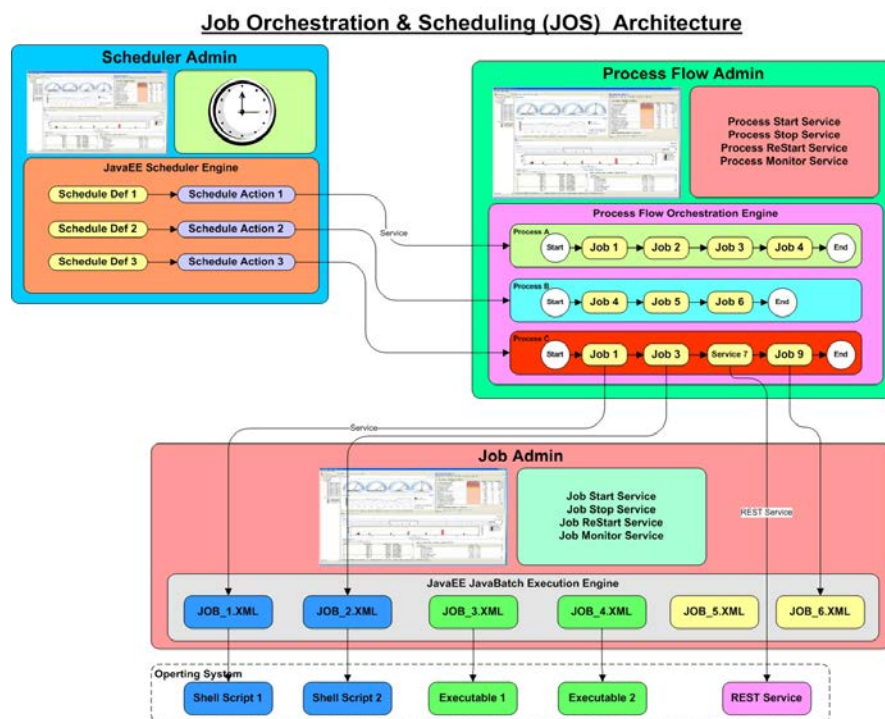
Job Orchestrator and Scheduler (JOS) is a generic tool to schedule tasks, manage and orchestrate dependencies between tasks, and run non interactive long running jobs.



## JOS Components

- **Scheduler** – A generic GUI tool used to define and manage time based scheduling work.
- **Process Flow** – Defines workflow by connecting and orchestrating multiple executable tasks. Provides an engine to execute the workflows.
- **Job Admin** – A robust task execution engine based on standard JavaBatch (JSR352) technology. Provides a GUI to manage and monitor jobs.

### JOS Architecture





---



---

## Job Admin

### Job Admin Concepts

- Application that provides management and monitoring of batch jobs.
- Built on JavaEE JSR352 JavaBatch standard, available in WebLogic.
- GUI to manage/start/restart batch jobs.
- JavaBatch Runtime Engine to execute job.xml files.
- Services available to start/restart/monitor jobs programmatically.
- Monitor executions and logs.

### Job Admin Components

#### RESTful Services

##### Batch Service

Batch service is a RESTful service that provides various endpoints to manage batch jobs. Here are key end points in Batch Service.

##### Start Job

This end point starts a job asynchronously based on a job name and returns the execution id of the job in the response.

Path: /batch/jobs/<jobName>

HTTP Method: POST

Inputs

Job Name as path parameter

Job Parameters as a query parameter. Job Parameters is a comma separated list of name value pairs. This parameter is optional.

Sample Request

```
http://localhost:7001/jos-batch-job-admin/resources/batch/jobs/ShellCommandRunnerJob?jobParameters=key=value
```

Successful Response

XML

```
<executionIdVo targetNamespace="">
  <executionId>1</executionId>
  <jobName>ShellCommandRunnerJob</jobName>
</executionIdVo>
```

JSON

```
{
  "executionId": 1,
  "jobName": "ShellCommandRunnerJob"
}
```

Error Response

XML

```
<exceptionVo targetNamespace="">
```

```
<statusCode>404</statusCode>
<status>NOT_FOUND</status>
<message>HTTP 404 Not Found</message>
<stackTrace></stackTrace> <!-- optional -->
</exceptionVo>
```

#### JSON

```
{
  "statusCode": "404",
  "status": "NOT_FOUND",
  "message": "HTTP 404 Not Found",
  "stackTrace": ""
}
```

### Restart Job

This end point restarts a job asynchronously using job execution id and returns the new job execution id.

Path: /batch/jobs/executions/{executionId}

HTTP Method: POST

Inputs: executionId as path parameter

#### Sample Request

http://localhost:7001/jos-batch-job-admin/resources/batch/jobs/executions/2

#### Successful Response

##### XML

```
<executionIdVo targetNamespace="">
<executionId>2</executionId>
<jobName>ShellCommandRunnerJob</jobName>
</executionIdVo>
```

##### JSON

```
{
  "executionId": 2,
  "jobName": "ShellCommandRunnerJob"
}
```

#### Error Response

##### XML

```
<exceptionVo targetNamespace="">
<statusCode>500</statusCode>
<status>INTERNAL_SERVER_ERROR</status>
  <message>Internal Server Error</message>
<stackTrace></stackTrace> <!-- optional -->
</exceptionVo>
```

##### JSON

```
{
  "statusCode": "500",
  "Status": "INTERNAL_SERVER_ERROR",
  "Message": "Internal Server Error",
  "stackTrace": ""
}
```

### Check Status of a Job

This endpoint returns the status of a job using job name and execution id.

**Path:** /batch/jobs/{jobName}/<jobExecutionId>

HTTP Method: GET

## Inputs

jobName as path parameter

jobExecutionId as path parameter

### Sample Request

```
http://localhost:7001/jos-batch-job-admin/resources/batch/jobs/ShellCommandRunnerJob/1
```

### Successful Response

#### XML

```
<jobInstanceExecutionsVo targetNamespace="">
  <jobName>ShellCommandRunnerJob</jobName>
<jobInstanceId>1</jobInstanceId>
<resource>http://localhost:7001/jos-batch-job-admin/resources/batch/jobs/ShellCommandRunnerJob/1</resource>
<jobInstanceExecutionVo>
  <executionId>1</executionId>
<executionStatus>COMPLETED</executionStatus>
<executionStartTime>2016-07-11 15:45:27.356</executionStartTime>
<executionDuration>10</executionDuration>
</jobInstanceExecutionVo>
</jobInstanceExecutionsVo>
```

#### JSON

```
{
  "jobName": "ShellComandRunnerJob",
  "jobInstanceId": 1,
  "resource": "http://localhost:7001/jos-batch-job-admin/resources/batch/jobs/ShellCommandRunnerJob/1",
  ["jobInstanceExecutionVo": {
    "executionId": 1,
    "executionStatus": "COMPLETED",
    "executionStartTime": "2016-07-11 15:45:27.356",
    "executionDuration": "10"
  }]
}
```

## Error Response

#### XML

```
<exceptionVo targetNamespace="">
  <statusCode>503</statusCode>
<status>SERVICE_UNAVAILABLE</status>
  <message>Service Unavailable</message>
  <stackTrace></stackTrace> <!-- optional -->
</exceptionVo>
```

#### JSON

```
{
  "statusCode": "503",
  "Status": "SERVICE_UNAVAILABLE",
  "Message": "Service Unavailable",
  "stackTrace": ""
}
```

## System Setting Service

See Appendix D for details about System Setting Service.

## Job Admin UI

Job Admin UI provides functionality to manage and monitor jobs. The screenshots for Job Admin UI can be found in Appendix G.

## Best Practices for jobs

- Use Batchlet if job needs to run a script or program that resides locally.
- Use ItemReader, ItemWriter and so on, if job needs more control over the processing of data. Chunk processing allows data to be processed in chunks and if job fails, it can only process the remaining chunks during restart.
- Use job parameters to dynamically pass parameters to a job.
- Use PartitionMapper to specify number of partitions and threads for chunk processing so that data is processed concurrently.

## Job Admin Security

Both Job Admin UI and REST Services are secured with SSL and basic authentication. The below mentioned roles are defined to restrict access to operations in Job Admin.

- JobAdminRole
- JobOperatorRole
- JobMonitorRole

Batch jobs can be run from Job Admin UI or through Batch REST service. Here are the operations that can be performed by the users based on their role

Function	Admin Role	Operator Role	Monitor Role
Edit configuration from UI	Yes	No	No
Create/update/delete system options	Yes	No	No
Create/update/delete system credentials	Yes	No	No
View credentials	Yes	No	No
Run Jobs	Yes	Yes	No
Monitor jobs	Yes	Yes	Yes

## Job Admin Customization

During the deployment of Job Admin, seed data gets loaded to various tables. Seed data files are located in `jos-<app>-home/setup-data/dml` folder. If seed data is changed, Job Admin need to be reinstalled and redeployed. For loading seed data again during the redeployment, `LOADSEEDDATA` flag in `BDI_SYSTEM_OPTIONS` table need to be set to `TRUE`.

## Job Admin Troubleshooting

This section describes the Job Admin errors and its troubleshooting.

### Deployment Error

**Issue:** Job Admin deployment can run into this error if database credentials are invalid:



```
Caught: javax.management.RuntimeMBeanException: java.lang.RuntimeException:
weblogic.management.provider.EditFailedException: java.lang.NullPointerException
```

```
javax.management.RuntimeMBeanException: java.lang.RuntimeException:
weblogic.management.provider.EditFailedException: java.lang.NullPointerException
```

```
at weblogic.utils.StackTraceDisabled.unknownMethod()
```

```
Caused by: java.lang.RuntimeException:
```

```
weblogic.management.provider.EditFailedException: java.lang.NullPointerException
```

1 more

```
Caused by: weblogic.management.provider.EditFailedException:
```

```
java.lang.NullPointerException
```

1 more

```
Caused by: java.lang.NullPointerException
```

1 more

#### **Solution:**

Undo all changes in the Weblogic domain session. Redeploy app with setting up new credentials and verify deployment is successful.

## Runtime WSMException

**Issue:** Log files contain this exception:

```
oracle.wsm.common.sdk.WSMException: WSM-07620 : Agent cannot enforce policies
due to either failure in retrieving policies or error in validations, detail= "WSM-02557 The
documents required to configure the Oracle Web Services Manager runtime have not
been retrieved from the Policy Manager application (wsm-pm), possibly because the
application is not running or has not been deployed in the environment. The query
"&(@appliesTo~="REST-CLIENT()")(policysets:global:%)" is queued for later retrieval.
```

**Solution:** Follow BDI Installation guide, and verify WSM- policy manager is configured for admin server URL.

Open weblogic domain console and Target wsm-pm app to Admin Server. Bounce Admin server and verify wsm-pm app is in Active State.

#### **Job Fails and Job Admin Log Files Contain No Details of the Failure**

**Issue:** A job fails and the Job Admin log files contain no evidence of or details about the failure.

**Solution:** Take a look at the WebLogic Server log files to identify the root cause of the job failure. One example of this is improper data source configuration.

Job admin UI throwing error: Job XML not found

**Issue:** Log files contain this exception:

```
Caused By: javax.ejb.EJBException: EJB Exception: : java.lang.RuntimeException: Could
not find jobName(ShellCommandRunnerBatchlet) xml file. You may have renamed the
job file or your job repository has more jobs than your application. To resolve the issue
either delete the job repository or add the correct job xml file to the app.
```

**Solution:** The job has been deleted from the jos-job-home before redeployment. Either add the missing Job xml or Delete the execution records of the Job from batch database.

#### **Job admin UI throwing error: Table or view doesn't exists**

**Issue:** Log files contain this exception:

```
<Error> <javax.enterprise.resource.webcontainer.jsf.application> <BEA-000000> <Error
Rendering View[/index.xhtml] javax.el.ELException: /index.xhtml @15,84
```

value="#{batchSummaryRequestBean.refreshPage}": javax.ejb.EJBException: EJB Exception: : com.ibm.jbatch.container.exception.PersistenceException: java.sql.SQLException: ORA-00942: table or view does not exist

**Solution:** BatchInfrastructure database is not pointing to a valid schema, Check if the schema has the following tables: CHECKPOINTDATA, STEPEXECUTIONINSTANCEDATA, STEPSTATUS, EXECUTIONINSTANCEDATA, JOBINSTANCEDATA, JOBSTATUS. If not then run the DDL

```
$Oracle_Home/oracle_common/common/sql/wlsservices/batch/oracle/jbatch.sql
```

### **IO exception or permissions issue on running a shell runner job**

**Issue:** java.io.IOException: Cannot run program "./TestShell1.sh" (in directory "/u00/webadmin/16.1.0/Scripts"): error=13, Permission denied

**Solution:** Check if the script the job is running is present in the specified location or not. Check If the required permissions are provided for running the script.

### **Missing Credentials Access permission**

**Issue:** Caused by: java.lang.RuntimeException: Cannot get the Credential Map with the specified appLevelKeyPartitionName(DEFAULT\_KEY\_PARTITION\_NAME).at com.oracle.retail.integration.common.security.credential.CredentialStoreManager.getAllUserNameAliases(CredentialStoreManager.java:1171) ~[retail-public-security-api-16.1.0.jar:?]Caused by: java.security.AccessControlException: access denied ("oracle.security.jps.service.credstore.CredentialAccessPermission" "context=SYSTEM,mapName=DEFAULT\_KEY\_PARTITION\_NAME,keyName=" "read")

**Solution:** Verify weblogic.policy file, credential access permissions should be added for the domain where the apps are deployed. Add the permissions and restart the Admin and managed server.

## **Missing system credentials**

### **Issue:**

Caused by: java.lang.IllegalArgumentException: alias(processCallbackServiceUrlUserAlias) not found in the credential store.

### **Solution:**

Add System Credentials using UI or REST service.

## **Missing system options**

**Issue:** 2017-03-31T02:49:42,628 [Thread-132] DEBUG Logger\$debug - Starting job: null/resources/batch/jobs/DiffGrp\_Fnd\_ExtractorJob 2017-03-31T02:49:42,658 [Thread-132] ERROR Logger\$error\$0 - Error calling activity. java.lang.NullPointerException: Cannot invoke method getBytes() on null object

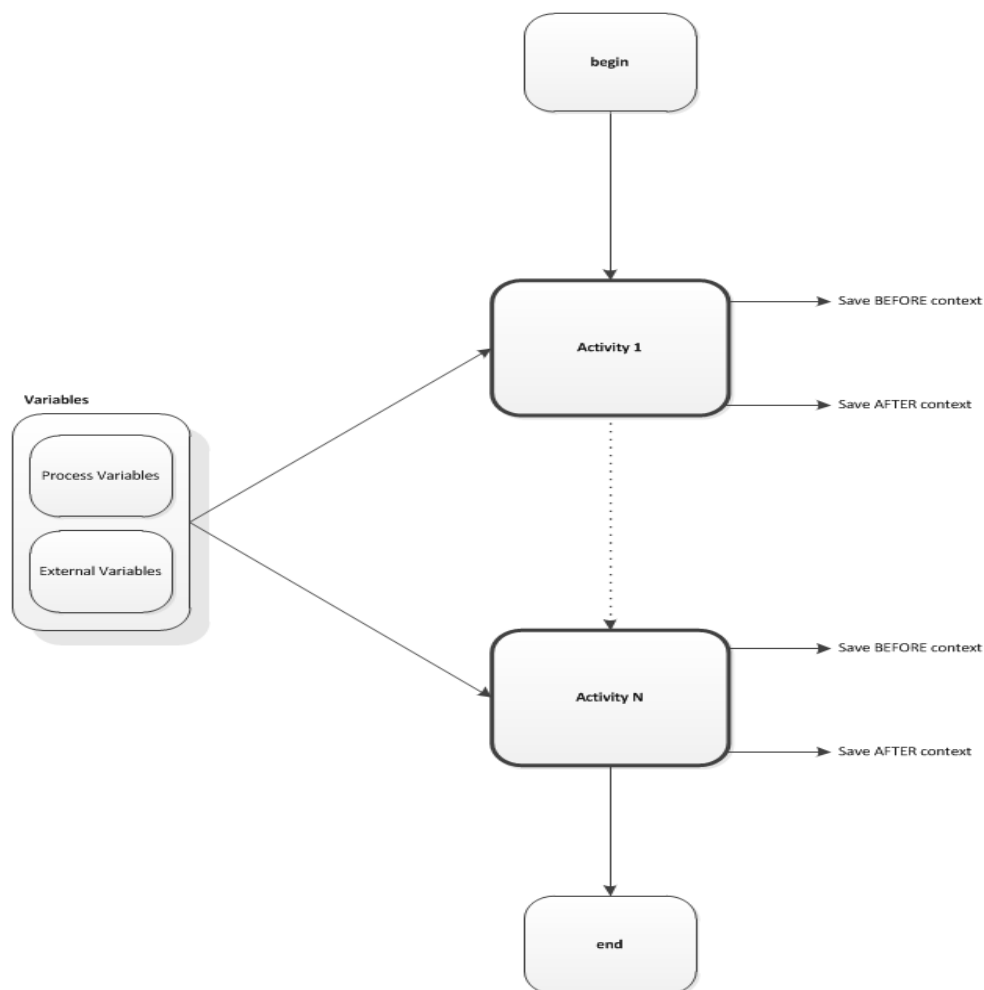
**Solution:** Add System Options using UI or ReST service.

## JOS Process Flows

A process flow is a composition of one or more activities. It is written in a DSL script that contains all the activities that makes up a process from start to finish.

A process flow encapsulates a sequence of activities. An activity can be synchronous or asynchronous. In JOS some of these activities may be invocations of batch jobs.

### Process Flow



## Process Flow Concepts

### DSL (Domain Specific Language)

Process flow definition is specified in a Domain Specific Language (DSL) built on the top of Groovy. Since Groovy is build on the top of Java Virtual Machine (JVM) Groovy can understand Java and Groovy language constructs. Hence the process flow DSL can understand the DSL, Groovy and Java language constructs.

A process is a list of activities. “begin”, “end” and “activity” are the main DSL keywords used in process flow definition. These are described in detail below.

## Begin Activity

“Begin” activity in the process flow definition appears as the first activity. There should be only one “begin” activity. This activity is intended to be the one used for any initialization needed for the process flow.

## Activity

Activity has two parts. Name and Action. Name attribute is mandatory and should be used to give a unique name for the activity.

Action section is where the executable code should reside. Any Groovy or Java code can be coded in this section.

There can be one or more Activities in a process.

## End Activity

“End” activity in process flow definition appears as the last activity. There should be only one “end” activity. This activity is intended to be the one used for any finalization needed for the process flow.

## Process Variables

Variables used between activities can be created and stored in the *processVariables* map. The process engine also uses some of the variables for its own working in the process variable map. These variables are prefixed with “bdi\_internal\_”. These variables must not be modified inside DSL code.

Here is how you can use the process variable map for your own use.

```
// Set Variable
processVariables["VariableName"] = "Some Value"
// Use a variable value
def anotherVariable = processVariables["VariableName"]
```

External Variables

Some of the system level configuration values are available in the *externalVariables* map. These values are read-only. The process flow DSL can use these values, but should not attempt to change it.

For Example;

```
externalVariables["rxmJobAdminBaseUrlUserAlias"]
```

## Process Flow DSL

### Process Flow DSL characteristics

- Every process flow must have a name. The process flow name must match with the filename that the process flow is saved into.
- Process flows are written in a DSL and saved as .flo files.
- Process flow is made up of two special activities called “begin” and “end” and bunch of user defined activity nodes.
- “begin” and “end” activity will always run.

- User defined activity may or may not run based on "SKIP" or *moveTo* logic.
- Every user defined activity must have a unique name within a process flow.
- The activity names are used to transfer control from one activity to another. Jumping to an activity is possible using *moveTo* function.
- Every activity has an "action" block that does the real work. Small amount of Groovy/Java code can be put inside the action block.
- Local variables can be defined within the action block.
- Process variables are defined on top and are accessible to all activities within the process.
- There are few implicit variables, like \$activityName, \$name.
- Errors can be thrown using "error <some message>" function.
- Built-in Conditional branching, looping, error handling.
- Predefined functions for common tasks to reduce boilerplate code.
- Built in REST service DSL to be able to call service with just one line.
- Services available to start/restart/monitor process flows programmatically.
- Can handle chaining of Process Flows.
- Service Credential management framework built in
- Hybrid Cloud ready
- Built in activity SKIP functionality
- Built in activity HOLD and RELEASE functionality
- Built in SPLIT and JOIN functionality between process flows
  - o SPLIT - one to many
  - o JOIN - many to one

## DSL Keywords

DSL Keywords	Description
process	Identifies the process flow. Only one keyword in a process flow.
name	Used for naming processes and activities
var	Used for initializing process variables
begin	Begin activity block is the first activity in the DSL. It is mandatory and can be used for initialization.
activity	The executable component of the process flow. A process flow is composed of many activities.
action	Action section is where the executable code should reside. Any Groovy or Java code can be coded in this section.
on "okay" moveTo	Use these keywords inside an activity to move to another activity
on "error" moveTo	Use these keywords inside an activity to move to error activity

## Process Flow API

DSL API	USAGE	Description
startOrRestartJob(def baseUrl, String jobName, String credentials)	startOrRestartJob(externalVariables["url"], "JobAbc", externalVariables["urlUserAlias"])	Method to start or restart a job in Job Admin. This method sends a POST request to a REST end point in Job Admin
waitForJobCompletedOrFailed(def targetActivity, def url, String credentials, int waitMinutes=1)	waitForJobCompletedOrFailed("JobAbcActivity", externalVariables["url"] + "/resources/batch/jobs/JobAbc/" + processVariables["jobExecutionId"], externalVariables["urlUserAlias"])	Method to wait for job to be completed or failed. This method checks the status of the job and waits until status is COMPLETED or FAILED.
waitForProcessInstancesToReachStatus(def processInstanceList, def targetStatus=PROCESS_COMPLETED, def logicalAndOrOr = LOGICAL_AND, int waitMinutes=1)	waitForProcessInstancesToReachStatus(["P~1", "Q~1"], PROCESS_COMPLETED, LOGICAL_OR)	Method to wait for other process instances to reach a status.
waitForProcessNamesToReachStatus(Map, processNameToNumberOfExecutionsAfterStartMarkerTime, LocalDateTime startMarkerTime, def targetStatus = PROCESS_COMPLETED, def logicalAndOrOr = LOGICAL_AND, def whichExecutionStatus = LAST_EXECUTION_STATUS, int waitMinutes = 1)	waitForProcessNamesToReachStatus([P:3, Q:3, R:3], now().minusDays(1), PROCESS_COMPLETED, LOGICAL_AND, LAST_EXECUTION_STATUS)	Method to wait for processes with names to reach a status.
persistGlobalUserData(String key, String value)	persistGlobalUserData("key", "value")	Method to persist data to be shared with other processes. Persists key value pairs in BDI_SYSTEM_OPTIONS table.
String findGlobalUserData(String key)	findGlobalUserData("key")	Gets value from BDI_SYSTEM_OPTIONS table for given key.
Map findAllGlobalUserData(String key)	findAllGlobalUserData()	Returns a Map with all user data.
removeGlobalUserData(String key)	removeGlobalUserData("key")	Removes data for given key.
error	error "report my error"	Generate an error condition and jump to the end activity. Process will be marked as failed.
POST	POST[externalVariables.url]^externalVariables.urlUserAlias	Method to make a POST call to a url.
GET	GET[externalVariables.url]^externalVariables.urlUserAlias	Method to make a GET call to a url.
DELETE	DELETE[externalVariables.url]^externalVariables.urlUserAlias	Method to make a DELETE call to a url.

