Oracle® Retail Process Orchestration and Monitoring

Implementation Guide

Release 23.2.401.0
F86441-01
October 2023

ORACLE

Oracle Retail Process Orchestration and Monitoring Implementation Guide, Release 23.2.401.0

F86441-01

Contents

Send Us Your Comments

Preface

Audience x
Documentation Accessibility x
Customer Support x
Improved Process for Oracle Retail Documentation Corrections xi
Oracle Retail Documentation on the Oracle Help Center (docs.oracle.com) xi
Conventions xi

1 Introduction

Architecture 1-1
- POM Application 1-2
  - POM Jet UI 1-2
  - Process Services 1-3
  - Execution Engine 1-3
  - Agent 1-3
  - Scheduler 1-3
- POM Agent 1-3
- Execution Sequence 1-4

2 Batch Concepts

Batch Schedule 2-1
- Batch Job 2-1
- Batch Process 2-2
- Batch Flow 2-3
- Batch Cycle 2-3
  - Adhoc / Standalone Cycle 2-3
  - Recurring / Hourly Cycles 2-3
  - Nightly Cycle 2-4

Batch Dependencies 2-4
Batch Schedule Links 2-4
Batch Metadata 2-5
Batch Scheduler Day 2-5

3 Integration

Setting Up a New Batch Schedule in POM 3-1
Invoking POM Entities 3-1
External Status Update (Callbacks) 3-1
Schedule Configuration 3-3
Job Configuration 3-4
Payload Specification 3-6
Payload Examples 3-6
External Dependency 3-7
Configuration 3-8
Releasing Dependency 3-10
Bulk Data Integration Jobs 3-10
Mode 1 : Fire-And-Wait 3-11
Set Mode 3-11
Implications 3-11
Mode 2 : Fire-And-Forget 3-11
Set Mode 3-12
Implications 3-12
Mode 3 : Fire-And-Wait-Later 3-12
Set Mode 3-13

4 Scheduler Tasks

Introduction 4-1
Scheduler Tasks 4-1
Access 4-1
Configuration 4-2
Special Restrictions 4-2
Run at the Start of the Scheduler Day 4-3
Scheduler Task Instances 4-3
Scheduling 4-3
Stage 1 : Transform Task Definitions Into Task Instances 4-3
Stage 2 : Schedule the Enabled Task Instances 4-4
One Nightly per Day 4-5
Execution 4-5

Notifications 4-5
Handling Restarts 4-6
Handling Daylight Savings 4-6
Observations 4-7

5 Custom Entities

Introduction 5-1
APIs 5-2
Lift-and-Shift 5-2

6 Custom Schedules

Introduction 6-1
Set Up the Schedule Spreadsheet 6-1
Configure the New Schedule 6-1
Load the New Schedule 6-2
Schedule the Batch Tasks 6-2
Run Batch 6-2

7 Generic ReST Jobs

Introduction 7-1
Features of Generic ReST Jobs 7-1
Execution Sequence 7-2
Handling Restarts 7-3
Handling Kills 7-3
Applications Notifications 7-3
Defining Custom Job Types 7-3
Batch Schedule Spreadsheet 7-3
POM UI 7-4
Environment-specific Information 7-4
Internal Representation 7-4
Job Parameter Restrictions 7-5
Custom Job Type Endpoint Specifications 7-5
Credentials & Scopes 7-5
Job Start API 7-6
Job Status API 7-6
Job Restart API 7-7
Job Kill API 7-8
Job Log API 7-9

Job System Check API 7-9

8 Export/Import Configuration

Schedule Configuration 8-1
Overview 8-1
Schedule Info Tab 8-2
Schedule Configuration Tab 8-2
Throttling Tab 8-3
Nightly Jobs Configuration Tab 8-4
Recurring Jobs Configuration Tab 8-6
Recurring Flows Configuration Tab 8-8
Adhoc Flows Configuration Tab 8-9
Adhoc Processes Configuration Tab 8-9
Adhoc Jobs Configuration Tab 8-9
Job Dependencies Tab 8-11
Job External Associations Tab 8-12
Scheduling Flows Tab 8-13
Scheduling Adhoc Tab 8-15
Notification Tab 8-17
Custom Entities Configuration 8-17

9 Emails and Notifications

Notifications 9-1
POM Post Install Validation 9-1
Schedule Upgrade 9-1
Schedule Configuration Import 9-1
Customer Modules Synchronization (Retail Home) 9-2
Scheduler Task Execution 9-2
Execution Engine 9-3
Scheduler Day Creation 9-3
Hourly Flow Execution 9-4
Nightly Flow Execution 9-4
Job Execution 9-5
External Integration 9-5
Batch Entity Operations 9-6
General 9-6
Emails 9-6
Schedule Change Summary Email 9-6
Schedule Config Import Summary Email 9-7

Nightly Summary Email 9-7
Hourly Flow Summary Email 9-8
Job Error Email 9-9
Job Start Email 9-10
Job Completion Email 9-11

10 User Roles and OAuth Scopes

11 Invoking POM Services

Oauth Token Generation 11-1
Prerequisite 11-1
Invoke IDCS Token Endpoint 11-1
Invoking the POM Service 11-2
Schedule Information API 11-3
View Batch Schedules 11-3
View Batch Cycles for Schedule 11-4
Batch Execution API 11-5
Execution Request Creation 11-6
Execution Request Status 11-8
Execution Requests for Batch Cycle 11-9
External Dependency API 11-12
Releasing External Dependency 11-12
Utilities API 11-13
Business Date Alignment 11-13
Solution Diagram 11-15
Custom Batch Entities API 11-17
Creating / Updating Custom Flows 11-17
Deleting Custom Flow 11-20
Fetching All Custom Flows 11-21
Fetching a Custom Flow 11-25
Creating / Updating Custom Processes 11-29
Deleting Custom Process 11-31
Fetching all Custom Processes 11-31
Fetching a Custom Process 11-34

A Batch Schedule Spreadsheet Template

Overview A-1
Structure A-1

Tabs A-1
Macro Validation A-2
Schedule Validation Processes A-2
Schedule Upgrade Directives A-2
Tab Definitions A-3
Process Tab A-3
Job Tab A-4
ProcessJob Mapping A-7
Dependency Tab A-8
Flow Tab A-9
FlowProcessMapping Tab A-10
Obsolete Tab A-11
SystemOption Tab A-12
Application Tab A-13
Schedule Tab A-14
ThrottlingConfiguration Tab A-15
InterScheduleDependency Tab A-16
Modules Tab A-17
ExternalDependencies Tab A-18
BatchLinks Tab A-18
JobTypes Tab A-19

Send Us Your Comments

Oracle Retail Process Orchestration and Monitoring Implementation Guide

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

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 http://www.oracle.com.

Preface

The *Oracle Retail Process Orchestration and Monitoring Implementation Guide* describes the requirements and procedures to install this Oracle Retail Product release.

Audience

This Implementation Guide is for the following audiences:

- System administrators and operations personnel
- Database administrators
- System analysts and programmers
- Integrators and implementation staff personnel

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.

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.

This process will prevent delays in making critical corrections available to customers. For the customer, it means that before you begin installation, you must verify that you have the most recent version of the Oracle Retail documentation set. Oracle Retail documentation is available on the Oracle Technology Network at the following URL:

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

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 Help Center (docs.oracle.com)

Oracle Retail product documentation is also available on the following Web site:

https://docs.oracle.com/en/industries/retail/index.html

(Data Model documents can be obtained through My Oracle Support.)

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1 Introduction

The Process Orchestration and Monitoring (POM) application provides comprehensive batch scheduling and monitoring capabilities for Oracle Retail SaaS Applications.

The key features for POM are

- Customizable Batch Schedule – Customers can tweak the Batch Schedule to suit their needs. These tweaks are maintained across Batch Schedule upgrades.
- Support for multiple scheduling modes:
  - Cyclical (or Hourly)
  - Adhoc (or Standalone)
  - End of Day (or Nightly)
- Support for Custom Batch Entities, either through the UI or the public APIs provided.
- Flexible Schedule Invocation options:
  - POM Scheduler - Supports multiple cadences and frequencies that can suit most scheduling requirements
  - Public APIs - Externally triggered
- Ability to integrate with External systems:
  - External Dependencies
  - Publish callbacks to external systems
  - Invoke ReST endpoints on external systems (Custom Job Types)
- Configurable Email Notifications
  - For different phases of Job Execution
  - Emails to alert users when the system is waiting on certain conditions or when Jobs encounter errors
  - Comprehensive summary of the Nightly Cycle
- Programmatic access through Public APIs

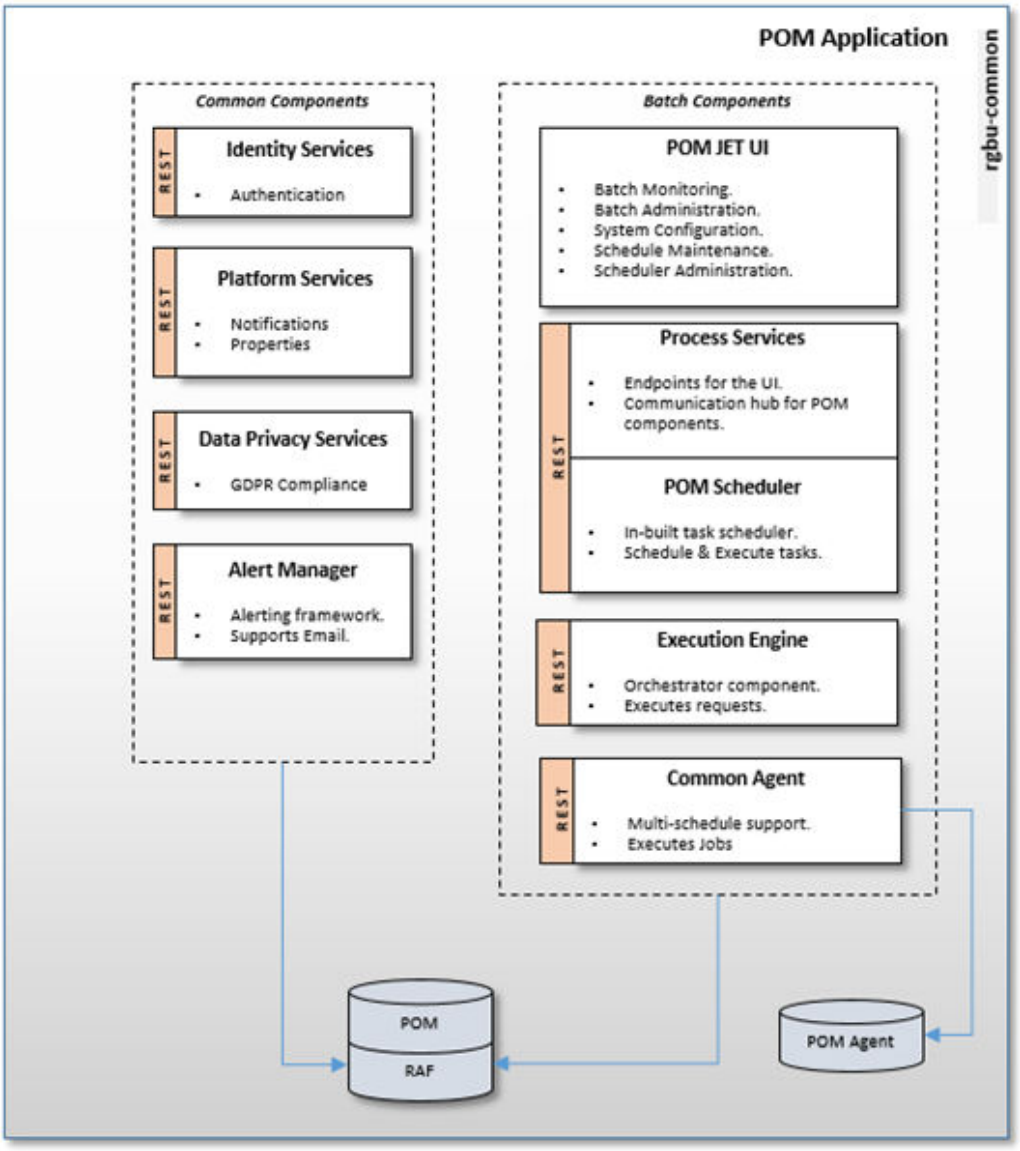
Architecture

From a deployment perspective, the various batch components of POM, can be classified into two logical groups:

- POM Application
- POM Agent

POM Application

The POM application is essentially composed of multiple components. These components are housed in the `rgbu-common` subnamespace.



POM Jet UI

This is a JET Application that serves as the User Interface for POM. It provides screens for the administration of Batch Schedules as well as monitoring screens to keep track of executing batches.

Different POM roles are provided for accessing the features in this application. Refer to User Roles and OAuth Scopes to understand the definition and use of these roles.

Process Services

This provides all the endpoints needed by the POM UI. It also exposes public APIs that can be used to integrate other applications with POM.

Execution Engine

The Execution Engine is the core component that drives all batch executions within POM. Basically, to run any Job in POM, there must be an Execution Request created on the Execution Engine. The Execution Engine then processes the request by creating Job requests on the respective POM Agents.

Agent

The POM Agent is a lightweight, schedule agnostic component that executes jobs requested by the POM application and returns a job status upon job completion or error. For each request received by the Agent, it spawns an executor, based on the Job type, to run the Job associated with that request.

The Common Agent depicted above is deployed within the `rgbu-common` namespace itself. This Agent can be used by multiple Batch Schedules, without needing to deploy a separate Agent within the Consuming Application namespace. The only restriction is that this Common Agent will only support ReST-based Job types.

Customers also have the option to create their own custom schedules and configure those to use this common agent to execute their batch. See Custom Schedules for more information.

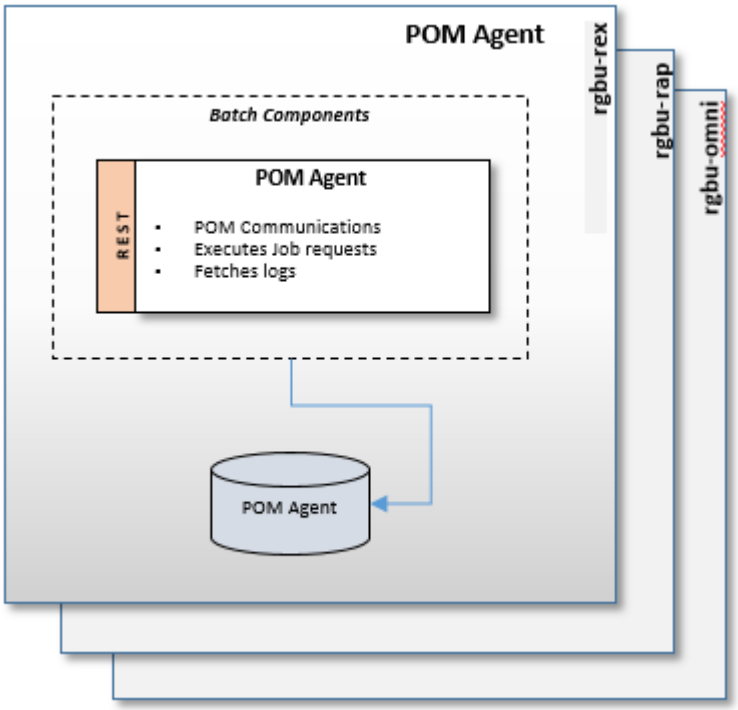
Scheduler

The POM Scheduler is the component that is mainly responsible for the scheduling of tasks within POM. The Scheduler processes tasks by creating Execution Requests on the Execution Engine.

Refer to Scheduler Tasks for further details.

POM Agent

The POM Agent is deployed alongside the Consuming Application (for example, Merchandising) and responds to requests from POM for running Jobs.

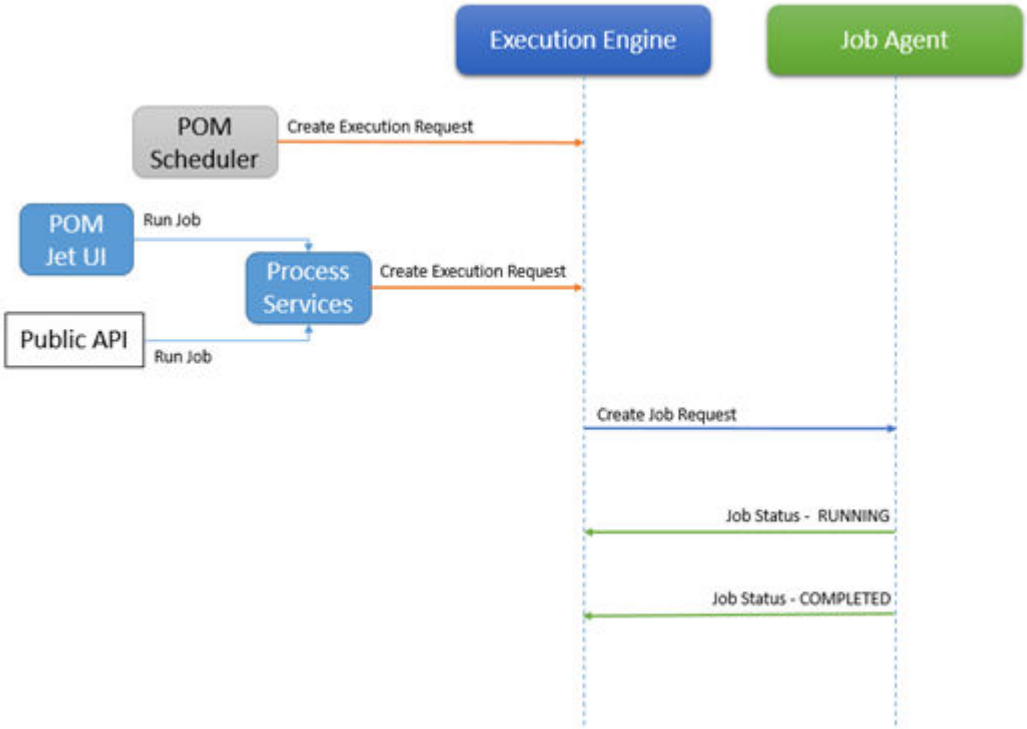


The POM Agent is a repave-aware component that does the following:

- Processes each Job request received by running the respective Job.
- Sends a status update to POM upon Job completion.
- Provides the ability to download Job logs.

Execution Sequence

Multiple components work together to execute batches in POM. The following diagram makes this evident.



- To run any Job in POM, an Execution Request must be created on the Execution Engine:
  - By scheduling a task using the POM Scheduler
  - Run an Adhoc Job, through the POM Jet UI
  - Invocation of public execution APIs
- The Execution Engine handles that request by creating a Job Request on the appropriate Job Agent.
- The Job Agent spawns a new thread to handle the Job Request. This thread dynamically selects an executor based on the Job type associated with the Job Request.
- The Job Agent invokes the Execution Engine to update the Job status periodically.

2

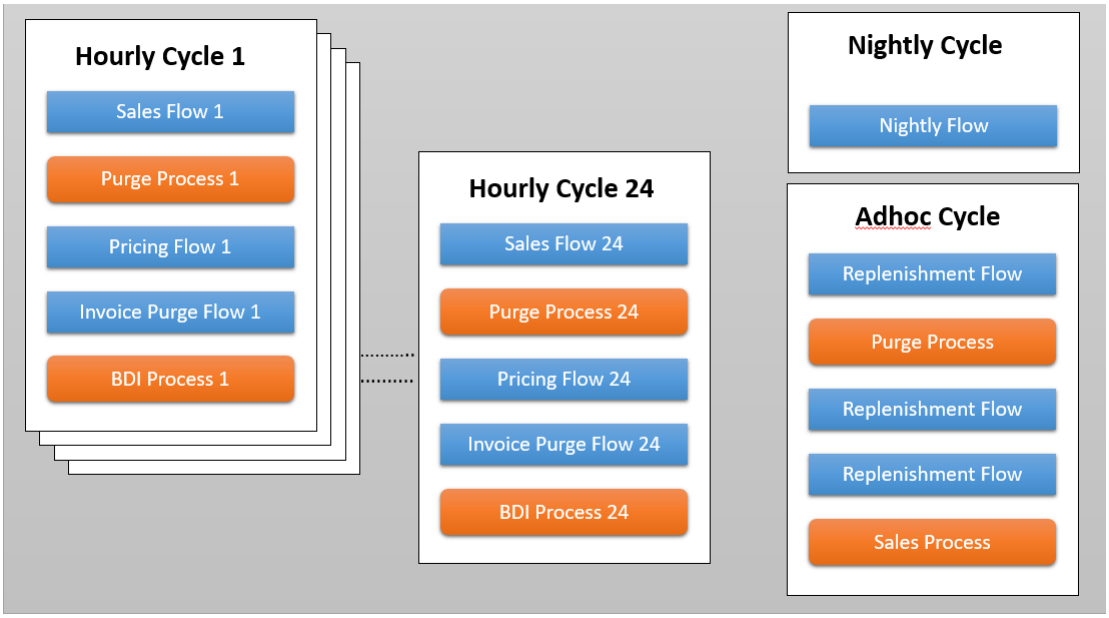
Batch Concepts

This chapter describes concepts in Process Orchestration and Monitoring (POM) that are key to configuring and implementing the product successfully.

Batch Schedule

POM logically groups batch jobs in a batch schedule into a hierarchy of Processes, Flows and Cycles based on functionality and expected execution frequency.

Figure 2-1 Batch Schedule Diagram



Batch Job

A Job is representative of a program that is meant to run and is an atomic unit of work that needs to be accomplished.

The following Job Types are supported by POM out-of-the-box.

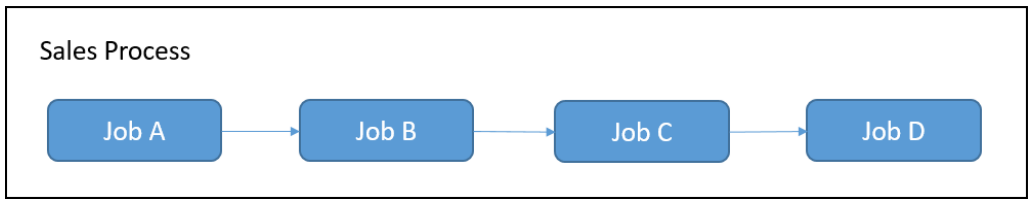
> **Note:**
>
> Other custom job types can be used that are built using the Job Type feature of the System Configuration screen. See Generic ReST Jobs for more information.

Job Type	Description
EXEC	Refers to shell-script based Jobs. POM is expected to start a shell script and track its execution. This is the default type of Job. If a Job type has not been specified on the batch schedule spreadsheet, it is assumed to be of this type.
BDI (Bulk Data Integration)	Indicative of BDI Jobs. POM invokes a BDI Process and tracks its execution for this type of Jobs.
RI (Retail Insights)	Refers to DIS (Data Intelligence Services) Jobs for the RI Schedule. POM invokes RI endpoints and tracks their execution.
RASE (Retail Advanced Science Engine)	Refers to DIS (Data Intelligence Services) Jobs for the RSP Schedule. The wrapper endpoints invoked here are the same as the ones used by the RI job type.
RPAS (Retail Predictive Application Server)	Indicative of RPAS-WebService Jobs for the RPAS schedule. POM invokes RPAS endpoints to start and track RPAS batches.
OB (Order Broker)	Indicative of OB-WebService Jobs for the OB schedule. POM invokes OB endpoints to start and track OB batches.
OMS (Order Management System)	Indicative of OMS-WebService Jobs for the OMS schedule. POM invokes OMS endpoints to start and track OMS batches.
DUMMY	Mock Job Type that doesn't call any executable. It is used for Jobs that are meant to be used as placeholders. Such Jobs are automatically skipped by the Execution Engine.
RDS (Retail Data Services)	Indicative of RDS Jobs. POM invokes RDS endpoints to start and track these Jobs.

Batch Process

A Batch Process is a collection of related Batch Jobs that will always run in sequential order. The dependencies between the Jobs within a Process ensure that these Jobs cannot be run in parallel.

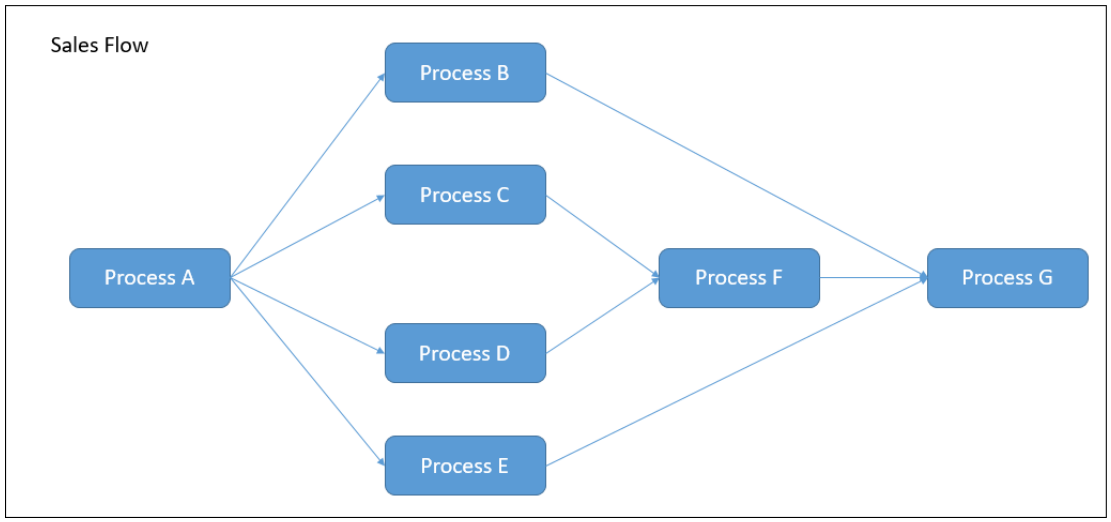
Batch Processes on the Adhoc Cycle can be run from the POM UI or invoked using the ReST API. See Invoking the POM Service in the Integration chapter of this guide.



Batch Flow

A Batch Flow is a collection of related Batch Processes that must be run as a single unit. Within a Batch Flow one Process must be designated as the "first" Process, and another as the "last" Process. Other Processes can be run between them in sequential or parallel fashion, based on the dependencies setup.

Batch Flows can be defined on the Hourly Cycles and the Adhoc Cycle. The Flows on the Hourly Cycle can only be run once, while the Flows on the Adhoc Cycle can be run any number of times. These Flows can be invoked from the POM UI and can also be invoked through the ReST API. See Invoking the POM Service in the Integration chapter of this guide.



Batch Cycle

A Batch Cycle is the container in which Flows, Processes and Jobs are setup. There are three types of Batch Cycles.

Adhoc / Standalone Cycle

The Flows and Processes defined in this Cycle can run multiple times a day on an as-needed basis. Such invocations can run independent of other Processes and Flows on other Cycles.

Example: A Flow can be defined whose main responsibility is to purge data in database tables to boost performance. This Flow can then be run every 15 minutes without impacting Flows or Processes on other Cycles.

Recurring / Hourly Cycles

Recurring or Hourly Cycles are specialized Batch Cycles with the following properties

- A Batch Schedule can have a maximum of 24 Hourly Cycles.
- The Flows and Processes defined on an Hourly Cycle are replicated across all the defined Hourly Cycles. Dependencies are built between these batch entities, to ensure that unless it is completed in a prior cycle, it will not run in the next cycle.

- All the Hourly Cycles are meant to be run prior to running the Nightly Flow. Once the Nightly Flow starts, all the Jobs on the Hourly Cycles are marked completed.

Example: The Sales processing Jobs in RMS support trickle processing by running every 30 minutes during the store trading hours. The schedule is pre-loaded with 24 Hourly Cycles. The Hourly Cycles are time triggered from the Scheduler or through ReST APIs externally. Based on the client's business operations, the individual flows of each cycle need to be scheduled.

> **Note:**
>
> If there are any errors on the Hourly Cycles, then the Nightly Cycle will not start until they are resolved.

Nightly Cycle

This Batch Cycle contains the set of Jobs that are executed at end of the business day. All of these Processes and Jobs belong to the default Nightly Flow of this Cycle. Once the Nightly Flow has started, all the loaded Flows and Processes on the Hourly Cycles are marked complete.

The Nightly Flow can be time-triggered or triggered using Schedule Links. It can be time-triggered through the POM Scheduler or externally through the ReST API.

Batch Dependencies

A dependency is a construct that prevents the execution of a Job, until a condition is satisfied.

Type	Description
Internal Dependency	Internal dependencies are the dependencies between Jobs of the same Batch Schedule.
Inter-schedule Dependency	Inter-Schedule dependencies are the dependencies between Jobs of different schedules running on the same POM instance.
External Dependency	External dependencies are dependencies between processes running on external systems, such as a Customer's system and Jobs running on the POM schedule.

Batch Schedule Links

Schedule Links or Execution Links, allow a Job from one Schedule, to trigger the Nightly Cycle of a different Schedule.

Batch Metadata

All information from the Batch Schedule spreadsheet is referred to as the Batch metadata. This information is modifiable from the POM UI on the Batch Administration Screen.

Batch Scheduler Day

Simply stated, this is a single copy of the Batch metadata or an instance of the Batch Schedule that is associated with a business date. All executions of batches for that business date are recorded on that Scheduler Day or Schedule Instance. Without a Scheduler Day, no Flows or Processes can be run.

A Batch Scheduler Day is marked Closed at the completion of all the Batch Jobs of the Nightly Cycle. The business date is then incremented by one, and a new Scheduler Day is created for the next day.

If however, there are Adhoc Flows or Processes executing when the Nightly Cycle completes, the closure of the Scheduler Day is then deferred until the completion of the running batches.

3
Integration

This chapter describes the various scenarios which involve configuring the Process Orchestration and Monitoring (POM) application and integrating it with external systems.

Setting Up a New Batch Schedule in POM

When POM is first installed for a specific customer, it does not include any application batch schedules out of the box such as Merchandising or Retail Intelligence, and so on. An Oracle administrator or a system integrator need to first configure those schedules before they get loaded with the scheduling data. Configuring a new schedule entails setting up schedule properties such as the schedule name and description, and customer environment information for callbacks. It also entails setting up the location of different components and services with which different POM components need to interact to function properly.

Refer to the section "Configure New Schedule" in the "System Configuration" section of the *POM User Guide*.

Invoking POM Entities

Different SaaS customers operate in different models for running their batch. Some may choose to use the POM Scheduler to schedule the different entities such as Nightly, Recurring or Standalone. Refer to the *POM User Guide* for documentation on the POM Scheduler.

