TIBCO® Fulfillment Order Management Administration
Software Release 2.1.1
July 2014
SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN LICENSE.PDF) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE “LICENSE” FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document contains confidential information that is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of TIBCO Software Inc.

TIBCO, Two-Second Advantage, TIBCO ActiveMatrix BusinessWorks, TIBCO Runtime Agent, TIBCO Administrator, TIBCO Enterprise Message Service, and TIBCO BusinessEvents are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

EJB, Java EE, J2EE, and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

THIS SOFTWARE MAY BE AVAILABLE ON MULTIPLE OPERATING SYSTEMS. HOWEVER, NOT ALL OPERATING SYSTEM PLATFORMS FOR A SPECIFIC SOFTWARE VERSION ARE RELEASED AT THE SAME TIME. SEE THE README FILE FOR THE AVAILABILITY OF THIS SOFTWARE VERSION ON A SPECIFIC OPERATING SYSTEM PLATFORM.

THIS DOCUMENT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. TIBCO SOFTWARE INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND “READ ME” FILES.

Copyright © 2010-2014 TIBCO Software Inc. ALL RIGHTS RESERVED.

TIBCO Software Inc. Confidential Information.
## Contents

**Preface** ........................................................................................................................... 9

- Related Documentation.................................................................................................. 10
- Typographical Conventions......................................................................................... 11
- Connecting with TIBCO Resources............................................................................. 12

**Chapter 1 Deployment** ............................................................................................... 13

- Recommended Setup for a Fulfillment Order Management Development Environment ...... 14
- Deployment Topologies................................................................................................. 16
  - Single Node Single Instance Topology........................................................................ 16
  - Single Node Multi-Instance Topology......................................................................... 17
  - Multi-node Multi-instance Topology........................................................................... 23
- Components Deployment.................................................................................................. 27
  - Jeopardy Deployment.................................................................................................. 27
  - AOPD Deployment....................................................................................................... 27
  - Orchestrator Deployment............................................................................................ 27
- Collocation....................................................................................................................... 29

**Chapter 2 Configuration** ............................................................................................ 31

- Queue Management....................................................................................................... 32
- Status Listener Queues.................................................................................................. 32
- Data Models.................................................................................................................... 34
  - Model Loading............................................................................................................ 34
  - Models Required by Fulfillment Order Management Engines .................................. 34
  - Consumer of Fulfillment Catalog Models.................................................................... 35
  - Invoking Model Publisher in TIBCO Fulfillment Catalog........................................... 35
  - Offline Catalog............................................................................................................. 35
- Configuration.................................................................................................................. 37
  - Third-party Product Requirement................................................................................ 37
  - OMS Architecture......................................................................................................... 37
  - OMS Configuration...................................................................................................... 37
- Jeopardy Management System.......................................................................................... 50
  - JeOMS Configuration.................................................................................................... 50
- Offline Catalog................................................................................................................ 52
  - Accessing Offline Catalog............................................................................................ 52
  - Flat File Format............................................................................................................. 53
- Logging............................................................................................................................... 54
  - How Logging Works..................................................................................................... 54
  - Configuring Logging..................................................................................................... 55
Process Component Model....................................................................................................96
Database Schema..................................................................................................................98
Glossary.................................................................................................................................101
Preface

The preface contains information about documentation related to the current document, typographical conventions, and information on how to contact TIBCO support.
Related Documentation

This section lists documentation resources you may find useful.

- **TIBCO Fulfillment Order Management Concepts and Architecture**  Read this manual for information on TIBCO Fulfillment Order Management terminology and concepts.
- **TIBCO Fulfillment Order Management Installation and Configuration**  Read this manual for instructions on installation and configuration.
- **TIBCO Fulfillment Order Management Administration**  Read this manual for instructions on administration tasks.
- **TIBCO Fulfillment Order Management User’s Guide**  Read this manual for information on Fulfillment Order Management features, functionality, and all the screens.
- **TIBCO Fulfillment Order Management Web Services**  Read this manual for information about the web services.
- **TIBCO Fulfillment Order Management Release Notes**  Read the release notes for a list of features. This document also contains the list of known issues for this release.
Typographical Conventions

The following typographical conventions are used in this manual:

Table 1: General Typographical Conventions

<table>
<thead>
<tr>
<th>Convention</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIBCO_HOME</td>
<td>Many TIBCO products are installed within the same home directory. This directory is referenced in documentation as TIBCO_HOME. The value of TIBCO_HOME depends on the operating system. For example, on Unix systems the default value is $HOME/tibco.</td>
</tr>
<tr>
<td>TRA_HOME</td>
<td>TIBCO Runtime Agent installs into a directory inside ENV_HOME. This directory is referenced in documentation as TRA_HOME. The value of TRA_HOME depends on the operating system. For example, on Unix systems the default value is $TIBCO_HOME/tra.</td>
</tr>
<tr>
<td>AF_HOME</td>
<td>TIBCO Fulfillment Order Management installs into a directory inside ENV_HOME. This directory is referenced in documentation as AF_HOME. The value of AF_HOME depends on the operating system. For example, on Unix systems the default value is $TIBCO_HOME/af/2.1.</td>
</tr>
<tr>
<td><strong>code font</strong></td>
<td>Code font identifies commands, code examples, filenames, pathnames, and output displayed in a command window. For example: Use MyCommand to start the foo process.</td>
</tr>
<tr>
<td><strong>bold code font</strong></td>
<td>Bold code font is used in the following ways:</td>
</tr>
<tr>
<td></td>
<td>• In procedures, to indicate what a user types. For example: Type admin.</td>
</tr>
<tr>
<td></td>
<td>• In large code samples, to indicate the parts of the sample that are of particular interest.</td>
</tr>
<tr>
<td></td>
<td>• In command syntax, to indicate the default parameter for a command. For example, if no parameter is specified, MyCommand is enabled:</td>
</tr>
<tr>
<td></td>
<td>MyCommand [enable</td>
</tr>
<tr>
<td><strong>italic font</strong></td>
<td>Italic font is used in the following ways:</td>
</tr>
<tr>
<td></td>
<td>• To indicate a document title. For example: See TIBCO BusinessWorks Concepts.</td>
</tr>
<tr>
<td></td>
<td>• To introduce new terms. For example: A portal page may contain several portlets. Portlets are mini-applications that run in a portal.</td>
</tr>
<tr>
<td></td>
<td>• To indicate a variable in a command or code syntax that you must replace. For example: MyCommand pathname</td>
</tr>
<tr>
<td>☑️</td>
<td>The note icon indicates information that is of special interest or importance, for example, an additional action required only in certain circumstances.</td>
</tr>
<tr>
<td>🔴</td>
<td>The warning icon indicates the potential for a damaging situation, for example, data loss or corruption if certain steps are taken or not taken.</td>
</tr>
</tbody>
</table>
Connecting with TIBCO Resources

How to Join TIBCOmmunity

TIBCOmmunity is an online destination for TIBCO customers, partners, and resident experts—a place to share and access the collective experience of the TIBCO community. TIBCOmmunity offers forums, blogs, and access to a variety of resources. To register, go to http://www.tibcommunity.com.

How to Access All TIBCO Documentation

After you join TIBCOmmunity, you can access the documentation for all supported product versions here: https://docs.tibco.com.

How to Contact TIBCO Support

For comments or problems with this manual or the software it addresses, please contact TIBCO Support as follows:

• For an overview of TIBCO Support, and information about getting started with TIBCO Support, visit this site:
  http://www.tibco.com/services/support

• If you already have a valid maintenance or support contract, visit this site:
  https://support.tibco.com

Entry to this site requires a username and password. If you do not have a username, you can request one.
Chapter 1

Deployment

Topics

- Recommended Setup for a Fulfillment Order Management Development Environment
- Deployment Topologies
- Components Deployment
- Collocation
Recommended Setup for a Fulfillment Order Management Development Environment

The following details are the recommended setup for a Fulfillment Order Management Development Environment:

<table>
<thead>
<tr>
<th>Component</th>
<th>Type</th>
<th>Instances</th>
<th>Primary Server</th>
<th>Secondary Server</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>OMS (AOPD + Orchestrator + JeOMS + UI)</td>
<td>JAVA</td>
<td>1</td>
<td>M1</td>
<td></td>
<td>4 GB</td>
</tr>
<tr>
<td>Oracle</td>
<td>Database</td>
<td>1</td>
<td>M2</td>
<td></td>
<td>8 GB</td>
</tr>
<tr>
<td>EMS</td>
<td>Messaging</td>
<td>1</td>
<td>M1</td>
<td></td>
<td>Default</td>
</tr>
<tr>
<td>Process Component (BW or BE)</td>
<td>JAVA</td>
<td>1</td>
<td>M1</td>
<td></td>
<td>2 GB</td>
</tr>
<tr>
<td>OCV Cache (optional)</td>
<td>BE</td>
<td>1</td>
<td>M1</td>
<td></td>
<td>1 GB</td>
</tr>
<tr>
<td>OCV agent (optional)</td>
<td>BE</td>
<td>1</td>
<td>M1</td>
<td></td>
<td>1 GB</td>
</tr>
</tbody>
</table>

**Hardware**

The recommended configuration for M1 and M2 are 8 core CPU @ 2 GHz with 16 GB RAM

**Disk Space**

To install Fulfillment Order Management and all the prerequisite software, a disk space of 10 GB is required on machine M1. An additional space of 10 GB is required on M2 for oracle software and tablespaces. The oracle tablespace created should be at least 100 MB.

**Temporary Disk Space for UNIX platform**

The installer launcher first extracts a Java Virtual Machine (JVM) in a temporary directory and uses this JVM to launch itself. The size of the extracted JVM differs from platform to platform.

On UNIX platforms, the following disk space is required in the temporary area:

256 MB of free disk space in /tmp location.

If your system does not have sufficient free disk space in the above temporary area, you can still run the installer with a different temporary area by using the following option when starting the installer:

```bash
install_package_name.bin -is:tempdir /new_tmp
```

where /new_tmp has sufficient free disk space.

**Jeopardy Deployment Configuration**

Jeopardy deployment, in the colocated mode, will be in the form of single instance, or multiple instance cluster setups, and will be deployed with the following configuration:

<table>
<thead>
<tr>
<th>Fields</th>
<th>Colocated configurations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profiles.properties</td>
<td>com.tibco.fom.jeoms.deploymentMode = JEOMS_colocated</td>
</tr>
<tr>
<td>ConfigValues_JeOMS.xml</td>
<td>Operational Datastore = cache</td>
</tr>
<tr>
<td>Tomcat webapps directory</td>
<td>jeoms.war file NOT present</td>
</tr>
</tbody>
</table>
• Local and ActiveSpace data store will not be supported in this version.
• If Jeopardy is enabled, which means that the JeOMS deploy mode is set to `JEOMS_colocated`, then every FOM cluster instance has to run JeOMS in colocated mode. What it means is that configurations having some server instances set to `JEOMS_disabled` and some server instances set to `JEOMS_colocated` will no longer be supported.
Deployment Topologies

TIBCO® Fulfillment Order Management server side components (i.e. omsServer (including Orchestrator), AOPD, and JeOMS) and the client side user interface components (i.e. configurator and omsUi with dashboard) need to be deployed in Apache Tomcat server for leveraging their functionalities. Any one of the three topologies explained below can be followed for the deployment.

Single Node Single Instance Topology

In Single Node Single Instance topology, default Apache Tomcat server instance under the TIBCO® Fulfillment Order Management installation is started on single server node in the production environment. The web application archive files (.war) for all the server side and client side components should ideally be deployed in the default Tomcat server so as to use the server components’ functionalities and also access the GUI.

This is the default topology that is supported immediately after the installation and initial configurations of Fulfillment Order Management.

Single Node Multi Instance (Vertical Scaling) Topology

In Single Node Multi Instance topology, the default Apache Tomcat server instance under the Fulfillment Order Management installation on single server node is duplicated so as to create and start multiple instances in the cluster. The user can opt to deploy all Fulfillment Order Management components in all Apache Tomcat servers so as to access the omsUi and configurator applications through the HTTP interface of any one of or multiple Tomcat servers. Optionally they can also choose to deploy only the server side components - omsServer (Orchestrator and AOPD included) and JeOMS into the duplicated Tomcat servers so as to just increase the processing capabilities.

This topology is also referred as Vertical Scaling. It allows the components to leverage the processing power efficiency of the machine. One Apache Tomcat instance runs in a single Java Virtual Machine process. However, the inherent concurrency limitations of a JVM process means that the components cannot fully utilize the processing power of the machine.

By running additional Apache Tomcat server instances, multiple JVM processes are started and provide multiple thread pools. This allows the server components deployed in the Tomcat server to utilize the maximum processing power of the machine.

Multi Node Multi Instance (Horizontal Scaling) Topology

In Multi Node Multi Instance topology, multiple Apache Tomcat server instances are started on multiple nodes so as to have multiple instances running in the cluster. The user can opt to deploy all Fulfillment Order Management components in all Apache Tomcat servers so as to access the omsUi and configurator applications through the HTTP interface of any one of or multiple Tomcat servers. Optionally they can also choose to deploy only the server side components - omsServer (Orchestrator and AOPD included) and JeOMS into the duplicated Tomcat servers so as to just increase the processing capabilities.

This topology is also referred as Horizontal Scaling. It allows the components to leverage the processing power efficiency of multiple machines. By running additional Apache Tomcat server instances on multiple machines, multiple JVM processes are started and provides multiple thread pools. This allows the server components deployed in the Tomcat servers to use the maximum processing power of all the machines.

Single Node Single Instance Topology

This topology is supported by default once the Fulfillment Order Management installation and configurations are done. The default Apache Tomcat server that comes with Fulfillment Order Management installation is started to use this topology. Fulfillment Order Management installer updates the values of couple of environment variables such as $AF_CONFIG_HOME in the environment script (setenv.sh) used by Apache Tomcat as per the installation directory path. $AF_CONFIG_HOME points to $AF_HOME/config directory containing various configuration files used by the Fulfillment Order Management components.
For example, omsServer component refers some of the configuration files as mentioned below:

1. `ConfigValues_OMS.xml` for all the configuration properties such as JMS connection parameters, database connections, and other functional configuration properties.
2. `OMSServerLog4j.xml` for Logging configurations
3. `ascache-config.properties` for ActiveSpaces configurations, only if the second level cache for Hibernate is enabled through the configuration property `com.tibco.af.oms.hibernate.cache.use_second_level_cache`.

![Figure 1: Single Node Single Instance Topology](image)

The configuration properties for omsServer and jeoms components in the respective configuration files are divided into two levels.

1. Member level properties which can be tweaked for each omsServer or jeoms member.
2. Cluster level properties which are common for all the omsServer or jeoms members.

The member level properties are identified and picked up by each deployed omsServer or jeoms member based on the NODE_ID value passed in the environment script (`setenv.sh`) in Apache Tomcat server. The default value of NODE_ID in `setenv.sh` is `Member1`. The respective configuration files also contains the default member level configuration section `<Server name="Member1">`.

Most of the required configurations such as JMS connection, database connection can be done using the TCT post installation. Optionally the configurations can also be changed using the configurator application. Once all the required configurations are in place, the default Apache Tomcat server containing the war files for all the Fulfillment Order Management components deployed can started to use this topology.

**Single Node Multi-Instance Topology**

The value of NODE_ID property need to be modified and unique in the `setenv.sh` script under each Apache Tomcat server instance. This is required for the components deployed in each Apache Tomcat server instance to refer the corresponding member level properties from the respective configuration files. It is also required from Orchestrator’s clustering standpoint.

Orchestrator component is an integral part of omsServer. By running multiple instances of Apache Tomcat server, Orchestrator instances form a cluster. The name of the cluster is passed as DOMAIN_ID property in the setenv script under each Apache Tomcat instance with the default value as 'ORCH-DOMAIN' and must be same in all the scripts.

The number of consumer counts (listeners) on various JMS queues and topic used by different Fulfillment Order Management components will be equal to the product of the count configured and the total number of instances running.

For example, if 4 Apache Tomcat Server instances are started on a node, the total number of listeners on `tibco.aff.orchestrator.order.submit.queue` will be 4 times the value configured in `ConfigValues_OMS.xml`.

The following steps will help to use this deployment topology.

**Prerequisites**
1. TIBCO Fulfillment Order Management is installed on top of all the underlying required products on the designated server node.
2. Correct configurations for the default topology i.e. *Single Node Single Instance* are in place.
3. The default Apache Tomcat server instance under $AF_HOME$ is up and running with all the server and client side components deployed successfully.