DSL API	USAGE	Description
log.info log.debug	log.debug "Activity Name: \$activityName"	Adds information to log file

## Process Flow Variables

Variables	Implicit or Explicit	Usage Examples	Description
externalVariables	Implicit variables	def myVar = externalVariables['myKey']	These are Global variables that apply to all process flows. It comes from System Options table. Installation specific key values will be here.
processVariables	Implicit variables	var([ "myVar1":"prq", "myVar2":"xyz", "myVar3":"mno" ])  //get value def aVar = processVariables['myVar1']  //put new value processVariables['myVar2'] = "abc"	These are process level variables that can be shared by all activities. Process variables are automatically persisted. Restart of a process recovers the process variables to the right value where it left off in the previous run. These are the most common variables you should use. Process variables must be declared using the var key word.
Local variables	Explicit variables	action{  def a = "xyz" def i = 7  i++  }	Any variables can be created with the action block and used as local variables. Local variables defined in one activity is not accessible in another activity.
Global external variables	Explicit variables	persistGlobalUserData("key1", "value1")  def xyz = findGlobalUserData("key1")  removeGlobalUserData("key1")	For inter process dynamic variable sharing one can persist new variable to DB.
activityName	Implicit variables	println "My activity is \${activityName}"	Current activity name.
name	Implicit variables	Println "My process name is \${processName}"	Current process name.

## Process Flow Instrumentation

When the process engine executes the process flow, the before and after snapshots of the Activity is recorded in the process schema.

The information is reported through Process Flow Admin application. Process Flow Admin is a Web application that provides GUI to manage workflows of tasks. This is useful for tracking the process flows as well as troubleshooting. The snapshots also help in case of restarting a failed process. From the schema, the process engine can recreate the context to execute a restart and can resume execution from the activity that failed in the previous run.

## Sub Processes

One process may invoke one or more processes asynchronously. All the processes may run at the same time.

In order to identify these subprocesses they are named accordingly. Once invoked, the main process has no control over the sub processes. Each of the process will run in the same way as they are invoked independently.

## Process Schema

The process instrumentation captures the state of the process at the beginning and end of each activity. This information is persisted into the process schema. For each activity there will be two records, one for before activity and the other for after activity.

Table Name	Description
BDI_PROCESS_DEFINITION	This table stores all the process flow definitions. It is loaded at deployment time.
BDI_PROCESS_EXEC_INSTANCE	This table tracks all the process flow executions. There is a row for each process flow execution.
BDI_ACTIVITY_EXEC_INSTANCE	This table tracks all the activity executions. There are 2 rows for each activity execution. One to store the before context and one to store after context
BDI_ACTIVITY_DYNAMIC_CONFIG	This table stores the user runtime choices like SKIP, HOLD etc at activity level
BDI_SYSTEM_OPTIONS	This table has all the system level information like URLs, credential aliases and so on.

## Process Restart

When the activities within a process flow fails, the process status is marked as failed. A failed (or stopped) process flow can be restarted. If there are multiple failed processes, only the latest failed instance can be restarted.

---

**Note:** that, restart is for an already run and failed instance. This is different from running a new instance of the process flow.

---

When a process flow is restarted, the system knows the activity that failed (or stopped) in the previous run. During restart, the process engine will skip all the activities prior to the failed activity. It will restore the context for the activity and resume execution at the failed activity.



Process flow execution does not keep the activity history at restart. It will overwrite the activity records on restart.

## Statuses

Each activity instance and the process instance maintain the status of execution in the process schema. Following are the possible values for Activities and Process.

At the “begin” activity, process is marked as PROCESS\_STARTED. If any activity fails, the process is marked as PROCESS\_FAILED. After the “end” action is completed, the process is marked PROCESS\_COMPLETED. A complete list of process flow status are:

- PROCESS\_STARTED
- PROCESS\_FAILED
- PROCESS\_COMPLETED
- PROCESS\_STOPPING
- PROCESS\_STOPPED

Similar to process statuses, each activity has also a status. There values are :

- ACTIVITY\_STARTED
- ACTIVITY\_FAILED
- ACTIVITY\_COMPLETED
- ACTIVITY\_WAITING\_DUE\_TO\_HOLD\_STARTED
- ACTIVITY\_WAITING\_DUE\_TO\_HOLD\_COMPLETED
- ACTIVITY\_WAITING\_DUE\_TO\_JOIN\_STARTED
- ACTIVITY\_WAITING\_DUE\_TO\_JOIN\_COMPLETED
- ACTIVITY\_SKIPPED
- ACTIVITY\_STOPPING
- ACTIVITY\_STOPPED

All the runtime status are persisted in the process schema at runtime when the DSL is executed.

## Steps for implementing a JOS Flow

1. Download “JosProcessFlow<version>ForAll16.x.xApps\_eng\_ga.zip” and unzip the file.
2. Create a process flow DSL file that stitches the jobs. DSL is groovy based and groovy or java code can be used with in “action” block. See the sample DSL below.
3. Copy DSL files to “jos-process-home/setup-data/dsl/flows-in-scope” folder.
4. Run the deployer script in “jos-process-home/bin” folder to deploy JOS Process flow.

## Activity Features

### Skip Activity

Activities in a process flow can be skipped by setting the skip activity flag through the Process Flow Configurations screen or REST endpoint. Skip flag can be set to expire based on date and time. If expiry date is not provided, then that activity will be skipped until skip flag is removed. When an activity is set to skip, process flow engine skips that activity and runs the next activity in the flow.

## Process Flow Configurations Screen

The screenshot shows the 'Process Flow Configurations' tab in a web application. The title bar indicates 'Process Flow Executions For Diff\_Fnd\_ProcessFlow\_From\_RMS:'. Below this is a table with the following columns: Activity Name, Action, Action Expiry Date and Time, Comments, and Action. The table contains several rows for different activities, each with 'Skip Activity' and 'Hold Activity' checkboxes, an expiry date field, a comments field, and a 'Save' button.

Activity Name	Action	Action Expiry Date and Time	Comments	Action
begin				
Diff_Fnd_ExtractorActivity	<input type="checkbox"/> Skip Activity <input type="checkbox"/> Hold Activity		processadmin	Save
Diff_Fnd_ExtractorStatusActivity	<input type="checkbox"/> Skip Activity <input type="checkbox"/> Hold Activity			Save
Diff_Fnd_GetDataSetIdActivity	<input type="checkbox"/> Skip Activity <input type="checkbox"/> Hold Activity			Save
Diff_Fnd_DownloaderAndTransporterActivity	<input type="checkbox"/> Skip Activity <input type="checkbox"/> Hold Activity			Save
Diff_Fnd_CheckDownloaderAndTransporterStatusActivity	<input type="checkbox"/> Skip Activity <input type="checkbox"/> Hold Activity		processadmin	Save
Diff_Fnd_UploaderActivity	<input type="checkbox"/> Skip Activity <input type="checkbox"/> Hold Activity			Save

### REST endpoint to set the skip activity flag

/batch/processes/<processName>/activities/<activityName>?skip=true

### Hold/Release Activity

Activities in a process flow can be paused by setting the hold activity flag through the Process Flow Configurations screen or REST endpoint. Hold flag can be set to expire based on date and time. If expiry date is not provided, then that activity will be paused until hold flag is removed. When an activity is set to hold, process flow engine waits on that activity until hold flag is removed or time expired.

### REST endpoint to set the hold activity flag

/batch/processes/<processName>/activities/<activityName>?hold=true

## Callback Service

Process Flow engine can be configured to call a rest service at each activity. This is useful if the process flow is invoked by an external system (typically a workflow system) and the system wants to be informed of the progress of each activity. This callback can be configured declaratively or programmatically as needed.

The external system will have to implement the CallBack Service that will allow it to receive information from the JOS process flow. The external system can call the the process flow passing the context information as process flow parameters. The process flow will pass the information back when it makes the CallBack Service call.

### How to start Process Flow with input parameters

To start a jos process flow user has to make a REST service call to URL

(http://<host>:<port>/bdi-process-flow/resources/batch/processes/operator/<processName> ) .

The call must be a POST call to the URL.

The process flow start call accepts http query parameters. The format of the query parameters are as follows:

```
http://localhost:7001/bdi-process-
flow/resources/batch/processes/<ProcessName>?processParameters=callerId=<value1>,c
orrelationId=<value2>,callBackServiceDataDetail.<name1>=<value3>,callBackServiceDa
taDetail.<name2>=<value4>
```

Spaces are not allowed in query parameters and must be separated by commas.

For Example;

```
http://localhost:7001/bdi-process-
flow/resources/batch/processes/Abc_Process?processParameters=callerId=123,correlat
ionId=abc,callBackServiceDataDetail.def=xyz,callBackServiceDataDetail.abc=123
```

The following are the context information that need to be passed to JOS process flow from calling system.

1. **callerId.** CallerId parameter is used to identify the invoker of process flow. In this case it is CAWA.
2. **correlationId.** Correlation id is the main identifier used by the calling system (CAWA) to tie the process flow Start call to the eventual CallBack Service call.
3. **callBackServiceDataDetail.<name>=** These are additional key value pairs that may be required in future as required by the caller.

All of the above parameters are optional. However, if the context is not passed the caller may not be able to associate the invocation with the callback.

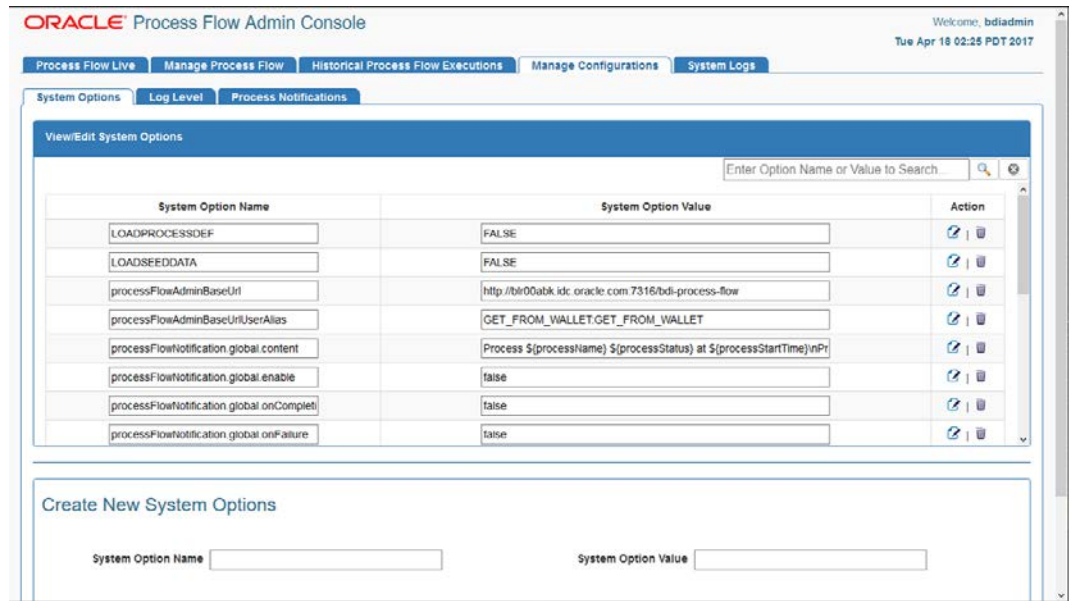
## Call back from Processflow

A new method (`invokeCallBackService`) is available for Process Flow DSL that will allow process flow to call an external service. This service has following features

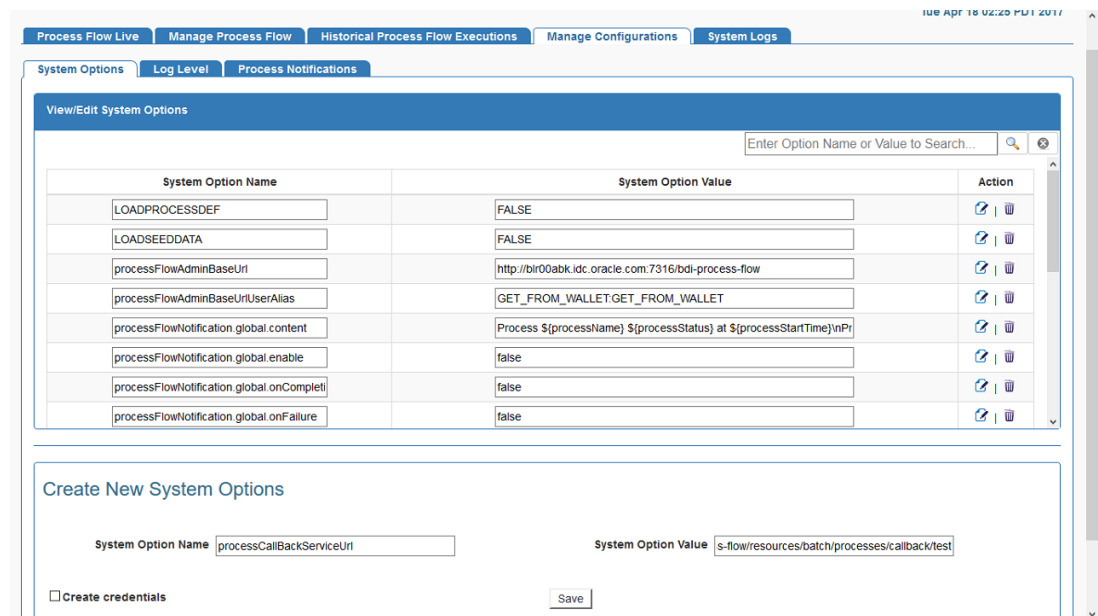
- The method internally invokes a REST call to the provided URL
- The method uses Basic Authentication for the rest call. The credentials for the method call must be available in the process flow.
- The payload sent from process flow to the invoking application (CAWA) follows the contract as shown in the example in the next section. All of the values, other than `keyValueEntryVo`, are populated by the Process Flow engine. The DSL writer can modify the `keyValueEntryVo` before the callback to pass any custom value from the DSL to invoking application (CAWA)
- The result of the callback REST service (in CAWA) must be a String value.
- If the callback service invocation fails for any reason (such as a network issue), the process flow activity fails and the process flow is marked as failed.

## How to invoke the Callback Service declaratively

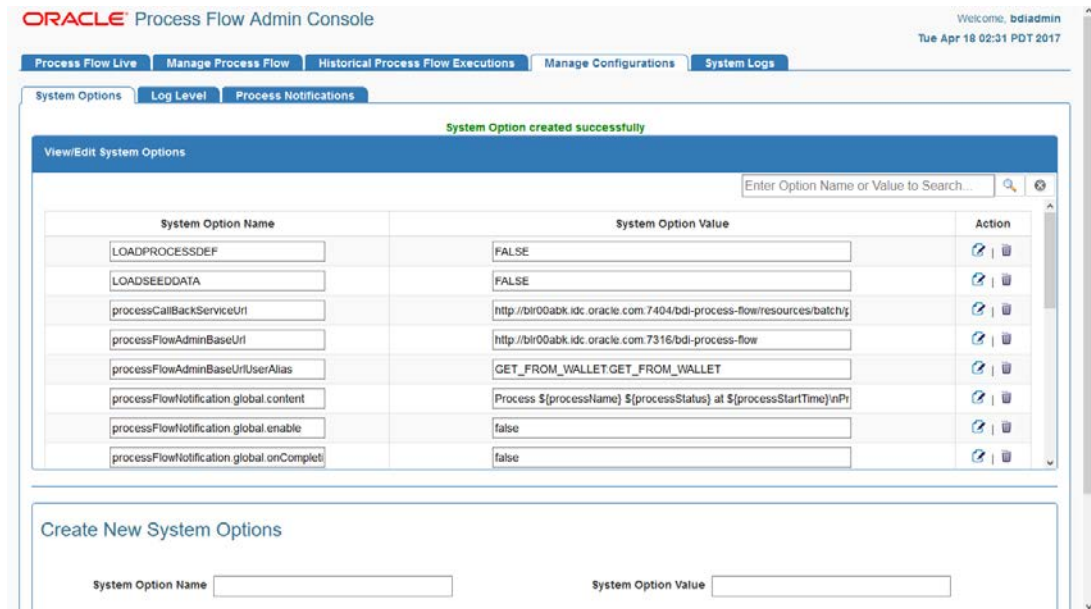
- Setup the callback URL in processflow system options. To configure a callback URL you should add system options like `<serviceName>CallBackServiceUrl`, for eg., `processCallBackServiceUrl`.
1. In Process Flow admin console, navigate to Manage Configurations tab and System Options sub-tab.



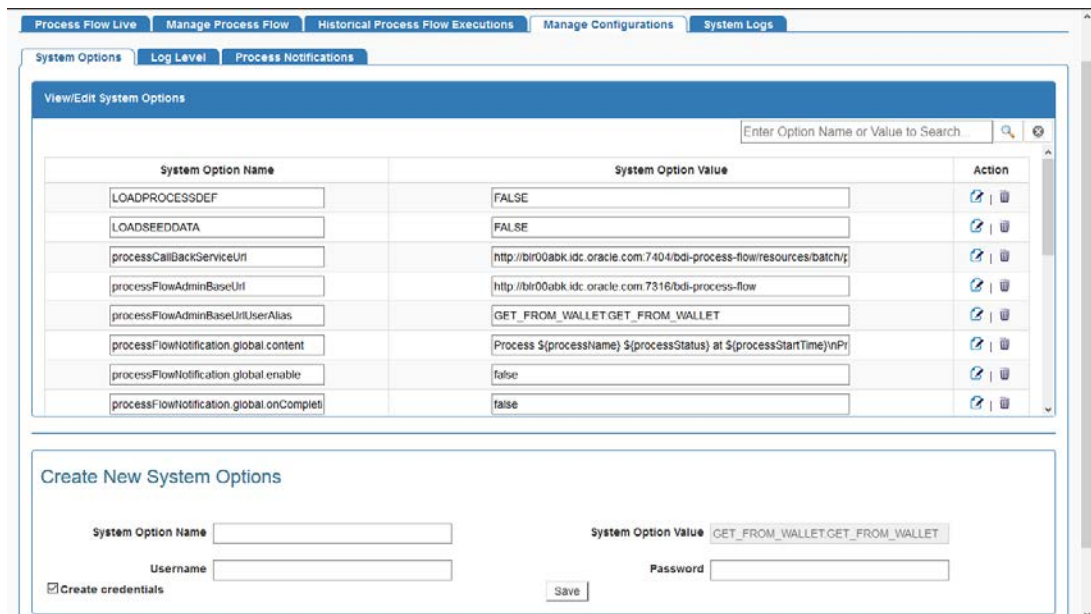
2. Scroll down to Create New System Options, enter **System Option Name** and **System Option Value**. Url should be a valid ReST Service.



3. Click **Save**.



4. Setup the callback URL credential alias in process flow. To add callback URL credential alias you should add credential alias like <serviceName>CallBackServiceUrlUserAlias, for eg., processCallBackServiceUrlUserAlias.
5. In the Create New System Options section, select **Create Credentials** checkbox.



6. Enter **System Option Name**, **Username** and **Password** for the URL provided in the previous step. If the System Option Name for the URL is **processCallBackServiceUrl** then System option name for credential should be **processCallBackServiceUrlUserAlias**.

The screenshot shows the 'View/Edit System Options' page in the Oracle Process Flow Admin Console. At the top, there are navigation tabs: 'Process Flow Live', 'Manage Process Flow', 'Historical Process Flow Executions', 'Manage Configurations', and 'System Logs'. Below these are sub-tabs: 'System Options', 'Log Level', and 'Process Notifications'. The main content area is titled 'View/Edit System Options' and contains a search bar and a table of system options.

System Option Name	System Option Value	Action
LOADPROCESSDEF	FALSE	<a href="#">Edit</a>   <a href="#">Delete</a>
LOADSEEDDATA	FALSE	<a href="#">Edit</a>   <a href="#">Delete</a>
processCallBackServiceUrl	http://b1r00abk.idc.oracle.com:7404/bdi-process-flow/resources/batch/j	<a href="#">Edit</a>   <a href="#">Delete</a>
processFlowAdminBaseUrl	http://b1r00abk.idc.oracle.com:7316/bdi-process-flow	<a href="#">Edit</a>   <a href="#">Delete</a>
processFlowAdminBaseUrlUserAlias	GET_FROM_WALLET.GET_FROM_WALLET	<a href="#">Edit</a>   <a href="#">Delete</a>
processFlowNotification.global.content	Process \${processName} \${processStatus} at \${processStartTime} in Pr	<a href="#">Edit</a>   <a href="#">Delete</a>
processFlowNotification.global.enable	false	<a href="#">Edit</a>   <a href="#">Delete</a>
processFlowNotification.global.onComplete	false	<a href="#">Edit</a>   <a href="#">Delete</a>

Below the table is the 'Create New System Options' form. It includes fields for 'System Option Name' (processCallBackServiceUrlUserAlias), 'System Option Value' (GET\_FROM\_WALLET.GET\_FROM\_WALLET), 'Username' (processadmin), and 'Password' (masked). There is a checked checkbox for 'Create credentials' and a 'Save' button.

7. Click Save.

The screenshot shows the 'View/Edit System Options' page after a successful operation. At the top right, it says 'Welcome: bdiadmin Tue Apr 18 02:46 PDT 2017'. A green message banner reads 'System Option and Credential created successfully'. The table of system options is updated, showing a new entry for 'processCallBackServiceUrlUserAlias' with the value 'GET\_FROM\_WALLET.GET\_FROM\_WALLET'. The 'Create New System Options' form is now empty.

**Note:** Credentials created through UI are available after server restart, but after redeployment of the application credentials have to be created again.

8. Navigate to Manage Process Flow tab and select process flow, go to Process Flow Configurations sub-tab.
9. Select **Callback** checkbox for the activities you want callback to be enabled. Select **Callback URL** from drop down list.

Diff_Fnd_ProcessFlow_From_RMS	Diff	RXM	no-split	Fri Apr 14 00:15:00 PDT 2017	2017		
DiffGrp_Fnd_ProcessFlow_From_RMS	DiffGrp	RXM	no-split	Fri Apr 14 00:20:00 PDT 2017			
InvAvailWh_Tx_ProcessFlow_From_RMS	InvAvailWh	RXM	no-split	Fri Apr 14 00:35:00 PDT 2017			
ItemHdr_Fnd_ProcessFlow_From_RMS	ItemHdr	RXM	no-split	Fri Apr 14 00:40:00 PDT 2017			
ItemImage_Fnd_ProcessFlow_From_RMS	ItemImage	RXM	no-split	Fri Apr 14 00:45:01 PDT 2017			
ItemLoc_Fnd_ProcessFlow_From_RMS	ItemLoc	RXM	no-split	Fri Apr 14 00:50:00 PDT 2017			
MerchHier_Fnd_ProcessFlow_From_RMS	MerchHier	RXM	no-split	Fri Apr 14 01:20:00 PDT 2017			
OrchHier_Fnd_ProcessFlow_From_RMS	OrchHier	RXM	no-split	Fri Apr 14 01:25:00 PDT 2017			

Process Flow Executions | Process Flow Configurations | Launch Process Flow | Process Flow Details

Process Flow Executions For Diff\_Fnd\_ProcessFlow\_From\_RMS:

JTA Timeout - 12 Hours 0 Minutes 0 Seconds

Activity Name	Action	Action Expiration	Call Back	Call Back Service URL	Comments
begin			<input type="checkbox"/>	Select URL	
Diff_Fnd_ExtractorActivity	<input type="checkbox"/> Skip <input type="checkbox"/> Hold		<input checked="" type="checkbox"/>	processCallBackServiceUrl	
Diff_Fnd_ExtractorStatusActivity	<input type="checkbox"/> Skip <input type="checkbox"/> Hold		<input type="checkbox"/>	Select URL	
Diff_Fnd_GetDataSetIdActivity	<input type="checkbox"/> Skip <input type="checkbox"/> Hold		<input type="checkbox"/>	Select URL	
Diff_Fnd_DownloaderAndTransporterActivity	<input type="checkbox"/> Skip <input type="checkbox"/> Hold		<input type="checkbox"/>	Select URL	

10. Click Save.

Diff_Fnd_ProcessFlow_From_RMS	Diff	RXM	no-split	Fri Apr 14 00:15:00 PDT 2017	Mon Apr 03 04:34:57 PDT 2017		
DiffGrp_Fnd_ProcessFlow_From_RMS	DiffGrp	RXM	no-split	Fri Apr 14 00:20:00 PDT 2017			
InvAvailWh_Tx_ProcessFlow_From_RMS	InvAvailWh	RXM	no-split	Fri Apr 14 00:35:00 PDT 2017			
ItemHdr_Fnd_ProcessFlow_From_RMS	ItemHdr	RXM	no-split	Fri Apr 14 00:40:00 PDT 2017			
ItemImage_Fnd_ProcessFlow_From_RMS	ItemImage	RXM	no-split	Fri Apr 14 00:45:01 PDT 2017			
ItemLoc_Fnd_ProcessFlow_From_RMS	ItemLoc	RXM	no-split	Fri Apr 14 00:50:00 PDT 2017			
MerchHier_Fnd_ProcessFlow_From_RMS	MerchHier	RXM	no-split	Fri Apr 14 01:20:00 PDT 2017			
OrchHier_Fnd_ProcessFlow_From_RMS	OrchHier	RXM	no-split	Fri Apr 14 01:25:00 PDT 2017			

Process Flow Executions | Process Flow Configurations | Launch Process Flow | Process Flow Details

Process Flow Executions For Diff\_Fnd\_ProcessFlow\_From\_RMS:

Activity configurations saved successfully for Process flow: Diff\_Fnd\_ProcessFlow\_From\_RMS

JTA Timeout - 12 Hours 0 Minutes 0 Seconds

Activity Name	Action	Action Expiration	Call Back	Call Back Service URL	Comments
begin			<input type="checkbox"/>	Select URL	
Diff_Fnd_ExtractorActivity	<input type="checkbox"/> Skip <input type="checkbox"/> Hold		<input checked="" type="checkbox"/>	processCallBackServiceUrl	
Diff_Fnd_ExtractorStatusActivity	<input type="checkbox"/> Skip <input type="checkbox"/> Hold		<input type="checkbox"/>	Select URL	
Diff_Fnd_GetDataSetIdActivity	<input type="checkbox"/> Skip <input type="checkbox"/> Hold		<input type="checkbox"/>	Select URL	
Diff_Fnd_DownloaderAndTransporterActivity	<input type="checkbox"/> Skip <input type="checkbox"/> Hold		<input type="checkbox"/>	Select URL	

**How to invoke the Callback Service programmatically**

From the Process Flow DSL activity, you can invoke the callback service as shown in the examples below. The callbackServiceUrl and callbackServiceUrlUserAlias property must be setup in the System Options inside process flow.