Others may choose to control the time and frequency of batch executions by invoking the provided ReST APIs. See the Batch Execution API section of the Invoking POM Services chapter of this guide for more details.

External Status Update (Callbacks)

The External Status Update feature provides the ability for external systems to optionally register with POM to receive the Job status notifications as a callback to their ReST interface.

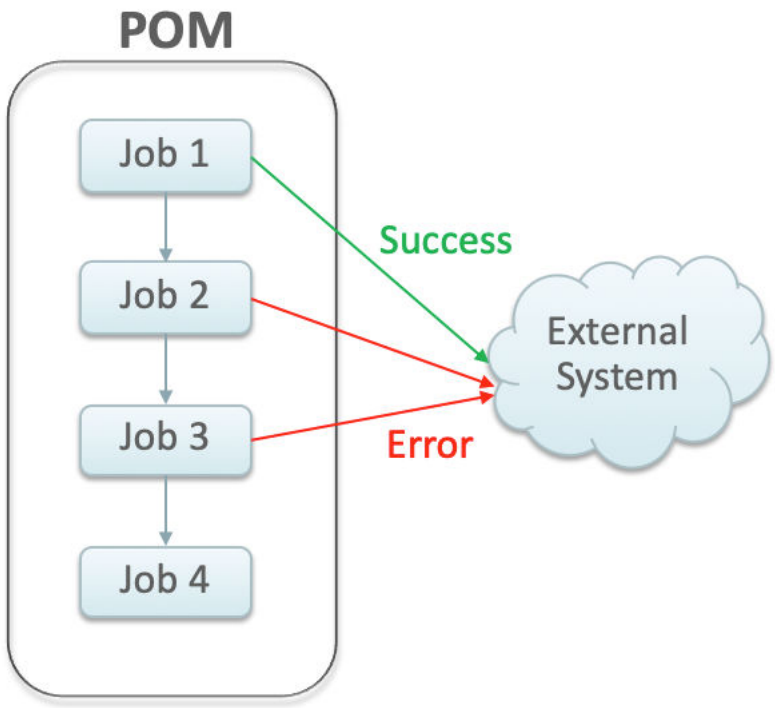
Note:

While ReST service calls from external systems (customers) to POM are required to use the OAuth2 authentication standard, ReST service calls from POM to external systems such as the call for External Status Update are limited to Basic Auth at this time.
POM has a health check feature that calls this ReST service to verify that the external status update (Callback) is up and running. This call uses the HTTP method `OPTIONS` or the `HEAD` method on the provided external status update URL. Customers need to make sure that their external status update API URL implements / allows the `OPTIONS` or `HEAD` method for the POM Health Check to work correctly.

Note:

Retailers are responsible for using valid, certificate authority (CA) signed certificates for TLS for the APIs, such as the one associated with External Status Update. For more information, see My Oracle Support Doc ID 2710163.1.

Figure 3-1 External Status Updates

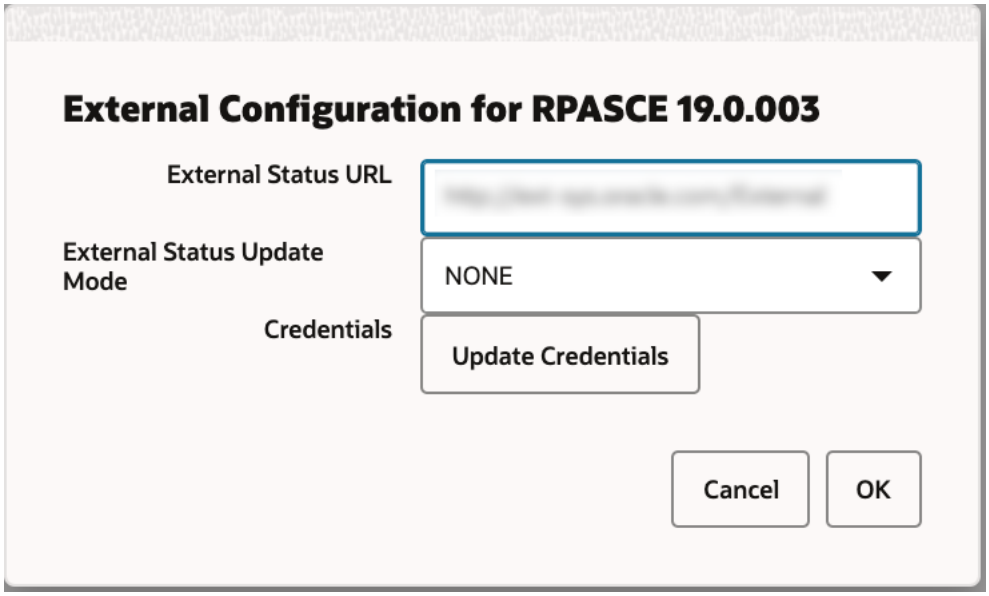


Schedule Configuration

This section details the steps to configure the External Status Updates feature at the schedule level:

1. Navigate to the System Configuration screen.
2. Click the Edit icon on the External Configuration Panel to open the External Configuration window.

Figure 3-2 External Configuration Window



3. Enter the configuration values:
   - **External Status URL** - External system's URL that needs to be called for status updates and also for health-check.

     > **Note:**
     >
     > In addition to this configuration, you must work with Oracle support to get the External Status URL added to the allowlist.
     > The POM health check validates the external systems that must have the `OPTIONS` request method implemented. If the `OPTIONS` method is not implemented, POM will validate the external systems using the `HEAD` request method.

   - **External Status Update Mode** - Choose one of the options below:
     - **ALL** - POM will send a status update to the external system for each job's execution in the schedule regardless of success or failure. This option may be an overuse of this feature and may impact performance.
     - **FAILED** - POM will send a status update only for failed jobs.

- **NONE** - No status updates will be sent by POM.

> **Note:**
>
> The External Status Update Mode defined on this screen applies to all the jobs in a schedule. If status update is desired only for specific jobs then set the mode on the above screen to NONE and follow the steps defined in the Job Configuration section below.

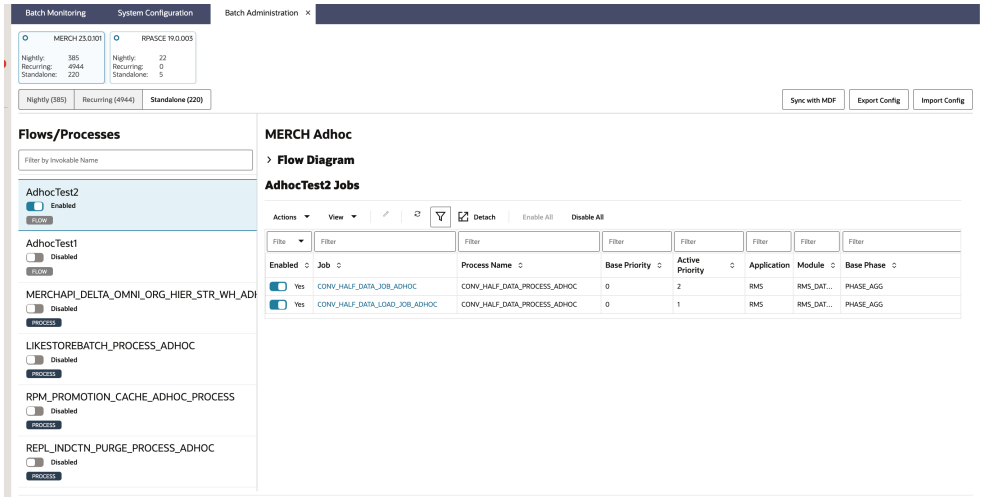
- Click **Update Credentials** and provide the credentials for the external system.

Job Configuration

This section describes the steps to configure the External Status Update Mode at job level.

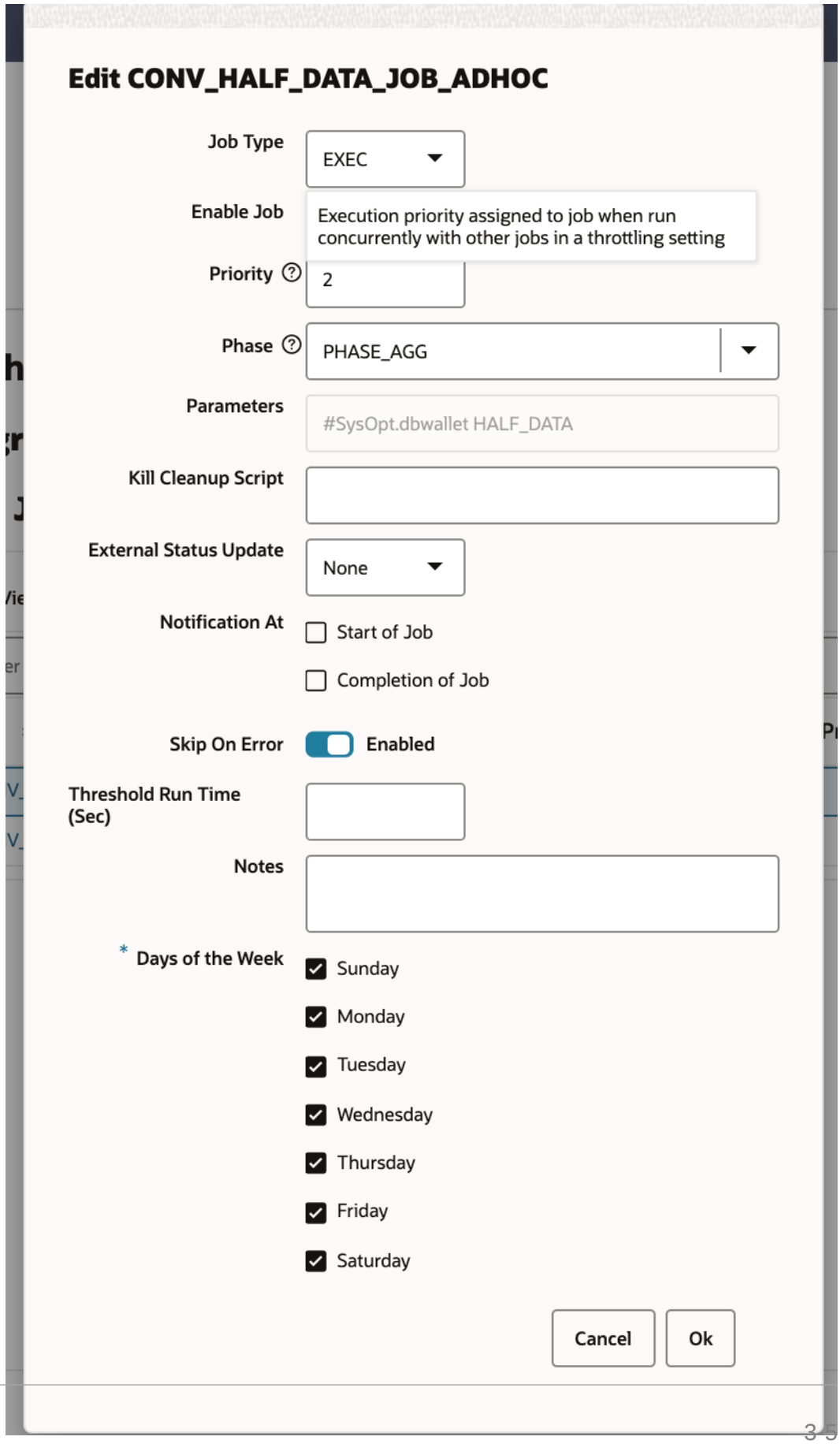
1. Navigate to the Batch Administration screen and select the desired schedule.

Figure 3-3 Batch Administration Screen



2. Select a Cycle - **Nightly/Recurring/Standalone**
   - If Standalone is selected, select the Flow / Process from the left pane, then select the **Process/Job** combination and click **Edit** from table action menu to open the popup below.
   - If Nightly or Recurring is selected, then select the **Process/Job** combination row and click **Edit** from the table action menu to open the popup below.

Figure 3-4 Edit Job Dialog



3. Set the **External Status Update Mode** to one of the following values:
   - **ALL** - POM will send a status update to the external system for this job's execution regardless of success or failure.
   - **FAILED** - POM will send a notification only when this job fails.
   - **NONE** - No status update will be sent by POM for this job.

Payload Specification

Attribute	Description
processName	Name of the root process in a given cycle/flow **Note:** Process names in the callback response are prefixed with the name of the schedule. For instance, a callback response sent for Process "P1" would have `processName` attribute as `MERCH_P1` on return.
processExecutionId	Unique identifier generated by POM to track the process executions.
activityName	Name of the job for which the callback/status update is sent.
activityExecutionId	Unique identifier generated by POM to track the job run instance.
callerId	Identifier provided by the caller to POM when submitting the invocation/execution request. POM returns the same ID back to the caller.
correlationId	Identifier provided by the caller to POM when submitting the invocation/execution request. POM returns the same ID back to the caller
callBackServiceDataDetail.<KeyName>	Key-value pairs supplied to POM when submitting the invocation/execution request. They are returned back to the caller
failedActivity	In the case where the callback is for a failed job, this field is populated with the details of the failed Job.
status	Status of the job execution: • COMPLETED • SKIPPED • ERROR • SKIPPED_ON_ERROR
activityStatus	Status of the job, and the derived activity state: • ERROR - ACTIVITY_FAILED • COMPLETED - ACTIVITY_COMPLETED • SKIPPED - ACTIVITY_COMPLETED • SKIPPED_ON_ERROR - ACTIVITY_COMPLETED

Payload Examples

Below are sample external status update payloads for the `MERCH` schedule.

Description	Payload

Hourly Job Callback	<pre>{
	" callerId " : "XXX",
	" correlationId " : "37",
	" processName " : "MERCH_ BATCH_PROCESS _01",
	" processExecutionId " : "MERCH_ BATCH_PROCESS_01 ~asfasfdasdfas",
	" activityName " : " MERCH_ BATCH_JOB",
	" activityExecutionId " : "123456",
	" status " : "COMPLETED",
	" activityStatus " : "ACTIVITY_COMPLETED",
	" failedActivity " : null
	}</pre>

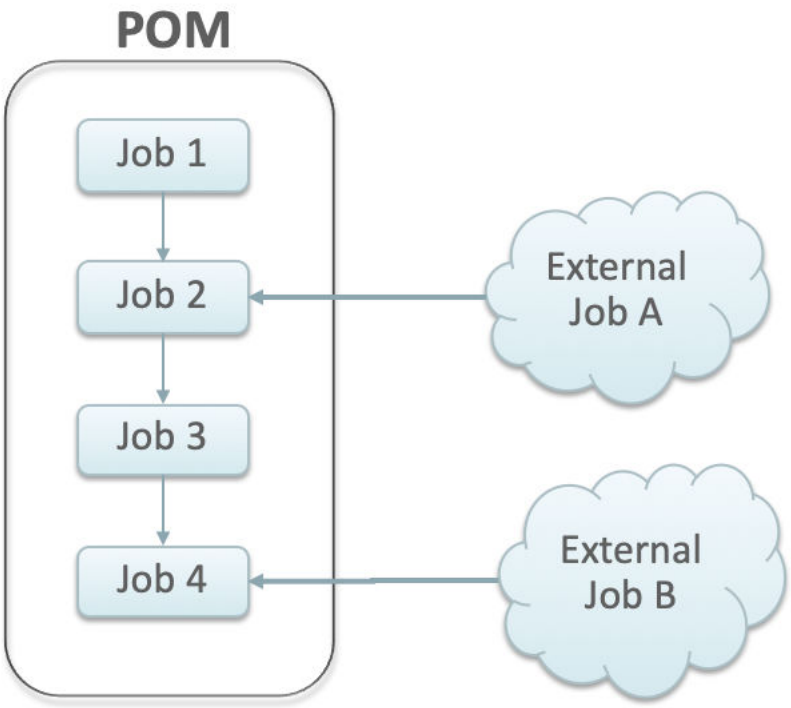
Nightly Job Callback	<pre>{
	" callerId " : "XXX",
	" correlationId " : "37",
	" processName " : "MERCH_ START_NIGHT_BATCH_PROCESS ",
	" processExecutionId " : "MERCH_ START_NIGHT_BATCH_PROCESS ~asfasfdasdfas",
	" activityName " : " MERCH_ START_NIGHT_BATCH_JOB",
	" activityExecutionId " : "123456",
	" status " : "COMPLETED",
	" activityStatus " : "ACTIVITY_COMPLETED",
	" failedActivity " : null
	}</pre>

External Dependency

This feature allows customers to control the execution of a schedule running in POM by defining custom pre-dependencies on certain Jobs. POM pauses the schedule execution upon encountering these external pre-dependencies and resumes the execution once they are released by customer.

> **Note:**
>
> External Dependencies can only be created for Jobs on the Nightly Cycle. They cannot be created for Jobs that belong to the either the Hourly or Adhoc Cycles.

Figure 3-5 External Dependency

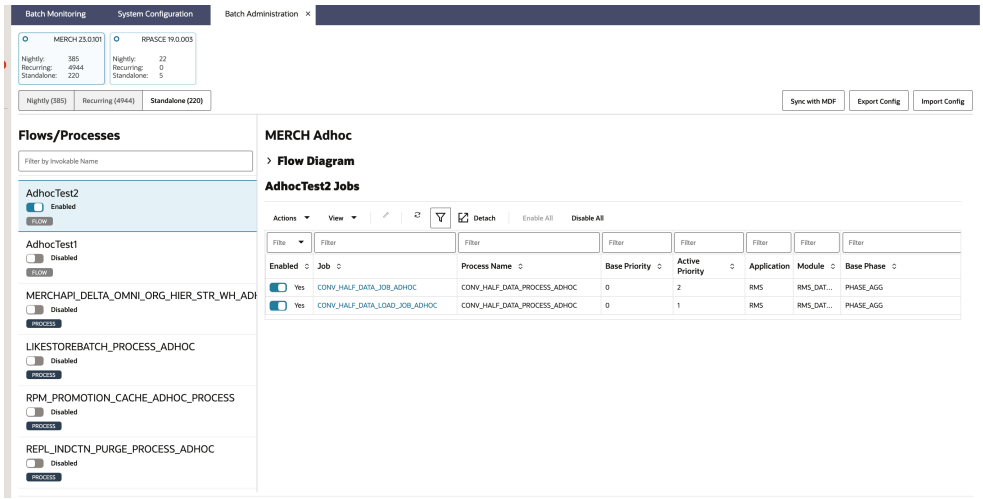


Configuration

This section details the steps involved in setting up the external dependency.

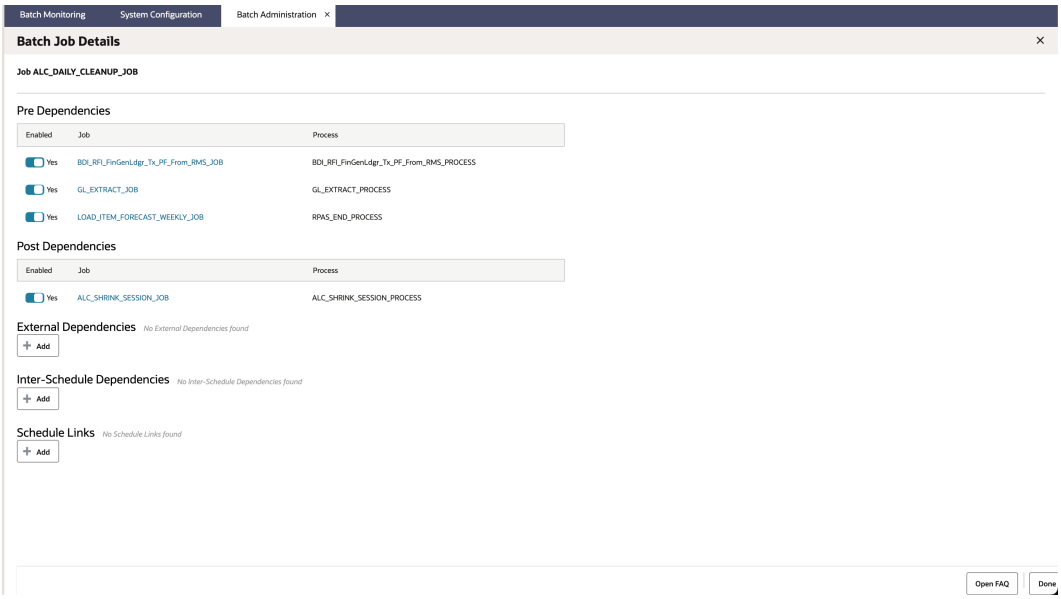
1. Navigate to the Batch Administration screen and select the schedule for which the external dependency is to be added.

Figure 3-6 Batch Administration Screen



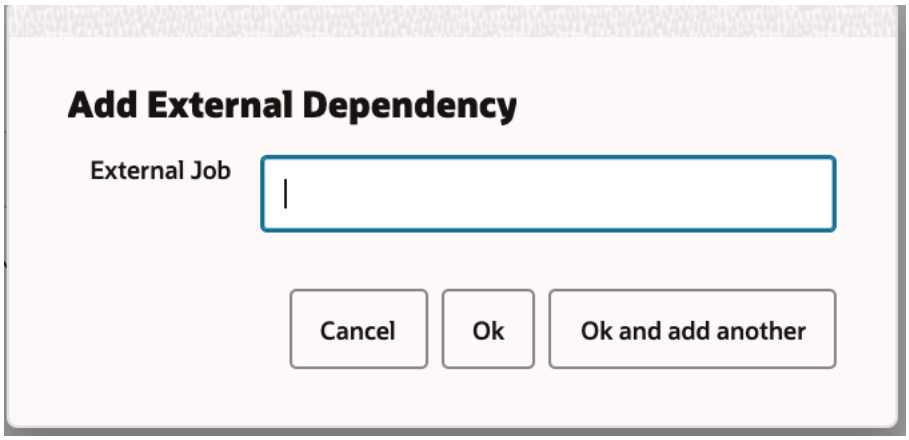
2. Select a Cycle - **Nightly/Recurring/Standalone**.
   - If **Standalone** is selected, select the Flow / Process from the left pane then select the **Process/Job** combination from the right pane to which the dependency needs to be added.
   - If **Nightly** or **Recurring** is selected, then select the **Process/Job** combination row from the table at the bottom of the screen.
3. Click on the **Job** name to open the Batch Job Details panel.

Figure 3-7 Batch Job Details



4. In the External Dependency section of the screen, click the **Add** button to create the external dependency.
5. Provide the external job name.
6. Click **Ok** to save and exit or **Ok and add another** to create another dependency.

Figure 3-8 Add External Dependency



Releasing Dependency

External systems need to invoke the corresponding ReST APIs to release/fulfill external dependencies. See the Releasing External Dependency section of the Invoking POM Services chapter for further details.

Bulk Data Integration Jobs

In POM, Jobs of type BDI, invoke BDI Processes on the BDI Process Flow component.

The specifications followed for the parameters to such Jobs is:

```
[bdi-process-name] [bdi-extractor-name] [callback-params] [filter-param]
```

For example, the parameters to the `BDI_RPAS_Store_Fnd_PF_From_RMS_JOB` on the `MERCH` schedule are:

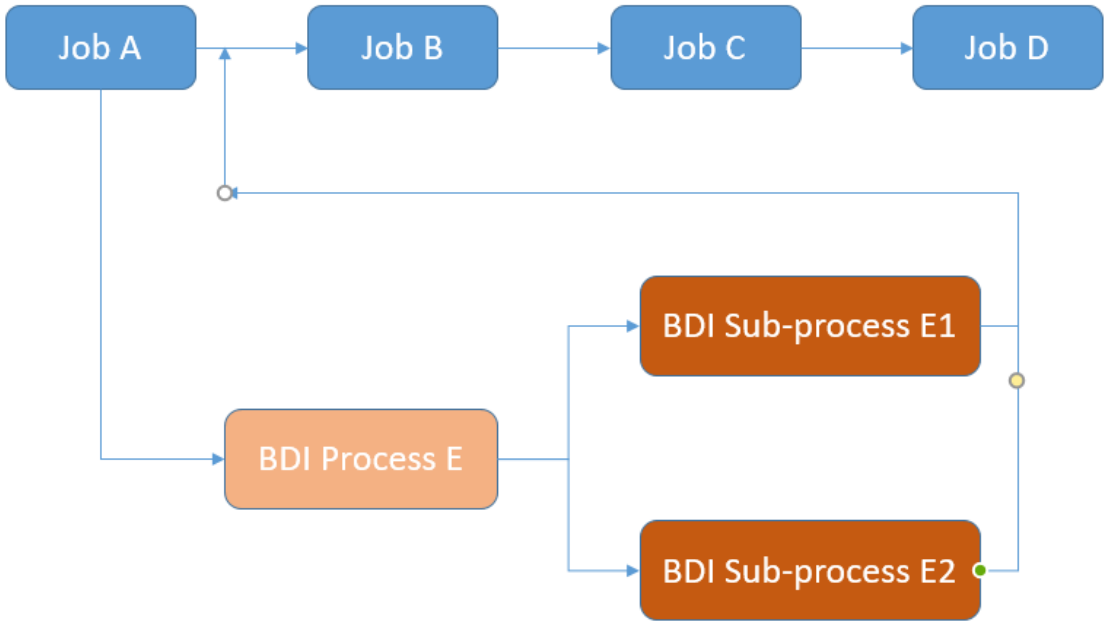
```
Store_Fnd_FileCreator_ProcessFlow_From_RMS Store_Fnd_Extractor #CallbackCtxt.CallerId___#CallbackCtxt.CorrelationId___#JobCtxt.rootProcessName___#JobCtxt.rootProcessExecId___#JobCtxt.jobRunId___#JobCtxt.processName filter=RmsDBDS:RMS_BATCH_STATUS_SQL.GET_EOW_RUN_SIGNAL
```

Parameter Name	Description	Example
`bdi-process-name`	Name of the BDI Process	`Store_Fnd_FileCreator_ProcessFlow`
`bdi-extractor-name`	Name of the Extractor Activity on the BDI Process	`Store_Fnd_Extractor`
`callback-params`	Parameters that need to be propagated to BDI, so that they can be set in the Callback, if configured.	`#CallbackCtxt.CallerId___#CallbackCtxt.CorrelationId___#JobCtxt.rootProcessName___#JobCtxt.rootProcessExecId___#JobCtxt.jobRunId___#JobCtxt.processName`
`filter-param`	This is an optional parameter. The presence of this parameter indicates that this Job must run only if it's the end of week. POM will invoke the MERCH "End-Of-Week" endpoint, and only if it returns true, will the Job actually run; otherwise the Job will be Skipped.	`filter=RmsDBDS:RMS_BATCH_STATUS_SQL.GET_EOW_RUN_SIGNAL`

The integration between POM and BDI is controlled by the parameters passed to the Job, and can be broadly classified into the following three modes:

Mode 1 : Fire-And-Wait

In this mode, the POM Job will trigger the BDI Flow and will proceed to wait for the main BDI Process and its sub-processes to complete. Any error in either the main BDI Process or its Sub-Processes will cause the POM Job to be marked Error.



Set Mode

This mode is set by ensuring that no “extractor” names (second parameter) are passed as parameters to the POM Job that is triggering the BDI Flow.

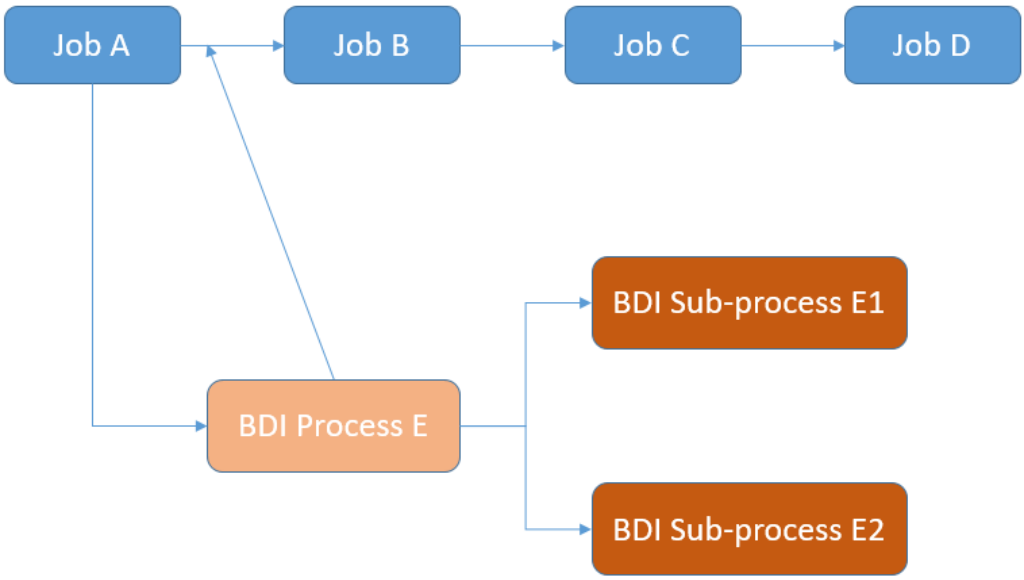
Implications

The impact of running Jobs in this mode are:

- The status of the POM Job reflects the status of the BDI Flow. Guarantees the completion of the BDI Flow before moving on to the next Job in POM.
- Target Batch Duration can be impacted if the BDI Process takes a long time to complete.

Mode 2 : Fire-And-Forget

In this mode, POM Job will trigger the BDI Flow and will only wait for the specified extractor activities of the main BDI Process to complete. If the extractor activities are completed, the Job is assumed to be completed in POM.



Set Mode

This mode is set by ensuring that “extractor” names (second parameter) are passed as parameters to the POM Job that is triggering the BDI Flow.

Implications

The impact of running Jobs in this mode are

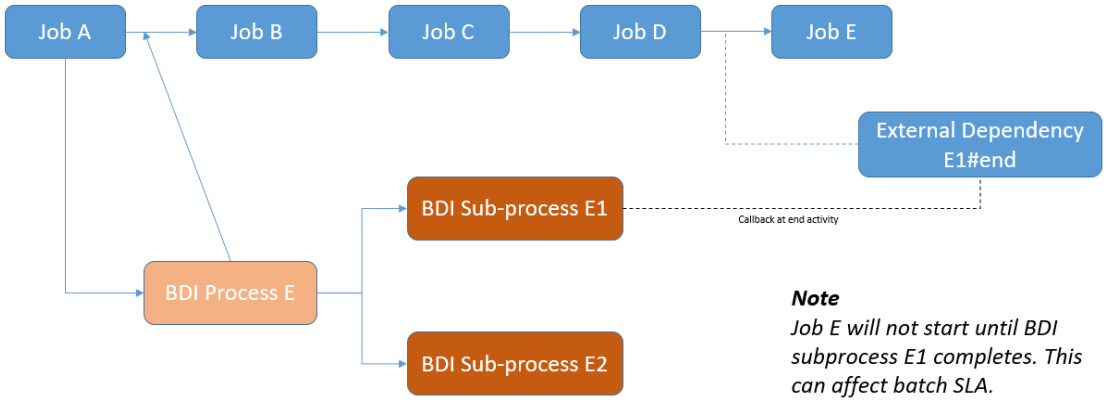
- It is possible that the BDI Flow eventually fails (after the Extractors have completed), while the POM Job is marked Completed. This could possibly impact the integrity of the data.
- It doesn’t cause the remaining batch schedule to wait for a BDI Flow to complete. Non-dependent Jobs can continue processing while the BDI Flow continues to run separately.