Creating New Cluster Members

2. Select Order Management System configuration.
3. Select existing member Member1 in the cluster. Click the right mouse button to pop up the menu. Select Clone Member as shown in the following figure:

4. In the Clone Member dialog, provide a unique name for the new member in cluster, for example, *Member2* and click on the Create button. A new cluster member namely *Member2* will be created.

At this step, a copy of the complete <Server name="Member1"> element will be created as <Server name="Member2"> containing the exactly same configuration properties as the original element in ConfigValues_OMS.xml.

5. Change the JMX RMI port for Member2 using the following configuration:
6. Save the configuration changes. Newly created Member2 is saved in the 
   `$AF_HOME/config/ConfigValues_OMS.xml` file.

7. Select User Interface configuration for Member2 cluster member and change the values for the following 
   properties.

   Provide unique values as displayed:

   ![Figure 5: User Interface](image)

8. Save the port configuration changes for Member2.

9. Perform steps 3 on page 18 to 8 on page 19 for creating additional members Member3, Member4, Member5 
   and so on in the cluster. This should be done as per the requirement.

10. Save the configuration changes. The Configuration Saved popup appears.

11. At this step, number of copies of the complete `<Server name="Member1">` element will be created as 
    `<Server name="Member2">`, `<Server name="Member3">`.

Please note that the deployment of multiple OMS instances does not provide HTTP load balancing capabilities 
out of the box. However any third party load balancer can be used to balance the load across multiple instances 
of OMS. For this, no specific configuration is required. The only requirement is the load balancer must have 
support for the sticky session. Sticky sessions means the load balancer always directs a given client to the 
same back-end server.

Hardware Load Balancer (HLB), which has Layer 7 capability, can direct the traffic and maintain session 
persistence for the web applications without relying on the user’s IP address with session cookies. Typically, 
HLB inserts a cookie that the load balancer creates and manages automatically to remember which back-end 
server a given HTTP connection should use. And then it would always direct the request originating from 
that client browser to the same server. Some of the HLBs which support layer 7 capability include Barracuda, 
jetNexus, and F5.

The load balancing for JMS interfaces is provided out of the box. The consumer count on each incoming JMS 
destination used by omsServer and jeoms will be automatically multiplied by the total number of deployed 
instances.

**Adding Cluster Members into Database**

1. The entry for default member Member1 in default cluster ORCH-DOMAIN is already present is 
   `DOMAINMEMBERS` table in OMS database.
2. In order to deploy and run additional members in Orchestrator cluster, corresponding entries must be added. For this, refer the following insert statement in $AF_HOME/db/oracle/oms/OMS_SeedData.sql script.

```
insert into domainmembers (memberid, description, domainid, clusterid, isclustermanager, seqnumber, heartbeattimestamp, lastupdatetimestamp, status) values ('Member1', 'This is Member1 in Orchestrator cluster domain.', 'ORCH-DOMAIN', null, null, 0, null, null, null);
```

3. Replace the values for memberid and description columns in the insert statement as per the names of the additional members created in previous steps. Prepare one insert statement per new member to be added.

4. Run the inserts statements prepared for all the new members on OMS DB schema. Also run the Commit statement to commit the insert changes.

5. Run select * query on DOMA IN MEMBERS table and verify the newly added members.

Always add only the required number of member entries into DOMAINMEMBERS table. For example, if 10 instances are required to be run in ORCH-DOMAIN cluster, add only the corresponding 10 entries into DOMAINMEMBERS table. For deleting the existing entries corresponding to the members which are not required to run, the following delete statement can be used by replacing the placeholder entries in the parenthesis with the actual memberid entries.

```
delete from domainmembers where memberid in ('member1_id', 'member2_id>', ...);
```

Configuration of ActiveSpaces Data Grid for Members

If the property `com.tibco.af.oms.hibernate.cache.use_second_level_cache` is set to false to disable the second level cache for Hibernate, the following steps corresponding to ActiveSpaces configurations are not required and should be skipped.

1. Go to the $AF_HOME/config directory.

2. The ascache-config.properties file contains the configuration properties for TIBCO ActiveSpaces such as metaspace and space configurations. This single file is referred by all the members in the cluster.

3. Change the value of interface address 0.0.0.0 in metaspace.discovery.url property with the static IP address assigned to the node. For example, the property should look like:

   ```
   metaspace.discovery.url=tibpgm://7889/10.107.134.105;239.8.8.9
   ```

   where 10.107.134.105 is the IP address of the node.

4. Change the value of interface address 0.0.0.0 in metaspace.listen.url property with the static IP address assigned to the node. For example, the property should look like:

   ```
   ```

   where 10.107.134.105 is the IP address of the node.

5. The default capacity for all the tuple spaces is configured to 1000. This means that at any given point in time maximum 1000 tuples of a specific type will be cached in the corresponding tuple space. For performance improvement, the capacity for each tuple space must be set to a higher value such as 200000. This will help Hibernate to cache more data and avoid the database round-trips. For example, the below mentioned properties should look like:

   ```
   space.default.capacity=200000
   region.<space_name>.space.capacity=200000
   ```

   where <space_name> is the name of the individual tuple space such as 'com.tibco.aff.oms.server.domain.ModelOrder'.

   The Java heap space must be configured to a higher value such as 8GB in setenv.sh script in each Apache Tomcat server in order to support a higher value like 200000 for tuple space capacity.
6. Refer TIBCO ActiveSpaces documentation for all the details on metaspace, space and other related concepts.

Creating Additional Apache Tomcat Instances

1. Stop the running instance of the default Apache Tomcat server.
2. Create as many number of copies of the apache-tomcat-<version> directory under $AF_HOME directory as the number of members configured previously to be run in the cluster. This can be done as follows (assuming total three members are required):

   > cd $AF_HOME
   > cp -r apache-tomcat-7.0.42 apache-tomcat-7.0.42-Member2
   > cp -r apache-tomcat-7.0.42 apache-tomcat-7.0.42-Member3

3. Modify the connector ports and SHUTDOWN port in the conf/server.xml file under each Tomcat instance as shown in the figure. Set the values of connector and redirect ports exactly same as the unique values assigned for each newly added member in configurator previously.

   ```xml
   <Connector port="8081" protocol="HTTP/1.1" 
   connectionTimeout="20000" 
   redirectPort="8445" URIEncoding="UTF-8"/>

   <!-- A "Connector" using the shared thread pool-->
   ```

   Figure 6: server.xml file

4. Modify the NODE_ID (member name) for each Tomcat instance in the bin/setenv file. Specify the NODE_ID as the name of the cluster member, for example, Member2.

   ```bash
   $AF_HOME/bin/omscfg.bat -file $AF_HOME/bin/omscfg.bat
   ```

The cluster deployment is primarily done to scale the server side Fulfillment Order Management components i.e. omsServer (Orchestrator and AOPD included) and jeoms.

The server components are actually scaled through EMS as more number of listeners and processors are activated on inbound destinations by deploying them in multiple Apache Tomcat servers. This also creates additional thread pools to increase the processing capabilities.

Cluster Management Helper Utilities

In order to manage the cluster activities on a particular node such as deploying web application archive files into multiple Apache Tomcat server instances, starting up or shutting down multiple Apache Tomcat server instances, backing up logs under multiple Apache Tomcat server instances and so on in one go, few helper utilities listed below can be used. These utilities are present under $AF_HOME/bin.

1. backupAllTomcatServersLogs.sh

   This utility can be used to backup the log directories under all Apache Tomcat server instances on a particular node in one go. The utility expects a string argument to create the zip file into which the log
directory will be backed up. This is typically done before restarting the cluster instances newly. For example:

$ ./backupAllTomcatServersLogs.sh logs_23aug

2. `deleteAllTomcatServersLogs.sh`
   This utility can be used to delete all the log files under log directory of all Apache Tomcat server instances on a particular node in one go. This is typically done after backing up the logs and before restarting the cluster instances newly.

3. `deleteAllTomcatServersWarFiles.sh`
   This utility can be used to delete all the existing web application archive files (.war) and the corresponding exploded directories deployed under webapps directory under all Apache Tomcat server instances on a particular node in one go. This is typically done as part of the cleanup activities.

4. `deployWarFilesToAllTomcatServers.sh`
   This utility can be used to deploy the web application archive files (.war) present under `$AF_HOME/oms/webapps` directory into the webapps directories under all Apache Tomcat server instances on a particular node in one go. The utility can also be used to deploy the `$AF_HOME/configurator/webapps/config.war`, if required. However for this, two command lines in the utility which are commented by default will need to be uncommented.

5. `killAllTomcatServers.sh`
   This utility can be used to kill the Unix processes corresponding to all the running instances of Apache Tomcat servers on a particular node in one go.

6. `shutdownAllTomcatServers.sh`
   This utility can be used to shutdown all the running instances of Apache Tomcat servers on a particular node in one go.

7. `startAllTomcatServers.sh`
   This utility can be used to startup all the instances of Apache Tomcat servers on a particular node in one go.

All these utilities have been developed considering total 10 Apache Tomcat server directories under `$AF_HOME`. Following are the assumptions/prerequisites to use them.

1. All 10 Apache Tomcat directories are under `$AF_HOME`.
2. The name of the default Apache Tomcat directory is `apache-tomcat-<version>` and has not been changed.
3. The names of the additional 9 Apache Tomcat directories are `apache-tomcat-<version>-Member2`, `apache-tomcat-<version>-Member3`...to `apache-tomcat-<version>-Member10`.

These are just the reference utilities that have been developed with certain assumptions. The user can enhance/modify them as per their requirements to manage more instances in much better way.

**Sanity Test**

1. Start all Apache Tomcat server instances to start the deployed Fulfillment Order Management components.
2. Monitor the logs of each Apache Tomcat server to make sure that the server and all the deployed components have started correctly without any errors.
3. The Orchestrator member in any one of the Tomcat servers must be started as a Cluster Manager for ORCH-DOMAIN cluster. Whereas the Orchestrator members in all other Tomcat servers must be started as Workers in the same cluster. This can be verified as per the sample log statements shown below in `catalina.out` log file.
4. The member in each Tomcat instance connects to the default metaspace `oms_metaspace` created in TIBCO ActiveSpaces as configured in `ascache-config.properties` file. Each member also discovers every other member joining the same metaspace and creates the configured tuple spaces to cache the data tuples. This enables the underlying Hibernate framework to use TIBCO ActiveSpaces as the second level cache. It can be verified as per the sample log statements shown below in `catalina.out` log file.

5. The user can also use as-admin Java utility under `$AS_HOME/lib` to monitor the various members connected to `oms_metaspace` metaspace and browse the tuple spaces. The following sample screen cap shows all the connected members to the `oms_metaspace` including Member1, Member2, and `as-admin` member itself.

6. Once it is confirmed that all the members in the cluster are started correctly, the user can submit couple of orders using the SOAP/JMS interface. The order messages will be equally distributed for processing among all the members.

7. Per default configuration, the member which has accepted the submit order message for a specific order will only completely process that order. This means all the further messages for that order from external systems such as process components will be picked up and processed only by that particular member.

**Multi-node Multi-instance Topology**

The Multi Node Multi Instance or Horizontal Scaling deployment topology can be achieved by running multiple instances of Apache Tomcat server on multiple nodes (machines). This is mainly done to increase the overall processing capacity of the Fulfillment Order Management server side components viz. omsServer.
and jeoms. The default Apache Tomcat server instance which comes with Fulfillment Order Management installation is duplicated on the main and other nodes so as to run multiple instances.

For the Apache Tomcat instances on other nodes to refer the configurations, the $AF_HOME/config directory on the main node needs to be copied on the other nodes.

The explanation for NODE_ID and DOMAIN_ID and the note for consumer counts given in Single Node Single Instance section holds true here as well.

The following steps will help to use this deployment topology.

**Prerequisites**
1. The setup for Single-node Multi-instance deployment topology is already in place and working on the main node on which Fulfillment Order Management is installed.
2. All the required underlying software such as Java, TIBCO ActiveSpaces and the libraries such as Oracle JDBC driver, EMS libraries which are being referred in setenv script of Apache Tomcat server are already available on the additional nodes. All the corresponding environment variables are set and exported.

**Creating New Cluster Members**
1. Access Fulfillment Order Management configurator GUI application in a supported browser through the HTTP interface of the default Apache Tomcat server instance on main node using the URL http://<HOST>:<PORT>/config/launchConfig.html.
2. Create additional members to be run on additional nodes by cloning the existing members. The easiest way is to create one clone of each existing member to have one new member and just change the member name. As the new member needs to be run on altogether different node, the ports can be kept as-is.
   a. For example, assume that there are 10 existing members namely Member1, Member2...Member10 in Single-node Single-instance topology. Clone member Member1 and rename the cloned member to create Member11.
3. The port numbers need not be changed for Member11 in configurator. As it will run on altogether different node, the ports, although same, will not conflict with Member1. Follow the same procedure for the remaining instances.
4. Once the required number of additional members to be run on other nodes is created in the configuration files, the configuration files need to be copied on the other nodes.
   a. Create a directory path on the other nodes similar to the directory path for $AF_HOME on the main node. For example, /home/fomuser/tibco/af/2.1.
   b. Copy the complete $AF_HOME/config directory from main node on all other nodes under the directory path created in step 4.a on page 25.
   c. Set $AF_CONFIG_HOME environment variable pointing to the directory path of the copied config directory. For example, /home/fomuser/tibco/af/2.1/config.

Adding Cluster Members into Database
The entries for additional members to be run on other nodes need to add into DOMAINMEMBERS table in the same way as explained in Single-node Multi-instance topology earlier.

Configuration of ActiveSpaces Data Grid for Members
If the property com.tibco.af.oms.hibernate.cache.use_second_level_cache is set to false to disable the second level cache for Hibernate, the following steps corresponding to ActiveSpaces configurations are not required and should be skipped.

1. Open ascache-config.properties file under the copied config directory on each additional node.
2. Change the value of interface address 0.0.0.0 in metaspace.discovery.url and metaspace.listen.url properties with the static IP address assigned to the node. For example, the properties should look like:


Where, 10.107.134.106 is the IP address of one particular additional node.

Creating Additional Apache Tomcat Instances
1. Additional Apache Tomcat instances can be created on other nodes by copying the existing Apache Tomcat directories from the main node.
2. The easiest way is to copy each Apache Tomcat directory and just change the member’s suffix number in the directory name.

   For example, assume that there are 10 existing Apache Tomcat directories on the main node namely apache-tomcat-<version>-Member1, apache-tomcat-<version>-Member2…
3. The port numbers in conf/server.xml need not be changed for Member11. As it will run on altogether different node, the ports, although same, will not conflict with Member1. Follow the same procedure for the remaining instances.
4. Modify the NODE_ID (member name) property in the bin/setenv file for each copied Apache Tomcat instance. Specify the NODE_ID exactly same as the name of the new cluster member, for example, Member11.

Cluster Management Helper Utilities
All the utilities cluster management helper utilities explained in Single-node Multi-instance topology earlier can also be used in the same way for Multi-node Multi-instance topology. The utilities need to be copied from the main node into the manually created $AF_HOME/bin directory on the other nodes.

Sanity Test
The sanity test for Multi-node Multi-instance topology is exactly same as explained in Single-node Multi-instance topology. Here additional number of Orchestrator instances running on other nodes will join the ORCH-DOMAIN cluster. Any one instance among all will act as the Cluster Manager and all other as Workers. Also, additional members will connect to the default metaspace oms_metaspace created in TIBCO
ActiveSpaces and discover each other to cache the data into the created tuple spaces acting as second level cache for Hibernate framework.
Components Deployment

Jeopardy Deployment

The Jeopardy component can be deployed only in colocated mode with OMS in its own application server. The deployment mode for Jeopardy can be configured in the setenv.bat or setenv.sh file, by using the following property:

• Local JeOMS Deployment: "-Dspring.profiles.active=JEOMS_colocated"

Or it can be more permanently set in the $AF_HOME/config/profiles.properties:

• com.tibco.fom.jeoms.deployMode=JEOMS_colocated

To start and stop Jeopardy, just use the main scripts to administrate Fulfillment Order Management:

• startup.sh: to start
• shutdown.sh: to stop

AOPD Deployment

The AOPD component can be deployed either collocated with OMS (i.e. default mode) or it can be deployed in standalone mode, in its own application server. The deployment mode for AOPD can be configured in the setenv.bat or setenv.sh file, by using the following property:

• Remote AOPD Deployment: "-Dspring.profiles.active=AOPD_standalone"
• Local AOPD Deployment: "-Dspring.profiles.active=AOPD_colocated"

Or it can be more permanently set in the $AF_HOME/config/profiles.properties:

• com.tibco.fom.aopd.deployMode=AOPD_colocated
• com.tibco.fom.aopd.deployMode=AOPD_standalone

To start and stop AOPD, just use the main scripts to administrate Fulfillment Order Management:

• startup.sh: to start
• shutdown.sh: to stop

Orchestrator Deployment

The functionality and interfaces of the default Java based Orchestrator component are enabled only through the colocated deployment with omsServer which is the default mode. Orchestrator cannot be deployed in a standalone mode like AOPD since there is no separate web application archive (.war) file for it. However, the Orchestrator component can be disabled on the deployment of omsServer by changing the configuration in $AF_HOME/config/profiles.properties file.