**Example 1: Short Form**

Add the following line inside JOS process flow activity.

```
def retValue =
invokeCallbackService(externalVariables.callBackServiceUrl,
externalVariables.callBackServiceUrlUserAlias)
```

### Example 2: Long Form

In the long form API the `callbackServiceData` is an implicit parameter that is automatically defined and user can update it with additional data inside an activity if they want.

Add the following line inside JOS process flow activity.

```
//optionally update some data
    callbackServiceData.keyValueEntryVo[0].key = "Some Key"
    callbackServiceData.keyValueEntryVo[0].value = "Some Value"

def retValue =
invokeCallbackService(externalVariables.callBackServiceUrl,
externalVariables.callBackServiceUrlUserAlias, callbackServiceData)
```

### Callback request Payload structure

The jos process flow will make a POST REST call to the `callbackServiceUrl` passing in the following payload. JSON is the default content type.

#### JSON Payload Contract

```
{
  "processName": "Abcdef_Process",
  "processExceutionId": "123456",
  "activityName": "Def_Activity",
  "activityExecutionId": "12345678",
  "callerId": "XYZ",
  "correlationId": "987654321",
  "keyValueEntryVo": [
    {
      "key": "abc",
      "value": "def"
    },
    {
      "key": "pqr",
      "value": "123"
    }
  ],
}
```

#### XML Payload Contract

```
<?xml version="1.0" encoding="UTF-8" ?>
<callbackServiceVo>
  <processName>Abcdef_Process</processName>
  <processExceutionId>123456</processExceutionId>
  <activityName>Def_Activity</activityName>
  <activityExecutionId>12345678</activityExecutionId>
  <callerId>XYZ</callerId>
  <correlationId>987654321</correlationId>
  <keyValueEntryVo>
    <key>abc</key>
    <value>def</value>
  </keyValueEntryVo>
  <keyValueEntryVo>
    <key>pqr</key>
    <value>123</value>
  </keyValueEntryVo>
</callbackServiceVo>
```



### Call Back Service Scenarios

Activity Type	Activity Action (Skip or Hold)	Callback behaviour (if callback enabled)	Activity Status sent by Callback	Activity Status if Callback fails
Any	None	Callback will be called after action part is complete	ACTIVITY_COMPLETE or ACTIVITY_FAILED according to the action part success or failure.	ACTIVITY_FAILED
	Skip	Callback will be called after action part is complete	ACTIVITY_SKIPPED	ACTIVITY_FAILED
	Hold	Callback will be called when hold is released and after the action part of the activity runs	ACTIVITY_COMPLETE or ACTIVITY_FAILED according to the action part success or failure.	ACTIVITY_FAILED
<b>Special Cases</b>				
startOrRestartJob Activity	None	Callback will be called as soon as the job start or restart call is complete	ACTIVITY_COMPLETE if the job was started or restarted successfully. ACTIVITY_FAILED if the job was not started or restarted successfully.	ACTIVITY_FAILED
waitForJobCompletedOrFailed	None	Callback will be called after the Job status has reached complete or failed	ACTIVITY_COMPLETE if the job completed successfully. ACTIVITY_FAILED if the job failed.	ACTIVITY_FAILED
<b>Restart Scenarios</b>				
startOrRestartJob Activity	None	Job will be started or restarted only if the Job was not started earlier or job failed. If the activity failed due to callback failure the job will not be started.	ACTIVITY_COMPLETE if the job was started or restarted successfully. ACTIVITY_FAILED if the job was not started or restarted successfully.	ACTIVITY_FAILED
waitForJobCompletedOrFailed	None	Callback will be called after checking the Job status, if it has reached complete or failed, otherwise process will wait for the job to reach complete or failed status.	ACTIVITY_COMPLETE if the job completed successfully. ACTIVITY_FAILED if the job failed.	ACTIVITY_FAILED

## Process Security

Process Flow Application uses basic authentication to access the system. The user must belong to BdiProcessAdminGroup or BdiProcessOperatorGroup or BdiProcessMonitorGroup to use the process flow REST services and process flow admin application.

There are two authorization roles designed for process flow application; Operator role and Admin Role. Admin role has permissions to use all the functions provided by process flow application. Operator Role has limited access compared to Admin, as identified in the table below. Monitor role has the least access permissions.

Service/Action	Monitor Role	Operator Role	Admin Role
Update Process DSL	No	No	Yes
Start/Restart Process	No	Yes	Yes
Skip/Hold/Release	No	Yes	Yes
All other services	Yes	Yes	Yes

## Troubleshooting

Since the process flow can be written in Groovy and DSL, it is prone to programmer’s mistakes. Any custom DSL must be properly tested before deploying. At present, the process flow engine can detect syntax errors only at runtime. So it is possible to load an incorrect process flow and fail during runtime.

At the end of an activity, the process engine invokes the next activity depending on the result of activity execution (The “moveTo” statement). If you have empty activities (possibly because you commented out the existing invocation statements), make sure the activity result is valid.

If any activity fails, the process is marked as failed. So in case of process failure, look at the activity details to find out which activity failed. Once the failed activity is identified, the process variables can be inspected to look for any issues. Next step would be to look at the logs, through the Process Flow Monitor application to see the details of the issue. Once the issue is fixed, either restart or a new run of the process flow can be used depending on the requirement.

### Process Flow Didn’t Start

Verify the logs, it could be due to the missing Credentials Access Permission, missing system credentials, missing system options or DSL parsing Error.

### Deleted Process Flow Still Listed in the UI

Deleting a process flow from jos-process-home doesn't deletes it from the process flow application, because the process flow application refers the database entries, so in order to delete a process flow from JOS Process Flow app, the script `DELETE_PROCESS_FLOW.sql(jos-process-home/setup-data/dml/)` has to be run in JOS Schema.

## Best Practices for Process Flow DSL

- Use naming conventions for process flows and activities in process flow so that they are easily identified. It is recommended that name of the process flow includes “Process” and the name of activities ends with “Activity”.
- Use built in “startOrRestartJob” method to start/restart job in Job Admin.
- Use built in “waitForJobCompletedOrFailed” method to wait until job is complete or failed.
- Access system options through “externalVariables”.

- Use “processVariables” to share variables between activities.
- Use built in “waitForProcessInstancesToReachStatus” to wait for other process instances.
- Use built in “waitForProcessNamesToReachStatus” to wait for other processes.
- It is recommended to use “flo” as extension for process flow DSL file.
- Use the built-in REST DSL to make rest calls.
- Organize process flows as hierarchical parent child flows where parent manages the child flows.
- Avoid using too many waitFor calls as active threads are getting blocked.



## Scheduler Overview

The Scheduler application (JOS) product suite assists in scheduling of batch processes to run at predefined configured intervals of times. A schedule determines when a job or a process or any program needs to be executed and the frequency of execution.

The Scheduler application runtime is based on container-managed Java EE timer service to execute the schedules and utilizes Oracle WebLogic Server's implementation and management of the timer service when deployed on WebLogic server.

Scheduler supports various schedules ranging from simple interval schedules such as hourly, daily, and so on, to advanced cron-like scheduling.

Scheduler currently supports calling of REST services.

The Scheduler Console (Admin UI) enables runtime monitoring and administration of schedules where user can view, create, edit, delete schedules, manually run a schedule, enable/disable schedule, set up notifications for schedules and so on.

## JOS Scheduler Features

- Scheduler is a Web application that provides GUI for managing schedule based workload.
- DSL based Schedule Action – Call process flows, run any local/remote programs.
- Run remote programs with REST calls.
- Externalized Schedule Definition and Schedule Actions. Easily import/export Schedule and Action definitions
- GUI to Create, Edit, Delete, Enable/Disable schedules
- Monitor schedule executions and logs
- Monitor schedule's progress and history
- Built in Email notification

## Scheduler Concepts

### Schedule Definition

A schedule definition comprises of details of a schedule such as Schedule Name, Schedule Group which indicates logical or functional grouping of schedules, and Schedule Description.

### Schedule Execution

A schedule execution is an instance of scheduled run of a schedule at the specified frequency.

### Schedule Types

A schedule can be an interval-based schedule or calendar-based schedule.

## Interval Schedules

An interval-based schedule is a schedule that repeats at fixed interval of time starting from a specific time. For example, hourly, daily, weekly, every 5 minutes and so on.

## Calendar Schedules

A calendar-based schedule is cron-type of schedule that specifies different times that the schedule runs. More complex schedules that can be specified as cron expression are defined as calendar-based schedules.

The following parameters define a calendar-based schedule, same as the parameters in a cron expression: Minutes, Hours, Day of Week, Day of Month and Month.

**Note:** that the Scheduler does not currently support Seconds and Year parameters in a calendar schedule.

## Scheduling Mechanisms

### Simple Scheduling

Simple schedules are predefined schedule frequencies that are available as options for the user to choose readily. The following are the simple schedules that the Scheduler supports.

- Hourly
- Daily
- Weekly
- Monthly
- Weekday [Mon-Friday]
- Weekend [Sat-Sunday]
- Saturday
- Sunday
- First day of every month
- Last day of every month
- One time only (run once), and
- User-specified frequency with interval in the units of:
  - minutes, hours, days or weeks.

### Advanced Scheduling

JOS Scheduler supports advanced scheduling which is cron-like scheduling. Calendar-based schedules that can be expressed in cron-format can be setup with the advanced scheduling capability of the Scheduler. Advanced scheduling is defined with the following parameters (similar to that of cron expression) and the corresponding range of values:

- Minutes : 0-59
- Hours : 0-23 (12:00 a.m. - 11:00 p.m.)
- Day of Week : Monday - Sunday
- Day of Month : 1-31
- Month : 1-12 (January - December)

If a schedule is created with multiple values for the above parameters, then the schedule will repeat at all those specified times.

## Schedule Frequency

The schedule frequency defines the frequency at which a schedule has to be repeated at the configured time and interval starting from a given point of time. The schedule frequency has the following parameters that determines when the schedule has to be run.

### Schedule Start Datetime

It specifies the start date and time when a particular schedule has to start executing.

For interval based schedules, this is the first time the schedule runs and then repeats based on the specified interval.

For example, a schedule with start datetime as 2016-08-15 10:00a.m. and repeat 'Daily' will first run at 2016-08-15 10:00 a.m. and next run at 2016-08-16 10:00 a.m. and so on.

For calendar schedules (cron schedules), this defines the time from when the schedule will become effective and starts executing based on the frequency. So it is not necessarily the first run of the schedule, though it very well may be.

For example, a schedule with start datetime as 2016-08-15 10:00 a.m. (which is a Monday) but repeat every Thursday, will first run at 2016-08-18 10:00 a.m. (Thursday) and subsequently next run at 2016-08-25 10:00 a.m. (Thursday) and so on.

So the Start Datetime here signifies the datetime the schedule becomes effective and that it will not run before that datetime. However, here the Start Datetime can very well be specified as 2016-08-18 10:00 a.m. (Thursday) and repeat every Thursday.

So in summary, for interval-based schedules, first run of the Schedule equals Schedule Start Datetime. For calendar-based schedules, first run of the Schedule may or may not be equal to the Schedule Start Datetime, based on the schedule recurrence specified.

### Schedule End Datetime

It specifies an end date and time when the schedule should stop executing and no longer run. When a schedule has no end datetime specified, it runs indefinitely.

---

**Note:** that the end datetime is inclusive for the schedule execution, meaning if the schedule recurrence coincides with the end datetime, the schedule will execute at the end datetime and only then does not repeat.

---

For example, say, Schedule Start Datetime: 2016-08-15 10:00 a.m., repeat 'Hourly', Schedule End Datetime: 2016-08-15 11:00 a.m., then the schedule will run at 10:00 a.m. and also at 11:00 a.m. before ceasing to run.

### Recurrence / Repeat Interval

This specifies the frequency at which the schedule repeats. This is same as described in Simple and Advanced Scheduling.

### Schedule Next Run Datetime

This indicates the date and time of next occurrence of the schedule, obtained based on the configured schedule frequency.

## Schedule Timezone

All the date and times in the Scheduler are based on the timezone of the server (JVM) where the application is deployed.

The Scheduler Console (UI) displays the server's current date and time with timezone (the current time displayed gets refreshed when the UI is refreshed).

When creating or updating a schedule and in monitoring schedule executions in Scheduler Console, users should note that the date and time are as per the timezone setup in the application server and not the local timezone.

## Schedule Action

### Schedule Action Definition

The Schedule Action defines what is executed when the schedule runs at the specified frequency. It is a DSL (domain specific language) that is based on Groovy. The schedule action has a simple syntax as follows.

```
action {  
  //Define what needs to be executed, here. Say invoke a REST service.  
}
```

Currently Schedule Action supports calling REST services. JOS process flows are called by the Scheduler as REST services.

For example, to trigger a JOS process flow named `Store_Fnd_ProcessFlow_From_RMS`, the following schedule action is defined.

```
action {  
  
  (POST[externalVariables.processFlowAdminBaseUrl +  
    "/resources/batch/processes/operator/Store_Fnd_ProcessFlow_From_RMS"]^externalVariables.processFlowAdminBaseUrlUserAlias) as String
```

POST denotes the REST method.

`processFlowAdminBaseUrl` is an entry key in 'externalVariables' map variable used by the Scheduler runtime and specifies the BDI Process Flow Admin's base URL. The value for `processFlowAdminBaseUrl` is specified during install time and gets stored in the BDI System Options. The value of `processFlowAdminBaseUrl` will be like <https://<host>:<port>/bdi-process-flow>.

For example, <https://example.com:8001/bdi-process-flow>

- `/resources/batch/processes/operator/Store_Fnd_ProcessFlow_From_RMS` is the relative REST URL to call the process flow.
- It is of the form `/resources/batch/processes/operator/<process flow name>`.
- `processFlowAdminBaseUrlUserAlias` is an entry key in 'externalVariables' map variable used by the Scheduler runtime and specifies the alias name for JOS Process Flow Admin's user credentials to access the process flow REST service.
  - The value for `processFlowAdminBaseUrlUserAlias` is specified during install time and gets stored in the BDI System Options.

Basic authentication is used to access the JOS process flows. The Scheduler uses `processFlowAdminBaseUrlUserAlias` to lookup the credentials in the runtime secure wallet where the credentials specified at install-time are stored.

Scheduler by itself does not manage executions of process flows called from within the schedule action and any dependencies associated thereof. Scheduler only triggers process flows. The execution of process flows is done by the Process Flow engine.



For any dependencies between execution of process flows to be managed, it is recommended that such dependencies are defined in the JOS Process Flow Admin and not in the Schedule Action.

For example, if process-flow-2 needs to be run after process-flow-1 completes, use Process Flow Admin to define this dependency and not in the Schedule Action.

It is recommended to avoid time based dependency management in execution of process flows from within the Scheduler, but rather use Process Flow Admin to coordinate such dependency execution requirements.

---

**Note:** For security reasons, usage of certain keywords are not allowed in the Schedule Action DSL. When defining the schedule action in the Scheduler UI, any such forbidden keywords if used will prevent schedule from being created or updated. A schedule cannot be run if such keyword is present in the schedule action definition.

---

## Schedule Action Type

There are two types of Schedule Action - Sync and Async. When creating a schedule and defining a schedule action, user needs to specify whether the schedule action is sync or async. Scheduler determines the action execution statuses according to the action type specified.

### Sync Action

Executes synchronously and returns a result after its successful or failed completion (however long the action may run).

### Async Action

The action is asynchronous and returns a response immediately when triggered, but will continue to execute. The actual process completes at a later time. The end result of the action is not known to Scheduler in this case.

### Schedule Action Execution Status

Indicates the status of execution of the schedule action when the schedule has run at the configured frequency of time.

A schedule execution can be in one of the following statuses depending upon the Schedule Action Type and its execution.

- Triggered (applicable only for 'Async' action)
- Started (applicable only for 'Sync' action)
- Failed (applicable for both Async and Sync actions)

### Schedule Action Type and Execution Status

Schedule action type determines the schedule action status during the execution lifecycle.

### Sync Action Execution Statuses

Executes synchronously and returns a result after its successful or failed completion (however long the action may run).

- When sync action starts, the Schedule Action status will be marked 'STARTED'.
- When the action completes and returns a successful result, the status will be marked 'COMPLETED'.

- When the action does not complete because of an exception or returns a failed response (return value = "FAILED"), then the status will be marked 'FAILED'.

### Async Action Execution Statuses

The schedule action status will only be 'TRIGGERED' when the Scheduler successfully invokes the schedule action.

In case there is an exception in invoking the action itself, then the status is 'FAILED'.

By default, all BDI process flows are asynchronous that return an execution Id when triggered, but continue to run to invoke the batch jobs that complete at a later time.

### How the Action Execution Statuses are determined?

- Scheduler marks the Action Execution Status as 'FAILED' when there is an exception in executing the action or when an exception is thrown from the schedule action. In order for the Scheduler to mark the execution of schedule action as 'FAILED' when the action has been executed, the action should either throw an exception or return value as 'FAILED'.
- If the schedule action returns null or any other return value gracefully, the action execution status will be marked 'TRIGGERED' for async action and 'COMPLETED' for sync action and the returned response is stored as such in the Schedule Action Execution Log.

## Schedule Status

A schedule can be in one of the following statuses

- **Active:** An Active schedule indicates that the schedule is running at the specified frequency.
- **Inactive:** An Inactive schedule indicates that the schedule has reached its end datetime and no longer runs.
- **Disabled:** A Disabled schedule indicates that the user has disabled the schedule to not run at its specified frequency.

## Scheduler Runtime

### Scheduler Startup

As the Scheduler is deployed and the application starts up, the Scheduler service performs the following actions:

- Loads the schedules defined in the seed data sql script in the installer. This means, schedule definitions are inserted in the corresponding Scheduler infrastructure table.
- Loads the schedule action dsl for each corresponding schedule from the \*\_Action.sch files in the installer. Each schedule definition in the table is updated to include its corresponding schedule action.
- The Scheduler service sets up the runtime timers for each schedule.

When the application is deployed first-time, all schedules will be set up new. However, when the application is redeployed or the application server is restarted, the schedule timers that are already created and exist, will not be recreated.

All seed data schedules need to be specified with status as 'Active'. This ensures that the schedule timers are created at startup and the schedules start to run as per the frequency defined.

When a schedule action dsl contains any restricted keyword, the schedule will be 'Disabled' at startup and will not run. User has to correct the schedule action definition from Scheduler UI and enable the schedule to make it active.

## Schedule Runtime Execution

Scheduler uses application server's implementation of Java EE compliant timer service to execute the schedules at runtime. When a schedule is created, Scheduler sets up a timer in the application server based on the schedule frequency configured. At each scheduled time, the application server invokes callback method where the Scheduler will execute the schedule action.

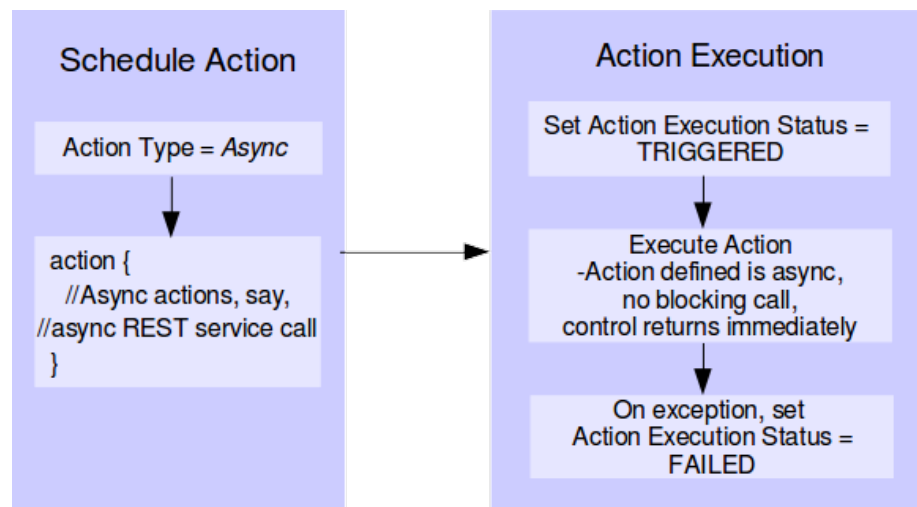
Each schedule timer executes in separate thread, so schedule executions do not block each other. Each schedule execution itself is run synchronously in its own thread, that is, the execution is blocked until it completes. But the schedule action can be specified to be asynchronous (Async action) or synchronous (Sync action) based on the action dsl defined for the schedule.

It is appropriate to specify a schedule action as 'async' when all the service calls made within the schedule action are non-blocking asynchronous calls and the action defined runs in different thread from that of the Scheduler.

If any of the service call within the schedule action is a blocking synchronous call and the action is not defined to run in separate thread, then the action type should be 'sync'.

Specifying the schedule action type 'async' or 'sync' based on the action dsl definition determines the runtime execution behavior and statuses of the schedule execution. This is explained below.

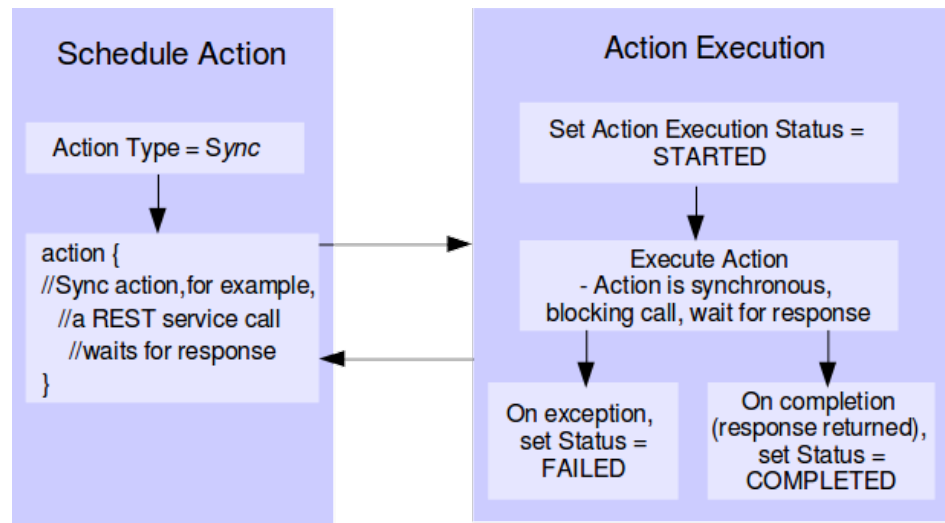
## Schedule Execution - Async Action



When the schedule action execution starts for async action, the action execution status is set to 'TRIGGERED', and the action is executed. As the action type is specified 'Async', the action should be non blocking, either returning a response immediately or not returning a response and continuing execution, but runs in separate thread returning the control immediately.