Mode 3 : Fire-And-Wait-Later

This is essentially an improved version of the previous mode. In this mode too, the POM BDI job waits only for the extractor activities to complete, to be marked Complete itself. However, a later Job can be configured to wait for the BDI subprocess to complete by configuring an external dependency and routing the callback from BDI to POM.

The external dependency must be configured as follows:

```
{Name of Subprocess} # {end}
```



Set Mode

To set this mode, the configuration needed is as follows:

1. Configure the POM BDI Callback URL on the BDI Process Flow. This ensures that BDI will call POM on the completion of the specified activity.
2. Configure an External Dependency in POM, based on the name of the BDI Subprocess. This external dependency doesn't need to be manually released. On receiving a callback from the BDI Process Flow, POM will auto-release this external dependency. This ensures that the POM Flow is not moved to completion until the relevant BDI Flows have also been completed successfully.

4
Scheduler Tasks

This chapter provides details on the POM Scheduler.

Introduction

POM is equipped with an in-built Scheduler, that can be used to schedule batch entities at different frequencies, times and dates.

Scheduler Tasks

The basic unit of work within the Scheduler, is the **Scheduler Task**. By default , when a Batch Schedule is seeded or patched in POM, Scheduler tasks are automatically created by the system for the following batch entities.

- A Task for the Nightly Flow.
- A Task for each Flow on every Hourly Cycle.
- A Task for every spreadsheet seeded invoke-able (Flow or Process) batch entity on the Standalone Cycle.

> **Note:**
>
> No Scheduler tasks are created automatically for custom batch entities created through the Batch Administration screen's Create Custom Entities feature

Scheduler tasks are created by the system and cannot be removed unless their corresponding invoke-ables are removed.

Similar to Batch Jobs, the Scheduler Task metadata is kept separate from actual Scheduler Task instances. The configuration of a Scheduler Task is considered its metadata or definition, and is used at the creation of the New Scheduler Day to create a Scheduler Task instance that can run for the current Scheduler Day. Within the POM schema, a separate table stores the definition for each task defined for a Schedule.

The Scheduler Task configuration is accessible from the **Scheduler Administration** screen within POM. Refer to the *Scheduler Administration* section of the *POM User Guide* for additional details on how to configure Scheduler Tasks.

Access

A user must access the Scheduler Administration screen to configure the Scheduler Tasks based on the Retailer's needs. To access this screen, this user must have either of the following roles

- BATCH_ADMINISTRATOR_JOB

- BATCH_SCHEDULE_ADMINISTRATOR_JOB

Configuration

A Scheduler Task can be configured at the following frequencies.

Frequency	Description
DAILY	Run daily at the specified time
WEEKLY	Run every <day of a week> at the specified time. The business date of the Schedule is used here, to determine the day of the week.
ONCE	Run once at a specified date and time. The business date of the Schedule is used here, to determine the date at which the task is being run.
MONTHLY	Run on a particular day of each month at a specified time. For example, on the 15th of each month at 10pm The business date of the Schedule is used here, to determine the day of the month.
MONTHLY_START	Runs at the beginning of every month. The business date of the Schedule is used here, to determine if it is the start of the month.
MONTHLY_END	Runs at the end of every month. The business date of the Schedule is used here, to determine if it is the end of the month
MONTHLY_BY_WEEKDAY	Runs on the First/Second/Third.../Last weekday of the month. The business date of the Schedule is used here, to determine whether it is the x day of the month. For example: Second Saturday of the Month

Within each of the frequencies above, the following recurrences are supported.

Recurrence	Description
SINGLE	Will run just once at the time defined on the frequency
MULTIPLE	Will run multiple times at regular intervals starting at the time defined on the frequency

Special Restrictions

- Recurring cycle supports only Daily frequency. It does not support the `MULTIPLE` recurrence.
- Nightly cycle supports only Daily and Weekly frequencies. It does not support the `MULTIPLE` recurrence.
- When Nightly is set up with Weekly frequency, it needs to be set up for every day of the week. This is due to the fact that when Nightly completes for a certain day, it sets up the schedule for the next business day that cannot be skipped.

Run at the Start of the Scheduler Day

The Scheduler schedules a task with no start time defined, to run immediately at the creation of a new Scheduler Day. This can be useful to schedule Jobs that need to run regularly throughout the day and that need to start running right away when the day starts.

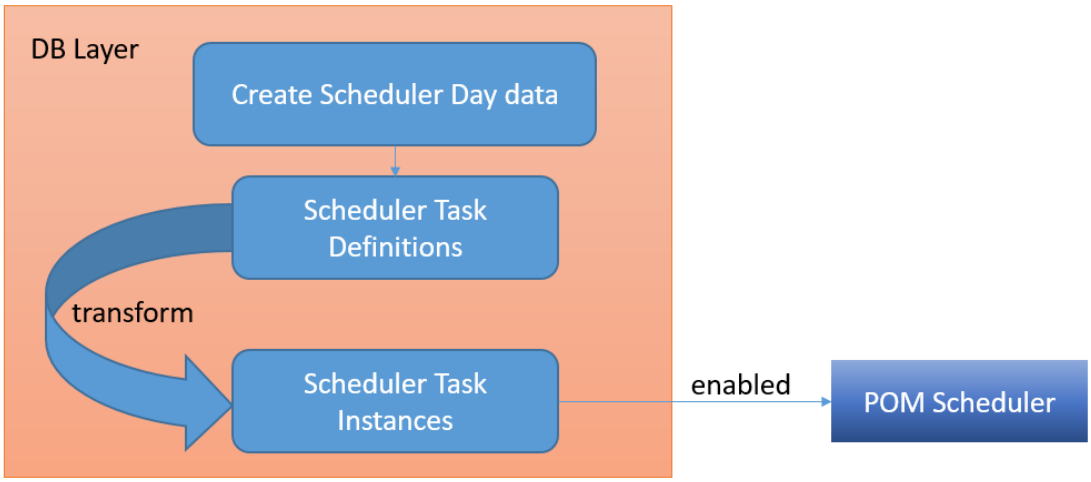
Scheduler Task Instances

Any configuration changes made to Scheduler Tasks are activated from the next Scheduler Day onwards (similar to Batch Job configurations). The Scheduler Task instances for a given Scheduler Day can be seen by clicking on the ***Scheduler Tasks*** link on the **Batch Monitoring** screen. Refer to the *Scheduler Tasks* section within the *Batch Monitoring* chapter of the *POM User Guide* for details.

Scheduling

At the end of the Nightly Cycle, when the Scheduler Day rolls over and a new Scheduler Day is created for the next business date, Scheduler Task instances are created for those Scheduler Tasks that are `ENABLED` and have been configured.

Every instance is given a status and a start time (if configured) and is associated with a corresponding Scheduler Task definition (metadata) and a Scheduler Day. These instances are then processed by the Scheduler based on the configuration that is setup for each of them.



The scheduling of tasks takes place in two stages.

Stage 1 : Transform Task Definitions Into Task Instances

Each Scheduler Task definition has a status and a configuration associated with it. These task definitions are converted into task instances based on the following rules.

- If the task is `DISABLED`, mark the task instance `DISABLED`.
- If task frequency is `ONCE`, then enable the task instance only if the scheduled date matches the business date.

- If task frequency is `WEEKLY`, then enable the task instance only if the business date matches one of the week days selected.
- If task frequency is `MONTHLY`, then enable the task instance only if the business date matches the day of the month selected.
- If task frequency is `MONTHLY_START`, then enable the task instance only if the business date is at the start of a month.
- If task frequency is `MONTHLY_END`, then enable the task instance only if the business date is at the end of a month.
- If task frequency is `DAILY`, then enable the task instance.

Stage 2 : Schedule the Enabled Task Instances

The Scheduler schedules task instances that are in the `ENABLED` status and have valid occurrences in the future. A task with a recurrence of `SINGLE` has one valid occurrence for a day. A task with a recurrence of `MULTIPLE`, may have many occurrences through the day.

For example, the valid occurrences for a task that starts at 3:30 and runs every 15 minutes, for a maximum of 4 times are – 3:30, 3:45, 4:00, 4:15.

The scheduling logic follows the following guidelines:

- If a task has a start time defined, then the Scheduler will try to schedule the task for that time in the specified time zone region for the current day.
  - If the time has passed and the task recurrence is `SINGLE`, then schedule it for the next day.
  - If the time has passed and the task recurrence is `MULTIPLE`, then check if there are any valid occurrences in the future. If they exist, then schedule adjusted remaining occurrences, else schedule for the next day.
- If a task has no start time defined, then schedule it for immediate execution.
- Ensure compliance with daylight savings changes at the specified time zone.

To better understand the scheduling logic around valid occurrences, see the examples below:

Use Cases	Expected Behavior
Task configured to run every 30 minutes, with `startTime` defined as 3:30 PM. No invocation limits specified.	New Scheduler Day – 5:15 PM Occurrences - 5:30 PM, 6 PM and so on.
Task configured to run every 30 minutes, with `startTime` defined as 3:30 PM. Max occurences - 10	New Scheduler Day – 5:15 PM Occurences – 5:30 pm, 6 pm, 6:30 pm, 7 pm, 7:30 pm, 8 pm Notification will be generated as the first 4 occurrences had elapsed

At the end of these two stages, the necessary schedules have been setup.

One Nightly per Day

The Scheduler has an additional validation for Nightly tasks named the **One Nightly per Day** rule. This rule is turned on by default, meaning that the Scheduler will try to ensure that two consecutive Nightly cycles are not run on the same calendar day (based on the time zone set). For example, consider the case where the Nightly cycle is setup with the `WEEKLY` frequency, and is scheduled to run at 10 am on Sunday and then 7 pm on Monday. If this validation is not enforced, the Scheduler would schedule the Nightly cycle at 10 am and then again at 7 pm on Sunday itself.

In such cases, if this rule is in place, the Scheduler bumps the Nightly Cycle to the same time for the next day. A Notification is also created to signal that the Nightly has been adjusted in adherence to this rule.

> **Note:**
>
> As stated previously, the default is to enforce this validation. This behavior can be changed by updating the System Option `EnsureOneNightlyPerDay` and setting it to `FALSE` on the **System Configuration – System Options** screen.

Execution

At the scheduled time, when a Scheduler Task instance fires, the first thing it does is to perform the following validations.

- Ensure the Scheduler is enabled for the Schedule.
- Ensure the Schedule is not in `PATCH` mode.
- Ensure the Scheduler Day associated with the Task instance, is in fact an active Scheduler Day.
- Ensure the Nightly Cycle has not started if the task instance is configured to not run while Nightly is running.
- Ensure the task has not reached its invocation limit.

Once these validations have passed, the task instance performs the following actions.

1. Create a new Execution Request to execute the relevant Batch entity, that was configured.
2. Create a Scheduler Task Run Instance, to record the Execution Request ID or any other information in case of invocation errors.

Notifications

The Scheduler generates the following Notifications, at different points of execution.

Notification Type	Description
SchedulerTaskFailed	This signals that something went wrong, when executing the Scheduler Task Instance.
SchedulerTaskSkipped	On failure to schedule an `ENABLED` task.

Notification Type	Description
SchedulerTaskCancelled	This is created when the scheduler is `DISABLED`, resulting in the scheduler tasks being cancelled.
SchedulerTaskDelayed	Occurs when we try to run the task manually from the Batch Monitoring screen, and the time has passed.
SchedulerTaskOccurenceSkipped	Occurs when the Scheduler has to adjust the invocations for a task instance, as the occurrence time for some invocations has passed.
SingleNightlyPerDayValidation	This is created, when the Scheduler applies the **One Nightly per Day** rule to the Nightly task and bumps it by a day.

Handling Restarts

The Scheduler is capable of re-initializing itself and is therefore able to handle unexpected or scheduled server restarts or repave events. To do so, the Scheduler runs an **Initializer** task every 2 minutes, from when the server starts up. The objectives of this task are

1. Identify all `SCHEDULED` task instances across all the Schedules, and verify that for each of them, a timer exists. In case a timer is found, nothing needs to be done. Otherwise, the timer is created.
2. The **Initializer** also cancels timers that don't have a corresponding task instance in `SCHEDULED` status.

> **Note:**
>
> In cases where the timer needs to be created, the Scheduler uses the same rules stated earlier for scheduling task instances. The Scheduler does what it can to ensure valid occurrences in the future are not missed.

Handling Daylight Savings

The Scheduler makes use of the `ZonedDateTime` classes in the JDK to perform date/time calculations specific to time zone regions. It is therefore necessary for the JDK to be always patched to the latest version, to ensure the latest time zone data is used by the Scheduler.

To better understand the behavior of the Scheduler around Daylight Savings, see the example below.

Timezone Region – Chicago, Illinois, USA

- On Mar 10th, 2024, the clock is adjusted after 1:59:59 AM to 3:00 AM, thereby moving an hour ahead. There is therefore, no 2AM on Mar 10th (Chicago).
- On Nov 5th, 2023, the clock is adjusted after 1:59:59 AM back to 1:00 AM, thereby moving an hour back. Therefore, the times between 1 AM and 2 AM occur twice on this day.

Let us consider, the following tasks scheduled at these times (CST)

- Task 1 – 12:30 am
- Task 2 – 1:00 am
- Task 3 – 1:30 am
- Task 4 – 2:00 am
- Task 5 – 2:30 am
- Task 6 – 3:00 am

Task - Date	Mar 9th	Mar 10th	Nov 4th	Nov 5th
Task 1 (Chicago)	12:30 AM	12:30 AM	12:30 AM	12:30 AM
Task 1 (UTC)	6:30 AM	6:30 AM	5:30 AM	5:30 AM
Task 2 (Chicago)	1:00 AM	1:00 AM	1:00 AM	1:00 AM
Task 2 (UTC)	7:00 AM	7:00 AM	6:00 AM	6:00 AM
Task 3 (Chicago)	1:30 AM	1:30 AM	1:30 AM	1:30 AM
Task 3 (UTC)	7:30 AM	7:30 AM	6:30 AM	6:30 AM
Task 4 (Chicago)	2:00 AM	**3:00 AM**	2:00 AM	**2:00 AM**
Task 4 (UTC)	8:00 AM	**8:00 AM**	7:00 AM	**8:00 AM**
Task 5 (Chicago)	2:30 AM	**3:30 AM**	2:30 AM	**2:30 AM**
Task 5 (UTC)	8:30 AM	**8:30 AM**	7:30 AM	**8:30 AM**
Task 6 (Chicago)	3:00 AM	3:00 AM	3:00 AM	3:00 AM
Task 6 (UTC)	9:00 AM	8:00 AM	8:00 AM	9:00 AM

Daylight Time start – Moves forward an hour

Daylight Time ends – Moves back an hour

Observations

- On Mar 10th, tasks scheduled between 2 AM and 3AM, run as if they were scheduled between 3 and 4 AM.
- On Nov 5th, there exists a 1.5 hour gap between the 1:30 AM and the 2 AM tasks.

5
Custom Entities

This chapter describes Custom Batch Entities (CBEs) and their benefits.

> **Note:**
>
> Support and maintenance for customizations, as well as any batch failures of custom jobs, are the retailer's responsibility. In addition, retailers should carefully evaluate the impact of all customizations to existing schedules, given that these could extend the batch window.

Introduction

Before this feature was implemented in POM, Batch Schedules were immutable, in the sense that they could only be modified by changing the spreadsheet. These Schedules could only be customized with External Dependencies, Schedule Links or Inter-schedule dependencies, and so on. However, there was no way to add a new Batch Process or Batch Job through the POM UI. Additionally, the dependencies between Jobs that are stitched into the Batch Schedule as specified within the Spreadsheet, could not be modified at runtime.

These limitations are quite restrictive, and in order to overcome them, Custom Batch Entities (CBEs) were introduced in POM. A Batch Entity (Flow or Process or Job) that is added to a Batch Schedule, using either the Custom APIs or the POM UI, is a Custom Batch Entity (CBE). These CBEs are distinctly separate from the Batch Entities that are seeded as part of the Batch Schedule spreadsheet. All CBEs added are associated with the Standalone cycle by default.

In POM, the execution APIs support the execution of either a Batch Process or a Batch Flow. These entities are therefore deemed *invoke-able*. The Batch Monitoring screen for the standalone cycle provides an *invoke-able* centered view, rather than a Job centric view seen on the Nightly and Recurring cycles. The Custom APIs and the UI, provide controls for the creation of custom invoke-ables, while custom Jobs can be added as part of these custom invoke-ables.

Type	Features
Custom Batch Job	• The name must end in `_CJOB`, to prevent naming conflicts. • A Custom Batch Job cannot be added as an invokable entity. It can be added as a part of a Custom Batch Process. • Can be deleted from the UI • Can be of any Job Type and has attributes similar to those of the jobs seeded from spreadsheet.
Custom Batch Process	• The name must end in `_CPROCESS` to prevent naming conflicts. • Can contain Custom Jobs as well as existing seeded Jobs. The execution order of these Jobs can be modified. • Can be deleted from the UI

Type	Features
Custom Batch Flow	• The name must end in `_CFLOW` to prevent naming conflicts. • Contains Custom Processes which can contain custom Jobs as well as seeded Jobs. • Just like seeded Flows, they have a start Process and an end Process. • Processes cannot be shared between Flows. However Jobs can be shared between Processes. • Can be deleted from the UI.

All modifications to CBEs are audited, and will show as a corresponding audit event on the Audit Events screen.

Today, all Custom Entities execute as standalone invoke-ables. They cannot be linked with or have a dependency on a Job in the Nightly Cycle. This capability will be added in a future release.

APIs

ReST APIs have been provided to programmatically add, modify or delete CBEs in POM. Please see Custom Batch Entities API in the Invoking the POM Service chapter for further details.

Lift-and-Shift

The Export / Import feature of POM can also perform the lift and shift of Custom Batch Entities in both the XLS and JSON formats.

The rules followed while importing are

1. Any CBE not present in the destination environment will be created.
2. Any CBE already present in the destination environment will not be merged or modified.

6
Custom Schedules

This chapter describes how a user sets up a custom batch schedule in POM.

> **Note:**
>
> Support and maintenance for customizations, as well as any batch failures of custom jobs, are the retailer's responsibility. In addition, retailers should carefully evaluate the impact of all customizations to existing schedules, given that these could extend the batch window.

Introduction

POM comes with a set of standard retail application batch schedules such as Merchandising and RI. Each schedule is defined in a spreadsheet with multiple tabs that detail the components of that schedule, such as the Process tab, Job tab, ProcessJobMapping tab, Dependency tab, and so on. For each retail application a customer has subscribed to, a number of steps are performed by either a system integrator or the customer themselves. These steps are to configure and load those schedules, then customize them.

POM also provides the ability for customers to set up their own custom schedule based on predefined retail application jobs and environment where these are run.

For a user to develop and deploy a custom schedule, the following activities must be completed:

- Set Up the Schedule Spreadsheet
- Configure the New Schedule
- Load the New Schedule
- Schedule the Batch Tasks
- Run Batch

The next sections detail these activities.

Set Up the Schedule Spreadsheet

The first step for creating a batch schedule in POM is to fill out the batch schedule spreadsheet using the template provided with the documentation set and specifications detailed in Batch Schedule Spreadsheet Template.

Configure the New Schedule

The second step for setting up a batch schedule in POM is to configure the schedule. Refer the “Configure New Schedule” subsection of the “System Configuration“ section of the *POM*

User Guide for details. When configuring the new schedule, you can configure it to use an existing agent including the common agent discussed in the Agent section of this guide.

Load the New Schedule

The third step for setting up a batch schedule is to load the schedule into POM. This is accomplished on the Schedule Maintenance screen by selecting this new schedule's tile at the top of the screen, then clicking the **Import Latest Schedule** button. Refer to the “Schedule Maintenance” section of the *POM User Guide* for details.

Schedule the Batch Tasks

The next step for setting up a batch schedule in POM is to schedule the execution of the different cycles. This is accomplished on the Scheduler Administration screen. Refer to the “Scheduler Administration” section of the *POM User Guide* for details.

Run Batch

The final step for setting up a batch schedule in POM is to actually run the batches. For that, open the Batch Monitoring screen, select the schedule tile at the top of the screen, chose the correct date right above the tile section and click the **Create Schedule** button right below the tile section to the right of the screen. Refer to the “Batch Monitoring” section of the *POM User Guide* for details.

7

Generic ReST Jobs

This chapter describes how a user can set up ReST Jobs that can be invoked and tracked by POM.

Introduction

As a Job Scheduler, POM provides the ability for applications to invoke and track different types of Jobs. The list of Job Types that are supported in POM by default are defined in the Batch Job section of the Batch Concepts chapter of this guide.

While this list of types meets the needs of Retail Applications that already have their Batch Schedules on POM, it is fixed and lacks extensibility.

To remedy this, POM provides generic interfaces to define and run ReST-based Jobs on external systems that meet certain requirements.

Features of Generic ReST Jobs

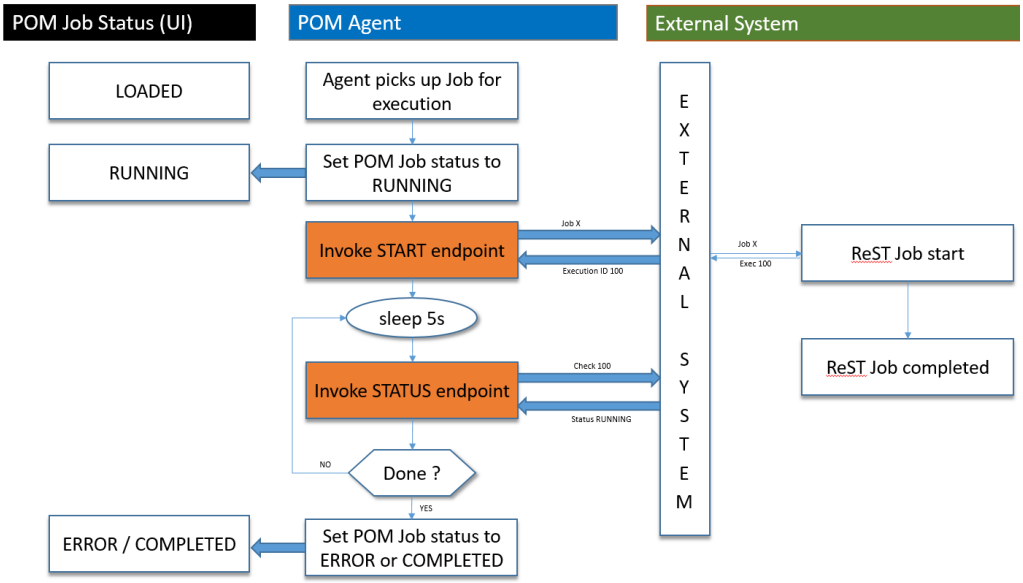
To have POM integrate with an external system and to invoke and track Jobs run on that external system, the external system must provide the following capabilities.

#	Ability / ReST endpoints	Description	Mandatory / Optional
1	Start requested Job	Ability to start a Job that was requested and return a unique ID to track its execution. If the Job can't be started or does not exist, throw an exception	Mandatory
2	Check status of Job	Ability to accept a unique execution ID and return a status. In case the execution ID is unknown, throw an exception	Mandatory
3	Restart Job	Ability to restart a Job, given an execution ID of a failed job. Once restarted, a unique execution ID to track the new execution must be returned.	Optional. If the Restart API is not defined, POM uses the Start API (#1)
4	Kill Job	Ability to stop/abort/kill a running Job. This is not meant to be a graceful stop. It should be a hard-stop and not just a request to the Job to stop when possible.	Optional. If a KILL is requested when it's not defined, then POM shows an exception popup.
5	Job Log	Ability to provide runtime logs for the given execution ID.	Optional.

#	Ability / ReST endpoints	Description	Mandatory / Optional
6	Check system status	Ability for POM to check whether the system is up and available. This will show up on the POM Health-check screen.	Mandatory. If this is not defined, this will show up in the Health check popup with an error and affects the automated patching process.

Execution Sequence

At runtime, POM invokes generic ReST jobs as depicted below.



1. Initially the status of the Job in POM is `LOADED`.
2. The Execution Engine sends the request to the Job Agent to run the Job. The Agent Poller picks the Job for execution.
3. The Agent sets the status of the Job to `RUNNING`.
4. The Start endpoint is invoked on the External System, with the details of the Job to run. The endpoint starts the Job and returns a unique ID to track the execution (`100` returned, as shown above).
5. The Agent then goes to sleep for 5 seconds (configurable interval).
6. It then invokes the Status endpoint on the External System with the execution ID returned previously (in this case `100`)
7. If a status of `ERROR` or `COMPLETED` is returned, then the Job is assumed to have finished and the status is set on the POM job accordingly.

8. In case the status is still `RUNNING`, the Agent then goes back to sleep for 5 seconds and thereafter polls the status again. It does so, until a terminal status of `ERROR` or `COMPLETED` is received.

Handling Restarts

A restart is handled in almost the same sequence as shown above. Except, instead of invoking the Start endpoint, the Restart endpoint will be used with the previously failed execution ID.

In case the Restart endpoint has not been defined, then the Start endpoint will be used.

Handling Kills

In case of killing/aborting a `RUNNING` Job, the Kill endpoint is invoked. A successful response from this endpoint will let POM perform its cleanup and also mark the Job with a status of `ERROR`.

In case the Kill endpoint has not been defined for a Custom Job Type, then invoking the Kill from the POM UI for a running Job of that type will simply show a popup stating the Kill is not defined for the selected Job type.

Applications Notifications

Applications that run their batches through POM are capable of having POM generate a contextual notification upon the completion of a job. The completion can be a success or failure. This is accomplished by the application optionally specifying notification details in the return payload of the Job Status endpoint. For example, an application can upon processing over a million records have the job return a notification object containing the message: 'Job abcd processed over a million records today'.

See the Job Status API specification in the Generic ReST Jobs section later in this chapter for details.

Defining Custom Job Types

Generic ReST Jobs are defined in POM as Custom Job Types. A class of Generic ReST Jobs is represented in POM as a single Custom Job Type.

All Jobs in POM today have a Job Type attribute. For Generic ReST Jobs, this Job Type will be a Custom Job Type that will encapsulate all the necessary information needed about the destination system.

Batch Schedule Spreadsheet

Custom Job Types can be defined through the **Job Types** tab of the Batch Schedule spreadsheet. Custom Job Types that are specified on the spreadsheet get seeded into POM when the Batch Schedule is loaded. See Batch Schedule Spreadsheet Template for more information.

POM UI

The Administrator can also maintain Custom Job Types in the **Job Types** section of the **System Configuration** screen. Refer to the “Edit Job Type” subsection of the “System Configuration” section of the *POM User Guide*.

New Custom Job Types can be added there as well.

Environment-specific Information

The only aspects of a Custom Job Type that will change based on its deployment are its URL and its OAuth Scope.

Today teams use an endpoint to configure a new Schedule in POM. This endpoint has been enhanced to accept the URL and the OAuth scope needed for a Custom Job Type. Refer to the “Configure New Schedule” subsection of the “System Configuration” section in the *POM User Guide* for further information.

Internal Representation

A JSON representation of a Custom Job Type stored within POM looks like the following:

```
{
    "scopes": "rgbu:pom:agent-administrator-DEV231",
    "useCustomCred": true,
    "actions": [
        {
            "action": "JOB_START",
            "resourcePath": "/start"
        },
        {
            "action": "JOB_RESTART",
            "resourcePath": "/restart"
        },
        {
            "action": "JOB_STATUS",
            "resourcePath": "/status"
        },
        {
            "action": "JOB_KILL",
            "resourcePath": "/kill"
        },
        {
            "action": "JOB_LOG",
            "resourcePath": "/log"
        },
        {
            "action": "VALIDATION",
            "resourcePath": null
        }
    ]
}
```

Job Parameter Restrictions

The definition of parameters to Generic ReST Jobs in POM is restricted to a double-pipe-delimited string of key value pairs.

For example: `key=value||key1=value1||key2=value2`

Custom Job Type Endpoint Specifications

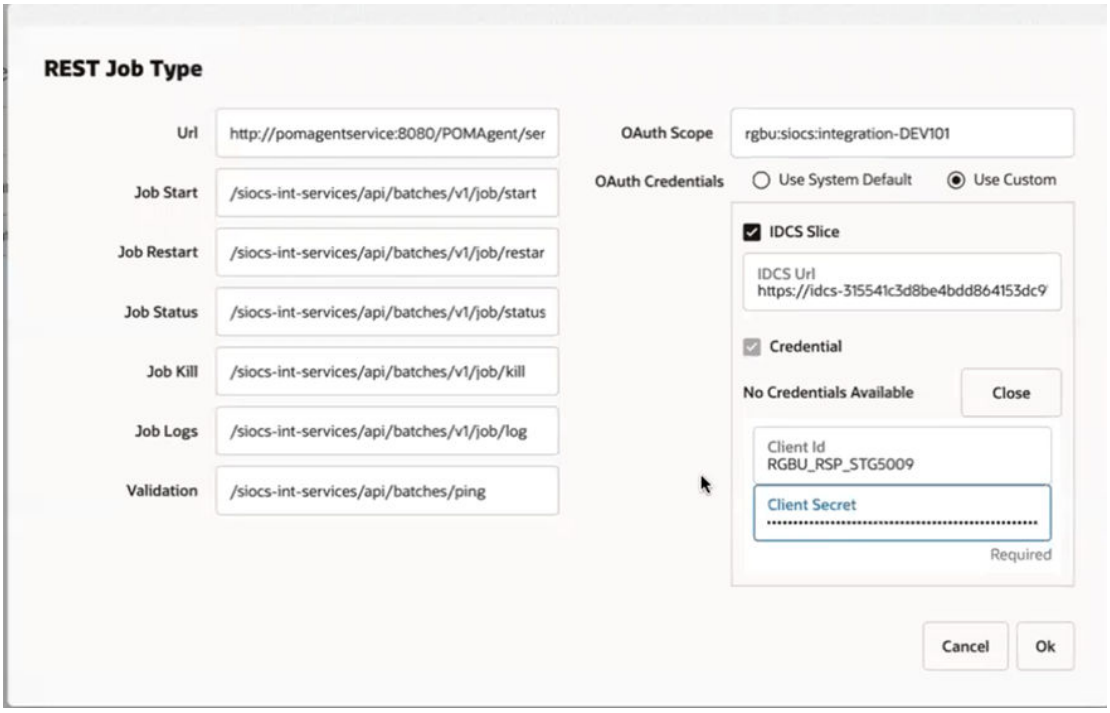
Credentials & Scopes

POM only supports the OAuth System Credential Grant mode of authentication. No other types of authentication are supported.