To keep Orchestrator enabled, keep the default configuration as follows
com.tibco.fom.orchestrator.deployMode=Orchestrator_colocated

To disable Orchestrator, configure as follows
com.tibco.fom.orchestrator.deployMode=Orchestrator_standalone

This configuration should only be done to use any external Orchestration engine such as iPC instead of the default Java based Orchestrator.

To start and stop Orchestrator, just use the main scripts to administrate Fulfillment Order Management:

• startup.sh: to start
• `shutdown.sh`: to stop

Starting with Fulfillment Order Management 2.1.0, there is no more need for ant scripts or TIBCO Administrator to start or stop Orchestrator.
Collocation

Some components can be either deployed on the same server (J2EE server) than OMS, or they can be deployed on their own. By default, all components are collocated as part of OMS.

See the Configuration section for more details on how to configure the different deployments.
Chapter

2

Configuration

Topics

• Queue Management
• Data Models
• Configuration
• Jeopardy Management System
• Offline Catalog
• Logging
Queue Management

It is recommended to use the Configurator to manage the queues configuration.

Status Listener Queues

When OMS receives orders, it sends the orders to JMS queues which are routed to the Orchestrator engine based on type of the router configurator and routing condition specified by the administrator.

Multiple concurrent listeners can be listening for incoming orders on the Listener queue to allow parallel processing of incoming orders. The number of such concurrent listeners is configurable.

OMS receives notification on status for orders in the system for fulfillment and supports two routers types, viz. Pass-through router and Filter based router.

Pass-through does not apply any condition on the incoming order message and passes the message to the default orchestrator.

Figure 11: Listener Queues

The following table describes the properties of the Listener Queues.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order Status Notification Queue</td>
<td>Destination queue for receiving status message from Orchestrator</td>
</tr>
<tr>
<td>Order Status Notification Queue Concurrent Listener</td>
<td>Order Status Notification Queue Concurrent Listener Count</td>
</tr>
<tr>
<td>Order Line Status Notification Queue</td>
<td>Order Line Status Notification Queue</td>
</tr>
<tr>
<td>Order Line Status Notification Queue Concurrent Listener</td>
<td>Order Line Status Notification Queue Concurrent Listener Count</td>
</tr>
<tr>
<td>Plan Status Notification Queue</td>
<td>Plan Status Notification Queue</td>
</tr>
<tr>
<td>Name</td>
<td>Descriptions</td>
</tr>
<tr>
<td>-----------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Plan Status Notification Queue Concurrent Listener</strong></td>
<td>Plan Status Notification Queue Concurrent Listener Count</td>
</tr>
<tr>
<td><strong>Plan Item Status Notification Queue</strong></td>
<td>Plan Item Status Notification Queue</td>
</tr>
<tr>
<td><strong>Order Amendment Status Notification Queue</strong></td>
<td>Order Amendment Status Notification Queue</td>
</tr>
<tr>
<td><strong>Set Plan Queue</strong></td>
<td>Destination queue for receiving execution plan from Orchestrator</td>
</tr>
<tr>
<td><strong>Set Plan Item Queue</strong></td>
<td>Destination queue for receiving execution plan item from Orchestrator</td>
</tr>
<tr>
<td><strong>Set Plan Fragment Model Queue</strong></td>
<td>Destination queue for receiving plan fragment from Fulfillment Catalog</td>
</tr>
<tr>
<td><strong>Status Notification Dead Queue</strong></td>
<td>Destination queue for poison message for status notification. OMS will move the message to the dead letter queue after retrying a pre-configured number of times to avoid messages becoming poison message</td>
</tr>
<tr>
<td><strong>Set Plan Dead Queue</strong></td>
<td>Dead letter queue destination for execution plan</td>
</tr>
<tr>
<td><strong>Set Plan Item Dead Queue</strong></td>
<td>Dead letter queue destination for execution plan item</td>
</tr>
<tr>
<td><strong>Set Plan Fragment Dead Queue</strong></td>
<td>Dead letter queue destination for plan fragment</td>
</tr>
<tr>
<td><strong>Update Jeopardy Configuration Rule Queue</strong></td>
<td>Destination to send jeopardy rule updates</td>
</tr>
<tr>
<td><strong>Enrich Migrated Plan Request Queue</strong></td>
<td>Destination to send request to JeOMS to enrich execution plan with the jeopardy information</td>
</tr>
</tbody>
</table>

The independent AFI, AFS, and TDS components have been rewritten in release 2.0.0 and they are now an integral part of OMS server. So apart from the above listed queue configurations for core OMS component which are present since version 1.1.0, the queue/topic configurations corresponding to these three components have been added newly under separate categories in the configurator.

For details, refer to the Global Variables and Configurations section in Fulfillment Order Management User Guide.
Data Models

Model Loading

On startup, OCV engines send a JMS message as a model publish request on the designated queues to OMS-AFI.

OCV on queue tibco.aff.catalog.events.request

JeOMS on queue tibco.aff.orchestrator.startup.event.request

Using the engineName header property in the request message, OMS-AFI identifies the requesting engine type and publishes the "required" & "optional" data models to them as given in the above matrix.

Model publish request is processed by OMS-AFI using either of the two modes based on the configurations for the particular model.

1. Online - By invoking catalog publish workflow in TIBCO Fulfillment Catalog.
2. Offline - By reading and publishing the contents from the offline catalog files that are exported earlier from TIBCO Fulfillment Catalog.

Product and Action model loading in AOPD follows the steps:

1. Store Action and Product models into data_model OMS database table for both online and offline loading.
2. AOPD Engine on startup will load Action and Product models from OMS database.
3. Repositories will be created locally in AOPD and each server will have its own copy of repositories in the $AF_HOME/aopd/work/.. folder created by the build script.
4. Execution plan and amendment plan listeners will be active only in standalone mode.
5. For collocated mode there will be direct API call from AFI to AOPD bypassing the queues and listeners configuration for Execution plan and amendment plan requests.
6. Action and Product model listeners will be active for both standalone and collocated mode.
7. If something gets published online AOPD will get latest updates on Action and Product model listeners.

If an existing product is loaded again then it will be first updated in OMS database, deleted from local repository and the incoming product model will be loaded newly in AOPD.

Models Required by Fulfillment Order Management Engines

The Fulfillment Order Management components - AOPD, OCV, and Orchestrator require the context of specific data models in order to generate and execute the execution plan for fulfillment of an order. Refer to the following matrix for the "required", "optional", and "not required" models for each component.

<table>
<thead>
<tr>
<th>Components</th>
<th>Data Models</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Product</td>
</tr>
<tr>
<td>AOPD</td>
<td>Required</td>
</tr>
<tr>
<td>OCV</td>
<td>Required</td>
</tr>
<tr>
<td>Orchestrator</td>
<td>Not Required</td>
</tr>
<tr>
<td>JeOMS</td>
<td>Not Required</td>
</tr>
</tbody>
</table>
All these components enable their incoming JMS destinations and process the requests only after the 'required' models are published to them and loaded.

**Consumer of Fulfillment Catalog Models**

In online mode, TIBCO Fulfillment Catalog processes the catalog publish workflow request received from OMS-AFI and publishes the bulk models on the designated JMS topics in the configured EMS server. These models should be available on the below mentioned queues for TIBCO Fulfillment Order Management to process and consume them.

- tibco.aff.catalog.customer.request
- tibco.aff.catalog.product.request
- tibco.aff.catalog.segment.request
- tibco.aff.catalog.action.request
- tibco.aff.catalog.planfragment.request
- tibco.aff.oms.planfragmentmodel

Create JMS bridges between the designated TIBCO Fulfillment Catalog topics and the corresponding queues.

<table>
<thead>
<tr>
<th>SOURCE TOPIC</th>
<th>TARGET QUEUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>tibco.ac.productmodel.topic</td>
<td>tibco.aff.catalog.product.request</td>
</tr>
<tr>
<td>tibco.ac.customermodel.topic</td>
<td>tibco.aff.catalog.customer.request</td>
</tr>
<tr>
<td>tibco.ac.segmentmodel.topic</td>
<td>tibco.aff.catalog.segment.request</td>
</tr>
<tr>
<td>tibco.ac.actionmodel.topic</td>
<td>tibco.aff.catalog.action.request</td>
</tr>
<tr>
<td>tibco.ac.planfragmentmodel.topic</td>
<td>tibco.aff.catalog.planfragment.request</td>
</tr>
<tr>
<td>tibco.ac.planfragmentmodel.topic</td>
<td>tibco.aff.oms.planfragmentmodel</td>
</tr>
</tbody>
</table>

The schema for plan fragment model is

```
$AF_HOME/schemas/schema/orchestrator/sharedResources/schemas/classes/planFragmentModel.xsd
```

The schema for product, customer, segment, and action models is available in the

```
$AF_HOME/schemas/schema/afi/SharedResources/Schemas/Classes directory.
```

**Invoking Model Publisher in TIBCO Fulfillment Catalog**

The user can invoke the catalog publish workflow in TIBCO Fulfillment Catalog directly by using the sample SOAP web service requests available in the $AF_HOME/samples directory. The request can be sent using any standard SOAP client tools such as SOAPUI.

Specify the correct enterprise name, user name, and password in the request. Also specify the correct MASTERCATALOGNAME key and a PRODUCTID to publish the specific catalog. The request should be invoked against the running instance of TIBCO Fulfillment Catalog on the following URL:

```
http://<HOST>:<PORT>/eml/services/router/MasterCatalogRecordAction
```

Where:

HOST and PORT are the machine name and port number where TIBCO Fulfillment Catalog is deployed and running.

**Offline Catalog**

Offline catalog is a feature in TIBCO Fulfillment Order Management that enables it to have the context of the required data models in AOPD, OCV, & Orchestrator engines through the use of catalog XML files exported
from TIBCO Fulfillment Catalog. It reduces the dependency on TIBCO Fulfillment Catalog to be online at all the times. For details, refer Offline Catalog on page 52.
Configuration

This section describes the functions of TIBCO Fulfillment Order Management.

Third-party Product Requirement

Refer to the TIBCO® Fulfillment Order Management Installation and Configuration document for third-party product requirements.

OMS Architecture

The Order Management System (OMS) consists of the following four application components:

1. **OMS Server** - Core OMS server component, which provides SOAP-based web services over HTTP and JMS, JMS data interfaces, offline catalog web service and file polling interfaces, and REST APIs for OMS UI. This component also provides plan generation, orchestration, and jeopardy management capabilities.

2. **OMS UI** - provides Web based interface to browse orders and execution plans and perform actions on the orders.

3. **Dashboard** - web gadget based container for providing summary information about the order fulfillment engine. Current version provides four gadgets, namely Order Summary, Orders in Execution, backlog Order, and Amended Orders.

4. **AOPD** - This component provides plan generation capability in standalone mode.

All the above components are shipped as a WAR (.war) file.

It is essential for all the components in an application to work together and support the architecture.

OMS Configuration

**Messaging Configuration**

All the application components of Fulfillment Order Management use JMS as one of the conduits for passing a message within Fulfillment Order Management and with the external applications. The following properties are required to connect to the JMS server:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JNDI Connection factory JNDI Name</td>
<td>GenericConnectionFactory</td>
<td>JNDI Connection factory JNDI Name</td>
</tr>
<tr>
<td>Topic Connection factory JNDI Name</td>
<td>TopicConnectionFactory</td>
<td>Topic Connection factory JNDI Name</td>
</tr>
<tr>
<td>JNDI Initial Context Factory</td>
<td>com.tibco.jms.naming.TKeystoreInitialContextFactory</td>
<td>JNDI Initial Context Factory</td>
</tr>
<tr>
<td>JNDI URL</td>
<td>bjesnsaming/J109.97.118.21.7336</td>
<td>JNDI URL for JMS Service</td>
</tr>
<tr>
<td>JNDI Username</td>
<td>admin</td>
<td>JNDI Username</td>
</tr>
<tr>
<td>JNDI Password</td>
<td>password</td>
<td>JNDI Password</td>
</tr>
</tbody>
</table>

**Figure 12: Messaging Configuration**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>JMS provider host name</td>
</tr>
<tr>
<td>Port</td>
<td>Port number of the JMS Provider</td>
</tr>
<tr>
<td>Username</td>
<td>Username</td>
</tr>
<tr>
<td>Password</td>
<td>Password</td>
</tr>
</tbody>
</table>
Jeopardy deployment in the colocated mode will not require jeopardy queues and bridges for communication. The colocated mode ensures mutual communication of Orchestrator and Jeopardy components using low-level API calls, instead of regular queue and bridge communication.

If jeopardy queues and bridges have already been setup as a part of an earlier version, then you can use the JEOMS_DeleteEMSChannel.txt to remove the queues and bridges.

User Interface Configuration

OMS provides a web user interface to browse and perform actions on the orders and execution plans. OMS UI is deployed as a separate application in tomcat server and it requires parameters to connect to OMS Server application.

OMS does not support deploying OMS UI application and OMS Server application separately in different containers. OMS UI also provides configurable parameters to control the access to the application.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Session Per User</td>
<td>Number of sessions allowed per user. By default, OMS allows only one session to be created for any user. In the case of a scenario in which multiple users share user id and password, this value needs to be set appropriately.</td>
</tr>
<tr>
<td>Error If Maximum Session Expired</td>
<td>You can either expire the user's previous login or you can report an error when the user tries to log in again, preventing the second login. Note that if you are using the second approach, a user who has not explicitly logged out (but who has just closed the browser, for example) will not be able to log in again until original session expires.</td>
</tr>
</tbody>
</table>
| Http Session Fixation Protection | Session fixation vulnerabilities occur when the application authenticates a user without first invalidating the existing session ID, thereby continuing to use the session ID already associated with the user. The behavior can be controlled using the session-fixation-protection attribute which has the following three options:  
  - migrateSession - creates a new session and copies the existing session attributes to the new session. This is the default.  
  - none - Don’t do anything. The original session will be retained.  
  - newSession - Create a new, clean session without copying the existing session data. |
Router Configuration

The Content-based router in OMS allows routing of the order to the correct destination based on the contents of the order message.

Content-based routing schedules the order of the messages that are based on the actual content of the message itself, rather than by a destination specified by the message. Content-based routing works by opening a message and applying a set of rules to its content to determine the destination of a message. By freeing the sending application from the need to know anything about where an order should be routed for fulfillment, content-based routing provides a high degree of flexibility to configure multiple types of Orchestration engines.

OMS supports two types of routers:
1. Pass-through router (passthroughRouter), and
2. Filter based router (filteringRouter).

### Figure 14: Router Types and Properties

A pass-through does not apply any condition on the incoming order message and passes the message to the default Orchestrator.

The following table shows configurable parameters for Filter based Router.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Router Condition</strong></td>
<td>XPath filter condition to be applied on the incoming order message. If the XPath condition is not satisfied, the message will be routed to the default AF Orchestrator or else the message will be routed to the IPC destination.</td>
</tr>
<tr>
<td>AF Orchestrator Submit Order Queue</td>
<td>Destination queue name of Orchestrator for submit order request.</td>
</tr>
<tr>
<td>AF Orchestrator Amend Order Queue</td>
<td>Destination queue name of Orchestrator for amend order request.</td>
</tr>
<tr>
<td>AF Orchestrator Suspend Order Queue</td>
<td>Destination queue name of Orchestrator for suspend order request.</td>
</tr>
<tr>
<td>AF Orchestrator Activate Order Queue</td>
<td>Destination queue name of Orchestrator for activate order request.</td>
</tr>
<tr>
<td>iPC Orchestrator Submit Order Queue</td>
<td>Destination queue name of iPC for submit order request.</td>
</tr>
<tr>
<td>iPC Orchestrator Amend Order Queue</td>
<td>Destination queue name of iPC for amend order request.</td>
</tr>
<tr>
<td>iPC Orchestrator Suspend Order Queue</td>
<td>Destination queue name of iPC for suspend order request.</td>
</tr>
</tbody>
</table>
The functional support for TIBCO iProcess Conductor (iPC) has been deprecated in Fulfillment Order Management version 2.0.0. However, orders to be sent to iPC are still routed by OMS to the above listed iPC queues, if configured.