The execution of the action and the eventual status thereof will not be known to Scheduler. Once the control is returned, the schedule action execution ends but the status remains 'TRIGGERED'. In case of an exception when the action is triggered, the status is set to 'FAILED' and the execution ends.

## Schedule Execution - Sync Action



When the schedule action execution is started for 'sync' action, the action execution status is set to 'STARTED'. As the action type is specified 'sync', the action is blocking and runs in the same thread as the schedule execution.

The schedule execution ends only when the action completes returning a response or throws an exception, thereby releasing the execution thread.

After the schedule action completes successfully returns, the status is set to 'COMPLETED'. But if the action return value is 'FAILED' or the action returns throwing an exception, the status is set to 'FAILED'.

For sync actions, the action execution status in Scheduler can indicate the actual execution status (either completed or failed) of the process that was executed.

## Schedule Execution Failover

All schedule timers created by the Scheduler are persistent. This enables failover feature that in case of unexpected server shutdown or downtime, the missed schedules will be run once the server is back up. That is, the schedules that should have been run during the downtime, will be run as soon as the server is back up and the application is in running state.

---



---

**Note** that a missed schedule will be run only once, not as many times as it was missed during the downtime. For example, if a schedule is scheduled to run every 5 minutes and the application server is down for 15 minutes and restarted, the schedule will be run only one time and not 3 times. This is a feature supported by the Java EE container.

---



---

## Schedule Notification

Scheduler supports email notification of scheduled runs at runtime. The available options of events for notifications on a scheduled run are:

- Notify when the schedule action execution begins
  - This occurs when the schedule action execution is 'Started' for sync action and before triggering of action execution for async action.
- Notify when the schedule action execution ends successfully

- This occurs when the schedule action execution status is ‘Triggered’ for async actions and ‘Completed’ for sync actions
- Notify when the schedule action execution fails
  - This occurs when the status of schedule action execution is ‘Failed’, when one of the following occurs: An exception is caught in the Scheduler service itself, when an exception is thrown by the schedule action dsl, when the schedule action dsl returns the string ‘FAILED’.

## Scheduler Infrastructure Schema

The Scheduler infrastructure relies on the following schema to store the schedule definitions and schedule executions.

Scheduler service captures all schedule executions at runtime and persist the execution instances in the corresponding infrastructure table.

Table Name	Description
BDI_SCHEDULE_DEFINITION	This table contains all the schedule definitions created, including schedule frequency, schedule notification information and schedule action dsl for each schedule. Seed data schedules are loaded in this table at deployment time during application startup.
BDI_SCHEDULE_EXECUTION	All schedule executions at runtime are persisted in this table.
BDI_SYSTEM_OPTIONS	This table contains system-level global parameters as key-value pairs used by the Scheduler at runtime, such as, Process Flow Admin Base URL, Process Flow Admin User Alias, which are configured at install time by the user. User can also add system parameters to be made available to the schedule actions.

## Best Practices for Scheduler

- Use “POST” DSL method to post to REST URL.
- Use “externalVariables” for accessing variables from BDI\_SYSTEM\_OPTIONS table.
- Use “sch” as extension for schedule action DSL file.
- Try not to use time-based dependency management between schedules, instead use process flow to manage dependency.
- To schedule any existing jobs or programs, try to expose them as REST services and use the built-in dsl POST method in schedule action for executing the programs.
- Minimize use of synchronous schedule actions since they block until completion during each schedule execution.

## Scheduler Console

Scheduler Console (Admin UI) is a web user interface provided by Scheduler where users can monitor and manage schedules, including creating, updating, deleting, disabling or enabling schedules, manually running schedules, viewing schedule executions and schedule logs.

The following describes various functions available in Scheduler Console in the current release.

**Note:** It is recommended to use Chrome web browser to access Scheduler Console since the calendar widget for datetime fields are supported by Chrome browser and not by Firefox or IE as of now.

## Schedule Summary

This is the home page that provides the overall summary of the scheduler runtime. It displays the the following information.

## Schedules and Executions

This displays the total count of:

- Active Schedules
- Schedule Executions today
- Schedule Executions that were successful today
- Schedule Executions that failed today

**Note:** that today here indicates the duration from midnight to now.

### Schedules and Executions Screen



## Upcoming Schedules

Lists the future schedules that are expected to run in the next 24 hours from now.

Schedule Id	Schedule Group	Schedule Name	Schedule Next Run	Schedule Status
1	CodeDetail	CodeDetail_Fnd_From_RMS_Schedule	Sat Aug 20 00:00:00 PDT 2016	ACTIVE
2	CodeHead	CodeHead_Fnd_From_RMS_Schedule	Sat Aug 20 00:05:00 PDT 2016	ACTIVE
3	DeliverySlot	DeliverySlot_Fnd_From_RMS_Schedule	Sat Aug 20 00:10:00 PDT 2016	ACTIVE
4	Diff	Diff_Fnd_From_RMS_Schedule	Sat Aug 20 00:15:00 PDT 2016	ACTIVE
5	Diff	DiffGrp_Fnd_From_RMS_Schedule	Sat Aug 20 00:20:00 PDT 2016	ACTIVE
6	FinisherAddr	FinisherAddr_Fnd_From_RMS_Schedule	Sat Aug 20 00:25:00 PDT 2016	ACTIVE
7	Inventory	InvAvailStore_Tx_From_RMS_Schedule	Sat Aug 20 00:30:00 PDT 2016	ACTIVE
8	Inventory	InvAvailWh_Tx_From_RMS_Schedule	Sat Aug 20 00:35:00 PDT 2016	ACTIVE
9	Item	ItemHdr_Fnd_From_RMS_Schedule	Sat Aug 20 00:40:00 PDT 2016	ACTIVE
10	Item	ItemImage_Fnd_From_RMS_Schedule	Sat Aug 20 00:45:00 PDT 2016	ACTIVE

## Schedule Executions Failed Today

This lists the schedule executions that have failed today (from midnight to now).

Schedule Executions Failed Today (1)					
Schedule Execution Id	Schedule Id	Schedule Name	Schedule Execution Datetime	Schedule Action Execution Status	Schedule Action Execution Log
1354	8	InvAvailWh_Tx_From_RMS_Schedule	Wed Aug 31 04:11:40 PDT 2016	FAILED	SCHEDULED RUN: Action triggered at: Wed Aug 31 04:11:40 PDT 2016 Action Type: ASYNC Action Status: FAILED Action Response: Exception: HTTP 500 Internal Server Error

## Schedule Executions Completed / Triggered Today

This lists the schedule executions that are completed or triggered today (from midnight to now). A status of 'Completed' represents sync actions and status of 'Triggered' represents async actions.

## Schedule Executions In Progress Today

This lists the schedule executions that were started but have not completed and in progress today (from midnight to now). This is applicable only for sync actions that are in 'Started' status.

## Schedules Past Due

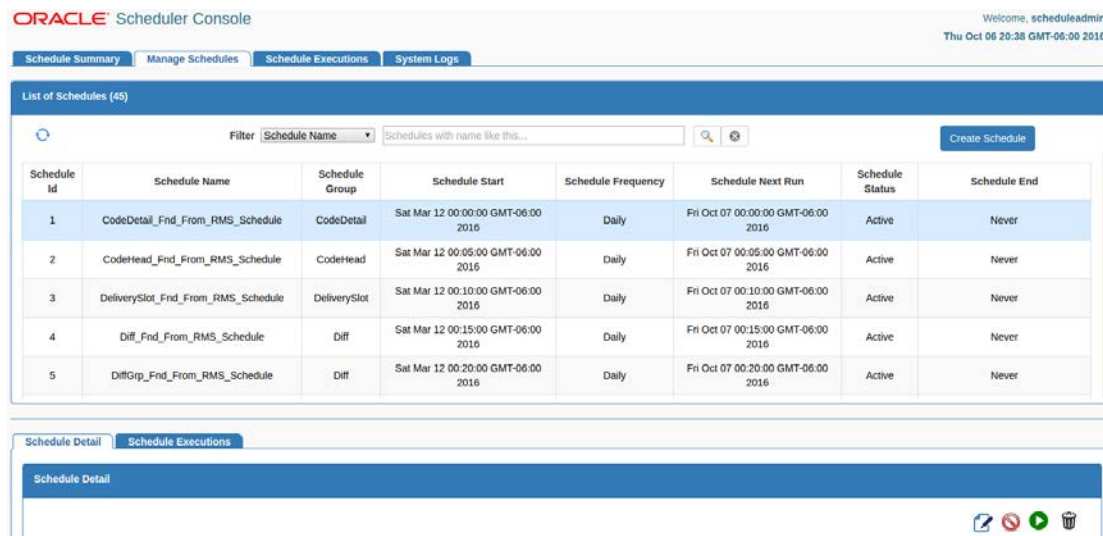
This lists the schedules that failed to run at the scheduled time (that is, schedules whose next run time is before the current time are displayed here). Ideally, there should be no missed schedules unless there maybe an internal server issue that the schedule timer failed to run.

## Manage Schedules

Manage Schedules page displays list of all schedules and details of each schedule in Schedule Detail view and corresponding schedule executions in Schedule Executions view for the schedule.

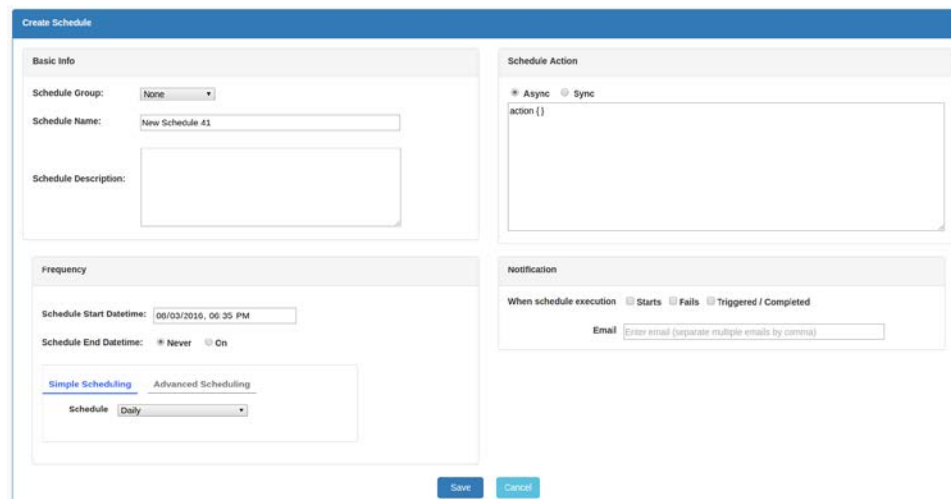
The schedules list provides options to filter schedules based on Schedule Name, Schedule Group, Schedule Status, Schedule Frequency. There is also an option to filter upcoming schedules based on date range.

The 'Create Schedule' function will be available in this page for admin users.



## Creating a Schedule

The 'Create Schedule' option displays one page where user can enter and save all required information to create a schedule. The page displays input fields under four sections as follows.



### Basic Info

Schedule Name, Schedule Group and Schedule Description are entered under Basic Info. Schedule Name and Schedule Group are required fields.

Schedule Name must be unique. User can choose an existing Schedule Group or add a new group name for the schedule.

There is limitation for the number of characters that these fields can accept.

### Schedule Action

Specify a valid schedule action definition here, that will get executed when the schedule runs.

If any restricted keyword is present in the action definition, schedule cannot be saved, and when saving the schedule an error highlighting the restricted keyword will be displayed.



Also choose here whether the schedule action is 'Async' (which is the default selected option) or 'Sync'.

**Note:** the schedule action is not validated or compiled for syntax when creating schedule, so any syntax or programming error in the action definition will result in an exception at runtime and the schedule execution will fail.

The screenshot shows a window titled "Schedule Action". At the top, there are two radio buttons: "Async" (which is selected) and "Sync". Below the radio buttons is a large text area containing the code "action {}". The text area has a small cursor icon at the bottom right corner.

## Schedule Frequency

It consists of Schedule Start Date time, End Date time and Schedule Recurrence.

Schedule End Datetime is 'Never' by default meaning the schedule never ends and repeats indefinitely. If the schedule has an end datetime, user can enter a specific datetime.

Start Datetime defaults to 5 minutes from current time and End Datetime defaults to 6 minutes from current time when chosen.

Start and End datetimes should be future dates. Schedule End datetime if specified should be after the scheduled start datetime. These validations will be done when saving the schedule.

Scheduler provides two options to specify recurrence of schedule - Simple Scheduling and Advanced Scheduling. Use the options tabs to toggle between Simple and Advanced Scheduling options.

Simple Scheduling provides the following predefined schedules that user can choose from dropdown list.

- Hourly
- Daily (selected by default)
- Weekly
- Monthly
- Every Weekday [Mon-Friday]
- On Weekends [Sat-Sunday]
- Every Saturday
- Every Sunday
- First day of every month
- Last day of every month

One time only

Specify a different frequency

User can use this option to specify a recurring interval in minutes, hours, days or weeks, for example, 30 minutes, 2 hours, 3 days, and so on.

Advanced Scheduling enables user to specify complex schedules similar to a cron expression. User can choose multiple values for Hours, Minutes, Day of Week, Day of Month and Month options using the multi-select lists.

The default schedule frequency here is daily midnight (Hours: 12 a.m., Minutes: 0 are the values selected by default).

Simple Scheduling	Advanced Scheduling			
Hours	Minutes	Day of Week	Day of Month	Month
12 a.m. ▲	0 ▲	Sun ▲	1 ▲	Jan ▲
1 a.m.	1	Mon	2	Feb
2 a.m.	2	Tue	3	Mar
3 a.m.	3	Wed	4	Apr
4 a.m.	4	Thu	5	May
5 a.m.	5	Fri	6	Jun
6 a.m.	6	Sat	7	Jul
7 a.m.	7		8	Aug
8 a.m.	8		9	Sep
9 a.m.	9		10	Oct
10 a.m.	10		11	Nov
11 a.m. ▼	11 ▼		12 ▼	Dec ▼

## Schedule Notification

Use schedule notification option to enable email notification for the schedule when schedule execution starts or fails or completed.

Enter valid email addresses for notification. When enabled, email alerts will be sent based on the options selected.

### Starts:

When this option is chosen, email will be sent when the schedule execution starts, that is, when the schedule runs at the scheduled interval, and just before the execution of schedule action.

### Fails:

Email will be sent when there is an exception in schedule execution or when the schedule action throws an exception or returns 'Failed' response. This means the schedule action execution will be in 'Failed' status.

### Triggered / Completed:

Email will be sent when the schedule action execution status is 'Triggered' (for async actions) and 'Completed' (for sync actions). This essentially means the schedule execution is successful.

**Notification**

When schedule execution  Starts  Fails  Triggered / Completed

Email

---

**Note:** for schedule notification to work, mail session needs to have been configured in the WebLogic server. Refer JOS Installation Guide for details on the configuration of mail session.

---

## Updating a Schedule

A schedule can be updated by selecting the schedule from the Manage Schedules page and using the Edit option in Schedule Detail view.

The Edit page is same as that of the Create Schedule page with the schedule information populated. Update the values as required in the relevant sections as explained previously for creating schedule. Only admin user can edit a schedule.

---

**Note:** that updating schedule frequency will validate schedule start datetime and end datetime (if specified) similar to when creating schedule.

---

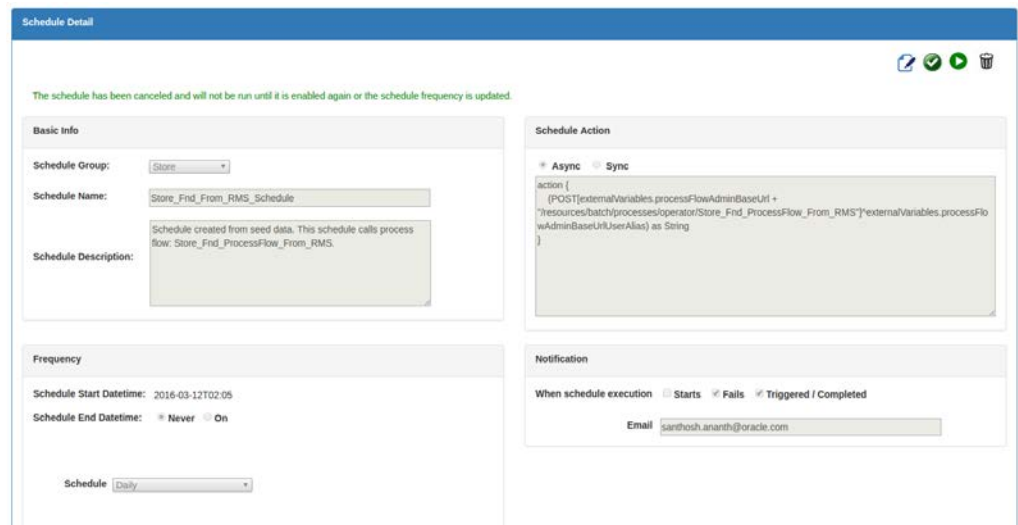
Updating any other details other than schedule frequency will not validate the existing schedule frequency as the schedule will continue to run at the already defined frequency and only the other details of schedule definition will get updated as modified by the user. When changing the schedule action definition, it will be verified for any restricted keywords.

## Disabling a Schedule

A schedule can be disabled by selecting the schedule from Mange Schedule page and using the 'Disable schedule' option in the Schedule Detail view. Only admin and operator users can disable a schedule.

Disabling a schedule will change the schedule status to 'Disabled' and the schedule will no longer run at the specified frequency. However the schedule can be manually run using the 'Run Schedule Now' option.

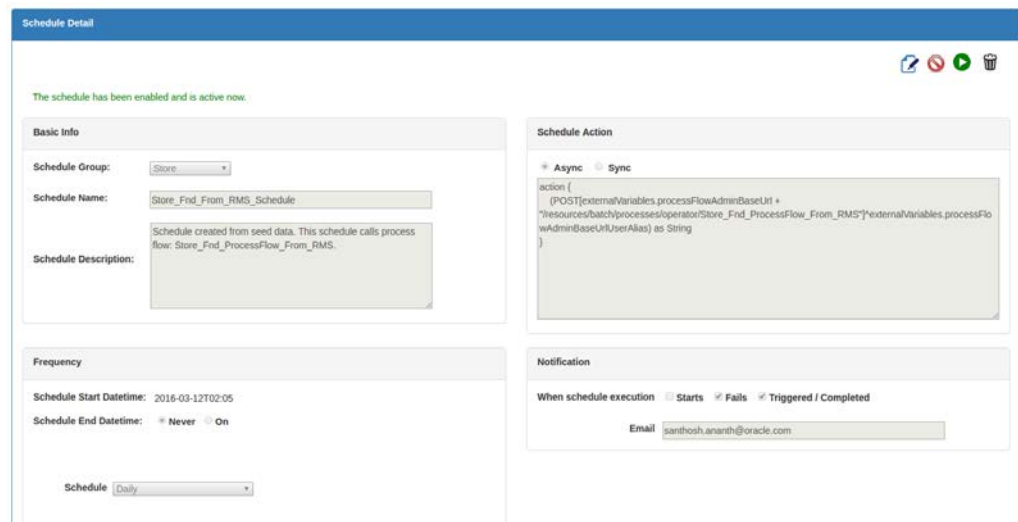
**Note:** that 'Inactive' schedule cannot be disabled, as an inactive schedule has reached its end already and no longer runs.



### Enabling a Schedule

A disabled schedule can be enabled again using the 'Enable schedule' option from the Schedule Detail view. Only admin and operator users can enable a schedule.

Enabling the schedule will change the status of the schedule to 'Active' and the schedule will resume running at the specified frequency.



### Deleting a Schedule

A schedule can be deleted using the 'Delete schedule' option in the Schedule Detail view. Only admin user can delete a schedule.

**Note:** Deleting a schedule will delete the schedule definition and also its entire execution history. The schedule will no longer exist and will not run after deletion. There is no way to restore a deleted schedule except by creating the schedule again.

The screenshot shows the 'Schedule Detail' view with a confirmation message at the top: "Deleting the schedule will also delete its execution history and the schedule will no longer run. Please confirm." Below the message are 'Delete' and 'Cancel' buttons. The form is divided into several sections:

- Basic Info:**
  - Schedule Group: Store
  - Schedule Name: Store\_Fnd\_From\_RMS\_Schedule
  - Schedule Description: Schedule created from seed data. This schedule calls process flow: Store\_Fnd\_ProcessFlow\_From\_RMS.
- Frequency:**
  - Schedule Start Datetime: 2016-03-12T02:05
  - Schedule End Datetime: Never (selected) / On
  - Schedule: Daily
- Schedule Action:**
  - Async (selected) / Sync
  - Action: 

```
action [
  (POST|externalVariables.processFlowAdminBaseUri +
  "/resources/batch/processes/operator/Store_Fnd_ProcessFlow_From_RMS")externalVariables.processFlo
  wAdminBaseUriUserAlias) as String
]
```
- Notification:**
  - When schedule execution: Starts (selected) / Fails / Triggered / Completed
  - Email: santhosh.ananth@oracle.com

## Schedule a Manual Run

Any schedule can be manually run using the 'Run Schedule Now' option from the Schedule Detail view. Inactive and disabled schedules can also be manually run.

This option is provided so that user can run a schedule on demand when required. Only admin and operators can access this function.

When the schedule is run manually, the schedule action is submitted for execution in the backend and the result of execution can be seen from the Schedule Executions view.

The screenshot shows the 'Schedule Detail' view with a confirmation message at the top: "Schedule Action successfully submitted for execution. See Schedule Executions for further detail on the status/result of execution." The form structure is identical to the previous screenshot, but the 'Delete' and 'Cancel' buttons are no longer present, and the confirmation message is displayed.

### Schedule a Executions

From the Schedule Executions page user can view all available historical schedule executions. The page will display schedule executions for the last one week by default. The User can use the search option to enter a different date range to fetch the corresponding schedule executions.

Within the list of schedule executions, the records can be filtered based on Schedule Name, Action Execution Status and any string within the Action Execution Log. The list of scheduled executions are sorted by schedule execution datetime, the latest first.

Schedule Executions From 09/26/2016, 12:48 PM To 10/03/2016, 12:48 PM Go					
List of Schedule Executions (281)					
Filter Schedule Name Name like this...					
1280	26	Store_Fnd_From_RMS_Schedule	Mon Oct 03 02:05:00 CDT 2016	TRIGGERED	SCHEDULED RUN: Action triggered at: Mon Oct 03 02:05:00 CDT 2016 Action Type: ASYNC Action Status: TRIGGERED Action Response:
1279	25	StoreAddr_Fnd_From_RMS_Schedule	Mon Oct 03 02:00:00 CDT 2016	TRIGGERED	SCHEDULED RUN: Action triggered at: Mon Oct 03 02:00:00 CDT 2016 Action Type: ASYNC Action Status: TRIGGERED Action Response:
1278	24	ReplitemLoc_Fnd_From_RMS_Schedule	Mon Oct 03 01:55:00 CDT 2016	FAILED	SCHEDULED RUN: Action triggered at: Mon Oct 03 01:55:00 CDT 2016 Action Type: ASYNC Action Status: FAILED Action Response: Exception: HTTP 500 Internal
1277	23	RelatedItem_Fnd_From_RMS_Schedule	Mon Oct 03 01:50:00 CDT 2016	FAILED	SCHEDULED RUN: Action triggered at: Mon Oct 03 01:50:00 CDT 2016 Action Type: ASYNC Action Status: FAILED Action Response: Exception: HTTP 500 Internal

### System Logs

The System Logs page displays list of all schedule log files and log contents. Each schedule has its own log file enabling easy access for the user to view the execution logs and other information from the log files for diagnosing and troubleshooting issues.