The OAuth scope stored for each Custom Job Type informs POM about which scopes are to be used when invoking the Custom Job Type APIs.

POM makes calls to these Job Type endpoints by using this scope and the default IDCS credentials it is configured with. However, these endpoints are generally hosted on an application that has its own IDCS application. In such cases, POM cannot invoke those endpoints using its default IDCS credentials. Therefore, POM provides an option to configure a custom IDCS slice and custom credentials to be used for this type of authentication. These credentials are the Client Id & Secret of the respective application/ OAuth client, as shown below. POM calls the custom job type endpoints that can be authenticated using an IDCS instance different that POM's. Custom credentials are also published to agents during new scheduler day creation.

It is expected that the IDCS Client App that these custom credentials point to will contain the necessary scopes to call these endpoints.



Job Start API

HTTP Method

POST

Request Payload (JSON)

The attributes of the JSON payload are as follows :

Attribute	Description
cycleName	POM Batch Cycle
flowName	POM Batch Flow.
processName	POM Batch process name
jobName	POM Batch job name
parameters	Job Parameters (Double-pipe delimited key-value pairs) as defined in POM
agentExecutionId	A unique ID assigned by POM for every job run

Authentication

OAuth System Credentials Grant

Response Payload (JSON)

In case of a successful invocation, a success HTTP code (`2xx`) is returned with a payload like the following:

```
{
   "executionId": "112",
   "executionInfo": Job started,
   "status": "STARTED"
}
```

Attribute	Description
executionId	Unique Id returned by the target app to POM for status tracking
status	Any status
executionInfo (optional)	Any additional info the target app would like to share with POM.

In case of an issue, an HTTP code other than the success code (that is, not `2xx`) is sent along with an error payload of any format which will be captured as part of the Job failure logs.

Job Status API

HTTP Method

GET

Query Parameters (JSON)

Attribute	Description
processName	POM Batch process name
jobName	POM Batch job name
executionId	Unique Id returned by the target app to POM for status tracking

Authentication

OAuth System Credentials Grant

Response Payload (JSON)

In case of a successful invocation, a success HTTP code (`2xx`) is returned with a payload like the following:

```
{
   "status": "COMPLETED",
   "executionInfo": null,
   "notification": {
      "info": "Number of stock counts: 231",
      "type": "StockCountErrorNotification",
      "severity": 1
   }
}
```

Attribute	Description
executionId	Unique Id returned by the target app to POM for status tracking
status	RUNNING / ERROR / COMPLETED.
executionInfo (optional)	Any additional info the target app would like to share with POM.
notification	An optional Notification Object containing details regarding an Application's Notification that needs to be generated through POM.
notification.info	Message to be used for the App Notification
notification.type	Notification Type to be used for the App Notification. This must be an existing Notification Type on the Consuming Application.
notification.severity	Severity of the App Notification generated. Valid values are 1-Critical, 2-Important, 3-Normal

In case of an issue, an HTTP code other than the success code (that is, not `2xx`) is sent along with an error payload of any format which will be captured as part of the Job failure logs.

Job Restart API

HTTP Method	POST

Request Payload (JSON)

The attributes of the request payload are as follows:

Attribute	Description
cycleName	POM Batch Cycle
flowName	POM Batch Flow.
processName	POM Batch process name
jobName	POM Batch job name
parameters	Job Parameters (Double-pipe delimited key-value pairs) as defined in POM
agentExecutionId	A unique ID assigned by POM for every job run
executionId	Unique Id of the previously failed execution returned by the target app to POM for status tracking

Authentication

OAuth System Credentials Grant

Response Payload (JSON)

In case of a successful invocation, a success HTTP code (`2xx`) is returned with a payload like the following:

```
{
   "executionId": "112",
   "executionInfo": Job re-started,
   "status": "STARTED"
}
```

Attribute	Description
executionId	Unique Id returned by the target app to POM for status tracking
status	Any status
executionInfo (optional)	Any additional info the target app would like to share with POM.

In case of an issue, an HTTP code other than the success code (that is, not `2xx`) is sent along with an error payload of any format which will be captured as part of the Job failure logs.

Job Kill API

HTTP Method

POST

Request Payload (JSON)

The attributes of the request payload are as follows:

Attribute	Description
processName	POM Batch process name
jobName	POM Batch job name
executionId	Unique Id of the previously failed execution returned by the target app to POM for status tracking

Authentication	OAuth System Credentials Grant

Response Payload (JSON)	In case of a successful invocation, a success HTTP code (`2xx`) is returned with a payload like the following:
	
	`{`
	`"executionId": "112",`
	`"executionInfo": Job Killed,`
	`"status": "CANCELLED"`
	`}`
	
	Attribute / **Description**
	executionId (optional) — Unique Id returned by the target app to POM for status tracking
	status (optional) — Any status
	executionInfo (optional) — Any additional info the target app would like to share with POM.
	
	In case of an issue, an HTTP code other than the success code (that is, not `2xx`) is sent along with an error payload of any format which will be captured as part of the Job failure logs.

Job Log API

HTTP Method	GET
Query Parameters (JSON)	**Attribute** / **Description** processName — POM Batch process name jobName — POM Batch job name executionId — Unique Id returned by the target app to POM for status tracking
Authentication	OAuth System Credentials Grant
Response Payload (JSON)	In case of a successful invocation, the response will contain the log data for the execution provided. This log will be viewable in the log downloaded from the POM UI. The logs from the response are also sent in the job log. Any other response while calling this endpoint would be captured as an error while downloading logs.

Job System Check API

HTTP Method	GET
Authentication	OAuth System Credentials Grant
Response Payload (JSON)	A success response of 200 or 204 indicates that the system is up and running. Any other response while calling this endpoint, should be interpreted as the system being down.

8
Export/Import Configuration

This chapter explains the export/import of schedule configuration and of custom entities configuration in POM.

Schedule Configuration

Overview

POM provides a feature, in the user interface, to export and import schedule configuration data such as data retention limits, throttle limits, enable/disable flags, schedule times, notifications configuration, and so on.

This feature can be used in two ways:

1. Data exported on a specific environment can be imported back on a different environment without any modifications. This is a typical use case where the batch schedule on a certain environment, such as Stage, is set up and fine-tuned according to a customer's requirements. This includes specifying which jobs should or shouldn't run. It includes ad hoc flows configuration. It includes specifying the times certain cycles or flows will start. It also includes specifying the e-mail addresses that will receive certain notifications and how long to keep those notification in the system before purging them.

   Once the schedule is configured and fine-tuned (on Stage, for example) the export/import feature can be used to export the configurations from Stage and import them back into production.

   > **Note:**
   >
   > The user has the option to export to a spreadsheet or a JSON file. In this first use case, it is recommended the user exports to a JSON file then imports it back into the other environment.

2. Data can be altered in the exported spreadsheet then imported back into the same environment. This is useful at provisioning time where an environment is first set up with the default batch schedule configuration. A user would then export the default configuration, then modify that configuration on the spreadsheet to conform to the customer's needs. The spreadsheet is then imported back into the same environment, applying the desired configurations.

> **Note:**
>
> When importing the spreadsheet on the Batch Administration screen, the user will have the option of importing job-related configuration or scheduler-related configuration (run times) or both. Refer to the "Batch Administration" section in the *POM User Guide* for more details.

This chapter describes each tab in the exported spreadsheet, along with the data on each sheet. It also indicates which fields can be modified.

Schedule Info Tab

This is an informational only tab. No fields are modifiable on it.

Table 8-1 Schedule Info

Field	Description	Modifiable?
Name	Schedule or application name.	No
Display Schedule Name	Display schedule name used on POM UI, reports and mail notifications. **Note:** This field is not modifiable using the export / import feature. To modify it, refer to the Schedule Tab section of the Batch Schedule Spreadsheet Template appendix of this guide.	No
Version	Schedule version.	No
Installation Date	Date the schedule was uploaded into POM.	No
Customer Name	This is used on notifications to identify which customer a notification is for.	No
Environment Name	This is used on notifications to identify which customer's environment a notification is for.	No

Schedule Configuration Tab

This tab contains schedule level settings.

Table 8-2 Schedule Configuration

Field	Description	Modifiable?
First Run Business Date	Date when batch was run for the first time. See subsection "Business Date Explained" in the "System Configuration" section of the *POM User Guide*.	No
Data Retention Days	Number of days historic data is to be retained in POM.	Yes - Recommended value is 30 days.
Long Run Average Multiplier	Number which is multiplied by a job's average run time to determine the threshold which, when exceeded, the job is deemed long running.	Yes - Usually a number between one and three.
External Dependencies	Indicates whether POM will respect external dependencies or not. They are usually dependencies on customers' internal processes.	Yes - Valid values are Y or N.
Inter-schedule Dependencies	Indicates whether POM will respect external dependencies or not. Those are dependencies on other schedules.	Yes - Valid values are Y or N.
Callback mode	This is also known as External Status Update Mode. Depending on the value, the customer's system is notified of success or failure of any job in the entire schedule. The value set here is overridden by this same setting for any individual job defined on the other job-specific tabs.	Yes - Valid values are ALL, FAILED or NONE. When value is NONE, no callback is made. When value is ALL, callback is made on either success or failure. When value is FAILED, callback is made only on failure

Throttling Tab

At this time, this tab should only be used to set throttle limits at the application level. Throttling is a technique used to limit the number of jobs that can run concurrently for a specific application. Throttling limits are set so a server's resources are not overwhelmed by too many concurrently running jobs.

> **Note:**
>
> Previously, throttle limits could be set at the application level and/or at the module level. Now they can only be set at the application level.

Also note that this tab was intended to provide the capability to enable/disable whole applications or modules. This is not functioning at this time. In order to achieve this objective, filter on application on the Nightly Jobs Configuration, Recurring Jobs Configuration or Adhoc Jobs Configuration, then change all enabled flags to 'Y' or 'N' as desired. For enabling/disabling whole modules, this has to be done on the Batch Administration screen at this time.

(Continued)

Table 8-3 Throttling

Field	Description	Modifiable? –Possible values
Application	Application code.	No
Module	Module name.	No
Agent	Agent is the lightweight component usually running on the consuming application's container. It executes that application's batch jobs based on requests from POM.	No
Throttle Limit	The max number of jobs that can run concurrently for the specified application.	Yes – A number that is greater than zero.
Enabled	This flag currently does not enable or disable jobs belonging to an application/module. See the second note at the top of this section for more information.	Yes – Valid values are Y or N.

Nightly Jobs Configuration Tab

This tab contains all Nightly cycle's jobs along with their configuration.

Table 8-4 Nightly Jobs Configuration

Field	Description	Modifiable? – Possible values
Job	Job name.	No
Process	Process name to which the job belongs; a process can contain multiple jobs.	No
Cycle	This is a fixed value for this tab of 'Nightly'.	No
Flow	This is a fixed value of "Nightly" for this tab.	No
Application	Application name to which this process-job belongs.	No
Parameter Change	Flag indicating whether the parameter can be changed after the initial load. If it can, then the new value must be loaded in the next field: Active Parameter.	Yes – Valid values are Y or N.
Job Type	Indicates name of the job type such as `EXEC`, `RI`, `RASE`, `BDI`, `RPAS`, `RDS`, `OMS`, `OB`.	No
Active Parameter	Parameter which overrides the Base Parameter when Parameter Change is 'Y'.	Yes – Can contain blank or any space-separated list of parameters.

Table 8-4 (Cont.) Nightly Jobs Configuration

Field	Description	Modifiable? – Possible values
Priority	An optional number between 1 and 10 (10 being highest priority) assigned to a process-job combination. This number determines the execution priority for the job amongst concurrent job run requests in a limited throttled setting.	Yes – Valid values are 1-10
Active Phase	An optional alphanumeric phase assigned to a process-job for grouping of jobs in the Nightly cycle. This is used for reporting purposes only. The Nightly Summary Report will then report total run time and elapsed time by Phase.	Yes – Valid values – alphanumeric text like `PHASE_1`
Kill Cleanup Script	Absolute path of the clean-up script to be run after killing a job from POM. Can include arguments as well along with the script.	Yes, Valid values – /u01/scripts/killCleanUp.sh “123” “345”
Skip on error	Flag indicating whether this job should automatically be skipped when it fails so batch can resume	Yes – Valid values are Y or N.
Callback mode	This is also known as External Status Update Mode. Depending on the value, the customer's system is notified of success or failure of this specific job by calling a predefined customer endpoint. The value set here overrides that of this same setting on the Schedule Configuration tab.	Yes – Valid values are ALL, FAILED or NONE. When value is NONE, no callback is made. When value is ALL, callback is made on either success or failure. When value is FAILED, callback is made only on failure
Day of week	Contains the day(s) of the week on which this specific Job will run. POM automatically skips these Jobs on remaining days. If this field is left blank, the job will run on every day of the week.	Yes – Valid values are blank or any number of comma separated days of the week (for example: SUNDAY,MONDAY,TUESDAY, WEDNESDAY,THURSDAY, FRIDAY,SATURDAY)

Table 8-4 (Cont.) Nightly Jobs Configuration

Field	Description	Modifiable? – Possible values
Threshold Runtime	Estimated runtime in seconds for this specific job. This is an optional field which, when entered, will be used as a threshold. When this threshold is exceeded, the job is deemed long running. If this field is blank, then the job's average run time is used instead to multiply by the Long Run Average Multiplier for determining the threshold.	Yes – Valid values are blank or an integer number of seconds.
Enabled	Flag indicating if this job is enabled or disabled	Yes – Valid values are Y or N.
Notify Job Start	Flag indicating if a notification is to be sent at start of this job.	Yes – Valid values are Y or N.
Notify Job Completion	Flag indicating if a notification is to be sent at successful completion of this job.	Yes – Valid values are Y or N.

Recurring Jobs Configuration Tab

This tab contains the list of Recurring cycle/Process/Job along with their configuration.

Table 8-5 Recurring Jobs Configuration

Field	Description	Modifiable? –Possible values
Job	Job Name	No
Process	Process name to which the job belongs; a process can contain multiple jobs.	No
Cycle	Name of the hourly/recurring cycle this job belongs to.	No
Flow	Name of the recurring flow	No
Application	Application name to which this process-job belongs.	No
Parameter Change	Flag indicating whether the parameter can be changed after the initial load. If it can, then the new value must be loaded in the next field: Active Parameter.	Yes – Valid values are Y or N.
Job Type	Name of the job type such as `EXEC`, `RI`, `RASE`, `BDI`, `RPAS`, `RDS`, `OMS`, `OB`	No

Table 8-5 (Cont.) Recurring Jobs Configuration

Field	Description	Modifiable? –Possible values
Active Parameter	Parameter which overrides the Base Parameter when Parameter Change is 'Y'.	Yes – Can contain blank or any space separated list of parameters.
Active Priority	An optional number between 1 and 10 (10 being highest priority) assigned to a process-job combination. This number determines the execution priority for the job amongst concurrent job run requests in a limited throttled setting.	Yes – Valid values are 1-10
Active Phase	An optional alphanumeric phase assigned to a process-job for grouping of jobs in the Nightly cycle. This is used for reporting purposes only. The Nightly Summary Report will then report total run time and elapsed time by Phase.	Yes – Valid values – alphanumeric text like PHASE_1
Kill Cleanup Script	Absolute path of the clean-up script to be run after killing a job from POM. Can include arguments as well along with the script.	Yes, Valid values – /u01/scripts/killCleanUp.sh “123” “345”
Skip on error	Flag indicating whether this job should automatically be skipped when it fails so batch can resume.	Yes – Valid values are Y or N.
Callback mode	This is also known as External Status Update Mode. Depending on the value, customer's system is notified of success or failure of this specific job by calling a predefined customer endpoint. The value set here overrides that of this same setting on the Schedule Configuration tab.	Yes – Valid values are ALL, FAILED or NONE. When value is NONE, no callback is made. When value is ALL, callback is made on either success or failure. When value is FAILED, callback is made only on failure
Day of week	Contains the day(s) of the week on which this specific Job will run. POM automatically skips these Jobs on the remaining days. If this field is left blank, the job will run on every day of the week	Yes – Valid values are blank or any number of comma-separated days of the week (for example: SUNDAY,MONDAY,TUESDAY, WEDNESDAY,THURSDAY, FRIDAY,SATURDAY)

Table 8-5 (Cont.) Recurring Jobs Configuration

Field	Description	Modifiable? –Possible values
Threshold Runtime	Estimated runtime in seconds for this specific job. This is an optional field which, when entered, will be used as a threshold. When this threshold is exceeded, the job is deemed long running. If this field is blank, then the job's calculated average run time is used instead to multiply by the Long Run Average Multiplier for determining the threshold.	Yes – Valid values are blank or an integer number of seconds.
Enabled	Flag indicating whether this job is enabled or disabled.	Yes – Valid values are Y or N.
Notify Job Start	Flag indicating if a notification is to be sent at start of this job.	Yes – Valid values are Y or N.
Notify Job Completion	Flag indicating if a notification is to be sent at successful completion of this job.	Yes – Valid values are Y or N.

Recurring Flows Configuration Tab

This tab contains a flow/cycle matrix. It contains a row for each flow and all recurring cycles as columns. An intersection of 'Y' means the given flow is defined to run as part of the given recurring cycle. It is recommended that all recurring cycles are enabled but not necessarily scheduled for running. See the Scheduling Flows Tab for further clarification.

Table 8-6 Recurring Flows Configuration

Field	Description	Modifiable?
Flow	Flow name.	No
Recurring cycle 1	A 'Y' or 'N' value indicating whether the given flow is run as part of recurring cycle 1. A value of N does not disable jobs belonging to the flow/cycle intersection. Those jobs need to be manually disabled on the Batch Administration screen.	Yes – Valid values are Y or N.
Recurring cycle 2	A 'Y' or 'N' value indicating whether the given flow is run as part of recurring cycle 2. A value of N does not disable jobs belonging to the flow/cycle intersection. Those jobs need to be manually disabled on the Batch Administration screen.	Yes – Valid values are Y or N.

Table 8-6 (Cont.) Recurring Flows Configuration

Field	Description	Modifiable?
Recurring cycle n	A 'Y' or 'N' value indicating whether the given flow is run as part of recurring cycle n. A value of N does not disable jobs belonging to the flow/cycle intersection. Those jobs need to be manually disabled on the Batch Administration screen.	Yes – Valid values are Y or N.

Adhoc Flows Configuration Tab

This tab allows for enabling/disabling of ad hoc flows.

Table 8-7 Adhoc Flows Configuration

Field	Description	Modifiable?
Name	Ad hoc Flow name.	No
Enabled	Flag indicating whether this ad hoc flow is enabled or disabled.	Yes – Valid values are Y or N.
Custom	Flag indicating whether this ad hoc flow is custom or not.	No

Adhoc Processes Configuration Tab

This tab allows for enabling/disabling ad hoc processes.

Table 8-8 Adhoc Processes Configuration

Field	Description	Modifiable?
Name	Ad hoc Process name.	No
Enabled	Flag indicating whether this ad hoc process is enabled or disabled.	Yes – Valid values are Y or N.
Custom	Flag indicating whether this ad hoc process is custom or not.	No

Adhoc Jobs Configuration Tab

This tab contains the list of ad hoc jobs (also known as standalone) along with their configuration.

Table 8-9 Adhoc Jobs Configuration

Field	Description	Modifiable? – Possible values
Job	Job name.	No

Table 8-9 (Cont.) Adhoc Jobs Configuration

Field	Description	Modifiable? – Possible values
Process	Process name to which the job belongs; a process can contain multiple jobs.	No
Cycle	This is a fixed value for this tab of 'Adhoc'.	No
Flow	This is a fixed value of “Adhoc” for this tab.	No
Application	Application name to which this process-job belongs.	No
Parameter Change	Flag indicating whether the parameter can be changed after the initial load. If it can, then the new value must be loaded in the next field: Active Parameter.	Yes – Valid values are Y or N.
Job Type	Name of the job type such as `EXEC`, `RI`, `RASE`, `BDI`, `RPAS`, `RDS`, `OMS`, `OB`	No
Active Parameter	Parameter which overrides the Base Parameter when Parameter Change is 'Y'.	Yes – Can contain blank or any space separated list of parameters.
Active Priority	An optional number between 1 and 10 (10 being highest priority) assigned to a process-job combination. This number determines the execution priority for the job amongst concurrent job run requests in a limited throttled setting.	Yes – Valid values are 1-10
Active Phase	An optional alphanumeric phase assigned to a process-job for grouping of jobs in the Nightly cycle. This is used for reporting purposes only. The Nightly Summary Report will then report total run time and elapsed time by Phase.	Yes – Valid values – alphanumeric text like PHASE_1
Kill Cleanup Script	Absolute path of the clean-up script to be run after killing a job from POM. Can include arguments as well along with the script.	Yes, Valid values – /u01/scripts/killCleanUp.sh “123” “345”
Skip on error	Flag indicating whether this job should automatically be skipped when it fails so batch can resume.	Yes – Valid values are Y or N.

Table 8-9 (Cont.) Adhoc Jobs Configuration

Field	Description	Modifiable? – Possible values
Callback mode	This is also known as External Status Update Mode. Depending on the value, the customer's system is notified of the success or failure of this specific job by calling a predefined customer endpoint. The value set here overrides that of this same setting on the Schedule Configuration tab.	Yes – Valid values are ALL, FAILED or NONE. When value is NONE, no callback is made. When value is ALL, callback is made on either success or failure. When value is FAILED, callback is made only on failure
Day of week	Contains the day(s) of the week on which this specific Job will run. POM automatically skips these Jobs on remaining days. If this field is left blank, the job will run on every day of the week	Yes – Valid values are blank or any number of comma separated days of the week (for example: SUNDAY,MONDAY,TUESDAY, WEDNESDAY,THURSDAY, FRIDAY,SATURDAY)
Threshold Runtime	Estimated runtime in seconds for this specific job. This is an optional field which, when entered, will be used as a threshold. When this threshold is exceeded, the job is deemed long running. If this field is blank, then the job's calculated average run time is used instead to multiply by the Long Run Average Multiplier for determining the threshold.	Yes – Valid values are blank or an integer number of seconds.
Enabled	Flag indicating whether this job is enabled or disabled.	Yes – Valid values are Y or N.
Notify Job Start	Flag indicating if a notification is to be sent at start of this job.	Yes – Valid values are Y or N.
Notify Job Completion	Flag indicating if a notification is to be sent at successful completion of this job.	Yes – Valid values are Y or N.
Custom	Flag indicating whether an ad hoc job is custom or not	No

Job Dependencies Tab

This tab contains the definitions of all intra-schedule dependencies (or job dependencies within the same schedule) for all cycles (Nightly, Recurring and Adhoc). External, inter-schedule dependencies and execution links are defined on the Job External Associations tab.

Only the Enabled flag can be changed on this tab. Caution should be exercised when disabling a dependency, as in certain situations this can cause a job to run before data is processed by the predecessor job. This can therefore cause data corruption.

Table 8-10 Job Dependencies Configuration

Field	Description	Modifiable? –Possible values
Job	Job name.	No
Process	Process name to which the job belongs.	No
Cycle	Nightly, Adhoc or specific recurring/hourly cycle.	No
Predecessor Process	Process name to which the predecessor job belongs.	No
Predecessor Job	Job which must complete before the job defined on the current row can run.	No
Enabled	Flag indicating whether this dependency is enabled or disabled.	Yes – Valid values are Y or N.

Job External Associations Tab

This tab contains the definitions of External dependencies, inter-schedule dependencies and execution links, so all dependencies other than the intra-schedule dependencies described in the previous section.

> **Note:**
>
> New external associations can be added in this tab.

External dependencies are usually those defined for customer processes. These are associated with a POM endpoint that the customer calls to satisfy the dependency.

Inter-schedule dependencies are those associated with another schedule. For instance, a Retail Insight job can be made to wait for a Merchandising job to complete.

An Execution link is a special dependency that sets up an application's schedule to be invoked based on a completion of a job in another application’s schedule. For instance, the Retail Insight schedule can be set up to be invoked when job A completes in the Merchandising schedule.

Table 8-11 Job External Associations

Field	Description	Modifiable? – Possible values
Job	Job name.	Yes – Needs to be a valid job name already defined on the nightly schedule.
Process	Process name to which the job belongs.	Yes – Needs to be a valid process name already defined in the nightly schedule.
Cycle	Cycle name.	Yes – This can only be Nightly.

Table 8-11 (Cont.) Job External Associations

Field	Description	Modifiable? – Possible values
Association Type	Type of dependency.	Yes – Valid values are Internal for Inter-Schedule, External and EXEC_LINK
External Schedule	Name of the schedule containing the inter-schedule dependency or execution link.	Yes – This must be an existing valid schedule defined in the same instance of POM. This is only required for inter-schedule dependencies and execution links.
External Job	Name of external job.	Yes – In the case of an external dependency, this is the name that will be included in the payload of an endpoint called by the external system, such as the customer's. In the case of inter-schedule dependencies and execution links, this is a valid job name defined in the dependent schedule.
External Process	Process name to which the external job belongs.	Yes – This is required for inter-schedule dependencies and execution links. It's the process name associated with the external job.
Enabled	Flag indicating whether this dependency is enabled or disabled.	Yes – Valid values are Y or N.

Scheduling Flows Tab

This tab and the next (Scheduling Adhoc Tab) contain multiple configurations for scheduling flows or processes to run at specified times.

This Scheduling Flows tab contains the definitions of scheduled run times for the Nightly and recurring flows.

It simply contains one row for the Nightly flow stating the time when the Nightly processes will start every day. In case of weekly configuration, nightly flows can contain multiple rows stating the different times for each day of the week to start the task.

The rest of the tab contains rows for each recurring flow, with columns for each of the hourly cycles making up the flow. For each cell at the intersection of Flow and hourly cycle, the time is entered for when the cycle is to start running. At present, there is a maximum of 24 hourly cycles that can be defined which, when spaced equally, would run one hour apart. Times are optional, so a blank cell indicates that the cycle will not be scheduled to run.

> **Note:**
>
> Although there is no validation that cross-references the presence of a time in a cell with the enabling of the cycle on the Flows Configuration tab, ideally these would match up. This means that, if you enter a time for running an hourly cycle, the same intersection on the Flows Configuration tab should be 'Y'. However, it is conceivable to enable the cycle on the Flows Configuration tab but not schedule it, leaving the option open to schedule it as needed. In fact, this is the recommended practice: Enable all hourly cycles and only schedule a few as needed, but have the option to run more to catch up when necessary.

The timezone region ID entered in cell B is used for all times entered for all subsequent cycles on a given row. It is possible to specify a different timezone for select time cells (for example, 5:00 America/Chicago).

> **Note:**
>
> If the timezone region ID is null in the database, UTC is exported as the default into cell B of this tab.
>
> Frequency entered in the Frequency cell denotes different task frequencies like Daily and Weekly.
>
> Day of week entered in the corresponding cell denotes the day on which the task is to be executed. It is left blank when the frequency is DAILY.

Table 8-12 Scheduling Flows

Field	Description	Modifiable? – Possible values
Flow	Flow name.	No
Timezone Region ID	Timezone corresponding to the time entered in the subsequent cell(s). Timezone region ID is required, rather than a UTC offset (such as "UTC-06:00"). This is to accommodate Daylight Saving Time.	Yes – Valid timezone region IDs such as US/Eastern can be found as TZ database name at https://en.wikipedia.org/wiki/List_of_tz_database_time_zones
Frequency	Task frequency	Yes – Valid values for this tab: DAILY, WEEKLY
Day of week	Day of the week for the task to run	Yes – Valid values: MONDAY or TUESDAY or WEDNESDAY or THURSDAY or FRIDAY or SATURDAY or SUNDAY. Should be left blank for the DAILY frequency
Nightly	Only fill this cell if Flow is Nightly. Enter the time for starting the Nightly cycle.	Yes – Enter a valid military time format such as 6:00 (6am) or 22:00 (10pm).

Table 8-12 (Cont.) Scheduling Flows

Field	Description	Modifiable? – Possible values
Recurring cycle 1	Enter a time in this cell to schedule recurring cycle 1 to run at that time. Leave blank to forego scheduling recurring cycle 1. It is possible to not schedule the cycle by default but elect to run it manually if needed.	Yes – Enter a valid military time format such as 6:00 (6am) or 22:00 (10pm). Conflict with the Nightly batch flow should be avoided as POM will skip running a recurring cycle if its start time is elapsed while Nightly is running.
Recurring cycle 2	Enter a time in this cell to schedule recurring cycle 2 for running at that time. Leave blank to forego scheduling recurring cycle 2. It is possible to not schedule the cycle by default but elect to run it manually if needed.	Yes – Enter a valid military time format such as 6:00 (6am) or 22:00 (10pm). Conflict with the Nightly batch flow should be avoided as POM will skip running a recurring cycle if its start time is elapsed while Nightly is running.
Recurring cycle n	Enter a time in this cell to schedule recurring cycle 1n for running at that time. Leave blank to forego scheduling recurring cycle n. It is possible to not schedule the cycle by default but elect to run it manually if needed.	Yes – Enter a valid military time format such as 6:00 (6am) or 22:00 (10pm). Conflict with the Nightly batch flow should be avoided as POM will skip running a recurring cycle if its start time is elapsed while Nightly is running.

Scheduling Adhoc Tab

This tab contains the definitions of scheduled run times for the Adhoc processes.

Table 8-13 Scheduling Adhoc

Field	Description	Modifiable? – Possible values
Name	Adhoc process/Adhoc flow name.	No
Type	Invokable type	Yes – Valid values: Flow or Process
Description	Description of reason for running this Adhoc flow/process at the specified time.	Yes – Optionally describe the purpose for scheduling the Adhoc flow/process at the specified time
Frequency	Frequency at which this Adhoc process/flow is to be run.	Yes – Valid values are DAILY, ONCE, WEEKLY, MONTHLY, MONTHLY_START, MONTHLY_END, MONTHLY_BY_WEEKDAY
Nth Day of Week in a Month	Nth day of week in a month on which task has to be run	Yes – Valid values are FIRST, SECOND, THIRD, FOURTH, LAST.
Day of week	Day of the week on which the task has to be run	Yes – Valid values: MONDAY or TUESDAY or WEDNESDAY or THURSDAY or FRIDAY or SATURDAY or SUNDAY.