Filtering Router

A filter condition for a router is applied only for a submit order request. Any subsequent request related to the order will always be routed through the same orchestrator where the original submit order request was submitted. The XPath condition specified must be based on Order schema.

For example, XPath

/SubmitOrderRequest/orderRequest/header/udf[name='Orchestrator']/value/text() specified for a filter condition results in sending all the orders containing User Defined Field (UDF) with name value pair 'Orchestrator' and IPC' to TIBCO iProcess Conductor. It is not required to be based on the UDF element. The only requirement is that it should be a valid XPath condition on the order schema. Fulfillment Order Management supports specifying a filter condition but it does not perform any validation on the XPath condition. It must be validated before specifying it in the Fulfillment Order Management Configurator.

1. Any XPath filter condition results in the Orchestrator name. The following two Orchestrators are supported for routing:
   - AFO
   - IPC (iProcess Conductor)

2. Orchestrator names are case sensitive. The OMS router matches the text value to the Orchestrator name (case sensitive). If the text value is iPC, the order is routed to the AFO Orchestrator.

The router uses an XML configuration to configure the routing and mediation rules which are added to a router-context.xml of the OMS component. The path to the router-context.xml file is

<AF_HOME>/apache-tomcat-<version>/webapps/omsServer/WEB-INF/classes/spring.

The following example shows a router configuration to route the order message to Business Events (BE) Orchestrator based on the User Defined Field (UDF) value in the order message:

/SubmitOrderRequest/orderRequest/header/udf[name='Orchestrator']/value/text()

You must restart the OMS server to apply the updated XPath filter condition after updating the router information in AF Configurator.

Routing Orders to Other Engines

OMS routes orders to AFO and IPC.

To enable OMS to route an order to any other engine, perform the following steps:

1. Configure the router-context.xml file.
2. Change ConfigValues using AF Configurator.
Configuring Router Context

Router uses XML configuration to configure routing and mediation rules which are added to a router-context.xml of the OMS. The path to router-context.xml file is

<AF_HOME>/apache-tomcat-<version>/webapps/omsServer/WEB-INF/classes/spring.

To submit an order, consider an example with engine name as Fulfillment Provisioning (kpsa).

The following sections step you through the process of configuring the router context:

1. Create an instance of submit order router.
   a. Create a JMS template. To do this, make the following changes:
      
      ```
      <bean id="kpsaJmsTemplate" class="org.springframework.jms.core.JmsTemplate">
          <property name="connectionFactory" ref="kpsaConnectionFactory"/>
          <property name="destinationResolver" ref="beoDestinationResolver"/>
          <property name="pubSubDomain" value="false"/>
      </bean>
      
      a. Create a connection factory.
      
      ```
      
      ```
      <bean id="kpsaConnectionFactory" class="org.springframework.jms.connection.CachingConnectionFactory">
          <constructor-arg>
              <bean class="org.springframework.jndi.JndiObjectFactoryBean">
                  <property name="jndiName" value="GenericConnectionFactory"/>
                  <property name="jndiTemplate" ref="kpsaJndiTemplate"/>
              </bean>
          </constructor-arg>
          <property name="reconnectOnException" value="true"/>
          <property name="cacheProducers" value="true"/>
      </bean>
      ```
      
      b. Add a JNDI template.
      
      ```
      <bean id="kpsaJndiTemplate" class="org.springframework.jndi.JndiTemplate">
          <property name="environment">
              <props>
                  <prop key="java.naming.provider.url">tcp://${com.tibco.af.oms.jms.jndi.host}:${com.tibco.af.oms.jms.jndi.port} </prop>
                  <prop key="java.naming.factory.initial">${com.tibco.af.oms.jms.jndi.initialContextFactory}</prop>
                  <prop key="java.naming.security.principal">${com.tibco.af.oms.jms.jndi.security.principal}</prop>
                  <!-- <prop key="java.naming.security.credentials">${jms.jndi.security.credentials}</prop> -->
              </props>
          </property>
      </bean>
      ```
      
      b. Assign a destination queue.
      
      ```
      <bean id="kpsaSubmitOrderDestination" class="com.tibco.tibjms.TibjmsQueue">
          <constructor-arg value="${com.tibco.af.oms.router.destination.kpsaSubmitOrder}"/>
      </bean>
      ```
      
      c. Create an instance of Router Message Create.
      
      ```
      <bean id="kpsaSubmitMessage" class="com.tibco.aff.oms.router.RouterMessageCreator"/>
      ```
      
      d. Create an instance of Router.
      
      ```
      <bean id="kpsaSubmitOrderRouter" class="com.tibco.aff.oms.router.RouterProcessor">
          <property name="jmsTemplate"><ref local="kpsaJmsTemplate"/></property>
          <property name="destination"><ref local="kpsaSubmitOrderDestination"/></property>
          <property name="messageCreator"><ref local="kpsaSubmitMessage"/></property>
      </bean>
      ```
      
      e. Add an entry to router context for the required action.
      
      ```
      <routeContext id="routerContext" xmlns="http://camel.apache.org/schema/spring">
          <route id="orderSubmitRoute">
              <from uri="direct:tibco.aff.routerproxy.order.submit"/>
              <choice>
                  <when><xpath>$orchestrator = 'IPC'</xpath></when>
                  <to uri="bean:ipcSubmitOrderRouter" pattern="InOnly"/>
              </choice>
          </route>
      </routeContext>
      ```
2. Changes to ConfigValues using AF Configurator.
   a. Add a new property in the config values for the submit order destination according to 1.b (Assign a destination queue).

**Managing Application Security**

OMS provides two application-level security options.
- Default Authentication Provider

Default authentication provider is database-based security, which does not require configuration to use the default authentication provider. OMS uses the configured database to store the operational data of orders and execution plans.

![AF Order Management System > Application Security](image)

**Figure 16: OMS Application Security**

OMS also supports LDAP-based authentication.
The following properties are required to configure OMS to use external LDAP server authentication.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDAP Server URL</td>
<td>LDAPServer URL: <code>ldap://&lt;hostname&gt;:port/&lt;root context&gt;</code>. Many LDAP servers also support SSL-encrypted LDAPS, preferred for security purposes and to configure OMS to use SSL LDAP to connect to server use <code>ldaps://</code> at the beginning of the LDAP server URL.</td>
</tr>
<tr>
<td>LDAP User Manager DN</td>
<td>User Manager Distinguished Name to be used to connect to LDAP Server.</td>
</tr>
<tr>
<td>LDAP User manager Password</td>
<td>Password of the user manager to be used for authentication.</td>
</tr>
<tr>
<td>User Search Base</td>
<td>A search base (the distinguished name of the search base object) defines the location in the directory from which the LDAP user search begins.</td>
</tr>
<tr>
<td>User Search Filter</td>
<td>Search filter to be used to locate the user. For example, the following filter can be used to substitute the login name with value for the uid (filter) in the directory: <code>filter (uid={0})</code>.</td>
</tr>
<tr>
<td>Search Subtree</td>
<td>Flag to enable deep search through the sub tree of the LDAP Server URL + User Search Base. True by default.</td>
</tr>
<tr>
<td>Group Search Base</td>
<td>It defines the base DN under which the LDAP integration should look for one or more matches for the users’ DN. The default value performs a search from the LDAP root.</td>
</tr>
<tr>
<td>Group Search Filter</td>
<td>It defines the LDAP search filter used to match user’s DN to an attribute of an entry located under Group Search Base. The default value is <code>(uniqueMember={0})</code>.</td>
</tr>
</tbody>
</table>
| Group Role Attribute        | It defines the attribute of the matching entries, which will be used to compose the user’s role in OMS. Default value is `cn`. Attribute must have either admin or user as the value for the role attribute. Role-based authorization provided.
Managing Users and Roles

OMS supports role-based authorization. The user must belong to either ROLE_USER or ROLE_ADMIN. The following table shows business functions and a list of roles that are authorized to perform the business functions.

<table>
<thead>
<tr>
<th>OMS Interface</th>
<th>Function</th>
<th>Roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>OMS UI</td>
<td>Dashboard</td>
<td>ROLE_USER, ROLE_ADMIN</td>
</tr>
<tr>
<td></td>
<td>Search Order</td>
<td>ROLE_USER, ROLE_ADMIN</td>
</tr>
<tr>
<td></td>
<td>Order Detail</td>
<td>ROLE_USER, ROLE_ADMIN</td>
</tr>
<tr>
<td></td>
<td>Suspend Order</td>
<td>ROLE_ADMIN</td>
</tr>
<tr>
<td></td>
<td>Cancel Order</td>
<td>ROLE_ADMIN</td>
</tr>
<tr>
<td></td>
<td>Resume Order</td>
<td>ROLE_ADMIN</td>
</tr>
<tr>
<td></td>
<td>Amend Order</td>
<td>ROLE_ADMIN</td>
</tr>
<tr>
<td></td>
<td>Withdraw Order</td>
<td>ROLE_ADMIN</td>
</tr>
<tr>
<td></td>
<td>Search Execution Plan</td>
<td>ROLE_USER, ROLE_ADMIN</td>
</tr>
<tr>
<td></td>
<td>View Execution Plan</td>
<td>ROLE_USER, ROLE_ADMIN</td>
</tr>
<tr>
<td>OMS Web Service</td>
<td>Submit Order</td>
<td>ROLE_ADMIN</td>
</tr>
<tr>
<td></td>
<td>Synchronous Submit</td>
<td>ROLE_ADMIN</td>
</tr>
<tr>
<td></td>
<td>Order Amend Order</td>
<td>ROLE_ADMIN</td>
</tr>
<tr>
<td></td>
<td>Get Orders</td>
<td>ROLE_USER, ROLE_ADMIN</td>
</tr>
<tr>
<td></td>
<td>Get Order Detail</td>
<td>ROLE_USER, ROLE_ADMIN</td>
</tr>
<tr>
<td></td>
<td>Get Execution Plan</td>
<td>ROLE_USER, ROLE_ADMIN</td>
</tr>
<tr>
<td></td>
<td>Get Enriched</td>
<td>ROLE_USER, ROLE_ADMIN</td>
</tr>
<tr>
<td></td>
<td>Execution Plan</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cancel Order</td>
<td>ROLE_ADMIN</td>
</tr>
<tr>
<td></td>
<td>Activate Order</td>
<td>ROLE_ADMIN</td>
</tr>
<tr>
<td></td>
<td>Perform Bulk Order</td>
<td>ROLE_ADMIN</td>
</tr>
<tr>
<td></td>
<td>Action</td>
<td></td>
</tr>
</tbody>
</table>

By default, OMS provides a set of user id and password for accessing OMS through the web and for submitting a web service request.

The following table shows the default user id and password provided by the OMS.

<table>
<thead>
<tr>
<th>User Name</th>
<th>Password</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>admin</td>
<td>admin</td>
<td>ROLE_ADMIN</td>
</tr>
</tbody>
</table>
Mentioned below is the UserManagement application available with Fulfillment Order Management.

Before running the UserManagement application, set the environment variable `AF_OMS_CONTEXT_URL` to the URL where the omsServer Web application is up and running. For example:

```
http://<machineIPAddress>:<port>/omsServer
```

Where:
- **host** is the computer where you installed the Fulfillment Order Management.
- **port** number is the port number of the machine where OMS Web server is listening to requests. The default port number is **8080**.
- **omsServer** is the application context.

The UserManagement application is used to manage users in the Fulfillment Order Management solution. The application has a `userservice` command line interface that allows you to:

- Create User. Refer to [Create user](#).
- Read User Details. Refer to [Read the details of user](#).
- Delete User. Refer to [Delete the user](#).
- Reset User Password. Refer to [Reset the password of the user](#).

You must have role assigned as ROLE_ADMIN in order to invoke the command-line application.

Running the `userservice` with `-help` option displays all the options with `userservice` and corresponding arguments.

### Creating User

To create the user, do the following:

1. On the command prompt, access the `AF_HOME/bin` directory.
2. Run the following command:

   ```
   $AF_HOME/bin/userservice.sh -action create
   -adminusername <adminusername> -adminuserpassword <adminpassword>
   -username <user> -userpassword <password> -userrole <role>
   $./startup.sh
   ```

### Reading User Details

To read the user details, do the following:

1. On the command prompt, access the `AF_HOME/bin` directory.
2. Run the following command:

   ```
   $AF_HOME/bin/userservice.sh -action read
   -adminusername <adminusername> -adminuserpassword <adminpassword>
   -username <user>
   ```

### Deleting User

To delete the user, do the following:

1. On the command prompt, access the `AF_HOME/bin` directory.
2. Run the following command:

   ```
   $AF_HOME/bin/userservice.sh -action delete
   -adminusername <adminusername> -adminuserpassword <adminpassword>
   -username <user>
   ```

### Resetting Password

To reset the user password, do the following:

1. On the command prompt, access the `AF_HOME/bin` directory.
2. Run the following command:

```bash
$AF_HOME/bin/userservice.sh -action reset
-adminusername <adminusername>
-adminuserpassword <adminpassword>
-username <user> -newpassword <newpasswordtoreset>
```

If the passwords have special characters, then enclose them with double quotes or proper escape sequence compatible with the underlying operating system. For example: "welcome>123", "ab\'c", etc.

**Order Configuration and Validation**

OMS provides different configurable options to enable or disable the validation of an order request against configured offers in the system.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable Offer Configuration and Validation</td>
<td>This parameter specifies if OCV validation needs to be performed for incoming orders or not. Add flag name for creating OCV Validation = com.tibco.af.oms.OCVEnabled. This value can be set true or false through the configurator.</td>
</tr>
<tr>
<td>Offer Configuration and Validation Timeout (Milliseconds)</td>
<td>OMS makes an asynchronous call to the OCV application component. This field represents the timeout value in milliseconds. If a timeout occurs, OMS throws a SOAP fault to the client.</td>
</tr>
<tr>
<td>Offer Configuration and Validation Request Queue</td>
<td>Default request queue for OCV validation. Request queue name is tibco.aff.ocv.events.offer.validate.request.</td>
</tr>
<tr>
<td>Offer Configuration and Validation Response Queue</td>
<td>Default response queue for OCV validation response from OMS = tibco.aff.ocv.events.offer.validate.reply.oms. The response is sent to the Offer Configuration and Validation response queue for the validation request sent from OMS. For other OCV clients, the response is sent to the default queue = tibco.aff.ocv.events.offer.validate.reply.</td>
</tr>
<tr>
<td>OCV Exception Logging Enabled</td>
<td>Allows you to enable or disable logging of the exception stack trace in OMS when validation fails.</td>
</tr>
</tbody>
</table>
Load Balancing

OMS itself does not provide any load balancing capability but any third party load balancer can be used to load balance across multiple instances of OMS. For this, no specific configuration is required. The only requirement is that the load balancer should have support for sticky session. Sticky sessions means the load balancer will always direct a given client to the same back-end server.

*Hardware Load Balancer (HLB)*, which has Layer 7 capability, can direct traffic and maintain session persistence for Web applications without relying on the user's IP address with session cookies. Typically, HLB inserts a cookie that the load balancer creates and manages automatically to remember which back-end server a given HTTP connection should use. Then it would always direct the request originating from that client browser to the same server.

Some of the HLBS which support layer 7 capability include:

- Barracuda
- jetNEXUS
- f5

OMS WebService

OMS supports both HTTP and JMS as transport protocols for invoking SOAP-based web services. Order services in OMS can be secured by enabling username token-based security. OMS supports the WS-Security Username Token mechanism, which allows for the sending and receiving of user credentials in a standards-compliant manner. The Username token is a mechanism for providing credentials to a Web service where the credentials consist of the UserName and Password. The password must be passed in clear text.

The UserName token mechanism provides a web service with the ability to operate without having the username and password in its URL or having to pass a session cookie with the HTTP request.