The list of log files are sorted by last modified time of file with most recently modified file first.

The screenshot displays the Oracle Scheduler Console interface. At the top, it says 'ORACLE Scheduler Console' and 'Welcome, bdiadmin Wed Aug 31 05:52 PDT 2016'. Below the navigation tabs (Schedule Summary, Manage Schedules, Schedule Executions, System Logs), the 'Scheduler Log Files' section shows a table of log files:

Log File Name	Size (in KB)	Last Modified
scheduler-default.log	312.16	Wed Aug 31 05:01:24 PDT 2016
CodeDetail_Fnd_From_RMS_Schedule1.log	649.76	Wed Aug 31 05:00:05 PDT 2016
MerchHier_Fnd_From_RMS_Schedule.log	59.39	Wed Aug 31 04:11:55 PDT 2016
PartOrgUnit_Fnd_From_RMS_Schedule.log	45.13	Wed Aug 31 04:11:55 PDT 2016
UomClass_Fnd_From_RMS_Schedule.log	23.26	Wed Aug 31 04:11:55 PDT 2016
ITSupCtry_Fnd_From_RMS_Schedule.log	16.11	Wed Aug 31 04:11:55 PDT 2016
PackItem_Fnd_From_RMS_Schedule.log	51.98	Wed Aug 31 04:11:55 PDT 2016
Udaltemlov_Fnd_From_RMS_Schedule.log	23.35	Wed Aug 31 04:11:55 PDT 2016
Replitemloc_Fnd_From_RMS_Schedule.log	66.56	Wed Aug 31 04:11:55 PDT 2016

Below this, the 'Log Content' section shows a snippet of log text:

```

2016-08-27T01:20:00,418 [[ACTIVE]] ExecuteThread: '7' for queue: 'weblogic.kernel.Default (self-tuning)'] INFO CalendarScheduleTimerBean - The schedule
MerchHier_Fnd_From_RMS_Schedule will next run at: 2016-08-28T01:20:00,418-0700
2016-08-27T01:20:00,427 [[ACTIVE]] ExecuteThread: '11' for queue: 'weblogic.kernel.Default (self-tuning)'] INFO ScheduleExecutorServiceBean - ***Schedule Run :
Action Execution Begin*** ScheduleId: 17 - ScheduleName: MerchHier_Fnd_From_RMS_Schedule - ScheduleExecutionId: 1197
2016-08-27T01:20:00,428 [[ACTIVE]] ExecuteThread: '11' for queue: 'weblogic.kernel.Default (self-tuning)'] INFO ScheduleExecutorServiceBean - Executing action
for the schedule: MerchHier_Fnd_From_RMS_Schedule
2016-08-27T01:20:00,528 [[ACTIVE]] ExecuteThread: '11' for queue: 'weblogic.kernel.Default (self-tuning)'] DEBUG Logger$Debug - Action status: COMPLETED
2016-08-27T01:20:00,528 [[ACTIVE]] ExecuteThread: '11' for queue: 'weblogic.kernel.Default (self-tuning)'] DEBUG Logger$Debug - Action response:

```

## Scheduler Security Considerations

### Scheduler Security

Scheduler application uses basic authentication to authenticate users and allow access to the requested resources based on authorization. Only valid users can access the Scheduler Console and the REST resources. Scheduler accesses BDI process flows using basic authentication.

Users need to belong to one of these roles:

- Admin (assigned to BdiSchedulerAdminGroup in WebLogic Server)
- Operator (assigned to BdiSchedulerOperatorGroup in WebLogic Server)
- Monitor (assigned to BdiSchedulerMonitorGroup in WebLogic Server)

Only authorized users of specific role are allowed to access certain functionalities in the Scheduler Console.

Users of Admin role have access to all the functions in Scheduler, users of Operator role have limited authorizations to use only certain functions, and users of Monitor role only have view/read-only access to Scheduler Console.

Function	Admin Role	Operator Role	Monitor Role
View and search	Yes	Yes	Yes
Create schedule	Yes	No	No
Edit schedule	Yes	No	No
Delete schedule	Yes	No	No
Manual run schedule	Yes	Yes	No
Disable schedule	Yes	Yes	No
Enable schedule	Yes	Yes	No

## Scheduler Operational Considerations

### Users Roles for Monitoring and Administration

Scheduler application is secured with role based security authorization. It is recommended to use separate users for Monitor, Operator and Admin roles.

### Monitoring Schedules

Schedules and executions can be effectively monitored using Scheduler Console. The console provides detailed action execution log and log files for each of the schedules that can be used to verify the runtime executions of schedules and related information.

#### Schedule Action Execution Log

Each schedule execution contains 'Schedule Action Execution Log' that provides descriptive information on the scheduled run or manual run of the schedule. The Schedule Action Execution Log provides information as follows.

```
<SCHEDULED or MANUAL> RUN: Action triggered at: <Date and time>
Action Type: <ASYN or SYNC>
Action Status: <TRIGGERED or STARTED or COMPLETED or FAILED>
Action Response: <The response string as returned by the schedule action dsl, or
the error message in case of an exception>
```

For example, for a successful execution of schedule ItemHdr\_Fnd\_From\_RMS\_Schedule at the scheduled frequency, and action that triggers the process flow ItemHdr\_Fnd\_ProcessFlow\_From\_RMS, the Schedule Action Execution Log will be:

```
SCHEDULED RUN: Action triggered at: Wed Jul 27 12:00:01 EDT 2016
Action Type: ASYNC
Action Status: TRIGGERED
Action Response: {"executionId":"ItemHdr_Fnd_ProcessFlow_From_RMS#0d3d656d-041a-4068-8daf-8d17eelad899","processName":"ItemHdr_Fnd_ProcessFlow_From_RMS" }
```

In case of an exception (say, connection error when invoking process flow), the action execution log will be like:

```
SCHEDULED RUN: Action triggered at: Sat Aug 06 00:40:00 EDT 2016
Action Type: ASYNC
Action Status: FAILED
Action Response: Exception: java.net.ConnectException: Tried all: '1' addresses,
but could not connect over HTTP to server: java.net.ConnectException: Connection
refused
Check the logs for more details.
```

The above action execution log examples indicate async actions. For sync actions, the the action execution log also shows when the schedule action started and when it completed, which is particularly useful for a long running action for which the Scheduler waits for the response until completion. For example,

```
SCHEDULED RUN: Action execution started at: Wed Aug 03 12:00:00 EDT 2016
Action Type: SYNC
Action execution ended at: Wed Aug 03 12:22:10 EDT 2016
Action Status: COMPLETED
Action Response: Batch process completed.
```

---

---

**Note:** The Action Response shows the value that the schedule action dsl finally returns after completion.

---

---



## Scheduler Log Files

Each schedule has its own log file. For example, a schedule named `Store_Fnd_From_RMS_Schedule` will have its log file named `Store_Fnd_From_RMS_Schedule.log`.

The log file contains detailed information on schedule executions which can be scheduled runs or manual runs, logs of actions such as disabling and enabling the schedule, action log on schedule updates such as change in schedule frequency, and in case of any exceptions, the exception stack trace.

Users can use the following keywords to search for specific information in the schedule log file.

Keyword	Description
ScheduleId	The primary key Id of schedule
ScheduleName	The schedule name
ScheduleExecutionId	The execution Id of schedule run instance
Action Execution Begin	Indicates the start of the log when schedule action begins.
Action Execution End	Indicates the end of the log when schedule action ends. The log of the schedule action execution can be found between the two strings: <b>***Schedule Run :</b> Action Execution Begin*** and ***Schedule Run : Action Execution End*** For manual run, it will be <b>***Manual Run :</b> Action Execution Begin*** and ***Manual Run : Action Execution End***
Action execution exception	The detailed exception message and stacktrace will be shown following this string, when an exception has occurred in schedule action execution.

## Maintaining Historical Schedule Executions

As the schedules run, schedule execution records are stored in the `BDI_SCHEDULE_EXECUTION` table.

This table will grow larger as the number of schedule executions increase. Hence it is recommended to periodically purge historical schedule executions from the table that are older and no longer necessary, and only retain recent schedule executions of a particular period, say for the last one month to now. This will help keep the table size within certain limit and prevent database growth.

## Scheduler Customization

Seed Data Reload

The sql script containing the seed data schedule definitions is located in `bdi-scheduler-home/setup-data/dml` folder.

During the initial deployment of Scheduler application, seed data schedules get loaded to schedule definition table and the corresponding schedules are created.

If the Scheduler application needs to be redeployed and the seed data schedules need to be reloaded during the redeployment (that is, to reset the schedules to the initial state as per seed data), set the `LOADSEEDDATA` column in `BDI_SYSTEM_OPTIONS` table to `TRUE`, and undeploy and redeploy the application.

---

**Note:** The above redeployment procedure will reset the current schedule definitions (that is, existing schedules and any changes will be deleted) and the schedules will be recreated as per seed data definitions. Use this option with caution and only when absolutely necessary.

---

## Customizing Seed Data Schedules

By default all BDI seed data schedules are scheduled to run daily starting at midnight (each schedule running with a gap of 5 minutes). The User can edit the seed data and add new schedules to be loaded during deployment, by updating the seed data sql script and adding corresponding schedule action scripts in the `bdi-scheduler-home` install directory, before starting the installation.

Seed data sql file: `bdi-scheduler-home/setup-data/dml/seed-data.sql`

Schedule Action dsl files: `bdi-scheduler-home/setup-data/dsl`

An insert statement for a schedule seed data definition will look like below (SQL for Oracle database):

```
INSERT INTO BDI_SCHEDULE_DEFINITION (schedule_id, schedule_name, schedule_group,
schedule_description, schedule_status, schedule_start_datetime, schedule_type,
schedule_frequency, schedule_notification, schedule_notification_email,
schedule_action_type, schedule_action_definition) VALUES (7,
'InvAvailStore_Tx_From_RMS_Schedule', 'Inventory', 'Schedule created from seed
data. This schedule calls process flow: InvAvailStore_Tx_ProcessFlow_From_RMS.',
'ACTIVE', TIMESTAMP '2016-03-12 00:30:00', 'SIMPLE', 'DAILY',
'ON_SUCCESS,ON_ERROR', 'user@example', 'ASync',
'InvAvailStore_Tx_From_RMS_Schedule_Action.sch')
```

---

**Note** the following when adding or editing schedule definitions in seed data to be loaded at application startup. All these fields (as shown in the sql statement above) are required fields to create a schedule at startup.

---

- `schedule_id` should be a unique number for each schedule.
- `schedule_name` should be unique.
- `schedule_status` needs to be 'ACTIVE' for schedule to be created and active.
- `schedule_type` should be 'SIMPLE' with any of the `schedule_frequency` values mentioned above. Advanced schedule (calendar schedules with complex cron expression) is not supported through seed data during deployment.
- `schedule_start_datetime`:  
Need to be in the format `yyyy-mm-dd hh:mm:ss`  
For example, 2016-01-01 00:00:00, 2016-01-01 18:30:00
- `schedule_frequency`:  
Valid values are: DAILY, HOURLY, WEEKLY, MONTHLY, WEEKDAY, WEEKEND, SATURDAY, SUNDAY, FIRSTDAYOFMONTH, LASTDAYOFMONTH, ONCE
- `schedule_notification`:  
Valid values are: ON\_START, ON\_SUCCESS, ON\_ERROR (separate multiple values by comma)

- `schedule_email`:  
Valid email-id for notification (separate multiple emails by comma). Email is required if `schedule_notification` is specified.
- `schedule_action_type`:  
Valid values are (based on the action specified): 'ASYNC' or 'SYNC'
- `schedule_action_definition` in seed data refers to the name of the corresponding schedule action dsl file (this will get loaded at startup).  
Each schedule should have corresponding schedule action dsl script defined. This will be the action that gets executed when the schedule runs.

To load the schedule action dsl during deployment, add the schedule action dsl file under `bdi-scheduler-home/setup-data/dsl` with file name convention: `<Schedule Name>_Action.sch`

For example for adding a new schedule named `Schedule_1`, add schedule action dsl script `Schedule_1_Action.sch`. During deployment, Scheduler will create `Schedule_1` and update the schedule definition with the action script from the corresponding file `Schedule_1_Action.sch`.

## Customizing Schedule Actions

The seed data schedules in Scheduler are the schedules that call the JOS process flows provided out-of-the-box. The Schedule Actions define the REST calls to the JOS process flows.

In an enterprise implementation, there will be requirements to schedule batch processes, any recurring jobs or activities that are not BDI process flows. There can also be existing batch processes or services that need to be scheduled.

The Scheduler can be used for such scheduling requirements by defining appropriate Schedule Action to invoke the services.

Scheduler can be used to schedule RESTful services and as the Schedule Action is a DSL based on Groovy, valid Groovy or Java code can also be used within the action part that will be executed by the Scheduler based on the defined schedule.

The syntax for Schedule Action is as simple as follows.

```
action {
    //your implementation goes here
}
```

The following Schedule Action syntax specifies how a REST service can be called from Scheduler (assuming the REST resource does not require any authentication). The response from the REST service will be treated as string.

```
action {
    (POST[<your REST service URL here>]) as String
}
```

This is a simple approach for scheduling existing and new services that can be exposed as REST services.

The Schedule Action syntax to call a REST service with authentication and with base URL configured in System Options will be like below.

```
action {
    POST[externalVariables.myRESTServiceBaseUrl +
        "/resources/myRESTresource" ]^externalVariables.myRESTServiceBaseUrlUserAlias) as
    String
}
```

The *externalVariables* is the name of the variable used internally by the Scheduler to access system options parameters. Any parameters (key-values) configured in System Options can be accessed using the notation *externalVariables.<my-system-option-parameter>*

Admin users can utilize System Setting RESTful service to add or update system options parameters, and setting up credentials (stored in wallet) for any authentication to be used by the application. Refer [Appendix D](#) for details on the System Setting REST resources.

In the above example, user can add system option parameters named 'myRESTServiceBaseUrl' with the REST resource base url value (for example, `http://<myserverhost>:<port>/myapp`) and 'myRESTServiceBaseUrlUserAlias' which will be the alias name to be used for authentication and the value of this parameter should be `GET_FROM_WALLET:GET_FROM_WALLET` to indicate that the corresponding credentials for the alias need to be obtained from the wallet during runtime by the application.

## Scheduler Troubleshooting

Any failure in schedule execution can be analysed in Scheduler application by checking the Scheduler log files for the corresponding schedule.

If a schedule execution is 'FAILED' due to an exception response from process flow, then the details of corresponding process flow execution instance, the exception details and any stack trace can be viewed in the corresponding process flow logs using Process Flow Admin console for further troubleshooting.

---

---

**Note:** that the schedule execution where JOS process flow is called is only a trigger for the process flow execution, hence the actual execution of process flow and the status and logs thereof can only be viewed in JOS Process Flow Admin console.

---

---

## Scheduler Known issues

Scheduler Console provides calendar widget for datetime fields that is currently supported only by Chrome browser. Hence it is recommended to use the latest version of Chrome browser to access the Scheduler Console.

If any other browser is used that does not support the calendar widget for the datetime input, the datetime fields may appear as textbox. Users can enter the datetime input as text, but the value should be in the format of '*yyyy-MM-ddTHH:mm*', for example, 2016-01-01T20:00. There is no loss of functionality due to this limitation however.

## How do I create a batch job in Job Admin?

1. Download "JosJobAdmin16.0.0ForAll16.x.xApps\_eng\_ga.zip" and unzip the file.
2. Create job XML files using Java Batch specification. See the sample Job XML below.
3. Copy job XML files to "jos-job-home/setup-data/META-INF/batch-jobs" folder.
4. Copy jar file that contains code related to jobs in "jos-job-home/lib" folder.
5. Run the deployer script in "jos-job-home/bin" folder.

## Sample Job XML

Below is sample Job XML that runs "ls" shell command.

```
<job id="ShellCommandRunnerBatchlet" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
version="1.0">
  <step id="shellCmd">
    <batchlet ref="ShellCommandRunnerBatchlet">
      <properties>
        <!-- externalCommand format - command param1 .. paramN
        parameters can be static or dynamic
        if a parameter is dynamic, then use #SysOpt.paramName.
        paramName should be setup in BDI_SYSTEM_OPTIONS table
        -->
        <property name="externalCommand" value="ls"/>
        <!-- externalCommandWorkingDir is optional -->
        <property name="externalCommandWorkingDir" value="."/>
      </properties>
    </batchlet>
    <end on="COMPLETE"/>
  </step>
</job>
```

## How do I pass job parameters to a shell script invoked by job?

Job parameters can be passed to a shell script from the job using the following syntax in the job.

```
#{jobParameters['param1']}
```

param1 - Name of the parameter

### Sample job that uses job parameters

```
<job id="ShellCommandRunnerBatchlet" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
version="1.0">
  <step id="shellCmd">
    <batchlet ref="ShellCommandRunnerBatchlet">
      <properties>
        <property name="externalCommand" value="ls
        #{jobParameters['param1']} #{jobParameters['param2']}" />
        <!-- externalCommandWorkingDir is optional -->
        <property name="externalCommandWorkingDir" value="."/>
      </properties>
    </batchlet>
```

```
        <end on="COMPLETE" />
    </step>
</job>
```

If the following parameters are entered in Job Admin UI during launching of the above job, the below command will be run by the job.

Job Parameters: param1=-a,param2=-l  
Command executed: ls -a -l .

## How do I pass system options to a shell script invoked by job?

System Options can be passed to a shell script from the job using the following syntax in the job.

```
#SysOpt.paramName
```

Sample job that uses system options

```
<job id="ShellCommandRunnerBatchlet" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
version="1.0">
    <step id="shellCmd">
        <batchlet ref="ShellCommandRunnerBatchlet">
            <properties>
                <property name="externalCommand" value="ls"/>
                <!-- externalCommandWorkingDir is optional -->
                <property name="externalCommandWorkingDir" value="#SysOpt.dir"/>
            </properties>
        </batchlet>
        <end on="COMPLETE" />
    </step>
</job>
```

If the following system option is set in Job Admin UI, the below command will be run by the job.

System Option: dir=/home/batch

Command executed in the working directory "/home/batch".

## How do I pass system properties to a shell script invoked by job?

Java system properties can be passed to a shell script from the job using the following syntax in the job.

```
#{systemProperties['prop1']}
```

Sample job that uses system property

```
<job id="ShellCommandRunnerBatchlet" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
version="1.0">
    <step id="shellCmd">
        <batchlet ref="ShellCommandRunnerBatchlet">
            <properties>
                <property name="externalCommand" value="ls"/>
                <!-- externalCommandWorkingDir is optional -->
                <property name="externalCommandWorkingDir"
value="#{systemProperties['batchDir']}" />
            </properties>
        </batchlet>
        <end on="COMPLETE" />
    </step>
</job>
```

If the following system property is set in the JVM for Job Admin, the below command will be run by the job.

System Property: `-DbatchDir=/home/batch`

Command executed in the working directory `"/home/batch"`.

## How do I chain multiple jobs in a single flow?

For running multiple jobs need to run in sequence, create a DSL to chain the jobs.

### Sample Process Flow

The below process flow runs two jobs, "jobA" and "jobB" in sequence. The activity "AbcActivity" starts jobA by calling a REST endpoint in Job Admin. The activity "AbcStatusActivity" calls a REST endpoint in Job Admin to check the status of the "jobA". It waits until the job is complete or failed. This is a standard pattern for running a batch job. After "jobA" is complete, process flow engine runs the "jobB".

```
process {
    name "AbcProcess"
    var ([a:"b", c:"d", e: 5])

    begin{
        action{
            println "$activityName Load variables"
            println "Access externalVariables=$externalVariables"
            return "okay"
        }
        on "okay" moveTo "AbcActivity"
    }

    activity{
        name "AbcActivity"
        action{
            startOrRestartJob(externalVariables["jobAdminUrl"], "JobA",
                externalVariables["jobAdminUrlUserAlias"])
            "okay"
        }
        on "okay" moveTo "AbcStatusActivity"
        on "error" moveTo "ErrorActivity"
    }

    activity{
        name "AbcStatusActivity"
        action{
            waitForJobCompletedOrFailed("AbcActivity", externalVariables["jobAdminUrl"] +
                "/resources/batch/jobs/JobA/" + processVariables["jobExecutionId"],
                externalVariables["jobAdminUrlUserAlias"])
            "okay"
        }
        on "okay" moveTo "DefActivity"
    }

    activity{
        name "DefActivity"
        Action{
            startOrRestartJob(externalVariables["jobAdminBaseUrl"], "JobB",
                externalVariables["jobAdminUrlUserAlias"])
            "okay"
        }
        on "okay" moveTo "DefStatusActivity"
    }
}
```

```

activity{
    name "DefStatusActivity"
    action{
        waitForJobCompletedOrFailed("DefActivity",externalVariables["jobAdminUrl"] +
"/resources/batch/jobs/JobB/" + processVariables["jobExecutionId"],
externalVariables["jobAdminUrlUserAlias"])
        "okay"
    }
    on "okay" moveTo "end"
}

activity{
    name "ErrorActivity"
    action{
        println "$activityName This is error activity"
        return "okay"
    }
    on "okay" moveTo "end"
}

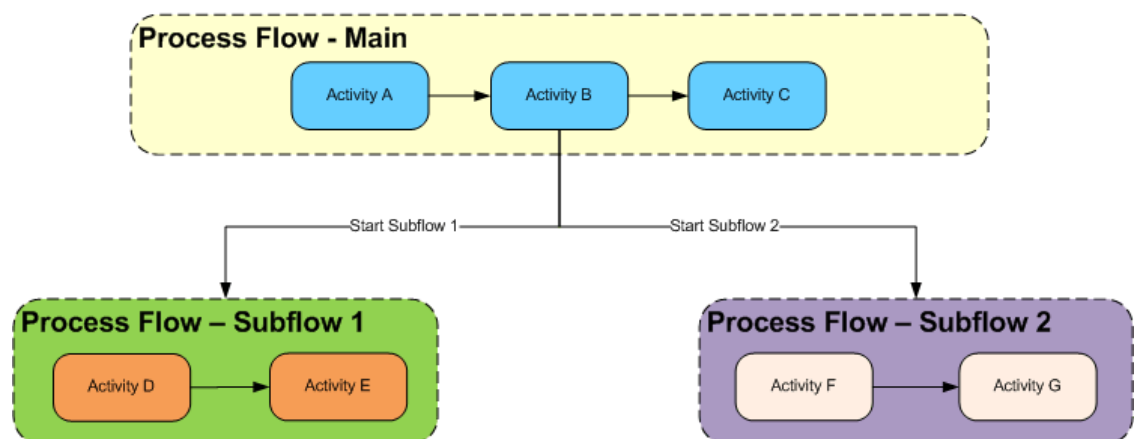
end{
    action{
        println "Got to end"
        return "COMPLETED"
    }
}
}

```

## How do I create split flows?

The main flow needs to fork other flows. Use “POST” method to start a process flow from another process flow.

### Split Flows



### Sample Split Flow

In this sample flow, the activity “GhiProcessActivity” posts a request to process flow application to start a new process flow “GhiProcess” and the main flow continues with rest of the activities. The sub flow runs independently of the main flow.