Table 8-13 (Cont.) Scheduling Adhoc

Field	Description	Modifiable? – Possible values
Day of the Month	Day of the Month on which the task has to be run	Yes – Valid values: [1-31]
Start Date	Start Date of the task, only for frequency set to ONCE	Yes – Valid value: valid date in the format yyyy-MM-dd. For example: 2023-01-05.
Recurrence	Indicates if an activated task is meant to run just once or recurs multiple times.	Yes – Valid values: SINGLE/ MULTIPLE.
Interval	Indicates an interval number of minutes the at which the task should run again.	Yes – Valid values: Any positive integer number of minutes starting from 1. For example: 5, meaning this Adhoc will run every 5 minutes.
Limit Occurrences	Relevant only when frequency of EVERY:x is used. If a Limit Occurrences value is entered, the process will run a maximum of times equal to the specified limit.	Yes – A positive integer. For example: Frequency of DAILY with a Limit Occurrences of 4 and Interval 2 means the process/flow will run every two minutes a maximum of four times
Prevent start during nightly	A flag which, when set to Y, indicates that this process is not to be started when the Nightly cycle is running. If the Scheduler tries to start a run while Nightly is running, the execution's status is set to Error with an Info message of "Nightly started running so can't run the process."	Yes – Valid values are Y or N.
Schedule Time	Entering a time in this field causes the process to be scheduled at that time in case of a DAILY frequency or to start the first run at that time in case of an EVERY:x frequency. Leaving this field blank causes the process to run immediately or the first run to start immediately when the scheduler day starts. If a specific time is entered in this field but the scheduler day starts after that time, the process will be scheduled for the next day at that time.	Yes – Enter a valid military time format such as 6:00 (6am) or 22:00 (10pm).
Enabled	Flag indicating if this process is to be scheduled.	Yes – Valid values are Y or N.
Timezone Region ID	Timezone corresponding to the time entered in the subsequent cell(s). Timezone region ID is required rather than a UTC offset such as UTC-06:00. This is to accommodate Daylight Saving Time.	Yes – Valid timezone region IDs such as US/Eastern can be found as TZ database name at https://en.wikipedia.org/wiki/List_of_tz_database_time_zones

Notification Tab

This tab contains all POM-defined notification types and associated e-mail addresses and retention periods. For a list of notification types, refer to Emails and Notifications.

Table 8-14 Notifications Configuration

Field	Description	Modifiable?
Notification Type	Type of notification. There are several events in POM for which notifications are generated. An example of a notification type is: NightlyStart.	No
Email Subscription	Email address to which notifications of this type will be sent.	Yes – Valid values are blank or correctly formed email addresses.
Retention Period	Period in days notifications generated for this type are to be retained in the system before purging..	Yes – Valid values are blank or an integer number of days.

Custom Entities Configuration

Custom Entities include custom flows, processes and jobs. POM provides a feature, in the user interface, to export and import configuration of these custom entities.

Custom entities exported on a specific environment have to be imported back on a different environment without any modifications. Once these are configured and fine-tuned on an environment (for example, Stage) the export/import feature can be used to export the configuration as a JSON file then import it back into another environment (for example, production).

> **Note:**
>
> Custom entities configuration can't be exported as or imported from spreadsheet. Only JSON export/import feature is supported for custom entities.

Following are the validation rules for custom entities import:

- If custom flows/processes are present in both source and target environments, importing custom entities won't add new entities, meaning the makeup of custom Flows or Processes in the target environment can't be altered.
- If custom flows are not present in the target environment, import will create these custom flows in the target environment.
- If custom processes are not present in the target environment, import will create those in the target environment.

9
Emails and Notifications

This chapter provides the list of notifications sent by POM to alert users about important events that occur during the batch lifecycle.

Notifications

Notifications generated by POM can be broadly put into the following categories.

1. **Schedule-level Notifications** – The configurations done here apply equally to all Batch Schedules within the system.
2. **Job-level Notifications** – These need to be configured at a Job level. This is done from the Batch Administration screen, while editing a Job.

All Notification Types within POM can be grouped together, based on the event that generates them as seen in the sections below.

POM Post Install Validation

The following notifications can be triggered during POM post-install validation step.

Event	Notification Type	Intended audience
Exception event during POM post install validation service	**POMUpgradeValidation** This notification is sent when there is any failure during asynchronous processing of POM post install validation service.	All

Schedule Upgrade

The following notifications are triggered, when a Schedule is upgraded.

Event	Notification Type	Configured At	Intended audience
Start of Schedule Upgrade	**BatchScheduleImport** Created when a Schedule update is triggered.	Schedule	All
Completion of Schedule Upgrade	**BatchScheduleImport** Created when a Schedule update has completed	Schedule	All

Schedule Configuration Import

The following notifications are triggered, when a configuration import is initiated by clicking the Import Config button on the Batch Administration screen.

Event	Notification Type	Configured At	Intended audience
Completion of Configuration import	**BatchScheduleConfigImport** Created at the completion of the import process. See Schedule Config Import Summary Email below.	Schedule	All

Customer Modules Synchronization (Retail Home)

The following notifications are triggered, when the “Sync with MDF” button is clicked on the Batch Administration screen.

Event	Notification Type	Configured At	Intended audience
On completion of MDF synchronization with Retail Home	**ApplicationModuleEnabled** Created if an Application or Module were to get enabled, during the process. Indicates that new Jobs may have been enabled.	Schedule	All
On completion of MDF synchronization with Retail Home	**ApplicationModuleDisabled** Created if an Application or Module were to get disabled, during the process. Indicates that existing Jobs may have been disabled.	Schedule	All

Scheduler Task Execution

The following notifications are generated by the Scheduler, when it runs into any issues with Scheduler Task execution.

Event	Notification Type	Configured At	Intended audience
On Scheduler Task firing, at its designated time	**SchedulerTaskFailed** Created for unexpected errors during Scheduler Task processing	Schedule	All
On scheduling a Scheduler Task at Scheduler Day creation time	**SchedulerTaskDelayed** Created for Tasks, that cannot be scheduled as their designated time to run has passed.	Schedule	All
On scheduling a Scheduler Task at Scheduler Day creation time	**SchedulerTaskSkipped** Created if a Task fails to get scheduled, due to unexpected errors.	Schedule	All
On scheduler task firing at its designated time	**SchedulerTaskCancelled** Created when the scheduler task is cancelled when the scheduler is disabled.	Schedule	All

Event	Notification Type	Configured At	Intended audience
On scheduling the Nightly Scheduler Task at Scheduler Day creation time	**SingleNightlyPerDayValidation** Created when the Nightly Task is delayed by a day, due to the validation which ensures only one Nightly runs, per day	Schedule	All

Execution Engine

The following notifications are generated by the Execution Engine, in case it runs into any issues.

Event	Notification Type	Configured At	Intended audience
Exception event during Engine processing	**ExecutionEngineIssue** This notification is sent when there are important/critical events/failures with the Execution Engine, such as • Job Agent invocation failure • Execution Request stuck in SUBMITTING state	Schedule	All

Scheduler Day Creation

The following notifications are all configured at a Schedule level, and triggered during the creation of a Scheduler Day.

Event	Notification Type	Intended audience
Scheduler Day creation	**InterSchedDepIssue** Created when either of the following occurs • The business date between the interdependent Schedules vary by more than a day. • The External Schedule is day ahead of the current Schedule and previous day data is not available. • The inter-schedule dependencies are not valid and disabled. • A schedule includes inter-schedule dependencies or execution links that are not valid	All
Scheduler Day creation	**NewSchedulerDayFailure** Created when the Scheduler Day creation fails.	All
Completion of Scheduler Day creation	**ScheduleChangesSummaryReport** Created after a Scheduler Day is successfully created, to create a report of the changes between the current Scheduler Day and the previous one. See Schedule Change Summary Email below.	All

Hourly Flow Execution

The following notifications are triggered, during the execution of Hourly Flows.

Event	Notification Type	Configured At	Intended audience
Hourly Flow Skipped	**IntradayCycleSkipped** Created when an Hourly Flow is skipped	Schedule	All
Hourly Flow Completed	**IntradayCycleCompleted** Created when an Hourly Flow is completed	Schedule	All
Hourly Flow Summary Report	**IntradayCycleSummaryReport** Created at the completion of a Hourly Flow, in order to send out a summary of its execution. See Hourly Flow Summary Email below	Schedule	All
Hourly Flow Pending execution	**HourlyPending** Created by the Execution Engine on encountering Hourly Execution Requests that are awaiting execution	Schedule	All

Nightly Flow Execution

The following notifications are triggered, during the execution of the Nightly Flow

Event	Notification Type	Configured At	Intended audience
Start of Nightly Flow	**NightlyStart** Created when the Nightly Flow starts.	Schedule	All
Completion of Nightly Flow	**NightlyBatchCompleted** Created at the completion of the Nightly Flow.	Schedule	All
Completion of Nightly Flow	**NightlySummaryReport** Created at the completion of the Nightly Flow, in order to send out a summary of its execution. This notification is for internal Oracle internal use. See Nightly Summary Email below.	Schedule	Oracle Internal
Completion of Nightly Flow	**NightlySummaryReportExternal** Created at the completion of the Nightly Flow, in order to send out a summary of its execution This notification is for customers use. See Nightly Summary Email below.	Schedule	Customers / System Integrators

Job Execution

The following notifications are triggered during the course of execution of the Job.

Event	Notification Type	Configured At	Intended audience
Start of Job	**JobStarted** Created when a Job starts. See Job Start Email below.	Job	All
Completion of Job	**JobCompleted** Created when a Job completes successfully. See Job Completion Email below.	Job	All
Job runs for longer than expected	**LongRunningJob** Created when the Job runs longer than its configured or calculated threshold runtime.	Schedule	All
Job Execution failure	**ErrorNotification** Created when a running Job fails. This notification is for Oracle internal use. See Job Error Email below.	Schedule	Oracle Internal
Job Execution failure	**ErrorNotificationExternal** Created when a running Job fails. This notification is for customers use. See Job Error Email below.	Schedule	Customers / System Integrators
Job waiting for execution	**ExternalDepPending** Created when a Job is waiting upon an External Dependency.	Schedule	All
Job waiting for execution	**InterSchedDepPending** Created when a Job is waiting upon an Inter-schedule Dependency	Schedule	All
Completion of Job, with warning	**JobCompletedWithWarning** This notification is sent when a job completes with a warning. This indicates that this job's shell script exited with a code that was defined in the System Options as a '*Completion with Warning code*'. For more information on how these codes are setup, see the SystemOption Tab in the A Batch Schedule Spreadsheet Template.	Schedule.	All
Execution Link failure	**ExecutionLinkIssue** Created when Execution Links fail. They are run, after the source Job has been invoked by POM (regardless of its status)	Schedule	All

External Integration

The following notifications are created, when integrating with External systems.

Event	Notification Type	Configured At	Intended audience
Release External Dependency	**ExternalDepComplete** Created when an External Dependency is released	Schedule	All
Callback publish failure	**CallbackFailure** Created when a Callback fails to get published.	Schedule	All

Batch Entity Operations

The following notifications are triggered, when Custom Batch Entities are created, modified or deleted.

Event	Notification Type	Configured At	Intended audience
Custom Batch Entity API invocation	**BatchEntityUpdateEvent** Created when a Custom Batch Entity (Flow or Process) is either created, modified or deleted.	Schedule	Oracle Internal

General

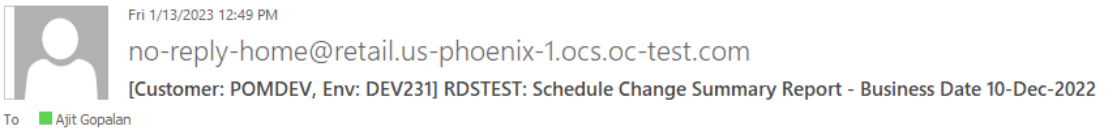
Event	Notification Type	Configured At	Intended audience
System Information	**InformationNotification** Internal informational	Schedule	Oracle Internal
System Warning	**WarningNotification** Internal warning	Schedule	Oracle Internal
System Error	**SystemErrorNotification** Internal error	Schedule	Oracle Internal

Emails

By default, all notifications are shown on the POM application. It is an option to configure notifications to also send e-mails. This is accomplished through the Notifications Administration function of Retail Home. Refer to the “Notifications Administration” chapter of the *Retail Home Administration Guide* for more information.

Schedule Change Summary Email

This summary is generated as part of the Scheduler Day creation process. It highlights any differences between the configuration of the previous Scheduler Day and the current. It also highlights differences in scheduling between the Schedule instances. This email has no attachments.



Fri 1/13/2023 12:49 PM
no-reply-home@retail.us-phoenix-1.ocs.oc-test.com
[Customer: POMDEV, Env: DEV231] RDSTEST: Schedule Change Summary Report - Business Date 10-Dec-2022
To Ajit Gopalan

Changes in Scheduler day:903 compared to the previous day:902

Modification in Job(s) are as below

Application	Cycle Name	Process	Job	Current Status	Previous Status	Comments
APP1	Adhoc	TEST_C_PROCESS	TEST_C_JOB	DISABLED	COMPLETED	-

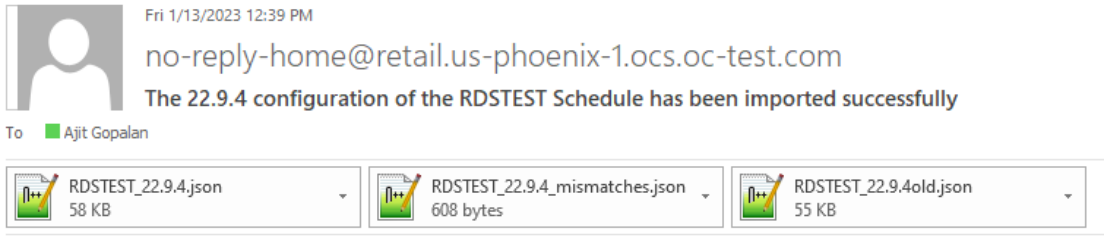
There is no modification in External or Inter-Schedule Dependency.

There is no modification in Scheduler Tasks.

Schedule Config Import Summary Email

This summary is generated once a configuration has been imported successfully into POM using the Import Config button of the Batch Administration screen. This email provides the following attachments

1. Batch Configuration prior to the Import (JSON). This backup can be used to restore the configuration, in case the new configuration is not good enough.
2. The new Batch Configuration.
3. A JSON that highlights the mismatches between the previous and the new configuration.

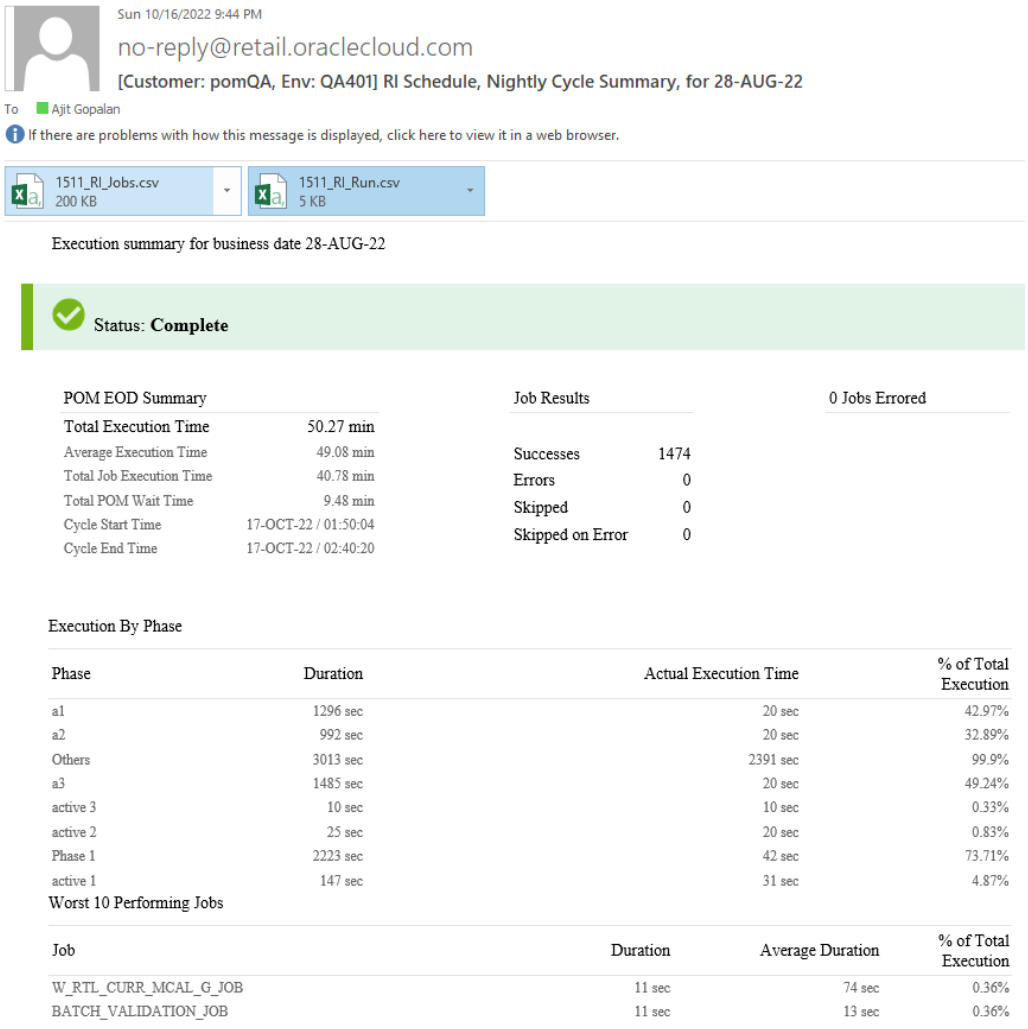


Fri 1/13/2023 12:39 PM
no-reply-home@retail.us-phoenix-1.ocs.oc-test.com
The 22.9.4 configuration of the RDSTEST Schedule has been imported successfully
To Ajit Gopalan
RDSTEST_22.9.4.json 58 KB
RDSTEST_22.9.4_mismatches.json 608 bytes
RDSTEST_22.9.4old.json 55 KB

Nightly Summary Email

At the end of the Nightly Cycle, a summary report is created that is sent as an email. This report is sent to email addresses, configured for the **NightlySummaryReportExternal** and **NightlySummaryReport** Notification Types. The former Notification Type is meant to be subscribed by Customers, while the latter is for Oracle Internal.

This email also contains two attachments. One is a summarized report (CSV) of all the Job executions of that Nightly cycle. The other attachment contains a summarized report of the Nightly cycle completion times for the last 15 days. The latter attachment is only available to email recipients of the **NightlySummaryReport** Notification Type.



Sun 10/16/2022 9:44 PM
no-reply@retail.oraclecloud.com
[Customer: pomQA, Env: QA401] RI Schedule, Nightly Cycle Summary, for 28-AUG-22
To Ajit Gopalan
If there are problems with how this message is displayed, click here to view it in a web browser.
1511_RI_Jobs.csv 200 KB
1511_RI_Run.csv 5 KB

Execution summary for business date 28-AUG-22

Status: **Complete**

POM EOD Summary	
Total Execution Time	50.27 min
Average Execution Time	49.08 min
Total Job Execution Time	40.78 min
Total POM Wait Time	9.48 min
Cycle Start Time	17-OCT-22 / 01:50:04
Cycle End Time	17-OCT-22 / 02:40:20

Job Results	
Successes	1474
Errors	0
Skipped	0
Skipped on Error	0

0 Jobs Errored

Execution By Phase

Phase	Duration	Actual Execution Time	% of Total Execution
a1	1296 sec	20 sec	42.97%
a2	992 sec	20 sec	32.89%
Others	3013 sec	2391 sec	99.9%
a3	1485 sec	20 sec	49.24%
active 3	10 sec	10 sec	0.33%
active 2	25 sec	20 sec	0.83%
Phase 1	2223 sec	42 sec	73.71%
active 1	147 sec	31 sec	4.87%

Worst 10 Performing Jobs

Job	Duration	Average Duration	% of Total Execution
W_RTL_CURR_MCAL_G_JOB	11 sec	74 sec	0.36%
BATCH_VALIDATION_JOB	11 sec	13 sec	0.36%

Hourly Flow Summary Email

This email is generated when a Flow on the Hourly Cycle completes successfully. It is sent to the email addresses that subscribed to the Notification Type – **IntradayCycleSummaryReport**.

Tue 8/2/2022 7:46 AM

no-reply@retail.oraclecloud.com

[Customer: POMQAChnage, Env: STAGING1000] MERCH Batch Execution Summary for 07-Mar-2022

To Ajit Gopalan

Execution Summary for Business Date: 07-Mar-2022

Statistics for Hourly_Cycle_3 : Flow:ACTIVITY_SCHED_PURGE_CYCLE

Total Execution Time (minutes)	Cycle Start Time	Cycle End Time	Job Count			
			Success	Error	Skipped	Skipped On Error
0	08/02/2022 12:43:05	08/02/2022 12:43:05	0	0	1	0

Skipped Job(s)
ACTIVITY_SCHED_PURGE_JOB

Execution Comments :

None

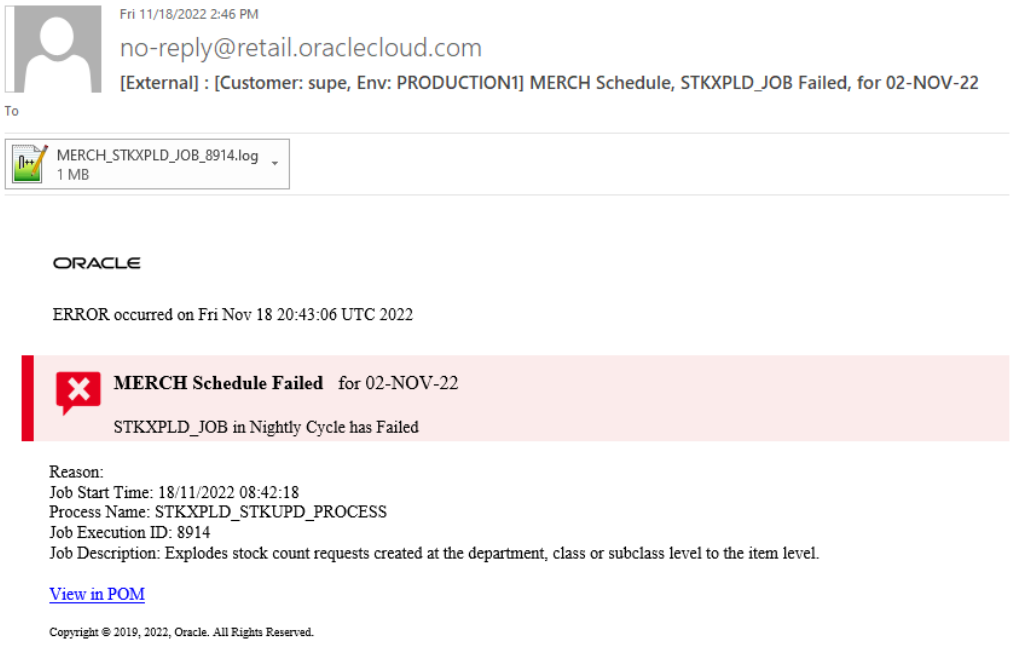
Addition Information :

None

Job Error Email

When a Job fails during execution, a Job Error email is sent. This email will include as an attachment the log file of the failed Job execution. This email will be generated, even if the `skip-on-error` flag is set on the Job.

> **Note:**
>
> The Job Error email sent to email addresses configured on the **ErrorNotification** type, will contain the log file attachment, while the email sent to email addresses configured on the **ErrorNotificationExternal** type will not contain this attachment.



Fri 11/18/2022 2:46 PM

no-reply@retail.oraclecloud.com

[External] : [Customer: supe, Env: PRODUCTION1] MERCH Schedule, STKXPLD_JOB Failed, for 02-NOV-22

To

MERCH_STKXPLD_JOB_8914.log
1 MB

ORACLE

ERROR occurred on Fri Nov 18 20:43:06 UTC 2022

MERCH Schedule Failed for 02-NOV-22

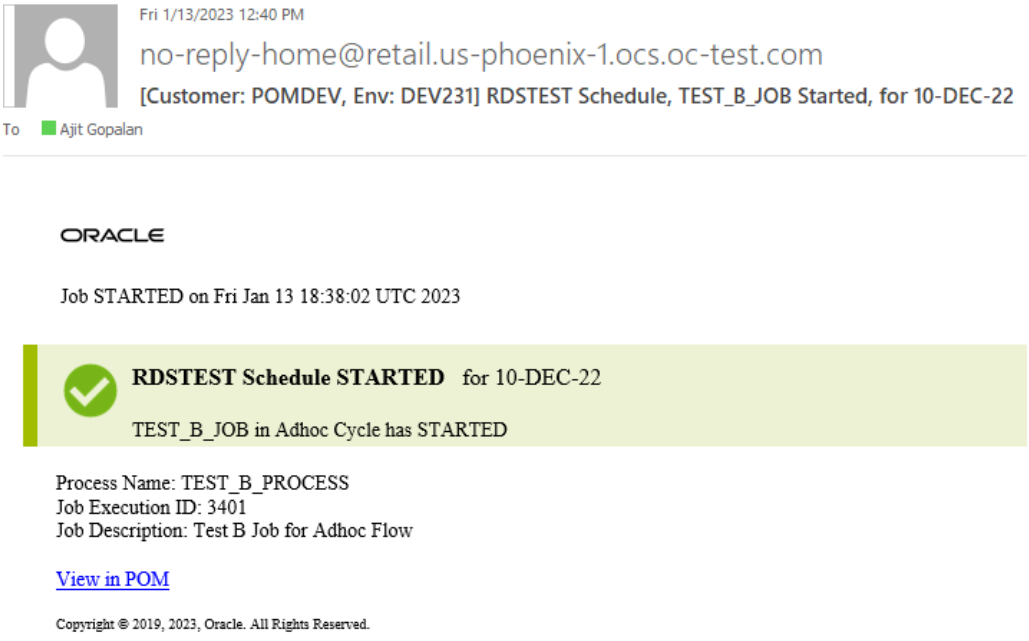
STKXPLD_JOB in Nightly Cycle has Failed

Reason:
Job Start Time: 18/11/2022 08:42:18
Process Name: STKXPLD_STKUPD_PROCESS
Job Execution ID: 8914
Job Description: Explodes stock count requests created at the department, class or subclass level to the item level.

View in POM

Job Start Email

To notify of a Job starting, is a setting available at an individual Job. This option is specified on the Edit Job function of the Batch Administration screen. This email has no attachments.



Fri 1/13/2023 12:40 PM

no-reply-home@retail.us-phoenix-1.ocs.oc-test.com

[Customer: POMDEV, Env: DEV231] RDSTEST Schedule, TEST_B_JOB Started, for 10-DEC-22

To Ajit Gopalan

ORACLE

Job STARTED on Fri Jan 13 18:38:02 UTC 2023

RDSTEST Schedule STARTED for 10-DEC-22

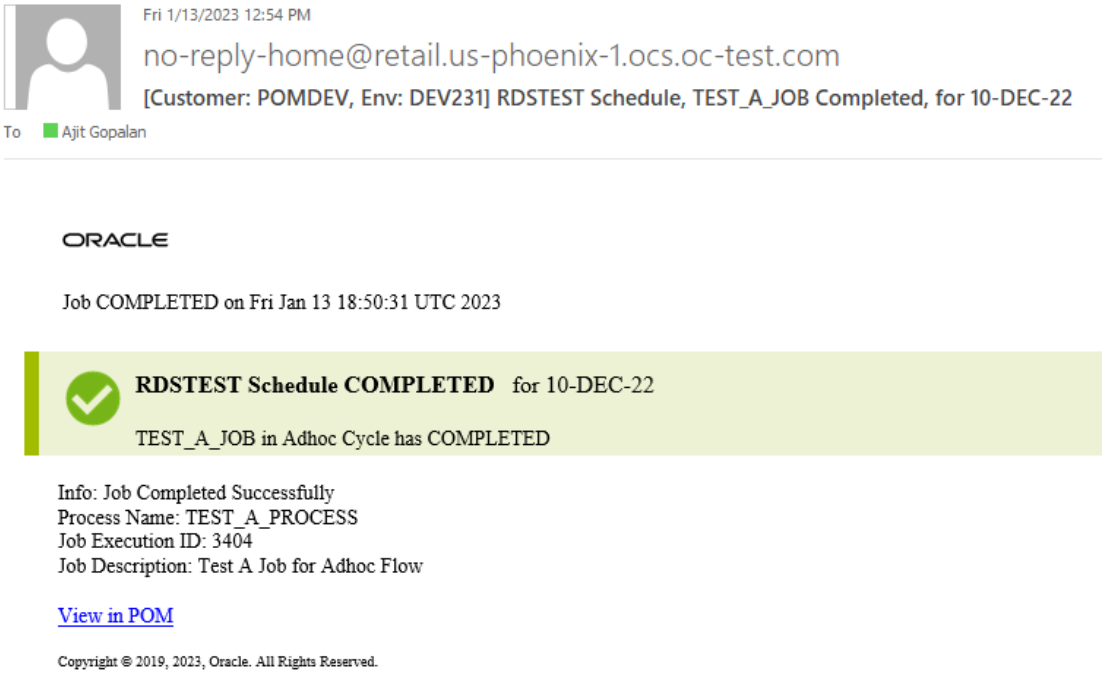
TEST_B_JOB in Adhoc Cycle has STARTED

Process Name: TEST_B_PROCESS
Job Execution ID: 3401
Job Description: Test B Job for Adhoc Flow

View in POM

Job Completion Email

To notify of a Job completing successfully, is a setting available at an individual Job. This option is specified on the Edit Job function of the Batch Administration screen. This email has no attachments.



Fri 1/13/2023 12:54 PM

no-reply-home@retail.us-phoenix-1.ocs.oc-test.com

[Customer: POMDEV, Env: DEV231] RDSTEST Schedule, TEST_A_JOB Completed, for 10-DEC-22

To Ajit Gopalan

ORACLE

Job COMPLETED on Fri Jan 13 18:50:31 UTC 2023

RDSTEST Schedule COMPLETED for 10-DEC-22

TEST_A_JOB in Adhoc Cycle has COMPLETED

Info: Job Completed Successfully
Process Name: TEST_A_PROCESS
Job Execution ID: 3404
Job Description: Test A Job for Adhoc Flow

View in POM

10

User Roles and OAuth Scopes

This chapter lists the pre-loaded user roles/scopes in POM.