The following is a sample of the UserName token showing the username and password:

```
<soapenv:Header>
<wsse:Security soapenv:mustUnderstand="1"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
<wsse:UsernameToken>
<wsse:Username>admin</wsse:Username>
<wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText" admin</wsse:Password>
<wsse:Nonce>WScqanjCEAC4mQoBE07sAQ==</wsse:Nonce>
<wsu:Created>2010-05-11T01:24:32Z</wsu:Created>
</wsse:UsernameToken>
</wsse:Security>
</soapenv:Header>
```

The table below shows configurable properties for order related WebServices in OMS.

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable UserName token based Security</td>
<td>true</td>
</tr>
<tr>
<td>Enable Schema validation</td>
<td>true</td>
</tr>
<tr>
<td>Enable Order Receiver Idempotency</td>
<td>true</td>
</tr>
<tr>
<td>HTTP Channel type</td>
<td>http</td>
</tr>
<tr>
<td>HTTP Port Number</td>
<td>18080</td>
</tr>
<tr>
<td>HTTPS Port Number</td>
<td>18443</td>
</tr>
<tr>
<td>Use external business transactionId as business transactionId</td>
<td>false</td>
</tr>
</tbody>
</table>

Figure 19: OMS WebService
<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable User Token based Security</td>
<td>The Order service provided by OMS can be secured by enabling username token-based security.</td>
</tr>
<tr>
<td>Enable Schema Validation</td>
<td>Defines a flag to specify if schema validation is required on the order requests submitted to OMS.</td>
</tr>
<tr>
<td>Enable Order Receiver Idempotency</td>
<td>Making the order Web services idempotent allows the client to submit orders with the same order reference multiple times without any side effects. The web service detects duplicate orders and responds with the same response for all the submission(s).</td>
</tr>
<tr>
<td>HTTP Channel Type</td>
<td>Defines channel type to be used for the transport. Specifying channel type to be HTTPS allows the client and server to use mutual authentication and encrypts the communication.</td>
</tr>
<tr>
<td>HTTP Port Number</td>
<td>Port number of HTTP Channel. This port number must match the port number specified for the tomcat server http port transport.</td>
</tr>
<tr>
<td>HTTPS Port Number</td>
<td>Port number of HTTPS Channel. This port number must match the port number specified for the tomcat server HTTPS transport.</td>
</tr>
</tbody>
</table>

**Collecting Order Summary Data**

OMS provides the Dashboard component to view summary information about the fulfillment engine. This requires collection of information about status orders in the system and other summary information. In the case of a heavy load, it is desirable to collect the summary information periodically by scheduled interval rather than updating summary data on every status change. Cron Expression is used to define the time interval for summary collection.

For example, Cron expression "0 0 ** * ?" defines an hourly time interval for the summary collection.

Order Summary Collection can be scheduled to run in off peak hours and not affect the order processing. Please look at highlighted values to be changed in ConfigValues_OMS.xml. The following configuration Order Summary runs every day at 10:15 a.m. Change it appropriately as per your environment and requirement.

```xml
<Category description="Order Summary Data Collection Method" name="Order Summary Data Collection Method" visibility="Basic">
  <Item name="onStatusChange">
    <ConfValue description="Order Summary collection on every order status change" isHotDeployable="true" name="Order summary Data Collection" propertyName="com.tibco.af.oms.summaryDataCollectionMethod" readonly="true" sinceVersion="1.1" visibility="Basic">
      <ConfString default="onStatusChange" value="onStatusChange"/>
    </ConfValue>
  </Item>
  <Item name="Scheduled">
    <ConfValue description="Order Summary collection on scheduled interval" isHotDeployable="true" name="Order summary Data Collection" propertyName="com.tibco.af.oms.summaryDataCollectionMethod" readonly="true" sinceVersion="1.1" visibility="Basic">
      <ConfString default="scheduled" value="scheduled"/>
    </ConfValue>
    <ConfValue description="schedule interaval for order summary data collection." isHotDeployable="true" name="Cron Expression" propertyName="com.tibco.af.oms.summaryDataCollection.scheduled.cronExpression" sinceVersion="1.1" visibility="Basic">
      <ConfString default="0 0 ** * ?" value="0 15 10 ** ?"/>
    </ConfValue>
  </Item>
</Category>
```
Audit Trail

Audit trail can be enabled/disabled with this version using the configuration parameter in `ConfigValues_OMS.xml` (Miscellaneous section):

```xml
<ConfValue description="Enable Audit Trail Entries" name="Audit Trail"
propname="com.tibco.af.oms.AuditTrailEnabled" sinceVersion="1.1" visibility="Basic">
  <ConfBool default="true" value="false"/>
</ConfValue>
```
Jeopardy Management System

The Jeopardy Management System consists of the following components:

- **Jeopardy Management Server** - Monitors execution plans and detects jeopardy conditions.
- **Notification Engine** - Evaluates user configured rules and sending notifications to user defined destinations.

The following figure depicts the component architecture of the Jeopardy Management System:

![JeOMS Component Architecture](image)

**Figure 20: JeOMS Component Architecture**

**JeOMS Configuration**

All the parameters of JeOMS have default values and no parameter value required to deploy and run JeOMS in basic mode. You may change the following parameters to tune the performance of the Jeopardy Management System.

**Jeopardy Management Tuning Parameters**

The Jeopardy Management Tuning Parameters are as follows:

**Risk Threshold**

Set this parameter to specify percentile increase over typical duration beyond which a task can be considered to be running in a hazard region. For example, if typical duration of a process component is 60 minutes and if risk threshold is 50, any plan item that is executing for more than 90 minutes is considered to be running in a hazard region. Default value is 25 percentile point.

**Out of Scope Threshold**

Set this parameter to specify Percentile increase over maximum duration beyond which an execution can be considered running out of scope for jeopardy detection. And no further jeopardy monitoring be performed on the plan. For example, if maximum duration of a process component is 60 minutes and if out of scope threshold is 100, any plan item that is executing for more than 120 minutes is considered as out of scope and JeOMS stops monitoring the plan.

**Time Window Value/Time Window Unit**
The Configuration parameter \textit{Time window value} and \textit{Time Window Unit} are combined together to determine the \textit{Time window} or \textit{Chronon} to which plan items of execution plans belong in JeOMS. It is a user-defined interval of time during which any execution of any plan item would result in grouping the plan item together. Configure this value based on execution characteristics of process components in fulfillment ecosystem. This value is used by jeoMS to cluster plan items in execution to optimize jeopardy detection cycle.

![Diagram](image)

**Figure 21: Time Window Value/Time Window Unit**

For example, if fulfillment tasks executed by the process components typically take few days to complete, it is probably meaningless to use seconds or milliseconds to track the progress of a plan item. In this case, the you can provide a value in terms of hour units. On the other hand, if JeOMS monitors very fast executing plans, JeOMS will require a Chronon of milliseconds or seconds to accurately predict the jeopardy condition. The default value is 10 Minutes.

**Messaging Configuration**

JeOMS piggyback on message configuration parameters of OMS and no additional configuration required.

**Datasource Configuration**

JeOMS piggyback on data source configuration parameters of OMS and no additional configuration required.

**Colocated Jeopardy Mode**

Colocated Jeopardy mode will only support "Cache" as the second level data store. The existing data store options like File and ActiveSpace will not be supported.

Two new tables have been introduced in this version as a part of the Cache data store. The tables are:

1. \textit{TIME\_WINDOW}
2. \textit{DAMPENING\_CRITERIA}

For more details see Database Schema on page 98
Offline Catalog

The Offline Catalog feature enables TIBCO Fulfillment Order Management to have no dependency on TIBCO Fulfillment Catalog to be online all the time for the data models.

All the required catalogs viz PRODUCT, CUSTOMER, SEGMENT, ACTION, and PLANFRAGMENT can be published out of TIBCO Fulfillment Catalog in the form of flat XML files. TIBCO Fulfillment Order Management can then be configured to read the catalogs from these XML files instead of publishing them on JMS topics from TIBCO Fulfillment Catalog as per the online mode.

Accessing Offline Catalog

Offline catalog mode can be enabled for each catalog separately using Fulfillment Order Management configurator.

For example, only Product catalog can be configured to use offline mode and the remaining catalogs to use the online mode.

The following figure shows the offline catalog configuration for the PRODUCT catalog. However, the configurations for other catalogs can be done in the similar way.

Figure 22: Offline Product Catalog Configuration

The offline catalog can be used in one of the following three ways:

1. Reading from the master directories to process the catalog publish requests sent by AOPD, OCV and Orchestrator engines on their startup. The master directory set for PRODUCT catalog is shown bordered in green color in the above screen cap. The model data contained in the xml files under the master directory will be read and published on the JMS topics for the respective engines.

   Once read, the files in the master directory will NOT be moved. Just a copy of the original file will be created under master success or master failure directory based on whether the ‘read & publish’ operation is successful or failed.

2. By using the polling mechanism to poll the non-master directories. The non-master directory set for PRODUCT catalog is shown bordered in red color in the above screen cap. The directory is polled according to the polling interval as shown in the following configuration figure:

   Figure 23: Offline Polling Configuration

3. By sending the SOAP web service request to read one or more catalogs from non-master directories on demand. The web service request will also be processed by reading the XML files under the non-master directory set.
Offline catalog invocation web service WDSL is

\$AF_HOME/schemas/wsdl/http/OfflineCatalogue.wsdl.

The following sample request can be sent on the URL

http://<HOST>:<PORT>/omsServer/api/offlineCatalogueWS:

```xml
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:off="http://www.tibco.com/AFF/OfflineCatalogue">
    <soapenv:Header/>
    <soapenv:Body>
        <off:OfflineCatalogRequest>
            <off:ActionOffline>?</off:ActionOffline>
            <off:ProductOffline>?</off:ProductOffline>
            <off:CustomerOffline>?</off:CustomerOffline>
            <off:SegmentOffline>?</off:SegmentOffline>
            <off:PlanFragmentOffline>?</off:PlanFragmentOffline>
        </off:OfflineCatalogRequest>
    </soapenv:Body>
</soapenv:Envelope>
```

Once read, the files in the non-master directory are removed. A copy of the original file will be created under non-master success or non-master failure directory based on whether the 'read and publish' operation is successful or not.

If a new file is added in a non-master directory for loading the models on demand, the same file must be synched up in the corresponding master directory. This is to ensure that the updated models are published to the BusinessEvents engines on their next restart.

**Flat File Format**

The offline catalog XML files have the following format:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<jms1:ActivityOutput
xmlns:jms1="http://www.tibco.com/namespaces/tnt/plugins/jms">
    <OtherProperties>
        <CURRENTBATCH>1</CURRENTBATCH> -- Current batch number.
        <TOTALBATCH>12</TOTALBATCH> -- Total catalog model present in repository.
    </OtherProperties>
    <Body>
        Actual Catalog Model Data in XML format as per Fulfillment Order Management schema.
    </Body>
</jms1:ActivityOutput>
```
Logging

Logging is used to record information about a program's execution. This information is typically used for debugging purposes, and additionally, depending on the type and detail of information contained in a trace log, to diagnose common problems with software.

How Logging Works

There are the following types of logging for every component:

- **Local Logging:** This refers to writing log output to a local log file for every component.
- **Central Logging:** This refers to publishing log messages from individual components to a central location where logs are collated and logged to a central log file.

The individual components publish log messages to a central queue and then the central log server picks up the log messages from this queue.

The advantage of having a central logging framework is that different individual components can log the data in an agreeable, common format and in a common location, which makes it easy to correlate the log records. This also helps you analyze the records more easily and effectively.

The logging can be effective in the following ways:

- **Standardizes the contents** of a log message for logging data across all the components.
- Provides with an ability to **collect the log messages in a central location** in addition to being available on an individual engine.
- Empowers **the correlation of different log messages** coming across from various components.
- Enables **filtering of messages** for effective analysis of the logs.

![Central Logging Diagram](image)

Figure 24: Central Logging

Contents of the Log Message

The log message is comprised of several log components that are required to explain the log message in its entirety. These log message components help you analyze the log.

<table>
<thead>
<tr>
<th>Log Message Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log level</td>
<td>One of the levels - DEBUG, INFO, WARN, ERROR, OFF.</td>
</tr>
<tr>
<td>BusinessTransactionID</td>
<td>Useful for correlating messages from various components.</td>
</tr>
<tr>
<td>OrderRef</td>
<td>An Identifier to identify the Order for which this log message is written.</td>
</tr>
</tbody>
</table>
Configuring Logging

Different configuration options are supported for different components.

This topic describes:

- BE Component - OCV
- Java Components

Configuring Logging for OCV

Logging configuration for BE components applies to the OCV component.

Use the following global variables to configure logging for OCV:

**CommonServicesClientLib/Events/logLevel**: This variable sets the logging level at which OCV logs are written to the local log file. The default value is set to 1 (INFO). It can be changed to 2, 3, 4, or maximum 5 to have additional logs at these levels. The logging is disabled if this variable is set to 0 or a negative number.

Configuring Logging for Java Components

Logging configuration for Java components applies to omsServer, Orchestrator, AOPD, omsUi, and jeoms.

Logging is done using Log4j framework. Each component has a separate file for Log4j configurations as explained below.

**OMS Server and Orchestrator**: `$AF_CONFIG_HOME/OMSServerLog4j.xml` file is used to configure logging for OMS Server and Orchestrator which is an integral component.

- The default logLevel is:
  - INFO for `com.tibco.aff` package and its subpackage
  - ERROR for all other packages
- Default publishLogLevel (for central logging)
  - WARN (Look for "JMSAppender" and the following entry within it `<param name="Threshold" value="WARN"/>")
- Local log file used by omServer and Orchestrator is omsServer-local.log. Published logs will go into central.log file same as for the other components.

**OMS UI**: `$AF_CONFIG_HOME/OMSUILog4j.xml` file is used to configure logging for OMS UI.

- The default logLevel is:
  - INFO for `com.tibco.aff` package and its subpackage
  - ERROR for all other packages
- Local log file used by omUi component is omsui-local.log.
- OMS UI does not send messages to the central log server.

**JeOMS**: `$AF_CONFIG_HOME/OMSServerLog4j.xml` file is used to configure logging for JeOMS component.
The default logLevel is:
- INFO for com.tibco.aff.jeoms and com.tibco.aff.eca.engine packages and their subpackages
- ERROR for all other packages

Default publishLogLevel (for central logging)
- INFO (Look for "JeomsJMSAppender" and the following entry within it `<param name="Threshold" value="INFO"/>`)

There is no local log file for JeOMS. The Tomcat console logs will go into catalina.out. Published logs will go into central.log file same as for the other components.

AOPD: $AF_CONFIG_HOME/AOPDLog4j.xml file is used to configure logging for AOPD component.

- The default logLevel is:
  - DEBUG for com.tibco.aff package and its subpackages
  - ERROR for all other packages

- Default publishLogLevel (for central logging)
  - WARN (Look for "JMSAppender" and the following entry within it `<param name="Threshold" value="WARN"/>`)

- Local log file used by AOPD in case of standalone deployment is aopd-local.log and in case of collocated deployment mode is omsServer-local.log. Published logs will go into central.log same as for the other components.

**Controlling Log Levels**

You can control logging using the configuration options before the engines are brought up. However, it may be required that the log levels be changed dynamically at runtime without bringing down the engines.

- **Controlling Logging for Java Components** on page 56

**Controlling Logging for Java Components**

Logging can be controlled using the configuration options mentioned in the Configuring Logging for Java Components on page 55 before the engines are brought up.

However, it may be required that the log levels be changed dynamically at runtime without bringing down the engines.

This ability to change logging levels at runtime is provided for various components. For OMS, it is as follows (JMS-based configuration is provided to change log levels at runtime):

1. Go to Tools -> Change OMS Log Level.
2. To change the log level of all `com.tibco.aff` loggers to ERROR, provide the following:

<table>
<thead>
<tr>
<th>OMS Log Level</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package</td>
<td><code>com.tibco.aff</code></td>
</tr>
<tr>
<td>Level</td>
<td><code>ERROR</code></td>
</tr>
<tr>
<td>JMX Host</td>
<td><code>localhost</code></td>
</tr>
<tr>
<td>JMX Port</td>
<td><code>100999</code></td>
</tr>
<tr>
<td>Username</td>
<td><code>admin</code></td>
</tr>
<tr>
<td>Password</td>
<td><code>adminpassword</code></td>
</tr>
</tbody>
</table>

To turn off logging, provide the Level as `OFF` for that package.
Chapter 3

Administration Tasks

Topics

- Order Purging
- Product Model Purge
- Bulk Order Actions
Order Purging

Purging is an act of removing undesired elements in the order information. Order purging enables you to remove old order information from the fulfillment system such as already fulfilled or delivered (met the delivery date line) orders, or an order replaced to accomplish a purpose and function effectively.

Order purging removes an order’s data from the OMS and Orchestrator components by allowing you to provide certain criteria in order to purge the unwanted data. You can track the purge progress by retrieving the status of your purge request.

You can purge orders based on the following criterion:

**Purge All Orders Within a Given Period of Time (Absolute Date/Time Range) and Matching Order Type (Order Type Match Parameter)**

You can delete the order data from the OMS for all the orders submitted within a given period of time with a specific order type. For example, you can initiate a request to delete all the order data from the OMS database and Orchestrator between 01/10/2010 09:30:00 and 10/10/2011 18:30:00 and having order type as 'COMPLETE'.

Valid values for order type are:
- COMPLETE
- CANCELLED

Serve restart is not required to purge order data.

Perform the following to initiate or track the purge process:
- Edit the `${AF_HOME}/bin/purgeorder.properties` file to specify the purge criteria and the choice of operation.
- Execute the `${AF_HOME}/bin/purgeorderservice.sh` shell script.

**WSDL Location**

This is the default location where all the WSDL files are copied after the installation.
- `${AF_HOME}/schemas/wsdl/http/PurgeOrderServiceHTTP.wsdl`

**Purge Configuration**

You can configure the required purge settings through the Configurator.

The purge configuration details are provided in the following sections:
- Archiving Database on page 60
- PurgeThreadCount on page 61

**Archiving Database**

You can archive the purged data by setting and configuring an archive database in Configurator.

1. Create a new user/schema similar to the one used for OMS. This can be done through Configurator or manually. For details, see Creating and Configuring Database section in the TIBCO Fulfillment Order Management Installation and Configuration guide on how to create a new user and schema specific to OMS.
2. Configure the newly created schema through Configurator as displayed:
3. Execute the steps mentioned in the Purging Orders Between Given Time Period and Matching Order Type section. Before you execute the shell script, in the `purgeorder.properties` file, set the archive value to `true`.

**PurgeThreadCount**

The purge thread count can be configured through the Configurator.

You must restart the OMS server after setting the property.

Set the Purge Thread Count Property as displayed:
Purging Orders Within a Given Time Period and Matching Order Type

Use this feature of the purge utility to clean-up your database by removing old or irrelevant orders within any given period of time and having a particular order type.

Procedure

1. Go to the command prompt.
2. Change the directory to `${AF_HOME}/bin`.
3. Open the `purgeorder.properties` file using any text editor to edit the purge input parameters.
4. Type 1 as your choice of operation.
5. Type the START_DATE in the `<MM-DD-YYYY'T'HH:MM:SS>` format. For example, if the start date is March 21 2014 and time is 9:30 am then the format is:
   
   ```
   START_DATE=03-21-2014T09:30:00
   ```

6. Type the END_DATE in the `<MM-DD-YYYY'T'HH:MM:SS>` format. For example, if the start date is March 21 2014 and time is 6:30 pm then the format is:

   ```
   END_DATE=03-21-2014T18:30:00
   ```

   END_DATE must be greater than START_DATE.

7. Type the ORDER_TYPE. For example, COMPLETE.

   ```
   ORDER_TYPE=COMPLETE
   ```

   You must specify valid order type. For example, either COMPLETE or CANCELLED. Invalid order type may result in execution error.

8. Type the ARCHIVE. For example, false.

   ```
   ARCHIVE=false
   ```

9. Save the file.
10. Execute the `purgeorderservice.sh` shell script.
Retrieving Purge Status

You can find the status of your purge request by using the purge ID returned in response to invoking the purge operation.

Procedure

1. Go to the command prompt.
2. Change the directory to `${AF_HOME}/bin`.
3. Open the `purgeorder.properties` file using any text editor to edit the purge input parameters.
4. Type 2 as your choice of operation.
5. Type the PURGE_ID. For example, 518c1fa1-a109-4f80-af69-0c53cb1c6350.

   ```
   PURGE_ID=518c1fa1-a109-4f80-af69-0c53cb1c6350
   ```
6. Save the file.
7. Execute the `purgeorderservice.sh` shell script.
   The purge status is visible on the console.

Retrieving Purge Estimate

You can retrieve the estimated number of orders to be purged, between a specific date and time range, by invoking the purge estimate operation.

1. Go to the command prompt.
2. Change the directory to `${AF_HOME}/bin`.
3. Open the `purgeorder.properties` file using any text editor to edit the purge input parameters.
4. Type 3 as your choice of operation.
5. Type the start date in the `<MM-DD-YYYY'T'HH:MM:SS>` format. For example, if the start date is March 21 2014 and time is 9:30 am then the format is:

   ```
   START_DATE_3=03-21-2014T09:30:00
   ```
6. Type the end date in the `<MM-DD-YYYY'T'HH:MM:SS>` format. For example, if the start date is March 21 2014 and time is 6:30 pm then the format is:

   ```
   END_DATE_3=03-21-2014T18:30:00
   ```
7. Type the order type. For example, `COMPLETE`.

   ```
   ORDER_TYPE_3=COMPLETE
   ```

   You must specify valid order type. For example, either `COMPLETE` or `CANCELLED`. Invalid order type may result in execution error.
8. Save the file.
9. Execute the `purgeorderservice.sh` shell script. The estimated number of orders to be purged, within the specified date and time range, will be displayed.
The Purge web service makes it possible to remove existing products from model repository in AOPD and models persisted in OMS database.

### Purge Request