## Main Flow

```

process {
  name "DefProcess"

  begin{
    action{
    }
    on "okay" moveTo "GhiProcessActivity"
  }

  activity{
    name "GhiProcessActivity"
    action {
      (POST[externalVariables.processFlowAdminBaseUrl +
"/resources/batch/processes/operator/ProcessGhi" ]
]^externalVariables.processFlowAdminBaseUrlUserAlias)
        "okay"
    }
    on "okay" moveTo "DefActivity"
  }

  activity{
    name "DefActivity"
    action{
      "okay"
    }
    on "okay" moveTo "end"
  }

  end{
    action{
      return "COMPLETED"
    }
  }
}

```

## Sub Flow

```

process {
  name "GhiProcess"

  begin{
    action{
    }
    on "okay" moveTo "GhiActivity"
  }

  activity{
    name "GhiActivity"

    action{
      //do something here
    }
    on "okay" moveTo "end"
  }

  end{
    action{
      return "COMPLETED"
    }
  }
}

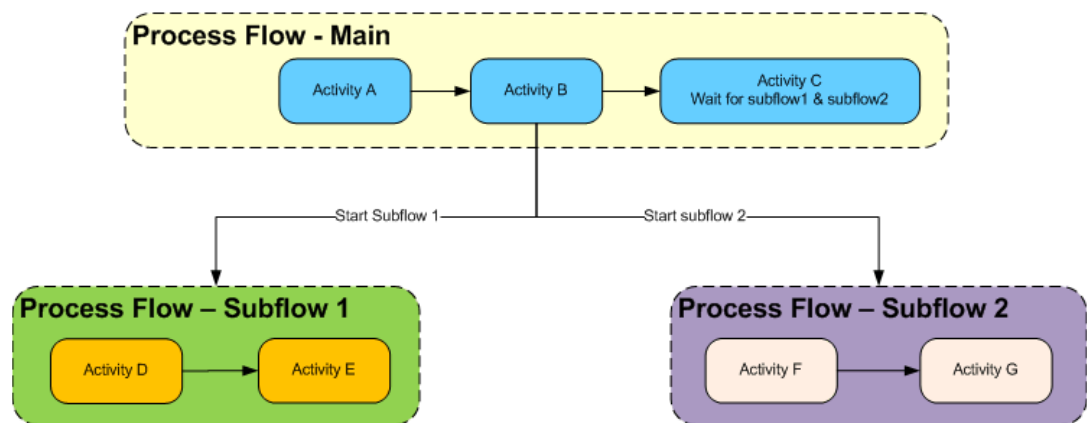
```

}

## How do I create split and join flows?

Process flow "Abc" starts process flow "Def" and "Xyz". Process flow "Abc" has to wait until "Def" and "Xyz" process flows are complete. The activity "AbcActivity" waits until "DefProcess" and "XyzProcess" are complete. Use "waitForProcessInstancesToReachStatus" method to wait for other flows to complete.

### Split And Join Flows



### Sample Split and Join Flow

```

process {
  name "AbcProcess"

  begin{
    action{
      "okay"
    }
    on "okay" moveTo "DefAndXyzActivity"
  }

  activity{
    name "DefAndXyzActivity"
    action {
      def defExecution = ((POST[externalVariables.processFlowAdminBaseUrl +
"/resources/batch/processes/operator/ProcessDef" ]
]^externalVariables.processFlowAdminBaseUrlUserAlias) as
ProcessExecutionIdsVo.ProcessExecutionIdVo)
processVariables['processDefExecution'] = defExecution.executionId
      def xyzExecution = ((POST[externalVariables.processFlowAdminBaseUrl +
"/resources/batch/processes/operator/ProcessXyz" ]
]^externalVariables.processFlowAdminBaseUrlUserAlias) as
ProcessExecutionIdsVo.ProcessExecutionIdVo)
processVariables['processXyzExecution'] = xyzExecution.executionId
      "okay"
    }
    on "okay" moveTo "AbcActivity"
  }

  activity{
  
```

```

        name "AbcActivity"
        Action{

            waitForProcessInstancesToReachStatus([processVariables['processDefExecution'],
processVariables['processXyzExecution']], PROCESS_COMPLETED, LOGICAL_AND)
"okay"
            }
            on "okay" moveTo "end"
        }

        end{
            action{
                println "Got to end"
                return "COMPLETED"
            }
        }
    }
}

```

## Def Process Flow

```

process{
    name "DefProcess"

    begin{
        action{

        }
        on "okay" moveTo "defActivity"
    }

    activity{
        name "defActivity"
        action{
            //do something here
        }
        on "okay" moveTo "end"
    }

    end{
        action{
            "COMPLETE"
        }
    }
}

```

## Xyz Process Flow

```

process{
    name "XyzProcess"

    begin{
        action{

        }
        on "okay" moveTo "xyzActivity"
    }

    activity{
        name "xyzActivity"
        action{
            //do something here
        }
    }
}

```

```

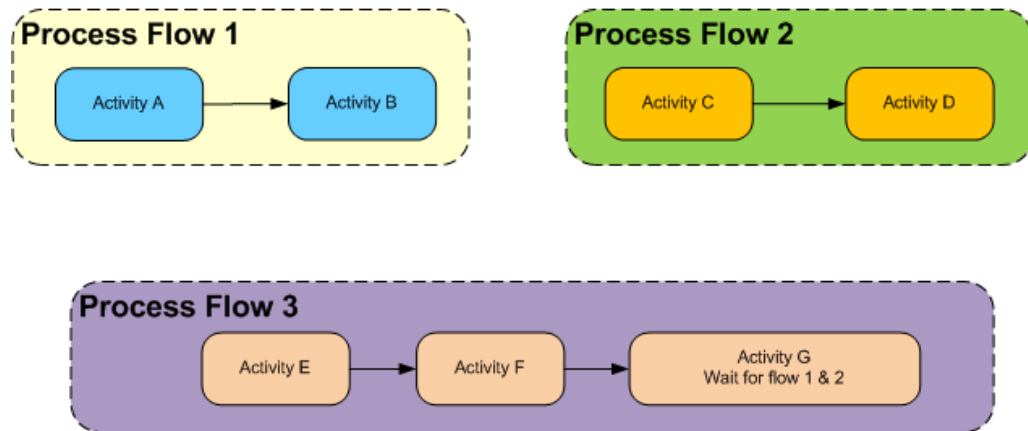
        }on "okay" moveTo "end"
    }
end{
    action{
        "COMPLETE"
    }
}
}

```

## How do I create a join flow with other flows?

Process flow "Def" and Process flow "Xyz" run independently. Process flow "Abc" has to wait until process "Def" and "Xyz" are complete. Use "waitForProcessNamesToReachStatus" to wait for other processes to complete.

### Join Flows



### Sample Join Flow

```

process {
    name "AbcProcess"

    begin{
        action{
        }
        on "okay" moveTo "AbcActivity"
    }

    activity{
        name "AbcActivity"
        action{
            waitForProcessNamesToReachStatus(['DefProcess':2, 'XyzProcess':2],
            now().minusDays(1), PROCESS_COMPLETED, LOGICAL_AND, LAST_EXECUTION_STATUS)
            "okay"
        }
        on "okay" moveTo "end"
    }
}

end{
    action{

```

```

        return "COMPLETED"
    }
}
}

```

## How do I share data between process flows?

Process flow "Abc" needs to share data with process flow "Def".

Use "persistGlobalUserData" and "findGlobalUserData" APIs to share information.

### Sample Flow that shares information with other flows

```

process {
    name "AbcProcess"
    begin{
        action{
        }
        on "okay" moveTo "AbcActivity"
    }
    activity{
        name "AbcActivity"
        action{
            // Persist date as String
            persistGlobalUserData("workDayStart", now().minusDays(1).toString())
            "okay"
        }
        on "okay" moveTo "end"
    }
    end{
        action{
            return "COMPLETED"
        }
    }
}

process {
    name "DefProcess"
    begin{
        action{
        }
        on "okay" moveTo "DefActivity"
    }
    activity{
        name "DefActivity"
        action{
            //fetch the date from db
            def workDayStartString = findGlobalUserData("workDayStart")
            LocalDateTime workDayStartDateObject = LocalDateTime.parse(workDayStartString)
            log.debug "WorkDayStart Global data asString({workDayStartString}) and
            asLocalDateTime({workDayStartDateObject})"
            "okay"
        }
        on "okay" moveTo "end"
    }
    end{
        action{
            return "COMPLETED"
        }
    }
}
}

```

## How do I create a schedule in Scheduler?

1. Download “JosScheduler16.0.0ForAll16.x.xApps\_eng\_ga.zip” and unzip the file.
2. Setup the schedule for the above created process through seed data or UI. See the sample seed data below.
3. Create DSL file for action. DSL is groovy based and groovy or java code can be used in “Action” block. See the sample DSL below.
4. Copy DSL file to “jos-scheduler-home/setup-data/dsl” folder.
5. Run the deployer script from “jos-scheduler-home/bin” folder.

### Sample seed data to create schedule

```
INSERT INTO BDI_SCHEDULE_DEFINITION (schedule_id, schedule_name, schedule_group,
schedule_description, schedule_status, schedule_start_datetime, schedule_type,
schedule_frequency, schedule_notification, schedule_notification_email,
schedule_action_type, schedule_action_definition) VALUES (1, 'Schedule1',
'Schedules', 'Schedule created from seed data. This schedule calls process flow:
AbcProcess.', 'ACTIVE', TIMESTAMP '2016-11-22 00:00:00', 'SIMPLE', 'DAILY',
'ON_SUCCESS,ON_ERROR', 'admin@example.com', 'ASYN', 'Abc.sch')
```

Here are important fields in seed data that need to be considered for the schedule being created.

schedule\_type - SIMPLE. if advanced scheduling is needed, it needs to be created using Scheduler UI

schedule\_start\_datetime - Specify the date and time when to start the schedule, for example, '2016-11-22 10:20:00'

schedule\_frequency - Valid values are: DAILY, HOURLY, WEEKLY, MONTHLY, WEEKDAY, WEEKEND, SATURDAY, SUNDAY, FIRSDAYOFMONTH, LASTDAYOFMONTH, ONCE

schedule\_action\_type - ASYN (asynchronous) or SYNC (synchronous)

schedule\_action\_definition - Name of the schedule action DSL file

### Schedule Action DSL

Each schedule has corresponding schedule action DSL. This will be the action that gets executed when schedule runs.

### Sample Action DSL

The following schedule action starts “AbcProcess” flow by sending a POST request to Process Flow.

```
action {
(Post[externalVariables.processFlowAdminBaseUrl +
"/resources/batch/processes/operator/AbcProcess"]^externalVariables.processFlowAd
minBaseUrlUserAlias) as String
```

## RESTful Services

RESTful services in Scheduler, Process Flow, and Job Admin are secured with SSL and basic authentication. Security credentials are stored in a wallet file during the installation of these components. The following information is stored in BDI\_SYSTEM\_OPTIONS table during the installation process.

baseUrl - URL of the application (Scheduler/Process Flow/Job Admin)

baseUrlUserAlias - Location of the credentials

## Sample seed data created during installation

```
insert into BDI_SYSTEM_OPTIONS VALUES('jobAdminBaseUrl',
'com.oracle.retail.integration_jos-batch-job-admin_war_16.0.0',
'http://localhost:7001/batch-job-admin')
insert into BDI_SYSTEM_OPTIONS VALUES('jobAdminBaseUrlUserAlias',
'com.oracle.retail.integration_jos-batch-job-admin_war_16.0.0',
GET_FROM_WALLET:GET_FROM_WALLET')
```

The information in BDI\_SYSTEM\_OPTIONS table can be accessed through “externalVariables” in the DSL. The APIs that access REST endpoints need “baseUrl” and “baseUrlUserAlias” for authentication.





---

---

## Pre-Implementation Considerations

### Thread Pool Size in WebLogic

If lot of concurrent schedules/process flows/jobs are going to run, increase the thread pool size in WebLogic. This value can be changed for a managed server from WebLogic Admin Console.

Servers -> Server Name -> Tuning -> Advanced -> Self Tuning Thread Maximum Pool Size

### Database Connection Pool Size in WebLogic

If lot of concurrent jobs are going to run, increase the maximum capacity of the connection pool for data sources that are associated with the jobs. The default value is 15. This value can be changed from WebLogic Admin Console.

Services -> Data Sources -> DataSource -> Connection Pool -> Maximum Capacity



---

---

# High Availability Considerations

## High Availability

Modern business application requirements are classified by the abilities that the system must provide. This list of abilities such as availability, scalability, reliability, scalability, audit ability, recoverability, portability, manageability, and maintainability determine the success or failure of a business.

With a clustered system many of these business requirement abilities gets addressed without having to do lots of development work within the business application. Clustering directly addresses availability, scalability, recoverability requirements which are very attractive to a business. In reality though it is a tradeoff, clustered system increases complexity, is normally more difficult to manage and secure, so one should evaluate the pros and cons before deciding to use clustering.

Oracle provides many clustering solutions and options; those relevant to JOS are Oracle database cluster (RAC) and WebLogic Server clusters.

## WebLogic Server Cluster Concepts

A WebLogic Server cluster consists of multiple WebLogic Server managed server instances running simultaneously and working together to provide increased scalability and reliability. A cluster appears to clients to be a single WebLogic Server instance. The server instances that constitute a cluster can run on the same machine, or be located on different machines. You can increase a cluster's capacity by adding additional server instances to the cluster on an existing machine, or you can add machines to the cluster to host the incremental server instances. Each server instance in a cluster must run the same version of WebLogic Server.

In an active-passive configuration, the passive components are only used when the active component fails. Active-passive solutions deploy an active instance that handles requests and a passive instance that is on standby. In addition, a heartbeat mechanism is usually set up between these two instances together with a hardware cluster (such as Sun Cluster, Veritas, RedHat Cluster Manager, and Oracle CRS) agent so that when the active instance fails, the agent shuts down the active instance completely, brings up the passive instance, and resumes application services.

In an active-active model all equivalent members are active and none are on standby. All instances handle requests concurrently.

An active-active system generally provides higher transparency to consumers and has a greater scalability than an active-passive system. On the other hand, the operational and licensing costs of an active-passive model are lower than that of an active-active deployment.

See the Oracle® Fusion Middleware Using Clusters for Oracle WebLogic Server documentation for more information:

[http://download.oracle.com/docs/cd/E15523\\_01/web.1111/e13709/toc.htm](http://download.oracle.com/docs/cd/E15523_01/web.1111/e13709/toc.htm).

## Scaling JOS

JOS needs to be scaled horizontally to handle large number of concurrent jobs. Single instances of Scheduler and Process Flow can be used since they are not resource

intensive. Job Admin can be very resource intensive. To handle large number of concurrent jobs, multiple instances of Job Admin can be used to distribute jobs. WebLogic Server cluster that consists of multiple managed server instances provide horizontal scalability for Job Admin.

## JOS on Cluster

As recommended above, for scaling JOS for large number of jobs, JOS components should be deployed to cluster. Following are some considerations to be taken into account when deploying JOS on cluster.

## Logging

### Issue

The “System Logs” tab in Scheduler, Process Flow, and Job Admin UIs show only logs from the server that UI is connected to.

### Solution

Use a common log directory for each of the JOS components.

JOS components use the following directory structure for creating log files.

```
$DOMAIN_HOME/logs/<server name>/<app name>
```

Example

```
$DOMAIN_HOME/logs/server1/com.oracle.retail.integration_jos-rms-job-admin_war_16.0.21
```

```
$DOMAIN_HOME/logs/server2/com.oracle.retail.integration_jos-rms-job-admin_war_16.0.21
```

1. Create a common log directory (for example; /home/logs/jobadmin) for each JOS application.
2. Create symbolic links to the common log directory for each server using the below command from \$DOMAIN\_HOME/logs directory.

```
ln -s /home/logs/jobadmin
```

```
server1/com.oracle.retail.integration_jos-rms-job-admin_war_16.0.21
```

```
ln -s /home/logs/jobadmin
```

```
server2/com.oracle.retail.integration_jos-rms-job-admin_war_16.0.21
```

3. If the directory \$DOMAIN\_HOME/logs/<server>/<app> already exists, it needs to be deleted before symbolic link is created.
4. App needs to be restarted after symbolic link is created.

When weblogic managed servers are in different machines a shared network disk has to be used.

## Update Log Level

### Issue

When log level is updated through UI or REST end point, it updates the log level only on the server it is connected to.

### Solution

Log level needs to be updated through the URL of all the nodes in the cluster using UI or REST endpoint.

Example

---

<http://server1:port1/jos-rms-batch-job-admin/system-setting/system-logs>  
<http://server2:port2/jos-rms-batch-job-admin/system-setting/system-logs>

## Create/Update/Delete System Options

### Issue

When system options are created/updated/deleted using UI or REST end point, the changes are reflected only on the server that client is connected to.

### Solution

The reset-cache REST endpoint need to be invoked on the other nodes in the cluster for that application in JOS.

### Example

<http://server1:port1/jos-rms-batch-job-admin/system-setting/reset-cache>

## Create/Update/Delete System Credentials

### Issue

When system credentials are created/updated/deleted using REST endpoint, the credentials are created/updated/deleted only on the server that client is connected to.

### Solution

The REST endpoint that creates/updates/deletes credentials need to be invoked on all the nodes in the cluster for that application in JOS.

### Example

<http://server1:port1/jos-rms-batch-job-admin/system-setting/system-credentials>  
<http://server2:port2/jos-rms-batch-job-admin/system-setting/system-credentials>

## Scheduler Configuration Changes for Cluster

### Cluster Job Scheduler Data Source

1. Specify the data source for schedule timers in the Admin Console
2. Login to Admin Console
3. Click **Environment** → **Clusters**
4. Click on the cluster name
5. Click **Scheduling**
6. Click **Lock & Edit** (For Production Mode only)
7. Select **BatchInfraDataSource** for the **Data Source For Job Scheduler** field
8. **Save**

**WebLogic EJB JAR XML**

1. Update the **weblogic-ejb-jar.xml** in WEB-INF folder of the *bdi-scheduler-ui-  
<version>.war* in *<jos-home>/dist* folder with the contents shown (The entry in red is the change from the existing contents of the file).
2. Instructions to update
3. `cd dist`
4. `jar xf bdi-scheduler-ui-<version>.war WEB-INF/weblogic-ejb-jar.xml`
5. Update the WEB-INF/weblogic-ejb-jar.xml with the contents below
6. `jar uf bdi-scheduler-ui-<version>.war WEB-INF/weblogic-ejb-jar.xml`
7. Delete dist/WEB-INF folder
8. Deploy the scheduler application

```
<weblogic-ejb-jar xmlns="http://xmlns.oracle.com/weblogic/weblogic-ejb-jar"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

```
  <security-role-assignment>
    <role-name>AdminRole</role-name>
    <principal-name>BdiSchedulerAdminGroup</principal-name>
  </security-role-assignment>

  <security-role-assignment>
    <role-name>OperatorRole</role-name>
    <principal-name>BdiSchedulerOperatorGroup</principal-name>
  </security-role-assignment>