Table 10-1 User Roles and OAuth Scopes

Roles	Scopes	Description
BATCH_ADMINISTRATOR_JOB	rgbu:pom:services-administrator	Users within this role are retailer administrators with full access to all POM actions. They monitor, maintain and configure the batch schedules. They may also maintain POM application configurations for efficient operations. They troubleshoot batch issues and work with Oracle support personnel to address those issues. Finally, they may apply batch schedule patches and upgrades. Additionally, users assigned this role are given access to the Oracle AMS Utilities screen.
BATCH_VIEWER_JOB	rgbu:pom:services-viewer	Users within this role are retailer business users responsible for just monitoring batch. They have view access to all POM screens except AMS Utilities.
BATCH_SCHEDULE_CONFIGURATION_MANAGER_JOB	rgbu:pom:services-customer-manager	Users within this role are typically retailer administrators responsible for just monitoring batch and configuring external dependencies and callbacks into the Company's systems. They have view access to all POM screens except AMS Utilities.
BATCH_SCHEDULE_ADMINISTRATOR_JOB	rgbu:pom:services-customer-administrator	Users within this role are typically retailer administrators responsible for maintaining monitoring and executing batch. They have view access to all POM screens except AMS Utilities. They can perform select activities on the Batch Monitor screen to move the schedule along. They also have update access to the Batch Administration screen. They can also configure some application properties and can configure a new schedule
BATCH_ORACLE_AMS_ADMINISTRATOR_JOB	rgbu:pom:services-ams-administrator	Users within this role are typically Oracle AMS administrators who monitor, maintain and configure the batch schedules. They also maintain POM application configurations for efficient operations. They troubleshoot batch issues and work with other Oracle development and support personnel to address those issues. Finally they apply POM and batch schedule patches and upgrades.

For more information regarding functional access of each POM role, refer to the *Oracle® Retail Process Orchestration and Monitoring Cloud Services Security Guide*.

11

Invoking POM Services

This chapter lists the ReST APIs that can be invoked and the steps needed to invoke them.

All the POM APIs are protected by the Oauth standard. It allows users to securely delegate access to resources without sharing their original credentials. Oauth2 has been around since 2012 as a standard and is built on lessons from other, earlier standards, including Oauth1 and SAML.

> **Note:**
>
> ReST service calls from POM to external systems (customers), such as the call for External Status Update, are limited to Basic Auth at this time.

Oauth Token Generation

Using the Oauth protocol is a two-step process:

- Request an access token from an authentication provider: IDCS or OCI IAM.
- Provide the access token as an authorization header when invoking a service.

Prerequisite

Customers are required to create an OAuth client using the Retail Home Create IDCS OAuth 2.0 Client function. The OAuth client must be created against the "POM" app with the scope

```
rgbu:pom:services-customer-administrator-<ENV_ID>
```

where <ENV_ID> represents the unique environment identifier such as PRD1, STG1, DEV1 and so on.

For example, the DEV1 scope would be:

```
rgbu:pom:services-customer-administrator-DEV1
```

For more information about creating the OAuth client (one-time setup), refer to the "Creating IDCS OAuth 2.0 Client Apps" chapter in the *Retail Home Administration Guide*.

Invoke IDCS Token Endpoint

To generate a token from IDCS, the following information is needed:

- IDCS URL
- Client Id and Client Secret

- OAuth Scope

The curl command below invokes an IDCS service to generate an access token:

```
curl -I
  -H 'Authorization: Basic <base64Encoded Oauth_Clientid:Secret>'
  -H 'Content-Type: application/x-www-form-urlencoded;charset=UTF-8'
  --request POST <IDCS_URL>/oauth2/v1/token
  -d 'grant_type=client_credentials&scope=rgbu:pom:services-customer-
administrator-<ENV_ID>'
```

This is a standard ReST call, with the following specifics:

- `<IDCS URL>` is the IDCS URL of this instance.
- `<base64Encoded OAuth_Clientid:Secret>` is the Base64-encoded OAuth Client Id and Client Secret provided as a Basic Authentication header.
- Specify the body as:

  ```
  grant_type=client_credentials&scope=
  rgbu:pom:services-customer-administrator-<ENV_ID>
  ```

  The response to this call will be in this format:

  ```
  {
        "access_token": "<TOKEN>",
        "token_type": "Bearer",
        "expires_in": 3600
  }
  ```

Invoking the POM Service

To invoke the POM ReST service, you must add an authorization header as Bearer <token>, that is:

- The word Bearer
- A space
- A valid token obtained as described above

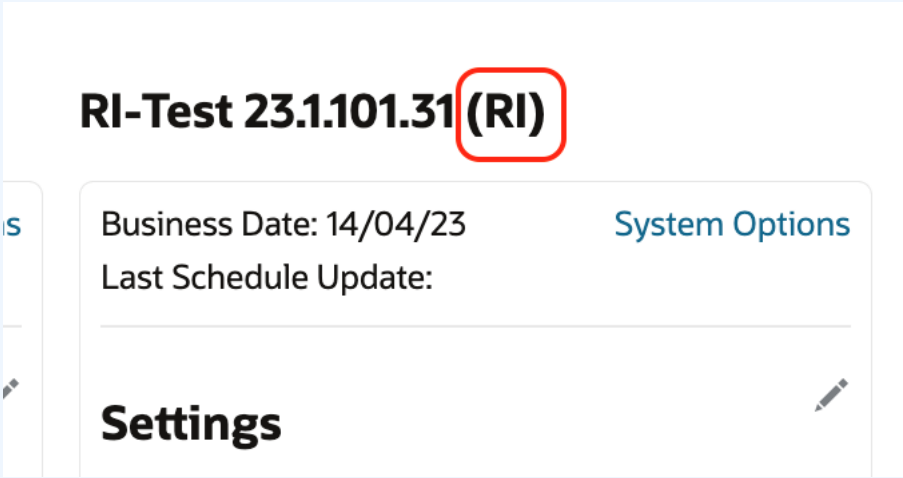
For example, the POM nightly cycle start request would look something like the following:

```
curl -i
  -H 'Authorization: Bearer <OAuth Token>'
  -H 'Content-Type: application/json'
  --request POST 'https://<pom-server-host>/ProcessServices/services/
private/executionEngine/schedules/<Schedule_Name>//execution?
skipVersion'
  -d '{ "cycleName" : "Nightly",  "flowName" : "Nightly"}'
```

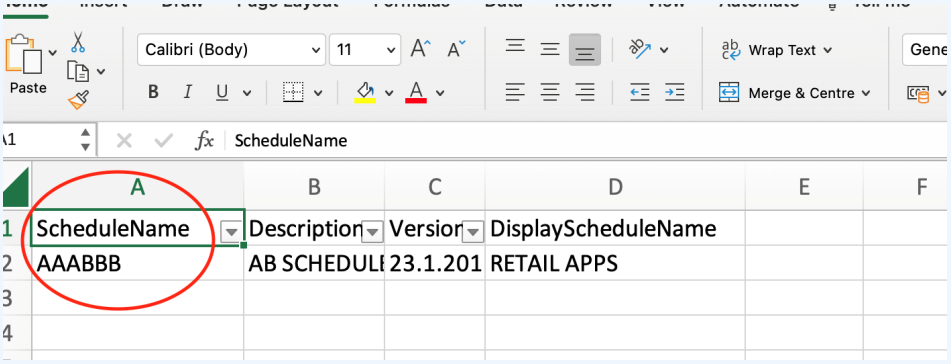
Note:

There are POM APIs where the schedule name must be passed in the URL itself, or as part of the request payload. To get the schedule name, follow one of the following two methods:

1. Login to POM UI and open the System Configuration screen from the Tasks menu. Under the **Schedules** tab, at the top of the desired schedule column, look at the name in parentheses. In the screenshot below for example, the schedule name to use in the API calls is **RI**. **RI-Test** is the Display Schedule Name.



2. In the Batch Schedule spreadsheet, get the schedule name from the `ScheduleName` field in the Schedule sheet. Refer to the screenshot below, for example:



Schedule Information API

These APIs provide general information regarding the Batch Schedule in POM.

View Batch Schedules

This endpoint will fetch all the Batch Schedules within POM.

HTTP Method	GET
Path	`https://<pom-server-host>/ProcessServices/ services/public/schedules/` **<pom-server-host>** - This is the POM URL
HTTP Headers	`Content-Type = application/json` *Include Authorization header as shown in* Oauth Token Generation
Response Body	Sample Response

```
{
  "schedules": [
     {
        "name": "MERCH",
        "displayScheduleName": "MERCH",
        "description": "Merch Batch Schedule",
        "version": "23.0.101",
        "creationDate": "2023-03-23"
     }
  ]
}
```

Parameter	Description
Schedules	Array containing all the schedules
name	Name of the Batch Schedule
displayScheduleName	The name of the Schedule to be used for display purposes within POM.
description	Description of the Batch Schedule
version	Version of the Batch Schedule
creationDate	Creation Date

View Batch Cycles for Schedule

This endpoint will provide a list of Batch Cycles for the given schedule.

HTTP Method	GET
Path	`https://<pom-server-host>/ProcessServices/ services/public/schedules/<schedule-name>/cycles` **<pom-server-host>** - This is the POM URL. **<schedule-name>** - This is the name of the Schedule.
HTTP Headers	`Content-Type = application/json` *Include Authorization header as shown in* Oauth Token Generation

Response Body Sample Response

```
{
  "cycles": [
    {
      "cycleNumber": 1,
      "cycleName": "Adhoc",
      "cycleType": "A",
      "sequenceNumber": 1,
      "invokableCount": 225
    },
    {
      "cycleNumber": 2,
      "cycleName": "Nightly",
      "cycleType": "N",
      "sequenceNumber": 1,
      "jobCount": 385
    }
  ]
}
```

Parameter	Description
cycles	Array containing all the cycles
cycleNumber	Cycle Number
cycleName	Cycle Name
cycleType	Type of Batch Cycle: • **Adhoc** - Adhoc or Standalone Cycle • **Nightly** - Nightly or EOD Cycle. • **Hourly_Cycle**_<**N**> - Hourly or Recurring Cycle. The <**N**> is a number between 1 and 24.
invokableCount	The number of invokeable batch entities on the Adhoc Cycle. This value is only defined for the Adhoc Cycle, and will not be present for the other Cycles.
jobCount	The number of invokeable Jobs on the Cycle. This value is present for the Nightly and Hourly Cycles only.

Batch Execution API

Different SaaS customers operate in different models for running their batch. Some may choose to use the POM Scheduler to schedule the different entities such as Nightly, Recurring or Standalone. Refer to the *POM User Guide* for documentation on the POM Scheduler.

Others may choose to control the time and frequency of batch executions by invoking the provided ReST APIs. This section describes these APIs.

Execution Request Creation

POM also provides users the capability to control the time and frequency of batch executions by invoking the following ReST service.

HTTP Method	POST
Path	`https://<pom-server-host>/ProcessServices/services/private/executionEngine/schedules/<schedule-name>/execution` **<pom-server-host>** - This is the POM URL. **<schedule-name>** - Name of the schedule being invoked.
HTTP Headers	`Content-Type = application/json` *Include Authorization header as shown in* Oauth Token Generation

Request Body	`{`
	`"cycleName" : "<Cycle name>",`
	`"flowName" : "<Flow name>",`
	`"processName" : "<Process name>",`
	`"requestParameters" : "<Comma-separated key-value pairs>"`
	`}`

Parameter	Description
`cycleName`	Name of the Batch Cycle being invoked. Valid values are: • **"Nightly"** • **"Adhoc"** • **"Hourly_Cycle_N"** – (Replace `N` with the Hourly cycle number)
`flowName`	Name of the Batch Flow being invoked. In case of the Nightly Cycle invocation, the value is "`Nightly`". For Adhoc Processes, the value is "`Adhoc`" itself.
`processName`	Name of the Batch Process being invoked. Needed only when invoking Processes on Adhoc Cycles.
`requestParameters`	Optional attribute. This is useful if external systems are providing custom identifiers to POM and expect them to be returned on callbacks sent from POM.

Response Body

```
{
   "value"               :" Execution Request ID",
   "cycleName"           :" Cycle name",
   "flowName"            :" Flow name",
   "processName"         :" Process name",
   "requestType"         :" Request type",
   "requestParameters"   :" Comma-separated key-value
pairs",
   "executionEngineInfo" :" Status of Execution Engine",
   "hyperMediaContent"   : {}
}
```

Parameter	Description
value	This unique identifier is the Execution Request ID. This can be used for tracking purposes.
executionEngineInfo	Indicates the status of the Execution Engine after creating the request. If this is not STARTED, then the Execution Engine may be experiencing some issues.
hyperMediaContent	Can be ignored. Used internally by the ReST APIs as part of the HATEOAS standards.

Examples of all the Batch Entities in POM that can be invoked by the endpoint above, are shown in the table below

Invocation	Request Payload
Nightly Flow	`{ "cycleName": "Nightly", "flowName": "Nightly", "requestParameters": "callerId=X,correlationId=123" }` **Note:** The Nightly Cycle contains a single default Nightly Flow. Hence a single invocation will suffice to start the Nightly Flow.
Hourly Flow	`{ "cycleName": "Hourly_Cycle_<N>", "flowName": "SALESPROCESS_FLOW", "requestParameters": "callerId=X,correlationId=456" }` **<N>** - This is the cycle number (1 to 24) **Note:** The Hourly Cycles comprise of many distinct Batch Flows and for each Batch Flow a separate invocation is required.

Invocation	Request Payload

Adhoc Flow	<pre>{
	"cycleName": "Adhoc",
	"flowName": "SALESPROCESS_ADHOC_FLOW",
	"requestParameters": "callerId=X,correlationId=456"
	}</pre>
	Note: The Adhoc Cycle comprises of many distinct Batch Flows and for each Batch Flow a separate invocation is required.

Adhoc Process	<pre>{
	"cycleName": "Adhoc",
	"flowName": "Adhoc",
	"processName": "RPM_LOCATION_PROCESS _ADHOC",
	"requestParameters": "callerId=E,correlationId=789"
	}</pre>
	In case of Adhoc Batch Processes, the parameters for Batch Jobs can be overridden by parameters specified as part of the invocation request.
	
	<pre>{
	"cycleName": "Adhoc",
	"flowName":"Adhoc ",
	"processName": "RPM_LOCATION_PROCESS _ADHOC",
	"requestParameters": "jobParams.RPM_JOB=param1|param2"
	}</pre>
	Note: Adhoc Cycles are composed of many discrete individual Batch Processes. For each Batch Process, a separate invocation is required.

Execution Request Status

The endpoint below provides the ability to check the status of an Execution Request in POM

HTTP Method	GET
Path	https://**<pom-server-host>**/ProcessServices/services/private/executionEngine/schedules/**<schedule-name>**/requests/**<execution-Id>**

Parameter	Description
<pom-server-host>	This is the POM URL.
<schedule-name>	Name of the schedule being invoked.
<executionId>	ID of the Execution Request returned by POM.

HTTP Headers

```
Content-Type = application/json
```

Include Authorization header as shown in Oauth Token Generation

Response Body

```
{
    "executionId": " Execution Request ID",
    "scheduleName": " Schedule Name",
    "cycleName": " Cycle name",
    "flowName": " Flow name",
    "processName": " Process name",
    "requestType": " Request type",
    "requestParameters": " Comma-separated key-value
pairs",
    "executionEngineInfo": " Status of Execution Engine",
    "hyperMediaContent": {}
}
```

Parameter	Description
executionId	ID of the execution request.
scheduleName	Name of the schedule.
cycleName	Name of the Cycle for which the execution request was created.
flowName	Name of the Flow for which the execution request was created
processName	Name of the Process. For Nightly/Hourly this is set to "ALL".
requestParameters	Parameters associated with the execution request.
status	Status of the execution request. Possible Values • **QUEUED** - Request is queued up for execution. • **RUNNING** - Jobs from this request are being executed. • **ERROR** - One of the job in this request has failed. Note that a failed job would be restarted by POM Admin; there is no need to re-submit the execution request. • **COMPLETED** - All jobs from this request were executed successfully.

Execution Requests for Batch Cycle

The endpoint fetches all the Execution Requests for a given Schedule and given Batch Cycle

HTTP Method GET

Path	`https://<pom-server-host>/ProcessServices/ services/public/ schedules/<schedule-name>/ executionRequests?cycleName=XX` **<pom-server-host>** - This is the POM URL. **<schedule-name>** - Name of the schedule being invoked. **cycleName** - Mandatory query parameter. This is the name of the Batch Cycle.
HTTP Headers	`Content-Type = application/json` Include Authorization header as shown in Oauth Token Generation

Response Body Sample Response

```
{
   "schedulerDay": 2001,
   "businessDate": "2023-4-18",
   "execRequests": [
      {
         "requestType": "SCHEDULER_REQUEST",
         "flowExecId": 121,
         "executionId": 2343,
         "status": "LOADED",
         "cycleName": "Nightly",
         "flow": "Nightly",
         "totalJobs": 385,
         "completedJobs": 0,
         "loadedJobs": 383,
         "skippedJobs": 2,
         "errorJobs": 0,
         "systemHeldJobs": 0,
         "submittingJobs": 0,
         "submittedJobs": 0
      }
   ]
}
```

Parameter	Description
schedulerDay	Current Scheduler Day ID
businessDate	Business Date of the current scheduler day
execRequests	Array containing all execution requests
requestType	Type of execution request • **SCHEDULER_REQUEST** - Created by the POM Scheduler • **MANUAL_RUN** - Run manually from the POM UI • **UNDEFINED** - Not defined. Possibly created through programmatic invocation of the API.
flowExecId	Unique ID generated for the Flow Execution
executionId	Execution ID to track this execution request
status	Status of the execution request.
cycleName	Name of the Batch Cycle on which the Flow or Process exists.
flow	Name of the Flow being executed. For a Process execution, the Flow is Adhoc.
totalJobs	Total number of Jobs
completedJobs	Total number of COMPLETED Jobs
skippedJobs	Total number of SKIPPED Jobs
errorJobs	Total number of Jobs in ERROR

Parameter	Description
submittedJobs	Total number of SUBMITTED Jobs
submittingJobs	Total number of SUBMITTING Jobs
systemHeldJobs	Total number of SYSTEMHELD Jobs

External Dependency API

This API operates solely on Batch External Dependencies.

Releasing External Dependency

The endpoint below releases an External Dependency setup in POM. Customers can call this endpoint to synchronize the execution of the Batch Schedule in POM with their other systems.

HTTP Method	POST
URL	https://**<pom-server-host>**/ProcessServices/services/private/schedules/**<schedule-name>**/external/jobs/**<ext-dependency-name>**/status/**COMPLETED** **<pom-server-host>** - This is the POM URL. **<schedule-name>** - Name of the schedule being invoked. **<ext-dependency-name>** - Name of the External Dependency set up in POM. **Note:** Ensure the status **COMPLETED** is specified in the path correctly, or the External Dependency will not be released.
HTTP Headers	Content-Type = application/json Include Authorization header as shown in Oauth Token Generation
Request Body	None

HTTP Method	POST

Response Body	<pre>{
	  "value": "true",
	  "links": [],
	  "hyperMediaContent": {
	  "linkRDO": []
	  }
	}</pre>

Parameter	Description
value	Boolean attribute indicating the success or failure of the attempt to release the external dependency.
links	Can be ignored. Part of the ReST API, HATEOAS standards.
hyperMediaContent	Can be ignored. Part of the ReST API, HATEOAS standards.

Utilities API

These APIs are general utilities, that have come as requirements from various Customers.

Business Date Alignment

This API provides the ability to adjust the business date of a Batch Schedule. This allows for aligning the business date with other schedules or a customer's internal processing date.

HTTP Method	POST
Path	`https://<pom-server-host>/ProcessServices/services/public/administration/utilities/alignBusinessDate` **<pom-server-host>** - This is the POM URL.
HTTP Headers	`Content-Type = application/json` Include Authorization header as shown in Oauth Token Generation

HTTP Method	POST

Request Body

```
{
  "businessDate": "Business Date in yyyy-MM-dd format",
  "scheduleName": "Schedule Name",
  "advanceDateOnly": "<true|false>",
  "updateDependentSchedules": "<true|false>",
  "comment": "<Comment>"
}
```

businessDate – Desired business date in yyyy-MM-dd format (API will determine if to move it backward or forward)

scheduleName – Schedule name for which the business date has to be aligned

advanceDateOnly – Optional boolean flag to enforce business dates only move forward. If the business date (for the given Schedule or its dependents) has to move backwards, the request will fail. Defaults to false, if not specified.

updateDependentSchedules – Optional boolean flag to enforce business date alignment for all Batch Schedules that are dependents of the specified Schedule. It includes both inter-schedule dependencies and execution links.

comment – Mandatory field. Provides the reason to align business date and is mainly used for auditing purposes.

Response Body

The response of this endpoint, provides a clear understanding of which Batch Schedules were adjusted, what were their previous business dates and what their current business date is.

```
{
    "instances": [
        {
            "status": "<SCHEDULE_STATUS>",
            "scheduleName": "<SCHEDULE_NAME>",
            "schedulerInstanceId": "<SCHEDULER_DAY>",
            "businessDate": "<BUSINESS_DATE>",
            "previousBusinessDate": "<PREV_BIZ_DATE>",
            "activationTime": "<ACTIVATION_TIME>"
        }
    ]
}
```

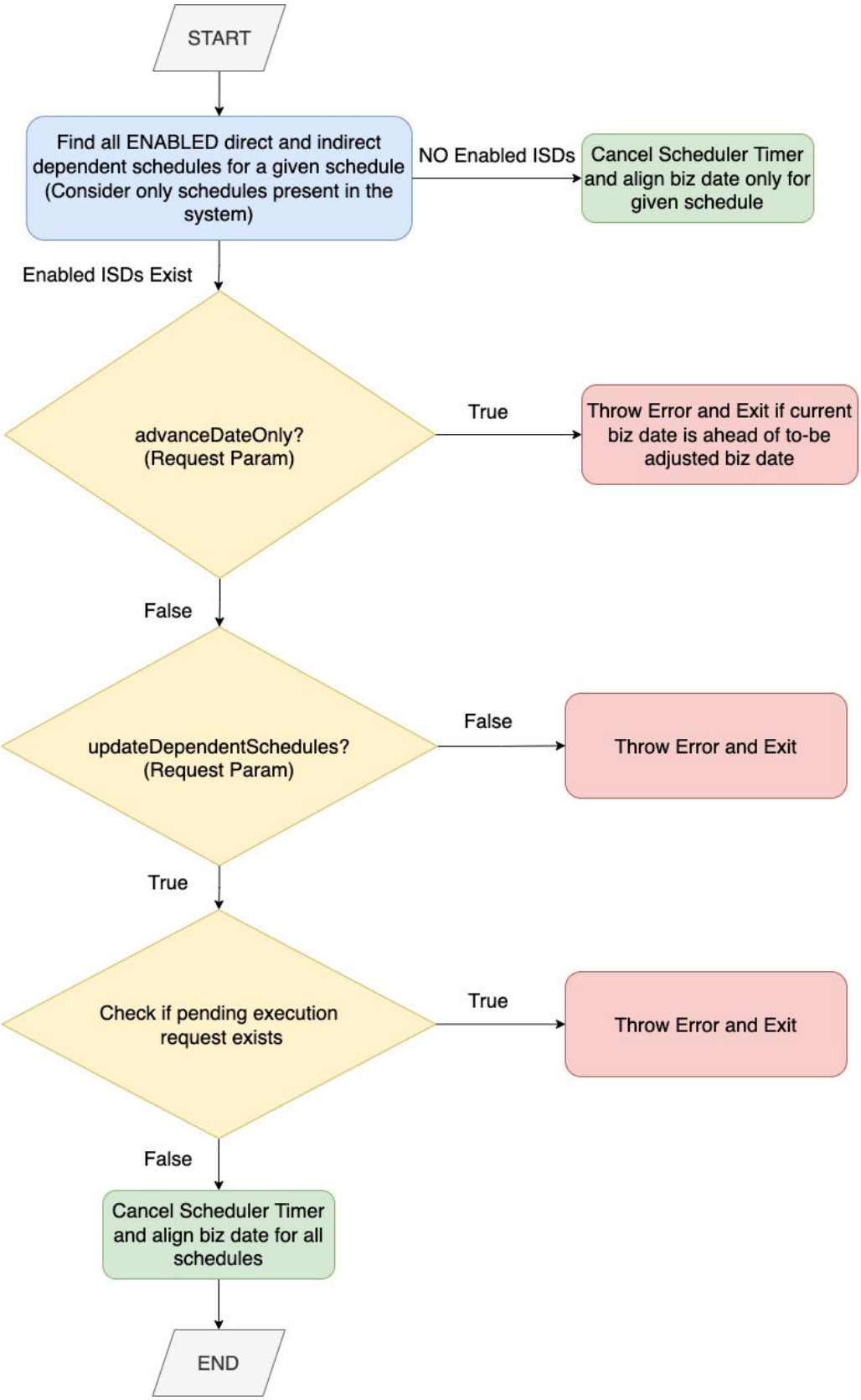
Parameter	Description
instances	Array containing Schedule objects.
scheduleName	Name of the Batch Schedule adjusted.
status	Status of the Batch Schedule.
businessDate	Current business date of the Schedule
previousBusinessDate	Previous business date of the Schedule, prior to adjustment.
activationTime	The time at which the Scheduler Day with the new business date was created.

Solution Diagram

The endpoint identifies dependent Schedules mainly by using the Inter-Schedule Dependency flag that is set on the System Configuration screen in POM, at a Schedule level and also the execution links.

Once dependent Schedules are identified, the EJB Timers used by the POM Scheduler are cancelled (if configured and in use), prior to moving the business date of the Schedule.

The general flow of this endpoint in moving the business dates of the Schedules is depicted in the flowchart below.


START
Find all ENABLED direct and indirect dependent schedules for a given schedule (Consider only schedules present in the system)
NO Enabled ISDs
Cancel Scheduler Timer and align biz date only for given schedule
Enabled ISDs Exist
advanceDateOnly? (Request Param)
True
Throw Error and Exit if current biz date is ahead of to-be adjusted biz date
False
updateDependentSchedules? (Request Param)
False
Throw Error and Exit
True
Check if pending execution request exists
True
Throw Error and Exit
False
Cancel Scheduler Timer and align biz date for all schedules
END

Custom Batch Entities API

The following endpoints are for the creation, updating, and deletion of Custom Batch Entities within POM.

Creating / Updating Custom Flows

This API provides the ability to create / update custom flows.

HTTP Method	PUT
Path	`https://<pom-server-host>/ProcessServices/services/public/schedules/<schedule-name>/customFlows` **<pom-server-host>** - This is the POM URL. **<schedule-name>** - Name of the schedule being invoked.
HTTP Headers	`Content-Type = application/json` Include Authorization header as shown in Oauth Token Generation

Request Body

Sample Request Body

```
{
  "name": "TEST_CUST1_CFLOW",
  "description": "Custom Flow Description",
  "creationComment": "Created for testing purposes",
  "processes": [
    {
      "name": "TEST_CUSTF1_CPROCESS",
      "description": "Custom Process Description",
      "isFirstProcess": true,
      "jobs": [
        {
          "name": "TEST_CUSTF1_CJOB",
          "description": "Custom Job Description",
          "type": "EXEC",
          "batchName": "dummy",
          "wrapperName": "dummy",
          "scriptDirectory": "dummy",
          "cleanupScript": "dummy",
          "application": "APP1",
          "parameters": "A B B C D E F ",
       "enabledDaysOfTheWeek":
"SUNDAY,MONDAY,TUESDAY,WEDNESDAY",
          "thresholdRunTime": "4000",
          "modifiableParameters": true,
          "skipOnError": false
        },
        {
          "name": "TEST_CUSTF21_CJOB",
          "description": "Custom Job Description",
          "type": "EXEC",
          "batchName": "dummy",
          "wrapperName": "dummy",
          "scriptDirectory": "dummy",
          "cleanupScript": "dummy",
          "application": "APP1",
          "parameters": "A B B C D E F ",
          "enabledDaysOfTheWeek":
"SUNDAY,MONDAY,TUESDAY,WEDNESDAY",
          "thresholdRunTime": "4000",
          "modifiableParameters": true,
          "skipOnError": false
        }
      ]
   },
    {
      "name": "TEST_CUSTF2_CPROCESS",
      "description": "Custom Process Description",
      "predecessors": [
        "TEST_CUSTF1_CPROCESS"
      ],
```

```
      "jobs": [
        {
          "name": "TEST_CUSTF3_CJOB",
          "description": "Custom Job Description",
          "type": "EXEC",
          "batchName": "dummy",
          "wrapperName": "dummy",
          "scriptDirectory": "dummy",
          "cleanupScript": "dummy",
          "application": "APP1",
          "parameters": "A B B C D E F ",
          "enabledDaysOfTheWeek":
"SUNDAY,MONDAY,TUESDAY,WEDNESDAY",
       "thresholdRunTime": "4000",
          "modifiableParameters": true,
          "skipOnError": false
        },
        {
          "name": "TEST_CUSTF41_CJOB",
          "description": "Custom Job Description",
          "type": "EXEC",
          "batchName": "dummy",
          "wrapperName": "dummy",
          "scriptDirectory": "dummy",
          "cleanupScript": "dummy",
          "application": "APP1",
          "parameters": "A B B C D E F ",
          "enabledDaysOfTheWeek":
"SUNDAY,MONDAY,TUESDAY,WEDNESDAY",
          "thresholdRunTime": "4000",
          "modifiableParameters": true,
          "skipOnError": false
        }
      ]
    },
    {
      "name": "TEST_CUSTF3_CPROCESS",
      "description": "Custom Process Description",
      "isLastProcess": true,
      "predecessors": [
        "TEST_CUSTF2_CPROCESS"
      ],
      "jobs": [
        {
          "name": "TEST_CUSTF4_CJOB",
          "description": "Custom Job Description",
          "type": "EXEC",
          "batchName": "dummy",
          "wrapperName": "dummy",
          "scriptDirectory": "dummy",
          "cleanupScript": "dummy",
          "application": "APP1",
```

```
            "parameters": "A B B C D E F ",
            "enabledDaysOfTheWeek":
"SUNDAY,MONDAY,TUESDAY,WEDNESDAY",
         "thresholdRunTime": "4000",
         "modifiableParameters": true,
            "skipOnError": false
          },
          {
            "name": "TEST_CUSTF51_CJOB",
            "description": "Custom Job Description",
            "type": "EXEC",
            "batchName": "dummy",
            "wrapperName": "dummy",
            "scriptDirectory": "dummy",
            "cleanupScript": "dummy",
            "application": "APP1",
            "parameters": "A B B C D E F ",
            "enabledDaysOfTheWeek":
"SUNDAY,MONDAY,TUESDAY,WEDNESDAY",
            "thresholdRunTime": "4000",
            "modifiableParameters": true,
            "skipOnError": false
          }
        ]
      }
  ]
}
```


Response Body


```
{
  "name": " TEST_CUST1_CFLOW ",
  "type": "Flow",
  "user": "POAM_ADMIN",
  "comment": "Created for testing purposes"
}
```


Deleting Custom Flow

This API provides the ability to delete the given custom flow.