```xml
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:off="http://www.tibco.com/AFF/OfflineCatalogue">
  <soapenv:Header/>
  <soapenv:Body>
    <off:PurgeRequest ExternalBusinessTransactionID="?">
      <!--You have a CHOICE of the next 2 items at this level-->
      <!--off:all>all</off:all-->
      <off:productId>Product_TV,PO_TV</off:productId>
    </off:PurgeRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

### Purge Response

```xml
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:PurgeResponse ExternalBusinessTransactionID="?"
                        xmlns:ns8="http://www.tibco.com/aff/orderservice/result"
                        xmlns:ns7="http://www.tibco.com/aff/planfragments"
                        xmlns:ns6="http://www.tibco.com/aff/plan"
                        xmlns:ns5="http://www.tibco.com/aff/commonTypes"
                        xmlns:ns4="http://www.tibco.com/aff/order"
                        xmlns:ns3="http://www.tibco.com/AFF/OfflineCatalogue">
      <ns2:purgeResult>
        <ns2:ProductID>Product_TV,PO_TV</ns2:ProductID>
        <ns2:Message>Purge Product request sent Successfully to AOPD</ns2:Message>
      </ns2:purgeResult>
    </ns2:PurgeResponse>
  </soap:Body>
</soap:Envelope>
```

### WSDL Location

This is the default location where all the WSDL files are copied after the installation.

- `$AF_HOME\schemas\wsdl\http\OfflineCatalogue.wsdl`
Bulk Order Actions

Operations on an order are performed depending upon the requirement. Performing the same action on individual orders is difficult and time-consuming. You can apply actions to the group of orders simultaneously using Bulk Order action.

The following operations can be performed on the group of orders:
• CANCEL
• SUSPEND
• RESUME
• WITHDRAW

These operations are exposed by the OMS order service.

Importance of Bulk Order Actions

The bulk order actions allow administrators to cancel, suspend, resume, or withdraw a group of orders in a single invocation of a web service. This is useful:
• To perform a specific action on all orders in a particular region.
• To prevent repetitive intervention to perform similar actions.

The bulk order actions are based on the existing OMS order service. This operation is called PerformBulkOrderAction.

The existing OMS order service is modified to include a new operation. This operation allows you to specify the type of action to be performed along with the group of orders on which the action must be performed.

You can monitor the request status through:
• Event log - contains the information about the status of the request.
• Order lists - shows the change in the order status when refreshed.

All the errors that occur during this process are logged and handled individually.

WSDL Location

This is the default location where all the WSDL files are copied after the installation.
• $AF_HOME\schemas\wsdl\http\OrderServiceHTTP.wsdl
• $AF_HOME\schemas\wsdl\jms\OrderServiceJMS.wsdl

Error Codes

The following table lists the error codes:

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIBCO-AFF-OMS-100046: INVALID_ACTION</td>
<td>Web service fault code for invalid values of action.</td>
</tr>
<tr>
<td>TIBCO-AFF-OMS-100047: NO ORDERS FOUND</td>
<td>Web service fault code when neither order id nor order reference is present in the request.</td>
</tr>
<tr>
<td>TIBCO-AFF-OMS-100048: BOTH ORDERID AND ORDERREF FOUND</td>
<td>Web service fault code when both order id and order reference are present in the request.</td>
</tr>
</tbody>
</table>
## Invoking Bulk Order Operation

The **PerformBulkOrderAction** bulk order operation requires the following input parameters to perform the selected action on all the orders contained in the request:

- Action type
- List of order IDs or order refs

The **PerformBulkOrderAction** bulk order operation is an asynchronous operation and the consumer of the operation receives an acknowledgment immediately upon the submission of the request. This acknowledgment is not an indication that the process is complete. This indicates that the request is under process by the OMS component. For tracking the status of individual orders, activity logs, or an order, use the browser screens of the Fulfillment Order Management UI. The operation can be invoked by a user with ADMIN role only.

A user with an ADMIN role can invoke the bulk order actions.

The property `Use external business transactionId as business transaction id within OMS` in the Configurator should be set to false while invoking bulk cancel operation.

## Tracking the Request Status

The request status for the invoked bulk order action can be tracked using:

- Fulfillment Order Management UI (Dashboard, Order Screen and Activity logs)
- Logs in the OMS and Orchestrator components

## Logging

TIBCO Fulfillment Order Management provides detailed logging and auditing capabilities to identify the system errors, and key errors that can be gracefully handled by the calling system.

For bulk order actions, the logging is done using the AFFLogger APIs and the log file (`omsServer-local.log`) is created in the corresponding location based on the configured Appender. The log location is `$AF_HOME/apache-tomcat-7.0.42/logs`. The incoming bulk order request is validated and an INFO level log is generated. The log contains the action to be performed along with the number of orders in the request.

For all bulk order actions, if a particular order is not found in the OMS component, an 'ERROR' level log is generated indicating that the order was not found.

## Schema

A schema is an organization or structure for the **PerformBulkOrderAction** bulk order actions web service.

### Bulk Order Schema

The following figure depicts the bulk order action operation added to the OMS order service.
Figure 29: Bulk order action operation added to the OMS order service

Bulk Orders Operation Request Schema
The following figure depicts the bulk operation request schema.

Figure 30: Request schema for bulk operation

Bulk Orders Operation Response schema
The following figure depicts the bulk operation response schema.

Figure 31: Response schema for bulk operation

Sample Request
The sample request applicable to the bulk operation is as follows:

```
  <soapenv:Header/>
  <soapenv:Body>
    <ord:PerformBulkOrderActionRequest businessTransactionID="bTranID">
      <ord:action>SUSPEND</ord:action>
      <ord:orderID>ddb74135-ee66-4eb4-8aaa-4a842ae23510</ord:orderID>
      <ord:orderID>567db5aa-06d3-44e4-abf1-c6f06a8fb110</ord:orderID>
      <ord:orderID>2611964b-5648-4b00-82c3-c2acdefe5240</ord:orderID>
      <ord:orderID>30015c77-b4fe-44d6-9eb4-0d7a433e49e4</ord:orderID>
      <ord:orderID>031c0e65-859d-4a58-8089-f9c25e5f2244</ord:orderID>
      <ord:orderID>b57da5dc-7b76-49c7-9fd6-9af3234be4d9</ord:orderID>
      <ord:orderID>27637077-4c7d-4748-bf47-3c43cf7ce640</ord:orderID>
      <ord:orderID>6716b0aa-a617-470a-9d7d-bbfc38dce71</ord:orderID>
    </ord:PerformBulkOrderActionRequest>
  </soapenv:Body>
</soapenv:Envelope>
```
Sample Response

The sample response applicable to the bulk operation is as follows:

```xml
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns3:PerformBulkOrderActionResponse xmlns="http://www.tibco.com/aff/order"
      xmlns:ns2="http://www.tibco.com/aff/common/types"
      xmlns:ns3="http://www.tibco.com/aff/orderservice"
      xmlns:ns4="http://www.tibco.com/aff/orderservice/result"
      xmlns:ns5="http://www.tibco.com/aff/plan"
      xmlns:ns6="http://www.tibco.com/aff/plantemplates">
      <ns3:timestamp>2012-08-01T15:36:54.166+05:30</ns3:timestamp>
      <ns3:message>Request Submitted Successfully</ns3:message>
      <ns3:noOfOrders>10</ns3:noOfOrders>
    </ns3:PerformBulkOrderActionResponse>
  </soap:Body>
</soap:Envelope>
```
Chapter 4

Tuning

Topics

- Tuning Performance
There have been significant performance improvements to OMS. The queue for order submission has multiple concurrent listeners to process incoming orders in parallel. The queues for receiving status notifications are separated for the different types of notifications such as Order, OrderLine, Plan, PlanItem, or OrderAmendment (See figure Figure 32: Listener Queues on page 70). Each of these queues supports multiple concurrent listeners for processing the notifications in parallel. The number of concurrent listeners for the various queues are configurable, and can be configured using the Fulfillment Order Management Configurator User Interface.

For details on the properties of the Listener Queues, see Status Listener Queues on page 32.

In addition, the number of concurrent listeners for JMS interfaces from AFI, AFS, and TDS components merged as the integral part of OMS server can also be configured as per the need.
<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetPendingOrder receiver queue</td>
<td>tibco:aff.tds.order.request</td>
<td>GetPendingOrder receiver queue</td>
</tr>
<tr>
<td>GetPendingOrder receiver count</td>
<td>3</td>
<td>GetPendingOrder receiver count</td>
</tr>
<tr>
<td>GetPendingOrder receiver dead queue</td>
<td>tibco:aff.tds.order.request</td>
<td>GetPendingOrder receiver dead queue</td>
</tr>
<tr>
<td>GetPendingOrder receiver dead counter</td>
<td>3</td>
<td>GetPendingOrder receiver dead counter</td>
</tr>
<tr>
<td>GetPlan/GetRandom Request receiver queue</td>
<td>tibco:aff.tds.plan.request</td>
<td>GetPlan/GetRandom Request receiver queue</td>
</tr>
<tr>
<td>GetPlan/GetRandom Request receiver count</td>
<td>3</td>
<td>GetPlan/GetRandom Request receiver count</td>
</tr>
<tr>
<td>GetPlan/GetRandom Request receiver dead queue</td>
<td>tibco:aff.tds.plan.request</td>
<td>GetPlan/GetRandom Request receiver dead queue</td>
</tr>
<tr>
<td>GetPlan/GetRandom Request sender queue</td>
<td>tibco:aff.tds.plan.reply</td>
<td>GetPlan/GetRandom Request sender queue</td>
</tr>
<tr>
<td>GetPlan/GetRandom Request sender counter</td>
<td>3</td>
<td>GetPlan/GetRandom Request sender counter</td>
</tr>
<tr>
<td>GetPlan/GetRandom Request sender dead queue</td>
<td>tibco:aff.tds.plan.request</td>
<td>GetPlan/GetRandom Request sender dead queue</td>
</tr>
<tr>
<td>SetPlan/SetRandom Request receiver queue</td>
<td>tibco:aff.tds.plan.request</td>
<td>SetPlan/SetRandom Request receiver queue</td>
</tr>
<tr>
<td>SetPlan/SetRandom Request receiver count</td>
<td>3</td>
<td>SetPlan/SetRandom Request receiver count</td>
</tr>
<tr>
<td>SetPlan/SetRandom Request receiver dead queue</td>
<td>tibco:aff.tds.plan.request</td>
<td>SetPlan/SetRandom Request receiver dead queue</td>
</tr>
<tr>
<td>SetPlan/SetRandom Request sender queue</td>
<td>tibco:aff.tds.plan.reply</td>
<td>SetPlan/SetRandom Request sender queue</td>
</tr>
</tbody>
</table>

Figure 33: AFI, TDS, AFS Configuration
Chapter 5

Fulfillment Order Management Disaster Recovery

This topic covers all details related to TIBCO® Fulfillment Order Management Disaster Recovery considerations.