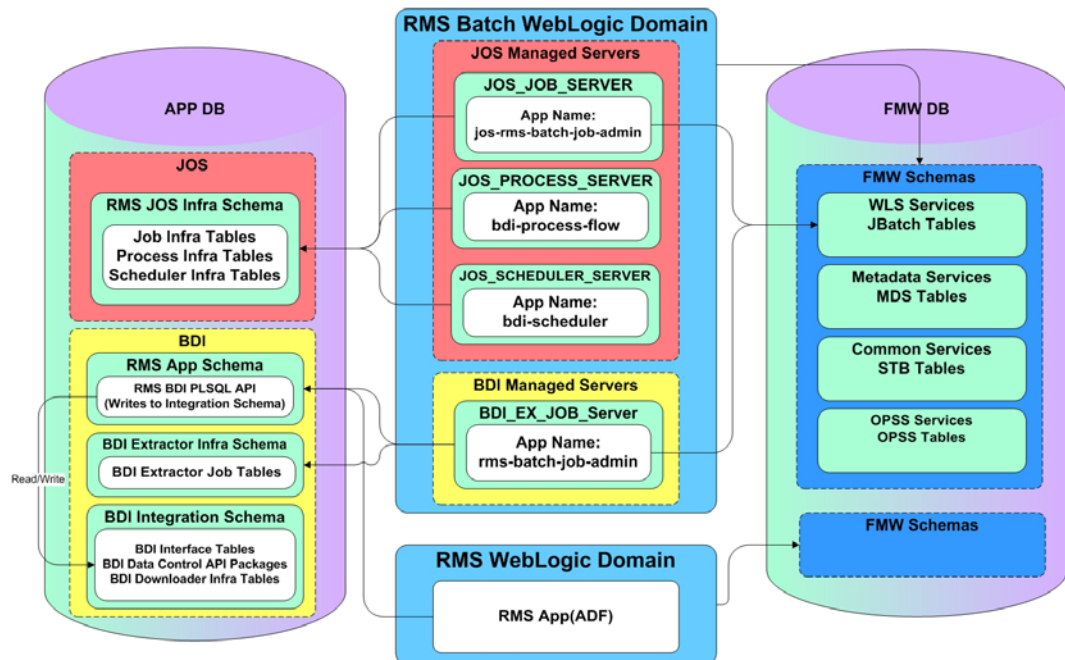
  <security-role-assignment>
    <role-name>MonitorRole</role-name>
    <principal-name>BdiSchedulerMonitorGroup</principal-name>
  </security-role-assignment>
  <timer-implementation>Clustered</timer-implementation>
</weblogic-ejb-jar>
```

# Deployment Architecture

## JOS and BDI deployment architecture for RMS

This diagram shows recommended deployment architecture for RMS that uses both JOS and BDI. Here JOS and BDI use same batch schema as they are deployed in same WebLogic domain. However they use different infrastructure schemas.

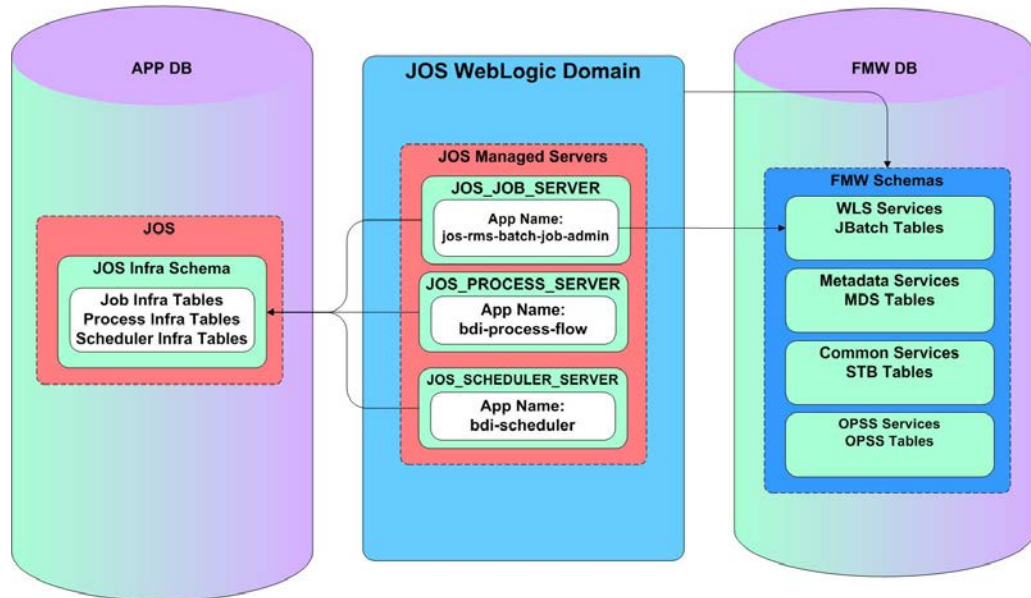
**RMS JOS And BDI Deployment Architecture**



## JOS Deployment Architecture

This diagram shows simple deployment architecture for JOS. In this architecture, JOS Job Admin, JOS Process Flow, and JOS Scheduler are deployed in separate managed servers in a WebLogic domain. This is the recommended architecture if batch jobs are simple and not resource intensive.

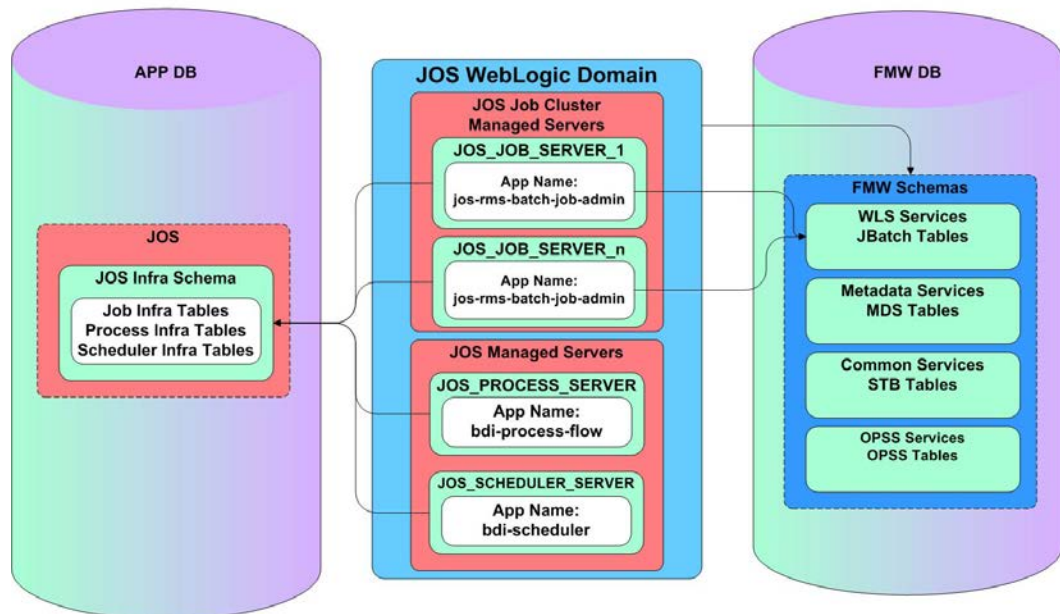
**JOS Deployment Architecture**



**JOS Scalable Deployment Architecture**

This diagram shows scalable deployment architecture for JOS. In this architecture, Job Admin is deployed in multiple managed servers in a cluster. Process Flow and Scheduler are deployed in their own managed servers. This is the recommended architecture if batch jobs are resource intensive. This architecture allows Job Admin to be scaled horizontally and jobs can be distributed.

**JOS Scalable Deployment Architecture**





---

---

## Performance Considerations

### CPU and Memory considerations

- JOS App's memory requirements are low, 1GB should be sufficient. If you are running shell scripts from JOS you will have to make sure whatever memory is needed by your scripts are available in the machine.
- CPU depends on number of concurrent JOBS you plan to run. If you plan to run many process flows concurrently you need to allocate at least that many threads to WebLogic thread pool.
- JavaBatch automatically throttles concurrent jobs based on how many threads are available to the process.



## Appendix A: Scheduler REST Endpoints

Scheduler provides RESTful services to retrieve information about schedules and to run the scheduler manually.

The endpoint “discover” can be used to identify all endpoints provided by the Scheduler.

REST resource	Method	Description
/discover	GET	Lists all the available Scheduler REST resources
/batch/schedules	GET	Returns all the schedules in the application (including active, inactive and disabled schedules)
/batch/schedules/active-schedules	GET	Returns all active schedules
/batch/schedules/{scheduleName}	GET	Returns the schedule definition of the specified schedule
/batch/schedules/upcoming-schedules/days/{days}	GET	Returns the upcoming schedules from now to next number of {days} specified
/batch/schedules/upcoming-schedules	GET	Returns the upcoming schedules for the next 1 day from now
/batch/schedules/executions/{scheduleName}	GET	Returns all the historical schedule executions of the given schedule since the beginning
/batch/schedules/executions/past/days/{days}	GET	Returns the historical schedule executions of the given schedule for past number of {days}
/batch/schedules/executions/failed	GET	Returns all the failed executions for all the schedules since the beginning
/batch/schedules/executions/today	GET	Returns today’s schedule executions starting from midnight today (12:00 a.m.) to now
/batch/schedules/executions/today/completed	GET	Returns today’s schedule executions that are either in ‘Triggered’ status (for async actions) or in ‘Completed’ status (for sync actions), starting from midnight today (12:00 a.m.) to now
/batch/schedules/executions/today/failed	GET	Returns today’s schedule executions that are in ‘Failed’ status, starting from midnight today (12:00 a.m.) to now
/batch/schedules/executions/past/days/{days}	GET	Returns schedule executions for last n days
/batch/schedules/operator/run-schedule-now/{scheduleName}	POST	Runs the specified schedule, that is, executes the Schedule Action of the schedule and returns the Schedule Execution detail response.  This is synchronous invocation, so client needs to wait for the response.

## Appendix B: Process Flow REST Endpoints

The endpoint “discover” can be used to identify all endpoints provided by Process Flow.

REST Resource	HTTP method	Description
/discover	GET	Lists all available endpoints
/batch/processes/operator/{processName}	POST	Start a new Process Flow execution
/batch/processes/executions/{processName}	GET	List Process Executions for the process name
/batch/processes/executions	GET	List all process execution ids
/batch/processes/executions/status/{status}	GET	List all process execution ids for the specified status
/batch/processes/executions/time/{startTime}/{endTime}	GET	List all process execution ids for the specified time range
/batch/processes/external-variables	GET	List external variables
/batch/processes/external-variables	PUT	Create external variables
/batch/processes/external-variables	POST	Update external variables
/batch/processes/external-variables/{key}	DELETE	Delete external variable
/batch/processes/currently-running-processes	GET	List all the currently running process flows
/batch/processes	GET	Get all the available process definitions
/batch/processes/{processName}	GET	Get process DSL for the specified process
/batch/processes/executions/{processName}/{processExecutionId}/activities/{activityExecutionId}	GET	Get Activity execution detail for the activity specified
/batch/processes/executions/{processName}/{processExecutionId}	GET	Get all the activities for the process flow execution
/batch/processes/{processName}/activities	GET	Get all the activities for the process specified
/batch/processes/operator/{processName}/{processExecutionId}	POST	Restart a process execution instance
/batch/processes/operator/{processName}/resolve	POST	Sets the status of process to PROCESS_FAILED
/batch/processes/{processName}/{processExecutionId}	DELETE	Stops running process
/batch/processes/executions	DELETE	Stops all running processes

---

REST Resource	HTTP method	Description
/batch/processes/{processName}/activities/{activityName}	POST	Sets skip, hold flags for activity. Query parameters that can be passed with this end point - "skip", "hold", "actionExpiryDate", "comments".
/batch/processes/{processName}/activities/{activityName}	GET	Returns dynamic configuration for activity



## Appendix C: Job Admin REST Endpoints

Batch service is a RESTful service that provides various endpoints to manage batch jobs in Job Admin. The endpoint “discover” can be used to identify all endpoints provided by Job Admin.

REST Resource	HTTP method	Description
/discover	GET	Lists all available endpoints in Job Admin
/batch/jobs	GET	Gets all available batch jobs
/batch/jobs/{jobName}	GET	Gets all instances for a job
/batch/jobs/{jobName}/executions	GET	Gets all executions for a job
/batch/jobs/executions	GET	Gets all executions
/batch/jobs/currently-running-jobs	GET	Gets currently running jobs
/batch/jobs/{jobName}/{jobInstanceId}/executions	GET	Gets job executions for a job instance
/batch/jobs/{jobName}/{jobExecutionId}	GET	Gets job instance and execution for a job execution id
/batch/jobs/{jobName}	POST	Starts a job asynchronously
/batch/jobs/executions/{jobExecutionId}	POST	Restarts a stopped or failed job
/batch/jobs/executions	DELETE	Stops all running job executions
/batch/jobs/executions/{jobExecutionId}	DELETE	Stops a job execution
/batch/jobs/executions/{jobExecutionId}	GET	Gets execution steps with details
/batch/jobs/executions/{jobExecutionId}/steps	GET	Gets execution steps
/batch/jobs/executions/{jobExecutionId}/steps/{stepExecutionId}	GET	Gets step details
/batch/jobs/job-def-xml-files	GET	Gets all job xml files





## Appendix D: System Setting Service

System Setting service is a RESTful service available in all JOS components (Job Admin, Process Flow and Scheduler) that provides endpoints to manage system option parameters and credentials to be used by the JOS. The system options are stored in BDI\_SYSTEM\_OPTIONS table.

REST Resource	HTTP method	Description
/system-setting/system-options	GET	Gets all system options from BDI_SYSTEM_OPTIONS table
/system-setting/system-options	PUT	Creates a system option in BDI_SYSTEM_OPTIONS table. Only admin user is allowed to perform this operation.
/system-setting/system-options	POST	Updates a system option in BDI_SYSTEM_OPTIONS table. Only admin user is allowed to perform this operation.
/system-setting/system-options/{key}	DELETE	Deletes a system option from BDI_SYSETM_OPTIONS table. Only admin user is allowed to perform this operation.
/system-setting/system-options/{key}	GET	Gets a system option from BDI_SYSTEM_OPTIONS table
/system-setting/system-logs	GET	Gets system logs
/system-setting/system-seed-data	GET	Gets system seed data file
/system-setting/system-seed-data/reset-after-bounce	POST	Resets system seed data after bounce
/system-setting/system-seed-data/reset-now	POST	Resets system seed data now
/system-setting/system-credentials	GET	Gets system credentials. Only admin user is allowed to perform this operation.
/system-setting/system-credentials	PUT	Creates system credentials. Only admin user is allowed to perform this operation.
/system-setting/system-credentials	POST	Updates system credentials. Only admin user is allowed to perform this operation.
/system-setting/system-credentials/{key}	DELETE	Deletes system credentials. Only admin user is allowed to perform this operation.

### Managing System Options using curl

Here are examples of curl commands to list/create/update/delete system options for Process Flow. These commands can be run for Job Admin, and Scheduler as well. Create/update/delete commands can only be run by administrator.

## Create system option

This command creates “reimappJobAdminBaseUrlUserAlias” system option in Process Flow.

```
curl --user userId:password -i -X PUT -H "Content-Type:application/json"
http://server:port/bdi-process-flow/resources/system-setting/system-options
-d '{"key":"reimappJobAdminBaseUrlUserAlias" , "value":"
GET_FROM_WALLET:GET_FROM_WALLET "}'
```

## Update System Option

This command updates “reimappJobAdminBaseUrl” system option in Process Flow.

```
curl --user userId:password -i -X POST -H "Content-Type:application/json"
http://server:port/bdi-process-flow/resources/system-setting/system-options
-d '{"key":"reimappJobAdminBaseUrl" , "value":"http://server:port/reim-batch-job-
admin"}'
```

## Delete System Option

This command deletes “reimappJobAdminBaseUrl” system option from Process Flow.

```
curl --user userId:password -i -X DELETE -H "Content-Type:application/json"
http://server:port/bdi-process-flow/resources/system-setting/system-options
-d '{"key":"reimappJobAdminBaseUrl"}'
```

## Reset System Options Cache

This command resets cache for system options and it needs to be run on all the nodes to refresh cache.

```
curl --user userId:password -i -X POST http://server:port/bdi-process-
flow/resources/system-setting/reset-cache
```

## List System Options

This command lists all system options from Process Flow.

```
curl --user userId:password -i -X GET -H "Content-Type:application/json"
http://server:port/bdi-process-flow/resources/system-setting/system-options
```

## Managing Credentials Using Curl

Here are examples of curl commands to list/create/update/delete credentials for Process Flow. These commands can be run for Job Admin, and Scheduler as well.

Create/update/delete commands can only be run by administrator.

### Create Credential

This command creates a credential in Process Flow.

```
curl --user userId:password -i -X PUT -H "Content-Type:application/json"
http://server:port/bdi-process-flow/resources/system-setting/system-credentials -d
'{"userAlias":" reimappJobAdminBaseUrlUserAlias" , "userName":"reimjobadmin" ,
"userPassword":"xyzxyz"}'
```

### Update Credential

This command updates a credential in Process Flow.

```
curl --user userId:password -i -X POST -H "Content-Type:application/json"
http://server:port/bdi-process-flow/resources/system-setting/system-credentials -d
'{"userAlias":" reimappJobAdminBaseUrlUserAlias" , "userName":"reimjobadmin" ,
"userPassword":"wwwqqq"}'
```

## Delete Credential

This command deletes a credential from Process Flow.

```
curl --user userId:password -i -X GET -H "Content-Type:application/json"  
http://server:port/bdi-process-flow/resources/system-setting/system-credentials -d  
'{"key": "reimappJobAdminBaseUrl}"'
```

## List Credentials

This command lists credentials from Process Flow.

```
curl --user userId:password -i -X GET -H "Content-Type:application/json"  
http://server:port/bdi-process-flow/resources/system-setting/system-credentials
```



# Appendix E: Scheduler UI Screenshots

## Schedule Summary

This is the home page that provides the overall summary of the scheduler runtime.

**Note:** today here indicates the duration from midnight to now. It lists the future schedules that are expected to run in the next 24 hours from now. It also lists the schedule executions that have failed today (from midnight to now).

ORACLE Scheduler Console

Welcome, scheduleradmin  
Wed Nov 30 15:34 CST 2016

Schedule Summary | Manage Schedules | Schedule Executions | System Logs

Schedules and Executions

Total Active Schedules: 37

Schedule Executions Today: 0

Schedule Executions Successful Today: 0

Schedule Executions Failed Today: 0

Upcoming Schedules (37)

Schedule Id	Schedule Group	Schedule Name	Schedule Next Run	Schedule Status
1	CodeDetail	CodeDetail_Fnd_From_RMS_Schedule	Thu Dec 01 00:00:00 CST 2016	ACTIVE
2	CodeHead	CodeHead_Fnd_From_RMS_Schedule	Thu Dec 01 00:05:00 CST 2016	ACTIVE
3	DeliverySlot	DeliverySlot_Fnd_From_RMS_Schedule	Thu Dec 01 00:10:00 CST 2016	ACTIVE
4	Diff	Diff_Fnd_From_RMS_Schedule	Thu Dec 01 00:15:00 CST 2016	ACTIVE
5	Diff	DiffGrp_Fnd_From_RMS_Schedule	Thu Dec 01 00:20:00 CST 2016	ACTIVE
6	FinisherAddr	FinisherAddr_Fnd_From_RMS_Schedule	Thu Dec 01 00:25:00 CST 2016	ACTIVE
7	Inventory	InvAvailStore_Tx_From_RMS_Schedule	Thu Dec 01 00:30:00 CST 2016	ACTIVE
8	Inventory	InvAvailWh_Tx_From_RMS_Schedule	Thu Dec 01 00:35:00 CST 2016	ACTIVE
9	Item	ItemHdr_Fnd_From_RMS_Schedule	Thu Dec 01 00:40:00 CST 2016	ACTIVE

Schedule Executions Failed Today (0)

Schedule Execution Id	Schedule Id	Schedule Name	Schedule Execution Datetime	Schedule Action Execution Status	Schedule Action Execution Log
-----------------------	-------------	---------------	-----------------------------	----------------------------------	-------------------------------

## Manage Schedules - Schedule Executions

Manage Schedules page displays list of all schedules and details of each schedule in Schedule Detail view and corresponding schedule executions in Schedule Executions view for the schedule.

**ORACLE Scheduler Console** Welcome, scheduleradmin  
Wed Nov 30 15:43 CST 2016

Schedule Summary | **Manage Schedules** | Schedule Executions | System Logs

List of Schedules (37)

Filter: Schedule Name Schedules with name like this... Create Schedule

Schedule Id	Schedule Name	Schedule Group	Schedule Start	Schedule Frequency	Schedule Next Run	Schedule Status	Schedule End
1	CodeDetail_Fnd_From_RMS_Schedule	CodeDetail	Sat Mar 12 00:00:00 CST 2016	Daily	Thu Dec 01 00:00:00 CST 2016	Active	Never
2	CodeHead_Fnd_From_RMS_Schedule	CodeHead	Sat Mar 12 00:05:00 CST 2016	Daily	Thu Dec 01 00:05:00 CST 2016	Active	Never
3	DeliverySlot_Fnd_From_RMS_Schedule	DeliverySlot	Sat Mar 12 00:10:00 CST 2016	Daily	Thu Dec 01 00:10:00 CST 2016	Active	Never
4	Diff_Fnd_From_RMS_Schedule	Diff	Sat Mar 12 00:15:00 CST 2016	Daily	Thu Dec 01 00:15:00 CST 2016	Active	Never
5	DiffOrp_Fnd_From_RMS_Schedule	Diff	Sat Mar 12 00:20:00 CST 2016	Daily	Thu Dec 01 00:20:00 CST 2016	Active	Never
6	FinisherAddr_Fnd_From_RMS_Schedule	FinisherAddr	Sat Mar 12 00:25:00 CST 2016	Daily	Thu Dec 01 00:25:00 CST 2016	Active	Never
7	InvAvailStore_Tx_From_RMS_Schedule	Inventory	Sat Mar 12 00:30:00 CST 2016	Daily	Thu Dec 01 00:30:00 CST 2016	Active	Never
8	InvAvailWh_Tx_From_RMS_Schedule	Inventory	Sat Mar 12 00:35:00 CST 2016	Daily	Thu Dec 01 00:35:00 CST 2016	Active	Never
9	ItemHdr_Fnd_From_RMS_Schedule	Item	Sat Mar 12 00:40:00 CST 2016	Daily	Thu Dec 01 00:40:00 CST 2016	Active	Never

Schedule Detail | Schedule Executions

[No ScheduleExecutions Found]

Schedule Execution Id	Schedule Id	Schedule Name	Schedule Execution Datetime	Schedule Action Execution Status	Schedule Action Execution Log
[No ScheduleExecutions Found]					

## Manage Schedules - Create Schedule

The 'Create Schedule' option displays one page where user can enter and save all required information to create a schedule.

**Create Schedule**

**Basic Info**

Schedule Group: None

Schedule Name: New Schedule 38

Schedule Description:

**Schedule Action**

Async  Sync

action { }

**Frequency**

Schedule Start Datetime: 2016-12-02T10:23

Schedule End Datetime:  Never  On

Simple Scheduling | Advanced Scheduling

Schedule: Daily

**Notification**

When schedule execution  Starts  Fails  Triggered / Completed

Email: Enter email (separate multiple emails by comma)

Save Cancel

## Schedule Executions

From the Schedule Executions page user can view all available historical schedule executions. The page will display schedule executions for the last one week by default. User can use the search option to enter a different date range to fetch the corresponding schedule executions.

ORACLE Scheduler Console Welcome, scheduleradmin  
Wed Nov 30 15:43 CST 2016

Schedule Summary | Manage Schedules | Schedule Executions | **System Logs**

Schedule Executions From: 2016-11-23T15:43 To: 2016-11-30T15:43

[No ScheduleExecutions Found]

Filter: Schedule Name

Schedule Execution Id	Schedule Id	Schedule Name	Schedule Execution Datetime	Schedule Action Execution Status	Schedule Action Execution Log

## System Logs

The System Logs page displays list of all schedule log files and log contents. Each schedule has its own log file enabling easy access for the user to view the execution logs and other information from the log files for diagnosing and troubleshooting issues.

ORACLE Scheduler Console Welcome, scheduleradmin  
Wed Nov 30 15:44 CST 2016

Schedule Summary | Manage Schedules | Schedule Executions | **System Logs**

Scheduler Log Files

Log File Name	Size (in KB)	Last Modified
FinisherAddr_Fnd_From_RMS_Schedule.log	0.98	Wed Nov 30 15:34:27 CST 2016
UdalltemDate_Fnd_From_RMS_Schedule.log	0.98	Wed Nov 30 15:34:27 CST 2016
ItemSupplier_Fnd_From_RMS_Schedule.log	0.99	Wed Nov 30 15:34:27 CST 2016
CodeDetail_Fnd_From_RMS_Schedule.log	0.98	Wed Nov 30 15:34:27 CST 2016
UomConversion_Fnd_From_RMS_Schedule.log	0.99	Wed Nov 30 15:34:27 CST 2016
Wh_Fnd_From_RMS_Schedule.log	0.96	Wed Nov 30 15:34:27 CST 2016
RelatedItem_Fnd_From_RMS_Schedule.log	8.12	Wed Nov 30 15:34:27 CST 2016
PackItem_Fnd_From_RMS_Schedule.log	0.97	Wed Nov 30 15:34:27 CST 2016
ItemSuppUom_Fnd_From_RMS_Schedule.log	0.98	Wed Nov 30 15:34:27 CST 2016

Log Content