HTTP Method	DELETE

Path	`https://<pom-server-host>/ProcessServices/services/public/schedules/<schedule-name>/customFlows/<flow-name>` **<pom-server-host>** - This is the POM URL. **<schedule-name>** - Name of the schedule being invoked. **<flow-name>** - Name of the flow to be deleted.
HTTP Headers	`Content-Type = application/json` Include Authorization header as shown in Oauth Token Generation

Request Body	Sample Request Body
	
	`{`
	`" comment": "Deleting custom flow TEST_CUST1_CFLOW”`
	`}`

Response Body	204 No content

Fetching All Custom Flows

This API provides the ability to fetch all custom flows.

HTTP Method	GET
Path	`https://<pom-server-host>/ProcessServices/services/public/schedules/<schedule-name>/customFlows` **<pom-server-host>** - This is the POM URL. **<schedule-name>** - Name of the schedule being invoked.
HTTP Headers	`Content-Type = application/json` Include Authorization header as shown in Oauth Token Generation
Request Body	N/A

Response Body


```
{
  "flows": [
    {
      "scheduleName": "RI",
      "displayScheduleName": "RITEST",
      "name": "TEST_CUST1_CFLOW",
      "description": "Custom Flow Description",
      "enabled": false,
      "useExisting": false,
      "customEntity": true,
      "processes": [
        {
          "name": "TEST_CUSTF3_CPROCESS",
          "description": "Custom Process Description",
          "enabled": false,
          "useExisting": false,
          "customEntity": true,
          "isFirstProcess": false,
          "isLastProcess": true,
          "predecessors": [
            "TEST_CUSTF2_CPROCESS"
          ],
          "jobs": [
            {
              "name": "TEST_CUSTF4_CJOB",
              "description": "Custom Job Description",
              "enabled": false,
              "useExisting": false,
              "customEntity": true,
              "type": "EXEC",
              "batchName": "dummy",
              "wrapperName": "dummy",
              "scriptDirectory": "dummy",
              "cleanupScript": "dummy",
              "application": "APP1",
              "parameters": "A B B C D E F ",
              "modifiableParameters": false,
              "skipOnError": false,
              "enabledDaysOfTheWeek":
"SUNDAY,MONDAY,TUESDAY,WEDNESDAY"
            },
            {
              "name": "TEST_CUSTF51_CJOB",
              "description": "Custom Job Description",
              "enabled": false,
              "useExisting": false,
              "customEntity": true,
              "type": "EXEC",
              "batchName": "dummy",
              "wrapperName": "dummy",
              "scriptDirectory": "dummy",
              "cleanupScript": "dummy",
```

```
                "application": "APP1",
                "parameters": "A B B C D E F ",
                "modifiableParameters": false,
                "skipOnError": false,
                "enabledDaysOfTheWeek":
"SUNDAY,MONDAY,TUESDAY,WEDNESDAY"
              }
            ]
          },
          {
            "name": "TEST_CUSTF1_CPROCESS",
            "description": "Custom Process Description",
            "enabled": false,
            "useExisting": false,
            "customEntity": true,
            "isFirstProcess": true,
            "isLastProcess": false,
            "jobs": [
              {
                "name": "TEST_CUSTF1_CJOB",
                "description": "Custom Job Description",
                "enabled": false,
                "useExisting": false,
                "customEntity": true,
                "type": "EXEC",
                "batchName": "dummy",
                "wrapperName": "dummy",
                "scriptDirectory": "dummy",
                "cleanupScript": "dummy",
                "application": "APP1",
                "parameters": "A B B C D E F ",
                "modifiableParameters": false,
                "skipOnError": false,
                "enabledDaysOfTheWeek":
"SUNDAY,MONDAY,TUESDAY,WEDNESDAY"
              },
              {
                "name": "TEST_CUSTF21_CJOB",
                "description": "Custom Job Description",
                "enabled": false,
                "useExisting": false,
                "customEntity": true,
                "type": "EXEC",
                "batchName": "dummy",
                "wrapperName": "dummy",
                "scriptDirectory": "dummy",
                "cleanupScript": "dummy",
                "application": "APP1",
                "parameters": "A B B C D E F ",
                "modifiableParameters": false,
                "skipOnError": false,
                "enabledDaysOfTheWeek":
```

```
"SUNDAY,MONDAY,TUESDAY,WEDNESDAY"
            }
          ]
        },
        {
          "name": "TEST_CUSTF2_CPROCESS",
          "description": "Custom Process Description",
          "enabled": false,
          "useExisting": false,
          "customEntity": true,
          "isFirstProcess": false,
          "isLastProcess": false,
          "predecessors": [
            "TEST_CUSTF1_CPROCESS"
          ],
          "jobs": [
            {
              "name": "TEST_CUSTF3_CJOB",
              "description": "Custom Job Description",
              "enabled": false,
              "useExisting": false,
              "customEntity": true,
              "type": "EXEC",
              "batchName": "dummy",
              "wrapperName": "dummy",
              "scriptDirectory": "dummy",
              "cleanupScript": "dummy",
              "application": "APP1",
              "parameters": "A B B C D E F ",
              "modifiableParameters": false,
              "skipOnError": false,
              "enabledDaysOfTheWeek":
"SUNDAY,MONDAY,TUESDAY,WEDNESDAY"
            },
            {
              "name": "TEST_CUSTF41_CJOB",
              "description": "Custom Job Description",
              "enabled": false,
              "useExisting": false,
              "customEntity": true,
              "type": "EXEC",
              "batchName": "dummy",
              "wrapperName": "dummy",
              "scriptDirectory": "dummy",
              "cleanupScript": "dummy",
              "application": "APP1",
              "parameters": "A B B C D E F ",
              "modifiableParameters": false,
              "skipOnError": false,
              "enabledDaysOfTheWeek":
"SUNDAY,MONDAY,TUESDAY,WEDNESDAY"
            }
```

```
            ]
          }
        ]
      }
    ]
}
```

Fetching a Custom Flow

This API provides the ability to fetch details of a given custom flow.

HTTP Method	GET
Path	`https://<pom-server-host>/ProcessServices/services/public/schedules/<schedule-name>/customFlows/<flow-name>` **<pom-server-host>** - This is the POM URL. **<schedule-name>** - Name of the schedule being invoked. **<flow-name>** - Name of the flow to be fetched.
HTTP Headers	`Content-Type = application/json` Include Authorization header as shown in Oauth Token Generation
Request Body	N/A

Response Body


```
{
  "scheduleName": "RI",
  "displayScheduleName": "RITEST",
  "name": "TEST_CUST1_CFLOW",
  "description": "Custom Flow Description",
  "enabled": false,
  "useExisting": false,
  "customEntity": true,
  "processes": [
    {
      "name": "TEST_CUSTF3_CPROCESS",
      "description": "Custom Process Description",
      "enabled": false,
      "useExisting": false,
      "customEntity": true,
      "isFirstProcess": false,
      "isLastProcess": true,
      "predecessors": [
        "TEST_CUSTF2_CPROCESS"
      ],
      "jobs": [
        {
          "name": "TEST_CUSTF4_CJOB",
          "description": "Custom Job Description",
          "enabled": false,
          "useExisting": false,
          "customEntity": true,
          "type": "EXEC",
          "batchName": "dummy",
          "wrapperName": "dummy",
          "scriptDirectory": "dummy",
          "cleanupScript": "dummy",
          "application": "APP1",
          "parameters": "A B B C D E F ",
          "modifiableParameters": false,
          "skipOnError": false,
          "enabledDaysOfTheWeek":
"SUNDAY,MONDAY,TUESDAY,WEDNESDAY"
        },
        {
          "name": "TEST_CUSTF51_CJOB",
          "description": "Custom Job Description",
          "enabled": false,
          "useExisting": false,
          "customEntity": true,
          "type": "EXEC",
          "batchName": "dummy",
          "wrapperName": "dummy",
          "scriptDirectory": "dummy",
          "cleanupScript": "dummy",
          "application": "APP1",
          "parameters": "A B B C D E F ",
```

```
          "modifiableParameters": false,
          "skipOnError": false,
          "enabledDaysOfTheWeek":
"SUNDAY,MONDAY,TUESDAY,WEDNESDAY"
        }
      ]
    },
    {
      "name": "TEST_CUSTF1_CPROCESS",
      "description": "Custom Process Description",
      "enabled": false,
      "useExisting": false,
      "customEntity": true,
      "isFirstProcess": true,
      "isLastProcess": false,
      "jobs": [
        {
          "name": "TEST_CUSTF1_CJOB",
          "description": "Custom Job Description",
          "enabled": false,
          "useExisting": false,
          "customEntity": true,
          "type": "EXEC",
          "batchName": "dummy",
          "wrapperName": "dummy",
          "scriptDirectory": "dummy",
          "cleanupScript": "dummy",
          "application": "APP1",
          "parameters": "A B B C D E F ",
          "modifiableParameters": false,
          "skipOnError": false,
          "enabledDaysOfTheWeek":
"SUNDAY,MONDAY,TUESDAY,WEDNESDAY"
        },
        {
          "name": "TEST_CUSTF21_CJOB",
          "description": "Custom Job Description",
          "enabled": false,
          "useExisting": false,
          "customEntity": true,
          "type": "EXEC",
          "batchName": "dummy",
          "wrapperName": "dummy",
          "scriptDirectory": "dummy",
          "cleanupScript": "dummy",
          "application": "APP1",
          "parameters": "A B B C D E F ",
          "modifiableParameters": false,
          "skipOnError": false,
          "enabledDaysOfTheWeek":
"SUNDAY,MONDAY,TUESDAY,WEDNESDAY"
        }
```

```
      ]
    },
    {
      "name": "TEST_CUSTF2_CPROCESS",
      "description": "Custom Process Description",
      "enabled": false,
      "useExisting": false,
      "customEntity": true,
      "isFirstProcess": false,
      "isLastProcess": false,
      "predecessors": [
        "TEST_CUSTF1_CPROCESS"
      ],
      "jobs": [
        {
          "name": "TEST_CUSTF3_CJOB",
          "description": "Custom Job Description",
          "enabled": false,
          "useExisting": false,
          "customEntity": true,
          "type": "EXEC",
          "batchName": "dummy",
          "wrapperName": "dummy",
          "scriptDirectory": "dummy",
          "cleanupScript": "dummy",
          "application": "APP1",
          "parameters": "A B B C D E F ",
          "modifiableParameters": false,
          "skipOnError": false,
          "enabledDaysOfTheWeek":
"SUNDAY,MONDAY,TUESDAY,WEDNESDAY"
        },
        {
          "name": "TEST_CUSTF41_CJOB",
          "description": "Custom Job Description",
          "enabled": false,
          "useExisting": false,
          "customEntity": true,
          "type": "EXEC",
          "batchName": "dummy",
          "wrapperName": "dummy",
          "scriptDirectory": "dummy",
          "cleanupScript": "dummy",
          "application": "APP1",
          "parameters": "A B B C D E F ",
          "modifiableParameters": false,
          "skipOnError": false,
          "enabledDaysOfTheWeek":
"SUNDAY,MONDAY,TUESDAY,WEDNESDAY"
        }
      ]
    }
```

```
  ]
}
```

Creating / Updating Custom Processes

This API provides the ability to create / update custom processes.

HTTP Method	PUT
Path	`https://<pom-server-host>/ProcessServices/services/public/schedules/<schedule-name>/customProcesses` **<pom-server-host>** - This is the POM URL. **<schedule-name>** - Name of the schedule being invoked.
HTTP Headers	`Content-Type = application/json` Include Authorization header as shown in Oauth Token Generation

Request Body

Sample Request Body


```
{
  "name": "TEST_CUST1_CPROCESS",
  "description": "Custom Process Description",
  "creationComment": "Created for testing purposes",
  "jobs": [
    {
      "name": "TEST_CUST5_CJOB",
      "description": "Custom Job Description",
      "type": "EXEC",
      "batchName": "dummy",
      "wrapperName": "dummy",
      "scriptDirectory": "dummy",
      "cleanupScript": "dummy",
      "application": "APP1",
      "parameters": "A B B C D E F ",
      "enabledDaysOfTheWeek": "SUNDAY,MONDAY,TUESDAY,WEDNESDAY",
      "thresholdRunTime": "4000",
      "modifiableParameters": true,
      "skipOnError": false,
      "customEntity": true
    },
    {
      "name": "TEST_CUST61_CJOB",
      "description": "Custom Job Description",
      "type": "EXEC",
      "batchName": "dummy",
      "wrapperName": "dummy",
      "scriptDirectory": "dummy",
      "cleanupScript": "dummy",
      "application": "APP1",
      "parameters": "A B B C D E F ",
      "enabledDaysOfTheWeek": "SUNDAY,MONDAY,TUESDAY,WEDNESDAY",
      "thresholdRunTime": "4000",
      "modifiableParameters": true,
      "skipOnError": false,
      "customEntity": true
    },
    {
      "name": "TEST_CUST71_CJOB",
      "description": "Custom Job Description",
      "type": "EXEC",
      "batchName": "dummy",
      "wrapperName": "dummy",
      "scriptDirectory": "dummy",
      "cleanupScript": "dummy",
      "application": "APP1",
      "parameters": "A B B C D E F ",
      "enabledDaysOfTheWeek": "SUNDAY,MONDAY,TUESDAY,WEDNESDAY",
      "thresholdRunTime": "4000",
      "modifiableParameters": true,
```

```
      "skipOnError": false,
      "customEntity": true
    }
  ]
}
```

Response Body

```
{
  "name": "TEST_CUST1_CPROCESS",
  "type": "Process",
  "user": "POAM_ADMIN",
  "comment": "Created for testing purposes"
}
```

Deleting Custom Process

This API provides the ability to delete the given custom process.

HTTP Method	DELETE
Path	`https://<pom-server-host>/ProcessServices/services/public/schedules/<schedule-name>/customProcesses/<process-name>` **<pom-server-host>** - This is the POM URL. **<schedule-name>** - Name of the schedule being invoked. **<process-name>** - Name of the process to be deleted.
HTTP Headers	`Content-Type = application/json` Include Authorization header as shown in Oauth Token Generation

Request Body	Sample Request Body
	
	`{`
	`" comment": "Delete custom process TEST_CUST1_CPROCESS”`
	`}`

Response Body	204 No content

Fetching all Custom Processes

This API provides the ability to fetch all custom processes.

HTTP Method	GET

Path	`https://<pom-server-host>/ProcessServices/services/public/schedules/<schedule-name>/customProcesses` **<pom-server-host>** - This is the POM URL. **<schedule-name>** - Name of the schedule being invoked.
HTTP Headers	`Content-Type = application/json` Include Authorization header as shown in Oauth Token Generation
Request Body	N/A

Response Body


```
{
  "processes": [
    {
      "scheduleName": "RI",
      "displayScheduleName": "RITEST",
      "name": "TEST_CUST1_CPROCESS",
      "description": "Custom Process Description",
      "enabled": false,
      "useExisting": false,
      "customEntity": true,
      "isFirstProcess": false,
      "isLastProcess": false,
      "jobs": [
        {
          "name": "TEST_CUST5_CJOB",
          "description": "Custom Job Description",
          "enabled": false,
          "useExisting": false,
          "customEntity": true,
          "type": "EXEC",
          "batchName": "dummy",
          "wrapperName": "dummy",
          "scriptDirectory": "dummy",
          "cleanupScript": "dummy",
          "application": "APP1",
          "parameters": "A B B C D E F ",
          "modifiableParameters": false,
          "skipOnError": false,
          "enabledDaysOfTheWeek":
"SUNDAY,MONDAY,TUESDAY,WEDNESDAY"
        },
        {
          "name": "TEST_CUST61_CJOB",
          "description": "Custom Job Description",
          "enabled": false,
          "useExisting": false,
          "customEntity": true,
          "type": "EXEC",
          "batchName": "dummy",
          "wrapperName": "dummy",
          "scriptDirectory": "dummy",
          "cleanupScript": "dummy",
          "application": "APP1",
          "parameters": "A B B C D E F ",
          "modifiableParameters": false,
          "skipOnError": false,
          "enabledDaysOfTheWeek":
"SUNDAY,MONDAY,TUESDAY,WEDNESDAY"
        },
        {
          "name": "TEST_CUST71_CJOB",
          "description": "Custom Job Description",
```

```
            "enabled": false,
            "useExisting": false,
            "customEntity": true,
            "type": "EXEC",
            "batchName": "dummy",
            "wrapperName": "dummy",
            "scriptDirectory": "dummy",
            "cleanupScript": "dummy",
            "application": "APP1",
            "parameters": "A B B C D E F ",
            "modifiableParameters": false,
            "skipOnError": false,
            "enabledDaysOfTheWeek":
 "SUNDAY,MONDAY,TUESDAY,WEDNESDAY"
          }
        ]
      }
    ]
}
```

Fetching a Custom Process

This API provides the ability to fetch details of a given custom process.

HTTP Method	GET
Path	`https://<pom-server-host>/ProcessServices/services/public/schedules/<schedule-name>/customProcesses/<process-name>` **<pom-server-host>** - This is the POM URL. **<schedule-name>** - Name of the schedule being invoked. **<process-name>** - Name of the process to be fetched.
HTTP Headers	`Content-Type = application/json` Include Authorization header as shown in Oauth Token Generation
Request Body	N/A

Response Body


```
{
    "scheduleName": "RI",
    "displayScheduleName": "RITEST",
    "name": "TEST_CUST1_CPROCESS",
    "description": "Custom Process Description",
    "enabled": false,
    "useExisting": false,
    "customEntity": true,
    "isFirstProcess": false,
    "isLastProcess": false,
    "jobs": [
        {
            "name": "TEST_CUST5_CJOB",
            "description": "Custom Job Description",
            "enabled": false,
            "useExisting": false,
            "customEntity": true,
            "type": "EXEC",
            "batchName": "dummy",
            "wrapperName": "dummy",
            "scriptDirectory": "dummy",
            "cleanupScript": "dummy",
            "application": "APP1",
            "parameters": "A B B C D E F ",
            "modifiableParameters": false,
            "skipOnError": false,
            "enabledDaysOfTheWeek":
"SUNDAY,MONDAY,TUESDAY,WEDNESDAY"
        },
        {
            "name": "TEST_CUST61_CJOB",
            "description": "Custom Job Description",
            "enabled": false,
            "useExisting": false,
            "customEntity": true,
            "type": "EXEC",
            "batchName": "dummy",
            "wrapperName": "dummy",
            "scriptDirectory": "dummy",
            "cleanupScript": "dummy",
            "application": "APP1",
            "parameters": "A B B C D E F ",
            "modifiableParameters": false,
            "skipOnError": false,
            "enabledDaysOfTheWeek":
"SUNDAY,MONDAY,TUESDAY,WEDNESDAY"
        },
        {
            "name": "TEST_CUST71_CJOB",
            "description": "Custom Job Description",
            "enabled": false,
            "useExisting": false,
```

```
                "customEntity": true,
                "type": "EXEC",
                "batchName": "dummy",
                "wrapperName": "dummy",
                "scriptDirectory": "dummy",
                "cleanupScript": "dummy",
                "application": "APP1",
                "parameters": "A B B C D E F ",
                "modifiableParameters": false,
                "skipOnError": false,
                "enabledDaysOfTheWeek":
"SUNDAY,MONDAY,TUESDAY,WEDNESDAY"
            }
        ]
}
```

A

Batch Schedule Spreadsheet Template

Overview

This appendix explains how to set up the Batch Schedule Spreadsheet template. The template was originally developed for internal Oracle use and has since been made available to System Integrators and Customers for customizing Retail Application provided Batch Schedules or for developing custom Schedules. For the latter use, custom Batch Schedules may be less complex and may not require some tabs and fields.

Structure

The purpose of this template is to design the batch metadata that can be seeded into a Batch Schedule within POM. A Batch Schedule, at minimum, must define a Nightly Flow, consisting of a start and end Job. The Recurring and Adhoc Cycles are optional and need not contain any Processes or Jobs in them.

The spreadsheet template has a default nightly flow pre-populated with start and end Nightly jobs. These jobs are seeded with `DUMMY` job type so that they will be skipped during execution.

Tabs

The following tabs are available on the Spreadsheet Template.

Tab Name	Purpose
Schedule	Defines the name, display schedule name and version of the Batch Schedule.
Flow	Defines all the Flows in POM. Since the Nightly Flow is already in-built within POM, this refers to the Flows on the Adhoc and Hourly Cycles.
Process	Defines all the Processes in POM, across all Batch Cycles.
Job	Defines all the Jobs in POM, across all Batch Cycles.
FlowProcessMapping	Creates mappings between Flows and Processes.
ProcessJobMapping	Creates mappings between Processes and Jobs and maintains the sequence order of jobs within a process and also the parameter value.
Dependency	Creates internal dependencies between Processes.
Application	Defines the Applications to which the Jobs belong. It also defines a mapping between an Application and the name of the Job Agent that must run those Jobs.
Modules	Defines Modules within Applications. These are mainly used to synchronize with the Modules defined for the Application in Retail Home. See Sync with MDF in the POM User Guide for further details.

Tab Name	Purpose
SystemOption	Lists all the system options available to the Jobs at execution time.
ThrottlingConfiguration	Defines the throttle settings at an Application level.
InterscheduleDependency	Defines dependencies with a different Batch Schedule on the same POM instance
ExternalDependencies	Defines External Dependencies usually on a customer's internal process
BatchLinks	Also known as Schedule Links. If the Nightly Flow is going to be invoked via a different Job on a different Schedule, then a Schedule Link can be defined.
JobTypes	Custom ReST based types of Jobs that need to be run
Obsolete	Defines Batch entities that are now obsolete and must be removed.
Schedule Errors	Empty tab that is populated with the list of errors, either by the macro or when being transformed by the Batch Schedule Transformer.

Macro Validation

This Excel Template is equipped with a macro that can be run to validate the entries on the spreadsheet. Upon encountering failures, the macro publishes the error entries in the Schedule Errors tab.

Schedule Validation Processes

Applications can set up processes and jobs for use by the provisioning process to validate the setup. After POM has been provisioned/patched, the provisioning tool will attempt to run the validation job(s) to ensure that POM is functional end to end. Following are the steps to setup these processes/jobs:

1. In the Process tab, add new processes that end with `_POM_VPROCESS`. Examples: `RMS_SALES_POM_VPROCESS`, `REIM_EJB_POM_VPROCESS`
2. In the Process tab, set the value in the **AdhocInd** field to `Y` because validation processes can only be part of the adhoc cycle.
3. In the **ProcessJobMapping** tab, make sure the newly created validation processes are mapped to any valid job that shouldn't run for more than 5 minutes. Validation jobs should obviously be benign in that they should not alter data.
4. If there is more than one job in a validation process, make sure the pre and post dependencies are correctly defined in the **Dependency** tab.

Schedule Upgrade Directives

These are directives solely to the Batch Schedule upgrade process, altering values as specified, when the default settings are not acceptable. For instance, all newly added Jobs are marked DISABLED by default. To enable a Job, the AMS team or the Customer is asked to do so through the POM UI. This is not ideal as it involves manual intervention. To overcome this, we use the "enableOnNew" directive on the Job tab, to ensure that the Job is marked ENABLED when its added for the first time.

The processing of these directives is designed as a stage in the Schedule Import process. The following directives are currently supported in POM.

Template Tab	Directive Name	Description
Job	enableOnNew	Only when being loaded for the very first time, the Job will be enabled. When this directive is not used, the job will be disabled by default when added for the first time.
Job	enableLikeJobOnNew	Only when being loaded for the very first time, the status of the Job (for which the directive is specified) will match that of the “LikeJob”.
Job	enableLikePJOnNew	Only when being loaded for the very first time, the status of the Job (for which the directive is specified) will match that of the “LikeProcessJob” defined.

If any Process-Job combination has multiple directives that are semicolon delimited, they get executed in the following order:

1. enableOnNew
2. enableLikePJOnNew
3. enableLikeJobOnNew

Tab Definitions

Process Tab

This tab allows you to define all processes for the schedule at hand. A process is a group of jobs that are to be run sequentially from start to end. The following is a description of all fields on this tab:

- **ProcessName**
  - Uniquely identifies a process. The Process Name should be in upper case only with no spaces. Use an underscore if needed. It should end with `XXX_PROCESS`. Validation processes should end with `_POM_VPROCESS`.

    See Schedule Validation Processes above for an explanation of the validation processes.
  - The process name is the batch job name appended with `_PROCESS`.

    For example: `DEALUPLD_PROCESS`

    `SA_TRANSACTION_LOADING_PROCESS`
  - For non-Merchandising processes, the process name should start with the Application name.

    For example: `REIM_POSTING_PROCESS, ALC_DAILY_CLEANUP_ PROCESS`
  - If a process is part of the nightly batch cycle but is also an ad hoc job, or if it’s also part of a recurring flow, three separate processes should be defined. The ad hoc process should end with `XXX_PROCESS_ADHOC` and the recurring process should end with `XXX_CYCLE_PROCESS`.

For example:

* SA_TRANSACTION_LOADING_PROCESS – for the sales audit transaction loading jobs that will run in the nightly batch cycle.
* SA_TRANSACTION_LOADING_PROCESS_ADHOC – for the sales audit transaction loading jobs that will run ad hoc.
* SA_TRANSACTION_LOADING_CYCLE_PROCESS – for the sales audit transaction loading jobs that will run multiple times a day.

- **Description**
  - Short description of the process. There should be no special characters. If the process contains just one job, the description can be the same as the batch job name.
- **DependencyType**
  - Valid values are TIME, JOB and BOTH.
  - If the first job in this process is triggered on completion of another job, mark as JOB.
  - If the process is scheduled to run at a specific time (for example, the first job of the nightly batch cycle, or part of a recurring flow process), mark as TIME.
  - If the process needs to run at a specific time and at the same time is dependent on another process, mark as BOTH.
  - If the process is ad hoc and is not part of the nightly batch cycle, leave as NULL.
- **ApplicationName**
  - This holds the application name which the batch process belongs to.
  - Valid values are RMS, RPM, REIM, RESA, ALLOC, RDE, MFP, AIPFSL, RDF, and so on.
- **AdhocInd**
  - Valid values are Y and N.
  - If the job(s) in the process are ad hoc jobs (can be run any time and not part of the nightly batch cycle), mark as Y. Otherwise, mark as N.
  - A process should not have a mix of ad hoc and scheduled jobs in it.

Job Tab

The Job tab contains all the individual jobs to be executed as part of the schedule at hand.

- **JobName**
  - Uniquely identifies a job. The Job Name should be in upper case only with no spaces. Use an underscore if needed. It should end with XXX_JOB. If the same job is part of multiple processes, define separate jobs for it.

    Examples:

    * EXPORT_DIFFS_JOB (under EXPORT_DIFFS_PROCESS) – for the export_diffs.ksh program passing in 'delta' as a parameter

    - EXPORT_DIFFS_FULL_JOB (under EXPORT_DIFFS_FULL_PROCESS) – for the same export_diffs.ksh program but this time passing in 'full'
  - For non-Merchandising jobs, the job name should start with the Application, (for example, REIM_POSTING_JOB, ALC_DAILY_CLEANUP_JOB, RDE_SEASNSDE_JOB)
  - If the job is part of the nightly batch cycle and is also part of a recurring flow, define two separate jobs for it.

    Examples:

    - UPLOADSALES_JOB belonging to SALESPROCESS_PROCESS – for the uploadsales.ksh program run as part of the nightly batch cycle
    - UPLOADSALES_CYCLE_JOB belonging to SALESPROCESS_CYCLE_PROCESS – for the same uploadsales.ksh program run as part of a recurring flow

- **Description**
  - Short description of the batch program. There should be no special characters.
- **RmsBatch**
  - This holds the exact batch executable that the job refers to. For Pro*C programs, this should be the binary without the .pc extension. For KSH scripts, include the .ksh extension. This field is case-sensitive.
  - For service type jobs (non-shell script EXEC type jobs), this field should be left blank.
- **RmsWrapper**
  - If applicable, this holds the exact batch wrapper name used to call the batch program. This field is case-sensitive.
  - For service type jobs (non shell script EXEC type jobs), this field should be left blank.
- **ScriptFolder**
  - This holds the directory path where the Wrapper file resides in the system.
  - For service type jobs (non-shell script EXEC type jobs), this field should be left blank.
- **ParameterValue**
  - This holds the entire parameter value to be passed in to a shell script type job. This field is case sensitive.
  - For parameters that can have multiple values (for example, purge days parameter), provide a default value so that the batch can still be executed.
  - Placeholder parameter #JobCtxt.businessDate can be defined on the Job sheet for jobs that require a POM business date parameter (DDMMYYYY format).
- Parameters are updated in the BASE_PARAMETER column of the BATCH_PROCESS_JOB_LIST table. On a schedule upgrade, the ACTIVE_PARAMETER column value is updated with the BASE_PARAMETER only if the parameter value has never been changed from the UI.
- **ApplicationName**
  - This holds the application that the batch program belongs to.
  - It should be a valid application mentioned in the Application tab.
- **Module**

  - This holds the name of the module under the application which the batch program belongs to. The Application / Module entered here should be a valid combination in the Application tab.
  - If there are no modules defined for the corresponding application in the Application tab or the Modules tab, then this column can be left blank. Otherwise it is required to enter the module(s) it belongs to.
  - A job can belong to multiple modules under the same application. In this case the list of those modules should be entered here with `||` as a delimiter.

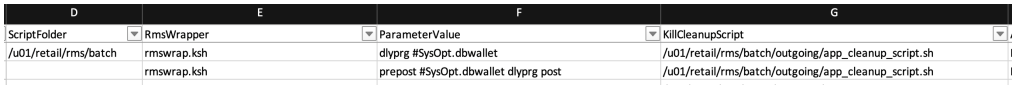
    For example:

    `MODULE1||MODULE2||MODULE3`
  - If the corresponding Application have some modules defined in Application Sheet then this column cannot be left blank.