Topics

- Fulfillment Order Management Topology for Disaster Recovery
- Storage and Volumes
- Network
- Database
- Messaging Server
- ActiveSpace
Fulfillment Order Management Topology for Disaster Recovery

FOM recommends Symmetric topology for disaster recovery. FOM configured for disaster recovery must be completely identical across tiers on the production site and standby site. In a symmetric topology, the production site and standby site have the identical number of hosts, load balancers, FOM component instances, and process components. The same ports are used for both sites. The systems are configured identically and the applications access the same data.

This document covers guidelines for planning and deploying FOM for disaster recovery and application continuous availability. This document doesn’t provide guidelines of deploying process components and north bound FOM client application for disaster recovery.
Storage and Volumes

Create the volumes on your storage device and mount them appropriately on your FOM nodes. These volumes must be created on the primary site and the standby site. Please follow the documentation provided by your storage vendor to create the volumes. Based on the capabilities of the disk replication technology available with your preferred storage device, you may need to create mount points directories and symbolic links on each of the nodes within a tier. The mount points and symbolic links are set up so that the same directory structure can be used on each Application Server host within a tier. FOM in addition to using file system for storing configuration files uses file systems to persist operation data in various directories of file system. These directory structure must be maintained on both production and stand by sites.

Following is the list of directory used by various components of FOM and its property name. These directories must have identical directory structure on both production site and standby site and must replicated using storage-to-storage DR such as SRDF, SRDF/A.

**Table 2: Order Management System**

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.tibco.af.oms.router.recoveryFileFolderPath</td>
<td>Path to folder containing Router Recovery Files. Uses by router to recover from system crush.</td>
</tr>
<tr>
<td>com.tibco.fom.oms.afi.offline.product.master.directory</td>
<td>Offline product catalog directory to be used in master (engine startup requests) mode.</td>
</tr>
<tr>
<td>com.tibco.fom.oms.afi.offline.product.directory</td>
<td>Offline product catalog directory to be used in non-master (offline poller and web service request) mode.</td>
</tr>
<tr>
<td>com.tibco.fom.oms.afi.offline.customer.master.directory</td>
<td>Offline customer catalog directory to be used in master (engine startup requests) mode.</td>
</tr>
<tr>
<td>com.tibco.fom.oms.afi.offline.customer.directory</td>
<td>Offline customer catalog directory to be used in non-master (offline poller and web service request) mode.</td>
</tr>
<tr>
<td>com.tibco.fom.oms.afi.offline.segment.master.directory</td>
<td>Offline segment catalog directory to be used in master (engine startup requests) mode.</td>
</tr>
<tr>
<td>com.tibco.fom.oms.afi.offline.segment.directory</td>
<td>Offline segment catalog directory to be used in non-master (offline poller and web service request) mode.</td>
</tr>
<tr>
<td>com.tibco.fom.oms.afi.offline.planfragment.master.directory</td>
<td>Offline plan fragment catalog directory to be used in master (engine startup requests) mode.</td>
</tr>
<tr>
<td>com.tibco.fom.oms.afi.offline.planfragment.directory</td>
<td>Offline plan fragment catalog directory to be used in non-master (offline poller and web service request) mode.</td>
</tr>
<tr>
<td>com.tibco.fom.oms.afi.offline.action.master.directory</td>
<td>Offline action catalog directory to be used in master (engine startup requests) mode.</td>
</tr>
<tr>
<td>com.tibco.fom.oms.afi.offline.action.directory</td>
<td>Offline action catalog directory to be used in non-master (offline poller and web service request) mode.</td>
</tr>
</tbody>
</table>

You also have to periodically replicate FOM configuration files from production to DR site.
<table>
<thead>
<tr>
<th>Directory</th>
<th>Content</th>
<th>Replication Requirement</th>
</tr>
</thead>
</table>
| TIBCO_AF_HOME        | FOM Software                   | Replicate this directory whenever there has been a change to the deployed TIBCO runtime software. For example, when:  
• a software has been installed.  
• a hotfix, service pack or upgrade to an existing software component has been added. |
| EMS_HOME             | TIBCO EMS Software             | You can either install EMS on the DR site, or else replicate this directory. Replicate this directory when there has been a change to the EMS runtime software. |
| AF_CONFIG_HOME       | FOM Configuration Directory    | Replicate this directory whenever there has been a change to the configuration. |
Network

It is recommended that all the ports required by components of FOM configured with same port number. FOM supports distributing components across network. All the components excluding OMS UI, communicated with other component using TIBCO EMS. OMS UI communicates with OMS Server using HTTP. If layer 7 load balancer using within a site, OMS UI must always configured point to local site load balancer if even geo targeted global load balancer is used to dynamically update DNS entry and activate the standby site. You must use host names instead of IP address when specifying location of services. The IP address scheme at the recovery site could be different from primary site. If IP address is specified, there is a possibility that it might refer to node that is not available.

Following is the list of host and port number used by OMS UI and OMS Server components of FOM for HTTP communication and its property name:

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.tibco.af.omsServer.proxyHost</td>
<td>Host address of the OMS server</td>
</tr>
<tr>
<td>com.tibco.af.omsServer.proxyPort</td>
<td>Port number of the OMS server</td>
</tr>
</tbody>
</table>

If Fulfillment Provisioning (FP) is integrated with FOM, then following additional properties must point to local site load balancer:

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.tibco.af.fpServer.proxyHost</td>
<td>Host address of the FP server</td>
</tr>
<tr>
<td>com.tibco.af.fpServer.proxyPort</td>
<td>Port number of the FP server</td>
</tr>
</tbody>
</table>
Database

On both the primary site and the DR site, create a service name alias that FOM can use to identify the database to which it connects.

FOM uses a JDBC connect string to identify the Oracle database to which it connects. The connect string identifies the database by its service name which, by default, is the same as its SID. Because the databases on the primary and DR site have different SIDs, you must define an Oracle service name that FOM can use as an alias, allowing it to connect to the database whether it is currently running on the primary system or the DR system.

Please refer to the Oracle Data Guard for best practices of database disaster recovery.
Messaging Server

TIBCO EMS servers provide fault tolerant mechanism that allows you to configure two servers - one being the primary server and the other as standby server. The primary server accepts client connections or requests and interacts with clients to send or receive messages. If the primary server fails, the standby server takes over; it becomes the new active server and resumes the operation. It does not support more than two servers in this configuration. TIBCO EMS provides you options to configure shared state and unshared state failover. To avoid message loss, duplication or out of order message delivery, it is always preferred to use shared state failover.

Implementing on a Shared Disk File System

To prevent disk failures and outages, include the TIBCO EMS server logs, configuration files and file stores on a shared SAN to be accessed by multiple nodes. The shared disk subsystem should be accessible from standby servers to prevent disk failures or node outages from causing a prolonged JMS Server outage. This will allow you to restart the Messaging server from another node.
ActiveSpace

FOM components can use the TIBCO ActiveSpace as a Level 2 cache with hibernate. ActiveSpace data can be persisted in the database. Applying appropriate replication technology for the database will address the recovery of ActiveSpace data at DR site.
Chapter 6

FAQs and Troubleshooting

This topic lists some common errors along with their causes and solutions.

Why are the web services returning activity timed out error code?

The Product Model, Customer Model, and Segment Model are not loaded into AOPD and OCV applications. To resolve this issue, load the models into the application.

Where can I find the log files generated by TIBCO Fulfillment Order Management?

The location of the log files is $TIBCO_HOME/tra/domain/<af_domain>/application/logs.

Why is the subscriber ID not found?

If the Segment, Customer, and Product models are not loaded, the web service does not return the Subscriber ID even if the subscribers are loaded.

In the Fulfillment Order Management Rule Configuration, using Internet Explorer browser, what should I do if the selected value is not inserted where cursor is pointing?

If you try to enter a value by selecting the tree attribute for template builder, enter the value at the position where the cursor is pointing. Due to the restriction in the Internet Explorer browser, put a symbol in the editor, for example, ‘?’, select the symbol so that it is highlighted and then select the attribute value from attribute tree and remove the symbol.

What do I do if I see the following message? java.lang.RuntimeException: Sorting of Plan Item Milestones failed due to Invalid PlanFragment Sections!!

Make sure that the plan fragment model contain the sections between all possible combinations of milestone pairs that may appear in the plan item in the execution plan generated by AOPD. Refer the detailed requirement in the 'Changes in Functionality' section in Fulfillment Order Management release notes.
Appendix

A

Schema References

Topics

• Plan Item
• Result Status
• Message
• Order Request
• Order Request Header
• Order Request Line
• Process Component Model
• Database Schema
Plan Item

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Cardinality</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>planItem/planItemID</td>
<td>String</td>
<td>Required</td>
<td>Unique identifier for the plan item within the plan to be executed.</td>
</tr>
<tr>
<td>planItem/description</td>
<td>String</td>
<td>Optional</td>
<td>Description for the plan item to be executed.</td>
</tr>
<tr>
<td>planItem/processComponentID</td>
<td>String</td>
<td>Required</td>
<td>Unique identifier for the Process Component to be executed.</td>
</tr>
<tr>
<td>planItem/processComponentName</td>
<td>String</td>
<td>Required</td>
<td>Process component name for the Process Component to be executed.</td>
</tr>
<tr>
<td>planItem/processComponentVersion</td>
<td>String</td>
<td>Optional</td>
<td>Process component version for the Process Component to be executed.</td>
</tr>
<tr>
<td>planItem/processComponentType</td>
<td>String</td>
<td>Optional</td>
<td>Process component type for the Process Component to be executed.</td>
</tr>
<tr>
<td>planItem/processComponentRecordType</td>
<td>String</td>
<td>Optional</td>
<td>Class of processComponentType.</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>--------</td>
<td>----------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>planItem/orderLine</td>
<td>Type</td>
<td>1-M</td>
<td>Order line type for the plan item to be executed.</td>
</tr>
<tr>
<td>planItem/orderLine/orderLineNumber</td>
<td>String</td>
<td>Required</td>
<td>Order line number for the order line of the plan item to be executed.</td>
</tr>
<tr>
<td>planItem/orderLine/productID</td>
<td>String</td>
<td>Required</td>
<td>Product ID for the order line of the plan item to be executed.</td>
</tr>
<tr>
<td>planItem/orderLine/productVersion</td>
<td>String</td>
<td>Optional</td>
<td>Product version for the order line of the plan item to be executed.</td>
</tr>
<tr>
<td>planItem/orderLine/action</td>
<td>String</td>
<td>Required</td>
<td>Order line action for the order line of the plan item to be executed.</td>
</tr>
<tr>
<td>planItem/orderLine/actionMode</td>
<td>String</td>
<td>Optional</td>
<td>Order line action mode for the order line of the plan item to be executed.</td>
</tr>
<tr>
<td>planItem/orderLine/quantity</td>
<td>Long</td>
<td>Required</td>
<td>Quantity for the order line of the plan item to be executed.</td>
</tr>
<tr>
<td>planItem/orderLine/uom</td>
<td>String</td>
<td>Required</td>
<td>Unit of measure for the order line of the plan item to be executed.</td>
</tr>
<tr>
<td>planItem/orderLine/subscriberID</td>
<td>String</td>
<td>Optional</td>
<td>Subscriber ID for the order line of the plan item to be executed.</td>
</tr>
<tr>
<td>planItem/orderLine/linkID</td>
<td>String</td>
<td>Optional</td>
<td>Link ID for the order line of the plan item to be executed.</td>
</tr>
<tr>
<td>planItem/orderLine/inventoryID</td>
<td>String</td>
<td>Optional</td>
<td>Inventory ID for the order line of the plan item to be executed.</td>
</tr>
<tr>
<td>planItem/orderLine/eol</td>
<td>Boolean</td>
<td>Required</td>
<td>End of line flag for the order line of the plan item to be executed. This indicates that this plan item is the final plan item for the order line.</td>
</tr>
<tr>
<td>planItem/action</td>
<td>String</td>
<td>Required</td>
<td>Plan item action for the plan item to be executed.</td>
</tr>
<tr>
<td>planItem/actionMode</td>
<td>String</td>
<td>Optional</td>
<td>Plan item action mode for the plan item to be executed.</td>
</tr>
</tbody>
</table>
Result Status

Figure 35: Result Status

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Cardinality</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>deployment</td>
<td>String</td>
<td>Required</td>
<td>Engine deployment that returned this result.</td>
</tr>
<tr>
<td>service</td>
<td>String</td>
<td>Required</td>
<td>Service name that returned this result</td>
</tr>
<tr>
<td>operation</td>
<td>String</td>
<td>Required</td>
<td>Operation within the service that returned this result.</td>
</tr>
<tr>
<td>component</td>
<td>String</td>
<td>Optional</td>
<td>Component within the operation and service that returned this result.</td>
</tr>
<tr>
<td>severity</td>
<td>String</td>
<td>Required</td>
<td>Severity result. Valid values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1. S - Success</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. W - Warning</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3. E - Error</td>
</tr>
<tr>
<td>code</td>
<td>String</td>
<td>Required</td>
<td>Message code for this result.</td>
</tr>
<tr>
<td>message</td>
<td>String</td>
<td>Required</td>
<td>Message details for this result.</td>
</tr>
</tbody>
</table>
### Message

#### Figure 36: Message

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Cardinality</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lineNumber</td>
<td>String</td>
<td>Optional</td>
<td>Order line number that this message refers to.</td>
</tr>
<tr>
<td>type</td>
<td>String</td>
<td>Required</td>
<td>Message type. Valid values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1. Information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. Warning</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3. Error</td>
</tr>
<tr>
<td>Code</td>
<td>String</td>
<td>Required</td>
<td>Message code for this message.</td>
</tr>
<tr>
<td>Description</td>
<td>String</td>
<td>Required</td>
<td>Message text for this message.</td>
</tr>
<tr>
<td>udf</td>
<td>Type</td>
<td>0-M</td>
<td>UDF type.</td>
</tr>
<tr>
<td>udf/name</td>
<td>String</td>
<td>Required</td>
<td>User defined field name.</td>
</tr>
<tr>
<td>udf/value</td>
<td>String</td>
<td>Required</td>
<td>User defined field value.</td>
</tr>
</tbody>
</table>
## Order Request

### Figure 37: Order Request

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Cardinality</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>orderRef</td>
<td>String</td>
<td>Required</td>
<td>External unique identifier for an order.</td>
</tr>
<tr>
<td>header</td>
<td>Type</td>
<td>Required</td>
<td>Order request header type. Refer to the Order Request Header definition for details.</td>
</tr>
<tr>
<td>line</td>
<td>Type</td>
<td>1-M</td>
<td>Order request line type. Refer to the Order Request Line definition for details.</td>
</tr>
<tr>
<td>extension</td>
<td>Type</td>
<td>Optional</td>
<td>Extension attributes for user-defined fields.</td>
</tr>
<tr>
<td>extension/#any</td>
<td>Any</td>
<td>Required</td>
<td>Any data</td>
</tr>
</tbody>
</table>

---

TIBCO® Fulfillment Order Management Administration
Order Request Header

Figure 38: Order Request Header
<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Cardinality</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>String</td>
<td>Optional</td>
<td>Description for the order.</td>
</tr>
<tr>
<td>customerID</td>
<td>String</td>
<td>Required</td>
<td>Unique identifier for the customer for this order.</td>
</tr>
<tr>
<td>subscriberID</td>
<td>String</td>
<td>Required</td>
<td>Unique identifier for the subscriber for this order.</td>
</tr>
</tbody>
</table>