```

2016-11-30T15:34:27,316 [[STANDBY] ExecuteThread: 'S' for queue: 'weblogic.kernel.Default (self-tuning)'] INFO SchedulerStartupServiceBean - Creating schedule: 6 -
FinisherAddr_Fnd_From_RMS_Schedule
2016-11-30T15:34:27,321 [[STANDBY] ExecuteThread: 'S' for queue: 'weblogic.kernel.Default (self-tuning)'] DEBUG CalendarScheduleTimerBean - Created daily schedule timer
2016-11-30T15:34:27,322 [[STANDBY] ExecuteThread: 'S' for queue: 'weblogic.kernel.Default (self-tuning)'] INFO CalendarScheduleTimerBean - Schedule created - ScheduleId: 6 -
ScheduleName: FinisherAddr_Fnd_From_RMS_Schedule - Frequency: [second=0;minute=25;hour=0;dayOfMonth=*;month=*;dayOfWeek=*;year=*;timezoneID=null;start=Sat Mar 12 00:25:00 CST
2016;end=null]
2016-11-30T15:34:27,323 [[STANDBY] ExecuteThread: 'S' for queue: 'weblogic.kernel.Default (self-tuning)'] INFO CalendarScheduleTimerBean - Scheduled - ScheduleId: 6 -
ScheduleName: FinisherAddr_Fnd_From_RMS_Schedule - Schedule First Run at: 2016-12-01T00:25:00.323-0600

```





## Appendix F: Process Flow UI Screenshots

### Process Flow Live

Process Flow Live tab shows the details of the currently running processes. The first section shows the summary of all processes running in the system. The next section shows the list of all processes running since midnight. The last section shows the activity details of the selected process.

### Manage Process Flow - Process Flow Executions

Manage Process Flow tab allows to start a process flow, restart a failed process flow, view/edit a process flow, list the executions instances of a process flow, and view/edit the process flow configuration. A failed process flow instance can be restarted only if it is the latest failed instance and there are no successful executions after that.

## Manage Process Flow - Process Flow Configurations

Process Flow Configurations tab allows to set skip/hold flags on activities in a flow.

ORACLE Process Flow Admin Console Welcome, processadmin  
Wed Nov 30 12:03 EST 2016

Process Flow Live | **Manage Process Flow** | Historical Process Flow Executions | Manage Configurations | System Logs

All Process Definitions

Enter process name to search...

Process Name	Family/Group	Applications	Flow Type	Last Failure	Last Success	Action
Diff_Fnd_ProcessFlow_From_RMS	Diff	RXM	no-split			Run   View   Executions   Configure
DiffGrp_Fnd_ProcessFlow_From_RMS	DiffGrp	RXM	no-split	Wed Nov 30 04:50:29 EST 2016		Run   View   Executions   Configure
InvAvailWh_Tx_ProcessFlow_From_RMS	InvAvailWh	RXM	no-split	Wed Nov 30 05:09:03 EST 2016		Run   View   Executions   Configure
ItemHdr_Fnd_ProcessFlow_From_RMS	ItemHdr	RXM	no-split	Tue Nov 29 01:22:17 EST 2016		Run   View   Executions   Configure
ItemImage_Fnd_ProcessFlow_From_RMS	ItemImage	RXM	no-split	Tue Nov 29 01:30:08 EST 2016		Run   View   Executions   Configure
ItemLoc_Fnd_ProcessFlow_From_RMS	ItemLoc	RXM	no-split	Wed Nov 30 05:39:58 EST 2016		Run   View   Executions   Configure
MerchHier_Fnd_ProcessFlow_From_RMS	MerchHier	RXM	no-split			Run   View   Executions   Configure
OrgHier_Fnd_ProcessFlow_From_RMS	OrgHier	RXM	no-split	Tue Nov 29 23:31:25 EST 2016		Run   View   Executions   Configure

Process Flow Executions For Diff\_Fnd\_ProcessFlow\_From\_RMS:

Activity Name	Action	Action Expiry Date and Time	Comments	Action
begin				
Diff_Fnd_ExtractorActivity	<input type="checkbox"/> Skip Activity <input checked="" type="checkbox"/> Hold Activity		processadmin	Save
Diff_Fnd_ExtractorStatusActivity	<input type="checkbox"/> Skip Activity <input type="checkbox"/> Hold Activity			Save
Diff_Fnd_GetDataSetIdActivity	<input type="checkbox"/> Skip Activity <input type="checkbox"/> Hold Activity			Save

## Manage Process Flow - Launch Process Flow

Launch Process Flow tab allows to start a process flow with provided process parameters.

ORACLE Process Flow Admin Console Welcome, processadmin  
Wed Nov 30 12:03 EST 2016

Process Flow Live | Manage Process Flow | Historical Process Flow Executions | **Manage Configurations** | System Logs

All Process Definitions

Enter process name to search...

Process Name	Family/Group	Applications	Flow Type	Last Failure	Last Success	Action
Diff_Fnd_ProcessFlow_From_RMS	Diff	RXM	no-split			Run   View   Executions   Configure
DiffGrp_Fnd_ProcessFlow_From_RMS	DiffGrp	RXM	no-split	Wed Nov 30 04:50:29 EST 2016		Run   View   Executions   Configure
InvAvailWh_Tx_ProcessFlow_From_RMS	InvAvailWh	RXM	no-split	Wed Nov 30 05:09:03 EST 2016		Run   View   Executions   Configure
ItemHdr_Fnd_ProcessFlow_From_RMS	ItemHdr	RXM	no-split	Tue Nov 29 01:22:17 EST 2016		Run   View   Executions   Configure
ItemImage_Fnd_ProcessFlow_From_RMS	ItemImage	RXM	no-split	Tue Nov 29 01:30:08 EST 2016		Run   View   Executions   Configure
ItemLoc_Fnd_ProcessFlow_From_RMS	ItemLoc	RXM	no-split	Wed Nov 30 05:39:58 EST 2016		Run   View   Executions   Configure
MerchHier_Fnd_ProcessFlow_From_RMS	MerchHier	RXM	no-split			Run   View   Executions   Configure
OrgHier_Fnd_ProcessFlow_From_RMS	OrgHier	RXM	no-split	Tue Nov 29 23:31:25 EST 2016		Run   View   Executions   Configure

Process Flow Executions | Process Flow Configurations | **Launch Process Flow** | Process Flow Details

Launch Process:

Process Name: Diff\_Fnd\_ProcessFlow\_From\_RMS

Process Parameters(Optional):  For eg. keyName1=value1,keyName2=value2,keyName3=value3

Run

## Manage Process Flow - Process Flow Details - Process Details

Process Details tab displays process activities and configuration in a tabular form.

The screenshot shows the Oracle Process Flow Admin Console interface. The top navigation bar includes 'Process Flow Live', 'Manage Process Flow', 'Historical Process Flow Executions', 'Manage Configurations', and 'System Logs'. The main content area is titled 'All Process Definitions' and contains a table of process definitions. The 'Process Details' tab is selected, showing a detailed view of the process 'Diff\_Fnd\_ProcessFlow\_From\_RMS'. The details include the process name, description, and a table of activities.

Process Name	Family/Group	Applications	Flow Type	Last Failure	Last Success	Action
Diff_Fnd_ProcessFlow_From_RMS	Diff	RXM	no-split			Run   View   Executions   Configure
DiffGrp_Fnd_ProcessFlow_From_RMS	DiffGrp	RXM	no-split	Wed Nov 30 04:50:29 EST 2016		Run   View   Executions   Configure
InvAvailWh_Tx_ProcessFlow_From_RMS	InvAvailWh	RXM	no-split	Wed Nov 30 05:09:03 EST 2016		Run   View   Executions   Configure
ItemHdr_Fnd_ProcessFlow_From_RMS	ItemHdr	RXM	no-split	Tue Nov 29 01:22:17 EST 2016		Run   View   Executions   Configure
ItemImage_Fnd_ProcessFlow_From_RMS	ItemImage	RXM	no-split	Tue Nov 29 01:30:08 EST 2016		Run   View   Executions   Configure
ItemLoc_Fnd_ProcessFlow_From_RMS	ItemLoc	RXM	no-split	Wed Nov 30 05:39:58 EST 2016		Run   View   Executions   Configure
MerchHier_Fnd_ProcessFlow_From_RMS	MerchHier	RXM	no-split			Run   View   Executions   Configure
OrgHier_Fnd_ProcessFlow_From_RMS	OrgHier	RXM	no-split	Tue Nov 29 23:31:25 EST 2016		Run   View   Executions   Configure

Activity Name	Description	Current Activity State	State Expiry Date
begin			N.A.
Diff_Fnd_ExtractorActivity		Hold	N.A.

## Manage Process Flow - Process Flow Details - Process DSL

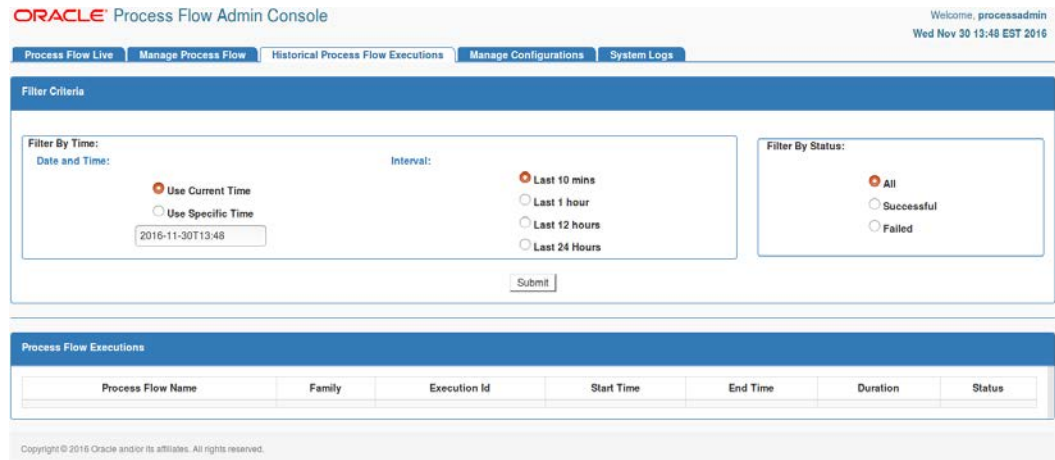
Process DSL tab displays DSL for the selected process flow.

The screenshot shows the Oracle Process Flow Admin Console interface. The top navigation bar is the same as in the previous screenshot. The main content area is titled 'All Process Definitions'. The 'Process DSL' tab is selected, displaying the DSL code for the process 'Diff\_Fnd\_ProcessFlow\_From\_RMS'.

```
//DO NOT DELETE THE BELOW COMMENT.
//destinationSystems=RXM
process {
    name "Diff_Fnd_ProcessFlow_From_RMS"
    var ([processFlowType:"no-split"])
}
```

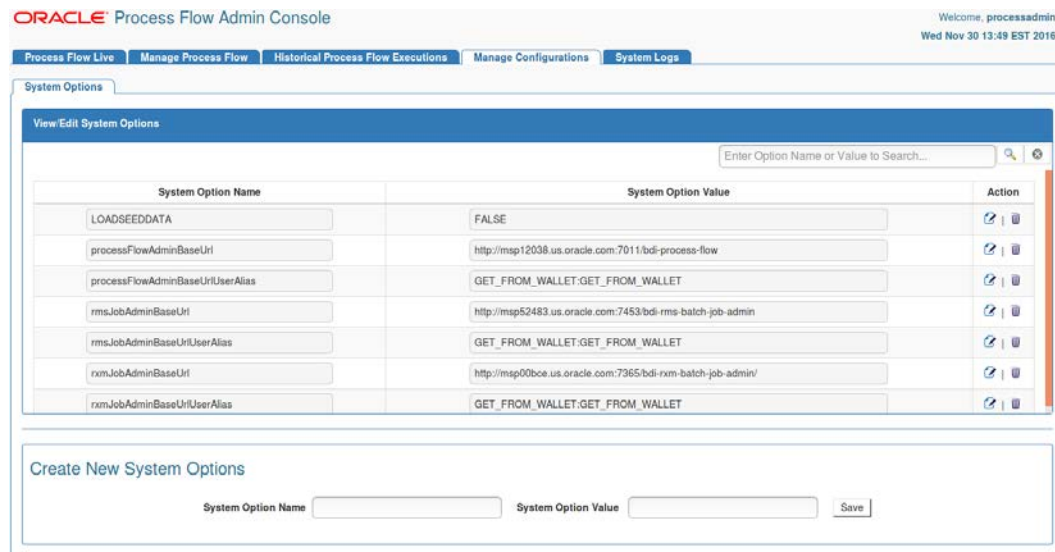
## Historical Process Flow Executions

Historical Process Flow Execution tab allows the user to look at the history of process flow executions. User can specify a date, a time interval and process status. The application will list all the process flow executions matching the criteria. User can select any of the flow to see the activities details of that execution instance. The page also provides option to view the before and after values of all process variables for each activity.



## Manage Configurations

Manage Configurations tab allows to view, edit and create system options. This page displays the list of system options of the application. User can modify the value of the existing system options, create new system options and delete the existing system options. User need admin privileges for editing and creating system options. Search option based on system options name and value is also provided on this page.



## System Logs

The System Logs tab shows all the log files created by process flow execution. Clicking on the View icon will show the log file contents in the screen.

**ORACLE** Process Flow Admin Console Welcome, processadmin  
Wed Nov 30 14:43 EST 2016

Process Flow Live | Manage Process Flow | Historical Process Flow Executions | Manage Configurations | **System Logs**

### Process Log Files

File Name	Size (in KB)	Last Modified
WhAddr_Fnd_ProcessFlow_From_RMS-system.log	275.44	Wed Nov 30 14:23:44 EST 2016
MerchHier_Fnd_ProcessFlow_From_RMS-system.log	4509.26	Wed Nov 30 14:18:59 EST 2016
DITGrp_Fnd_ProcessFlow_From_RMS-system.log	8886.79	Wed Nov 30 14:18:59 EST 2016
ItemHdr_Fnd_ProcessFlow_From_RMS-system.log	4651.97	Wed Nov 30 14:18:59 EST 2016
bdi-default.log	600.42	Wed Nov 30 14:07:43 EST 2016
ItemLoc_Fnd_ProcessFlow_From_RMS-system.log	1115.15	Wed Nov 30 08:58:02 EST 2016
RelatedItem_Fnd_ProcessFlow_From_RMS-system.log	730.25	Wed Nov 30 06:16:43 EST 2016
StoreAddr_Fnd_ProcessFlow_From_RMS-system.log	206.67	Wed Nov 30 05:52:51 EST 2016
InvAvailWh_Tx_ProcessFlow_From_RMS-system.log	546.96	Wed Nov 30 05:14:00 EST 2016

### File Content

```

2016-11-30T09:26:17,705 [[ACTIVE] ExecuteThread: '21' for queue: 'weblogic.kernel.Default (self-tuning)'] INFO ProcessExecutorServiceBean - Starting new
process(WhAddr_Fnd_ProcessFlow_From_RMS) appName(bdi-process-flow-16.0.0.CloudRelease.war).
2016-11-30T09:26:17,709 [[ACTIVE] ExecuteThread: '21' for queue: 'weblogic.kernel.Default (self-tuning)'] INFO ProcessExecutorServiceBean - Finished new
process(WhAddr_Fnd_ProcessFlow_From_RMS) appName(bdi-process-flow-16.0.0.CloudRelease.war).
2016-11-30T09:26:17,850 [Thread-292] DEBUG NativeMethodAccessorImpl - processName(WhAddr_Fnd_ProcessFlow_From_RMS).
2016-11-30T09:26:17,852 [Thread-292] DEBUG NativeMethodAccessorImpl - Since this is first time start so initializing processVariables({processFlowType=no-split}).
2016-11-30T09:26:17,854 [Thread-292] DEBUG Logger$debug - checkBeginCalledFirstAndOnlyOnce.
2016-11-30T09:26:17,863 [Thread-292] DEBUG Logger$debug - =====
2016-11-30T09:26:17,863 [Thread-292] DEBUG NativeMethodAccessorImpl - begin : start
2016-11-30T09:26:17,891 [Thread-292] DEBUG NativeMethodAccessorImpl - Fire
beforeActivityEvent(ActivityStateEvent{activityState=com.oracle.retail.bdi.process.dsl.ActivityState(begin=f45a8f77-0493-4683-89d9-3fe938335899, begin,
WhAddr_Fnd_ProcessFlow_From_RMS-64d66a0f-cbb5-48d4-8beb-e0a335a5f56a, 1, ACTIVITY_STARTED, Wed Nov 30 09:26:17 EST 2016, null), processVariables={processFlowType=no-split}}).
2016-11-30T09:26:17,899 [Thread-292] DEBUG ProcessOrchestratorServiceBean - Notify Process Start

```



## Appendix G: Job Admin UI Screenshots

### Batch Summary

This tab shows the summary of the system and details about latest batch job executions. It can be used to quickly find out whether latest jobs are successful or not. The last section of this page displays the step summary of the selected job.

The screenshot shows the Oracle Job Admin UI. The top navigation bar includes 'Batch Summary', 'Manage Batch Jobs', 'Manage Configurations', and 'System Logs'. The 'Batch Summary' tab is active. The page is titled 'ORACLE JOS RMS BATCH JOB ADMIN' and shows a welcome message for 'jobadmin' on 'Thu Dec 01 15:26 CST 2016'.

**System Summary**

Batch Application JOS-RMS	System Health 🟢	Total Jobs 1	Total Executions 1	Total Successful Executions 1	Total Failed Executions 0
------------------------------	--------------------	-----------------	-----------------------	----------------------------------	------------------------------

**Latest Job Executions**

Enter job name to search...

Job Name	Family	Instance Id	Execution Id	Start Time	Status
ShellCommandRunnerBatchlet		1	1	Thu Dec 01 15:25:44 CST 2016	COMPLETED

**Execution Step Summary for ShellCommandRunnerBatchlet Execution Id: 1**

Step Execution Id	Duration	Status	Resource
1	0 Hours 0 Minutes 0 Seconds	COMPLETED	jobs/executions/1/steps/1

### Manage Batch Jobs - Job Executions

This tab displays the list of available jobs with their details and allows to Start a job, Restart failed jobs, list the executions of a job.

This tab shows the executions of the selected jobs. It can be used to restart the failed executions of a job. Restart button is available only for restartable executions in the status column. When user clicks the restart button it is redirected to the job launch tab with restart option and pre-populated value of job parameters from last run of the execution. User can edit the value of the existing parameters and enter new parameters in comma separated format.

**Note:** that url is an infrastructure parameter, user is not allowed to change its value.



The screenshot shows the Oracle JOS RMS Batch Job Admin interface. The top navigation bar includes 'Batch Summary', 'Manage Batch Jobs', 'Manage Configurations', and 'System Logs'. The main content area is divided into two sections: 'All Jobs Definition' and 'Job Executions'. The 'Job Executions' section is active, displaying a table with the following data:

Job Name	Instance Id	Execution Id	Job Parameters	Start Time	End Time	Duration	Status
ShellCommandRunnerBatchlet	1	1	url=http://127.0.1.1:18001/jos-rms-batch-job-admin/resources/batch-jobs/ShellCommandRunnerBatchlet	Thu Dec 01 15:25:44 CST 2016	Thu Dec 01 15:25:44 CST 2016	0 Hours 0 Minutes 0 Seconds	COMPLETED

## Manage Batch Jobs - Job Launch

This tab can be used to launch the jobs. Job Parameters is an optional input from the user to launch the jobs. Multiple job parameters can be entered in comma separated value format.

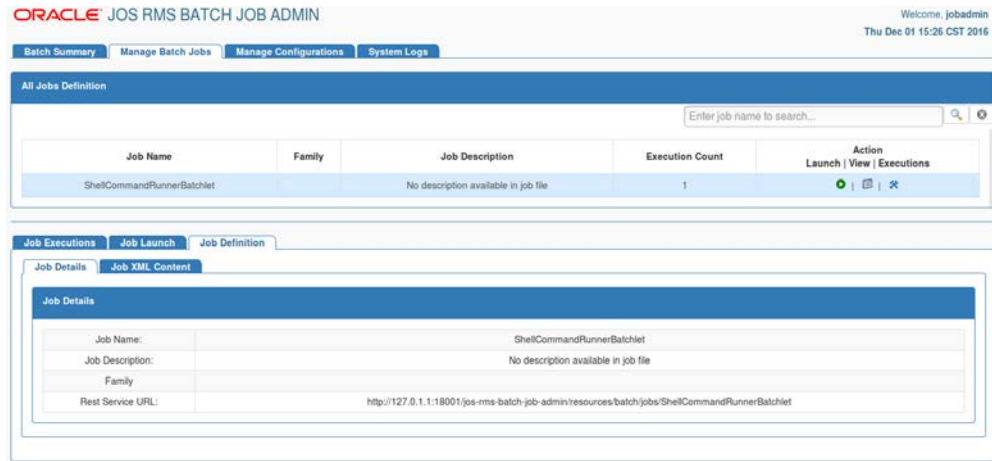
The screenshot shows the Oracle JOS RMS Batch Job Admin interface with the 'Job Launch' tab selected. The 'All Jobs Definition' section shows the 'ShellCommandRunnerBatchlet' job with an execution count of 0. The 'Job Launch' section contains a form with the following fields:

- Job Name:** ShellCommandRunnerBatchlet
- Job Description:** No description available in job file
- Job Parameters (Optional):** A text input field with a placeholder example: "For eg: keyName1=value1,keyName2=value2,keyName3=value3".
- Launch:** A button to initiate the job.

## Manage Batch Jobs - Job Definition - Job Details

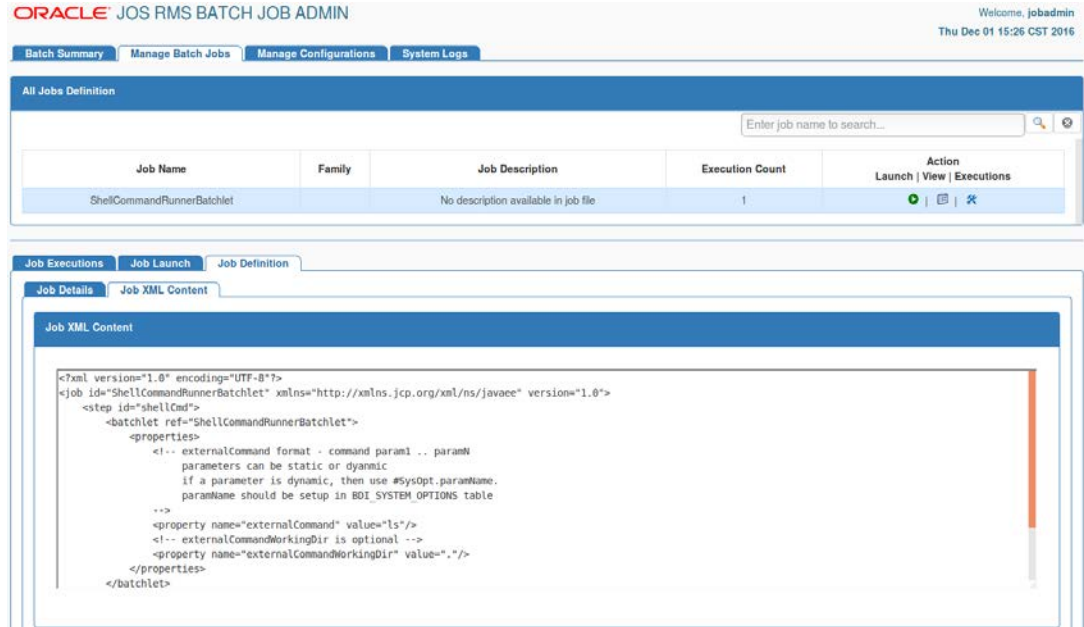
This tab shows the details of the selected job like Job Description, Family, and Rest Service Url.





## Manage Batch Jobs - Job Definition - Job XML Content

This tab shows the details of the selected job XML content.



## Manage Configurations

This tab shows system options from BDI\_SYSTEM\_OPTIONS table. It allows user to add new system options.



ORACLE JOS RMS BATCH JOB ADMIN Welcome, jobadmin  
Thu Dec 01 15:20 CST 2016

Batch Summary | Manage Batch Jobs | Manage Configurations | **System Logs**

System Options

View/Edit System Options

Enter Option Name or Value to Search...

System Option Name	System Option Value	Action
LOADSEEDATA	FALSE	 

Create New System Options

System Option Name  System Option Value

## System Logs

This tab shows logs at job and system level.

ORACLE JOS RMS BATCH JOB ADMIN Welcome, jobadmin  
Thu Dec 01 15:29 CST 2016

Batch Summary | Manage Batch Jobs | Manage Configurations | **System Logs**

Jobs Log Files

File Name	Size(in KB)	Last Modified
ShellCommandRunnerBatchlet-system.log	6.1	Thu Dec 01 15:25:44 CST 2016

File Content

```

2016-12-01T15:25:44,062 [[ACTIVE] ExecuteThread: '0' for queue: 'weblogic.kernel.Default (self-tuning)'] INFO JobOperatorServiceBean - Starting job: ShellCommandRunnerBatchlet
2016-12-01T15:25:44,026 [[ACTIVE] ExecuteThread: '39' for queue: 'weblogic.kernel.Default (self-tuning)'] DEBUG SystemOptionServiceBean - There is no cached data. Either cache was
reset or this is the first call, fetching the data from the DB
2016-12-01T15:25:44,678 [[ACTIVE] ExecuteThread: '39' for queue: 'weblogic.kernel.Default (self-tuning)'] INFO ShellCommandRunnerBatchlet - Starting command(ls) using working
directory(.).
2016-12-01T15:25:44,683 [[ACTIVE] ExecuteThread: '39' for queue: 'weblogic.kernel.Default (self-tuning)'] DEBUG ShellCommandRunnerBatchlet - Command array({ls}).
2016-12-01T15:25:44,752 [Thread-65] DEBUG ShellCommandRunnerBatchlet - ***** BEGIN SYSOUT FOR PROGRAM: ls
2016-12-01T15:25:44,753 [Thread-65] DEBUG ShellCommandRunnerBatchlet - ***** BEGIN SYSOUT FOR PROGRAM: ls
2016-12-01T15:25:44,753 [Thread-65] DEBUG ShellCommandRunnerBatchlet - ***** BEGIN SYSOUT FOR PROGRAM: ls
2016-12-01T15:25:44,753 [Thread-66] DEBUG ShellCommandRunnerBatchlet - ***** BEGIN SYSERR FOR PROGRAM: ls
2016-12-01T15:25:44,754 [Thread-66] DEBUG ShellCommandRunnerBatchlet - ***** BEGIN SYSERR FOR PROGRAM: ls
2016-12-01T15:25:44,768 [Thread-65] DEBUG ShellCommandRunnerBatchlet - EXT_PROG_SYS_OUT:ls: autodeploy
2016-12-01T15:25:44,769 [Thread-65] DEBUG ShellCommandRunnerBatchlet - EXT_PROG_SYS_OUT:ls: bdi_batch_job_infra_create.sql
2016-12-01T15:25:44,769 [Thread-65] DEBUG ShellCommandRunnerBatchlet - EXT_PROG_SYS_OUT:ls: bdi_jpa1.log
2016-12-01T15:25:44,770 [Thread-65] DEBUG ShellCommandRunnerBatchlet - EXT_PROG_SYS_OUT:ls: bdi_jpa.log
2016-12-01T15:25:44,770 [Thread-65] DEBUG ShellCommandRunnerBatchlet - EXT_PROG_SYS_OUT:ls: bdi_receiver_infra_create.sql
2016-12-01T15:25:44,770 [Thread-65] DEBUG ShellCommandRunnerBatchlet - EXT_PROG_SYS_OUT:ls: bin
2016-12-01T15:25:44,771 [Thread-65] DEBUG ShellCommandRunnerBatchlet - EXT_PROG_SYS_OUT:ls: cmm
    
```