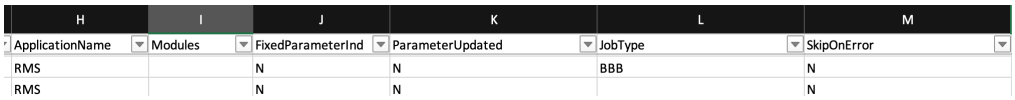
- **FixedParameterInd – VARCHAR(1)**
  - Valid values are `Y` and `N`.
  - This indicates whether the parameter value can be changed on the Batch Administration screen. If this is `Y`, the parameter will be changeable on the UI. This is applicable if the parameter cannot have a default value, or if the parameter can have different values.
  - If this is `N`, the parameter value is fixed and will not be changeable on the UI.
- **ParameterUpdated – VARCHAR(1)**
  - Valid values are `Y` and `N`.
  - This indicates whether the parameter value column has changed from the previous value.
  - If `Y`, then the parameter value is passed and updates the existing job information with the newly passed parameter value. If `N` the parameter value won't be updated for the existing job information.
- **SkipOnError – VARCHAR(1)**
  - Valid values are `Y` and `N`.
  - A value of `Y` indicates that the job should be skipped if an error occurs and the batch schedule continues to run. Otherwise the batch schedule is stopped.
- **JobType – VARCHAR (50)**
  - The default value set is `EXEC`, which represents shell-script based Jobs. Other pre-defined service based job types are `RI`, `RASE`, `BDI`, `RPAS`, `OMS`, `OB`, `RDS`, and `DUMMY`.
  - Custom Job types defined in the JobType tab can also be used in this field. In this case POM is directed to execute the job using the endpoint defined on the JobType tab.
- **KillCleanupScript – VARCHAR (1000)**
  - Absolute path of the cleanup script to be run after killing a shell script (`EXEC`) type job from the POM UI. Example: `/u01/retail/rms/batch/outgoing/app_cleanup_script.sh`
  - Can include arguments as well along with the script.

The post-kill cleanup script is executed automatically by POM after a Kill is performed from the UI. It does not perform the kill itself. It is defined by the application teams such as Merch. Here's how it works: The user performs a Kill on a running job from the POM Batch Monitoring screen. If a `KillCleanupScript` is defined for that job on the Job worksheet of the batch schedule spreadsheet, POM will execute it after killing the job on the server. Application teams usually include database clean-up as part of the clean-up script.

For example:



D	E	F	G
ScriptFolder	RmsWrapper	ParameterValue	KillCleanupScript
/u01/retail/rms/batch	rmswrap.ksh	dlyprg #SysOpt.dbwallet	/u01/retail/rms/batch/outgoing/app_cleanup_script.sh
	rmswrap.ksh	prepost #SysOpt.dbwallet dlyprg post	/u01/retail/rms/batch/outgoing/app_cleanup_script.sh



H	I	J	K	L	M
ApplicationName	Modules	FixedParameterInd	ParameterUpdated	JobType	SkipOnError
RMS		N	N	BBB	N
RMS		N	N		N

ProcessJob Mapping

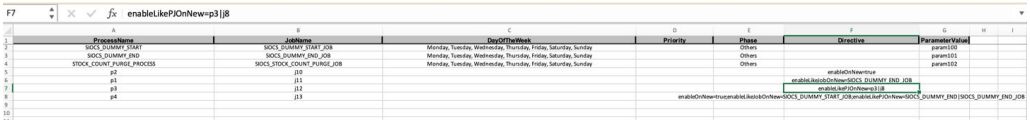
The ProcessJobMapping tab allows you to map or group jobs within processes.

- **ProcessName – VARCHAR(50)**
  - This should match the **ProcessName** in the Process tab.
- **Job Name – VARCHAR(50)**
  - This should match the **JobName** in the Job tab.
- **Job Sequence – NUMBER(4)**
  - Contains the sequence of execution for the Job in a Process starting from `1` for the first job in a Process. This number must be unique and contiguous for a process.
  - Valid values are `1` to `9999`.
- **DayOfTheWeek – VARCHAR(100)**
  - Contains the day(s) of the week on which the specific Process/Job needs to run. POM automatically skips these Jobs on the remaining days.
  - This field is optional, leaving it blank will cause the Job to run on a daily basis.
- **Priority – NUMBER (2)**
  - Contains the Priority set for the specific Process / Job.
  - A simple number representing the priority at which a job will run in a throttled setting compared to other jobs that are set to run concurrently
  - Valid values: 1 to 10 with 10 being the highest priority and 1 being the lowest
  - This field is optional, leaving it blank will set the priority to the default value of zero.
- **Phase – VARCHAR (50)**
  - Contains the Phase name designated for the Process / Job.
  - This grouping of jobs in the Nightly cycle is used for reporting purposes only. Nightly Summary Report will then report total run time and elapsed time by Phase.

  - This field is optional, leaving it blank will default the phase to Others.
- **Directive – VARCHAR (200)**
  - Contains the directive for Process / Job.
  - Enables/disables a new Job according to the directive provided.
  - Valid values: enableOnNew, enableLikePJOnNew, enableLikeJobOnNew. See the Schedule Upgrade Directives section above for more details on the use of these values.
  - This is an optional column. Leaving it blank will create the job as disabled.
  - The value of this column is a semi-colon separated string of directive key-value pairs. A process job can have one or more than one directive defined.
- **ParameterValue–VARCHAR(2000)**
  - Parameter value for the process–job combination.
  - Value will be updated during schedule patching only if the value has never been updated using the UI.
  - This field should be part of either the Job sheet or the ProcessJobMapping sheet.

Example

If a process has 2 jobs in it, then there should be 2 entries in this tab for each job.



Dependency Tab

The Dependency tab allows you to define the dependencies between processes.

- **ProcessName – VARCHAR(50)**
  - This should match the **ProcessName** in the ProcessJobMapping tab.
- **PredecessorProcessName – VARCHAR(50)**
  - This holds the process name that should run before the current job. This can be the same process (if there are multiple jobs in the process) or a different process than the current one.

Example

ProcessName	PredecessorProcessName
SAPURGE_PROCESS	SAEXPRMS_PROCESS
SAPURGE_PROCESS_START	SAPURGE_PROCESS
SAPURGE_PROCESS_END	SAPURGE_PROCESS_START

> **Note:**
>
> Dependencies between adhoc processes of named adhoc flow / invokable flow can also be defined in this tab.
> If an invokable flow only contains one adhoc process, there's no need to fill up the Dependency tab for such flow.
>
> If an invokable flow has more than one adhoc process, the dependencies of processes 2, 3, and so on need to be defined. There is no need to define the dependency of the first process, because this will not be dependent on another process.

Example

The invokable / named adhoc flow `FCUSTOMERUPLOAD_ADHOCFLOW` has 2 processes:

- `FCUSTOMERUPLOAD_PROCESS_ADHOC`
- `FCUSTOMERPURGE_PROCESS_ADHOC`

Only the dependency of the second process (`FCUSTOMERPURGE_PROCESS_ADHOC`) needs to be defined:

ProcessName	PredecessorProcessName
FCUSTOMERPURGE_PROCESS_ADHOC	FCUSTOMERUPLOAD_PROCESS_ADHOC

Flow Tab

The Flow tab allows you to define jobs that runs Adhoc flows and hourly or multiple times a day.

- **RecurringFlowName – VARCHAR(50)**
  - Uniquely identifies the recurring flow / adhoc flow. The Recurring Flow Name should be in uppercase only with no spaces. Use an underscore if needed. It should end with `XXX_CYCLE`.
- **Description – VARCHAR(50)**
  - Short description of the recurring / adhoc flow. There should be no special characters.
- **NumberOfRuns – NUMBER(2)**
  - This contains the number of runs for this recurring flow in a day.
  - Valid values are `1-24`. For hourly runs, maximum value is `12`. For half hourly runs, the maximum value is `24`.
- **Interval – NUMBER(2)**
  - The interval between job runs. The Interval is specified in hours. Upper Limit is `12`.
- **StartTime – VARCHAR(50)**
  - The time for the first batch run.
- **AdhocInd – VARCHAR (1)**

  - Indicates if flow belongs to adhoc flow or not.
  - Valid values: Y/N

Example

AdhocFlowTest1 and AdhocFlowTest2 are the adhoc flows with AdhocInd set to Y. HourlCycle is the hourly cycle defined to run 4 times at an interval of 4.

A	B	C	D	E	F
FlowName	Description	NumberOfRuns	Interval	StartTime	AdhocInd
HourlCycle	Hourly Test	4	4		
AdhocFlowTest1	Adhoc Flow Testing				Y
AdhocFlowTest2					Y

FlowProcessMapping Tab

The FlowProcessMapping tab allows you to define the processes for each recurring / adhoc flow defined in the Flow tab.

- **RecurringFlowName – VARCHAR(50)**
  - This should match the **RecurringFlowName** in the Flow tab.
- **ProcessName – VARCHAR(50)**
  - This holds the process name(s) that are part of the recurring/ adhoc flow.

    This should match the **ProcessName** in the Process tab. A recurring / adhoc flow can have more than 1 process.
- **FirstProcessInd – VARCHAR(1)**
  - Valid values are `Y` and `N`.
  - If the process holds the first job of the recurring / adhoc flow, mark as `Y`. Otherwise, mark as `N`.
- **LastProcessInd – VARCHAR(1)**
  - Valid values are `Y` and `N`.
  - If the process holds the last job of the recurring/ adhoc flow, mark as `Y`. Otherwise, mark as `N`.

Example

The `SALESPROCESS_CYCLE` recurring flow has only 1 process, so the first and last jobs of the recurring flow is in the same process.

The `REPLENISHMENT_CYCLE` recurring flow has multiple processes. The first job of the recurring flow is in the first process, and the last job is in the last process.

RecurringFlowName	ProcessName	FirstProcessInd	LastProcessInd
SALESPROCESS_CYCLE	SALESPROCESS_CYCLE_PROCESS	Y	Y
REPLENISHMENT_CYCLE	REPLENISHMENT_CYCLE_PROCESS	Y	N

RecurringFlowName	ProcessName	FirstProcessInd	LastProcessInd
REPLENISHMENT_CYCLE	SUPSPLIT_CNTRPRSS_CYCLE_PROCESS	N	N
REPLENISHMENT_CYCLE	INVESTMENT_BUY_CYCLE_PROCESS	N	N
REPLENISHMENT_CYCLE	RPLBLD_CYCLE_PROCESS	N	N
REPLENISHMENT_CYCLE	REPLENISHMENT_END_CYCLE_PROCESS	N	Y
AdhocFlowTest1	APIPGHL_ADHOC_PROCESS	Y	Y
AdhocFlowTest2	RDF_ADHOC_PROCESS	Y	Y

Obsolete Tab

The Obsolete tab allows you to keep track of any process or job that has been deleted from an active environment. Enter the Name and Type of deleted items on this tab.

- **Name – VARCHAR**
  - Holds the name (for example, Job Name, Process Name, Recurring Flow Name) of the item of type (Job, Process, Flow) being removed from the schedule.
  - For ProcessJobMapping, Dependency / Inter Schedule Dependency/ External Dependency type, it holds the pattern.
  - For ProcessJobMapping type, the pattern should be `process#job`
  - For dependency type, the pattern should be `preJob#preProcess#job#process`
  - For InterScheduleDependency type, the pattern should be `job#process#externalScheduleName#externalJob#externalProcess`
  - For ExternalDependency type, the pattern should be `job#process#externalJob`
  - For BatchLink (Execution Links) type, the pattern should be `job#process#externalScheduleName#externalJob#externalProcess`
- **Type – VARCHAR**
  - Valid values are `Process`, `Job`, `ProcessJobMapping`, `Flow`, `Dependency`, `InterScheduleDependency`, `ExternalDependency`, `BatchLink`, `Application` and `Module`.

Example

Name	Type
RMSE_MFP_INVENTORY_JOB	Job
RMSE_RDF_DAILY_SALES_PROCESS	Process
DELETE_TAB_STATS_PROCESS#DELETE_TAB_STATS_JOB	ProcessJobMapping
SAIMPTLOGI_POST_JOB#SA_TRANSACTION_LOADING_PROCESS_ADHOC#SAVOUCH_JOB#SA_TRANSACTION_LOADING_PROCESS_ADHOC	Dependency

Name	Type
DELETE_TAB_STATS_JOB#DELETE_TAB_STATS_PROCESS#RDE#RDE_RTLRDEZIP_JOB#RDE_RTLRDEZIP_PROCESS	InterSchedule Dependency
ORBATCH_VERIFY_RCI_JOB#FILE_VALIDATION_PROCESS#EXTERNAL_JOB_NAME	ExternalDependency
DELETE_TAB_STATS_JOB#DELETE_TAB_STATS_PROCESS#RDE#RDE_RTLRDEZIP_JOB#RDE_RTLRDEZIP_PROCESS	BatchLink
RCI	Application
RCICUSTOMER	Module
BDI_PRICING_PC_TX_CYCLE	Flow

SystemOption Tab

The SystemOption tab contains the system level options used to control certain aspects of the POM application, such as whether to enable throttling. It can also be used to define shell script error or warning exit codes. If any such codes are specified, they will cause a shell script based job to fail or complete with warning.

This tab can be left blank for service (non-EXEC) type jobs.

- **Name – VARCHAR(255)**
  - Holds the name of the System Option value needed in JOS; for example, `enableThrottling`, `WarningCode110`, `ErrorCode40`, `RIRestartableErrorMessages`, `TARGET_BATCH_DURATION`.

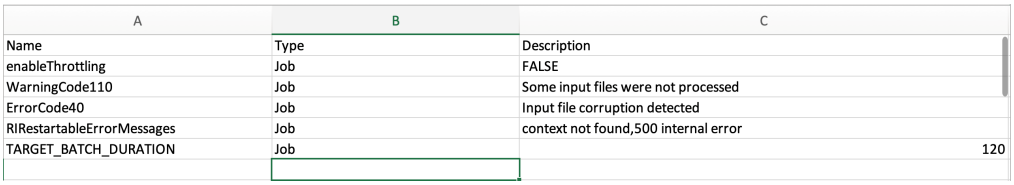
    A value of `WarningCode110`, for example, causes POM to mark the job status as completed with a warning if the job's shell script exits with a code of 100. A value of `ErrorCode40`, for example, causes POM to mark the job status as error if the job's shell script exits with a code of 40.

    For a value of `<<JOB_TYPE>>RestartableErrorMessages`, the description field holds a list of comma-separated error messages that determine the auto-restart capability of the jobs. If the response contains one of those error messages, POM will retry the job for a maximum of 3 attempts.

    For example, to add this capability to the RI Job Type when encountering either of the messages "context not found" or "500 internal error".
- **Type – VARCHAR**
  - Valid values are `Process` and `Job` (for example, `Job` for system option `enableThrottling` or error/warning shell script exit codes or Restartable Error Messages).
  - This defines the JOS admin component the System Option is for. `Job` is for `JosJobAdmin` and `Process` is for `BdiProcessFlowAdmin`.
- **Description – VARCHAR(255)**
  - This holds the value for the System Option. For example, `TRUE` for system option `enableThrottling`, or the error/warning message to be conveyed in the notification sent in the case of an error/warning shell script exit codes, or the error messages to determine whether to auto-restart rest-based jobs or not, or

the number of minutes for the `TARGET__BATCH_DURATION`. The system options value changes are updated during schedule patching.

Here's an example **SystemOption** tab with all the aforementioned system options:



A	B	C
Name	Type	Description
enableThrottling	Job	FALSE
WarningCode110	Job	Some input files were not processed
ErrorCode40	Job	Input file corruption detected
RIRestartableErrorMessages	Job	context not found,500 internal error
TARGET_BATCH_DURATION	Job	120

Application Tab

The Application tab allows you to define the Application name and its modules.

- **Name – VARCHAR(50)**
  - Holds the name of the application supported in the schedule (for example, `RMS`, `RI`, `RDS`).
  - This cannot be blank.
- **Description – VARCHAR(1000)**
  - Description of the application.
- **Modules – VARCHAR(50)**
  - List of module name(s) associated with the Application.
  - `||` is used as delimiter (for example, `MODULE1||MODULE2||MODULE3`)
  - This field can be blank when there are no modules defined for the Application.
  - If the module list is large (> 3000 chars), rather than filling modules in this field, list all modules in **Modules** tab. If larger than 3000 chars, the schedule generation will fail. If you decide to use the **Modules** tab, the modules in this field in the **Application** tab need to be blank, as it directs POM to look for modules in the **Modules** tab instead.
- **JosJobAdminName – VARCHAR(50)**
  - Corresponds to the Job Admin name (in POM version 19.x) or Agent name (in POM version 22.x or later) for the application. This usually is the same or similar name to the application name.
  - This cannot be blank.

Example

- Application with modules defined.

Name	Description	Modules	JosJobAdminName
COMMON	All apps required programs	COMMONMAINTAIN||PRE_VERIFY||CALENDAR||COMMONPRODUCT||COMMONORG||COMMONPROMO||EMPLOYEE||COMMONCO||COMMONSALES||REASON||COMMONAC||COMMONCS||COMMONASO||COMMONCDT||COMMONDT	RIS1
ORASERCI	ORASE and RCI required programs	ORASERCICUSTOMER||CONSUMER||CATMAN||ORASERCIMARKET||TRADEAREA||CUSTSEG||MARKETAGG||ORASERCIAC||ORASERCICS||ORASERCIASO||ORASERCICDT||ORASERCIDT	RIS2
RCI	RCI required programs	RCICUSTOMER||LOYALTY||PROMOFORECAST||RCIMARKET	RIS1
RSP	RSP required programs	AC||CS||ASO||CDT||DT	RIS1

- Applications without any modules defined.

Name	Description	Modules	JosJobAdminName
RMS	RMS		RMS
ALLOC	ALLOC		RMS
RDE	RDE		RDE
REIM	REIM		RMS
RESA	RESA		RMS
RPM	RPM		RMS

Schedule Tab

The Schedule tab contains the name of the schedule along with version and description. Increasing the version is necessary to upload a spreadsheet with any changes to POM.

- **ScheduleName – VARCHAR(10)**
  - Holds the name of the Schedule (for example, `MERCH`, `RDE`, or `RI`).
- **Description – VARCHAR(100)**
  - Description of the Schedule.
- **Version – VARCHAR(50)**
  - Version of the Schedule.
- **DisplayScheduleName – VARCHAR(1000)**
  - Holds the display name of the Schedule (for example, `MERCH`, `RDE`, or `RI`).

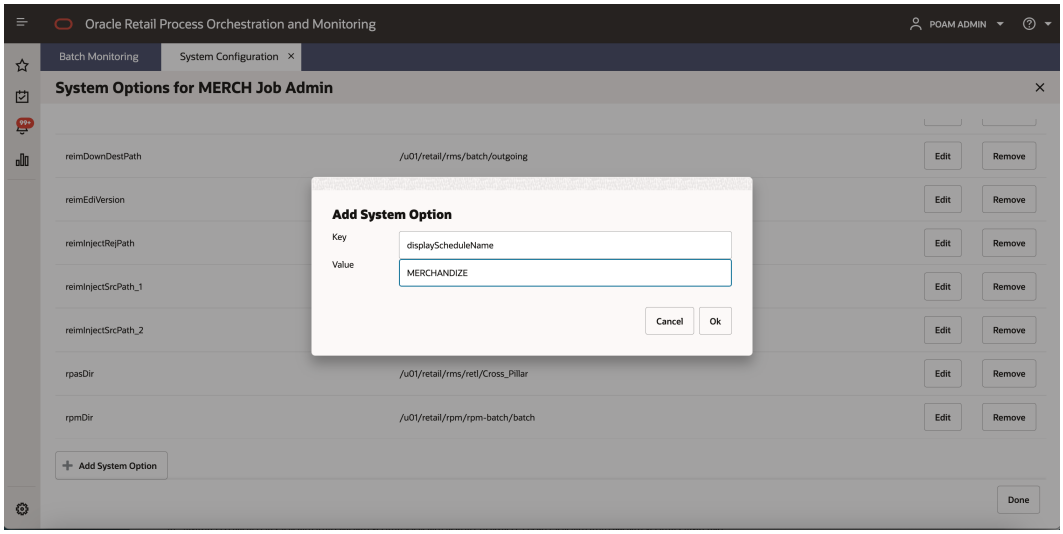
- Optional field. If a value is not provided, value in the **ScheduleName** field will be used as the display schedule name.
- Display schedule name will be used in the UI, reports and e-mails.

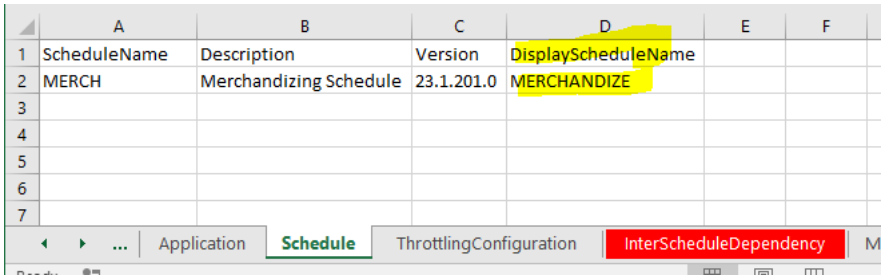
If a display schedule name value is not provided on the batch schedule spreadsheet, it can be provided at a later time through the UI. To accomplish that, an entry can be added in the **SystemOptions** panel from the System Configuration screen.

Example:

Key: displayScheduleName

Value: MERCHANDIZE





	A	B	C	D	E	F
1	ScheduleName	Description	Version	DisplayScheduleName		
2	MERCH	Merchandizing Schedule	23.1.201.0	MERCHANDIZE		
3						
4						
5						
6						
7						

Application | Schedule | ThrottlingConfiguration | InterScheduleDependency

This tab cannot be empty for any given schedule.

ThrottlingConfiguration Tab

The ThrottlingConfiguration tab contains the Application / Module Level throttling values. Throttling determines how many jobs can run concurrently for the given application, so a value of `10` means a maximum of ten jobs can run concurrently. It is advisable to leave this tab empty unless you have good knowledge of the server memory and CPU capacity.

- **Application – VARCHAR(50)**
  - Holds the valid application name for which we need to set a throttle value.
  - If an entry is made in this tab, the Application can't be blank and should be the value listed in Application tab.

- **Module – VARCHAR(50)**
  - Holds the module name in the case where we need to set a throttle value at module level.
  - It is possible to just set throttling at the application level. Modules would then inherit the application's throttle value. In this case, leave module blank.
  - This Application / Module combination must be valid per the Application tab.
- **ThrottledValue – NUMBER**
  - Holds the throttle value for this Application / Module.
  - This can't be blank and should be a non-zero positive number.

Example

- For an Application without any modules defined.

Application	Module	ThrottledValue
RMS		2
ALLOC		5
REIM		3
RDE		2

- For an Application with modules defined, define throttle values at either application or module level or both.

Application	Module	ThrottledValue
COMMON		3
COMMON	COMMONMAINTAIN	1
ORASE		2

If you don't want any jobs to be throttled at Application / Module level, then leave this tab empty.

InterScheduleDependency Tab

The InterScheduleDependency tab allows you to define dependencies from jobs on this schedule to jobs on other schedules on the same instance of POM. For example, job A of this schedule can be made dependent on job B from another schedule. Job A will then wait for completion of job B.

- **ProcessName – VARCHAR(50)**
  - Holds the name of the process from the current schedule that is dependent on a different schedule.
  - This process name should be listed in the **Process** tab.
- **JobName – VARCHAR(50)**
  - Holds the name of the corresponding job within the above process that has the dependency on another external schedule.
  - This Process Job combination should be valid per the **Process Job Mapping** tab.

- **ExternalScheduleName – VARCHAR(10)**
  - Holds the name of the external schedule that contains the dependency.
- **ExternalPredecessorProcessName – VARCHAR(50)**
  - Holds the name of the process that contains the job from an external schedule which needs to complete before the job in the current schedule starts running.
- **ExternalPredecessorJobName – VARCHAR(50)**
  - Holds the name of the corresponding job (from the external schedule) that must complete before the job in the current schedule starts running.

Example

ProcessName	JobName	ExternalScheduleName	ExternalPredecessorProcessName	ExternalPredecessorJobName
ALLOCBT_PROCESS	ALLOCBT_JOB	RDE	RDE_SETUP_PROCESS	RDE_REFRESHODIVARIABLES_JOB
DISTROPCPUB_PROCESS	DISTROPCPUB_JOB	RDE	RDE_SUPSDE_PRDITMSUPSDE_PROCESS	RDE_SUPSDE_JOB
CMPUPLD_PROCESS	CMPUPLD_JOB	RPAS	RPAS_PROCESS	RPAS_JOB
ALC_PURGE_WRK_PROCESS	ALC_PURGE_WRK_JOB	RIORASE	RIORASE_PROCESS	RIORASE_JOB

If there are no inter schedule dependencies for any jobs in the schedule, then leave this tab empty.

Modules Tab

The Modules tab allows you to define the mapping between the POM Modules and the Module Definition Framework (MDF) Modules. It also serves a second purpose by defining the modules for an application. If the module field on the **Application** tab is not blank, POM will use those modules. Otherwise, if blank, POM will use the modules from this tab.

MDF holds the applications and modules to which a customer has subscribed. POM can then optionally sync with MDF to activate or deactivate applications and/or modules.

- **Application – VARCHAR(50)**
  - Holds the valid application name for which we need to set an MDF Module Path.
  - This can't be blank and should be listed in **Application** tab.
- **Module – VARCHAR(50)**
  - Holds the module name for cases when the MDF Module Path must be set at module level.
  - This can be blank when the MDF Module Path is only set at the application level.
  - This Application / Module combination should be valid per the **Application** tab.
- **MDFModulePath – VARCHAR(4000)**
  - Holds the module path in the MDF application.
  - This can't be blank. This should be a slash (/) separated path.

    For example: `/RMS`, `/COMMON/COMMONAC`

Example

Application	Module	MDFModulePath
COMMON	COMMONAC	/COMMON/COMMONAC
COMMON	CALENDAR	/COMMON/CALENDAR
ORASE		/ORASE
COMMON		/COMMON
ORASERCI		/ORASERCI

ExternalDependencies Tab

This tab allows you to define the dependencies of POM jobs on external schedulers like Ctrl-M.

- **ProcessName – VARCHAR(50)**
  - Holds the name of the process in the current schedule for which the external job is to be configured.
  - This process name should be listed on the **Process** tab.
- **JobName – VARCHAR(50)**
  - Holds the name of the corresponding job of the above process for which the external job is to be configured.
  - This Process Job combination should be valid per **Process Job Mapping** tab.
- **ExternalPredecessorJobName – VARCHAR(300)**
  - Holds the name of the external job that needs to complete before the job in the current schedule can start.

Example

ProcessName	JobName	ExternalPredecessorJobName
FILE_VALIDATION_PROCESS	ORBATCH_VERIFY_RICOMMON_JOB	EXT_JOB_1
FILE_VALIDATION_PROCESS	ORBATCH_VERIFY_RMI_JOB	EXT_JOB_2
FILE_VALIDATION_PROCESS	ORBATCH_VERIFY_RCI_JOB	EXT_JOB_3

BatchLinks Tab

This tab contains Batch Links. When you set up a batch link, you direct POM to start a job in schedule B upon completion of a job in schedule A. It's a way of linking two schedules together and having one start the other. You want the job in schedule B to be the first job of the schedule.

If no such link is desired, leave this tab empty.

- **ProcessName – VARCHAR(50)**

  - Holds the name of the process in the current schedule that needs to be invoked by a process / job from an External Schedule.
  - This process name should be listed in **Process** tab.
- **JobName – VARCHAR(50)**
  - Holds the name of the corresponding job of the above process that needs to be invoked from the other external schedule.
  - This Process Job combination should be valid as per the **Process Job Mapping** tab.
- **InvokerScheduleName – VARCHAR(10)**
  - Holds the name of the external schedule whose job is to invoke the current schedule's job.
- **InvokerProcessName – VARCHAR(50)**
  - Holds the name of the process in the invoking schedule.
- **InvokerJobName – VARCHAR(50)**
  - Holds the name of the corresponding job from the invoking schedule.

Example

	A	B	C	D	E
1	**ProcessName**	**JobName**	**InvokerScheduleName**	**InvokerProcessName**	**InvokerJobName**
2	RDE_SETUP_PROCESS	RDE_REFRESHODIVARIABLES_JOB	MERCH	START_BATCH_PROCESS	START_BATCH_JOB
3					

If there are no batch links needed then leave this tab empty.

JobTypes Tab

This tab allows users to define custom job types. These are job types other than the POM-provided ones, namely: `EXEC`, `RI`, `RASE`, `BDI`, `RPAS`, `RDS`, `OMS`, and `OB`.

These custom job types are associated with the ReSTful endpoints necessary for POM to execute batch. Once defined on this tab, these can then be used in the JobType field on the **Job** tab.

- **Type – VARCHAR (10)**
  - Name of the job type (for example, `RDS`).
- **ValidationPath – VARCHAR (4000)**
  - ReST Endpoint path for validating that the endpoints for this job type are reachable (for example, `/validation`).
  - Can be left blank on this tab but, if so, needs to be provided later on the UI.
- **JobStartPath – VARCHAR (4000)**
  - ReST Endpoint path to start a job (for example, `/start`).
  - Mandatory field if a job type is defined on this tab.
- **JobRestartPath – VARCHAR (4000)**
  - ReST Endpoint path to restart a failed job (for example, `/restart`).

  - Can be left blank on this tab but, if so, needs to be provided later on the UI.
- **JobStatusPath – VARCHAR (4000)**
  - ReST Endpoint path to check the status of a previously submitted job (for example, `/status`).
  - Mandatory field if a job type is defined on this tab.
- **JobLogPath – VARCHAR (4000)**
  - ReST Endpoint path to fetch log file of a job (for example, `/logs`).
  - Can be left blank on this tab but, if so, needs to be provided later on the UI.
- **JobKillPath – VARCHAR (4000)**
  - ReST Endpoint path to kill a running job.
  - Can be left blank on this tab but, if so, needs to be provided later on the UI.
- **OAuthScopes – VARCHAR (4000)**
  - Comma-separated list of OAuth scopes for invoking the endpoints.
  - Mandatory field if a job type is defined on this tab.

If there are no custom job types in the schedule, then leave this tab empty.