**Figure 39: Invoice Address**

**Figure 40: Delivery Address**
<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Optional/Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>requiredByDate</td>
<td>DateTime</td>
<td>Optional, Choice</td>
<td>Date and time when this order is required to start.</td>
</tr>
<tr>
<td>requiredOnDate</td>
<td>DateTime</td>
<td>Optional, Choice</td>
<td>Date and time by which this order is required to complete.</td>
</tr>
<tr>
<td>invoiceAddress</td>
<td>Type</td>
<td>Required</td>
<td>Invoice address type.</td>
</tr>
<tr>
<td>invoiceAddress/line1</td>
<td>String</td>
<td>Required</td>
<td>Invoice address line 1.</td>
</tr>
<tr>
<td>invoiceAddress/line2</td>
<td>String</td>
<td>Optional</td>
<td>Invoice address line 2.</td>
</tr>
<tr>
<td>invoiceAddress/line3</td>
<td>String</td>
<td>Optional</td>
<td>Invoice address line 3.</td>
</tr>
<tr>
<td>invoiceAddress/locality</td>
<td>String</td>
<td>Required</td>
<td>Invoice address locality.</td>
</tr>
<tr>
<td>invoiceAddress/region</td>
<td>String</td>
<td>Optional</td>
<td>Invoice address region.</td>
</tr>
<tr>
<td>invoiceAddress/country</td>
<td>String</td>
<td>Required</td>
<td>Invoice address country.</td>
</tr>
<tr>
<td>invoiceAddress/postcode</td>
<td>String</td>
<td>Required</td>
<td>Invoice address post code.</td>
</tr>
<tr>
<td>invoiceAddress/supplementaryLocation</td>
<td>String</td>
<td>Optional</td>
<td>Invoice address supplementary location.</td>
</tr>
<tr>
<td>deliveryAddress</td>
<td>Type</td>
<td>Required</td>
<td>Delivery address type.</td>
</tr>
<tr>
<td>deliveryAddress/line1</td>
<td>String</td>
<td>Required</td>
<td>Delivery address line 1.</td>
</tr>
<tr>
<td>deliveryAddress/line2</td>
<td>String</td>
<td>Optional</td>
<td>Delivery address line 2.</td>
</tr>
<tr>
<td>deliveryAddress/line3</td>
<td>String</td>
<td>Optional</td>
<td>Delivery address line 3.</td>
</tr>
<tr>
<td>deliveryAddress/locality</td>
<td>String</td>
<td>Required</td>
<td>Delivery address locality.</td>
</tr>
<tr>
<td>deliveryAddress/region</td>
<td>String</td>
<td>Optional</td>
<td>Delivery address region.</td>
</tr>
<tr>
<td>deliveryAddress/country</td>
<td>String</td>
<td>Required</td>
<td>Delivery address country.</td>
</tr>
<tr>
<td>deliveryAddress/postcode</td>
<td>String</td>
<td>Required</td>
<td>Delivery address post code.</td>
</tr>
<tr>
<td>deliveryAddress/supplementaryLocation</td>
<td>String</td>
<td>Optional</td>
<td>Delivery address supplementary location.</td>
</tr>
<tr>
<td>notes</td>
<td>String</td>
<td>Optional</td>
<td>Order notes.</td>
</tr>
<tr>
<td>slaID</td>
<td>String</td>
<td>0-M</td>
<td>Unique identifier for an SLA that is applied to this order.</td>
</tr>
<tr>
<td>udf</td>
<td>Type</td>
<td>0-M</td>
<td>UDF type.</td>
</tr>
<tr>
<td>udf/name</td>
<td>String</td>
<td>Required</td>
<td>User defined field name.</td>
</tr>
<tr>
<td>udf/value</td>
<td>String</td>
<td>Required</td>
<td>User defined field value.</td>
</tr>
<tr>
<td>udf/extension</td>
<td>Type</td>
<td>Optional</td>
<td>Extension attributes for user-defined fields.</td>
</tr>
<tr>
<td>udf/extension/#any</td>
<td>Any</td>
<td>Required</td>
<td>Any data</td>
</tr>
<tr>
<td>extension</td>
<td>Type</td>
<td>Optional</td>
<td>Extension attributes for user-defined fields.</td>
</tr>
<tr>
<td>extension/#any</td>
<td>Any</td>
<td>Required</td>
<td>Any data</td>
</tr>
</tbody>
</table>
Order Request Line

Figure 41: Order Request Line
Figure 42: Order Line Characteristics

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Cardinality</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lineNumber</td>
<td>String</td>
<td>Required</td>
<td>Unique identifier for this order line within this order.</td>
</tr>
<tr>
<td>subscriberID</td>
<td>String</td>
<td>Optional</td>
<td>Unique identifier for the subscriber for this order line.</td>
</tr>
<tr>
<td>productId</td>
<td>String</td>
<td>Required</td>
<td>Product identifier for this order line.</td>
</tr>
<tr>
<td>productName</td>
<td>String</td>
<td>Optional</td>
<td>Product version for the product for this order line.</td>
</tr>
<tr>
<td>quantity</td>
<td>Integer</td>
<td>Required</td>
<td>Quantity of the product being ordered.</td>
</tr>
<tr>
<td>uom</td>
<td>String</td>
<td>Required</td>
<td>Unit of measure of the product being ordered.</td>
</tr>
<tr>
<td>deliveryAddress</td>
<td>Type</td>
<td>Required</td>
<td>Delivery address type.</td>
</tr>
<tr>
<td>deliveryAddress/line1</td>
<td>String</td>
<td>Required</td>
<td>Delivery address line 1.</td>
</tr>
<tr>
<td>deliveryAddress/line2</td>
<td>String</td>
<td>Optional</td>
<td>Delivery address line 2.</td>
</tr>
<tr>
<td>deliveryAddress/line3</td>
<td>String</td>
<td>Optional</td>
<td>Delivery address line 3.</td>
</tr>
<tr>
<td>deliveryAddress/locality</td>
<td>String</td>
<td>Required</td>
<td>Delivery address locality.</td>
</tr>
<tr>
<td>deliveryAddress/region</td>
<td>String</td>
<td>Optional</td>
<td>Delivery address region.</td>
</tr>
<tr>
<td>deliveryAddress/country</td>
<td>String</td>
<td>Required</td>
<td>Delivery address country.</td>
</tr>
<tr>
<td>Field</td>
<td>Type</td>
<td>Required</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>------</td>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>deliveryAddress/postcode</td>
<td>String</td>
<td>Required</td>
<td>Delivery address post code.</td>
</tr>
<tr>
<td>deliveryAddress/supplementaryLocation</td>
<td>String</td>
<td>Optional</td>
<td>Delivery address supplementary location.</td>
</tr>
<tr>
<td>action</td>
<td>String</td>
<td>Required</td>
<td>Action for this order line. Valid values are: 1. Provide 2. Update 3. Cease 4. Cancel</td>
</tr>
<tr>
<td>actionMode</td>
<td>String</td>
<td>Optional</td>
<td>Supplementary action mode for the action.</td>
</tr>
<tr>
<td>requiredByDate</td>
<td>DateTime</td>
<td>Optional, Choice</td>
<td>Date and time by which this order line is required to start.</td>
</tr>
<tr>
<td>requiredOnDate</td>
<td>DateTime</td>
<td>Optional, Choice</td>
<td>Date and time by which this order line is required to complete.</td>
</tr>
<tr>
<td>linkID</td>
<td>String</td>
<td>Optional</td>
<td>Unique identifier used to link across order lines on this order.</td>
</tr>
<tr>
<td>inventoryID</td>
<td>String</td>
<td>Optional</td>
<td>Unique identifier that identifies this order line product with an entry in an external inventory management system.</td>
</tr>
<tr>
<td>notes</td>
<td>String</td>
<td>Optional</td>
<td>Order line notes.</td>
</tr>
<tr>
<td>slaID</td>
<td>String</td>
<td>0-M</td>
<td>Unique identifier for an SLA that is applied to this order line.</td>
</tr>
<tr>
<td>udf</td>
<td>Type</td>
<td>0-M</td>
<td>UDF type.</td>
</tr>
<tr>
<td>udf/name</td>
<td>String</td>
<td>Required</td>
<td>User defined field name.</td>
</tr>
<tr>
<td>udf/value</td>
<td>String</td>
<td>Required</td>
<td>User defined field value.</td>
</tr>
<tr>
<td>udf/extension</td>
<td>Type</td>
<td>Optional</td>
<td>Extension attributes for user-defined fields.</td>
</tr>
<tr>
<td>udf/extension/#any</td>
<td>Any</td>
<td>Required</td>
<td>Any data</td>
</tr>
<tr>
<td>characteristic</td>
<td>Type</td>
<td>Required</td>
<td>Characteristic type.</td>
</tr>
<tr>
<td>characteristic/name</td>
<td>String</td>
<td>Required</td>
<td>Characteristic name.</td>
</tr>
<tr>
<td>characteristic/description</td>
<td>String</td>
<td>Required</td>
<td>Characteristic description.</td>
</tr>
<tr>
<td>characteristic/value</td>
<td>Type</td>
<td>0-M</td>
<td>Characteristic value type.</td>
</tr>
<tr>
<td>characteristic/value/name</td>
<td>String</td>
<td>Required</td>
<td>Characteristic value name.</td>
</tr>
<tr>
<td>characteristic/value/type</td>
<td>String</td>
<td>Required</td>
<td>Characteristic value type.</td>
</tr>
<tr>
<td>characteristic/value/value</td>
<td>String</td>
<td>Optional</td>
<td>Characteristic value value.</td>
</tr>
<tr>
<td>Characteristic/extension/valueFrom</td>
<td>String</td>
<td>Optional</td>
<td>Characteristic value value from.</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>--------</td>
<td>----------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>Characteristic/extension/valueTo</td>
<td>String</td>
<td>Optional</td>
<td>Characteristic value value to.</td>
</tr>
<tr>
<td>Characteristic/extension/extension</td>
<td>Type</td>
<td>Optional</td>
<td>Extension attributes for user-defined fields.</td>
</tr>
<tr>
<td>Characteristic/extension/extension/#any</td>
<td>Any</td>
<td>Required</td>
<td>Any data</td>
</tr>
<tr>
<td>Characteristic/extension/extension/#any</td>
<td>Any</td>
<td>Required</td>
<td>Any data</td>
</tr>
<tr>
<td>customerItemID</td>
<td>String</td>
<td>Optional</td>
<td>Customer item unique identifier.</td>
</tr>
<tr>
<td>extension</td>
<td>Type</td>
<td>Optional</td>
<td>Extension attributes for user-defined fields.</td>
</tr>
<tr>
<td>extension/#any</td>
<td>Any</td>
<td>Required</td>
<td>Any data</td>
</tr>
</tbody>
</table>
Figure 43: Process Component Model

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Cardinality</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>processComponentID</td>
<td>String</td>
<td>Required</td>
<td>Unique identifier for the Process Component to be executed.</td>
</tr>
<tr>
<td>processComponentName</td>
<td>String</td>
<td>Optional</td>
<td>Process component name for the Process Component to be executed.</td>
</tr>
<tr>
<td>processComponentVersion</td>
<td>String</td>
<td>Optional</td>
<td>Process component version for the Process Component to be executed.</td>
</tr>
<tr>
<td>processComponentType</td>
<td>String</td>
<td>Optional</td>
<td>Process component type for the Process Component to be executed.</td>
</tr>
<tr>
<td>processComponentRecordType</td>
<td>String</td>
<td>Optional</td>
<td>Class of processComponentType.</td>
</tr>
<tr>
<td>errorHandler</td>
<td>String</td>
<td>Optional</td>
<td>Error handler to use in the event of the Process Component returning an incomplete or unsuccessful execution response message.</td>
</tr>
<tr>
<td>retry</td>
<td>Type</td>
<td>Optional</td>
<td>Retry type.</td>
</tr>
<tr>
<td>retry/retryFailed</td>
<td>Boolean</td>
<td>Required</td>
<td>Flag indicating that Orchestrator should retry failed plan items.</td>
</tr>
<tr>
<td>Field</td>
<td>Type</td>
<td>Required</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>retry/retryCount</td>
<td>Integer</td>
<td>Required</td>
<td>Number of times Orchestrator should retry the failed plan item before referring it to Plan Item Failed Handler for manual intervention.</td>
</tr>
<tr>
<td>retry/retryDelay</td>
<td>Long</td>
<td>Required</td>
<td>Delay in msec between calls to the Process Component in the event that the plan item is retried.</td>
</tr>
<tr>
<td>section</td>
<td>Type</td>
<td>1-M</td>
<td>Process component model section type.</td>
</tr>
<tr>
<td>section/startMilestoneID</td>
<td>String</td>
<td>Required</td>
<td>Unique identifier for the start milestone that describes this section.</td>
</tr>
<tr>
<td>section/endMilestoneID</td>
<td>String</td>
<td>Required</td>
<td>Unique identifier for the end milestone that describes this section.</td>
</tr>
<tr>
<td>section/typicalDuration</td>
<td>Long</td>
<td>Required</td>
<td>Typical duration for this section in msec.</td>
</tr>
<tr>
<td>section/maximumDuration</td>
<td>Long</td>
<td>Required</td>
<td>Maximum duration for this section in msec.</td>
</tr>
</tbody>
</table>
The database schema introduces two new tables in the co-located mode architecture:

1. **TIME_WINDOW**
2. **DAMPENING_CRITERIA**

**Figure 44: TIME_WINDOW**

<table>
<thead>
<tr>
<th>Element</th>
<th>Data Type</th>
<th>Cardinality</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TimeWindowSID</td>
<td>String</td>
<td>Required</td>
<td></td>
</tr>
<tr>
<td>ProcessComponentID</td>
<td>String</td>
<td>Required</td>
<td>Unique identifier for the Process Component to be executed.</td>
</tr>
<tr>
<td>TimeWindowID</td>
<td>Number</td>
<td>Required</td>
<td></td>
</tr>
<tr>
<td>OrderID</td>
<td>String</td>
<td>Required</td>
<td>Internal unique identifier for the order associated with the plan containing the plan item with the milestone to be released.</td>
</tr>
<tr>
<td>StartMilestoneID</td>
<td>String</td>
<td>Required</td>
<td>Unique identifier for the start milestone that describes the section.</td>
</tr>
<tr>
<td>EndMilestoneID</td>
<td>String</td>
<td>Required</td>
<td>Unique identifier for the end milestone that describes the section.</td>
</tr>
<tr>
<td>PlanID</td>
<td>String</td>
<td>Required</td>
<td>Internal unique identifier for the plan that contains the plan item to execute.</td>
</tr>
<tr>
<td>PlanItemID</td>
<td>String</td>
<td>Required</td>
<td>Unique identifier for the plan item that contains the milestone to be released.</td>
</tr>
</tbody>
</table>

**Figure 45: DAMPENING_CRITERIA**

<table>
<thead>
<tr>
<th>Element</th>
<th>Data Type</th>
<th>Cardinality</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RuleID</td>
<td>String</td>
<td>Required</td>
<td>-</td>
</tr>
<tr>
<td>Value</td>
<td>Blob</td>
<td>Optional</td>
<td>Unique identifier to store dampening criteria</td>
</tr>
<tr>
<td>Element</td>
<td>Data Type</td>
<td>Cardinality</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-----------</td>
<td>-------------</td>
<td>---------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>information for a jeopardy rule.</td>
</tr>
</tbody>
</table>
Glossary

AOPD

Automatic Order Plan Development (AOPD) is a component that generates execution plan based on product model and submitted order(s).

Accessed through a JMS event interface.

Cache

Cache saves order data in a persistent data store, as well as stores transient order data throughout the order lifecycle. This component serves order and plan data to client Process Components when requested, and updates plan data.

Accessed through a JMS event interface.

Feasibility Provider

Feasibility Provider (FP) is a component, which analyzes the order to determine if it can be fulfilled. Feasibility checking is an optional step in the order lifecycle and it may involve validating the order containing the required products, physical network capacity checking, or inventory stock level check. The Feasibility Provider is a customer-implemented component because feasibility checking is highly customized to the requirements of a customer.

Accessed through a JMS event interface.

Product Catalog

The Product Catalog component stores the product data model which details relationships between products. It also links products to the associated fulfillment Process Components. The Fulfillment Order Management AOPD system accesses the Product Catalog through JMS. The Plan Fragment Model definitions repositories describing the Process Components are contained within the Product Catalog.

Order

An order is a set of items that a customer requests to be fulfilled. Orders consist of a series of order lines, with each line corresponding to a requested product. Order lines are an abstraction of the work that will be done as part of the plan, with order lines broadly mapping onto plan items.

Plan

A plan is a representation of the tasks to be completed to reach the Fulfillment goal.

Plan Item

Plan item is a set of work that must be performed to fulfill an order. Like an order consists of order lines, a plan consists of plan items.

Plan Fragment

A plan fragment is an abstraction of a Process Component that contains configuration information that Orchestrator requires in order to handle errors and SLA notifications. Plan fragments are optional.

Product Affinity

Product Affinity allows different plan fragment types to be grouped together on the same order. For instance, when two plan items in an execution plan have an Affinity to each other, these two items are grouped together and are executed as a single task during provisioning. Product Affinity is specified in the product catalog.